I Jornada Latino-Americana de Atualização em Informática

André Luiz Satoshi Kawamoto
Ana Grasielle Dionísio Corrêa
Valéria Farinazzo Martins

Organizadores

André Luiz Satoshi Kawamoto

Ana Grasielle Dionísio Corrêa

Valéria Farinazzo Martins

I Jornada Latino-Americana de Atualização em Informática

ANAIS

1ª edição

São Paulo

Sociedade Brasileira de Computação - SBC

2018

J82

JOLAI – I Jornada latino-americana de atualização em informática, 01 a 05 de outubro de 2018 [livro eletrônico] / Organizadores André Luiz Satoshi Kawamoto, Ana Grasielle Dionisio Correa, Valéria Farinazzo Martins – Porto Alegre: Sociedade Brasileira de Computação, 2018.
16.74 MB ; PDF.

Anais do JOLAI – I Jornada latino-americana de atualização em informática.
ISBN: 978-85-7669-458-8.
Livro eletrônico.

1. Ciência da computação. 2. Sistemas computacionais. 3. Sistemas de informação. 4. Educação em computação. 5. Infraestrutura de rede. 6. Inteligência artificial. 7. Processamento de imagem. 8. Interação humano-computador. 9. Gestão de dados. 10. Informação I. Kawamoto, André Luiz Satoshi. II. Correa, Ana Grasielle Dionisio. III. Martins, Valéria Farinazzo. IV.Título.

CDD 004

Bibliotecária Responsável: Marta Luciane Toyoda – CRB 8/ 8234

Apoio:


Mackenzie


UFABC


UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ


CLEI
Centro Latinoamericano de
Estudios en Informática


LACLO

Comitê Científico

Adriana Porto Proença – Universidade Federal de Uberlândia (Brasil)

Arlino Henrique Magalhães de Araújo – Universidade Federal do Piauí (Brasil)

Bruno Luis Soares de Lima – Universidade Presbiteriana Mackenzie (Brasil)

Camilo de Lellis Barreto Junior – Universidade Federal de Uberlândia (Brasil)

Cesar Collazos – Universidad del Cauca (Colômbia)

Charles Rodamilans – Universidade Presbiteriana Mackenzie (Brasil)

Daniela Cunha – Universidade Presbiteriana Mackenzie (Brasil)

Daniela Marques – Instituto Federal de São Paulo (Brasil)

Eurico Luiz Prospero Ruivo – Universidade Presbiteriana Mackenzie (Brasil)

Flávio Eduardo Aoki Horita – Universidade Federal do ABC (Brasil)

Gilda Aparecida de Assis – Universidade Federal de Ouro Preto (Brasil)

Heitor Augustus Xavier Costa – Universidade Federal de Lavras (Brasil)

Ismar Frango Silveira – Universidade Presbiteriana Mackenzie (Brasil)

Israel Florentino – Universidade Presbiteriana Mackenzie (Brasil)

Ivanilton Polato – Universidade Tecnológica Federal do Paraná (Brasil)

João Soares de Oliveira Neto – Universidade Federal do Recôncavo da Bahia (Brasil)

Edgard Lamounier Junior – Universidade Federal de Uberlândia (Brasil)

Leandro Augusto da Silva – Universidade Presbiteriana Mackenzie (Brasil)

Leandro Pupo Natale – Universidade Presbiteriana Mackenzie (Brasil)

Lucio Geronimo Valentin – Universidade Tecnológica Federal do Paraná (Brasil)

Marcelo de Paiva Guimarães – Universidade Federal de São Paulo (Brasil)

Maria Amélia Eliseo – Universidade Presbiteriana Mackenzie (Brasil)

Mario Olímpio de Menezes – Universidade Presbiteriana Mackenzie (Brasil)

Mauricio Marengoni – Universidade Presbiteriana Mackenzie (Brasil)

Orlando Bisacchi Coelho – Universidade Presbiteriana Mackenzie (Brasil)

Rodrigo Campiolo – Universidade Tecnológica Federal do Paraná (Brasil)

Rogério de Leon Pereira – University of Manitoba (Canadá)

Rogério Aparecido Gonçalves – Universidade Tecnológica Federal do Paraná (Brasil)

Thiago Schumacher Barcelos – Instituto Federal de São Paulo (Brasil)

Vivaldo Breternitz – Universidade Presbiteriana Mackenzie (Brasil)

Apresentação

A Jornada Latino-Americana de Atualização em Informática (JoLAI) compreende trabalhos de pesquisadores sêniores da nossa comunidade, oferecendo uma oportunidade única para acadêmicos e profissionais atualizarem-se em temas diversos, interagindo com líderes das mais diversas áreas de pesquisa no Brasil e da América Latina.

Nessa primeira edição, a JoLAI integra o conjunto de eventos-satélite oferecidos no escopo dos XLIV Conferência Latino-americana de Informática (CLEI 2018) e XIII Conferência Latino-americana de Tecnologias de Aprendizagem (LACLO 2018), sediados pela Universidade Presbiteriana Mackenzie, em São Paulo-SP, Brasil.

Em primeiro lugar, gostaríamos de agradecer imensamente aos autores que acreditaram nessa proposta e submeteram seus textos, bem como a todos os membros do Comitê Científico, os quais contribuíram para a seleção de um conjunto de trabalhos que abordam um leque bastante diversificado de temas com ampla possibilidade de contribuição em soluções de alto impacto social, didático e de pesquisa. Agradecemos, ainda, a atenção e o apoio da organização geral do CLEI 2018 e do LACLO 2018, bem como à Universidade Presbiteriana Mackenzie (UPM), Universidade Federal do ABC (UFABC), Instituto Tecnológico da Aeronáutica (ITA) e à diretoria da Sociedade Brasileira de Computação (SBC), que propiciaram todas as condições para a realização desta 1a. edição da JoLAI.

É importante destacar que a JoLAI conta com a participação tanto de pesquisadores renomados (nacional e internacionalmente) quanto de talentos emergentes (pesquisadores e alunos) de toda a comunidade do CLEI e do LACLO, propiciando, assim, um ambiente altamente propício para a troca de ideias e a disseminação do conhecimento.

Os textos selecionados abordam variados temas relevantes para a Computação e suas aplicações, além de contribuir para a formação de alunos e pesquisadores. Os temas incluem Big Data, Análise de Dados, Desenvolvimento de Aplicações utilizando diferentes tecnologias, Tecnologias Assistivas, entre tantos outros, que se caracterizam pela atualidade e relevância para o bom desempenho acadêmico e profissional.

Desejamos a todos os uma excelente Jornada Latino-Americana de Atualização de Informática e reiteramos nossos imensos agradecimentos à comunidade.

Muito Obrigado!

André Luiz Satoshi Kawamoto
Ana Grasielle Dionísio Corrêa
Valéria Farinazzo Martins

Sumário

1. Capítulo 1

Autor:

Vivaldo José Breternitz
Universidade Presbiteriana Mackenzie
email: vivaldojose.breternitz@mackenzie.br

Capítulo 1

Enterprise Resource Planning (ERP): visão geral de uma área com muitas oportunidades

Vivaldo José Breternitz – vjbreternitz@mackenzie.br

Abstract

Enterprise Resource Planning (ERP), in Portuguese known as Sistemas Integrados de Gestão, is a computer application that claim to be the backbone of Information Technology in organizations, is gaining increasing importance in the current scenario, to the point that some consider them essential for the survival of companies. Given this scenario, it was decided to develop this work, which aims to generate reflections that produce knowledge that may contribute to the proper administration of the process of implementing ERP systems in the organizational environment. These reflections center on the processes of selection and implementation of these applications, alerting those involved in these processes about the critical factors to their success.

Resumo

Os Sistemas Integrados de Gestão, mais conhecidos como Enterprise Resource Planing (ERP), são aplicativos de computador que pretendem ser a espinha dorsal de Tecnologia da Informação nas organizações, vem ganhando cada vez mais importância no cenário atual, a ponto de alguns considerá-los essenciais para a sobrevivência das empresas. Dado esse cenário, decidiu-se desenvolver este trabalho, que tem como objetivo gerar reflexões que produzam conhecimentos que possam contribuir para a adequada administração do processo de implementação de ferramentas ERP no ambiente organizacional. Essas reflexões centram-se nos processos de seleção e implementação desses aplicativos, alertando os envolvidos nesses processos acerca dos fatores críticos para o sucesso dos mesmos.

1. Introdução

Enterprise Resource Planning (ERP) pode ser definido de diversas maneiras, dependendo de como se posiciona o estudioso do assunto: como uma solução de sistemas de informação para toda a empresa [Lieber 1995]; como uma arquitetura de *software* que facilita o fluxo de informações entre todas as áreas de uma companhia, como por exemplo, manufatura, logística, finanças, recursos humanos, etc. [Hicks 1997]; como uma solução de software que integra as várias esferas de uma organização [Chou 2018] etc.

Do ponto de vista de um profissional de Tecnologia da Informação (TI), uma boa definição talvez fosse "tecnologia capaz de organizar e integrar as informações armazenadas nos computadores de uma organização, de forma a eliminar dados redundantes ou desnecessários, racionalizar processos e distribuir a informação *on line* pelas várias áreas da mesma, de forma estruturada e aceita como fidedigna por todas elas. Pode ser entendido como a espinha dorsal (*backbone*) de TI na empresa, dentro da filosofia de centralizar a complexidade e distribuir a informação".

De um ângulo mais funcional, idealmente seria um sistema que captura uma dada informação uma única vez e a partir dela dispara uma série de operações na empresa e as rotinas de computador a elas vinculadas. O exemplo clássico seria o do representante de vendas que emite um pedido cujo registro aciona os sistemas de suprimentos, de fabricação, entrega, faturamento, custos, etc., permitindo que as informações pertinentes sejam acompanhadas em tempo real, de forma sintética e/ou analítica pela empresa e por seus parceiros de negócios, armazenando dados para consultas futuras.

Por qualquer ângulo que se defina ERP, não se pode deixar de considerar sua extrema importância no ambiente empresarial atual, importância essa que pode ser avaliada pelas palavras de Lemonakis (2018), afirmando que a implementação de ERP é essencial para a sobrevivência de uma empresa.

2. Objetivos e Aspectos Metodológicos

Dado esse cenário, decidiu-se desenvolver este trabalho, que tem como objetivo gerar reflexões que produzam conhecimentos que venham a contribuir para a adequada administração do processo de implementação de ferramentas ERP no ambiente organizacional, centrando-se nos processos de seleção e colocação em operação desses aplicativos, de forma a alertar os envolvidos nesses processos acerca dos fatores críticos para o sucesso dos mesmos.

O artigo é baseado na experiência do autor na utilização de ERP no meio empresarial e em sua vivência docente em cursos da área, a nível de graduação e de pós-graduação (estrito e lato sensos); assim, pode ser considerado um ensaio, que Ortega e Gasset (2004) definem como "ciência sem prova explicita", qualificando-o como um texto literário breve, que expõe ideias, críticas e reflexões a respeito de um dado tema, defendendo um ponto de vista pessoal e subjetivo sobre o mesmo sem se pautar por formalidades como documentos e provas empíricas ou dedutivas de caráter científico.

Meneghetti (2011) diz que os ensaios são uma forma de produção científica que valoriza aspectos relacionados às mudanças qualitativas que ocorrem nos objetos ou fenômenos analisados. Paviani (2009) complementa dizendo que "...afinal, o ensaio, mesmo o quando expõe uma teoria, nunca o faz de maneira doutrinal e dogmática. Ele não tem a pretensão de oferecer conteúdo acabado. Limita-se a coordenar ideias, pontos

de vistas. Mas, sendo livre, cultiva o rigor, coincidindo ou não sua forma com a exposição filosófica ou com a expressão literária, e, ainda, com a escrita científica".

Ressalte-se a importância da palavra "contribuir" quando se explicitou o objetivo deste trabalho; ao se falar em administração é difícil que a pesquisa leve à construção de modelos que possam ser aplicados sempre com segurança; Mattos (2009) diz ser ainda objeto de discussão a questão apresentada há décadas por Koontz e O'Donnell (1986): seria a administração ciência ou arte? A resposta cabal a essa questão (se existir) determinaria os métodos e técnicas aplicáveis à pesquisa na área; por ora, adota-se a postura de Ferreira *et al.* (1997), que afirmam ser a administração um amalgama de arte e ciência, seguindo essencialmente uma série de conceitos ideológicos e adotando modelos que parecem funcionar e que aos poucos poderão vir a constituir teoria.

3. A Evolução dos ERP

Até os anos 1960, a indústria manufatureira utilizava pouca coisa mais sofisticada que técnicas como as de EOQ (Economic Order Quantity, lançada em 1913, e que foi uma das primeiras aplicações de técnicas de modelagem matemática ao que então se chamava "Administração Científica"), de forma a que cada item dos estoques fosse analisado em termos de custo e consumo, procurando estabelecer lotes econômicos para compra e fabricação. Enfim, se administrava estoques, quase sempre de forma reativa, e praticamente mais nada se fazia na área.

Do ponto de vista de TI, cabe lembrar que nos anos 1960 o foco dos sistemas de computador voltados às áreas industriais ainda estava no controle de inventário. A maioria dos pacotes de *software* disponíveis na época, normalmente bastante customizados para atender a uma dada empresa, fora projetada para trabalhar com base nos conceitos tradicionais de controle de inventário, como os EOQ, e com a automatização do tratamento das listas de materiais componentes dos produtos, o BOM (Bill of Materials).

Nessa época, como relatam Slack *et al*. (2015) começou a ser utilizada a técnica denominada "Material Requirements Planning" (MRP), que podia ser vista como uma forma proativa de administração. A ideia básica era a de construir-se sistemas mais abrangentes, já procurando controlar os processos de compra e armazenagem de matérias primas e componentes em função do processo de produção.

A lógica dos MRP era simples, porém de ponto de vista de informática, de implementação bastante complexa; como os volumes de dados e de trabalho envolvidos eram bastante grandes, tornava-se praticamente impossível fazer-se qualquer coisa manualmente, sendo por isso obrigatório o uso de computadores. MRP foi um sucesso por ter permitido diminuição de estoques, de atrasos, etc., enfim, por ter permitido que se aumentasse a produtividade na área industrial.

Nos anos 70, MRP teve sua lógica mudada, pela inclusão do conceito "Closed Loop MRP", que basicamente provia *feedback* aos sistemas em termos de alerta quanto à efetiva capacidade da organização como um todo produzir um dado volume de um produto. Nessa década, o aumento da capacidade de processamento dos computadores disponíveis aumentou, o que permitiu que esses sistemas se tornassem mais populares, se bem que ainda praticamente confinados à indústria manufatureira - eram dotados de uma arquitetura fechada, rodando em ambiente *batch* e "não amigáveis" (*user-hostile*). Os mais conhecidos à época eram o DBS, Cullinet, McCormack and Dodge, SAP R/2 (este

um pouco mais recente) e Mapics, que tendo sido lançado pela IBM nos anos 1970, ainda é utilizado por algumas empresas.

Nos anos 1980, o conceito foi novamente ampliado, surgindo então o MRP-II (Manufacturing Resources Planning), que segundo Slack *et al*. (2015) ia além dos materiais propriamente ditos, atuando fortemente na produção, no controle de chão de fábrica e gerenciamento de distribuição, nos casos em que essa última função fosse relevante para a empresa. Wight (1981) foi o introdutor dessa denominação, que propôs de forma a tornar marcante o envolvimento de uma parcela maior da organização no processo.

Nesse período, as indústrias passaram também a utilizar recursos como CAD (Computer Aided Design) e CAM (Computer Aided Manufacturing), tendo surgido nessa época as primeiras iniciativas no sentido de integra-los ao MRP-II.

Nos anos 90, como relatam Martins e Laugeni (2010), o conceito foi novamente ampliado, desta vez procurando integrar as áreas de engenharia, finanças, recursos humanos, administração de projetos etc. Idealmente, a série completa de atividades dentro de qualquer organização empresarial. Assim, a primeira letra da sigla, que foi um "M" para "Material" e "Manufacturing", foi substituída por um "E", de "Enterprise", dada a pretensão desses sistemas de cobrir todas as áreas de uma empresa. Essa denominação foi criada pelo Gartner Group; evidentemente é possível discutir-se a adequação dessa denominação, o que foge ao escopo deste trabalho.

Na virada do século, a Internet e a tecnologia a ela associada geraram uma revolução nos sistemas ERP, tornando-os mais facilmente conectáveis com o ambiente externo à organização, interligando-a com clientes, fornecedores etc. e incorporando também aplicações como Business Inteligence, Big Data Analytics e na atualidade, buscando integrar-se ao processo conhecido como 4ª Revolução Industrial, permitindo a viabilização da Internet of Things; o acesso aos ERP através de *smartphones* e *tablets* também se tornou uma funcionalidade comum na atualidade. Nota-se também uma migração dos ERP para a modalidade Software as a Service, em que as empresas usuárias fazem uso do sistema através de “assinaturas”, sem a necessidade de aquisição de licenças de uso ou utilização de *hardware* próprio. Algumas organizações não vão até esse ponto, mas utilizam *cloud computing* para processamento e/ou armazenagem de dados [Zhao e Kirche 2013].

Além disso, observa-se queda no montante do investimento necessário à sua implantação, graças ao aumento da concorrência entre fornecedores e ao fato de boa parte das grandes organizações já terem a ferramenta implementada, forçando os fornecedores a buscarem novos clientes entre as pequenas e médias empresas. Apesar dessa queda, os investimentos necessários ainda são consideráveis.

4. Benefícios e Riscos Trazidos às Organizações pela Utilização de ERP

A ausência de ERP leva ao uso de sistemas que operam de forma independente, às vezes trocando informações através de estruturas improvisadas ou então totalmente isolados, gerando com frequência redundâncias indesejadas, retrabalho, inconsistências, dificuldades para acesso a informações consolidadas etc.

A utilização de ERP permite que diferentes áreas acessem uma única base de dados (ao menos do ponto de vista lógico) e em tempo real, eliminando os problemas

citados; além disso, tecnologias como acesso via Internet/Intranets e *smartphones* podem ser disponibilizados de maneira mais fácil, trazendo benefícios adicionais a todos aqueles que interagem com a instituição: clientes, funcionários e até mesmo o público externo.

Diversos autores, como Sabau *et al* (2009) e Breternitz e Galhardi (2011) afirmam que a utilização de ERP na gestão das organizações gera diversos benefícios, dentre os quais:

- Aumenta a capacidade de planejamento e decisão em nível estratégico, especialmente por darem suporte à análise de dados sofisticada, com a utilização de ferramentas como Business Intelligence e Big Data/Analytics, por exemplo.
- Permite que diferentes unidades da organização compartilhem os mesmos dados, induzindo-as a cooperarem entre si.
- Melhora das comunicações no seio da organização, inclusive ao consolidar a terminologia utilizada.
- Estimula e facilita a reengenharia de processos, padronizando-os e baseando-os em melhores práticas.
- Torna o gerenciamento de Tecnologia da Informação mais simples.
- Elimina redundâncias em termos de dados e processos.
- Facilita o uso de ferramentas de *workflow*.
- Facilita operações conjuntas com parceiros que usam ferramentas similares;
- Atende a requerimentos globais (os ERP usualmente são multilíngues, multimoedas, multipaís, etc.).
- Reduz os processos manuais, também por permitir que dados sejam introduzidos no sistema uma só vez.
- Facilita o trabalho e o treinamento dos usuários, pois diferentes aplicações utilizem interfaces similares.
- Facilita a implantação de autosserviço para toda a comunidade organizacional.
- Diminui os períodos em que os sistemas permanecem "fora do ar".
- Aumenta o nível de integridade dos dados.
- Aumenta o nível de segurança dos aplicativos e dados.
- Permite o acesso aos dados em tempo real.

Evidentemente, a esses benefícios se contrapõem alguns riscos e dificuldades, dentre os quais se destacam [Breternitz e Galhardi, 2011]:

- A necessidade de alteração dos processos funcionais (adaptação do sistema aos processos da empresa *versus* adaptação da empresa aos processos do sistema); apesar dos fornecedores negarem, esses sistemas não são muito flexíveis, sendo sua customização, quando necessária, muito cara, como dizem Cyrus *et al* (2018).
- A complexidade da customização (modificação do sistema para que esse possa se adequar a determinadas regras de negócio impossíveis de serem atendidas por parâmetros já existentes) – essas dificuldades acabam frequentemente sendo contornadas pela adoção de "sistemas paralelos", aplicações "não oficiais", geralmente utilizando ferramentas como Excel e SQL Server que substituem algumas funcionalidades do ERP, mas trazendo

riscos à organização. Situações como essas acabam constituindo o que se chama “Shadow IT”;
- A impossibilidade de atendimento às necessidades muito específicas;
- Os impactos sobre os recursos humanos, gerando resistências;
- A dependência da organização em relação ao fornecedor da ferramenta;
- A necessidade de altos investimentos - licenças de uso de *software*, consultoria para seleção do aplicativo e implantação, treinamento, atualização de *hardware* e outros.
- Expectativas não atingidas de forma plena: a fonte anteriormente citada afirma, considerando o mesmo período de cinco anos, que 53% das organizações obtiveram menos de 50% dos benefícios mensuráveis esperados.

5. A seleção da Ferramenta ERP

A escolha do ERP mais adequado a uma dada organização é o fator mais importante para que haja sucesso na sua implementação. Ela tipicamente acontece em uma atmosfera repleta de expectativas exageradas pelos vendedores, atribulada por problemas políticos internos e, quase sempre, em meio a uma grande crise que se espera ERP possa ajudar a solucionar.

Além disso, os custos e prazos envolvidos são muito grandes: em Panorama (2018) tem se a informação de que, no mercado norte-americano esses custos foram em média de US$ 3,8 milhões e os prazos cerca de 21 meses, tendo esses custos e prazos superado os previstos em 57%. É lícito acreditar que para o mercado brasileiro esses números não sejam muito diferentes. Fracassos em projetos desse porte podem gerar consequências catastróficas para as empresas, como nos casos da FoxMeyer Drugs [Scott 1999] e Hewlett Packard [Chaturvedi 2005], por exemplo.

Os processos de reengenharia normalmente desenvolvidos em paralelo com a seleção e posterior implantação desses sistemas também tornam o assunto digno de maiores cuidados. Nesse ponto cabe colocar que autores que tratam reengenharia e ERP conjuntamente recomendam que este deve ser precedido por aquela - nada impedindo, e até propondo que continuem em paralelo [Curran 1998].

Se antigamente a implantação de um sistema MRP envolvia no máximo os responsáveis pelas áreas de produção e TI, agora uma implantação de ERP precisa envolver também os mais altos executivos da empresa, o que sinaliza a importância do assunto. Apesar dessa tendência, muitas empresas ainda simplesmente dizem às suas áreas de TI: "encontrem a solução ERP que resolva todos os nossos problemas", diminuindo ou eliminando a participação e consequente responsabilidade dos usuários [Farley 1998]. Cientes dessa realidade, há fornecedores de ERP que concentram seus esforços de *marketing* nos profissionais de TI, enfatizando as características de seu produto nessa área.

As empresas são diferentes e, portanto, uma solução qualquer de prateleira dificilmente se encaixará muito bem neste contexto. Processos diferentes demandam soluções e abordagens diferentes. Um problema na indústria petroquímica normalmente não tem a mesma dimensão que teria na indústria de construção civil. O exemplo típico seria controle de processos contínuos *versus* controle de mão de obra.

Em outro exemplo, analisando uma variável apenas, que seria a interface da produção com vendas, encontramos duas alternativas básicas: fabricação compondo estoques (fabrica-se para depois vender) e fabricação sob encomenda (vende-se para então comprar). No primeiro caso isto significa vendas unitárias de pouco valor para um universo grande de clientes. No segundo, vende-se poucas unidades, mas de valor significativo, para um número pequeno de clientes; dificilmente uma mesma solução atenderia bem empresas diferentes.

O resultado destas diferenças reflete-se nos critérios de análise das soluções que estão sendo consideradas. Por exemplo, na indústria que trabalha com estoques, é altamente necessário que o *software* seja muito eficiente na gestão da cadeia de suprimentos. No outro caso, com produto de longo ciclo de fabricação, seria interessante haver eficiência na gestão do projeto. São requisitos bastante diferentes, que provavelmente levarão à escolha de soluções diferentes.

Se analisarmos outras variáveis como tecnologia de processo e tipos de controle - operações discretas ou contínuas - veremos que as necessidades serão muito diversas entre si. E quando caminhamos para outras áreas, como serviços financeiros, saúde, comércio, etc., as prioridades são outras.

Assim, reitera-se: não existe *silver bullet*, solução mágica e universal. O que funcionou bem na organização "A" não necessariamente dará certo na "B". Lozinsky (1996) enfatiza aspectos relativos à avaliação e seleção, ao processo de "evangelização" (obtenção do consenso interno), ao treinamento e motivação dos usuários finais e à manutenção, concluindo que a experiência é útil, mas sua simples transposição quase nunca é suficiente.

Por tudo isso é que se faz necessária uma abordagem muito cuidadosa nessa fase do processo. Discutiremos a seguir alguns pontos que devem rigorosamente ser considerados nessa hora, sempre se levando em consideração que a utilização e seleção de uma ferramenta ERP deve estar solidamente ancorada e alinhada com as grandes diretrizes estratégicas definidas pela empresa.

6. ERP ou Melhores Soluções de Mercado?

Esta é a primeira grande dúvida que usualmente assola os responsáveis pela seleção: adota-se uma solução ERP, ou busca-se no mercado a melhor solução (*best-of-breed*) para cada área da empresa e depois, na medida de suas necessidades, tenta-se dar a elas um certo grau de integração? Como já se disse anteriormente, os ERP cobrem uma vasta área das necessidades da empresa (Produção, Finanças, Recursos Humanos, etc.) - isso acaba tornando menor a necessidade de reconciliar dados entre os diversos módulos (pela não existência de redundâncias), torna mais fácil a utilização de ferramentas de análise e permite mais facilidade para *backup*, ajuste fino (*tuning*) do sistema e outras atividades de manutenção. De qualquer forma, soluções *best-of-breed* não devem ser descartadas sem qualquer análise.

O fato de se usar, com ERP, uma única interface para navegação, *workflow* e geração de relatórios, também permite treinamento mais fácil do pessoal envolvido. Adicionalmente, adquirindo-se um maior número de módulos de um fornecedor de ERP, pode-se ter custos finais menores.

Porém, ERPs não são uma panaceia. Sua implantação normalmente exige um amplo consenso dentro da organização - pacotes separados podem ser implantados de forma menos traumática, menos trabalhosa - certamente o gerente industrial não vai se preocupar com o pacote de recursos humanos.... Nos ERPs, geralmente a modelagem é mais complexa, por abranger, senão todas, quase todas as áreas da empresa.

Ainda em termos de soluções ERP, a tendência é de que sejam funcionalmente mais amplas, porém menos profundas que as *best-of-breed*, o que pode ser crítico em determinadas situações.

Deve-se considerar também aspectos como a arquitetura de TI das soluções em estudo; Hecht (1997) mostra como funcionalidade e arquitetura técnica se relacionam nos sistemas ERP mais populares.

Além disso, apesar do que dizem seus fornecedores, ERPs tendem a ser difíceis de se integrar com sistemas antigos, que já rodavam na organização (sistemas legados ou *legacy systems*), assim como com sistemas de terceiros que porventura sejam necessários para cobrir necessidades muito específicas da empresa. De qualquer forma, esse é um aspecto que deve ser cuidadosamente considerado. Não há respostas prontas e em muitos casos, é melhor sacrificar integração por soluções mais adequadas em determinadas áreas.

Muitos fornecedores ERP já perceberam essa realidade e tendem a buscar tornar mais fácil a conexão de seus produtos com alguns *best-of-breed* de classe mundial; a solução para isso tem sido dada por *middleware* sofisticado, como *data brokers* (para mover dados de um banco de dados ou sistema de arquivos para outro), *message brokers* (integrando aplicações numa base programa a programa) etc.

7. Estratégia Padrão para Seleção

A estratégia mais adequada para selecionar ferramentas ERP consiste em convidar fornecedores a apresentar seus produtos e serviços e a fornecer propostas comerciais. Como há muitos fornecedores, o trabalho pode ser grande e demorado, havendo necessidade de refinar essa estratégia.

A estratégia pode ser desdobrada em duas etapas; a primeira, visando obter informações sobre um número relativamente grande de fornecedores e selecionar os mais bem qualificados. Os passos componentes dessa etapa seriam a preparação de uma solicitação de informações ou RFI (Request For Information), a ser enviada a um razoável número de potenciais fornecedores seguida pela análise das respostas recebidas e seleção de um número menor de possíveis fornecedores; na prática, limitaríamos esses fornecedores a três ou quatro.

Na segunda etapa, deve-se fazer a seleção de um entre os fornecedores mais bem qualificados anteriormente. As atividades que comporiam essa dessa etapa seriam a preparação de uma solicitação de proposta ou RFP (Request For Proposal) a ser enviada aos fornecedores pré-selecionados, à qual se seguiria a avaliação das propostas, com análise mais detalhada do produto segundo critérios como os que são propostos por Breterniz e Galhardi (2011), Medeiros *et al* (2010), Hecht (1997) ou Colangelo (2001) por exemplo. Os passos seguintes, seriam verificar a performance da ferramenta em outras organizações, negociação e escolha do fornecedor.

É importante registrar que a decisão não deve ficar a cargo apenas de um executivo ou técnico. A participação de funcionários que estarão diretamente envolvidos nos aspectos operacionais e gerenciais decorrentes da implantação de ERP é muito importante, organizados em comitê. Mesmo nos casos em que o fornecedor e consultorias assessorarem a empresa, a concretização da implantação depende do pessoal interno.

Infelizmente, muitas organizações usam um procedimento *quick-pick* (escolha rápida, numa tradução livre) para escolha de sua solução - não procuram descobrir qual a melhor solução para a sua realidade, mas tendem a seguir uma lógica inversa: por que não escolher logo a solução mais popular ou por que não contratar logo uma empresa de consultoria e deixar com ela o problema?

A escolha da ferramenta mais popular é uma abordagem tentadora. A maior parte dos profissionais de TI que ocuparam cargos executivos, ao menos no Brasil até o início dos anos 1990, no momento de definir uma arquitetura certamente se viram tentados a dizer: "vou escolher logo IBM e se algo não funcionar, ao menos não poderei ser acusado de haver escolhido mal...".

Ambas são abordagens claramente equivocadas. Permitir que pressões de fornecedores, medos ou política interna levem precocemente ao foco numa única solução, frequentemente leva a uma decisão não baseada em dados confiáveis, sem critérios sólidos e sem nenhuma visão mais ampla da solução (e às vezes, até mesmo do problema...). Simplesmente contratar uma empresa de consultoria para conduzir o processo também pode ser arriscado: a maioria destas prefere, por razões diversas, operar com apenas um ou dois fornecedores, o que também pode levar a soluções equivocadas, pois estas tendem a colocar as "suas" soluções no topo da lista de soluções possíveis e a bloquear a análise de outras alternativas.

Tipicamente pode-se apresentar quatro fatores que dificultam uma adequada seleção: o primeiro é o tempo. A prática tem mostrado que o processo de seleção consume muito tempo e mão de obra, com muito do esforço direcionado apenas para a definição dos critérios-chave de avaliação e para levantamento de alternativas [Hecht 1997].

O segundo problema é o de custo. O custo dos profissionais mencionados, despesas de viagem para visitas e contatos com outros usuários, etc. gera um montante de custos que pode acabar induzindo as empresas (especialmente as de menor porte) a adotarem o *quick-pick*, quando essas despesas na realidade deveriam ser vistas como investimento, podendo inclusive reduzir o montante dos custos envolvidos.

A seguir, pode-se mencionar a falta de objetivos claros e a ausência de dados validados, que deveriam ter sido levantados na etapa anteriormente mencionada, levando o potencial usuário a frequentemente confiar apenas em informações genéricas, material promocional e argumentação de fornecedores para tomada de sua decisão final.

Finalmente, observa-se quase sempre a falta de um processo estruturado para seleção. Sem uma metodologia, por simples que seja, muitas companhias terminam fixando-se num número muito pequeno de critérios para escolha final, ou até mesmo adotando critérios políticos ou a sensibilidade (*feeling*) de alguns envolvidos para a escolha. A simples falta desse processo estruturado pode levar a uma escolha equivocada e ao fracasso do projeto como um todo, como diz Flowers (1996), até justificando o título de sua obra: "falha de *software* é falha de gerenciamento"...

8. Fatores Críticos de Sucesso para a Implementação de uma Ferramenta ERP

Num ambiente corporativo, os Fatores Críticos de Sucesso (FCS) são aqueles aspectos, limitados em número, em que resultados satisfatórios são necessários para o sucesso da organização, indivíduo ou projeto [Bullen e Rockart 1981]. Os casos de fracasso, pela não presença desses FCS são inúmeros [Chen *et al* 2009], inclusive em empresas globais, como Nike, HP e Hershey [Wailgum 2009]. Esses fracassos vão desde o cancelamento total do projeto, até o não cumprimento de prazos e orçamentos, implementação apenas parcial, não obtenção de relação custo/benefício adequada etc.

Assim, além da adequada seleção da ferramenta ERP, que pode ser considerado "o mais crítico dos fatores críticos", há outros fatores a serem considerados para que a implementação tenha êxito; esses fatores são relacionados a seguir, classificados por ordem de importância, embora essa ordem ser alterada em função de características específicas organização; cabe registrar que são todos interconectados.

- Apoio da alta administração: o envolvimento e o compromisso dos níveis mais altos da hierarquia organizacional são importantes pela sua autoridade para alocar os recursos necessários ao projeto, resolver conflitos e eliminar resistências à mudança, processo esse sempre presente em implementações de sistemas desse porte, especialmente porque estes alteram o balanço de forças no seio da organização. A alta administração deve ter claros os objetivos da implementação do ERP, bem como certificar-se de que estes estejam alinhados com a estratégia da organização. Alguns pesquisadores dizem ser comum a não conscientização prévia da alta administração acerca do elevado grau de complexidade, dos esforços necessários para implementação de ferramentas ERP e das mudanças que a mesma traz à organização [Flowers 1996; Chen *et al* 2009; Ekman 2014].
- Consultoria: é praticamente impossível implementar um ERP sem apoio de consultoria externa especializada. A seleção dessa consultoria é vital para que o processo tenha sucesso: quando a mesma não detém *expertise* na solução a ser implementada e elevados padrões éticos, é possível que, mesmo com todos os demais FCS atendidos, o processo não tenha sucesso [Sommers e Nelson 2004].
- Equipe alocada ao projeto: deve ser constituída pelos melhores quadros da organização, com capacidade de liderança, conhecimento da organização como um todo e comprometimento com o projeto. Todas as áreas alcançadas pelo ERP devem estar representadas na equipe [Malhotra e Temponi 2010].
- Gestão de projetos: implementação de ERP é um trabalho de porte, que se não for visto como um projeto tumultuará o andamento das atividades rotineiras e contribuirá para atrasos, aumento de custos etc. Um gerente de projeto experiente talvez seja o profissional mais importante em uma implementação de ERP, especialmente nas etapas de definição da arquitetura da ferramenta, seleção, definição da estratégia de implantação, cronogramas, orçamentos, pontos chave de controle do projeto (*key milestones*). Esse gerente deve ter capacidade de comunicação com todos os envolvidos no projeto e de gerenciamento de crises. Cabe lembrar que Panorama (2018) afirma que 79% dos projetos de implementação de ERP atrasam e em 64%

dos casos os custos superam os orçamentos e cabe ao gerente zelar no sentido de evitar esses problemas, como afirmam Dezdar e Ainin (2011). Em termos de implantação, além das tradicionais faseada ou *big bang*, já há autores, como Gren *et al.* (2018) discutindo a utilização de métodos ágeis nessa fase.

- Reengenharia: Hammer (1990) já dizia "*it is time to stop paving the cow paths*", algo como "é tempo de parar de pavimentar as picadas", no sentido de que tem muito pouca utilidade implementar sistemas de computador sofisticados para automatizar processos ruins. É importante implementar processos racionalizados, estruturados, o que envolve o conhecimento profundo dos processos e sistemas em operação (gestão de conhecimento) e das relações custo/benefício, levando à correta definição dos novos processos e dos quais serão ou não tratados pelo ERP. Cabe lembrar que neste contexto, Reengenharia deve ser vista nos termos propostos por Davenport (2013), buscando-se melhorar continuamente os processos e não altera-los radicalmente. A mesma Panorama (2018) relata que em 75% dos casos houve melhoria de todos os processos de negócio e em 18% os processos chave foram melhorados.
- Comunicação e treinamento: o apoio e o treinamento adequado dos usuários do sistema são fundamentais para que as funcionalidades do mesmo sejam plenamente utilizadas e para que a resistência passiva, difícil de ser detectada e combatida, não adquira proporções que possam a vir comprometer o sucesso do projeto. Em termos de apoio, é fundamental a garantia que os gestores das áreas atingidas pelo sistema apoiem sua implementação; seu apoio e dos usuários chave só pode ser obtido com a adoção de um processo de comunicação eficiente, que deixe claro as razões pelas quais ERP está sendo implementado e quais os ganhos que se pretende obter.
- Customização do sistema: critérios rígidos, que permitam customização apenas em casos muito especiais (*vanilla flavour approach*) também contribuirão para evitar aumento de custos e prazos, além, de facilitarem a manutenção do sistema após o *go live* [Leyh *et al.* 2016]. É importante dividir as funcionalidades em dois grandes grupos, que o mercado chama de "*nice to have*" (seria bom se estivessem presentes) e "*need to have*", aquelas que necessitam estar presentes – essa clara divisão facilita decisões ao se discutir customização [Haddara 2018]. Note-se que os ERP usualmente incorporam opções de melhores práticas, formas padronizadas de ser executar certas atividades na organização, permitindo a esta que escolha a que mais se adapta às suas necessidades, sem que sejam feitas customizações.
- Legados: muita atenção deve ser dada aos sistemas legados (aqueles já existentes na organização), desde o processo de seleção do novo sistema. Sua absorção pelo ERP deve ser planejada cuidadosamente, de forma a não dar aos seus usuários a sensação de que este dificultou ou aumentou sua carga de trabalho, ou, ainda pior, fique impossível essa absorção. Cabe registrar que, com alguma frequência, pode ser útil ou necessária a manutenção de algum desses sistemas, devendo, nesse caso, ser planejada sua integração ao ERP.
- Gerenciamento de mudanças: o desenvolvimento de uma cultura de aceitação de mudanças e de seu gerenciamento levam à criação de um clima adequado

e diminuição das resistências à mudança, que usualmente são muito fortes, como se disse anteriormente [Xu *et al*. 2013].

– Pós implementação: há um velho aforismo em ERP que diz "a vida não termina quando se implementa ERP. Ela começa nesse momento". Isso porque alterações ambientais, internas ou externas à organização, podem exigir alterações na parametrização e/ou customização do aplicativo, *bugs* sempre aparecem, pessoas chave podem deixar a organização etc. Para operar adequadamente um ERP, evitando que funcionalidades comecem a ser substituídas por Shadow IT, é necessário que se mantenha um relacionamento adequado com o fornecedor do aplicativo e consultoria envolvida na implementação, além de revisões constantes nos processos e cuidados com o treinamento contínuo dos usuários e do pessoal de Tecnologia da Informação envolvido [Nicolaou 2004].

Neste momento, torna-se importante o estudo das implicações e cuidados a serem tomados ao se estudar a implementação ou migração de aplicações ERP para a modalidade *cloud computing*. Há indícios de que essa modalidade pode ser vantajosa do ponto de vista de custos, como afirmam Peng e Gala (2014) e de mais fácil implementação [Castellina 2011], mas cuidados especiais, especialmente no que se refere à segurança, devem ser tomados. O autor acompanhou caso de empresa industrial que processa seu ERP nessa modalidade e que, em 2018, sofreu um ataque do tipo *ramsomware* – sequestro de dados – tendo que pagar aos criminosos para que os mesmos fossem liberados; mais relevante é o fato de os dados estarem hospedados em nuvem gerenciada por um grande provedor desse serviço, que se eximiu de qualquer reponsabilidade.

Também devem ser consideradas as maiores dificuldades e limitações para customizações e integrações de ERP processado na modalidade *cloud* com outras ferramentas de *software*, como mostram Duan *et al*. (2013).

9. Considerações Finais

Do que foi visto até o momento, resta a convicção de que a implementação de ERP pode auxiliar as empresas a obterem vantagens competitivas, permitindo-lhes reagir de forma rápida e flexível às mudanças no ambiente de negócios, bem como melhor atender às demandas dos diversos *stakeholders.*

Essa implementação, no entanto, não é isenta de riscos, pelo que deve ser conduzida com extremo cuidado, levando-se em conta os FCS desde o momento em que se inicia o estudo acerca de sua adoção até o *go live*, passando pelos processos de seleção do fornecedor, definição dos responsáveis pela customização e treinamento, *design* de processos, comunicação etc.

Estudos futuros acerca do tema podem ser desenvolvidos seguindo-se diversos eixos, como por exemplo peculiaridades do processo de implementação de ERP em determinados tipos de organização, estudos de caso, aspectos tecnológicos, comportamentais e conexões com tecnologias emergentes como Blockchain, Impressão 3D, Big Data/Analytics, Realidade Virtual/Aumentada, Internet of Things, Robótica, In-memory Processing (estruturas de *hardware* e *software* de base projetadas para processar ERP de forma mais rápida) etc.

Referências

Breternitz, V. J e Galhardi, A. C. (2011) "Contribuições ao processo de seleção de sistemas ERP (Enterprise Resource Planning) para pequenas e médias empresas", Revista Eletrônica de Tecnologia e Cultura. v. 3, nº 2.

Bullen, C. V. e Rockart, J. F. (1981) "A primer on critical success factors". Center for Information Systems Research, Sloan School of Management, MIT, 1981.

Castellina, N. (2011) "SaaS and cloud ERP trends, observations, and performance 2011", Aberdeen Group - Analyst Inside.

Chaturvedi, R. N. (2005) "ERP Implementation Failure at HP", ICMR, https://astro.temple.edu/~wurban/Case%20Studies/HP's%20ERP%20Failure.pdf

Chen, C. C; Law, C. C. H; e Yang, S. C. (2009) "Managing ERP implementation failure: a project management perspective". IEEE Transactions on Engineering Management, v. 56, nº 1.

Chou, P (2018) "Workplace social support and attitude toward Enterprise Resource Planning system: a perspective of organizational change", International Journal of Information Systems and Social Change, v.9, nº 1.

Curran, T. e Keller, G., SAP R/3 Business Blueprint, Prentice-Hall, 1998.

Cyrus, K. M; Aloini, D. e Karimzadeh, S. (2018) "How to disable mortal loops of Enterprise Resource Planning (ERP) implementation: A system dynamics analysis", Systems, v. 6, nº 1.

Davenport, T. H., Process Innovation: Reengineering Work through Information Technology, Harvard Business Press, 2013.

Dezdar, S. e Ainin, S. (2011) "The influence of organizational factors on successful ERP implementation", Management Decision, v. 49, nº 6.

Duan, J; Faker, P; Fesak, A. e Stuart, T. (2013) "Benefits and drawbacks of cloud-based versus traditional ERP systems", Proceedings of the 2012-13 Course on Advanced Resource Planning.

Ekman, P; Thilenius, P. e Windahl, T. (2014) "Extending the ERP system: considering the business relationship portfolio", Business Process Management Journal, v. 20, nº 3.

Farley, G. A. (1998) "Software selection: are you solving the wrong problem?". APICS-The Performance Advantage", edição de março de 1998.

Ferreira, A. A., Reis, A. C. F. e Pereira, M. I., Gestão empresarial: de Taylor aos nossos dias evolução e tendências da moderna administração de empresas, Pioneira, 1997.

Flowers, S., Software failure: management failure. Amazing stories and cautionary tales, John Wiley & Sons, 1996.

Gren, L.; Wong, A.; e Kristoffersson, E. (2018) "Choosing agile or plan-driven enterprise resource planning (ERP) implementations—A study on 21 implementations from 20 companies", https://www.researchgate.net.

Haddara M. (2018) “ERP systems selection in multinational enterprises: a practical guide”, International Journal of Information Systems and Project Management, v. 6, nº 1.

Hammer, M. (1990) “Reengineering Work: Don’t Automate, obliterate”, Harvard Business Review, vol. 68, nº 4.

Hecht, B. (1997) “Choose the right ERP software”, Datamation, v. 43, nº 3.

Hicks, D. A. (1997) “The manager’s guide to supply chain and logistics problem-solving tools and techniques”, IEEE Solutions, v. 29, nº 10.

Koontz, H. e O’Donnell, C., Administração, organização, planejamento e controle, Pioneira, 1986.

Lemonakis, C; Sariannidis, N; Garefalakis, A. e Adamou, A. (2018) “Visualizing operational effects of ERP systems through graphical representations: current trends and perspectives”. Annals of Operations Research (2018).

Leyh, R; Heinisch, C; Kungl, M.T. e Spangler, G. (2016) “Attachment representation moderates the influence of emotional context on information processing”, Frontiers in Human Neuroscience”, https://www.frontiersin.org/articles/10.3389/fnhum.2016.00278/full.

Lieber, R. B. (1995) “Here comes SAP”, Fortune, v.132, nº 7.

Lozinsky, S., Software: tecnologia do negócio, Imago, 1996.

Malhotra, R. e Temponi, C. (2010) “Critical decisions for ERP integration: small business issues”, International Journal of Information Management, v. 30, nº 1.

Martins, P. G. e Laugeni, F. P., Administração da Produção, Saraiva, 2010.

Medeiros Junior, A; Perez, G. e Shimizu, T. (2010) “Classificação de critérios para seleção de ERP: um estudo utilizando a Técnica Delphi”, Ryhevista Eletrônica de Sistemas de Informação, v. 9, nº. 1.

Meneghetti, F. K. (2011) “O que é um ensaio-teórico?”. Revista de Administração Contemporânea, vol. 15, nº 2.

Nicolaou A.I., (2004). “Quality of postimplementation review for Enterprise Resource Planning systems”. International Journal of Accounting Information Systems, v.5, nº 1.

Ortega y Gasset, J., Meditaciones del Quijote - in: Obras Completas, vol. I, Taurus, 2004.

Panorama Consulting (2015) “2015 ERP report”, go.panorama-consulting.com/rs/panoramaconsulting/images/2015%20ERP%20Report.pdf

Panorama Consulting (2018), “Panorama’s 2018 ERP Report“, www.panorama-consulting.com/resource-center/erp-industry-reports/panoramas-2018-erp-report/

Paviani, J. (2009) “O ensaio como gênero textual”, V Simpósio Internacional de Estudo de Gêneros Textuais (Anais).

Peng, G. C. A. e Gala, C. (2014) ”Cloud ERP: a new dilemma to modern organisations?”, Journal of Computer Information Systems, v. 54, nº 4.

Sabau, G., Munten, M., Bologa, A. R., Bologa, R. e Surcel, T. (2009) "An evaluation framework for higher education ERP systems", WSEAS Transactions on Computers, v. 8, n. 11.

Scott, J. (1999) "The FoxMeyer Drug's bankruptcy: Was it a failure of ERP?" Americas Conference on Information Systems (AMCIS).

Slack, N; Chambers, S e Johnston, R. Administração da produção. Atlas, 2015.

Somers T.M. e Nelson K.G. (2004) "A taxonomy of players and activities across the ERP project life cycle", Information & Management, v. 41, nº 3. .

Wailgum, T. (2009) "10 famous ERP disasters, dustups and disappointments", CIO Magazine2009, Nova Iorque, IDG Publishing.

Wight, O. W., MRP II: unlocking America's productivity potential, Oliver Wight Limited Publications, 1981.

Xu, H., Rondeau, P. J. e Mahenthiran, S. (2011) "Teaching case the challenge of implementing an ERP system in a small and medium enterprise-A teaching case of ERP project management", Journal of Information Systems Education, v. 22, nº 4.

Zhao, F., e Kirche, E. (2013) "Continuing on-premise or adopt on-demand? An empirical study of ERP adoption in SMEs". International Conference on Human-Computer Interaction.

2. Capítulo 2

Autores:

Allan de Godoi Contessoto

Universidade Estadual Paulista “Júlio de Mesquita Filho” (UNESP) - São José do Rio Preto

email: allan.contessoto@unesp.br

Anderson Rici Amorim

Universidade Estadual Paulista “Júlio de Mesquita Filho” (UNESP) - São José do Rio Preto

email: anderson.amorim@unesp.br

Rogéria Cristiane Gratão de Souza

Universidade Estadual Paulista “Júlio de Mesquita Filho” (UNESP) - São José do Rio Preto

email: rogeria.souza@unesp.br

Capítulo 2

Abordagem prática para gestão de projetos de software utilizando o guia PMBoK

Allan de Godoi Contessoto, Anderson Rici Amorim e Rogéria Cristiane Gratão de Souza

Abstract

Due to the current and highly competitive scenario of software development, it is important to use techniques and strategies that contribute to the final quality of the developed product. This way, the application of Project Management techniques in the software life cycle is necessary to allow the desired monitoring and control. Therefore, this chapter aims to introduce different knowledge areas from Project Management Body of Knowledge – PMBoK, a well-known guide to Project Management, applied to software development. Thus, it is used a theoretical-practical approach, presenting important concepts with help of tools to apply the activities of management through the software life cycle.

Resumo

Devido ao presente cenário altamente competitivo do mercado de desenvolvimento de software, é de suma importância o uso de técnicas e estratégias que contribuam para a qualidade do produto desenvolvido. Dessa forma, a aplicação de técnicas de Gerenciamento de Projetos durante todo o ciclo de desenvolvimento de um software se faz necessária para permitir o monitoramento e controle almejados. Assim, o presente capítulo tem o objetivo de introduzir as diferentes áreas do conhecimento estabelecidas no Project Management Body of Knowledge – PMBoK, um conceituado guia voltado para o Gerenciamento de Projetos, com ênfase no desenvolvimento de software. Para tanto, é utilizada uma abordagem teórico-prática, apresentando os conceitos relevantes com o apoio de ferramentas existentes que auxiliam a efetiva aplicação das atividades de gestão ao longo do ciclo de desenvolvimento do software.

1.1. Introdução

O alto nível de competição existente entre as organizações que desenvolvem software proporciona um cenário no qual é imprescindível a realização de projetos fundamentados em um planejamento sólido que contribua para o pleno atendimento da qualidade esperada pelas partes interessadas. Dessa forma, o gerenciamento de projetos ganha força e importância nas organizações que desenvolvem software, uma vez que proporciona um ambiente no qual se permite administrar a complexidade dos projetos [Pressman 2011]. A busca pelo gerenciamento de projetos eficiente é um atrativo para as organizações, pois um mau gerenciamento pode resultar em falhas no produto de software decorrente do processo de desenvolvimento [Sommerville 2011].

Um projeto é definido como sendo um esforço temporário que é empreendido para criar um produto, serviço ou resultado exclusivo, que visa satisfazer as reais necessidades do cliente [PMBoK 2017]. Assim, o gerenciamento de projetos é definido como sendo a aplicação de conhecimentos, habilidades, ferramentas e técnicas nas atividades do projeto para, assim**,** cumprir os requisitos dentro do prazo e custo acordados com o cliente e com a qualidade almejada [PMBoK 2017, Sommerville 2011, Laruccia *et al*. 2012].

O gerenciamento de projetos tem caráter interdisciplinar e, por isso, pode ser aplicado em diversos setores da economia. Apesar disso, sua aplicação é complexa, uma vez que a gestão de projetos exige o conhecimento de diversas áreas do conhecimento por parte da equipe de desenvolvimento. Com o intuito de auxiliar na aplicação e promover a gestão de projetos, o PMI – *Project Management Institute* desenvolveu o Guia PMBoK – *Project Management Body of Knowledge* [PMBoK 2017], que reúne as boas práticas e diretrizes amplamente reconhecidas pelos gerentes de projetos. O guia é dividido em 10 áreas de conhecimento, a saber: Integração, Escopo, Cronograma, Custo, Qualidade, Recursos, Comunicação, Risco, Aquisição, Partes interessadas.

Atualmente o guia PMBoK é o principal referencial da literatura, estando na sua sexta versão, lançada em setembro de 2017. Em tal versão, o guia apresenta um conjunto de textos complementares, nos quais existe o foco no desenvolvimento de projetos de forma ágil [PMBoK 2017, PMI – Agile Alliance 2017]. Para organizações que desenvolvem software, a metodologia ágil representa atualmente um diferencial de mercado. As abordagens ágeis permitem que as organizações desenvolvam projetos de software de pequeno e médio porte com qualidade, por meio de um processo de desenvolvimento que permite o controle da complexidade dos projetos. Apesar da abrangência apresentada pelo guia PMBoK, é possível a sua aplicação em projetos de desenvolvimento de software, independentemente do tamanho do software a ser produzido.

Nesse sentido, esse capítulo tem como objetivo apresentar os principais conceitos e definições a respeito do gerenciamento de projetos presentes no guia PMBoK, com ênfase no processo de desenvolvimento de software. Para tanto, as próximas seções estão organizadas da seguinte forma: na seção 1.2 é apresentado o contexto histórico a respeito do gerenciamento de projetos, partindo das primeiras ocorrências da gestão de projetos até os dias atuais; na seção 1.3 é descrita a estrutura do Guia PMBoK, seus conceitos principais e as áreas do conhecimento que compõem o gerenciamento de projetos; na seção 1.4 são apresentadas ferramentas computacionais,

comerciais e acadêmicas, de apoio ao gerenciamento de projetos; na seção 1.5 é apresentado um estudo de caso, exemplificando a aplicação dos conceitos discutidos ao longo do capítulo; e, por fim, na seção 1.6 são apresentadas as conclusões do capítulo.

1.2. Contexto Histórico

O gerenciamento de projetos ganha cada vez mais importância e destaque nos meios empresariais e acadêmicos, o que caracterizaria ser algo recente. No entanto, estudos mostram que essa prática acontece, mesmo que de forma rudimentar, desde os primórdios da humanidade, presente em mega-construções que marcaram a história, como as Pirâmides do Egito, a Muralha da China e o Coliseu [Mattos *et al.* 2012]. Porém, foi apenas no século XX que se deu início aos primeiros estudos científicos com o intuito de fazer uma formalização do gerenciamento de projetos.

Frederick Winslow Taylor (1856-1915), conhecido como o pai da administração científica, foi o responsável pela formulação dos primeiros conceitos do gerenciamento moderno. Taylor aplicou um raciocínio científico ao trabalho, mostrando que este pode ser analisado e melhorado se dividido e focado em suas partes elementares [Oliveira 2003].

Simultaneamente aos estudos de Taylor, seu sócio Henry Gantt (1861-1919), que é considerado o pai do gerenciamento de projetos, realizou um estudo detalhado sobre a ordem das operações no trabalho. Gantt propôs a apresentação das atividades do projeto por meio de um gráfico com barras e marcos conhecido como Gráfico de Gantt. Esse gráfico ainda é amplamente utilizado, mais de noventa anos depois de sua criação, em softwares de gerenciamento de projetos sem nenhuma alteração dos originais, com o intuito de planejar e controlar as atividades de um projeto [Oliveira 2003].

Com o avançar dos anos, o gerenciamento de projetos foi ganhando importância e destaque nas organizações. Diante disso, inúmeros investimentos foram realizados com o intuito de promover e desenvolver os conhecimentos a respeito da gestão de projetos. Como resultado, diversas metodologias e normas para a correta aplicação dessa área foram desenvolvidas e publicadas ao longo dos anos.

No ano de 1969, durante o auge da corrida espacial, um conjunto de cinco profissionais da área fundou o PMI – *Project Management Institute*, na Philadelphia, Pensilvania, Estados Unidos [Esteca 2010], com o objetivo de agrupar as boas práticas de gestão de projetos, definir padrões para a área e promover o seu desenvolvimento. A reunião dos profissionais da área de gestão de projetos e as discussões regulares acabaram por consolidar um guia com os principais conceitos da área e boas práticas. Esse guia recebeu o nome de PMBoK – *Project Management Body of Knowledge* e sua primeira versão foi lançada no ano de 1996. Atualmente, o PMI é composto por mais de 700 mil membros, que estão espalhados por todo o mundo, aplicando a gestão de projetos segundo as boas práticas presentes no guia PMBoK.

O gerenciamento de projetos foi e continua sendo uma disciplina fundamental para o desenvolvimento de projetos, uma vez que a sua correta aplicação proporciona às organizações um melhor controle sobre suas atividades, as quais, assim, conseguem ser bem-sucedidas em um maior número de projetos [Bomfin *et al*. 2012]. Com isso,

destaca-se a importância do guia PMBoK, sendo atualmente a principal referência para os profissionais da área.

Especificamente no contexto de software, observa-se que a criação do manifesto ágil, em meados de 2001, foi um marco importante para as organizações e profissionais da área. Descontentes com as abordagens tradicionais de desenvolvimento de software, um grupo de especialistas, liderados por Kent Beck, criaram um conjunto de princípios para uma nova abordagem de desenvolvimento de software [Sommerville 2011]:

- Indivíduos e interações mais que processos e ferramentas.
- Software em funcionamento mais que documentação abrangente.
- Colaboração com o cliente mais que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

Devida a importância da utilização de métodos ágeis para o desenvolvimento de software, o PMI apresenta, em sua versão atual do guia PMBoK, todos os conceitos de gestão de projetos adaptados para o ambiente de desenvolvimento ágil. Além disso, foi publicado, em conjunto com o PMBoK, o *Agile Practice Guide* - um guia para a correta aplicação da gestão de projetos ágeis [PMI – Agile Alliance 2017]. Assim, com o suporte do PMBoK, o contexto de gestão de projetos ágeis ganha destaque, o que é relevante para organizações que desenvolvem software.

1.3. A Estrutura do Guia PMBoK

O guia PMBoK, ao longo dos anos, passou por atualizações, de modo a estar sempre em conformidade com as necessidades do mercado. Atualmente o PMBoK se encontra na versão 6 e o grande diferencial em relação à versão anterior é a promoção de gestão de projetos ágil. O PMBoK é um conjunto de 49 processos divididos em cinco grupos de processos, a saber [PMBoK 2017]:

- Iniciação: processos que definem um novo projeto ou uma nova fase de um projeto existente**,** por meio da obtenção de autorização para iniciar o projeto ou fase.
- Planejamento: processos executados para definir o escopo do projeto, refinar os objetivos e definir a linha de ação necessária para alcançar os objetivos do projeto.
- Execução: processos utilizados para executar o trabalho definido no plano de gerenciamento do projeto, buscando satisfazer aos requisitos do projeto.
- Monitoramento e controle: processos responsáveis por acompanhar, analisar e controlar o progresso e o desempenho do projeto, identificar áreas que necessitam de mudanças e iniciar as mudanças correspondentes.
- Encerramento: processos executados para finalizar as atividades dos grupos de processos, com o intuito de encerrar formalmente o projeto ou fase.

É importante destacar que os processos estão dispostos entre 10 áreas do conhecimento que compõem o gerenciamento de projetos. Assim, as áreas do

conhecimento são aplicadas durante o desenvolvimento de um projeto, seguindo a interação prevista entre os cinco grupos de processos.

A aplicação de conhecimentos e habilidades e a utilização de ferramentas e técnicas a respeito de cada uma das áreas do conhecimento visam proporcionar uma gestão eficiente de cada projeto. A seguir são descritas cada uma das áreas do conhecimento, bem como os processos acomodados em cada área.

1.3.1. Gerenciamento de Integração do Projeto

Essa área do conhecimento define o conjunto de processos utilizados para identificar, definir, combinar, unificar e coordenar os diferentes processos e atividades dentro dos grupos de processos de gerenciamento do projeto. Assim, essa área do conhecimento é fundamental, uma vez que, por meio dela, ocorre a integração entre os diferentes processos e pessoas envolvidas no desenvolvimento do projeto, fazendo com que tudo trabalhe em harmonia. Assim, é possível afirmar que o gerenciamento de integração é o núcleo de um bom gerenciamento de projetos e deve ser aplicado desde o início do projeto até a sua conclusão.

Para a aplicação dessa área do conhecimento é fundamental que a equipe do projeto realize constantes reuniões, com o intuito de monitorar e revisar como o trabalho está sendo realizado, além de acompanhar o progresso do projeto. O gerenciamento de integração não muda quando a abordagem de desenvolvimento do projeto é ágil, apenas o controle sobre o planejamento do produto e sobre as entregas passam a ser delegadas à toda a equipe de desenvolvimento. Assim, as grandes responsabilidades do gerente de projeto são criar uma equipe que seja capaz de reagir a mudanças e estabelecer um ambiente que auxilie a tomada de decisão colaborativa. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Desenvolver o termo de abertura do projeto: responsável por desenvolver um documento que formalmente autoriza a existência de um projeto e fornece autoridade ao gerente do projeto para aplicar recursos organizacionais sobre as atividades do projeto.
- Desenvolver o plano de gerenciamento do projeto: responsável por definir, preparar e coordenar todos os componentes do plano, de modo a criar um plano integrado do gerenciamento do projeto.
- Orientar e gerenciar o trabalho do projeto: responsável por liderar e realizar o trabalho definido no plano de gerenciamento do projeto e pela implantação das mudanças aprovadas para atingir os objetivos do projeto.
- Gerenciar o conhecimento do projeto: responsável por utilizar os conhecimentos existentes e criar novos conhecimentos, com o intuito de alcançar os objetivos do projeto e contribuir para a aprendizagem da organização.
- Monitorar e controlar o trabalho do projeto: responsável por realizar o acompanhamento, análise e relato do progresso geral do projeto, de modo a atender aos objetivos de desempenho definidos no plano de gerenciamento do projeto.

- Realizar o controle integrado de mudanças: responsável por revisar todas as solicitações de mudanças, aprovar as mudanças e gerenciá-las nas entregas, nos ativos da organização, nos documentos do projeto e no plano de gerenciamento do projeto.
- Encerrar o projeto ou fase: responsável por finalizar todas as atividades, fases ou contratos do projeto.

1.3.2. Gerenciamento do Escopo do Projeto

Essa área do conhecimento define os processos utilizados para certificar que o projeto inclui todo o trabalho necessário, e apenas o trabalho necessário, para que o projeto seja concluído com sucesso, de modo a evitar que a equipe de desenvolvimento execute atividades desnecessárias. Assim, o gerenciamento do escopo busca controlar os limites do projeto, delimitando o que está e o que não está incluso no projeto.

O termo escopo pode ser utilizado para se referir ao escopo do produto ou ao escopo do projeto. Quando o termo escopo é utilizado para definir o produto, ele é entendido como sendo os limites do produto, ou seja, são características, restrições e detalhes que permitem definir o produto, serviço ou resultado almejado com o projeto. Por sua vez, quando o termo é utilizado para definir o projeto, ele é entendido como sendo o trabalho que deve ser realizado para entregar um produto, serviço ou resultado com as características e funções especificadas. O escopo do projeto engloba o escopo do produto [Alexander e Beus-Dukic 2009, Lampa 2015].

A realização de uma boa gestão do escopo, durante o ciclo de vida do projeto, permite o controle sobre se o planejado está sendo executado. Nesse sentido, quando o gerenciamento do escopo é mal executado, fatalmente ocorrem atrasos, aumentos nos custos e insatisfação das partes interessadas [Degrossi 2011].

Os projetos que utilizam uma abordagem ágil de desenvolvimento, como grande parte dos projetos de desenvolvimento de software, tem um ciclo de vida do projeto diferente daqueles que utilizam uma abordagem preditiva ou tradicional de desenvolvimento. Em uma abordagem preditiva, as entregas do projeto são definidas no início do projeto e qualquer mudança precisa ser gerenciada. Por sua vez, em uma abordagem ágil, as entregas são desenvolvidas em várias iterações, nas quais o escopo é definido e aprovado para cada iteração. Assim, o escopo geral do projeto é dividido em um conjunto de requisitos e atividades a serem executadas, chamado *backlog* do produto. No começo de cada iteração, a equipe define quais os requisitos e atividades serão desenvolvidas na iteração atual que irão resultar na entrega. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento do escopo: responsável por criar o plano de gerenciamento do escopo que indica como os escopos do projeto e do produto serão definidos, validados e controlados.
- Coletar os requisitos: responsável por definir, documentar e gerenciar as necessidades e requisitos das partes interessadas, de modo a atender aos objetivos do projeto.

- Definir o escopo: responsável por desenvolver uma descrição detalhada do projeto e do produto.
- Criar a Estrutura Analítica do Projeto – EAP: responsável por criar a EAP, subdividindo as entregas e o trabalho do projeto em componentes menores que são mais facilmente gerenciáveis.
- Validar o escopo: responsável por formalizar a aceitação das partes interessadas sobre as entregas concluídas do projeto.
- Controlar o escopo: responsável por monitorar o *status* do projeto e do produto e gerenciar as mudanças feitas na linha de base do escopo.

1.3.3. Gerenciamento do Cronograma do Projeto

Essa área do conhecimento define os processos necessários para gerenciar o término pontual do projeto. Assim, o cronograma fornece o plano detalhado de como e quando serão executadas atividades que irão resultar no produto, serviço ou resultado esperado e quando irão ocorrer as entregas do projeto.

Para a criação do cronograma, a equipe de desenvolvimento define os dados específicos do projeto, como atividades a serem realizadas, datas previstas das entregas, durações de cada uma das atividades, dependências e restrições. Tais dados são inseridos em algum modelo de cronograma, sendo os mais relevantes o Gráfico de Gantt, a Rede de Cronograma e a Lista de Atividades, de modo a criar uma representação do cronograma do projeto.

A aplicação do gerenciamento do cronograma em abordagens ágeis considera que o projeto é criado em ciclos curtos de trabalho, os quais formam o cronograma geral do projeto. Assim, cada ciclo precisa ser acoplado dentro do cronograma geral do projeto, sendo o papel do gerente do projeto fundamental para gerir o cronograma que é dividido em ciclos pequenos de desenvolvimento. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento do cronograma: responsável por estabelecer as políticas, os procedimentos e a documentação para o planejamento, desenvolvimento, gerenciamento, execução e controle do cronograma do projeto.
- Definir as atividades: responsável por identificar e definir as ações específicas que devem ser realizadas para produzir as entregas do projeto.
- Sequenciar as atividades: responsável por identificar e documentar os relacionamentos entre as atividades do projeto.
- Estimar as durações das atividades: responsável por estimar o número de períodos de trabalho que serão necessários para terminar as atividades individuais com os recursos estimados.
- Desenvolver o cronograma: responsável por criar o modelo de cronograma do projeto para execução, monitoramento e controle, por meio da análise da sequência de atividades, durações, recursos e restrições do cronograma.
- Controlar o cronograma: responsável por monitorar o *status* do projeto, de modo a atualizar e gerenciar mudanças no cronograma.

1.3.4. Gerenciamento dos Custos do Projeto

Essa área do conhecimento define os processos envolvidos no planejamento, estimativas, orçamentos, financiamentos, gerenciamento e controle dos custos, com o intuito de garantir que o projeto possa ser concluído dentro do orçamento aprovado. Assim, o gerenciamento dos custos evolui ao longo do ciclo de vida do projeto, visto que com o desenvolvimento do projeto, os custos precisam ser revisados e atualizados. É importante mencionar que os custos incluem também o suporte e manutenção do projeto [Salim 2013].

Os projetos desenvolvidos por meio de abordagens ágeis não têm o escopo do projeto muito bem definido nas suas fases iniciais, o que torna o cálculo dos custos algo muito difícil de ser realizado com sucesso, uma vez que muitas incertezas ainda existem no projeto. Assim, a utilização de métodos de estimativas é sugerida, tais como estimativa análoga, estimativa paramétrica, estimativa *bottom-up* e estimativa de três pontos, uma vez que podem gerar uma previsão rápida dos custos do projeto, principalmente dos custos com mão de obra e do desenvolvimento de um componente específico. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento dos custos: responsável por definir os custos do projeto, os quais serão estimados, orçados, gerenciados, monitorados e controlados.
- Estimar os custos: responsável por desenvolver uma aproximação dos recursos monetários necessários para terminar o trabalho do projeto.
- Determinar o orçamento: responsável por agregar os custos estimados para cada uma das atividades ou para os pacotes de trabalho, com o intuito de estabelecer uma linha de base dos custos autorizados para o projeto.
- Controlar os custos: responsável por monitorar o *status* do projeto, com o intuito de atualizar os custos e gerenciar mudanças na linha de base dos custos.

1.3.5. Gerenciamento da Qualidade do Projeto

Essa área do conhecimento define os processos e atividades que visam incorporar as políticas de qualidade, os objetivos e as responsabilidades da organização, para que o projeto possa satisfazer às necessidades para as quais foi empreendido. Além disso, o gerenciamento da qualidade oferece o suporte para a realização de atividades que visam a melhoria contínua dos processos realizados pela organização.

A aplicação da gestão da qualidade em projetos desenvolvidos por meio de abordagens ágeis é realizada por meio de frequentes revisões da qualidade, que verificam a eficácia dos processos de qualidade e procuram encontrar a causa-raiz dos problemas, de modo a sugerir novas abordagens para que a qualidade seja melhorada. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento da qualidade: responsável por identificar os requisitos e os padrões de qualidade do projeto e de suas entregas, bem como documentar como o projeto demonstrará a conformidade com os requisitos e os padrões de qualidade do projeto.

- Gerenciar a qualidade: responsável por transformar o plano de gerenciamento da qualidade em atividades a serem executadas, com o intuito de incorporar ao projeto as políticas de qualidade da organização.
- Controlar a qualidade: responsável por monitorar e registrar os resultados da execução das atividades do gerenciamento de qualidade, com o intuito de avaliar o desempenho e garantir que as entregas do projeto sejam completas, corretas e atendam as expectativas.

1.3.6. Gerenciamento dos Recursos do Projeto

Essa área do conhecimento define os processos para identificar, adquirir e gerenciar todos os recursos necessários para a conclusão com sucesso do projeto. Além disso, busca garantir que os recursos estejam disponíveis para a equipe de desenvolvimento no instante em que forem necessários.

É importante definir os recursos que são gerenciados por essa área do conhecimento. Assim, recursos físicos incluem os equipamentos, materiais, instalações e infraestrutura necessária para a implantação do projeto. Por sua vez, recursos de equipe e pessoal referem-se a todos os profissionais envolvidos no desenvolvimento do projeto que atuam em tempo integral ou parcial.

Os projetos desenvolvidos por abordagens ágeis têm como características equipes que buscam ao máximo a colaboração entre os indivíduos, de modo a facilitar a integração das atividades e dos recursos. Assim, a produtividade é impulsionada e os recursos são gerenciados por todos. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento dos recursos: responsável por definir como estimar, adquirir, gerenciar e utilizar os recursos físicos e a equipe durante o ciclo de vida do projeto.
- Estimar os recursos das atividades: responsável por estimar os recursos da equipe, o tipo e as quantidades de materiais, equipamentos e suprimentos necessários para realizar o projeto.
- Adquirir recursos: responsável por identificar os membros da equipe e obter instalações, equipamentos, materiais, suprimentos e outros recursos necessários para concluir o projeto.
- Desenvolver a equipe: responsável por melhorar o ambiente de trabalho, as competências e a integração da equipe de desenvolvimento, com o intuito de aprimorar o desempenho do projeto.
- Gerenciar a equipe: responsável por acompanhar o desempenho dos membros da equipe, de modo a fornecer *feedback*, resolver problemas e gerenciar mudanças, com o intuito de otimizar o desempenho do projeto.
- Controlar os recursos: responsável por garantir que os recursos físicos atribuídos e alocados para o projeto estejam disponíveis conforme o planejado, bem como monitorar o uso real dos recursos em comparação com o que foi estimado, de modo a realizar ações corretivas quando necessário.

1.3.7. Gerenciamento das Comunicações do Projeto

Essa área do conhecimento define os processos necessários para garantir que as informações do projeto sejam atendidas, por meio do desenvolvimento de artefatos e da implantação de atividades voltadas para a troca eficiente de informações. O gerenciamento das comunicações é muito importante para a realização do projeto com sucesso, uma vez que busca criar uma relação entre as diversas partes interessadas no projeto, de modo a respeitar suas diferentes culturas, níveis de conhecimento e expectativas sobre o projeto [Azevedo 2017].

Para a execução da gestão das comunicações o gerente de projeto tem um papel fundamental, sendo o responsável por gerenciar as comunicações internas e externas à sua organização e, ainda, por gerenciar o engajamento das diferentes partes interessadas [Maretti *et al*. 2016, Azevedo 2017]. Em projetos desenvolvidos por meio de abordagens ágeis a comunicação é fundamental, pois as constantes mudanças e detalhes da evolução do projeto necessitam ser comunicados com frequência e rapidez para as diferentes partes interessadas. Além disso, as abordagens ágeis são baseadas no constante *feedback* dos clientes. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento das comunicações: responsável por desenvolver uma abordagem e um plano adequado para as atividades de comunicação durante o ciclo de vida do projeto, considerando as informações necessárias de cada parte interessada, dos ativos organizacionais e do projeto.
- Gerenciar as comunicações: responsável por garantir a coleta, distribuição, armazenamento, recuperação, gerenciamento, monitoramento e disposição final das informações do projeto, de forma que sejam compreensíveis e adequadas.
- Monitorar as comunicações: responsável por garantir que as necessidades de informação do projeto e de suas partes interessadas sejam atendidas.

1.3.8. Gerenciamento dos Riscos do Projeto

Essa área do conhecimento define os processos necessários para o tratamento dos eventos incertos que podem afetar os objetivos do projeto. A gestão dos riscos tem como objetivo aumentar a probabilidade de ocorrência e o impacto dos riscos positivos, denominados oportunidades, e diminuir a probabilidade de ocorrência e o impacto dos riscos negativos, denominados ameaças, com o intuito de otimizar as chances de sucesso do projeto [Contessoto 2015, PMBoK 2017].

A gestão de riscos acompanha todo o ciclo de vida do projeto, contudo os riscos são mais propensos a ocorrer nas fases iniciais do projeto, devido à grande quantidade de incertezas associadas ao projeto. Apesar disso, os riscos que ocorrem nas fases iniciais têm um impacto menor sobre o projeto, pois poucos recursos foram gastos no início do projeto [Sant’ Ana 2015].

O gerenciamento dos riscos ganha força e destaque nas organizações que desenvolvem software, pois quando essa área do conhecimento é aplicada corretamente, a possibilidade de sucesso do projeto aumenta, visto que os eventos incertos que podem ocorrer passam a ser gerenciados. Apesar disso, a aplicação dessa área do conhecimento ainda enfrenta alguns desafios, tais como a falta de ferramentas computacionais que

apoiem a aplicação da gestão de riscos e a falta de conhecimento e experiência dos profissionais que desenvolvem software. Nesse sentido, surgiu o *Risk Management Maturity Method for Software Development* - R3M-SD, com o intuito de auxiliar as organizações que desenvolvem software na correta aplicação da gestão de riscos [Contessoto 2017].

Projetos desenvolvidos por meio de abordagens ágeis, por definição, apresentam mais incertezas do que projetos desenvolvidos por meio de abordagens tradicionais. Assim, a aplicação da gestão de riscos demanda revisões frequentes sobre as entregas incrementais e acompanhamento das equipes de desenvolvimento, com o intuito de acelerar o compartilhamento de informações e garantir que todos os riscos possam ser identificados, compreendidos e gerenciados. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento dos riscos: responsável por definir como são conduzidas as atividades de gerenciamento de riscos de um projeto.
- Identificar os riscos: responsável por identificar os riscos individuais do projeto e as fontes de riscos, de modo a documentar suas características.
- Realizar a análise qualitativa dos riscos: responsável por realizar a priorização de riscos do projeto para análise ou ação posterior, por meio da avaliação de sua probabilidade de ocorrência e impacto, assim como outras características.
- Realizar a análise quantitativa dos riscos: responsável por realizar uma análise numérica dos efeitos combinados dos riscos identificados nos projetos e outras fontes de incertezas sobre os objetivos do projeto.
- Planejar as respostas aos riscos: responsável por desenvolver alternativas, selecionar estratégias e acordar ações para lidar com os riscos individualmente.
- Implementar respostas aos riscos: responsável por implementar as respostas planejadas para lidar com os riscos.
- Monitorar os riscos: responsável por monitorar a implementação das respostas, acompanhar os riscos identificados, identificar e analisar novos riscos e avaliar a eficácia da gestão de riscos ao longo do ciclo de vida do projeto.

1.3.9. Gerenciamento das Aquisições do Projeto

Essa área do conhecimento define os processos necessários para comprar ou adquirir produtos, serviços ou resultados externos, como também para gerenciar todos os contratos referentes ao projeto. É fundamental para a aplicação dessa área do conhecimento que o gerente de projeto consiga tomar decisões inteligentes a respeito das aquisições, de modo a obter vantagens para a sua organização.

Para os ambientes ágeis pode ocorrer o trabalho colaborativo entre a equipe de desenvolvimento e o vendedor de produtos ou serviços. Nesse caso, o vendedor é considerado como uma extensão da equipe de desenvolvimento e as recompensas e riscos do projeto são compartilhados. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Planejar o gerenciamento das aquisições: responsável por documentar as decisões de compras do projeto, de modo a especificar a abordagem de compras e identificar potenciais vendedores.
- Conduzir as aquisições: responsável por escolher um vendedor, dentre os possíveis vendedores, e realizar a adjudicação de um contrato.
- Controlar as aquisições: responsável por gerenciar as aquisições, monitorar o desempenho do contrato, fazer alterações e correções conforme apropriado e encerrar contratos.

1.3.10. Gerenciamento das partes interessadas do projeto

Essa área do conhecimento define os processos determinados para identificar todas as pessoas, grupos ou organizações que podem impactar ou serem impactadas pelo projeto, analisar as expectativas das partes interessadas e seu impacto no projeto, e desenvolver estratégias para o envolvimento das partes interessadas nas decisões e execução do projeto. O gerente de projeto e a equipe de desenvolvimento precisam ter a habilidade de identificar corretamente e engajar as partes interessadas de forma apropriada, pois isso pode fazer a diferença entre o sucesso e o fracasso do projeto.

Por conta de suas características, projetos desenvolvidos por abordagens ágeis requerem um grande engajamento e participação ativa das partes interessadas, com o intuito de auxiliar na tomada rápida de decisões e dar *feedback* sobre o projeto que está sendo desenvolvido. Os processos que compõem essa área do conhecimento são [PMBoK 2017]:

- Identificar as partes interessadas: responsável por identificar as partes interessadas do projeto, analisando e documentando informações relevantes sobre suas necessidades, interesses, expectativas e, por fim, o impacto potencial no sucesso do projeto.
- Planejar o engajamento das partes interessadas: responsável por desenvolver abordagens para envolver as partes interessadas do projeto, com base em suas necessidades, expectativas, interesses e impacto potencial.
- Gerenciar o engajamento das partes interessadas: responsável por realizar a comunicação e trabalho com as partes interessadas, de modo a atender suas necessidades e expetativas. Além disso, esse processo busca promover o engajamento das partes interessadas quando adequado.
- Monitorar o engajamento das partes interessadas: responsável por monitorar as relações entre as partes interessadas do projeto e realizar adequações nas estratégias para engajar as partes interessadas.

1.4. Ferramentas de Apoio à Gestão de Projetos

No mercado e na literatura atual existem inúmeras ferramentas que têm como objetivo oferecer um gerenciamento de projetos ágil, planejado e seguro, dando suporte, assim, para a tomada de decisão dos gerentes de projetos. De certa forma, essas ferramentas disponibilizam funções semelhantes que se diferem na abrangência em que cada uma apresenta as áreas de conhecimento, sem serem completas, uma vez que priorizam

determinadas áreas em prejuízo das outras e atendem parcialmente ou, até mesmo, não atendem, as diretrizes previstas no Guia PMBoK. Para ilustrar esse cenário são apresentadas, a seguir, informações sobre ferramentas comerciais e acadêmicas com o intuito de resumir suas particularidades e as respectivas abordagens com relação ao gerenciamento de projetos.

1.4.1. Microsoft Project [MSProject, 2018]

O MS Project é um sistema *desktop* com inúmeros recursos disponíveis na *web*, comercializado pela Microsoft, sendo um dos sistemas de gerenciamento de projetos mais utilizados por profissionais da área em todo o mundo. O MS Project teve sua primeira versão em 1985 e, com o passar dos anos, sofreu diversas modificações que o tornou, atualmente, um sistema robusto, complexo e dinâmico. Dentre suas particularidades, destaca-se a possível importação e exportação de informações para outros sistemas desenvolvidos pela Microsoft, como por exemplo, o pacote *Office.* Entre outros pontos, destaca-se o fácil acesso aos recursos do sistema, bem como a possibilidade de emissão de relatórios *web.* Apesar da grande aceitação por parte dos profissionais da área, o sistema não segue todas as diretrizes propostas pelo guia PMBoK, principalmente com relação ao gerenciamento de riscos e gerenciamento das partes interessadas, que são realizadas de maneira superficial.

1.4.2. NetProject [NetProject, 2018]

O NetProject é um sistema *web*, desenvolvido pela Seed Intelligence Company, que ganha destaque entre os profissionais, principalmente do mercado brasileiro. Um ponto de destaque do NetProject é o fato do desenvolvimento do sistema ter sido baseado nas boas práticas apresentadas pelo guia PMBoK. Com uma interface intuitiva e de fácil utilização, mostra-se um dos sistemas mais completos disponíveis atualmente no mercado. Como características importantes, o sistema apresenta planejamento automático por meio de Canvas, quadro Kanban, uma *Timeline* do Projeto para acompanhar seu ciclo de vida e a possibilidade de realizar as estimativas de tempo de cada atividade por hora.

1.4.3. Sistema de Apoio à Gerência de Projetos – SAGP

O SAGP é um sistema *web* desenvolvido no meio acadêmico por alunos do Grupo de Estudos e Pesquisas em Engenharia de Software - GEPES, da Universidade Estadual Paulista "Júlio de Mesquita Filho" - UNESP do campus de São José do Rio Preto. O objetivo do sistema é fornecer às organizações um ambiente confiável, intuitivo e simples para o desenvolvimento de projetos, seguindo as diretrizes propostas pelo guia PMBoK [Souza *et al.* 2011]. Desde sua criação o SAGP passa por atualizações e melhorias de áreas específicas, de modo a refinar a ferramenta e acompanhar as atualizações do PMBoK [Contessoto *et al*. 2016].

Por se tratar de uma ferramenta acadêmica, o SAGP está disponível para uso gratuito, podendo ser acessado por meio do link www.dcce.ibilce.unesp.br/sagp. Para cadastrar um usuário e uma senha é necessário apenas entrar em contato com o administrador do sistema, pelo e-mail allan.contessoto@unesp.br. O acesso ao sistema permite a utilização de todas as funções presentes no SAGP, bem como a possibilidade de fazer sugestões de melhorias.

Como o SAGP segue as diretrizes e boas práticas apresentadas pelo guia PMBoK, o sistema automatiza inúmeras ferramentas e técnicas propostas pelo guia, de modo a auxiliar os gerentes e equipes de desenvolvimento durante o ciclo de vida do projeto, independentemente da abordagem de desenvolvimento utilizada. Atualmente o SAGP contempla as 10 áreas do conhecimento apresentadas pelo guia PMBoK, sendo que o gerenciamento das partes interessadas é aplicado em conjunto com o gerenciamento das comunicações.

É importante mencionar que o SAGP apresenta também módulos complementares que visam contribuir, de modo específico, para a aplicação de conhecimentos auxiliares à gestão de projetos. Assim, o sistema disponibiliza a possibilidade de acesso ao módulo de SCRUM, que apoia a aplicação do método ágil de desenvolvimento de software, bem como ao módulo R3M-SD, que apoia o desenvolvimento da maturidade organizacional na aplicação da gestão de riscos em organizações que desenvolvem software.

1.5. Estudo de Caso: Sistema de Gestão Médica

Com o intuito de exercitar os principais conceitos das áreas do conhecimento discutidas nesse capítulo, o estudo de caso "Sistema de Gestão Médica" é desenvolvido ao longo dessa seção, utilizando como suporte o SAGP, apresentado na subseção 1.4.3. Com isso, também é possível observar as contribuições que uma ferramenta automatizada baseada no PMBoK pode oferecer à efetiva aplicação da gestão de projetos. O sistema descrito busca proporcionar uma melhor gestão das atividades realizadas em uma clínica médica, por meio de funções simples e objetivas.

Como discutido anteriormente, o gerenciamento de um projeto é realizado por meio da aplicação dos processos que compõem cada uma das 10 áreas do conhecimento. No entanto, a forma de execução de cada um dos processos, bem como a ordem em que são executados, dependem da experiência do gerente de projetos e da organização responsável pelo desenvolvimento do projeto. Além disso, fatores como a natureza do projeto, tempo e custo para o desenvolvimento, escopo, riscos envolvidos e tamanho da equipe de desenvolvimento, também influenciam como e em que ordem os processos são aplicados ao longo do ciclo de vida do projeto. Assim, para cada projeto a ser desenvolvido, o gerente de projetos analisa as melhores opções e aplica os processos necessários para que o sucesso seja alcançado dentro do prazo e custo determinados no início do projeto e com a qualidade esperada por todas as partes interessadas.

Para o estudo de caso em questão, um conjunto de processos representativos das 10 áreas do conhecimento é aplicado com o intuito de gerenciar as fases de inicialização, planejamento e parte da execução do processo de desenvolvimento do software. Assim, a gestão do referido projeto é iniciada a partir de processos envolvidos no gerenciamento de integração, gerenciamento das comunicações e do gerenciamento das partes interessadas, uma vez que as informações geradas nessas áreas do conhecimento são fundamentais para a correta aplicação das demais áreas.

Desse modo, faz-se necessária a aplicação dos seguintes processos do gerenciamento de integração: "Desenvolver o termo de abertura do projeto", no qual formalmente o gerente de projetos recebe a autorização para utilizar os recursos

disponíveis para o desenvolvimento do projeto, e "Desenvolver o plano de gerenciamento do projeto", o qual gera o plano do projeto que é utilizado por todos os colaboradores durante o ciclo de vida. Na Figura 1.1 é apresentado o termo de abertura definido para o projeto do estudo de caso junto ao SAGP. No termo de abertura são inseridas informações importantes para o gerenciamento do projeto, como o orçamento inicial previsto, a data de início, a data de término, o setor responsável pelo projeto e o escopo preliminar. Também é possível observar na Figura 1.1 que, ao cadastrar o termo de abertura do projeto no SAGP, o gerente do projeto tem a opção de empregar o *Personal Software Process* – PSP ao projeto, que é um processo de capacitação e monitoramento individual dos membros da equipe de desenvolvimento do projeto, o qual tem colaborado para o aumento significativo nos índices de sucesso de um projeto [Esteca, 2013].

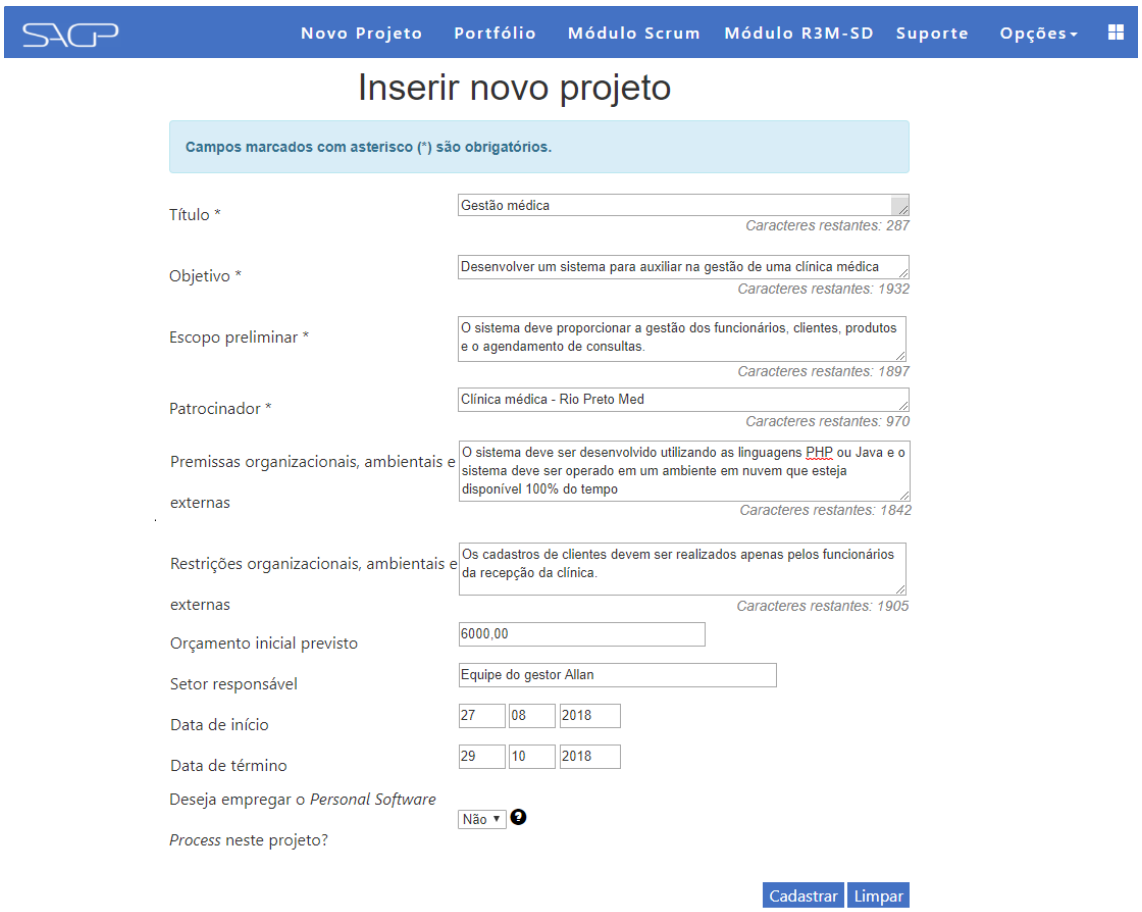


Figura 1.1 - Termo de abertura do estudo de caso utilizando o SAGP

Na aplicação do gerenciamento das partes interessadas, devem ocorrer os processos "Identificar as partes interessadas", "Planejar o engajamento das partes interessadas" e "Gerenciar o engajamento das partes interessadas", de modo a identificar e catalogar as características de cada uma das partes interessadas, planejar como cada uma será envolvida ao projeto e, por fim, como e quando a comunicação será realizada com as partes interessadas, de modo a manter o engajamento de todos.

Observa-se que a área de gerenciamento das partes interessadas está fortemente ligada ao gerenciamento das comunicações, uma vez que todo o engajamento das partes interessadas é mantido por meio de comunicações bem planejadas e executadas. Assim, também deve ser aplicado o processo “Planejar o gerenciamento das comunicações”, de modo a desenvolver o plano individual de comunicação com cada uma das partes interessadas, considerando suas particularidades.

Na Figura 1.2 é apresentado o plano de comunicações definido junto ao SAGP, o qual, para ser executado, necessita do cadastro prévio das partes interessadas. É possível observar na Figura 1.2 que os dados da parte interessada aparecem dentro da caixa “Ficha da parte interessada” e ao lado é possível inserir as informações do plano de comunicação daquela parte interessada, informando os requisitos para a comunicação, as tecnologias disponíveis para o contato e uma breve análise da comunicação com aquela pessoa. É importante mencionar que, após o cadastro de uma parte interessada e do seu plano de comunicação, o SAGP cria automaticamente a “Matriz de Influência x Interesse”, uma importante ferramenta para ambas áreas do conhecimento, uma vez que, por meio da matriz, é possível visualizar de forma simples as pessoas que necessitam de maior atenção do gerente de projetos.

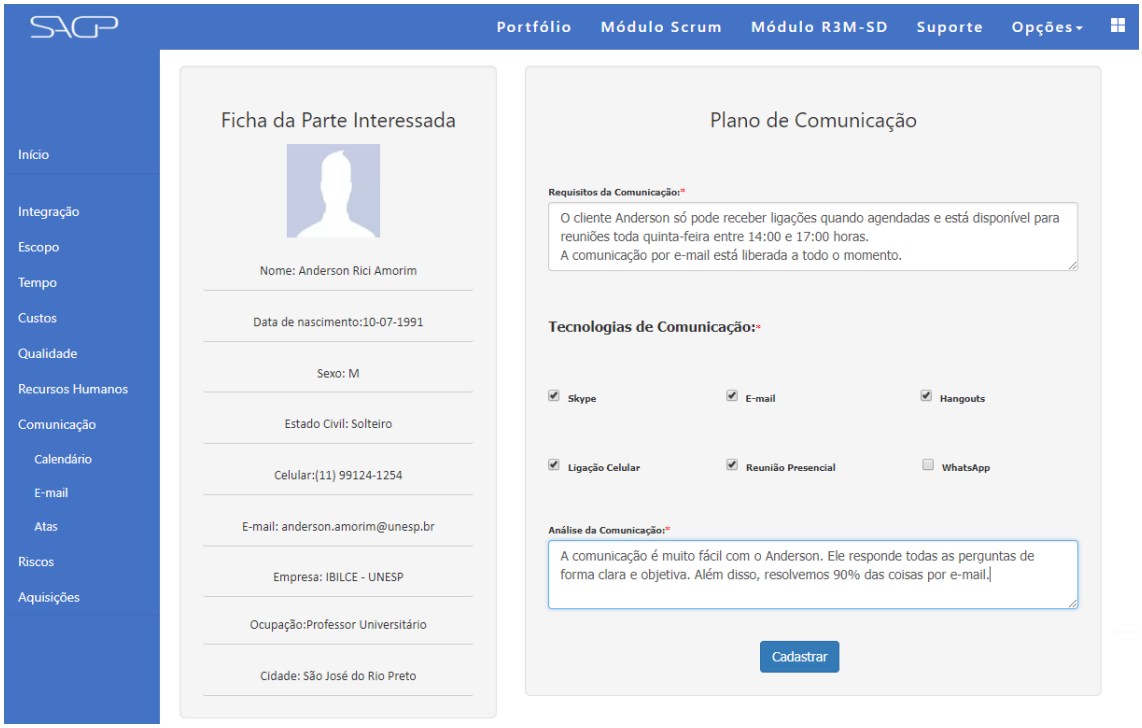


Figura 1.2 – Plano de comunicação criado junto ao SAGP para cada uma das partes interessadas

Na Figura 1.3 é apresentada a “Matriz de Influência x Interesse”, criada automaticamente após o cadastro da primeira parte interessada no projeto e a inserção do seu plano de comunicação.

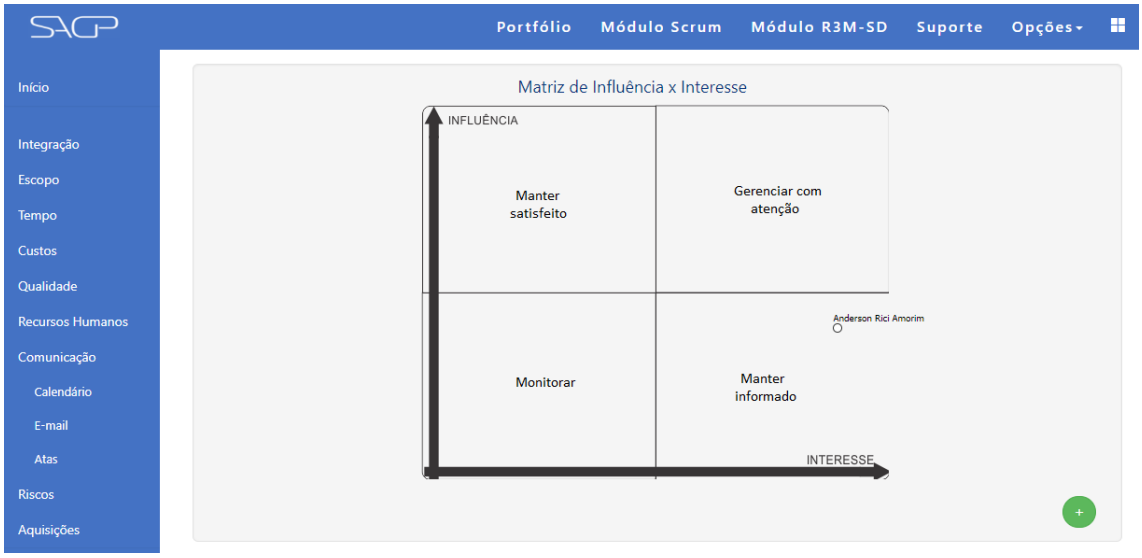


Figura 1.3 – Matriz de influência x interesse do SAGP

No estudo de caso, para o cliente cadastrado, observa-se que seu interesse no desenvolvimento do projeto é alto e sua influência sobre o projeto é moderada. Assim, esse cliente foi categorizado como uma parte interessada para "Manter informado" a respeito das decisões do projeto. As informações geradas na aplicação dos processos das três primeiras áreas do conhecimento formam a base necessária para as demais áreas.

A próxima etapa é a realização dos processos voltados para a gestão do escopo do projeto, os quais definem o trabalho a ser desenvolvido para alcançar o sucesso do projeto. Assim, inicialmente é aplicado o processo "Planejar o gerenciamento do escopo", com o intuito de definir de forma genérica o escopo do produto e o escopo do projeto e indicar como o escopo será validado e controlado ao longo do ciclo de vida do projeto. Para o estudo de caso, de forma geral é possível dizer que o escopo do produto é o software de Gestão Médica e o escopo do projeto são as atividades necessárias para desenvolver o software.

Para que ocorra a definição correta do escopo do produto e do projeto devem ocorrer os processos "Coletar os requisitos" e "Definir o escopo". Dessa forma, os requisitos do produto são corretamente analisados, documentados e validados pelas partes interessadas e, posteriormente, a equipe do projeto será capaz de realizar a definição final do escopo do produto e do projeto. Para projetos de desenvolvimento de software, a aplicação desses dois processos é de extrema importância, uma vez que um escopo mal definido do produto levará à construção de um software fora das especificações do cliente e um escopo mal definido do projeto levará a equipe de desenvolvimento a realizar atividades além das necessárias para o desenvolvimento do software. Na Figura 1.4 é apresentado o cadastro de um requisito no SAGP, função fundamental para a gestão do escopo. É relevante mencionar que o SAGP apresenta todo o suporte para o processo "Coletar os requisitos" utilizando os recursos da engenharia de requisitos [Lampa 2015].

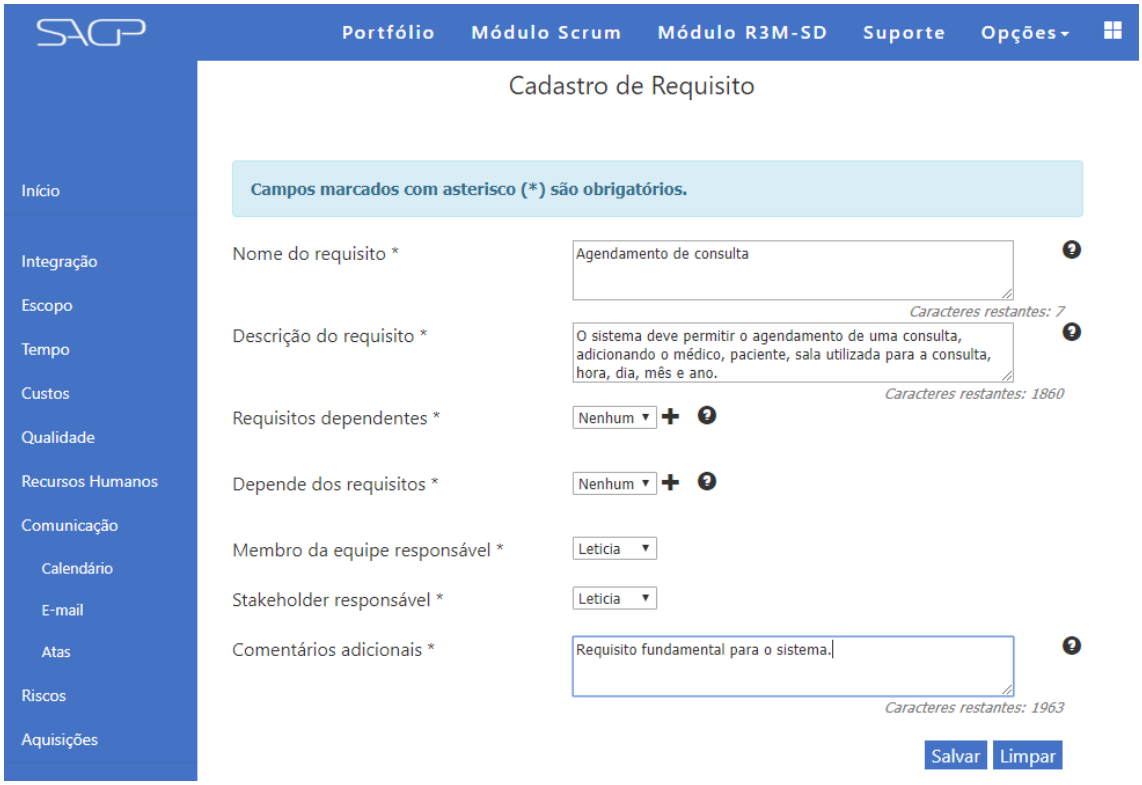


Figura 1.4 – Inserção de um requisito no SAGP.

A partir da aplicação dos processos “Coletar os requisitos” e “Definir o escopo”, a equipe de desenvolvimento é capaz de aplicar os demais processos do gerenciamento do escopo e, também, os processos do gerenciamento do cronograma do projeto. Assim, devem ser aplicados os processos “Planejar o gerenciamento do cronograma”, “Definir as atividades”, “Sequenciar as atividades”, “Estimar as durações das atividades” e “Desenvolver o cronograma”. A execução desses processos irá resultar na organização das atividades necessárias para o desenvolvimento do projeto, bem como nas informações relevantes de cada atividade, como tempo de duração, descrição, responsável e componentes do projeto afetados. O SAGP permite que a equipe de desenvolvimento consiga analisar a organização das atividades por meio de duas técnicas, o Gráfico de Gantt e o Caminho Crítico. Na Figura 1.5 é apresentado o Gráfico de Gantt criado no SAGP.

Após a definição das atividades, devem ocorrer o gerenciamento dos custos e o gerenciamento dos recursos, uma vez que essas duas áreas do conhecimento são responsáveis por estimar os recursos e os seus respectivos custos, necessários para que uma atividade seja concluída com sucesso.

Nesse sentido, os processos do gerenciamento dos recursos que devem ser aplicados são “Planejar o gerenciamento dos recursos”, “Estimar os recursos das atividades” e “Adquirir recursos”. Dessa forma, é possível identificar quais os recursos necessários para a realização das atividades e, também, quais recursos não estão disponíveis e devem ser providenciados para o projeto. Como exemplo de um recurso interno, tem-se a realocação de um membro chave da equipe de desenvolvimento de um projeto para outro, visando o cumprimento das atividades estabelecidas para o projeto considerado prioritário.

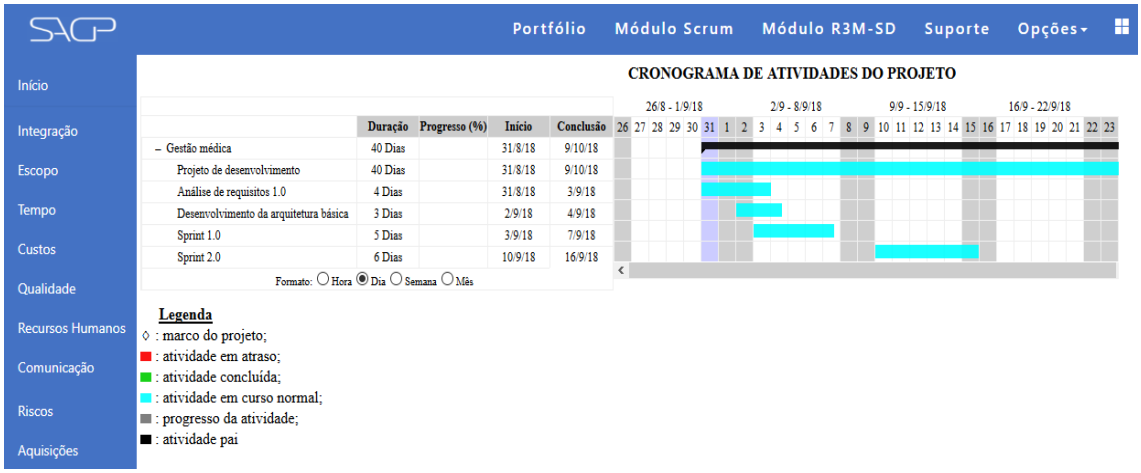


Figura 1.5 – Gráfico de Gantt do SAGP representando as atividades do estudo de caso

Do mesmo modo, os processos do gerenciamento dos custos aplicados são "Planejar o gerenciamento dos custos", "Estimar os custos" e "Determinar o orçamento". Dessa maneira, os custos monetários para que o projeto seja finalizado são estimados e o custo de cada uma das atividades ou grupo de atividades é determinado, de modo que a equipe de desenvolvimento tenha uma base dos custos autorizados para cada atividade. Determinar o custo de uma atividade pode ser uma tarefa complexa para a equipe de desenvolvimento de software, visto que inúmeros projetos tendem a seguir uma abordagem incremental no processo de desenvolvimento. Nesse sentido, os custos das atividades devem ser medidos a partir de unidades de trabalho que tenham um começo e um fim determinados, como por exemplo, um *Sprint* do *Scrum*. Na Figura 1.6 é apresentado o cadastro de um custo extra no SAGP, função relacionada ao processo "Controlar os custos".

Em paralelo à definição dos custos e dos recursos, deve ocorrer o gerenciamento das aquisições, uma vez que essa área do conhecimento é responsável pela gestão de compra de qualquer recurso que seja externo à organização e por gerenciar os contratos do projeto. Nesse sentido, os processos de "Planejar o gerenciamento das aquisições" e "Conduzir as aquisições" são aplicados, com o intuito de suprir qualquer demanda de aquisição que o projeto tenha, dando suporte ao gerente de projetos na tomada de decisão. Na Figura 1.6 é possível observar que será necessária uma visita ao cliente, com o intuito de sanar problemas com a análise de requisitos. Nesse sentido, o gerenciamento das aquisições irá atuar nas aquisições de serviços e produtos que garantam a ida do analista até o cliente, como por exemplo, compra de uma passagem de ônibus, pagamento de um serviço de táxi, entre outros.

Após, é aplicado o gerenciamento dos riscos com intuito de identificar e tratar os eventos incertos que podem ocorrer ao longo do ciclo de vida do projeto e que podem afetar os seus objetivos. É importante realçar que o gerenciamento dos riscos apresenta tratamentos semelhantes para os eventos que prejudicam os objetivos do projeto, as ameaças, e os eventos que são benéficos para os objetivos do projeto, as oportunidades. Nesse sentido, devem ser aplicados os processos "Planejar o gerenciamento dos riscos", "Identificar os riscos", "Realizar a análise qualitativa dos riscos", "Realizar a análise quantitativa dos riscos" e "Planejar as respostas aos riscos".

Figura 1.6 – Cadastro de custo extra no SAGP

Por meio da aplicação desses processos, já nas fases iniciais do projeto, a equipe de desenvolvimento é capaz de analisar todo o cenário dos riscos, gerando informações fundamentais como as atividades que apresentam mais riscos, causa-raiz comum entre os riscos e resposta comum entre os riscos. Uma importante forma de visualização de informações é a matriz de probabilidade e impacto, que tem o intuito, dentre outras coisas, de mostrar pontos de concentração de riscos e que merecem uma revisão da equipe de desenvolvimento. Na Figura 1.7 é apresentada a matriz de probabilidade e impacto gerada pelo SAGP para o estudo de caso após o cadastro de 13 ameaças e 7 oportunidades. As cores na matriz representam a prioridade que a equipe de desenvolvimento deve ter para tratar os riscos, sendo que riscos nas áreas vermelhas têm prioridade máxima, nas áreas amarelas têm prioridade média e riscos nas áreas verdes têm prioridade mínima.

Ao longo de todo o ciclo de vida de um projeto, a equipe de desenvolvimento busca realizar atividades que resultem no software final, cumprindo os objetivos e zelando pela qualidade. Assim, para buscar garantir a qualidade no software, a equipe de desenvolvimento aplica os processos do gerenciamento da qualidade ao longo de todo o projeto. Os processos “Planejar o gerenciamento da qualidade”, “Gerenciar a qualidade” e “Controlar a qualidade” devem ser aplicados desde a concepção do projeto, de modo a serem criadas atividades que são executadas para incorporar políticas de qualidade ao projeto. Para o estudo de caso, um exemplo de aplicação do gerenciamento de qualidade seria a realização de análise de erros que o software apresenta, pois isso poderia resultar no estabelecimento de uma forma de diminuir os erros, como promover um curso para os programadores se atualizarem na tecnologia que estão utilizando.

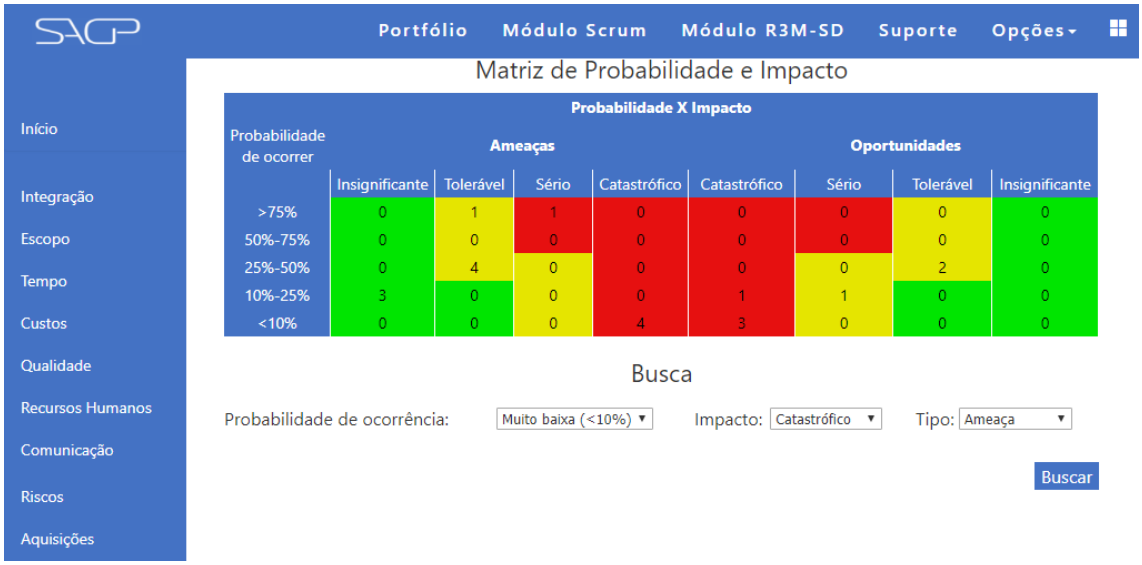


Figura 1.7 – Matriz de probabilidade e impacto no SAGP

Diante do exposto, observa-se que o estudo de caso foi conduzido de modo que a iniciação, o planejamento e parte da execução de um projeto de desenvolvimento de software fossem concluídos por meio da realização de um conjunto de processos envolvidos nas 10 áreas do conhecimento. A partir disso, os processos de execução, monitoramento e controle e encerramento podem ser conduzidos para gerar o sistema proposto no estudo de caso.

1.6. Conclusão

O gerenciamento de projetos é uma disciplina fundamental nos tempos atuais, independentemente do setor da economia. Para organizações que desenvolvem software, a dependência da gestão de projetos está ficando cada vez maior, visto que os projetos estão cada vez mais complexos de serem executados, o cronograma de desenvolvimento está cada vez mais curto e as tecnologias estão em um ritmo constante de evolução e desenvolvimento. Assim, é muito importante para organizações que desenvolvem software terem profissionais com conhecimento técnico a respeito da gestão de projetos, bem como terem o suporte de ferramental computacional que seja eficiente e confiável.

O mercado de ferramentas computacionais de suporte à gestão de projetos está em constante evolução, sendo importante que a organização tenha maturidade para escolher a ferramenta que melhor atenda suas necessidades. Apesar de muitas ferramentas estarem disponíveis no mercado, poucas estão em conformidade com as diretrizes e boas práticas apresentadas pelo guia PMBoK, sendo mais um ponto a ser analisado durante a escolha de uma ferramenta.

O SAGP é uma ferramenta computacional de destaque, pois proporciona auxílio para a aplicação da gestão de projetos de forma confiável e eficiente, seguindo as diretrizes do PMBoK. Além disso, o sistema apresenta módulos complementares específicos para organizações que desenvolvem software, de modo a auxiliar essas organizações no desenvolvimento de software com melhor qualidade.

Referências

Alexander, I.; Beus-Dukic, L. Discovering Requirements: how to specify products and services. 1ª ed. West Sussex: John Wiley, 2009.

Azevedo, R. R. (2017) "Gerenciamento eficiente de comunicação em projeto". Monografia - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Contessoto, A. G. (2015) "Gerenciamento dos riscos de um projeto: o caminho para o sucesso". Monografia - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Contessoto, A. G.; Sant'Ana, L. A.; Souza, R. C. G.; Valêncio, C. R.; Donegá, G. F.; Amorim, A. R.; Esteca, A. M. N. (2016) "Improving risk identification process in project management", In: Proceedings of the 28th International Conference on Software Engineering and Knowledge Engineering (SEKE), São Francisco Bay, Estados Unidos da América, p. 555-558.

Contessoto, A. G. (2017) "Maturidade em gerenciamento de riscos em projetos de software". Dissertação - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Bomfin, D. F.; Nunes, P. C. Á.; Hastenreiter, F. (2012) "Gerenciamento de projetos segundo o guia PMBoK: desafios para os gestores". Revista de Gestão e Projetos – GeP, v.3, n.3, p. 58-87.

Degrossi, L. C. (2011) "Automatização das atividades de gestão de escopo em um ambiente de gerência de projetos". Monografia - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Esteca, A. M. N. (2010) "Gerência de projetos: apoio automatizado para efetivação das atividades". Monografia – Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Esteca, A. M. N. (2013) "Apoio à maturidade pessoal visando a melhoria dos projetos de software". Dissertação - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Lampa, I. L. (2015) "Gestão do escopo de projetos: uma estratégia orientada à Engenharia de Requisitos". Monografia - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Laruccia, M. M.; Ignez, P. C.; Deghi, G. J.; Garcia, M. G. (2012) "Gerenciamento de projetos em pesquisa e desenvolvimento". Revista de Gestão e Projetos – GeP, v.3, n.3, p. 109-135.

Maretti, V.; Júnior, P. A.; Costa. H. (2016) "Uma revisão Sistemática da Literatura sobre Comunicação no Contexto da Gerência de Projetos de Sistemas de Informação". In: XII Brazilian Symposium on Information Systems, Florianópolis, SC, Brasil, p. 84-91.

Mattos, M. P. P.; Ferreira, A. S.; Oliveira, D. C.; Resende, R. (2012) "Gerenciamento de projetos: Uma análise da gestão de riscos em um projeto de construção e montagem em uma empresa petrolífera". In: IX Congresso Virtual Brasileiro de Administração, Brasil.

MSPROJECT. (2018) "Microsoft Project 2013", http://office.microsoft.com/pt-br/project, Março.

NETPROJECT. (2018) "NetProject", http://www.netproject.com.br, Março.

Oliveira, R. C. F. (2003) "Gerenciamento de projetos e a aplicação da análise de valor agregado em grandes projetos". Dissertação – Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil.

Pressman, R. S. "Engenharia de software: Uma abordagem profissional", 7ª ed. Artmed, 2011.

Project Management Institute – PMI. "Guide of Project Management Body of Knowledge – PMBoK", 6ª edição, 2017.

Project Management Institute – PMI - Agile Alliance. "Agile Practice Guide", 1ª edição, 2017.

Salim, G. A. (2013) "O módulo de gerenciamento de custos em projetos integrado ao SAGP". Monografia - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Sant' Ana, L. A. (2015) "Potencialização das informações para gerência dos riscos em projetos". Monografia - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, Brasil.

Sommerville, I. "Engenharia de Software". 9ª ed. Pearson Addison-Wesley, 2011.

Souza, R. C. G.; Esteca, A. M. N.; Santos A. B.; Valêncio C. R.; Honda, M. T. (2011) "Web System to Aid Project Management", In: Proceedings of the 23th International Conference on Software Engineering and Knowledge Engineering (SEKE), Estados Unidos da América, p. 325-330.

3. Capítulo 3

Autores:

Camilo Barreto
Universidade Federal de Uberlândia (UFU)
email: barretojuniormail@gmail.com

Alexandre Carvalho
Universidade Federal de Uberlândia (UFU)
email: acs.carvalho10@gmail.com

Alexandre Cardoso
Universidade Federal de Uberlândia (UFU)
email: alex.cardoso.ufu@gmail.com

Edgard Lamounier
Universidade Federal de Uberlândia (UFU)
email: elamounier@gmail.com

Capítulo 3

Técnicas para Criação de Aplicativos Mobile com Suporte à Vídeos e Imagens em 360°

Camilo Barreto, Alexandre Carvalho, Alexandre Cardoso, Edgard Lamounier

Abstract

The use of mobile devices in various areas of human knowledge is increasingly frequent. In the face of the scenario of massification of these devices in the individuals' daily lives, the form of interaction became more dynamic and enriched. Under this perception, several strategies focused on mobility have surfaced, in this sense it is possible to highlight the use of videos and images in 360º in mobile applications, so, with the use of this technique it is possible to maximize the sensation of immersion of users with presented content. Therefore, in view of this horizon of potentialities presented, this proposal aims to present in a theoretical and practical way all the necessary steps for the creation of applications for mobile platform with support for viewing video content and images in 360º.

Resumo

A utilização de dispositivos móveis em diversas áreas de atuação do conhecimento humano é cada vez mais frequente. Diante do cenário de massificação destes dispositivos no cotidiano dos indivíduos, a forma de interação tornou-se mais dinâmica e enriquecida. Sob essa percepção diversas estratégias centradas na mobilidade vieram à tona e, neste sentido é possível salientar a utilização de vídeos e imagens em 360º nas aplicações mobile. Com a utilização desta técnica, é possível aprimorar a sensação de imersão dos usuários com conteúdo apresentado. Este capítulo trata das possibilidades de concepção de conteúdos de vídeos e imagens em 360º, por meio de aspectos teóricos e aplicações práticas.

1.1. Introdução

Equipamentos para captura de imagens em 360° foram por muito tempo inacessíveis aos usuários, por conta de custos proibitivos e pouca difusão de conteúdo e de aplicações para os resultados. Nos últimos anos, este cenário se modificou, pela rápida redução de custos das câmeras e demanda dos usuários por acesso a tais conteúdos.

Associado a isto, há crescente interesse em implantar a Realidade Virtual e aplicações de Realidade Aumentada, catalisando a demanda por vídeos em 360 graus como uma nova maneira de fornecer experiência de visualização imersiva, permitindo que usuários mudem interativamente seu ponto de vista e visualizar qualquer parte do conteúdo capturado.

Utilizando câmeras dispositivos de captura em 360°, é possível reproduzir em dispositivos computacionais, ambientes por meio de imagens, possibilitando a visualização detalhada dos mesmos por distintos ângulos e pontos de interesse.

Contextualizando possíveis aplicações, tal tecnologia é aderente à promoção de passeios virtuais, culturais e acesso a pontos turísticos e históricos. No campo das Engenharias, imagens em 360° são utilizadas para realizar mapeamentos de áreas, estruturas e edifícios com intuito de gerar modelos virtuais tridimensionais. Um exemplo prático é o mapeamento de estruturas (como pontes suspensas) por meio de um *drone* equipado com uma câmera 360°, as imagens capturadas podem ser processadas por um *software* específico e assim gerar modelos 3D no qual projetos de recuperação e manutenção podem ser planejados.

Sendo assim, esse minicurso tem como objetivo realizar fundamentações referentes à adequações e desenvolvimento associado ao uso de aplicativos mobile com suporte a técnicas de visualização de vídeos e imagens em 360º. Serão evidenciadas as etapas associadas aos procedimentos de coleta e adequação do material (imagem e/ou vídeo), implementação dos mecanismos de visualização e interação, tendo como alvos as plataformas *Android*, *Windows* e *Web*.

1.2. Metodologia

1.2.1. Procedimentos de Coleta de Imagens

Nessa seção serão apresentadas algumas convenções para coleta de materiais oriundos de dispositivos de captura. Também será abordado o uso de dispositivos para captura, como câmeras 360° comerciais e dispositivos móveis.

O procedimento para coleta de imagens é a primeira etapa do desenvolvimento de uma aplicação com suporte a imagens e vídeos em 360°. Levando em consideração a captura de imagens com qualidade, é necessário utilizar equipamentos adequados e seguir alguns procedimentos para que os resultados sejam compatíveis.

Para tais propósitos, o uso de equipamentos adequados, dotados de câmeras 360° são essenciais. Para propiciar a captura, os seguintes equipamentos são demandados:

1. **Tripé**: é um equipamento importante para auxiliar o posicionamento e estabilidade. Com o tripé é possível posicionar a câmera à uma altura adequada, garantindo que a imagem não fique próxima do piso e assemelhando-se à altura

de uma pessoa. Alguns tripés podem ser eliminados por processamento gráfico interno nas câmeras ou por meio de *softwares* externos. No caso de processamento interno os tripés menores podem ser eliminados da imagem totalmente e os maiores podem ser eliminados parcialmente, advém de quão próximos as partes sobressalentes estão da câmera.

2. **Controle Remoto**: caso não queira ser notado na imagem, um controle remoto pode ser utilizado. Ao ausentar-se do campo de visão, basta acionar a captura pelo controle remoto. Ou se preferir, pode-se utilizar o *timer* da câmera.
3. **Bastão de *Selfie***: semelhante ao tripé, o bastão de *selfie* é um suporte que facilita a captura de imagens. Tal equipamento não é fixado ao piso, é segurado pelo usuário. Algumas câmeras conseguem eliminar parcialmente o bastão da imagem.
4. **Cartão de Memória Rápido**: é mais utilizado na captura dos vídeos com alta resolução como Full HD e 4K. Como a quantidade de transferência de dados é grande para vídeos, é necessário um cartão de memória rápido para evitar congestionamento no momento da gravação.

Algumas práticas no ato da captura podem melhorar consideravelmente a qualidade das imagens, como:

1. **Enquadramento**: o posicionamento da câmera 360° é fundamental para que alguns elementos do cenário não apareçam demasiadamente distantes ou próximos. A melhor posição para capturar as imagens, quando possível, é no centro do ambiente ou próximo do alvo, porém há de se atentar o quão próximo a câmera será posicionada. As lentes das câmeras 360° possuem ângulos abertos, e quando as imagens são vistas em uma tela de computador ou *smartphone* os objetos podem parecer distantes, porém, muito próximos utilizando dispositivos HMD (*Head-Mounted Display*). Outro aspecto importante é a captura em lugares fechados, posicione mais distante possível de paredes, pilastras e objetos que podem ocluir coisas importantes.
2. **Nivelamento**: um bom nivelamento garante que a imagem não será observada com erro no ângulo horizontal e vertical, isso pode prejudicar a experiência do usuário ao ver uma imagem que não condiz com o real. Há equipamentos que realizam o nivelamento horizontal e vertical das câmeras. Em vários tipos de tripés existem o nivelamento digital ou por bolha.
3. **Iluminação**: durante o dia ambientes abertos podem apresentar uma boa iluminação, porém dependendo da intensidade do sol a imagem pode ser ofuscada pelo brilho causado por um *flare* nas lentes da câmera. Durante a noite a iluminação pode ser muito baixa e ocasionar uma imagem escura, dificultando a visualização da informação. Sendo assim, é necessário encontrar um ponto de equilíbrio entre ambiente muito claro e muito escuro.

1.2.2. Tipos de Câmeras 360°

Atualmente, encontram-se disponíveis no mercado, grande gama de câmeras de captura em 360°, o Quadro 1 apresenta algumas das principais câmeras comerciais e suas características.

Quadro 1. Câmeras 360° e suas características.

 **Figura 1.1. Câmera Ricoh Theta S (RICOH, 2018).**	Resolução de imagem: 5376x2688 Resolução de vídeo: 3840x1920/30fps/56Mbps 1920x960/30fps/16Mbps Memória interna: 19GB Wi-Fi: a/b/g/n/ac (2.4Ghz e 5Ghz)
 **Figura 1.2. Samsung Gear 360° (SAMSUNG, 2017).**	Resolução de imagem: 5472x2736 Resolução de vídeo: 4096x2048/24fps 2560x1280/30fps Memória: MicroSD (até 256Gb) Wi-Fi: a/b/g/n/ac (2.4Ghz e 5Ghz)
 **Figura 1.3. Bubl Câmera 360° (BUBL, 2018).**	Resolução de imagem: 5376x2688 (px) Resolução de vídeo: 1984x992(px)/30fps 2688x1344(px)/15fps Memória: MicroSD (até 32Gb) Wi-Fi: n/ (2.4Ghz)
 **Figura 1.4. Mi Sphere Camera (XIAOMI, 2018).**	Resolução de imagem: 6912x3456(px) Resolução de vídeo: 3456x1728(px)/30fps 2304x1152(px)/60fps Memória: MicroSD (até 128Gb) Wi-Fi: b/g/n/ (2.4Ghz)
 **Figura 1.5. Kodak SP360 (KODAK, 2018).**	Resolução de imagem: 2880x2880(px) Resolução de vídeo: 3840x2880(px)/30fps 1280x720(px)/60fps Memória: MicroSD (até 128Gb) Wi-Fi: b/g/n/ (2.4Ghz)

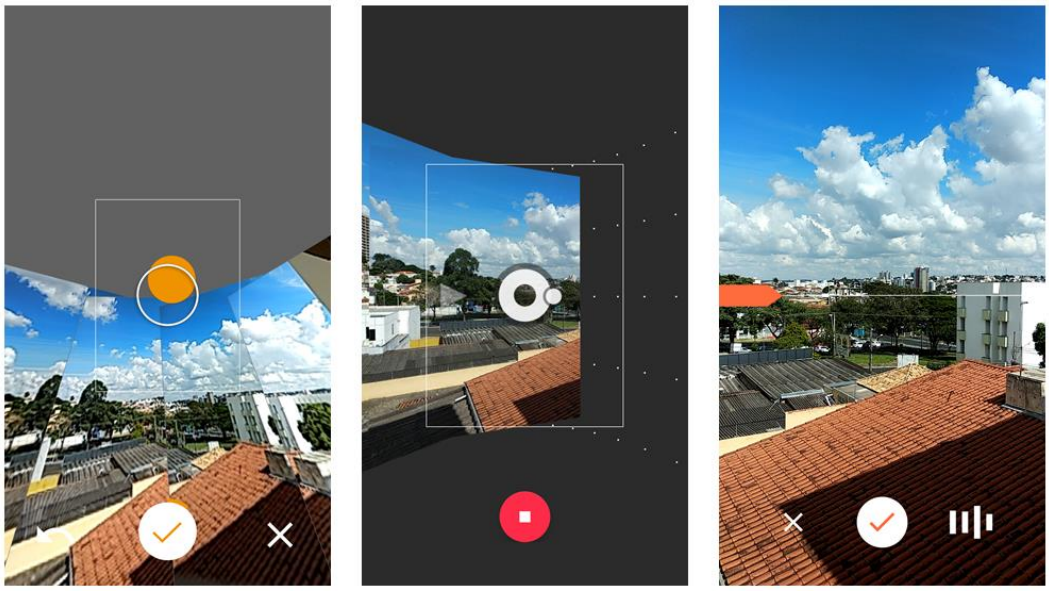
1.2.3. Captura de Imagens em 360° por Dispositivos Móveis

Outra forma de capturar imagens em 360° consiste na utilização de dispositivos móveis equipados com câmera. Há diversos aplicativos capazes de fotografar o ambiente e gerar uma imagem esférica ou panorâmica em 360°. Neste contexto, destacam-se três softwares, não nativos, para dispositivos com sistema operacional *Android* e iOS.

O *Google Street View*[1] é um ótimo aplicativo para captura de imagens em 360° completas, isto significa que é possível gerar uma imagem com todos os ângulos possíveis. O *software* auxilia o usuário a posicionar corretamente o *smartphone* em relação ao nivelamento horizontal e vertical, permitindo melhor qualidade no momento do processamento de junção de malhas. A Figura 1.6 (a) mostra o processo de captura de imagens utilizando sistema de guia.

Outro aplicativo é o *Google Camera Cardboard*[2], permite ao usuário capturar imagens panorâmicas em 360°. As imagens panorâmicas não são imagens completas, isto significa que quando se tem uma panorâmica, não foi capturado a parte superior e inferior do ambiente. A Figura 1.6 (c) mostra o processo de captura.

Por fim, o aplicativo *Panorama 360 Camera*[3] realiza captura de imagens panorâmicas em 360°. É similar ao *Camera Cardboard*, utiliza um sistema de guia de nivelamento para o *smartphone*. A Figura 1.6 (b) exemplifica como é o processo de captura.



(a) (b) (c)

Figura 1.6. Aplicativos de captura em 360° e panorâmica, (a) Google StreetView, (b) Panorama Camera 360, (c) Google Camera Cardboard

[1] https://goo.gl/sseZz4

[2] https://goo.gl/vnHuDn

[3] https://goo.gl/brzP9b

1.3. Desenvolvimento

Nessa seção serão apresentadas as etapas de desenvolvimento do aplicativo com suporte a imagens e vídeos em 360°. Serão abordados métodos e estratégias de criação do sistema de projeção, organização do projeto dentro da *Game Engine Unity*[4] e visualização. Também, apresentar métodos e ferramentas para preparação e adequação dos materiais oriundos da coleta de imagens. Será abordado quais características de uma imagem ou vídeo deve conter para serem utilizados no aplicativo 360°, posteriormente os *softwares* que podem ser utilizados para tal e por fim os arquivos de exportação.

Assim sendo, essa seção foi dividida em partes. Primeiramente o projeto *Unity* será preparado e organizado para a implementação, posteriormente será desenvolvido o mecanismo de projeção de imagens, e finalmente a implementação de scripts para visualização do conteúdo.

Destaca-se a necessidade de evidenciar o que é preciso realizar antes da etapa de desenvolvimento. Nas seções anteriores foram abordados os passos para coleta de imagens por meio de dispositivos específicos e a adequação do material para ser utilizado no desenvolvimento da aplicação. A Figura 1.7 exemplifica os processos recomendados para produzir um aplicativo suportado por imagens e vídeos em 360°.

Na próxima seção será abordado o desenvolvimento da camada de visualização, possuindo como plataformas dispositivos móveis com sistema operacional *Android* e sistemas operacionais *Windows*.

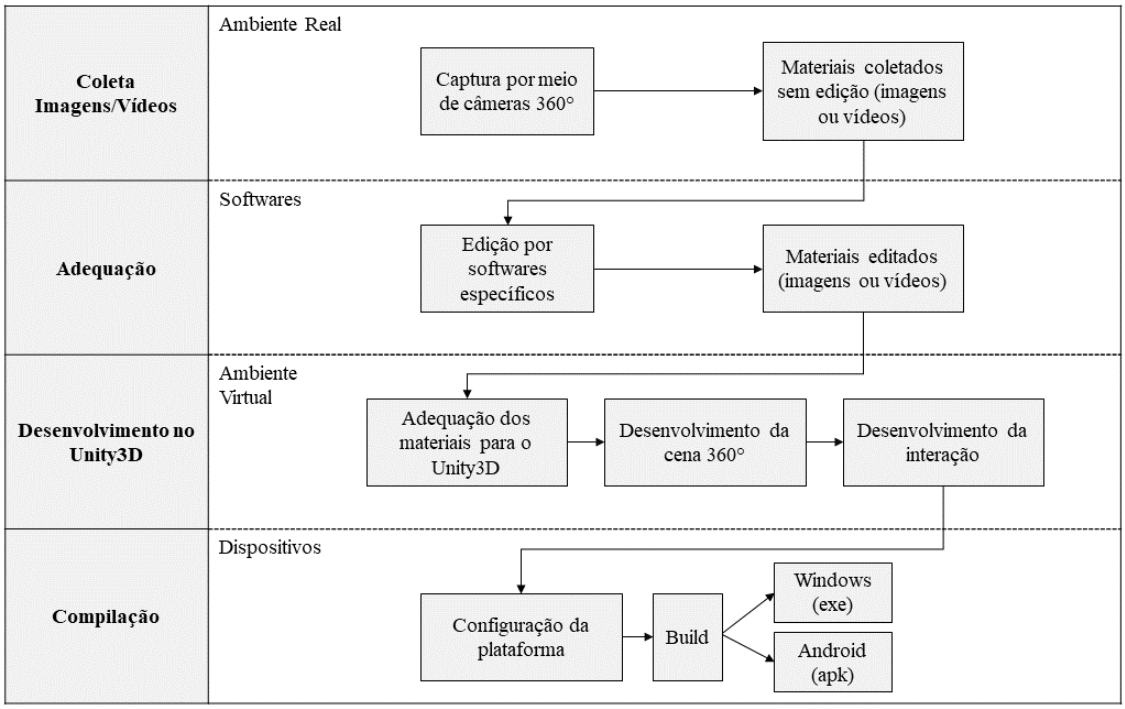


Figura 1.7. Etapas do processo de desenvolvimento da aplicação com suporte a imagens e vídeos

[4] https://unity3d.com

1.3.1. Tipos de Materiais Gerados pela Coleta

O processo de coleta de imagens gera distintos tipos de materiais, tanto para vídeos quanto para imagens. Nesta subseção serão abordados os tipos de materiais gerados por câmeras 360°. Na próxima subseção (1.3.2) será apresentado as estratégias de preparação e adequação dos materiais coletados para serem utilizados no desenvolvimento com o Unity3D.

No contexto de captura de imagens 360º, tratar-se-á sobre as imagens panorâmicas e esféricas em 360°. A imagem panorâmica constitui-se apenas da captura do horizonte, ignorando partes superiores e inferiores do ambiente. Esse tipo de imagem geralmente é feito por dispositivos móveis utilizando aplicativos como o *Camera Cardboard* e *Panorama 360 Camera*. Exemplo na Figura 1.8 (a).

As imagens esféricas, em 360°, são feitas por dispositivos dedicados, como as câmeras apresentadas na subseção anterior. As imagens esféricas contêm todas as partes do ambiente, isto é, toda área ao redor da câmera é capturada, horizonte e partes superiores e inferiores. Sendo assim é gerado uma malha esférica completa, a Figura 1.8 (b) mostra uma imagem esférica em 360°.

(a) (b)

Figura 1.8. Tipos de imagens capturadas em 360°: (a) imagem panorâmica; (b) imagem esférica 360°

As imagens panorâmicas 360° geralmente possuem o *Aspect Ratio* de 4:1 e 5:1, isso significa que possuem largura 4 a 5 vezes maior que a altura. A Figura 1.9 mostra um exemplo de captura panorâmica.

Figura 1.9. Imagem 360° Panorâmica.

Em contrapartida, as imagens esféricas possuem o *Aspect Ratio* de 2:1. Exibidas em uma tela parecem destorcidas, isso é pelo fato de não estarem com a malha em formato esférico.

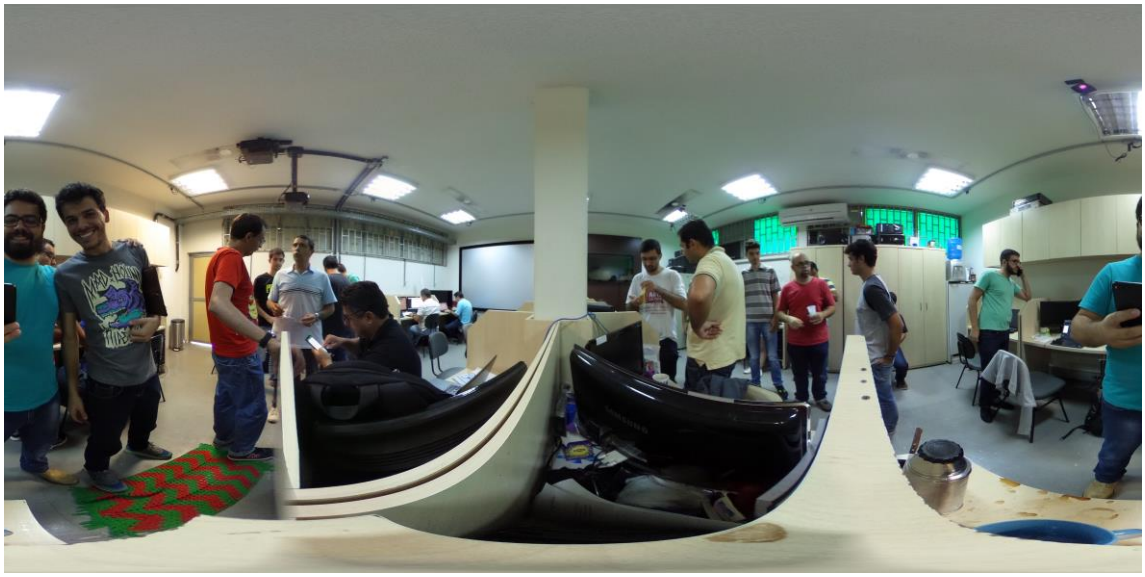


Figura 1.9. Imagem esférica 360°

Os materiais gerados pela captura de vídeos são dados brutos que necessitam de edição. Os vídeos possuem formatos relacionados ao tipo de câmera, no caso da Ricoh Theta o formato de vídeo é MP4 com *Aspect Ratio* de 2:1 e possui duas imagens esféricas no vídeo de seus sensores. Semelhante a Ricoh Theta a Samsung Gear 360 também possui o formato com duas esferas no vídeo. A Figura 1.10 mostra como é um vídeo sem edição.

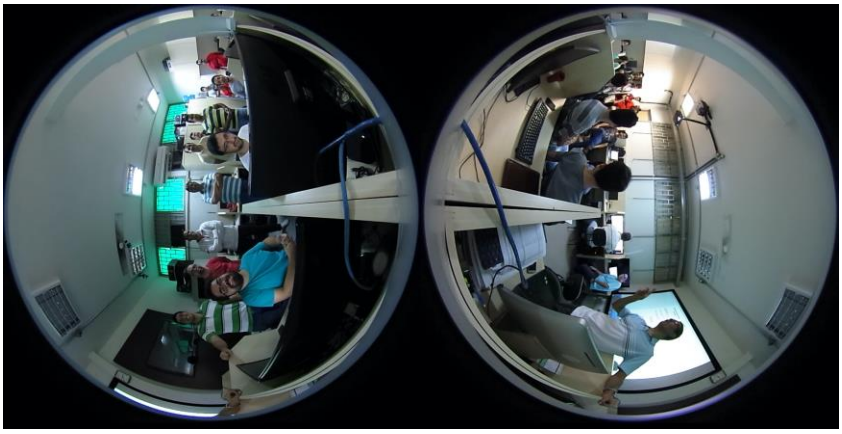


Figura 1.10. Formato de vídeo capturado pela Câmera Ricoh Theta

Sendo assim, é possível constatar que cada câmera gera formatos e estilos de imagens e vídeos distintos. Para que os materiais possam ser utilizados em um projeto de desenvolvimento de aplicativos, com uso da Unity3D, é necessário passar por adequações, isto é, realizar edições para adaptar à plataforma alvo. Na próxima subseção, será apresentado os métodos e estratégias para preparação dos materiais.

1.3.2. Adequação da Imagem e Vídeo em 360°

Antes de dar início ao desenvolvimento do ambiente em 360° é necessário verificar alguns atributos da imagem e vídeo. Como foi mencionado anteriormente, cada câmera produz materiais com distintos tipos e estilos, sendo assim, o primeiro aspecto a ser verificado é o tamanho do quadro. Um vídeo ou imagem em 360° pode manter qualquer resolução, como HD (1280px x 720px), Full HD (1920px x 1080px), 4K (4096px x 2160px) e 8K (7680px x 4320px), porém deve-se manter o *Aspect Ratio* (proporção da tela). Para esse trabalho a proporção deve ser mantida em 2:1, isto é, a largura dever ser 2 vezes a altura da imagem/vídeo.

Para calcular o *Aspect Ratio* pode-se utilizar a seguinte equação:

$$Aspect\ Ratio = \frac{Largura}{Altura} = \frac{5376}{2688} = \frac{2}{1}$$

A equação acima exemplifica como foi encontrado o tamanho do quadro para uma imagem de resolução 5376x2688 pixels. Para simplificar a divisão entre largura e altura pode-se utilizar o Máximo Divisor Comum (que é 2688) e dividir a largura e altura por ele, neste caso foi encontrado:

$$Aspect\ Ratio = \frac{5376}{2688} \frac{\div 2688}{\div 2688} = \frac{2}{1}$$

Após encontrar o *Aspect Ratio* a imagem/vídeo esférica deve ser editada para se adequar ao tamanho de quadro necessário para imagens em 360°. A Figura 1.11 mostra um quadro com proporções de 2:1.

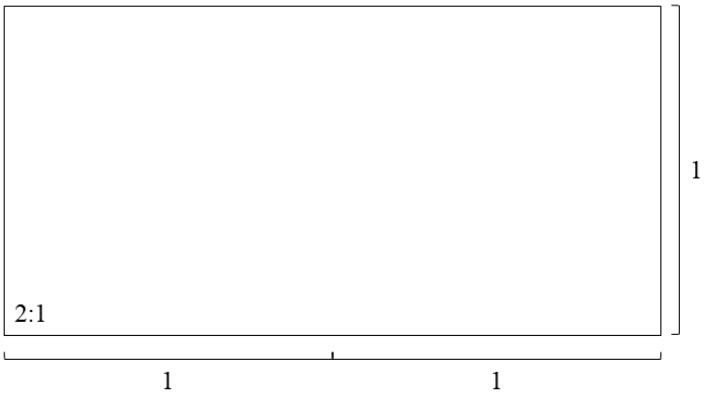


Figura 1.11. Quadro com *Aspect Ratio* de 2:1

Em caso de imagens 360° panorâmicas, é necessário adequar a imagem no tamanho do quadro especificado. Assim, para criar uma imagem 360° com resolução de 9348x1624 pixels (como a da figura 1.12), é necessário realizar cálculos para saber qual será a resolução final utilizando um *Aspect Ratio* 2:1.

Figura 1.12. Fotografia panorâmica 360°

Sendo assim, a largura da imagem é de 9348 pixels, para se encontrar a altura atualizada para a imagem pode-se utilizar a equação a seguir:

$$Altura = \frac{Largura}{2} = \frac{9348}{2} = 4674$$

Aplicando a largura da imagem na equação, obtém-se o valor de 4674 pixels. O próximo passo é criar uma nova imagem com as resoluções encontradas, de 9348px x 4674px pixels e *Aspect Ratio* de 2:1. A Figura 1.13 mostra a criação de uma nova imagem com as resoluções atualizadas. A imagem panorâmica original deve ser centralizada horizontalmente e os espaços remanescentes podem ser coloridos ou texturizados.

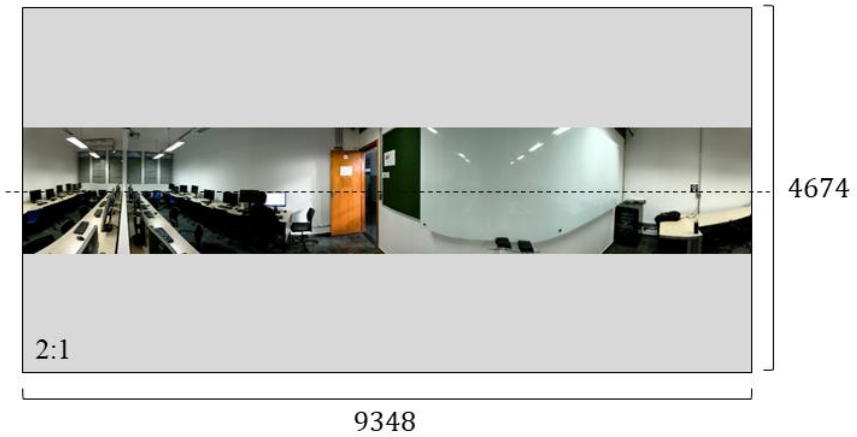


Figure 1.13. Adequação de uma imagem panorâmica para *Aspect Ratio* de 2:1

Um aspecto importante para melhorar a qualidade de visualização da imagem é verificar se o contraste e o brilho estão adequados. Uma imagem ou vídeo inadequado é quando seus quadros apresentam alta ou baixo brilho e contraste. Sendo assim é necessário editar a imagem e aplicar níveis próximos ao ideal, na próxima subseção será apresentado ferramentas que facilitam a edição de imagens. A Figura 1.14 exemplifica dois tipos de imagens com altos e baixos níveis de brilho e contraste, e por fim uma imagem com tonalidades próximas ao ideal.

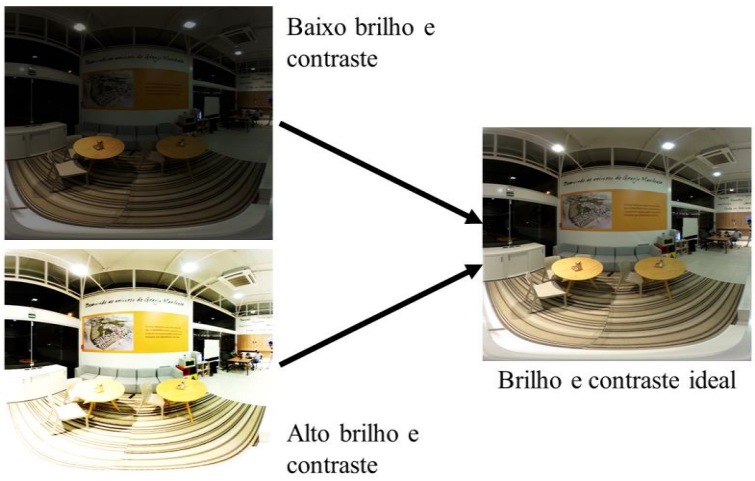


Figura 1.14. Imagens com problemas de alto e baixo contraste e brilho, em seguida uma imagem com níveis ideais

Os vídeos gerados pelas câmeras estão inicialmente em um estado bruto de informação, o vídeo, possui duas esferas oriundas dos dois sensores de captura de imagens. Não há um *player* de vídeo 360° que faça a conversão em tempo real para execução, sendo assim é necessário editar por meio de *softwares* específicos o vídeo no estado bruto para um formato adequado, para posteriormente ser integrado com outros *softwares*, inclusive o *Unity*. A resolução do vídeo depende das características dos sensores da câmera e o *Aspect Ratio* deve ser 2:1. A Figura 1.15 mostra o formato bruto de um vídeo e sua conversão. Na subseção seguinte, será abordado os *softwares* para edição dos vídeos e imagens.

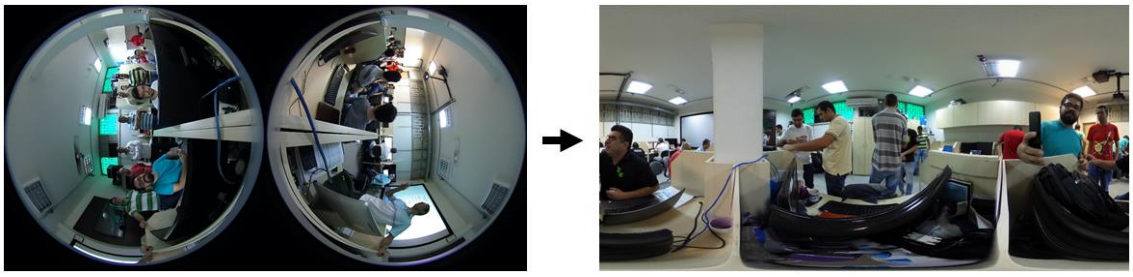


Figura 1.15. Conversão de vídeo bruto 360° para formato compatível com outros *softwares*

1.3.3. Softwares para Adequação de Materiais

Nessa subseção será apresentado os *softwares* e métodos de edição de imagens e vídeos. Primeiramente será utilizado o *software* Gimp 2 para edição das imagens esféricas e panorâmicas em 360°.

O Gimp[5] é um editor de imagens *Open Source* poderoso. Mesmo contendo muitas ferramentas de edição, será utilizado apenas ferramentas simples de tratar.

[5] https://www.gimp.org/

Edição de Brilho e Contraste

Ao constatar que é necessário editar o brilho e contraste, utilizar-se-á a ferramenta “Brilho e Contraste” para correções e ajustes. Primeiramente, deve-se abrir a imagem no Gimp. Após o carregamento, abra a aba “Cores” e selecione a ferramenta “Brilho e Contraste”. A Figura 1.16 ilustra os passos para ajustar os tons da imagem, o ajuste do brilho é feito pela barra nomeada de “Brilho” e a de contraste pela barra “Contraste”. Após a edição basta exportar a imagem nos formatos especificados no Quadro 1.

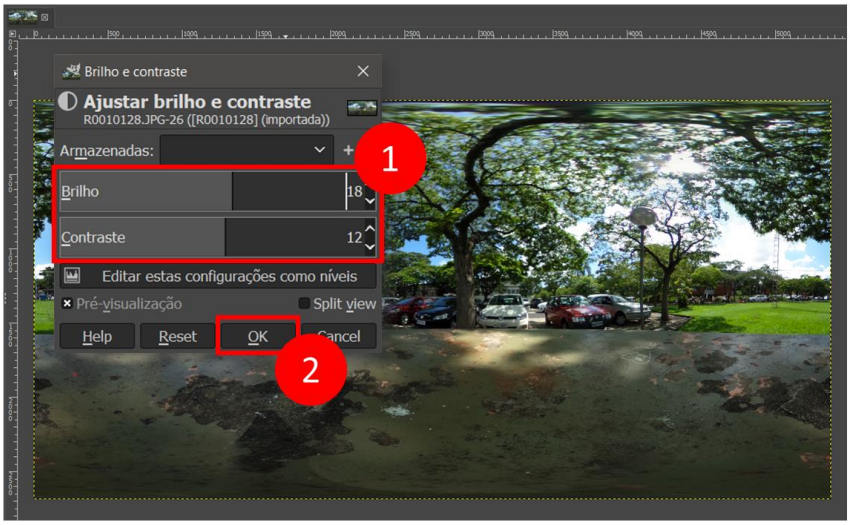


Figura 1.16. Ajustes de brilho e contraste utilizando Gimp 2

Edição de Imagem Panorâmica 360°

Para imagens panorâmicas é necessário adequá-las ao *Aspect Ratio* 2:1. Na subseção anterior, apresentaram-se as equações necessárias para verificar o formato do quadro e obter a altura em relação a largura da imagem. Sendo assim, primeiramente deve-se abrir a imagem utilizando o Gimp. A imagem utilizada para esse exemplo possui originalmente 9348x1624 pixels, aplicando a equação, a resolução atualizada é de 9348x4674 pixels. O próximo passo é utilizar a ferramenta “Tamanho da Tela de Pintura”. Abra o menu “Imagem” e selecione a ferramenta. A Figura 1.17 exemplifica o processo, (1) desative o *link* entre largura e altura, (2) insira os valores de largura e altura, (3) centralize a imagem e por fim (4) aplique a nova resolução pressionando *Resize*. A seguir, deve-se criar uma camada transparente para cobrir as partes remanescentes que surgiram após a atualização da resolução. A Figura 1.18 ilustra as etapas para a criação, (1) crie uma nova camada e (2) insira o nome, (3) finalize pressionando o botão “*Ok*”. Após a camada de fundo criada, deve-se aplicar a cor de fundo, a Figura 1.19 mostra como deve ser feito, (1) passe a camada transparente para segunda linha, (2) selecione a ferramenta “Preenchimento”, (3) selecione a cor, (4) ao abrir a janela selecione uma cor desejada e (5) pressione “ok”, (6) por fim aplique a cor na imagem de fundo.

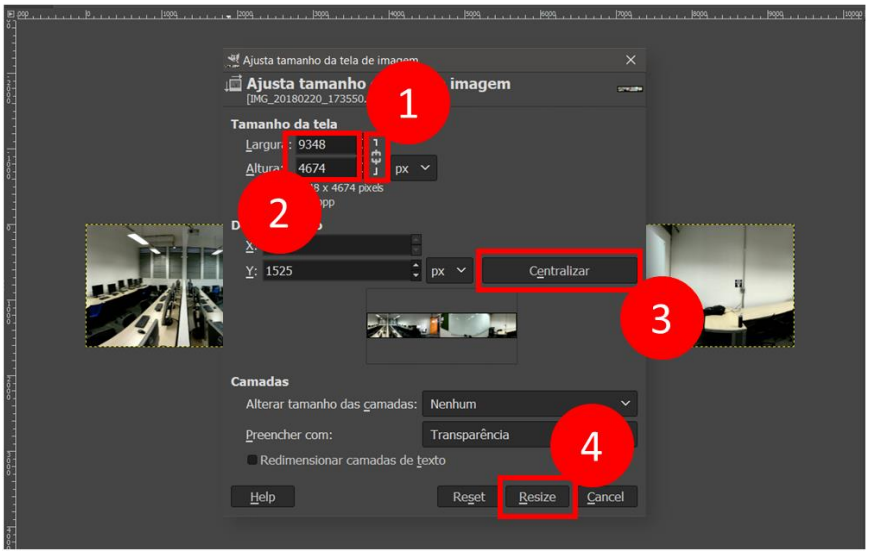


Figura 1.17. Ajuste da resolução final de uma imagem panorâmica 360°

Edição de Vídeo 360°

A edição é vídeos 360° é feita por *softwares* específicos. Esses *softwares* transformam os vídeos brutos em vídeos compatíveis em outros players de vídeos. O *software* que será utilizado neste exemplo é desenvolvido pela Ricoh Theta, pode ser baixado em (RICOH, 2018).

A manipulação é bem simples, basta selecionar o vídeo 360° bruto e arrastar para dentro do *software* da Theta. A Figura 1.20 exemplifica como selecionar o vídeo a ser convertido. Após a seleção do arquivo, indique o destino do vídeo e pressione o botão continuar. Após alguns minutos o vídeo final estará pronto.

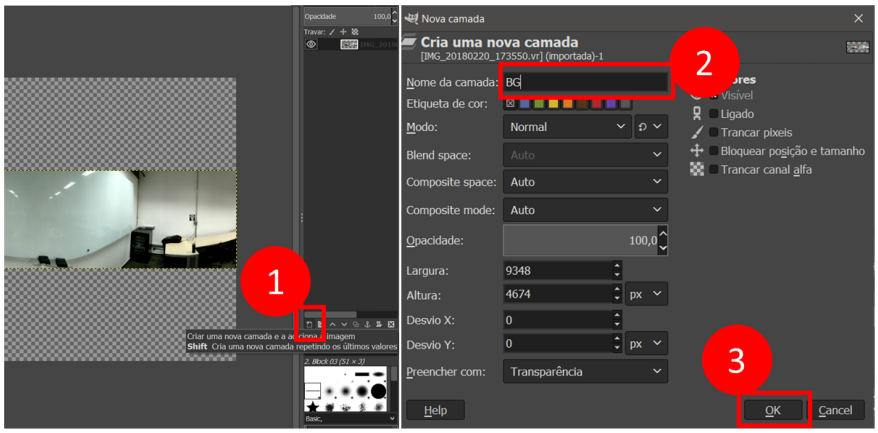


Figura 1.18. Criando uma camada transparente para aplicar cor de fundo

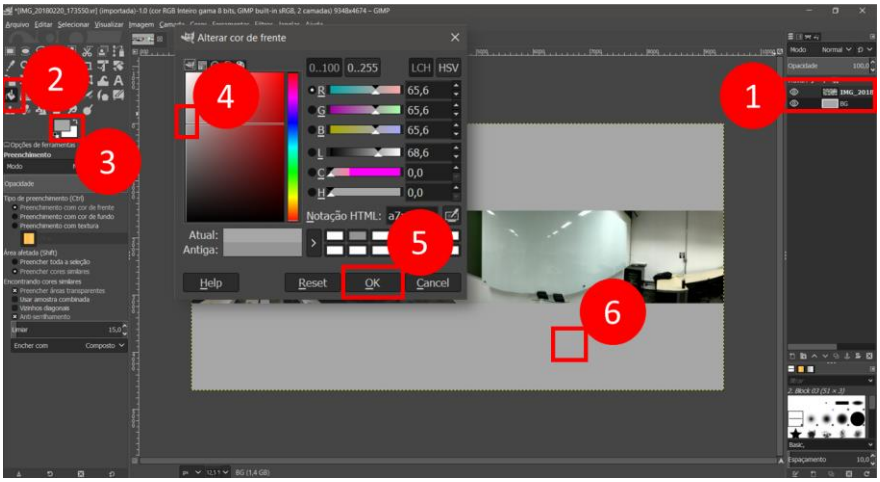


Figura 1.19. Aplicando cor à camada de fundo

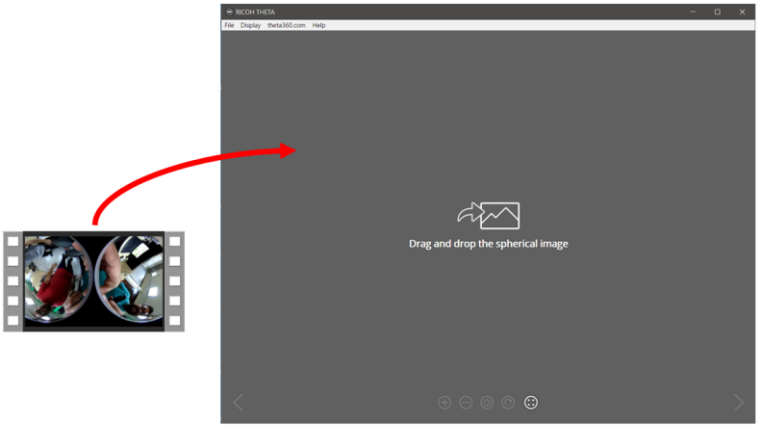


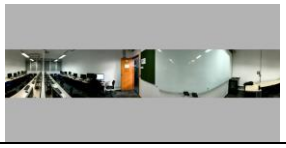
Figura 1.20. *Software* da câmera Ricoh Theta utilizado para converter vídeos

1.3.4. Exportação dos Materiais Adequados

Por fim, após as adequações de imagens e vídeos os arquivos resultantes são vários. Para utiliza-los na *Game Engine* Unity eles devem possuir as características especificadas no Quadro 2.

O Unity suporta diversos formatos de imagens e vídeos, é importante que todos os arquivos estejam adequados. Assim que todo o insumo estiver preparado, a próxima etapa é o desenvolvimento da aplicação.

Quadro 2. Informações sobre tipos de imagens e seus formatos após a adequação

Tipo:	Exemplo:	Aspect Ratio:	Formato:
Imagem 360° Esférica		2:1	jpg, png, tiff, psd, gif, bmp
Imagem 360° Panorâmica		2:1	
Vídeo 360°		2:1	avi, mp4, wmv, mov, mpeg, ogv

1.3.5. Preparação do Ambiente para Desenvolvimento no Unity3D

Antes de iniciar o desenvolvimento do projeto, é uma boa prática preparar o ambiente da *Unity* e realizar algumas configurações prévias. Como primeiro passo, define-se que o projeto será implementado para a plataforma *Android*, posteriormente pode-se alterar a plataforma alvo. No *Unity*, selecione no menu superior o item "*File*/*Build Settings"*. Na nova janela, selecione em *Plataform* a opção *Android* e pressione *Switch Plataform*. Isso fará que a solução compile os arquivos em um formato adequado. A Figura 1.21 (a) mostra as etapas.

No menu superior selecione a opção "*Edit/Project Settings/Player*". As configurações do *Player* irão ser exibidas à direita, as seguintes configurações devem ser aplicadas:

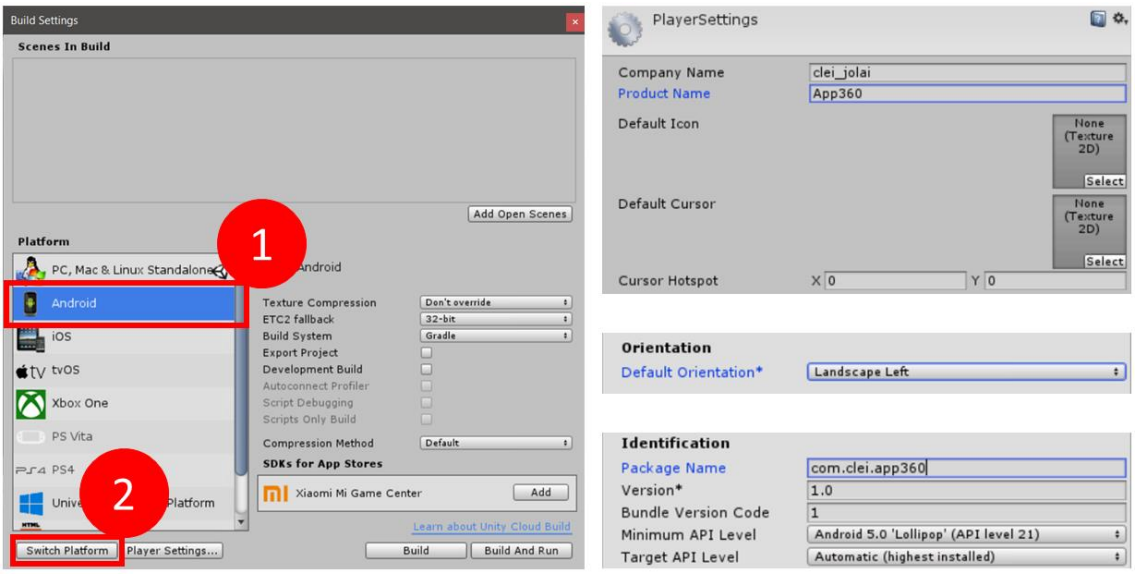
- Company name: clei_jolai;
- Product name: App360;

Em *Resolution and Presentation*, definir o item *Default Orientation* como *Landscape Left*.

Por fim, na aba *Other Settings*, definir os seguintes itens:

- Package name: com.clei.app360
- Minimum api level: *Android* 5.0 'Lollipop' (API level 21)

A Figura 1.22 (b) exemplifica os itens a serem configurados.



(a) (b)

Figura 1.21. Configurações do projeto, (a) configuração da plataforma, (b) configuração do *player*

Além das configurações iniciais, é necessário organizar o projeto em pastas destinadas a tipos de arquivos distintos. Assim sendo, na aba *Project* do Unity, crie as seguintes pastas dentro de *Assets*: *Content360* (Subpastas: *Video*, *Images*), *Images*, *Models*, *Scenes*, *Scripts* (Figura 1.22).

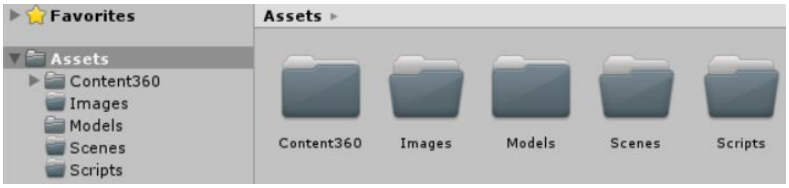


Figura 1.22. Pastas criadas em *Assets*

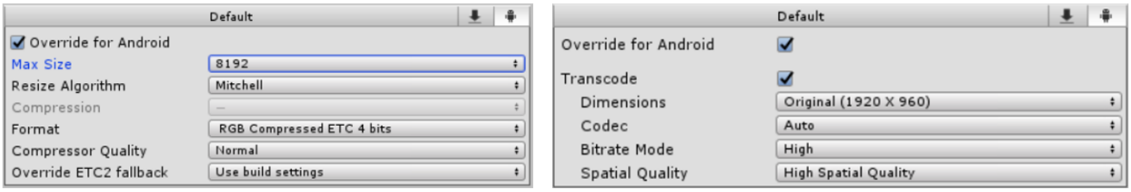
Importação dos Materiais

Após organizar em pastas, deve-se cumprir a etapa de importação dos materiais coletados e editados. A importação é simples: basta selecionar os arquivos e arrastar para as respectivas pastas, caso sejam imagens, arrastar para a pasta *Content360/Images* e caso sejam vídeos para a pasta *Content360/Videos*. O próximo passo é a configuração da qualidade dos arquivos.

Uma configuração importante é a qualidade da imagem para fotografias ou vídeos. Para cada arquivo importado para o Unity, faça a seguinte configuração. Caso sejam imagens, selecione todas e no menu direito selecione a opção *Texture Type: Default*. Na parte inferior do menu selecione o ícone *Android* (caso não esteja selecionado), marque a opção *Override for Android* e defina em *Max Size* o maior valor possível, neste caso o valor de 8192 pixels. Clique em aplicar. Essa configuração garante a máxima resolução da imagem dentro do ambiente virtual. A Figura 1.23 (a) mostra as opções corretas de configuração.

Para vídeos a configuração é semelhante, selecione todos os arquivos de vídeo e no menu direito marque a opção *Import Audio* (caso não esteja marcada). Selecione o

ícone do *Android*, marque a opção *Override for Android* e *Transcode*, em *Dimensions* mantenha a qualidade original, ou se possível a maior. A Figura 1.23 (b) mostra as configurações sugeridas para arquivos de vídeos. Caso queira criar um aplicativo para *Windows* ou WebGL, faça as mesmas configurações para o ícone PC, Mac & Linux e WebGL (se necessário).



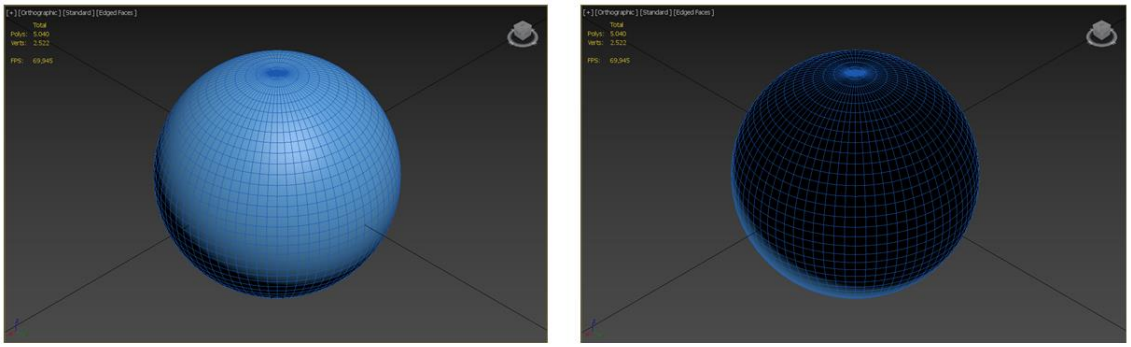
(a) (b)

Figura 1.23. Configurações de qualidade do material: (a) configuração para imagens; (b) configuração para vídeos

1.3.6. Desenvolvimento da Projeção dos Vídeos/Imagens

A estratégia abordada nesse trabalho foi baseada na modelagem de uma esfera para a projeção das imagens em 360°. Foi utilizado a princípio o *software* 3Ds Max, pode-se utilizar qualquer outro *software* de modelagem. A Figura 1.24 mostra a modelagem da esfera para projeção. A modelagem da esfera é básica, possui as seguintes características:

- Raio: 0.5 metros;
- Seguimentos: 72;
- Posicionamento: centro;
- *Normals*: invertida, sentido centro da esfera (Figura 1.25 (b));



(a) (b)

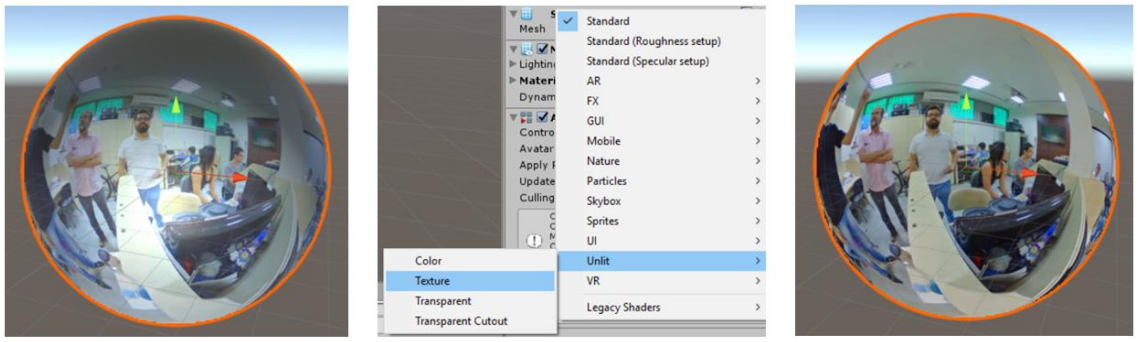
Figura 1.24. Modelagem de uma esfera com normais invertidas

Após a modelagem é necessário exportar o modelo no formato FBX. A exportação deve conter nas configurações: *Units – Meters* e *Up Axis – Y-Up*. Importe o modelo para o Unity, armazene em *Assets/Models*.

Sistema de Projeção

O sistema de projeção trata-se da sobreposição de textura em uma esfera, utilizando uma imagem ou vídeo 360°. A esfera foi modelada de forma que, ao receber uma textura ela se mantenha projetada na parte interna da esfera, assim pode-se visualizar a imagem por meio de uma câmera no seu interior. Primeiramente é necessário inserir os componentes no ambiente virtual. Dentro de *Assets/Models* selecione e arraste para dentro de *Hierarchy* o modelo da esfera. Posicione utilizando o menu *Inspector* do lado direito, assegure-se que *position* e *rotation* estejam com todos os valores em zero.

Após inserir e posicionar a esfera de projeção, vamos aplicar a textura. Selecione na pasta *Assets/Content360/* a imagem ou vídeo 360° que será utilizada, arraste para dentro do modelo da esfera. A Figura 1.25 (a) ilustra a componente esfera de projeção após a aplicação da textura. O próximo passo é definir o tipo de *shader*, pode-se observar que a Figura 1.25 (a) não está adequada para visualizar a imagem, sendo assim é necessário aplicar o *shader Unlit/Texture*. Selecione a esfera. No *Inspector* selecione a opção *shader* da textura, encontre a opção *Unlit* e aplique o *shader Texture*. A Figura 1.25 (b) exemplifica a seleção da textura. A Figura 1.25 (c) mostra a textura com o *shader Until/Texture*.



(a) (b) (c)

Figura 1.25. Texturização da esfera de projeção: (a) *shader standard;* (b) seleção do *shader* correto; (c) imagem com *shader until/texture*

Selecione a componente *Camera* dentro de *Hierarchy* e posicione-a no centro, isto é, definir *position* e *rotation* em zero. Em *Clipping Planes* defina *Near* em 0.01 e *Far* em 1.

Para vídeos é necessário configurar o *player* de vídeo e o áudio. Selecione a esfera e no menu *Inspector* adicione o componente *Audio Source*. Em *Video Player*, arraste para dentro de *Track 0 [1 ch] – Audio Source*, a esfera de projeção, referenciando o componente *Audio Source*.

Por fim o sistema de projeção está finalizado, a próxima etapa é implementar o sistema de interação que fará a câmera rotacionar por meio do *touch screen* do dispositivo móvel *Android*, e mouse quando compilado para Web/Windows.

1.3.7. Desenvolvimento da Interação em 360°

A interação para esse trabalho será a implementação de um mecanismo para rotação da câmera e seleção de objetos. Sendo assim, primeiramente é necessário criar um *script* na pasta *Scripts*. Dentro da pasta pressione com o botão direito e selecione *Create/C# Script*, nomeie como *TouchRotateCamera*. Abra o *script* e use o código do Quadro 3.

Quadro 3. Algoritmo de translação da câmera por toque na tela.

```
using UnityEngine;

public class TouchRotateCamera : MonoBehaviour {
    //Android
    private Vector3 FirstPoint; // Primeiro ponto de contado do dedo na tela
    private Vector3 SecondPoint; // Segundo ponto quando o dedo se movimenta
pela tela
    private float xAngle;
    private float yAngle;
    private float xAngleTemp;
    private float yAngleTemp;
    public float perspectiveZoomSpeed = 0.5f; // Taxa Zoom em Perpectiva

    //Web-Windows
    private float sensitivity = 8f; // Sensibilidade da rotação
    private float maxYAngle = 80f;  //Angulo máximo no eixo Y
    private Vector2 currentRotation;  // Rotação atual

    void Start() {
        xAngle = 0;
        yAngle = 0;
        this.transform.rotation = Quaternion.Euler(yAngle, xAngle, 0);
    }

    void Update() {
#if UNITY_ANDROID
        if (Input.touchCount == 1) {
            if (Input.GetTouch(0).phase == TouchPhase.Began) {
                FirstPoint = Input.GetTouch(0).position;
                xAngleTemp = xAngle;
                yAngleTemp = yAngle;
            }
            else if (Input.GetTouch(0).phase == TouchPhase.Moved) {
                SecondPoint = Input.GetTouch(0).position;
                xAngle = xAngleTemp - (SecondPoint.x - FirstPoint.x) * 180 /
Screen.width;
                yAngle = yAngleTemp + (SecondPoint.y - FirstPoint.y) * 90 /
Screen.height;
                this.transform.rotation = Quaternion.Euler(yAngle, xAngle,
0.0f);
            }
        }
        else if (Input.touchCount == 2) {
            // Armazenar os dois toques
            Touch touchZero = Input.GetTouch(0);
            Touch touchOne = Input.GetTouch(1);

            // Encontrar a posição no frame anterior de cada toque.
            Vector2 touchZeroPrevPos = touchZero.position -
touchZero.deltaPosition;
            Vector2 touchOnePrevPos = touchOne.position -
```

```
touchOne.deltaPosition;

            // Encontrar a distância entre os toques em cada frame.
            float prevTouchDeltaMag = (touchZeroPrevPos -
touchOnePrevPos).magnitude;
            float touchDeltaMag = (touchZero.position -
touchOne.position).magnitude;

            // Encontre a diferença nas distâncias entre cada frame.
            float deltaMagnitudeDiff = prevTouchDeltaMag - touchDeltaMag;

            // Altere o campo de visão com base na alteração da distância entre
os toques.
            GetComponent<Camera>().fieldOfView += deltaMagnitudeDiff *
perspectiveZoomSpeed;
            // Prenda o campo de visão para certificar-se de que esteja entre 0
e 10.
            GetComponent<Camera>().fieldOfView =
Mathf.Clamp(GetComponent<Camera>().fieldOfView, 10f, 60f);
        }
#endif

#if UNITY_STANDALONE_WIN || UNITY_WEBGL
        if (Input.GetMouseButton(0)) {
            currentRotation.x += Input.GetAxis("Mouse X") * sensitivity;
            currentRotation.y -= Input.GetAxis("Mouse Y") * sensitivity;
            currentRotation.x = Mathf.Repeat(currentRotation.x, 360);
            currentRotation.y = Mathf.Clamp(currentRotation.y, -maxYAngle,
maxYAngle);
            Camera.main.transform.rotation = Quaternion.Euler(currentRotation.y,
currentRotation.x, 0);
        }
#endif
    }
}
```

Selecione o componente *Camera* dentro do menu *Hierarchy*, no menu *Inspector* pressione o botão *Add Component* e digite *TouchRotateCamera*, selecione o *script* para adicionar. Por fim, modifique o valor de *Perspective Zoom Speed* para 0.1. Ao adicionar o script na câmera, garante que quando o usuário tocar e arrastar o dedo na tela do *smatphone*, sua rotação seja modificada. A Figura 1.26 (a) mostra o sistema de projeção em modo de edição Unity, em Figura 1.26 (b) mostra a rotação da câmera em modo de execução.

Após configurar o mecanismo de rotação da câmera, é necessário salvar a cena, assim sendo, selecione o menu *File/Save Scene As*, salve dentro da pasta *Scenes*.

(a) (b)

Figura 1.26. Desenvolvimento da interação: (a) sistema de projeção; (b) visualização por meio de toque na tela

1.4. Camada de Visualização

Nessa seção será apresentado a continuação do desenvolvimento do aplicativo visando a plataforma escolhida para a compilação. Sendo assim, vamos abordar a construção da camada de visualização para a plataforma *mobile*, *Windows* e *Web*.

1.4.1. Definição da Camada de Visualização

A camada de visualização é a forma como as imagens são apresentadas ao usuário e o tipo de interação que pode ser aplicada. No contexto de dispositivos móveis a camada de visualização pode ser tanto imagem apresentada na tela com interação de rotação, quanto aplicada em realidade virtual.

Em sistemas operacionais *Windows* ou *Web*, a imagem é apresentada em tela e a interação é realizada com o *mouse*. Sendo assim, é necessário definir qual a plataforma alvo para realizar as configurações adequadas. Continuando o desenvolvimento da seção anterior, para plataforma móvel, vamos acrescentar componentes de realidade virtual.

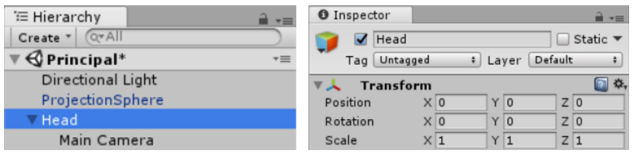
Realidade Virtual

Um bom framework de Realidade Virtual (RV) que está conquistando espaço é o *GoogleVR*[6]. Possui suporte para várias plataformas de desenvolvimento, incluindo *Android*, Unity, Unreal e Web. Para realizar o download, acesse o *site* e entre na área do desenvolvedor. Baixe o pacote de desenvolvimento para Unity.

No Unity *editor*, clique com o botão direito na pasta *Assets* da aba *Project*, selecione *Import Package/Custom Package*. Ao abrir a nova janela, selecione o arquivo baixado do *GoogleVR*. Ao fazer a importação, selecione todas as opções e pressione *Import*.

Crie um objeto vazio dento da aba *Hierarchy*, nomeie para "*Head*", esse será o componente que anexará a câmera. Posicione e rotacione o objeto *Head* para X, Y e Z em 0, para ambos. Por fim, arraste a câmera para dentro do objeto. A Figura 1.27 (a) exemplifica a hierarquia, o objeto chamado *Head* tem seu posicionamento na Figura 1.27 (b), tendo em seu interior o objeto câmera.

[6] https://vr.google.com/



(a) (b)

Figura 1.27. Adequação da câmera para camada de visualização em Realidade Virtual: (a) posicionamento da câmera dentro de um objeto vazio posicionado em 0; (b) ajustes de posicionamento do objeto *Head*

É necessário definir que o projeto é uma aplicação em realidade virtual. Para tanto, selecione o menu *Edit/Project Settings/Player*. No menu direito (*Android*) navegue até *XR Setting* e marque a opção *Virtual Reality Supported*. Em *Virtual Reality SDKs* pressione o botão "+" e selecione "*None"*, repita o passo e selecione agora "*Cardboard"*. Em ordem: primeiro *None* e segundo *Cardboard*.

Na pasta *Assets/Scripts*, crie um *C# Script* e nomeie para "ToogleViewer", escreva o código do Quadro 4:

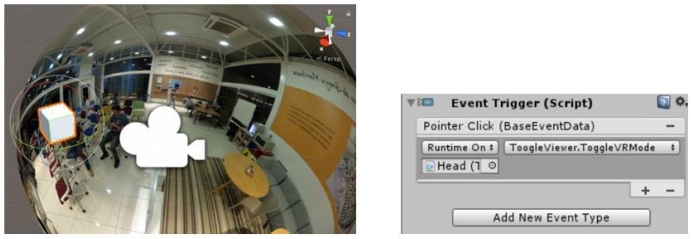
Quadro 4. Algoritmo de troca do visualizador de tela convencional para realidade virtual.

```
using System.Collections;
using UnityEngine;
using UnityEngine.XR;

public class ToogleViewer : MonoBehaviour {
    enum VRMode {
        SingleCamera,
        VRCamera,
    }

    void Start() {
        PlayerPrefs.SetInt("VRMode", (int)VRMode.SingleCamera);
    }

    public void ToggleVRMode() {
        if (PlayerPrefs.GetInt("VRMode") == (int)VRMode.SingleCamera) {
            PlayerPrefs.SetInt("VRMode", (int)VRMode.VRCamera);
            XRSettings.enabled = true;
            StartCoroutine(LoadDevice("cardboard"));
        }
        else if (PlayerPrefs.GetInt("VRMode") == (int)VRMode.VRCamera) {
            PlayerPrefs.SetInt("VRMode", (int)VRMode.SingleCamera);
            XRSettings.enabled = false;
            StartCoroutine(LoadDevice("None"));
        }
    }

    IEnumerator LoadDevice(string newDevice) {
        XRSettings.LoadDeviceByName(newDevice);
        yield return null;
        XRSettings.enabled = true;
    }
}
```

Adicione no componente *Head* o *script* ToogleViewer. O *script* é um mecanismo para alterar entre aplicações de visualização em tela e realidade virtual.

Para utiliza-lo, é necessário adicionar a funcionalidade de chamada de função *ToogleVRMode* do *script*, sendo assim é necessário inserir em cena um objeto que faça a chamada quando a tela do dispositivo for tocada. Crie um objeto cubo, configure a escala para X, Y e Z de 0.1, posicione em X:0, Y:0 e Z:-0.4, rotacione em X:0, Y:-45 e Z:0.

Selecione o cubo, pressione o botão *Add Component* (*Inspector*) e adicione o componente *Event Trigger*. Em *Event Trigger* pressione *Add New Event Type*, adicione o evento *Pointer Click*. Em *Pointer Click* arraste o componente *Head* para dentro de *None (object)*, após isso, selecione o método por meio do menu *No Function/ToogleViewer/ToggleVRMode()*. Agora o objeto está pronto para ser utilizado como botão para mudar a visualização. A Figura 1.28 exemplifica o processo.



(a) (b)

Figura 1.28. Mecanismo para troca de visualização: (a) cubo com evento de *click*; (b) configuração do evento de *click*

Acesse a pasta *Assets/GoogleVR/Prefabs/Cardboard/*, arraste para dentro da câmera o *prefab GvrReticlePointer*, garanta sua posição em zero. Selecione *GvrReticlePointer* e adicione o componente *GrvPointerPhysicsRaycaster*. Acesse a pasta *Assets/GoogleVR/Prefabs/EventSystem*, e arraste para dentro da aba *Hierarchy* o componente *GvrEventSystem*. Após essas configurações o aplicativo está pronto para ser compilado na plataforma *Android*.

Windows* e *Web

A camada visualização para plataforma *Windows* ou *Web* juntamente com suas interações foram implementadas na subseção "Desenvolvimento da Interação em 360°". No código foi implementado uma diretiva de compilador que define qual será a interação ao selecionar a plataforma alvo.

Sendo assim, basta seguir os passos da próxima seção para criar a compilação de cada plataforma.

1.5. Compilação da Aplicação

Nessa seção será apresentada etapas para compilar o aplicativo para plataformas específicas. Será abordado a compilação para dispositivos móveis *Android*, plataforma *Windows*/*Web*.

A compilação é algo relativamente simples, selecione o menu *File/Build Settings*. A Figura 1.29 mostra as etapas para construir um aplicativo para uma plataforma alvo. Os passos são, (1) adicionar ao *Scenes In Build* a cena desenvolvida,

(2) definir qual a plataforma (PC, *Android* e WebGL) e por fim (3) pressionar o botão *Build* e seleciona a pasta de destino.

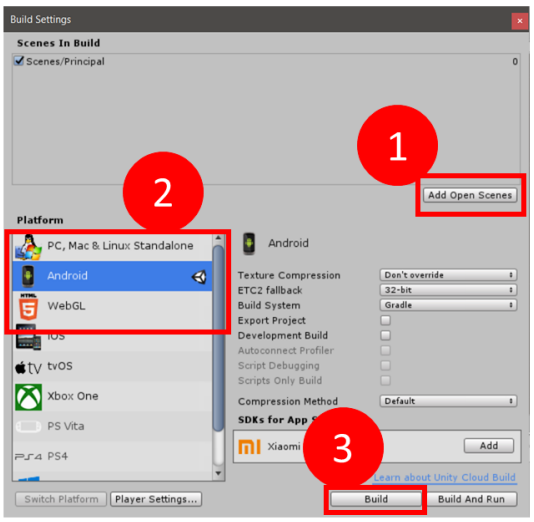


Figura 1.29. Compilação da aplicação para uma plataforma alvo

A Figura 1.30 mostra a execução do aplicativo na plataforma *Android*, a Figura 1.30 (a) mostra a camada de visualização em tela, a Figura 1.30 (b) mostra a camada de visualização em realidade virtual. Toda a troca de camada de visualização é feita ao clicar no cubo 3D em cena.

(a) (b)

Figura 1.30. Execução do aplicativo em plataforma *Android*: (a) execução padrão; (b) execução em realidade virtual

1.6. Comentários Finais

Esse trabalho exemplifica como criar aplicativos com suporte a imagens e vídeos em 360° de forma simples e rápida, como uma pequena amostra do que pode ser útil e gerado com conteúdo de imagens e vídeos em 360º.

Há uma gama de possibilidades a serem implementadas, utilizando Unity como plataforma de desenvolvimento para aplicativos 360°. Uma das peças importantes que podem ser destacadas é a implementação de interação com objetos 3D introduzidos no ambiente 360°, assemelhando-se a uma aplicação de Realidade Aumentada. Esses objetos podem complementar parte do cenário e enriquecer o ambiente. Algo

interessante é implementar interação dos objetos 3D com os usuários da aplicação, algo que melhore a experiência e uso, que forneça informações adicionais sobre o determinado ambiente.

Também pode-se destacar a utilização de interfaces gráficas de usuário (GUI – *Graphical User Interface*) para navegação dentro do cenário. Uma das grandes carências em aplicativos deste estilo são de interfaces adequadas, na visualização em RV ou *Single View*.

Desta forma, uma ótima opção de estudos futuros é o desenvolvimento de ambientes em 360° capazes de suportar elementos 3D interativos e navegação por meio de GUI. Há vários materiais para estudos, com destaque para: Ghimire (2016), Harmonic (2018) e Jaunt Studios (2017).

Referências

Bubl. (2018) “Bubl Camera 360°”. Disponível em: https://www.bublcam.com/. Acesso em: 28/04/2018.

Ghimire, Sujan. (2016) “Production of 360° Video”. Bachelor´s thesis, Degree Programme in Media Engineering, Helsinlo Metropolia University of Applied Sciences.

Harmonic. (2018) “360° VR Video”. Disponível em: http://harmonicinc.com. Acesso em 05/03/2018.

Jaunt Studios. (2017) “The Cinematic VR Field Guide: A Guide to Best Practices for Shooting 360°”. Disponível em: https://www.jauntvr.com/cdn/uploads/jaunt-vr-field-guide.pdf. Acesso em: 05/03/2018.

Kodak. (2018) “SP360 Camera”. Disponível em: https://kodakpixpro.com/Americas/cameras/actioncam/sp360/. Acesso em: 28/04/2018.

Ricoh Theta. (2018) “360° Experience”. Disponível em: https://theta360.com/en/. Acesso em: 28/04/2018.

Ricoh Theta. (2014) “User Guide Theta M15”. Disponível em: https://theta360.com/en/support/manual/m15/. Acesso em: 05/03/2018.

Samsung. (2017) “Gear 360°”. Disponível em: http://www.samsung.com/br/wearables/gear-360-2017/. Acesso em: 28/04/2018.

Xiaomi. (2018) “Sphere Camera”. Disponível em: http://www.mi.com/us/mj-panorama-camera/. Acesso em: 28/04/2018.

4. Capítulo 4

Autores:

Marcia Luciana Aguena Castro
Centro Universitário Hermínio Ometto
email: marciaaguena@fho.com.br

Ana Grasielle Dionísio Corrêa
Universidade Presbiteriana Mackenzie
email: ana.correa@mackenzie.br

Maria Amélia Eliseo
Universidade Presbiteriana Mackenzie
email: mariaamelia.eliseo@mackenzie.br

Valéria Farinazzo Martins
Universidade Presbiteriana Mackenzie
email: valeria.farinazzo@mackenzie.br

Capítulo

4

Análise de Dados em Pesquisa Científica: estudo de caso sobre a percepção dos estudantes de Computação no desenvolvimento de jogos

Marcia Luciana Aguena Castro, Ana Grasielle Dionísio Corrêa, Maria Amélia Eliseo, Valéria Farinazzo Martins

Abstract

The interconnectivity of the current world presents us with a very large volume of information continuously. All this information needs to be organized and combined in order to make sense and becomes knowledge. The role of a researcher or educator, interested in disseminating knowledge is not restricted to researching and obtaining results, but also doing a critical analysis about them, emphasizing in base form its main contributions and presenting them directly and clearly, so that can be understood as concise and concrete as possible. Statistical analysis is still the most powerful tool for such purposes; unfortunately, misinterpretations of statistical results are quite common. This work re-read the main statistical measures based not only on how the calculations are to be performed, but mainly on what they mean. In this process, the main misconceptions committed in such interpretations and how not to incur such errors will be highlighted. The choice of charts that maximize the understanding of information will be discussed and compared. All the text will be exemplified using the data analysis of a study case based on the data obtained from the perception of students in the use of game development (proficiency, motivation, engagement, proactivity) in first semester subjects of a Computing course, using the Excel tool.

Resumo

A interconectividade do mundo atual apresenta-nos um volume de informação muito grande continuamente. Toda essa informação precisa ser organizada e combinada para fazer sentido e tornar-se conhecimento. O papel de um pesquisador ou educador, interessado em disseminar conhecimento, não se restringe à pesquisa e obtenção de resultados, mas também a uma análise crítica sobre eles, ressaltando de forma embasada suas principais contribuições e apresentando-as de maneira direta e clara,

para que possa ser entendida da forma mais concisa e concreta possível. A análise numérica pautada em estatística ainda é a mais poderosa ferramenta para tais fins; infelizmente, porém, interpretações equivocadas sobre resultados estatísticos são bastante comuns. Este trabalho faz uma releitura das principais medidas estatísticas não só baseada em como os cálculos devem ser realizados, mas principalmente no que significam. Nesse processo, serão ressaltados os principais equívocos cometidos em tais interpretações e como não incorrer em tais erros. A escolha de gráficos que maximizem a compreensão das informações será discutida e comparada. Todo o texto será exemplificado utilizando a análise dos dados de um estudo de caso pautado nos dados obtidos da percepção de estudantes na utilização do desenvolvimento de jogos (perfil, motivação, engajamento, proatividade) em disciplinas de primeiro semestre de um curso de Computação, utilizando a ferramenta Excel.

1.1. Introdução

A quantidade de dados gerada atualmente, seja através de sensores, medições, redes sociais, fotos, mensagens e outros meios de comunicação, traz à tona um problema evidente: como trabalhar com estes dados e como produzir informação realmente relevante a partir deles. Ao analisarmos todos os dados que chegam a um indivíduo diariamente, podemos verificar que são gerados por uma série de acontecimentos culturais ou naturais (fenômenos), cujas representações envolvem um grande número de variáveis. Assim, a ciência tem por pretensão conhecer a realidade e interpretar estes acontecimentos, baseada na análise das variáveis intervenientes consideradas importantes nestes eventos. Em verdade, cada vez mais as pessoas são expostas aos dados e à Estatística que os envolve, mesmo que não se deem conta da presença desta, em maior ou menor intensidade.

A ciência tem, assim, o papel de estabelecer relações e encontrar ou propor leis explicativas para os fenômenos ou acontecimentos investigados. Para tanto, é necessário controlar, manipular e medir as variáveis consideradas relevantes a este entendimento. Existem alguns desafios inerentes a este fato que podem ser de natureza epistemológica, desde que a ciência não conhece a realidade, apenas a representa por meio de modelos e teorias dos diversos ramos do conhecimento, ou pela simplificação da aspiração de universalidade das explicações científicas, através da implicação e condicionamento da pesquisa a uma simplificação metodológica (Moita Jr., 2015).

Pelo exposto acima e de acordo com o apontado por Saraiva, Victer e Siqueira (2017), a Estatística é importante nas mais variadas áreas do conhecimento, pois permite uma análise até mesmo de questões sociais e econômicas. Assim, seu estudo não pode se pautar apenas na aplicação de fórmulas e cálculos para a resolução de exercícios de maneira mecânica. Torna-se necessário o desenvolvimento do chamado Pensamento Estatístico, com a absorção de conhecimento estatístico suficiente para resolver problemas do cotidiano.

Diante da importância que a Estatística assume para a ciência, é objetivo deste capítulo apresentar a elaboração de uma análise estatística para pesquisas, focado na área de Computação. Para complementar os conceitos, são discutidos exemplos práticos para elucidar aspectos inerentes à análise.

Tais exemplos práticos foram retirados de um estudo que foi realizado no primeiro semestre de 2017, na Faculdade de Computação e Informática da Universidade Presbiteriana Mackenzie, localizada na cidade de São Paulo, Brasil. Este estudo obteve dados provenientes de perfil, motivação, engajamento e proatividade de estudantes provenientes de três disciplinas do 1o ano (Laboratório de Programação, Tecnologia Web II e Linguagem de Programação I), com atividades relacionadas ao desenvolvimento de jogos nas disciplinas. Os cursos de graduação envolvidos neste estudo foram: Ciência da Computação, Sistemas de Informação e Tecnologia em Análise e Desenvolvimento de Sistemas. Os cursos são oferecidos nos períodos diurno e noturno [Martins, et al., 2018].

Para captar os dados sobre a percepção dos estudantes e professoras envolvidos nesta experiência, foram desenvolvidos dois instrumentos de coleta de dados, a saber: a) questionário com o objetivo de medir a motivação, o engajamento e a proatividade dos alunos, além do próprio perfil e dos pontos positivos e negativos do uso de desenvolvimento de jogos na disciplina (Apêndice A, Tabela A.1); b) uma entrevista com as professoras a fim de verificar a impressão delas com o uso de tal abordagem metodológica (Apêndice A, Tabela A.2). O capítulo está organizado como segue. A seção 1.2 apresenta a fundamentação teórica necessária para o entendimento do restante do trabalho. A seção 1.3 mostra a aplicação dos conceitos apresentados ao estudo de caso e a 1.4 os gráficos escolhidos para representar tais informações. Na seção 1.5 é apresentado as conclusões do trabalho.

1.2 Fundamentação Teórica

1.2.1 Análise de Dados

De acordo com Silva et al., (2016), um dado é um valor documentado ou resultado de medição. Quando um sentido semântico ou significado é atribuído aos dados, gera-se uma informação. A análise dessas informações gera conhecimentos que permitem comprovar hipóteses e/ou tomar decisões.

1.2.2 Conceitos de Estatística Descritiva

A estatística descritiva é uma ferramenta capaz de descrever ou resumir dados, mostrando aspectos relevantes de um conjunto de dados. Além disso, a estatística descritiva permite criar visualizações desses dados de forma a torná-los mais fáceis de se compreender.

É chamado de população o conjunto finito ou infinito de "unidades" que se deseja estudar, seja esse um conjunto de pessoas, eventos, ou quaisquer outros objetos. Por diversas razões, nem sempre é possível analisar todos os elementos desse conjunto. Um subconjunto dessa população, chamado amostra, então é eleito para que se possa observar mais detalhadamente as características da população que se deseja estudar [Everitt e Skrondal, 2002].

As características da amostra que variam de elemento a elemento, são chamadas de variáveis aleatórias e podem ser:

- **Quantitativas**: são as baseadas em alguma medida numérica, por exemplo: tamanho, temperatura, preço, quantidade de filhos ou tempo de serviço. Note que é possível executar operações aritméticas com esses valores (somar dois salários ou saber qual a diferença de altura entre duas pessoas).

- **Qualitativas**: representam classificações ou categorizações dos indivíduos Se algum tipo de ordem é denotada por essa classificação, a variável é denominada qualitativa ordinal (exemplos: grau de instrução, conceitos bom, regular e ruim, tamanhos pequeno, médio ou grande etc.), caso contrário qualitativa nominal (cor dos olhos, nacionalidade, estado civil ou modelos de carro).

A Figura 1.1 apresenta os passos da análise descritiva dos dados, que serão detalhados a seguir.

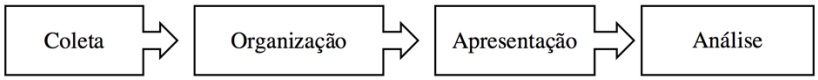


Figura 1.1. Passos de uma Análise Descritiva de Dados

a) Coleta dos dados: consiste na busca dos valores das variáveis que compõem o fenômeno a ser estudado em uma amostra.

b) Organização dos dados: consiste na estruturação dos dados, que geralmente ocorre em formato de tabelas, ou seja, tabulação de dados.

c) Apresentação dos dados: consiste na apresentação dos dados em formato de tabelas ou gráficos, de modo que facilite a leitura. Os gráficos permitem ilustrar os dados em padrões visuais que podem facilitar nas conclusões sobre a evolução do fenômeno ou de como os dados se relacionam.

d) Análise: com base no estudo dos dados trabalhados da amostra é possível inferir informações sobre a população e formular hipóteses que as justifiquem.

As ferramentas mais simples e significativas da estatística descritiva são as medidas de tendência central e as medidas de dispersão. O papel de tais medidas é tentar resumir toda a informação de uma amostra em poucos números, de forma que se possa conhecer seu comportamento como um todo. As medidas de tendência central tentam apontar o que é típico, característico, médio ou simplesmente o centro. As medidas de dispersão complementam essa informação indicando quanto da população está longe ou perto desse centro [Weisberg, 1992].

1.2.3 Medidas de Tendência Central

Segundo Weisberg (1992) medidas de tendência central resumem um valor típico de um conjunto de dados e são frequentemente associadas às médias (aritmética, ponderada, geométrica por exemplo). As médias são sempre utilizadas com a intenção de se transmitir a informação mais relevante e concisa sobre uma população. Há três medidas estatísticas que focam três diferentes aspectos do que é típico para uma variável: moda, mediana e média.

A **média aritmética** (M_e) estabelece uma relação entre o conteúdo total de todos os elementos de uma amostra e a quantidade desses elementos, sendo especialmente adequada aos dados quantitativos, tirando total proveito de suas propriedades aritméticas. É certamente a mais utilizada e expressa o valor médio de n observações ($x_1, x_2, \ldots, x_n$) da variável X através da fórmula:

$$M_e(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Com uma pequena alteração na fórmula, podemos calcular a média a partir da frequência ou porcentagem (f_i) em que cada um dos valores de x_i ocorre:

$$M_e(X) = \sum_{i=1}^{n} x_i \, f_i$$

Segundo Everitt e Skrondal (2002), a **moda** é o valor mais frequente em um conjunto de observações, ou seja, ela aponta a categoria mais típica de uma variável aleatória. Segundo Weisberg (1992) essa medida é particularmente importante em dados qualitativos nominais (divididos em categorias não ordenadas), visto que não há outra maneira de medir o que é mais típico. Além disso, também pode ser aplicada em dados qualitativos ordinais ou quantitativos (basta adequar as categorias a intervalos de valores). Uma amostra pode ser unimodal (ocorrência de apenas uma moda), bimodal (ocorrência de duas modas), multimodal (ocorrência de várias modas) e também amodal (não ocorrem modas).

Quando os valores de uma variável aleatória podem ser ordenados, a **mediana** pode ser utilizada para indicar a posição central nessa ordenação. Em dados qualitativos ordinais, a mediana é particularmente expressiva, visto que moda não leva em consideração a ordenação, e a média pode não traduzir exatamente as categorias da classificação. Nesses casos a obtenção da mediana é feita somando-se os elementos presentes em cada categoria, respeitando a ordenação entre eles, até chegar a metade dos dados. Esse é o valor que a mediana representa.

1.2.4 Medidas de Dispersão

Não é o suficiente sabermos qual é o valor típico na distribuição de uma variável aleatória. Também é preciso saber o quão típico é esse valor, ou seja, como todos os outros valores estão espalhados. A variância e o desvio padrão, as medidas de dispersão mais importantes, foram desenvolvidas para dados numéricos. Para outros tipos de dados medidas semelhantes à variância foram criadas [Weisberg, 1992].

As medidas de dispersão aumentam quanto maior o espalhamento de valores da variável. Os menores valores de dispersão acontecem quando não há dispersão alguma e todos as observações estão centralizadas em algum ponto e os maiores valores ocorrem quando a dispersão aumenta, ou seja, as ocorrências tendem a distribuírem-se uniformemente sobre todos os valores possíveis da variável.

O **desvio médio** é a média dos módulos das diferenças entre as observações e a sua média:

$$DM(X) = \frac{1}{n} \sum_{i=1}^{n} |x_i - M_e(X)|$$

A escolha de somar-se os módulos da diferença é feita pois se simplesmente considerássemos somente as diferenças entre os valores observados e a média, as diferenças positivas (quando x_i é maior que a média) e as negativas (quando x_i é menor que a média) se anulariam e o valor do desvio médio seria zero.

A **variância** de uma população é a média do quadrado da soma das diferenças entre os valores observados e a sua média, ou seja:

$$\mathrm{Var}(X) = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{x_i} - \mathrm{M_e}(X))^2$$

A escolha pelo quadrado dos desvios da variância é feita quando deseja-se enfatizar as distâncias maiores na soma dos desvios.

Intimamente relacionado à variância, o **desvio padrão** é a raiz quadrada da variância, e é preferido quando se deseja medir o espalhamento com o mesmo modelo da variância, porém com valores mais próximo à escala da amostra em questão.

$$DP(X) = \sqrt{Var(X)} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - M_e)^2}$$

1.2.5 Variáveis Multidimensionais e Medidas de Correlação

Uma outra maneira de entender o comportamento de uma variável aleatória é compará-lo ao de outras variáveis. Em análises multivariadas (que se referem a múltiplas variáveis) é possível analisar o grau de dependência, ou correlação, entre um grupo de variáveis (duas ou mais) [Hair et al., 2009], ou seja, o quanto o valor de uma variável pode estar relacionado ao de outra variável (ou outras variáveis).

Dados *n* pares de valores de duas variáveis aleatórias *X* e *Y* dados por (x_1, y_1), (x_2, y_2), ... , (x_n, y_n). O **coeficiente de correlação** entre as variáveis *X* e *Y* é dado pela fórmula:

$$Corr(X) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - M_e(X)}{DP(X)} \right) \left(\frac{y_i - M_e(Y)}{DP(Y)} \right)$$

O coeficiente de correlação é um número no intervalo [-1,1] e quanto mais próximo de zero seu valor, menos correlacionada estão as variáveis *X* e *Y*. Quanto maior o módulo do coeficiente mais dependentes são as variáveis, porém se ele é negativo as variáveis são inversamente correlacionadas, ou seja, quando uma cresce a outra decresce. O coeficiente positivo indica uma correlação direta.

Note que os indicativos de dependência (ou correlação) não são suficientes para estabelecer causalidade entre as variáveis aleatórias.

1.3 O Estudo de Caso e suas Estatísticas

São várias as dificuldades enfrentadas por professores e estudantes no processo de ensino-aprendizado de linguagens de programação. As dificuldades são frequentemente associadas à alta dependência do pensamento lógico-matemático desses cursos [Silva et al., 2015]. Por essa razão, professores e tutores sempre procuram por novas metodologias de ensino-apredizado que possam motivar os estudantes a aprender a programar [Eseryel et al., 2014].

O Estudo de Caso apresentado neste capítulo corresponde a uma experiência do desenvolvimento de jogos digitais em disciplinas de programação que ocorrem no primeiro e segundo semestres de cursos de graduação em Computação de uma universidade privada brasileira. O estudo traz a opinião de alunos e professores quanto à

aplicação da abordagem de desenvolvimento de jogos nessas disciplinas, assim como a metodologia aplicada em cada uma delas. Os resultados obtidos apontam que essa abordagem intensifica o aprendizado dos conceitos inerentes às disciplinas de programação, além de fomentar a motivação, o engajamento e a proatividade, fatores esses muito importantes na construção de suas competências e habilidades.

1.3.1 Os Cursos e as Disciplinas Envolvidas

O estudo aconteceu no primeiro semestre de 2017, na Faculdade de Computação e Informática da Universidade Presbiteriana Mackenzie, localizada na cidade de São Paulo, Brasil. Os cursos de graduação envolvidos neste estudo foram: Ciência da Computação, Sistemas de Informação e Tecnologia em Análise e Desenvolvimento de Sistemas. Os cursos são oferecidos nos períodos diurno e noturno. Nestes três cursos são oferecidas as seguintes disciplinas relacionadas ao ensino de programação e que foram elencadas neste estudo: Laboratório de Programação I; Linguagem de Programação II; Tecnologia Web.

As metodologias adotadas para o ensino de programação, utilizadas nas três disciplinas supracitadas, empregaram, além dos conceitos teóricos e exercícios de fixação, projetos para o desenvolvimento de jogos. No entanto, tais metodologias não seguem o que é preconizado pelas abordagens de Aprendizado Baseado em Problema (Problem Based Learning - PBL) [Loyens; Kirschner; Paas, 2011] ou Aprendizado Baseado em Projeto (Project Based Learning) [Krajcik; Blumenfeld, 2010]. Nestas abordagens problema/projeto é apresentado ao estudante, e o conteúdo é ministrado pelo professor ou o estudante é impulsionado a buscar uma solução. Na abordagem proposta neste capítulo, o conteúdo é ministrado antes do estudante ser apresentado ao projeto de jogos. Também vale ressaltar que é o estudante que traz o projeto que ele queira desenvolver, a partir de um tema amplo, proposto pelo professor. Neste caso, o estudante pode querer ou necessitar desenvolver algo mais complexo do que o conteúdo de aula e, então, exerce sua proatividade para descobrir e estudar outros assuntos pertinentes ao projeto.

1.3.2 Metodologia utilizada para captação de dados nas disciplinas

O sucesso do processo de ensino-aprendizado pode ser medido, além das tradicionais avaliações de conteúdo, por alguns comportamentos desenvolvidos durante o seu andamento. Alguns desses comportamentos podem ser notados tanto pelo professor quanto pelo próprio aluno. Dentre eles:

- A motivação, que está relacionada ao impulso que faz com que as pessoas ajam para atingir seus objetivos.
- O engajamento é referente ao ato de participar de modo voluntário para algum trabalho ou atividade. Pode ainda ser utilizado no sentido de dedicação, ou seja, fazer algo com afinco e vontade.
- A proatividade é o comportamento de antecipação e de responsabilidade pelas próprias escolhas e ações frente às situações impostas pelo meio.

A fim de captar os dados sobre a percepção dos estudantes e professoras envolvidos nesta experiência, foram desenvolvidos dois instrumentos de coleta de dados, a saber: a) questionário com o objetivo de medir os quesitos de motivação, engajamento e proatividade dos alunos, além do próprio perfil e dos pontos positivos e negativos do uso de desenvolvimento de jogos na disciplina (Tabela A.1); b) uma

entrevista com as professoras a fim de verificar a impressão delas com o uso de tal abordagem metodológica (Tabela A.2).

O questionário de satisfação (a) dos estudantes era, basicamente, composto por questões de múltipla escolha (Escala Likert (1932)), com exceção das questões 1, 13, 14, 15 e 16 que tinham outros padrões de respostas e as questões 2 e 17 que eram questões abertas para que os alunos pudessem expor sua percepção de maneira mais aprofundada. Já a entrevista com as professoras (b) das disciplinas foi composta por questões abertas.

O questionário de satisfação e a entrevista com professoras ficaram disponíveis para resposta por duas semanas. Os alunos e as professoras foram convidados a responderem livremente, sem a identificação dos participantes.

1.3.3 A Análise Estatística

A análise estatística deste estudo de caso foi feita sobre as respostas dos alunos (Tabela A.1). As questões cujo padrão de resposta era a Escala de Likert foram consideradas variáveis qualitativas ordinais e as questões 1, 13, 14, 15 e 16 variáveis qualitativas nominais. Não foi feito estudo sobre as questões abertas.

Muito se discute sobre a adequação do cálculo da média (e também das medidas de dispersão) sobre a Escala de Likert [Jamieson, 2004], pois qual seria a média entre "concordo totalmente" e "discordo totalmente"? A abordagem mais adequada às variáveis qualitativas é apresentar os percentuais de cada classe de respostas. Assim, para analisar as respostas de forma global e medir o impacto do experimento, foi feito uma junção geral das nove (9) questões da Tabela A.1 que utilizaram a escala Likert e o resultado encontra-se na Tabela 1.1.

Tabela 1.1. Apresentação das porcentagens da junção das questões que utilizam a escala de Likert do questionário, em que as categorias são: Concordo Totalmente (CT), Concordo parcialmente (C), Não concordo ou discordo (N), Discordo parcialmente (D), Discordo totalmente (DT)

	CT	C	N	D	DT
GERAL	27.19%	26.90%	25.86%	10.32%	9.73%

Apesar da ressalva sobre o caráter de variáveis qualitativas, é inegável que escala Likert denota variáveis qualitativas ordinais, ou seja, há uma escala crescente de aceitação em suas categorias. Nas variáveis qualitativas ordinais, ainda que não se possa medir com clareza qual a distância, ou quantificar a diferença entre as categorias, é possível criar uma ordenação entre duas observações. No estudo de caso, as questões aplicadas aos alunos foram polarizadas de forma que a maior concordância com um grupo de observações, quando analisadas em conjunto, denotaria uma "nota" nas características que se desejava observar (*motivação, engajamento, proatividade*).

Dessa forma, podemos adaptar as observações atribuindo valores crescentes de 1.1 (indo de "discordo totalmente" a "concordo totalmente") para as respostas, é possível analisá-las também como variáveis quantitativas.

1.3.4 Moda

Das porcentagens da Tabela 1.2, é possível notar que as **modas** das questões 4 e 5, respectivamente são as categorias CT e C, pois suas porcentagens de ocorrência são maiores que as das outras categorias. A moda, enquanto medida que reflete a maioria, é mais significativa quanto maior o seu valor. Isso quer dizer que quanto maior a moda, menos uniforme será a amostra, enquanto quando ela é baixa a amostra tende a ser mais uniforme. Podemos observar essa tendência observando a Tabela 1.2 com o resultado das questões 4 e 5. Na questão 4 a moda é mais alta e por essa razão sobrará menos casos para serem divididos entre as outras categorias. Já na questão 5, a moda é menor, e por consequência sobrará mais casos para serem divididos entre as outras categorias. Para entender a importância desse detalhe, podemos utilizar os mesmos números em um cenário de eleição. Nesse caso, no cenário da questão 4, o candidato na liderança estaria com uma grande margem e numa situação muito mais tranquila do que no cenário da questão 5 em que teríamos pelo menos três candidatos com uma representatividade bastante parecida.

Tabela 1.2 – Comparação da moda nas questões 4 e 5

4. Eu já gostava de programar antes de fazer essa disciplina.					
	CT	C	N	D	DT
GERAL	**38,49%**	20,75%	24,15%	7,17%	9,81%
	Moda				
5. Passei a gostar mais de programar após fazer essa disciplina.					
	CT	**C**	N	D	DT
GERAL	24,91%	**26,04%**	23,77%	10,19%	15,47%
		Moda			

1.3.5 Mediana

Como já foi dito na seção anterior, a medida de tendência central que mais se adequa a variáveis qualitativas ordinais é a mediana. Podemos calculá-la sobre os dados da Tabela 1.3, somando as porcentagens das categorias cumulativamente da direita para a esquerda (ou vice-versa), como mostrado na Tabela 1.3. O procedimento é ordenar todas as observações enfileiradas em ordem crescente, e ir somando elemento a elemento até a posição central dessa fila (metade das observações ou os 50% de frequência). No exemplo essa posição estaria em algum ponto da categoria “C -

concordo parcialmente", sendo considerada por essa razão a mediana da variável aleatória.

Tabela 1.3. Cálculo da Mediana

	CT	**C**	N	D	DT
GERAL	27,19%	**26,90%**	25,86%	10,32%	9,73%
Soma Acumulativa (direita para esquerda)	100,00%	**72,81%**	45,91%	20,05%	9,73%
Soma Acumulativa (esquerda para direita)	27,19%	**54.09%**	79.95%	90,27%	100.00%
		Mediana			

1.3.6 Média Aritmética

A média aritmética só pode ser utilizada sobre a amostra do estudo de caso se utilizarmos a quantificação de 1 a 5, como explicado anteriormente. Dessa forma a média pode ser calculada pela soma de cada um desses valores multiplicado pela sua porcentagem (frequência), como mostrado na Tabela 1.4.

Tabela 1.4. Cálculo da Média

	CT	**C**	**N**	D	DT
Valor Quantitativo	5	**4**	**3**	2	1
GERAL (frequência)	27.19%	**26.90%**	**25.86%**	10.32%	9.73%
$f_i x_i$	1,36	**1,08**	**0,78**	0,21	0,01
		Média 3,52			

Note que a média, diferentemente da mediana e a moda, indica aqui um valor que não existe nas observações. Isso se deve ao fato de que a média é um valor calculado sobre as observações e não um valor escolhido dentre elas (como é a moda). Embora bastante popular e representativa na maioria dos casos, se analisada

isoladamente pode levar a resultados equivocados. Isso se deve ao erro frequente de se confundir os significados de média, moda e mediana. Alguns exemplos:

- É comum que a maioria dos elementos sejam maiores ou menores do que a média pois ela não precisa coincidir com a mediana, essa sim dividiria as observações ao meio.
- A média também não precisa coincidir com o elemento que mais ocorre. Esse é o papel da moda.
- A média é afetada por valores extremos mesmo que pouco representativos. Imaginemos que todas as observações desse experimento tivessem o valor 2 (D – Discordo parcialmente) menos uma com o valor 1 (DT – discordo totalmente). A média nesse caso seria um valor levemente menor que 2, porém não seria falso afirmar que praticamente todas as respostas estão acima da média mesmo sendo um valor baixo relativamente à escala escolhida.

1.3.7 Medidas de dispersão

Na prática, salvo casos onde as variáveis têm comportamento muito particulares, as medidas de dispersão trazem informações muito semelhantes. Analisemos as medidas de tendência central e de dispersão das variáveis do estudo de caso baseadas as questões apresentadas na Tabela 1.5.

Tabela 1.5. Medidas de Dispersão

	Média	Moda	Mediana	Desvio Médio	Desvio Padrão	Variância
Questão 5	3.346	4	4	1.160	1.363	1.857
Questão 6	3.602	4	4	0.958	1.132	1.282
GERAL	3.515	5	4	1.048	1.259	1.585

Como foi visto na seção 1.2, o cálculo do desvio médio é baseado no módulo das diferenças, já a variância e o desvio padrão são baseados no quadrado das diferenças. O desvio médio ser menor que o desvio padrão, indica que as diferenças entre os valores da amostra diferem em valores maiores que uma unidade, pois enquanto o quadrado de números maiores que 1 aumenta o quadrado de números menores que 1 diminui.

O cálculo do desvio padrão e da variância, no entanto, são baseados no quadrado das diferenças. Por essa razão ambos são mais influenciados por grandes diferenças do que por pequenas. Ambos dão uma noção muito parecida, mas é preciso um passo a mais para o cálculo do desvio padrão (raiz quadrada) para fazê-lo representar de maneira mais próxima os valores das diferenças.

Na Tabela 1.5, podemos concluir que a dispersão dos números é maior que 1 pois todos os desvios padrões são maiores que os desvios médios. No caso da questão 6, temos um desvio médio menor do que 1 e ainda assim o desvio padrão é maior que o desvio médio. Isso indica que deve haver desvios maiores do que, por exemplo, na questão 5, mas em menor número.

1.3.8 Coeficiente de Correlação

As questões analisadas foram concebidas no intuito de analisar os quesitos: motivação, engajamento, proatividade e perfil dos alunos. Assim, era preciso estabelecer e quantificar a influência dos quesitos entre si para esclarecer dúvidas tais como:

- O perfil do aluno, ou seja, as características do seu comportamento advindas de sua personalidade e gostos, influenciaria nos outros quesitos?
- Como os outros quesitos da pesquisa estariam relacionados?

Para estudar a relação entre os quesitos da pesquisa, coeficientes de correlação foram calculados cruzando-se todos os itens do questionário, e o mapa da Figura 1.2 foi gerado para interpretá-los. As diferenças de correlações foram evidenciadas associando-se uma paleta de tons de cinza aos seus valores: tons mais claros representam alta correlação inversa; os tons intermediários representam correlação menor correlação; e os tons escuros alta correlação direta. Inicialmente foi pensado em um gráfico 3D onde cada célula se ergueria de um plano de acordo com o seu valor, porém a visualização não era tão boa quanto a obtida com o mapa.

Uma maneira simples de analisar um mapa desse tipo é fazê-lo em dois passos:

1. Analisar os grandes blocos formados por valores semelhantes, e como as questões pertencem a categorias (perfil, motivação, engajamento e proatividade), analisar onde esses blocos se localizam sobre as áreas de tais categorias.
2. Analisar cada quadrado do mapa para extrair uma relação entre os quesitos relacionados nele de forma mais detalhada.

		Perfil			Motivação					Engajamento		
	Questões	**1**	**2**	**4**	**3**	**5**	**10**	**11**	**12**	**7**	**8**	**9**
Perfil	**2**	0.50										
	4	0.45	0.08									
Motivação	**3**	-0.19	-0.19	0.24								
	5	-0.29	-0.27	0.30	0.70							
	10	-0.23	-0.21	0.14	0.57	0.57						
	11	-0.24	-0.18	0.23	0.51	0.49	0.64					
	12	-0.17	-0.10	0.17	0.37	0.38	0.50	0.77				
Engajamento	**7**	-0.21	-0.14	0.17	0.61	0.52	0.61	0.67	0.51			
	8	-0.16	-0.14	0.15	0.35	0.44	0.43	0.35	0.23	0.28		
	9	0.06	0.01	0.15	-0.08	-0.00	0.14	-0.06	-0.03	-0.05	0.25	
Proatividade	**14**	0.19	0.14	0.13	-0.05	-0.01	0.00	-0.06	-0.03	-0.07	0.10	0.40

Correlação										
Inversa					Neutra	Direta				
Forte		Média		Fraca			Média		Forte	
-1.00	-0.80	-0.60	-0.40	-0.20	0.00	0.20	0.40	0.60	0.80	1.00

Figura 1.2. Mapa do cruzamento das correlações entre as questões de perfil, motivação, engajamento e proatividade

A partir dessa análise, algumas observações foram feitas e aliado à vivência dos pesquisadores na sala de aula, algumas teses foram formuladas [Martins et al, 2018]:

1. As respostas sobre a motivação formaram um bloco coeso e diretamente correlacionado de forma que as respostas se validam duas a duas. Assim,

podemos dizer que a motivação se manifestou em de forma coesa, em todos os aspectos retratados nas questões.

2. De um modo geral, as respostas de perfil do estudante sobre sua experiência prévia têm uma correlação razoável com a quantidade de linguagens conhecidas e com o gosto pela programação, porém o gosto pela programação se correlaciona fracamente com a quantidade de linguagens conhecidas.
3. Ainda sobre as respostas referentes ao perfil do estudante, é interessante notar que, no geral, a experiência prévia do estudante e o gosto pela programação se correlacionam indireta e fracamente com sua motivação, engajamento e proatividade. Podemos concluir que a abordagem das aulas não dependeu do conhecimento do perfil "pregresso" do aluno, funcionando igualmente para todos.
4. As respostas de motivação tendem a se correlacionar mais fortemente com o engajamento dos estudantes na sala de aula (questões 7 e 8), diminuindo sua força nas tarefas fora do período de aula e na proatividade. Podemos concluir que o aluno se sente mais motivado se está engajado na sala de aula, porém essa motivação se enfraquece quando fora dela.
5. A baixa correlação entre as questões de engajamento talvez seja pelo fato desse quesito dar-se de forma diferente dentro e fora da sala de aula. Principalmente em se tratando de problemas mais complexos e que demandam mais tempo.

Vale notar que a leitura do mapa de correlação ajuda a formular teses para algumas hipóteses, porém sem o conhecimento sobre o assunto em questão é impossível tirar qualquer conclusão, que não numérica, sobre o que o mapa representa.

1.4. Apresentando Informações em Gráficos

Gráficos são ferramentas utilizadas na estatística que permitem a apresentação da informação de forma visual, facilitando a absorção e comparação do conteúdo. Segundo [Saraiva, 2017]. A representação gráfica de um fenômeno deve obedecer a certos requisitos fundamentais:

- Simplicidade – o gráfico deve ser destituído de detalhes de importância secundária, que possam levar o observador a uma análise errônea;
- Clareza – o gráfico deve possibilitar uma correta interpretação dos valores representativos do fenômeno em estudo;
- Veracidade – o gráfico deve sempre expressar a verdade sobre o fenômeno em estudo.

Cada tipo de gráfico tem diferentes apelos visuais que podem ressaltar diferentes características de um cenário. A seguir serão apresentados alguns tipos de gráficos utilizados nesse estudo de caso e suas principais características.

1.4.1 Gráficos de Linha

Quando a intenção é apresentar um valor que varia ao longo do tempo, ou ao longo de qualquer outro valor também contínuo, os gráficos de linha são os mais tradicionalmente indicados. Não são indicados para variáveis qualitativas e por essa razão não foram empregados nesse estudo de caso. Na Figura1.3 é mostrado um gráfico de linha que representa a distribuição Normal (2,5; 0,75). Na figura 1.3 foi utilizado um

gráfico de linha para mostrar a quantidade de disciplinas já conhecida pelos alunos em cada disciplina. Compare a visibilidade da informação com a Figura 1.4.

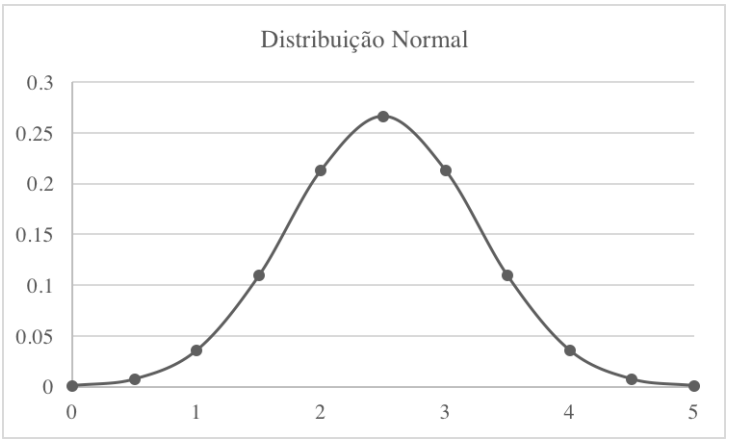


Figura 1.3. Gráfico de linha representando a distribuição Normal (2,5; 0,75)

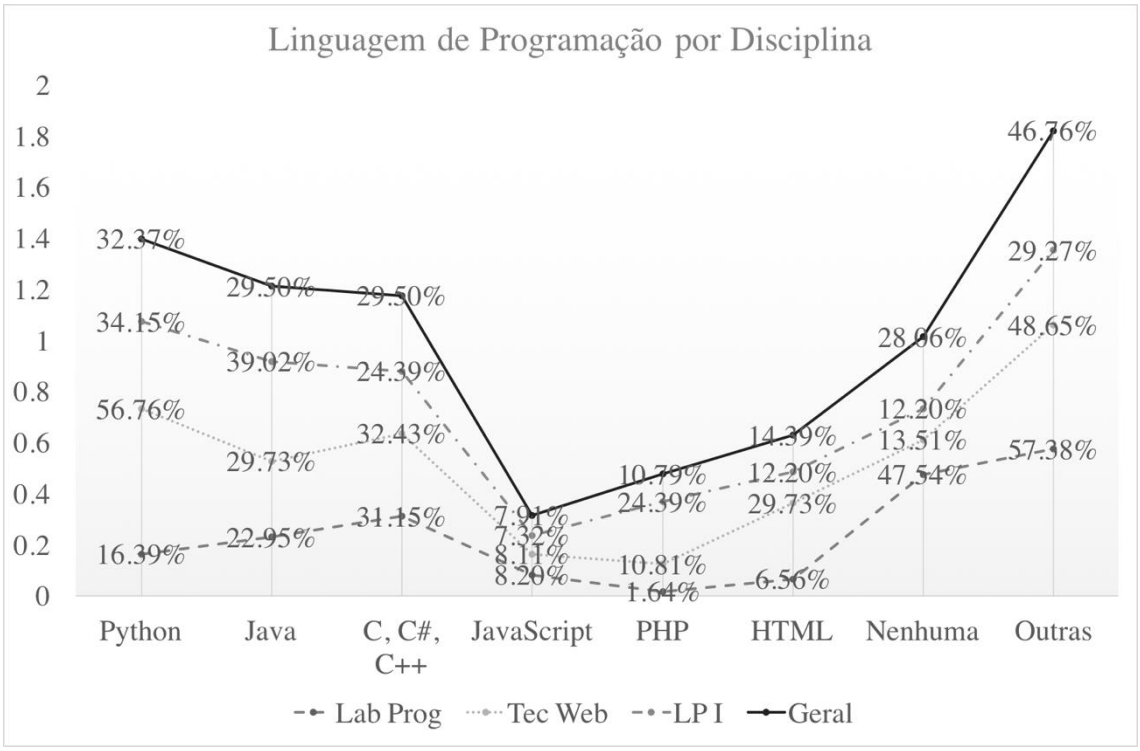


Figura 1.4. Gráfico de linha representado as linguagens de programação por disciplina do curso

1.4.2 Gráficos de Pizza

Os gráficos de pizza mostram quantidades que compõe um todo, como por exemplo porcentagens que juntas completam 100% ou a frequência de determinadas categorias em uma população que somadas dão 1. Se ajustam bem a variáveis qualitativas nominais. Um exemplo é a Figura 1.5, que mostra o conteúdo da Tabela 1.3. Devido a sua simplicidade não são recomendados para dados mais complexos, com muitas categorias ou multidimensionais. Podem ser utilizados para representar variáveis

qualitativas ordinais, mas somente para evidenciar o volume de cada categoria pois não passam a noção de ordem necessária.

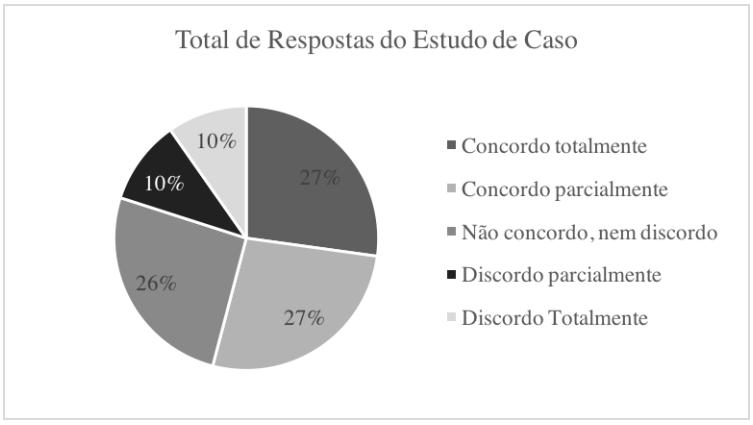


Figura 1.5. Gráfico relativo aos dados da Tabela 1.3.

Embora o gráfico de pizza seja bastante representativo e simples, seu uso é bastante restrito. Observe a Figura 1.6 praticamente indecifrável e outras representações dela que podem ser vistas nas Figuras 1.3 e 1.8, de forma mais clara e significativa.

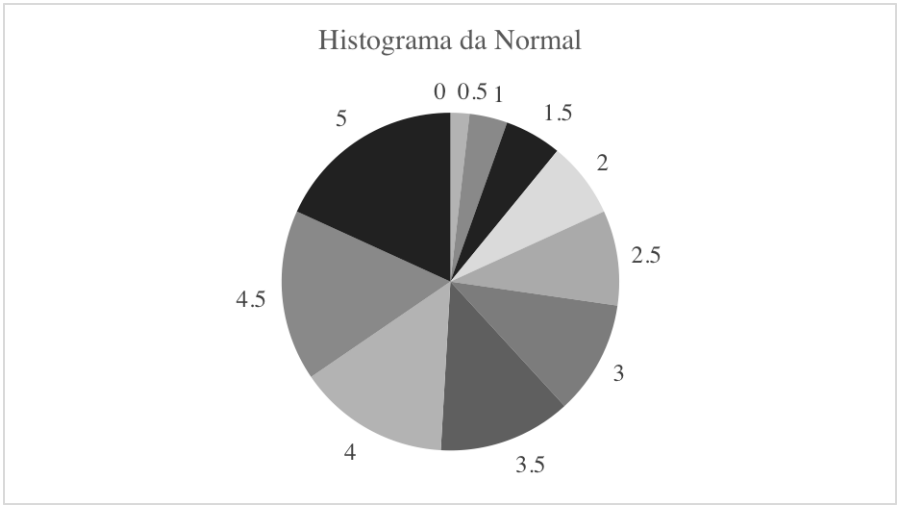


Figura 1.6. Gráfico da distribuição normal. Mesma informação das Figuras 1.3 e 1.8

1.4.3 Gráficos de Teia

Esse tipo de gráfico apesar de não ser comum, tem como principal característica destacar blocos e seus vieses em variáveis multidimensionais. Podemos observar na Figura 1.7 a distribuição da porcentagem dos alunos que conhecem as linguagens de programação citadas.

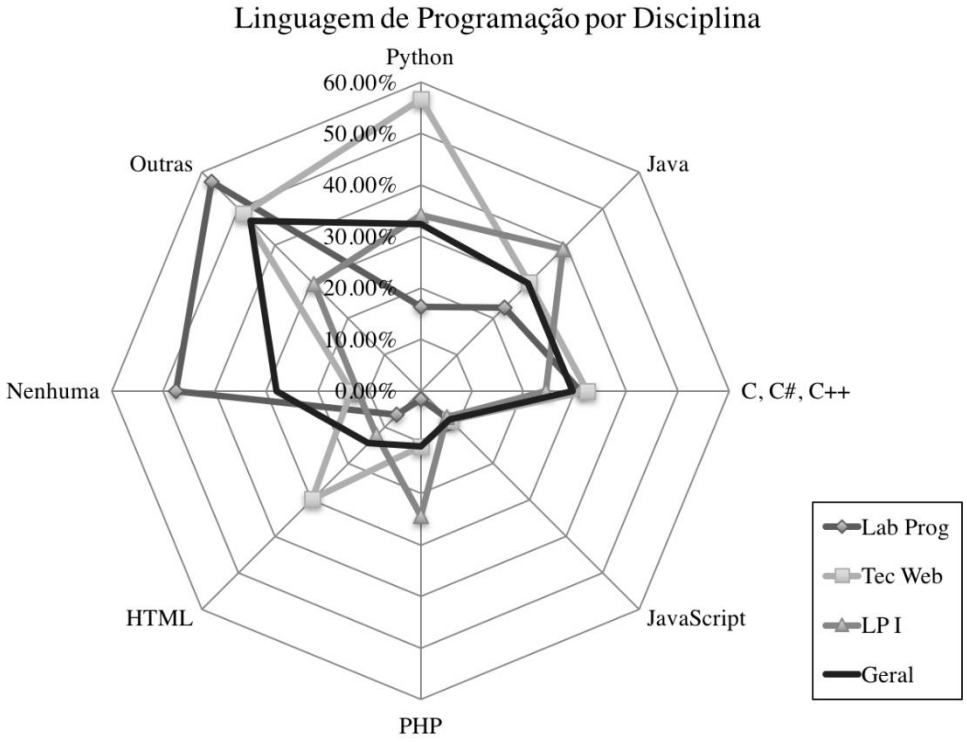


Figura 1.7. Gráfico de Teia

Podemos analisar as variáveis desse gráfico, devemos observar as seguintes características sobre cada polígono formado:

1. Área: Quando maior a área, maiores os valores relacionados à variável;
2. Forma: Quando mais irregular a forma do polígono, mais irregular serão os valores de cada variável;
3. Centralização: O deslocamento do polígono para um lado indica que os números seguem uma tendência de localização em algumas categorias.

Note ainda que se as categorias do gráfico forem cíclicas ou "circulares" mais ajustado e representativo o gráfico será. Por exemplo: meses do ano, alinhamento político, etapas de desenvolvimento de software em cascata, etc.

A análise desse tipo de gráfico pode não ser tão óbvia para quem não está familiarizado com ele. Dessa forma, dependendo do público alvo, é válida a complementação com uma análise descritiva. Da análise específica do gráfico da Figura 1.5 podemos concluir:

1. Analisando a área de cada forma, concluímos que os alunos da disciplina de Linguagem de Programação I (linha com triângulos) por exemplo sabem menos linguagens do que os de Tecnologia Web II (linha com quadrados).
2. Os polígonos cobrem pouco os eixos das linguagens Javascript, C, C# e C++ e PHP, indicando que não há muitos alunos que conhecem essas linguagens.
3. Python é a linguagem mais popular entre os alunos.
4. Existe uma diversidade muito grande de conhecimento de linguagens pois os polígonos cobrem consideravelmente o eixo de "outras linguagens".

5. Pelo mesmo motivo do item anterior, concluímos que muitos estudantes não conhecem nenhuma linguagem. Principalmente na disciplina de Linguagem de Programação.

Como as categorias desse gráfico não são cíclicas, houve uma tentativa de agrupá-las por caraterísticas em comum, para que assim a localização das formas torne o gráfico mais informativo.

1.4.4 Gráficos de Barras

Gráficos de barras são bastante versáteis. Embora o uso mais comum seja para representar variáveis qualitativas ordinais, eles podem ser adaptados tanto para variáveis qualitativas nominais mais simples quanto para variáveis quantitativas mais complexas. Em estatística a ferramenta de histograma é tradicionalmente um gráfico de barras (Figura 1.8).

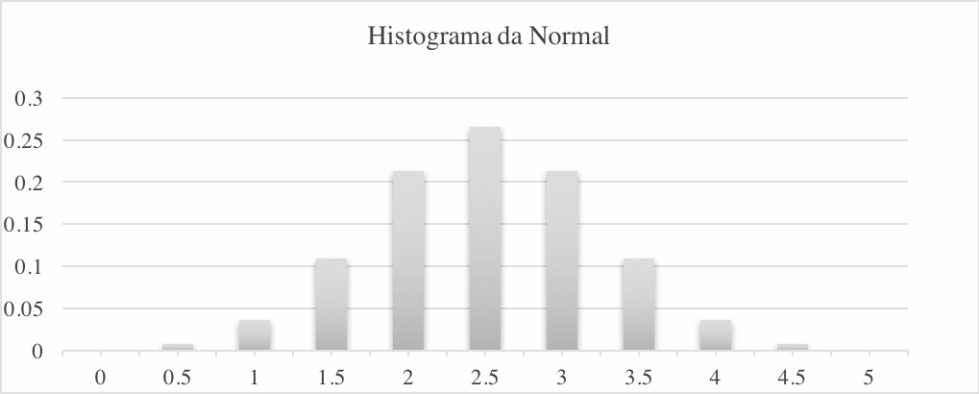


Figura 1.8. Histograma da Distribuição Normal com média 2,5 e desvio padrão de 0,75

Por ser um gráfico bastante intuitivo, quanto utilizados com variáveis bidimensionais, é possível preservar essa característica com organizações adequadas ao que se deseja representar. Dentre os diferentes gráficos de barras bidimensionais que o Excel fornece, duas são bastante diferentes entre si: Valores grupos de barras e barra empilhadas.

Na Figura 1.9 temos um exemplo de gráficos com os valores de variáveis agrupadas. Esse é um gráfico que traz muita informação conjunta e pode tornar-se confuso. Ainda assim é um gráfico adequado para mostrar variáveis bidimensionais em que uma das categorias têm aproximadamente o mesmo comportamento para todas as observações. Assim a primeira informação percebida é o comportamento geral dos grupos de categorias da variável que possui maior variação de valores e em cada agrupamento teremos variações menores da segunda variável. No exemplo podemos observar os mesmos pontos do gráfico da Figura 1.7, com as linguagens agrupadas por categorias, e dentro dela, os valores dos cursos. Na Figura 1.10 temos um gráfico bastante similar ao da Figura 1.9, porém com variáveis de comportamentos menos uniformes e a compreensão bastante comprometida.

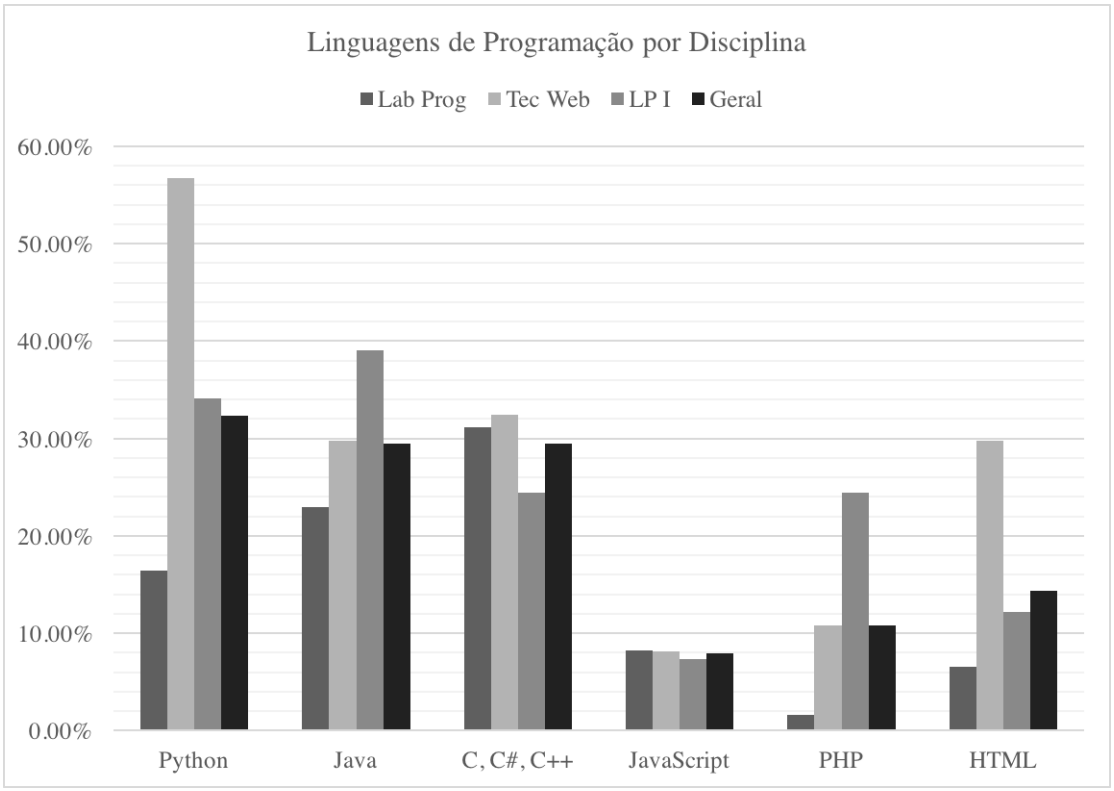


Figura 1.9. Gráfico de barras bidimensional - variáveis de comportamento semelhante agrupadas.

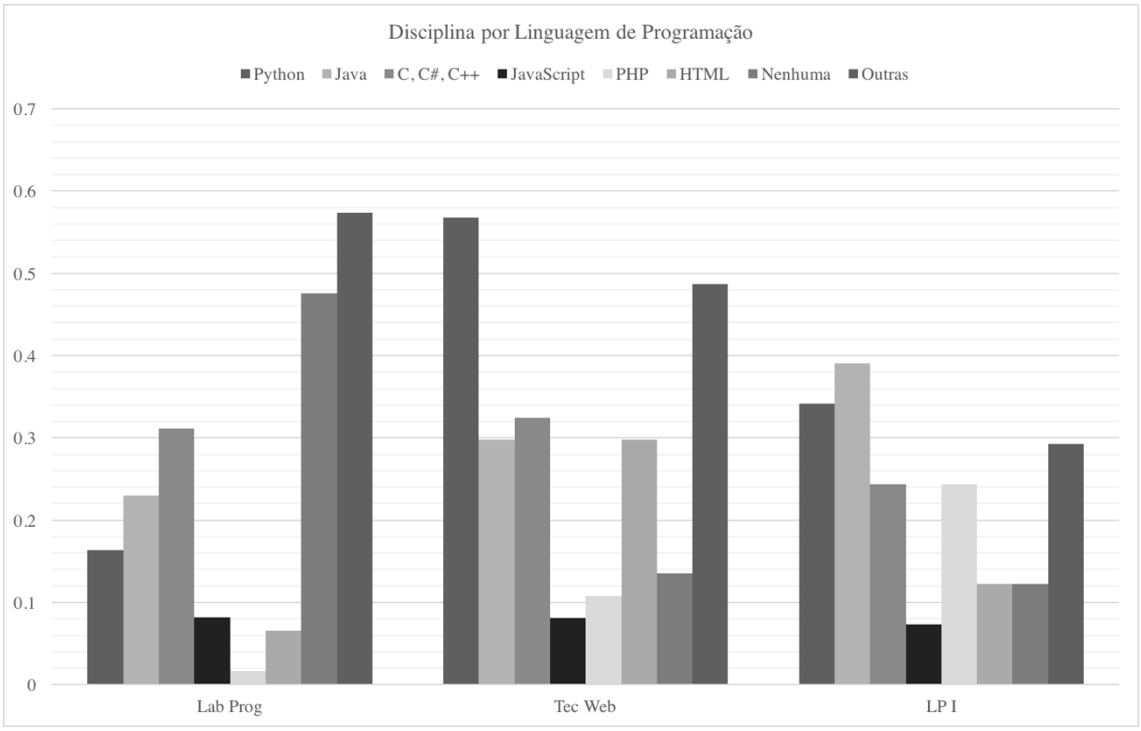


Figura 1.10. Gráfico de barras bidimensional variáveis de comportamento diferentes agrupadas

Na Figura 1.11 temos um outro tipo de organização chamada colunas empilhadas, que tira vantagem do fato de que cada uma das colunas tem as mesmas categorias que somadas representam um todo (100%).

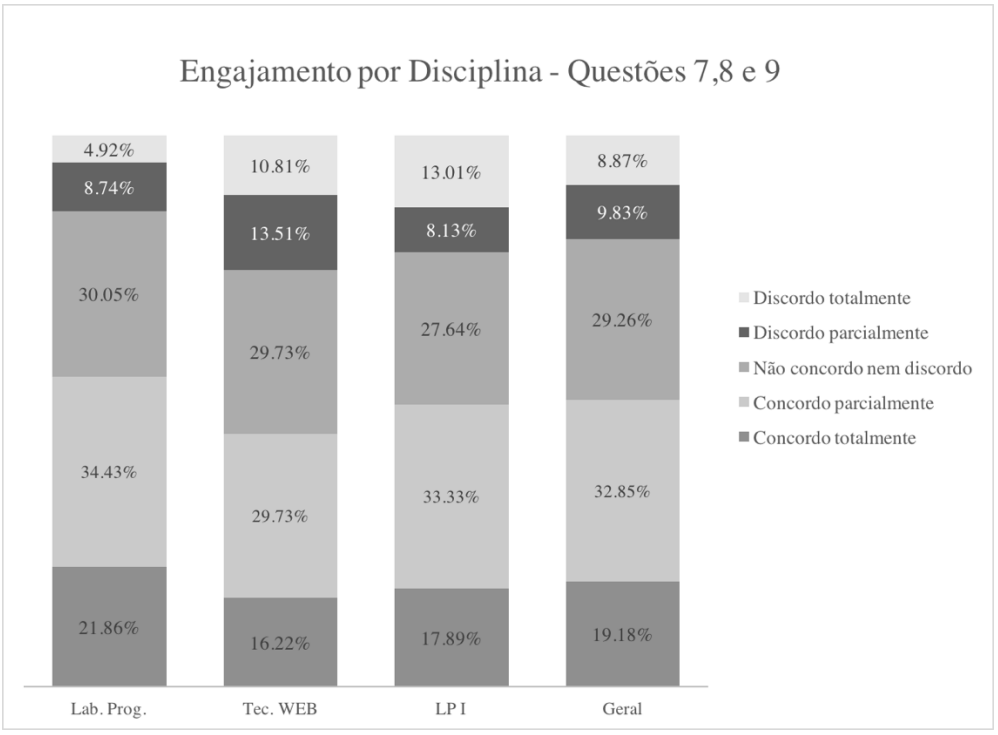


Figura 1.11. Gráfico de barras multidimensional agrupado por categorias que somam 100%

O gráfico da Figura 1.11 estrutura melhor a informação, porém pode dar a falsa ilusão que cada agrupamento tem o mesmo número de elementos, o que não é necessariamente verdade, como no próprio exemplo. No estudo de caso, esse é um gráfico complementar, que deve ser apresentado após outro com a quantidade de alunos por disciplina, como o da Figura 1.12. Note como é a percepção do gráfico da Figura 1.11 quando está sozinho ou acompanhando a Figura 1.12.

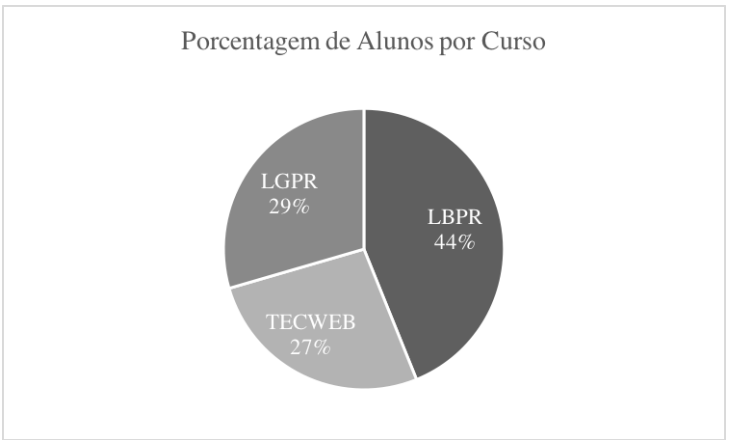


Figura 1.12. Porcentagem de alunos por curso. Esse gráfico contém um contexto necessário para uma melhor compreensão do gráfico da Figura 1.9

Um outro exemplo de colunas empilhadas pode ser visto na Figura 1.13. Apesar da informação da quantidade relativa de cada coluna ter se perdido, a composição de cada uma das barras recebe bastante destaque. Novamente esse modelo de gráfico seria melhor compreendido se fosse o complemento de um gráfico com o número de elementos de cada uma das colunas.

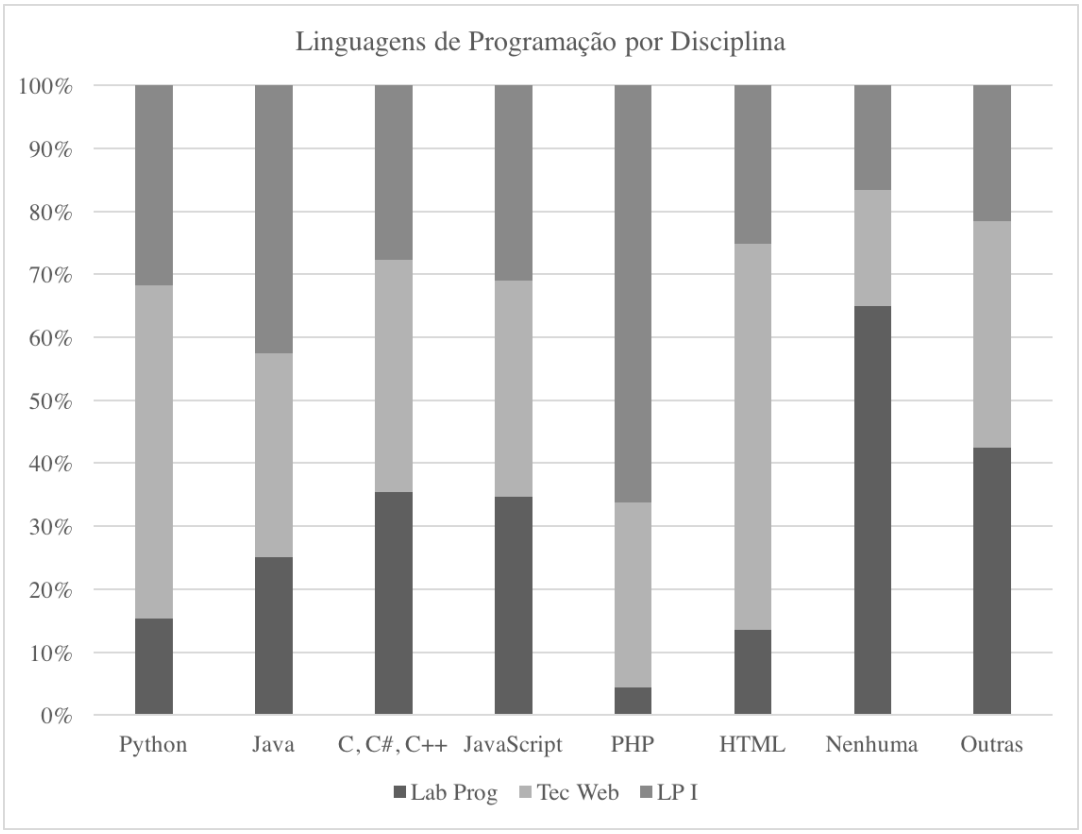


Figura 1.13. Gráfico de colunas empilhadas. A informação de quantidade em cada coluna se perde porém existe um grande destaque para as composições das colunas

Uma escolha também possível é o gráfico de colunas empilhadas em que cada coluna representa também a quantidade relativa entre as categorias como na Figura 1.14. Note que, no entanto, a porcentagem empilhada em cada coluna perde destaque.

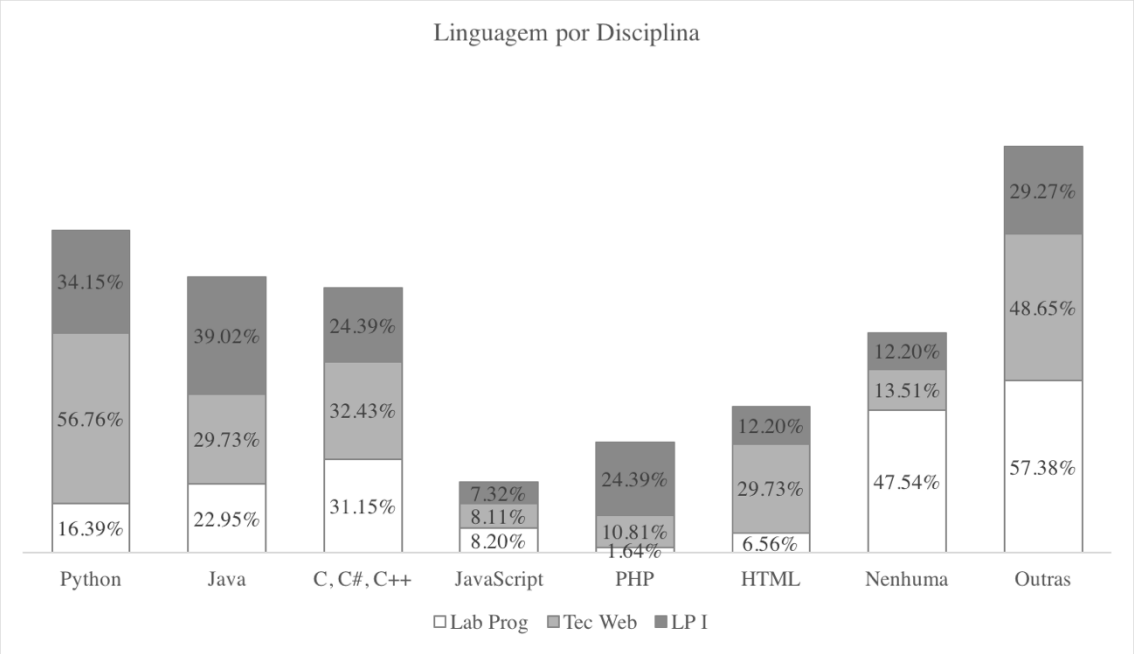


Figura 1.14. Gráfico de colunas empilhadas relativo. A quantidade total de cada coluna é exibida em troca de menos destaque das porcentagens que compõe cada coluna.

1.5. Conclusões

Neste trabalho foram revistos os conceitos mais comuns de estatística sob um enfoque prático e intuitivo para que, considerando-se alguns detalhes como propriedades matemáticas e adequação ao tipo de variável, seja possível obter o máximo de informação a partir de poucas medidas.

A base de dados utilizada foi proveniente do estudo de caso sobre a percepção de estudantes na utilização do desenvolvimento de jogos (perfil, motivação, engajamento, proatividade) em disciplinas de primeiro semestre de um curso de computação.

Utilizando-se as definições de medidas de tendência centrais, foram analisados vários exemplos que ilustraram não somente como trabalhar diferentes tipos de dados, mas também como interpretar corretamente e em conjunto as medidas de média, mediana e moda, levando em consideração todos os detalhes e consequência das fórmulas utilizadas para obtê-las. Também foram abordados erros comuns de interpretação, como confundir o papel de cada uma das medidas.

As medidas de dispersão completam as medidas de tendência central informando como acontece o espalhamento das observações em torno do centro estabelecido. Normalmente utiliza-se apenas uma das medidas, mas elas podem ser analisadas em conjunto para um panorama mais completo. Considerando a obtenção do desvio médio através do módulo das diferenças e o desvio padrão e variância através do quadrado das diferenças, tem-se uma interpretação de maior profundidade.

A última medida analisada sobre a dependência de variáveis multidimensionais foi o coeficiente de correlação. A análise sugerida foi um mapa de cruzamento de todas variáveis aleatórias de interesse e como tirar algumas conclusões das concentrações dos valores das correlações.

A mais poderosa ferramenta de interpretação de dados estatísticos, no entanto, são os gráficos. Há inúmeros modelos de gráficos e alguns modelos foram discutidos na seção 1.4 considerando sempre o tipo de variável e o que se deseja destacar.

Como consideração final, é importante ressaltar que a análise numérica e estatística dos dados é apenas uma ferramenta de auxílio para a elaboração de teses e confirmação de hipóteses, porém nada substitui o conhecimento sobre o tema que se deseja estudar.

Referências

Bussab. W. O., Morettin, P. A, Estatística Básica, Saraiva, 2013.

Couceiro, M.R.; Veloso, A.I.; Papastergiou, M.; Kordaki, M. Design of a Computer Game for an Information Technology Class. Revista de Ciências e Tecnologias de Informação, vol. 1, n.14, 2011 p. 1-11.

Eseryel, D.; Law, V.; Ifenthaler, D.; Gen, X.; Miller, R. An Investigation Interrelationships between Motivation, Engagement and Complex Problem Solving in Game-based Learning. In: International Forum of Educational Technology $ Society, vol. 1, n.1, 2014.

Everitt, B. S., Skrondal A. The Cambridge Dictionary of Statistics, Cambridge, 2002.

Ferreira, A. B. D. H. (2004) Novo dicionário Aurélio da língua portuguesa, In Novo dicionário Aurélio da língua portuguesa, 2004.

Krajcik; J.S.; Blumenfeld, P.C. Project-Based Learning, Cambridge University Press, 2010, pp 317-334. https://doi.org/10.1017/CBO9780511816833.020

Likert, R. (1932) "A technique for the measurement of attitudes", Archives of Psychology, 22(140), p. 1-55.

Loyens, S. M. M., Kirschner, P. & Paas, F. (2011). Problem-based Learning. In K. R. Harris, S. Graham & T. Urdan (Eds.), APA Educational Psychology Handbook: Vol 2 (p. a). Washington: American Psychological Association.

Martins, V. F., Eliseo, M. A., Omar, N., Castro, M. L. A., Corrêa, A. G. D. (2018) "Using Game Development to Teach Programming". In: Nunes, F. B. "Handbook of Research on Immersive Digital Games in Educational Environments", IGI Global.

Moita Neto, J. M. (2015). Estatística multivariada: uma visão didática-metodológica.

Saraiva, J. C. C., Victer, E. D. F., and Siqueira, A. S. (2017) "SISTAT: Ferramenta Computacional como Proposta para o Ensino da Estatística Descritiva", Revista de Educação, Ciências e Matemática, 7(1).

Silva, T.R.; Medeiros, T.; Medeiros, H.; Lopes, R.; Aranha, E. Teaching-learning of programming: a systematic literature review. Brazilian Journal of Computer in Education, vol. 23, n. 1, 2015, p.182-196

Weisberg, H. F. (1992) "Central Tendency and Variability", Sage University Paper Series on Quantitative Applications in the Social Sciences, ISBN 0-8039-4007-6 p.2

Apêndice A

O questionário dos estudantes (a) era, basicamente, composto por questões de múltipla escolha (Escala Likert (1932)), com exceção das questões 1, 13, 14, 15 e 16 que tinham outros padrões de respostas e as questões 2 e 17 que eram questões abertas para que os alunos pudessem expor sua percepção de maneira mais aprofundada. Já a entrevista com as professoras (b) das disciplinas foi composta por questões abertas. O questionário dos estudantes e a entrevista com professoras ficaram disponíveis para resposta por duas semanas. Os alunos e as professoras foram convidados a responderem livremente, sem a identificação dos participantes.

Tabela A.1. Questionário de Satisfação dos estudantes

QUESTÃO	POSSÍVEIS RESPOSTAS	OBJETIVO
1. Qual é o nível de conhecimento em programação antes de iniciar a disciplina?	Não tinha nenhum conhecimento; Iniciante; Intermediário; Avançado; Expert	Perfil
2. Quais linguagens de programação você já conhecia antes de iniciar a disciplina?	Questão aberta	Perfil
3. Considero que aprendi (ou aprendi mais) programação nesta disciplina para aplicação nos exercícios e/ou projetos.	Escala Likert de 5 pontos	Motivação
4. Eu já gostava de programar antes de fazer essa disciplina.	Escala Likert de 5 pontos	Perfil
5. Passei a gostar mais de programar após fazer essa disciplina.	Escala Likert de 5 pontos	Motivação
6. Eu acho que os projetos que desenvolvi nesta disciplina são mais complexos que os de outras disciplinas.	Escala Likert de 5 pontos	Opinião sobre a disciplina
7. Considerando meu(s) projeto(s) com jogos, senti maior facilidade em lidar esta complexidade do(s) projeto(s) por causa deste contexto de jogos.	Escala Likert de 5 pontos	Engajamento
8. Me dediquei mais nas aulas presenciais desta disciplina do que em outras disciplinas.	Escala Likert de 5 pontos	Engajamento

9. Dediquei mais horas de estudo em casa, ou nos intervalos de aula para essa disciplina do que para outras disciplinas.	Escala Likert de 5 pontos	Engajamento
10. O uso de jogos em projetos nesta disciplina me motivou a me esforçar mais durante a aula do que normalmente me esforçaria.	Escala Likert de 5 pontos	Motivação
11. Preferi fazer projetos envolvendo jogos em comparação aos demais exercícios da disciplina.	Escala Likert de 5 pontos	Motivação
12. Prefiro desenvolver projetos que envolvam jogos do que projetos de outros gêneros, como sistemas de cadastros, inventários, estoques, problemas matemáticos, folha de pagamento, etc.	Escala Likert de 5 pontos	Motivação
13. Quais as principais dificuldades encontradas durante o desenvolvimento do jogo?	A linguagem de programação utilizada; Compreensão das regras do jogo; Tempo disponível para o desenvolvimento; Tópicos da disciplina; Nenhum; Outros	Opinião sobre a disciplina
14. Para o desenvolvimento do jogo, além do que foi aprendido em aula, busquei mais informações e funcionalidades para tornar o jogo mais completo.	Sim, várias vezes; Sim, algumas vezes; Não	Proatividade
15. Aponte aspectos positivos do uso de jogos na disciplina.	Aumento da motivação; Crescimento na disciplina; Interação com os membros do grupo; Satisfação pessoal em finalizar	Opinião sobre a disciplina

	um projeto complexo; Nenhum; Outros	
16. Aponte aspectos negativos do uso de jogos na disciplina.	Não gosto de jogos; Tempo para o desenvolvimento do jogo; Dificuldades no desenvolvimento do jogo; Considerei a aula chata e desestimulante; Nenhum; Outros	Opinião sobre a disciplina
17. Este espaço é reservado caso você queira deixar algum comentário ou sugestão.	Questão aberta	Opinião sobre a disciplina

Tabela A.2. Entrevista com professoras das disciplinas

QUESTÕES
Qual(is) disciplinas você ministrou utilizando jogos no semestre de 2017/1?
Você já ministrou disciplinas utilizando jogos em outros semestres? Quais?
O que o(a) levou a utilizar jogos nas disciplinas?
Qual sua opinião sobre o uso de jogos na(s) sua(s) disciplina(s)?
Quais os pontos positivos do uso de jogos na(s) sua(s) disciplina(s)?
Quais os pontos negativos do uso de jogos na(s) sua(s) disciplina(s)?
Houve mais aprovações? Maiores médias?
Você sentiu os alunos mais empolgados em resolver os projetos usando jogos? Conte um pouco sobre essa experiência.

5. Capítulo 5

Autores:

Adriana Porto Proença
Universidade Federal de Uberlândia (UFU)
email: adrianaportodesigner@gmail.com

Renato Aquino
Universidade Federal de Uberlândia (UFU)
email: profrenatolopes@gmail.com

Alexandre Cardoso
Universidade Federal de Uberlândia (UFU)
email: alex.cardoso.ufu@gmail.com

Pollyana Notargiacomo
Universidade Presbiteriana Mackenzie
email: pollyana.notargiacomo@mackenzie.br

Capítulo

5

Usabilidade no contexto de Ambientes Virtuais para Educação ou Treinamento

Adriana Porto Proença, Renato Aquino, Alexandre Cardoso, Pollyana Notargiacomo

Abstract. *The Usability Test is a technique for evaluating the facility of using interfaces to make user experiences more relevant, friendly, and meaningful. When applied to Virtual Reality (VR) systems aimed at learning, they aim to reduce cognitive overload so that the user's focus may be on learning a content or acquiring a skill, or even training a particular procedure. In this chapter we will conceptualize the characteristics of the environments that will be measured for Education and Training and presented two usability assessment processes: Heuristic Assessment and Questionnaire Evaluation. The usability measures of these systems serve as tools to evaluate, correct and improve the interaction between users, machine and software.*

Resumo. *O Teste de Usabilidade é uma técnica para avaliar a facilidade de uso de interfaces de forma a tornar as experiências do usuário mais relevantes, amigáveis e significativas. Quando aplicado a sistemas de Realidade Virtual (RV) voltados à aprendizagem propõem-se a reduzir a sobrecarga cognitiva para que o foco do usuário possa ser a aprendizagem de um conteúdo ou a aquisição de uma habilidade, ou mesmo o treinamento de um determinado procedimento. Neste capítulo serão conceituadas as características dos ambientes que serão mensurados voltados à Educação e ao Treinamento e apresentados dois processos de aferição de usabilidade: Avaliação Heurística e Avaliação por Questionário. As medidas de usabilidade desses sistemas servem de ferramenta para avaliação, correção e melhoria da interação entre usuários, máquina e software.*

1.1. Introdução

Os Ambientes Virtuais baseados na tecnologia de Realidade Virtual (RV) possuem um caráter exploratório e interativo baseado na instrumentalização dos sentidos do usuário, instituindo uma aprendizagem experimental e ativa, com maior potencial de visualização e representação. A partir disso, é necessário ter métodos de avaliação não só da tecnologia envolvida (o artefato) como também da qualidade da interação. E, nesses ambientes, sem as devidas condições de usabilidade, ou seja, sem as devidas condições de realização de

uma determinada tarefa, a interface pode representar obstáculo para o usuário e implicar em uma sobrecarga cognitiva.

O presente texto aborda, então, aspectos cognitivos associados às questões de interação e fornece métodos de Avaliação de Usabilidade direcionados a interfaces com conteúdos voltados para aprendizagem suportados pela tecnologia de Realidade Virtual com ênfase em Educação e Treinamento.

1.2. Características de Ambientes para Educação e Treinamento

Segundo Richards et al. (2012), quando ambientes baseados em tecnologia de RV associam-se a conteúdo instrucional, eles se tornam o que se denomina de Ambiente Virtual de Aprendizagem (AVs 3D). E, quando estes ambientes são voltados para aspectos relacionados com processos de interação educacional, possibilitando na aprendizagem fazer análises e refletir sobre o foco do estudo, esses ambientes são denominados como Ambientes Virtuais com Ênfase em Educação (AVEd). Já no caso do Ambiente Virtual de Aprendizagem com Ênfase em Treinamento (AVTr), estes devem possibilitar um aprender específico com tempo determinado, com o objetivo de desenvolver habilidades para determinada tarefa, dentre as quais podem se destacar questões de memória, sequência e aspectos de destreza e a repetição de comandos. Segundo Pass (2010), estes ambientes referem-se à identificação e superação de lacunas pertinentes ao desempenho de colaboradores no âmbito organizacional, bem como à preparação destes para funções diferenciadas ou ainda para adaptação decorrente da inserção de novas tecnologias no cotidiano de trabalho. Todos esses grupos de características que diferenciam AVEd e AVTr, podem ser observados resumidamente, na Tabela 01.

Tabela 1. Categorias de características, adaptadas da tabela original de Housell et al. (2008), alteradas para agrupar características complementares nos aspectos de forma, objetivo e navegação

Grupo	Característica	Tendência dos AVEd	Tendência dos AVTr
Conteúdo	Foco	Valores/Abstrações/Visões	Instruções/Operações
	Forma	Teoria/Conceito	Prática/Procedimento
		Definições/Comparações	Instrução/Sequências
	Conhecimento	Formal	Experiência
		Curricular	Técnico
Modelo Pedagógico	Objetivo	Percepção	Destreza
		Processo/Formação	Capacitação
		Construção do conhecimento	Habilidade/Comportamento
	Aprendizagem	Reflexão/Tomada de decisão	Ação/Técnicas
		Construcionismo/Sócio-Interacionismo	Instrucionismo/Behaviorismo
	Procedimento	Explicação	Comandos/Ordens
		Visualização	Informação/Dado

		Variado	Repetitivo
Comunicação	*Feedback*	Abrangente/Discursivo	Específico/Direto
	Colaboração	Multiusuário	Monousuário
	Navegação	Liberdade/Exploração	Direcionamento/Orientação
	Comportamento	Geral/Aproximado	Específico/Fidedigno
	Grafismos	Caricato	Realista
	Percepção	Sentidos Variados	Sentido Específico
	Cadência	Eficiência/Controlada	Eficácia/Real
	Dificuldade	Modelagem do Aprendizado	Modelagem do Fenômeno
Avaliação	Estratégia	Contínua	Final
		Processos Mentais	Processos Manuais
	Resultado	Entendimento	Condicionamento

1.3. Design Centrado no Usuário

O Design Centrado no Usuário é uma metodologia de projetos para concepção, desenvolvimento e avaliação de artefatos e serviços com foco nos usuários, considerando suas necessidades, limitações, expectativas e características e na realização de tarefas que dependem das funcionalidades do sistema em uso.

Essa metodologia propõe a utilização de uma abordagem interativa, ou seja, baseada na incorporação de um conjunto quase sequencial de tarefas no ciclo de desenvolvimento do produto. Essa metodologia é desenvolvida por meio de uma combinação de métodos investigativos (por exemplo, pesquisas e entrevistas), criativos (por exemplo, *brainstorming*) e ferramentas. Agni (2016) aponta que a abordagem de design centrada no usuário consiste na implementação de um processo iterativo, que geralmente consiste em 4 fases distintas (Tabela 2).

Tabela 2. Fases da Abordagem do Design Centrado no Usuário

Fases	Descrição
Identificar requisitos	Levantar necessidades e entender os pontos de conflitos dos usuários poe meio de pesquisas, observações e entrevistas.
Criar soluções alternativas	Indicar hipóteses de soluções para as necessidades levantadas.
Construir protótipos testáveis	Tirar as ideias do papel e criar modelos testáveis do que pode vir a ser o produto final.
Avaliação	Levar os protótipos para testes com usuários, colhendo os *feedbacks* sobre o que funciona e o que pode melhorar.

Logo, a abordagem deste estudo é sobre as questões relativas às formas de se avaliar e de se conceber, por meio de diferentes métodos, a interação em AVs 3D, de

forma a buscar soluções para melhorar a experiência do usuário e aspectos de cognição humana nesses sistemas.

1.4. Teoria da Carga Cognitiva

Nos processos de interação em AVs 3D, caracterizados pela imersão e envolvimento de um usuário em um mundo artificial, pode-se simular condições reais ou criar condições de um novo mundo. Esta interação torna-se mais envolvente quando há o acionamento das diversas funções cognitivas, dentre elas os processos codificados de ver, ouvir, falar, e as diversas interações do usuário no ambiente virtual.

Segundo Morris e Maisto (2004) a estrutura cognitiva humana pressupõe a existência de três subsistemas distintos: A Memória Sensorial (MS), A Memória de Trabalho (MT) e a Memória de Longo Prazo (MLP). As informações captadas pelo registro sensorial MS desaparecem rapidamente se não passarem por novos processamentos. Já a memória de curto prazo, ou MT, que retém e manipula sons e imagens na consciência ativa, é limitada em capacidade e duração, é responsável por processar as informações antes de serem armazenadas na memória de longo prazo; e, por fim, a MLP é que armazena elevadas quantidades de informações previamente adquiridas e as retém por períodos de tempo indefinidos. Segundo Paas, Van Gog e Sweller (2010), a MLP é vista como a estrutura central da cognição humana e os conhecimentos armazenados podem ser descritos como esquemas hierarquicamente organizados, que nos permitem categorizar diferentes problemas e decidir sobre a solução mais apropriada.

Assim, essa estrutura dos processos de memória e cognição humana apresenta-se como uma capacidade de absorver conhecimentos que fazem parte do desenvolvimento intelectual e comportamental. Tal estrutura relaciona-se com todas as funções necessárias para controlar as ações, emoções e pensamentos, sendo fundamental na interação do ser humano bem como nas suas atividades diárias e nos processos de aprendizagem.

Essas capacidades são naturais, mas são também utilizadas em novos contextos e devem ser combinadas com o modo virtual. Assim, em ambientes baseados na tecnologia de RV, quando mal projetados, prejudicam a qualidade da experiência e a interação do usuário e podem, segundo Cybis (2010), sobrecarregar os usuários em três aspectos: perceptivo, física e cognitiva.

Cybis (2010) aponta que a sobrecarga perceptiva acontece quando o usuário encontra diante de si uma interface que, de alguma forma, prejudica a leitura dos seus elementos. Isso pode tornar a percepção da interface uma tarefa cansativa e de difícil compreensão. Em AVs 3D, o uso de metáforas de interação consistentes com o mundo real e apropriadas às funcionalidades dos dispositivos de RV facilita os processos de percepção do usuário com o sistema. Esses processos estão vinculados às diversas técnicas de interação nesses sistemas em relação aos dispositivos que estimulam a aparelhagem perceptiva humana (visão, audição, propriocepção, etc.). Por sua vez, a sobrecarga física, segundo Cybis (2010), advém quando o usuário aciona alguma função do sistema e é prejudicado por um mecanismo demasiado complexo. Um exemplo disso em Ambientes Virtuais suportados pela tecnologia de RV são os chamados *cybersickness* em processos que incluem desorientação espacial, instabilidade postural e até mesmo náuseas. E, por fim, a Sobrecarga Cognitiva é aquela em que Cybis (2010) observa que

ocorre quando o usuário passa a não reconhecer, dentro da interface, os meios necessários para conseguir executar a sua tarefa com eficácia e eficiência.

Quanto à sobrecarga cognitiva, Preece et al. (2005) aponta que é necessária a compreensão desses processos associados à cognição com os procedimentos de concepção das interfaces, de forma a preparar projetos eficientes para que de fato a aprendizagem dos usuários seja efetiva. O designer deve considerar princípios e diretrizes adequadas a cada situação acerca do domínio do problema, dos usuários e das suas atividades. Preece et al. (2005) também apresentam estratégias (Tabela 3) para o design de interação de acordo com os tipos de processos cognitivos e suas descrições que, uma vez adotadas, podem aumentar os processos de cognição humana com o sistema.

Tabela 3. Processos cognitivos e implicações para o design - Adaptado de Preece et al. (2005)

Processos Cognitivos	Descrição	Implicações de Design	Implicações de Design em AVs 3D
Atenção	Processo de selecionar dentre a variedade de possibilidades disponíveis.	Visibilidade da informação. Informações necessárias na interface. Usar imagens e gráficos.	Facilitar a visibilidade da informação de forma a manter os usuários cientes de suas atividades e interações com outros objetos e usuários.
Percepção	Informações (visuais, auditivas, táteis,etc.), captadas pelos aparelhagem perceptiva humana.	Ícones reconhecíveis. Sons e textos legíveis.	Mapear a percepção normal do usuário e a mudança do ponto de vista pelo movimento do corpo, deve ser processado sem demora de forma natural.
Memória	Recordação para agir adequadamente.	Reconhecimento de funcionalidades sem esforço.	Opções de interação visíveis. Facillitar os acessos no espaço virtual por meio de índices de referência ou mini-mapas, de modo que os usuários possam ver o todo.
Aprendizagem	Facilidade de manusear o sistema.	Interfaces com ações intuitivas. Restringir opções para guiar a seleção de ações de forma mais adequadas.	Desenvolver novos ícones 3D para representar conceitos mais reconhecidos e memoráveis.

A partir disso, Preece et al. (2005) propõem que é necessário avaliar os processos cognitivos dos usuários e as implicações para o design das interfaces; na seleção e avaliação da forma apropriada de combinar os diversos tipos de mídia disponibilizados pela tecnologia que fazem parte destes sistemas, e isso pode incluir AVs 3D. Isso porque, nesses ambientes, a interação do usuário com a tecnologia RV procura simular um ambiente mais real e imersivo, em que o usuário adquire o conhecimento por meio da utilização dos seus sentidos, de tal forma que esses elementos potencializam e contribuem para uma aprendizagem mais significativa.

Portanto, deve-se observar quais recursos precisam ser utilizados para a transmissão dos conteúdos nesses ambientes de acordo com o processo cognitivo humano. Ou seja, pode-se concluir que os elementos multissensoriais bem como os objetos de interação presentes nos AVs 3D devem ser interpretados pelo usuário e, consequentemente, demandam grande esforço mental.

Quando essa quantidade de informações excede a capacidade cognitiva para lidar com as várias informações, o usuário pode se sentir sobrecarregado e até mesmo abandonar a tarefa. Por isso, existem métodos de avaliação de usabilidade, como apoio na verificação da qualidade da eficiência, eficácia e satisfação de uso de uma interface.

1.5. Usabilidade para AVs 3D

Segundo a norma ISO 9241-11:1998, a usabilidade é a "medida em que um produto pode ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso especificado". Nielsen (1994) aponta que a usabilidade é um processo e é composta por diferentes variáveis, logo, não existe uma única solução ou regra aplicável. E, segundo o autor, a usabilidade de um sistema pode ser medida quanto à facilidade de aprendizagem, eficiência, facilidade de memorização, segurança e satisfação dos usuários. Preece et al. (2005) apontam que, em função das oportunidades ocasionadas pelo avanço tecnológico, surgem outras metas a serem observadas durante o desenvolvimento sistemas, o que inclui, por exemplo, interfaces interativas, como AVs 3D, objeto deste capítulo. Segundo Preece et al. (2005), novas tecnologias propocionam o aumento das áreas de aplicação dessas interfaces digitais e o design de interação deve considerar, além das metas de usabilidade, as metas decorrentes da experiência do usuário, conforme Figura 1.1.

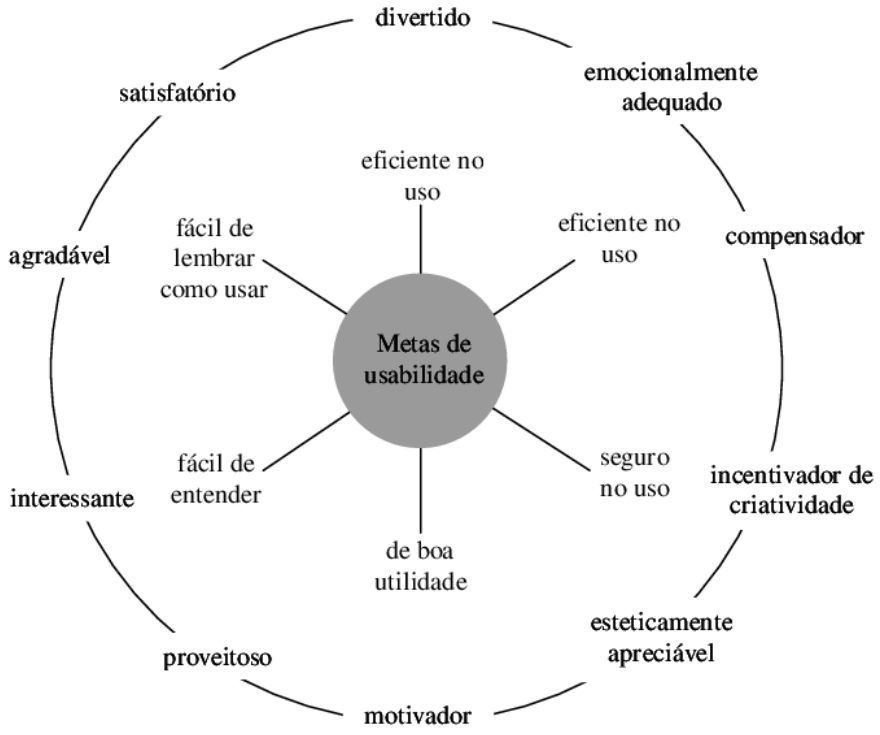


Figura 1.1. Metas de Usabilidade – Preece et al. (2005, p. 40)

Ressalta-se que no círculo interno as metas estão mais diretamente ligadas às diretrizes da norma ISO 9241-11:1998 quanto à eficiência e à eficácia dos sistemas, enquanto no círculo externo, segundo Preece et al. (2005), são mostradas as metas decorrentes da satisfação e experiência do usuário. A partir disso, observa-se que existem técnicas para avaliação de usabilidade tanto dos aspectos objetivos e formais dos sistemas

interativos quanto de aspectos subjetivos que meçam a experiência e o grau de satisfação do usuário.

Nas avaliações de usabilidade em ambientes de aprendizagem tridimensionais há tanto características ligadas aos softwares desses sistemas – questões apontadas por Nielsen (1994) – quanto na avaliação de características referentes à adequação da interface ao sistema tridimensional proposto. São usadas técnicas que objetivam encontrar padrões que melhor se adaptem às interfaces humano-computador e aos sistemas analisados, gerando métricas e evidências necessárias para a tomada de decisão, por exemplo, quanto à aderência e adequação da tecnologia aplicada.

As medidas de desempenho e análise de usabilidade destes sistemas servem de ferramenta para avaliação, correção e melhoria da interação entre usuário e sistema. Neste capítulo são abordadas duas importantes formas de avaliação de usabilidade, a Avaliação por meio de Questionários e Avaliações Heurísticas.

Assim, a usabilidade se aplica a todos os aspectos do sistema com os quais a pessoa pode interagir e deve ser sempre medida relativamente a determinados usuários executando determinadas tarefas.

1.6. Testes de Usabilidade Analíticos e Empíricos

Prates (2003) aponta que os métodos de avaliação de interface diferem entre si em vários aspectos. É preciso entender as diferentes características de cada método para se definir qual deles é o mais apropriado para se avaliar a interface de um sistema em um determinado contexto. A autora faz uma distinção entre os métodos analíticos e empíricos de avaliação usabilidade.

Os métodos de avaliação analíticos são aqueles que, em geral, os avaliadores são especialistas em usabilidade, mas também podem ser consultores de desenvolvimento de software especializados em um determinado estilo de interface. Estas avaliações são assim chamadas por analisarem problemas com o objetivo de fazer recomendações e, a partir disso, melhorar a usabilidade do projeto. Já os métodos de avaliação empíricos envolvem usuários para a coleta de dados. Segundo Prates (2003), estes métodos são apreciados posteriormente pelo especialista para identificar os problemas da interface.

Sendo assim, neste capítulo são descritos dois métodos de usabilidade, tanto analíticos quanto empíricos, que realizem uma aferição da utilização do sistema, focando na eficiência, eficácia (Avaliação Heurística) e satisfação do usuário (Avaliação por Questionários) em interação com AVs 3D com ênfase em Educação e Treinamento.

1.7. Materiais e Métodos

Para atingir o objetivo estabelecido neste capítulo foi delineada uma metodologia desenvolvida em quatro etapas:

- Etapa 01: descrição dos ambientes que serão analisados, descritos na seção 1.2 deste capítulo.

- Etapa 02: descrição do Métodos de avaliações para AVs 3D com ênfase em educação e/ou treinamento: Avaliação por Questionários e Avaliação Heurística.
- Etapa 03: apresentação dos Procedimentos para a realização dos Métodos de Avaliação de Usabilidade.
- Etapa 04: proposta com Questionário ou Lista de Heurísticas de Usabilidade para AVs 3D.

1.8. Métodos de Avaliação de Usabilidade por Questionário

1.8.1. Descrição – Método de Avaliação por Questionário

O método de Avaliação de Usabilidade por Questionários depende de um conjunto de perguntas que devem ser elaboradas de acordo com a funcionalidade do Sistema sob avaliação. É um tipo de avaliação empírica, pois envolve usuários que, após a coleta, deverão ser analisados por especialistas para identificar os problemas da interface.

Rodrigues (2012) aponta que o objetivo de um questionário é estabelecer uma comunicação particular. O pesquisador espera que os respondentes às perguntas propostas possuam certas informações sobre o assunto investigado. É necessário obter tais dados, com o mínimo de distorção possível. Para isso, é possível optar pela utilização de dois tipos de perguntas: as perguntas fechadas e as perguntas abertas.

Rodrigues (2012) aponta que, na pergunta fechada, apresenta-se um número de alternativas para uma determinada questão e solicita-se ao respondente marcar a opção que considerar mais apropriada, ou solicita-se que registre sua resposta em uma escala de acordo com a sua opinião. As perguntas têm que ser suficientemente bem elaboradas para fazer com que as categorias de respostas sejam significativas. A mesma autora observa que, em questionários abertos, solicita-se que os respondentes escrevam suas próprias respostas para as questões levantadas. Portanto, uma resposta aberta não é seguida por qualquer tipo de escolha. Essa resposta tem que ser registrada na íntegra. Este tipo de questionário pode ser útil em situações nas quais o pesquisador não sabe quais assuntos são mais importantes para serem tratados. As perguntas podem ser elaboradas de maneira mais ampla, permitindo que os respondentes destaquem os assuntos que consideram mais relevantes.

Para a presente pesquisa (apresentada como exemplo no contexto deste capítulo), foi utilizado o questionário de avaliação de Silva (2014), sendo as questões adaptadas para AVs 3D. O autor baseou-se na ferramenta Questionnaire for User Interaction Satisfaction (QUIS), projetado para avaliar a satisfação subjetiva dos usuários com aspectos específicos da interface humano-computador (QUIS, 2014).

1.8.2. Procedimento: Método de Avaliação por Questionário

Rodrigues (2012) assinala que o questionário deve funcionar como um diálogo, de modo que cada pergunta revela uma informação sobre os respondentes e provê dados para o pesquisador. A autora ainda observa que é importante prestar atenção em todos os detalhes dos procedimentos e suas particularidades, pois algumas pessoas dão muita

importância para estas. Caso contrário, a ferramenta pode oferecer o risco dos respondentes desconfiarem da sua credibilidade. A possibilidade é ordenar as perguntas e sugerir um passo a passo, indo dos itens mais fáceis para os itens mais complexos. Um primeiro princípio de estruturação é direcionar-se:

- do mais geral para o mais específico;
- do mais simples para o mais complexo;
- do menos delicado para o mais delicado;
- do menos pessoal para o mais pessoal.

Rodrigues (2012) ainda observa que o questionário deve começar com uma introdução que apresente: o propósito da pesquisa; as instituições envolvidas; a qual público-alvo se dirige e porque o participante foi escolhido; o retorno dos resultados da pesquisa como benefícios concretos dos usuários em questão; a importância da participação do receptor; o respeito à confidencialidade da identificação do participante; não há respostas corretas, apenas a opinião de cada participante; as instruções de preenchimento; o tempo de duração de preenchimento e, por fim, as informações de contato do pesquisador.

1.8.3 Questionário para AVs 3D

Silva (2014) apresenta um questionário de avaliação de interfaces para ambientes de tecnologia de RV baseados na ferramenta QUIS (*Questionnaire for User Interaction Satisfaction*), projetado para avaliar a satisfação subjetiva dos usuários com aspectos específicos da interface humano-computador (QUIS, 2014). O autor primeiramente propõe tarefas a serem desenvolvidas por usuários dentro de um dado ambiente imersivo tridimensional e depois aplica o questionário abordando questões a respeito da experiência, da aprendizagem, da interação e satisfação do usuário com o sistema. A seguir está a Tabela 4 com a lista integral deste questionário.

Tabela 4. Formulário de Avaliação de Interface – Silva (2014)

<table>
<tr><td colspan="3">Nome:</td><td colspan="4">Data de Nascimento: __/__/__</td><td>Instituição:</td></tr>
<tr><td colspan="8">Avalie sua experiência em ambientes virtuais 3D:</td></tr>
<tr><td colspan="3">( ) Nenhuma Experiência</td><td colspan="5">( ) Experiência Moderadamente Alta</td></tr>
<tr><td colspan="3">( ) Alguma Experiência</td><td colspan="5">( ) Experiência Alta</td></tr>
<tr><td colspan="3">( ) Experiência Moderada</td><td colspan="5"></td></tr>
<tr><td colspan="2"></td><td></td><td colspan="5">Grau de Concordância</td><td></td></tr>
<tr><td colspan="2">a) Reação ao Sistema</td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr>
</table>

Satisfação em relação à proposta do Protótipo: Utilização para Monitoramento e Controle de Subestações.	*Desinteressante*						*Interessante*
	Tedioso						*Estimulante*
	Difícil						*Fácil*
	Frustrante						*Satisfatório*
	Inadequado						*Adequado*
Comentários:							
b) Tela – Interface de Controle – Menu, Interfaces Alternativas, Janelas		*1*	*2*	*3*	*4*	*5*	
Com relação à utilização:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Difícil						*Fácil*
Com relação à eficiência:	*Pouco*						*Muito*
Com relação à visualização:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Pouco Legível						*Muito Legível*
Com relação à estética:	*Desinteressante*						*Interessante*
Oferece facilidade de aprendizagem:	*Pouco*						*Muito*
Transmite a sensação de Integração ao ambiente 3D.	*Pouco*						*Muito*
Comentários:							
c) Camadas – Rótulos, Leitura e Controle, Envoltória		*1*	*2*	*3*	*4*	*5*	
Com relação à utilização:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Difícil						*Fácil*
Com relação à eficiência:	*Pouco*						*Muito*
Com relação à visualização:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Pouco Legível						*Muito Legível*
Com relação à estética:	*Desinteressante*						*Interessante*
Oferece facilidade de aprendizagem:	*Pouco*						*Muito*
Transmite a sensação de Integração ao ambiente 3D.	*Pouco*						*Muito*

Comentários:							
d) Formas de Navegação – 1ª Pessoa, 3ª Pessoa, Visão Geral e MiniMap		*1*	*2*	*3*	*4*	*5*	
Com relação à navegação:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Difícil						*Fácil*
Com relação à eficiência:	*Pouco*						*Muito*
Com relação à visualização:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
Oferece facilidade de aprendizagem:	*Pouco*						*Muito*
Transmite a sensação de presença durante a utilização.	*Pouco*						*Muito*
Comentários:							
e) Capacidades do Sistema		*1*	*2*	*3*	*4*	*5*	
A velocidade do sistema é:	*Muito Baixa*						*Rápida o Bastante*
O tempo de resposta para a maioria das operações é:	*Muito Longo*						*Rápido o Bastante*
A velocidade em que tela é atualizada:	*Muito Baixa*						*Rápida o Bastante*
A facilidade de operar o sistema depende do nível de experiência – (em relação a sistemas 3D e jogos):	*Nunca*						*Sempre*
É possível realizar tarefas conhecendo poucos comandos:	*Com Dificuldade*						*Com Facilidade*
Comentários:							
f) Aprendizagem do Sistema		*1*	*2*	*3*	*4*	*5*	
Aprender a operar o sistema é:	*Difícil*						*Fácil*
Iniciar o uso é:	*Difícil*						*Fácil*
Aprender a funções avançadas (Formas de Navegação – Visão Geral).	*Difícil*						*Fácil*
O tempo de aprendizagem é:	*Longo*						*Curto*
Explorar uso por tentativa e erro é:	*Desencorajador*						*Encorajador*
Relembra nomes e uso de comandos é:	*Difícil*						*Fácil*
O número de etapas por tarefa é:	*Excessivo*						*Adequado*

As etapas para completar tarefas seguem uma sequência lógica:	*Nunca*						*Sempre*
A resposta do sistema ao completar uma sequência de etapas é:	*Confusa*						*Clara*
Comentários:							
g) Terminologia e Informações do Sistema		*1*	*2*	*3*	*4*	*5*	
Uso dos termos relacionados à atividade.	*Inconsistente*						*Consistente*
Uso dos termos relacionados à tecnologia.	*Inconsistente*						*Consistente*
Os termos utilizados se relacionam com as tarefas desempenhadas.	*Nunca*						*Sempre*
Os termos utilizados são:	*Ambíguos*						*Precisos*
Lembrança de nomes e uso dos comandos.	*Nunca*						*Sempre*
A posição das interfaces são:	*Inconsistente*						*Consistente*
Instruções para comandos ou funções são:	*Confusas*						*Claras*
O sistema mantém informado sobre o que está sendo feito.	*Nunca*						*Sempre*
Realizar uma operação/função no sistema leva a resultados previsíveis:	*Nunca*						*Sempre*
Duração da espera entre operações/funções do sistema é:	*Inaceitável*						*Aceitável*
Comentários:							
h) Imersão		*1*	*2*	*3*	*4*	*5*	
Sentiu empatia com sistema:	*Pouca*						*Muita*
Sentiu interesse no sistema:	*Pouco*						*Muito*
Sentiu-se envolvido no sistema:	*Pouco*						*Muito*
Sentiu realismo com as ações exercidas.	*Pouco*						*Muito*
Gostou da qualidade gráfica do sistema (Modelagem, Construção da Cena e Interfaces).	*Pouco*						*Muito*
Gostou de utilizar o protótipo:	*Pouco*						*Muito*
Os controles e comandos foram fáceis de utilizar:	*Pouco*						*Muito*
Comentários:							

g) Utilização dos óculos 3D (esterescopia) e GamePad (controle)		*1*	*2*	*3*	*4*	*5*	
Sentiu algum desconforto em utilizar óculos:	*Pouco*						*Muito*
Conseguiu visualizar efeito de profundidade:	*Nunca*						*Sempre*
Com relação às interfaces de controle (menu, janelas, interface alternativa) utilizando o óculos:	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Pouco Legível						*Muito Legível*
Com relação à utilização do GamePad.	*Inadequado*						*Adequado*
	Desinteressante						*Interessante*
	Difícil						*Fácil*
Sentiu algum desconforto em utilizar o GamePad.	*Pouco*						*Muito*
Sentiu facilidade de utilizar o GamePad.	*Pouca*						*Muita*
Comentários:							

É importante observar que o Questionário apresentado pode ser adaptado de acordo com as funcionalidades e os requisitos existentes no AVs 3D sob avaliação. Por exemplo, a questões (g) relacionada a utilização dos óculos 3D poderia ser adaptada a utilização de outros equipamentos como luvas hápticas ou qualquer uso de outro dispositivo especial multissensorial. Outra observação importante é que logo após cada questão objetiva proposta apresenta-se outra seção com local para os comentários dos usuários. Isso possibilita que os usários se posicionem e apresentem suas próprias opiniões, dúvidas ou críticas quanto as questões de usabilidade levantadas.

1.9. Métodos de Avaliação de Usabilidade – Heurísticas de Usabilidade

1.9.1. Descrição do Método de Avaliação por Heurísticas

As ações em Ambientes virtuais baseados em RV necessitam “ocorrer em tempo real, sejam as mais naturais, intuitivas, interativas possíveis e exista a sensação de imersão, seja física, seja mental, através de estimulação dos canais sensoriais” [Pimentel et al. 2008, p. 01]. Essas concepções de envolvimento podem compor também a base para ambientes tridimensionais de aprendizagem, já que é fundamental no processo de instrução comprometimento e envolvimento do aluno com a atividade que está sendo desempenhada. Estes objetivos são conjugados na busca pela qualidade da interação usuário-computador e dados por meio do grau de usabilidade da interface. Dentre os processos de avaliação estão as heurísticas de usabilidade que são métodos analíticos de inspeção, pois utilizam especialistas que examinam as interfaces por meio de uma série

diretrizes à procura por problemas para se manter o projeto consistente com as necessidades de interface do usuário.

Segundo Nielsen (1994), estas avaliações podem ser realizadas por especialistas em usabilidade ou em fatores humanos. E, segundo o autor, sua eficiência depende da capacidade destes avaliadores de reconhecerem problemas de usabilidade. O mesmo autor aborda estes dez princípios: visibilidade do status do sistema, compatibilidade do sistema com o mundo real, liberdade e controle do usuário, consistência e padronização, prevenção de erros, reconhecimento ao invés de memorização, flexibilidade e eficiência de uso, estética e design minimalista, identificação e resolução de erros, ajuda e documentação. Essa lista de diretrizes pré-estabelecidas propõe recomendações de melhoria adequadas de forma a proporcionar maior interação do usuário com o sistema.

Assim, este capítulo também propõe apresentar quais problemas de usabilidade já foram identificados para ambiente suportados pela tecnologia de RV que contribuam para melhor interação do usuário com tais ambientes. Utilizaremos a lista de usabilidade para AVEd e AVTr presente em Proença et al. (2017), em que foi realizada a compilação da lista com base em uma revisão sistemática sobre heurísticas de usabilidade para ambiente tridimensionais imersivos com ênfase em Educação e Treinamento.

1.9.2. Procedimentos: Método de Avaliação por Heurísticas

Nielsen (1994) propõe que, para realização de avaliações heurísticas, deve haver de 3 a 5 especialistas em usabilidade e que estes avaliem, isoladamente, evitando que os achados de um sejam influenciados pelos do outro, para que, posteriormente, possa haver a comparação dos resultados. Estas avaliações podem ser aplicadas em qualquer fase do desenvolvimento da interface, desde a prototipagem até após a implementação.

Como procedimentos para este tipo de avaliação em AVs 3D, primeiramente, os avaliadores recebem a lista de heurísticas e instruções sobre o procedimento das ações de participação do estudo e disponibilização dos resultados. Em seguida, os avaliadores deverão seguir as etapas:

a) Definir o usuário: compreender a quem se destina a tarefa. Em toda a fase de desenvolvimento do projeto deve ser observado o perfil do usuário final.
b) Definir a tarefa: entender qual a tarefa a ser desempenhada. Para exemplificar AVs 3D, pode-se designar as seguintes tarefas:

    Exemplo 1:

    Navegação: familiarizar-se com o ambiente, realizando passeios pelo Ambiente 3D, alterando o modo de navegação para 3º Pessoa, posteriormente, 1ª Pessoa e, por fim, voltar ao modo visão geral.

    Exemplo 2:

    Manipulação Direta: selecione o objeto, realize a leitura pela janela de informações e, posteriormente, feche a janela.

c) Avaliar o Sistema de acordo com a lista de heurísticas. Neste Capítulo é proposta uma lista apropriada para AVs 3D com ênfase em Educação ou Treinamento.
d) Classificar os níveis de problemas de usabilidade encontrados na interface, ou seja, o grau de severidade de violação das heurísticas apresentadas e quais atitudes tomar (Tabela 05).

Tabela 5. Referência para Avaliação - Nielsen (1994)

Nielse - Grau de severidade	Tipo	Descrição
0	Sem importância	Não afeta a operação da interface
1	Cosmético	Não há necessidade imediata de solução
2	Simples	Problema de baixa prioridade (pode ser reparado)
3	Grave	Problema de alta prioridade (deve ser reparado)
4	Catastrófico	Muito grave, deve ser reparado de qualquer forma.

e) Gerar relatórios. É importante gerar relatórios com informações claras e objetivas. E que valorizem os aspectos positivos da interface e apontando sugestões de melhorias das falhas encontradas.

1.9.3. Heurísticas de Usabilidade para AVEd e AVTr

As heurísticas de usabilidade para AVs 3D com ênfase em educação e treinamento (Tabela 6 e Tabela 7) foram levantadas de acordo com os procedimentos metodológicos realizados na pesquisa de Proença et al. (2017).

Tabela 6. Análise Heurísticas para Educação – AVEd adaptada de Proença et al. (2017) e Nielsen (1994)

Verificação	Gravidade Do Problema
(H1) Feedback e visibilidade do status do sistema. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
1. *Feedback*: o ambiente virtual deve dar aos usuários *feedback* adequado em um prazo razoável (0,1s) e no ritmo do usuário sobre o que está acontecendo no momento.	
Local:	
Problema:	
Solução/Comentário:	
(H2) Compatibilidade do sistema com o mundo real. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2
2. Uma linguagem comum, familiar e de fácil compreensão para o usuário deve ser escolhida.	
Local:	

Problema:	( ) Grave - 3
Solução/Comentário:	( ) Catastrófico - 4
(H3) Controle do usuário e liberdade. *Nielsen (1994)*	( ) Sem importância - 0
3.Usuários capazes de controlar o ambiente e desenvolver estratégias para alcançar o objetivo pedagógico.	( ) Cosmético - 1 ( ) Simples - 2
Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4
Solução/Comentário:	
• (H4) Consistência e padrões. *Nielsen (1994)*	( ) Sem importância - 0
A consistência deve estar presente no uso das cores, escalas, sons 3D e forças tácteis. Elementos da interface devem estar na mesma língua.	( ) Cosmético - 1 ( ) Simples - 2
Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4
Solução/Comentário:	
• (H5) Prevenção ao erro. *Nielsen (1994)*	( ) Sem importância - 0
Itens de ajuda menores podem ser oferecido por meio da interface. Ou por meio de explicações e visualizações. Os usuários não precisam usar um manual para interagir.	( ) Cosmético - 1 ( ) Simples - 2
Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4
Solução/Comentário:	
(H6) Reconhecimento ao invés de recordação. *Nielsen (1994)*	( ) Sem importância - 0
6. Ícones 3D devem representar conceitos que possibilitem ao usuário descobrir rapidamente seu significado facilitando a sua memorização.	( ) Cosmético - 1 ( ) Simples - 2
Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4
Solução/Comentário:	
(H7) Flexibilidade e eficiencia de uso. *Nielsen (1994)*	( ) Sem importância - 0
7. Configurar os ambientes para diferentes níveis de habilidade do usuário. Muitas vezes esta flexibilidade é fornecida com níveis de dificuldade variável.	( ) Cosmético - 1 ( ) Simples - 2
Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4
Solução/Comentário:	
(H8) Estética e design minimalista. *Nielsen (1994)*	( ) Sem importância - 0
8.O Mundo Virtual não deve conter objetos ou ações que raramente são necessários. Painéis de controle devem ser organizados e não estarem sobrecarregados.	( ) Cosmético - 1 ( ) Simples - 2
Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4

Solução/Comentário:	
(H9) Ajuda a reconhecer, diagnosticar e recuperar de erros. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
9. Mensagens de erro devem fornecer ao usuário informações relevantes sobre o problema e guia-los, em linguagem simples, para a resolução.	
Local:	
Problema:	
Solução/Comentário:	
(H10) Ajuda e documentação. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
As informações devem ser facilmente acessíveis no 3D VLE. Os pop-ups ou elementos flutuantes explicativos podem fornecer essas informações sobre objetos ou ações específicas.	
Local:	
Problema:	
Solução/Comentário:	

Tabela 7. Análise Heurísticas para treinamento – AVTr adaptada de Proença et al. (2017) e Nielsen (1994)

Verificação	Gravidade Do Problema
(H1) Feedback e visibilidade do status do sistema. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
1. *Feedback*: realista, o efeito das ações deve ser imediatamente visível e em conformidade com as leis físicas e expectativas de percepção do usuário.	
Local:	
Problema:	
Solução/Comentário:	
(H2) Compatibilidade do sistema com o mundo real. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
2. O design e uso das diferentes ferramentas no mundo virtual devem ser construídas o mais fiel ao modelo real.	
Local:	
Problema:	
Solução/Comentário:	
(H3) Controle do usuário e liberdade. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
3. Os usuários precisam tomar decisões com tempo programado. Suas ações quando em treinamento não podem ser reversíveis.	
Local:	
Problema:	
Solução/Comentário:	
• (H4) Consistência e padrões. *Nielsen (1994)*	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2
4. Produção de estratégias de interfaces de controle com mecanismos de ativação e desativação, com possibilidade de deslocar para qualquer espaço desejado no ambiente virtual.	

Local: Problema: Solução/Comentário:	( ) Grave - 3 ( ) Catastrófico - 4
• ***(H5) Prevenção ao erro.*** *Nielsen (1994)* Metas, procedimentos e objetivos claros devem ser exibidos principalmente por meio de um tutorial em que o usuário é guiado sobre como proceder na operação. Local: Problema: Solução/Comentário:	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
(H6) Reconhecimento ao invés de recordação. *Nielsen (1994)* Ao realizar ações as opções de interação devem estar visíveis. Lugares dentro dos ambientes virtuais devem ser facilmente acessíveis por meio de índices de referência ou mini-mapas, de modo que os usuários podem ver o todo. Local: Problema: Solução/Comentário:	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
(H7) Flexibilidade e eficiencia de uso. *Nielsen (1994)* 7. Os ambientes possuem execução de comandos repetitivos com intuito de aquisição de habilidades específicas para instituir mecanismo de progressão das tarefas. Local: Problema: Solução/Comentário:	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
(H8) Estética e design minimalista. *Nielsen (1994)* 8. O usuário deve agir como se estivesse no mundo real, por isso a interação deve ser o mais natural possível. A interface deve ser a mais fiel ao original. Local: Problema: Solução/Comentário:	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
(H9). Ajuda a reconhecer, diagnosticar e recuperar de erros. *Nielsen (1994)* 9. As mensagens de erro devem fornecer ao usuário informações relevantes sobre o problema, incluindo as ações que levaram ao erro. Local: Problema: Solução/Comentário:	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2 ( ) Grave - 3 ( ) Catastrófico - 4
(H10). Ajuda e documentação. *Nielsen (1994)* As informações devem ser facilmente acessíveis no AVs 3D.Os pop-ups ou elementos flutuantes explicativos podem fornecer essas informações sobre objetos ou ações específicas.	( ) Sem importância - 0 ( ) Cosmético - 1 ( ) Simples - 2

Local:	( ) Grave - 3
Problema:	( ) Catastrófico - 4
Solução/Comentário:	

1.9.4. Exemplos de Ambientes a serem analisados no mini-curso

O intuito desta seção é exemplicar a aplicação das Heurísticas de Usabilidade para AVs 3D de forma a facilitar a compreensão do estudo apresentado neste capítulo. Para exemplificar as propostas de avaliação deste capítulo serão analisados três ambientes tridimensionais baseados na tecnologia de RV. Dois ambientes com ênfase em treinamento e um ambiente com ênfase em educação. E, para elucidar a proposta, serão apresentados os resultados apenas de uma heurística dos AVs 3D apresentados nesta seção. Esses ambientes terão a sua avaliação com a lista completa de Heurísta de usabilidade presentes nas Tabelas 6 e 7.

O primeiro ambiente avaliado (Figura 1.2) é um sistema de RV com ênfase em treinamento aplicado na aprendizagem de normas de segurança em manutenção de redes elétricas urbanas disponíveis na norma NR 10. Esta estabelece os requisitos e condições mínimas de controle e sistemas preventivos, de forma a garantir a segurança e a saúde dos trabalhadores que, direta ou indiretamente, interajam em instalações elétricas e serviços com eletricidade. Moraes (2013) observa que o sistema implementado tem como objetivo auxiliar o usuário a compreender a importância dos equipamentos de segurança no trabalho.

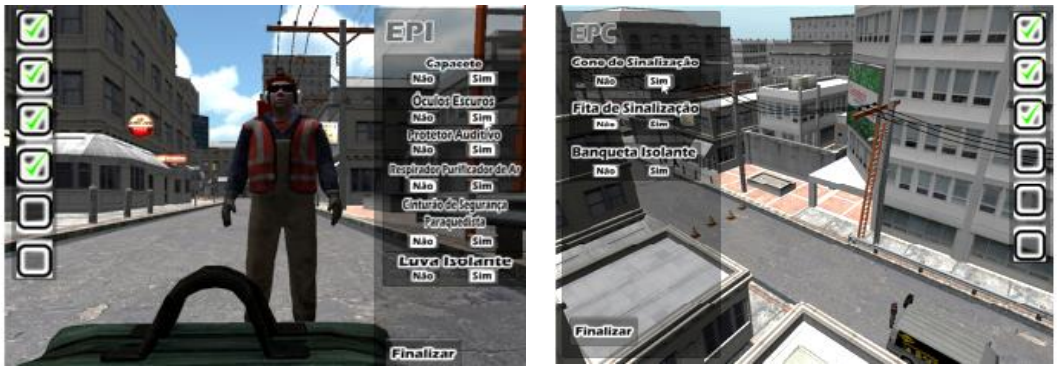


Figura 1.2. Tela de Seleção dos Equipamentos de Proteção Coletiva - Moraes (2013, p. 2)

No ambiente voltado para a aprendizagem sobre A Normas NR10 verificou-se que os componentes da interface de seleção apresentam-se com transparência de 50%, favorecendo o aspecto da integração e criando um padrão de design para o Sistema. Na análise Heurística para este ambiente (Tabela 8) encontra-se o problema e solução sobre as questões referentes a Consistência e padrões próprias para ambientes de treinamento.

Tabela 8. Análise Heurísticas norma NR10

Verificação	Gravidade Do Problema
• ***(H4) Consistência e padrões.*** *Nielsen (1994)*	( ) Sem importância - 0
4. Produção de estratégias de interfaces de controle com mecanismos de ativação e desativação, com possibilidade de deslocar para qualquer espaço desejado no ambiente virtual.	( ) Cosmético - 1 (x) Simples - 2
Local:Interface de Seleção	( ) Grave - 3
Problema: Tipografia ilegível.	( ) Catastrófico - 4
Solução/Comentário:O Sistema comporta-se muito bem com relação a disposição da interface, porém poderia ser escolhida uma tipografia mais legível.	

O segundo ambiente avaliado (Figura 1.3) é um AVs 3D com ênfase em treinamento aplicado na aprendizagem de controle e monitoramento de redes elétrica: o RV–CEMIG. O ambiente possibilita diversas formas de navegação, facilitando a aprendizagem de detalhes físicos que compõem uma subestação de uma concessionária de energia elétrica, além de simular diferentes operações sem comprometer a segurança do usuário e o desempenho do sistema, assim como proporcionar a exploração de alguns equipamentos de forma interativa.

Figura 1.3. Representação virtual de uma Substação Elétrica-

Cardoso et. Al. (2013, p. 3)

No segundo ambiente avaliado o RV–CEMIG verificou-se que o Sistema possui ferramentas como uso de Mini mapa com recurso para tarefas que necessitam de deslocamento rápido e visualização de localização do avatar no AVs 3D. Porém, o sistema ainda necessita de sinalizar claramente os locais de saída e entrada no ambiente e também necessita de restrições de espaço para que o avatar não saia da área da subestação e fique "flutuando" no espaço sem o piso. Na análise Heurística para este ambiente (ver Tabela 9) encontra-se o problema e solução sobre as questões de Reconhecimento ao invés de recordação próprias para ambientes de treinamento.

Tabela 9. Análise Heurísticas RV-CEMIG

Verificação	Gravidade Do Problema
(H6) Reconhecimento ao invés de recordação. *Nielsen (1994)*	( ) Sem importância - 0
6. Ao realizar ações as opções de interação devem estar visíveis. Lugares dentro dos ambientes virtuais devem ser facilmente acessíveis por meio de	( ) Cosmético - 1

índices de referência ou mini-mapas, de modo que os usuários podem ver o todo.	(x) Simples - 2
Local: Interface Gráfica - Cenário	( ) Grave - 3
Problema: Necessita de sinalizar claramente os locais de saída e entrada no ambiente.	( ) Catastrófico - 4
Solução/Comentário: Sinalizar claramente os locais de saída e entrada do usuário no ambiente e restringir apenas para o espaço da área de piso da subestação o deslocamento do usuário.	

O terceiro e último ambiente avaliado (Figura 1.4) é uma interface de RV com ênfase em educação com interações baseadas em estratégias de jogos para a aprendizagem da cor luz: o NColors. Segundo Salustiano (2017), o ambiente utiliza a visualização por meio do óculos Rift em primeira pessoa e ângulo à 360° e as interações do usuário com o Sistema são desenvolvidas pelos controles Rift Touch, que desempenham os mecanismos básicos que são: andar e pegar.

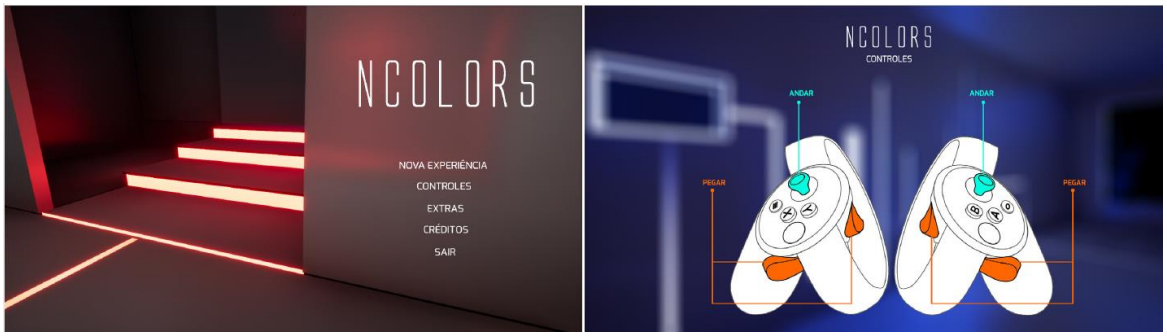


Figura 1.4. Tela inicial e Controles - Salustiano (2017, p. 39)

No ambiente NColors verificou-se que este apresenta a interface (Figura 1.5) mais intuitiva levando o usuário a aprender de forma mais fácil. No ambiente são bem definidos os ícones de interação, por meio da tipografia clara e legível, e seu contraste de cor com o cenário minimalista proposto. Na análise Heurística para este ambiente (Tabela 10) encontra-se o problema e solução sobre as questões de ajuda e documentação.

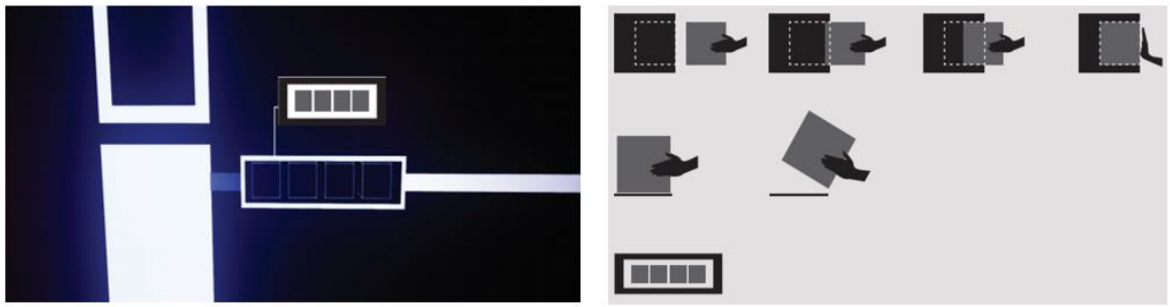


Figura 1.5. Interface e Ícones de ação - Salustiano (2017, p. 40)

Tabela 10. Análise Heurísticas NCOLORS – os autores

Verificação	Gravidade Do Problema
(H10). Ajuda e documentação. *Nielsen (1994)*	(x) Sem importância - 0
As informações devem ser facilmente acessíveis no 3D VLE. Os pop-ups ou elementos flutuantes explicativos podem fornecer essas informações sobre objetos ou ações específicas.	( ) Cosmético - 1 ( ) Simples - 2
Local: Nos cenários, a interface é intuitiva, com gráficos 2D reduzidos (ver Figura-4), assim como as linhas de sinalização iluminadas, direcionam a atenção e percepção do usuário por meio de informações adquiridas pelo ambiente.	( ) Grave - 3 ( ) Catastrófico - 4
Problema: Sem importância	
Solução/Comentário: Os indícios destacam e assinalam os procedimentos a serem realizados pelo usuário, diminuindo a sua sobrecarga cognitiva.	

1.10. Considerações Finais

A tecnologia de RV tem sido amplamente utilizada como importante ferramenta de auxílio no processo pedagógico, podendo proporcionar conteúdos atrativos tanto para as ênfases educacionais quanto para treinamento. Porém, essas ferramentas que auxiliam o aprendizado, em alguns casos, são inseridas sem que haja uma verificação formal de seus benefícios, ou seja, a partir de suposições. Por isso, há necessidade de pesquisas que se apoiem em métodos de avaliação de usabilidade apropriados para ambientes com estes conteúdos singulares, específicos e, como vimos, distintos em várias características, como forma, conteúdo, comunicação e avaliação.

Assim, neste capítulo, foram também abordadas as teorias da carga cognitiva e o design centrado no usuário com o objetivo ressaltar a importância de reduzir as demandas na memória de trabalho dos alunos, para que eles aprendam de forma mais eficaz e satisfatória. E, com isso, ampliar a facilidade de aprendizagem destes ambientes e a melhoria continua dos mesmos.

Referências

Agni, E. (2016) “Etapas do design centrado no usuário”. Disponível em: <https://uxdesign.blog.br/etapas-do-design-centrado-no-usu%C3%A1rio-cec46ae47770>. Acessado em: 10/06/2016.

Cardoso, A., Edgard Lamounier, Gerson Lima, Leandro Mattioli. (2013). “VRCEMIG: A virtual reality system for real time control of electric substations”. In: IEEE Virtual Reality (VR) (pp. 165-166). IEEE.

Cybis, W., Holtz, A. & Faust, R. (2010) “Ergonomia e usabilidade”. São Paulo: Novatec.

Hounsell, M. S., Silva, E. L. & Miranda, J. J. (2008a) “Detalhando aspectos de educação e treinamento em ambientes virtuais 3D”, In: International Conference on Engineering and Technology Education, São Paulo, SP.

Hounsell, M. S., Silva, E. L. & Kenczinsk, A. (2008b) “Medindo as ênfases em educação e treinamento de ambientes virtuais 3D”, In: International Conference on Engineering and Technology Education, São Paulo, SP.

Moraes, Ígor Andrade. (2013) “Realidade Virtual para treinamento em normas de segurança em manutenção de redes elétricas urbanas". Instituto Luteranos de Ensino Superior de Itumbiara.

Morris, C. G. & Maisto, A. A. (2004) “Introdução À Psicologia”. Peason Education Do Brasil.

Nielsen, J. (1994) “ Usability Engineering”. San Francisco: Morgan Kauffman.

Paas, F., Van Gog, T. & Sweller, J. (2010) “Cognitive Load Theory: New Conceptualizations, Specifications, And Integrated Research Perspectives”. Educational Psychology Review. v. 22, n. 2, p. 115-121.

Pimentel, Â., Dias, P. & Santos, B. S. (2008) “Avaliação de Usabilidade em Sistemas de Realidade Virtual e Aumentada: principais métodos”. Revista do Detua, v. 4, n. 9.

Prates, R. O. & Barbosa, S. D. J. (2003) “Avaliação de interfaces de usuário–conceitos e métodos”. In: Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, p. 28.

Preece, J., Rogers, Y. & Sharp, H. (2005) “ Design De Interação: Além da Interação Homen-Computador”, Bookman.

Proenca, A. P., Miranda Neto, M. M., Domingues, R. G., Borges, L. R., Cardoso, A. & Notargiacomo, P. (2017) “ Influence Degree Analysis of the Emphases of Education and Training in Virtual Environments dimensional Learning”. IEEE Latin America Transactions, v.15, n.5, p.974-980.

Richards, D. & Kelaiah, I. (2012) “Usability attributes in virtual learning environments”. In: Proceedings of The 8th Australasian Conference on Interactive Entertainment: Playing the System”. ACM. p. 9.

Rodrigues, C. R. M. A. B. (2012) “Proposta metodológica para avaliações otimizadas de usabilidade em websites desenvolvidos com método ágil: Um Estudo de Caso”. Tese de Doutorado. PUC-Rio.

Salustiano, Larissa Rodrigues. (2017) “N COLORS - experiência em ambiente virtual na exploração da cor. Dissertação Universidade Federal de Uberlândia

Silva, A. C. (2014) “Uma proposta de camadas de objetos de interface para realidade virtual”. Dissertação de Mestrado. Universidade Federal de Uberlândia.

6. Capítulo 6

Autores:

Arlino Magalhães
Universidade Federal do Ceará (UFC)/ Universidade Federal do Piauí (UFPI)
email: arlino@lia.ufc.br/ arlino@ufpi.edu.br

José Maria Monteiro
Universidade Federal do Ceará (UFC)
email: monteiro@dc.ufc.br

Ângelo Brayner
Universidade Federal do Ceará (UFC)
email: brayner@dc.ufc.br

Capítulo 6

Gerenciamento e Processamento de *Big Data* com Banco de Dados em Memória

Arlino Magalhães, José Maria Monteiro e Ângelo Brayner

Abstract

The growing main memory capacity coupled with the decrease in its cost has fueled researches in development of in-memory databases. Main-memory systems eliminate disk I/O bottleneck to support ultra-low latency service and real-time data analytics. However, in-memory systems are much more sensitive to other sources of overhead that do not matter in traditional disk-based systems, such as failure tolerance. Innovative approaches to fundamental issues such as concurrency control and query processing are required to unleash the full performance potential of main-memory databases. This work provides an overview of recent developments in main-memory database systems focusing around design issues and architectural choices that must be made when building a main-memory database system, such as: data storage, indexing, concurrency control, durability and recovery techniques, query processing and compilation, support for high availability, and ability to support hybrid workloads. Moreover, we present some in-memory data storage systems solutions; and concepts and techniques for efficient in-memory data management.

Resumo

O crescente aumento na capacidade de armazenamento em memória principal juntamente com a diminuição em seu custo impulsionaram pesquisas no gerenciamento e processamento de Big Data com Banco de Dados em Memória. Os sistemas em memória eliminam o gargalo de E/S em disco suportando serviços de latência muito baixa e a análise de dados em tempo real. No entanto, esses sistemas são mais sensíveis a outras fontes de sobrecarga que não estão presentes em sistemas tradicionais baseados em disco, como tolerância a falhas, por exemplo. Abordagens inovadoras para questões, como controle de concorrência e processamento de consultas, são necessárias para alavancar o potencial de desempenho dos bancos de dados em memória. Este trabalho fornece uma visão

geral dos desenvolvimentos recentes focados em questões de projeto e escolhas arquitetônicas que devem ser feitas ao criar um sistema de banco de dados em memória, tais como: armazenamento, indexação, controle de concorrência, durabilidade e técnicas de recuperação, processamento e compilação de consultas, suporte para alta disponibilidade e capacidade para suportar cargas de trabalho híbridas. Além disso, apresentamos alguns sistemas de armazenamento em memória; e conceitos e técnicas para um gerenciamento eficiente de dados na memória.

6.1. Introdução e motivação

O fenômeno *Big Data* motivou o desenvolvimento de sistemas com suporte a serviços com latência muito baixa e análise de dados em tempo real. Os sistemas de bancos de dados, tradicionalmente, têm sido voltados para armazenamento em disco que pode fornecer grandes quantidades de armazenamento de dados a um valor relativamente baixo. Entretanto, os sistemas baseados em disco existentes podem não ser capazes de oferecer tempos de resposta aceitáveis para sistemas de desempenho crítico devido à alta latência de acesso aos discos rígidos. Esse desempenho inaceitável foi inicialmente encontrado por empresas de *Internet* como Amazon, Google, Facebook e Twitter, mas agora também está se tornando um obstáculo para outras empresas/organizações que desejam fornecer um serviço em tempo real significativo (por exemplo, informações em tempo real, publicidade, jogos, entre outros). Por exemplo, muitas empresas comerciais precisam detectar uma mudança súbita nos preços de negociação e reagir instantaneamente (em poucos milissegundos), o que é muito difícil de conseguir usando sistemas tradicionais. Os sistemas baseados em memória principal eliminam o gargalo de E/S em disco proporcionando latência muito baixa e possibilidades de análise em tempo real. Além disso, nas últimas décadas os *chips* de memória tem dobrado em capacidade de armazenamento a cada 3 anos, enquanto que seus preços têm caído em um fator de 10 a cada 5 anos. Como consequência, usar bancos de dados em memória principal para aplicações que tradicionalmente usam bancos de dados em disco tornou-se tecnicamente possível e economicamente viável [Zhang et al. 2015, Hazenberg and Hemminga 2011].

Pesquisas e desenvolvimento de sistemas baseados em memória principal são datados desde o início da década de 80, como IMS/Fast Path [Strickland et al. 1982], MARS MMDB [Eich 1988], System M [Salem and Garcia-Molina 1990]. Entretanto, o preço elevado e a capacidade limitada da memória RAM, naquela época, tornou a aplicação de sistemas em memória uma solução inviável para a grande maioria dos problemas. Recentemente, dois fatores impulsionaram as pesquisas em sistemas em memória: preço/capacidade da memória RAM e paralelismo *multicore*. Os servidores modernos possuem CPUs que proveem estratégias de processamento em paralelo e podem armazenar uma parte significativa do banco de dados (ou até mesmo o banco inteiro) em memória a um preço razoável. Por exemplo, atualmente, um servidor com 32 núcleos e 1 TB de memória RAM pode custar em torno de $40.000. Além disso, muitos dos servidores contemporâneos possuem vários *sockets*, onde podem ser conectados muitos *terabytes* de DRAM e processadores com centenas de núcleos [Garcia-Molina and Salem 1992, Zhang et al. 2015].

Os sistemas de bancos de dados em disco utilizam a memória principal como uma *cache* e podem aumentar sua capacidade de armazenamento de maneira que o sistema

inteiro possa ser armazenado em memória. Essa estratégia tem as vantagens de necessitar poucas configurações no banco e aumentar o tempo de resposta do sistema, pois minimiza a quantidade de acesso ao disco. Entretanto, ainda há a sobrecarga de checar, para cada requisição ao banco, se um dado está no *buffer* (porção da memória destinada aos dados) ou não. Além disso, as estruturas de dados ainda continuam projetadas para otimizar o acesso ao disco. Nos bancos de dados em memória principal, como os dados residem diretamente na memória, as estratégias focam em otimizar o tempo computacional, uso da memória, ciclos de CPU e paralelismo *multicore*. Assim, para utilizar bancos de dados em memória da forma mais eficiente possível são necessárias novas abordagens nos algoritmos para processamento de consultas, indexação, controle de concorrência, durabilidade e recuperação após falha [Hazenberg and Hemminga 2011]. Na Seção 6.2 serão discutidas as estratégias de projeto e arquiteturais utilizadas frequentemente nos bancos de dados em memória. Na Seção 6.3 serão apresentadas as implementações utilizadas por uma amostra representativa dos sistemas de banco de dados em memória modernos. A Seção 6.4 mostra algumas pesquisas em sistemas de bancos de dados em memória com tecnologias emergentes.

6.2. Questões de *projeto* e escolhas arquitetônicas

Big Data frequentemente utiliza operações de análise em tempo real e uma maneira de atingir respostas em tempo real de um banco de dados é utilizar um servidor em memória. Pesquisas no gerenciamento e processamento de bancos dados em memoria focam em: indexação, leiaute dos dados, paralelismo, controle de concorrência, processamento de consultas e tolerância a falhas [Zhang et al. 2015]. Nessa seção serão discutidos aspectos chaves que diferem os bandos de dados em memória dos bancos tradicionais. Para cada questão, primeiro serão abordados os aspectos em bancos de dados em disco para, em seguida, serem apresentados como os mesmos aspectos são implementados em bancos de dados em memória. Adicionalmente, são discutidas as vantagens e desvantagens de cada implementação.

6.2.1. Organização dos dados

Os bancos de dados baseados em disco foram construídos com a suposição de que um banco inteiro não caberia na memória. Os dados devem residir no disco e a memória é utilizada apenas como uma *cache*. Assim, quando requisições a dados são realizadas, os blocos que contêm esses dados devem ser copiados do disco para páginas no *buffer* da memória principal para só depois disso serem processados pela CPU. Atualizações nas páginas devem ser descarregadas de volta para o disco. Geralmente uma página na memória possui um tamanho fixo semelhante ao do bloco no disco, para evitar traduções entre as duas representações. Por esse motivo, os bancos de dados em disco são classificados como orientados a blocos. Como o acesso a uma página/bloco é a unidade de E/S no disco e a E/S é o gargalo em sistemas em disco, esses sistemas tentam otimizar as E/Ss para atingir ganhos de desempenho. Por exemplo, acessar dados sequencialmente no disco é mais rápido do que aleatoriamente [Ramakrishnan and Gehrke 2009, Elmasri and Navathe 2015].

Muitas implementações de gerenciamento de *buffer* utilizam acesso indireto às páginas através de tabela *hash*, que indexa o pageId de uma página para sua localização física na memória. Depois de uma página ser carregada do disco para o *buf-*

fer, para acessar um registro dessa página ainda são necessárias duas etapas: (1) acessar a página no *buffer* através da tabela *hash*; e (2) calcular o ponteiro para o registro utilizando seu descolamento (*offset*) dentro página [Ramakrishnan and Gehrke 2009, Elmasri and Navathe 2015].

O *buffer* é uma solução simples para abstrair o acesso ao disco dos componentes de *software*. Entretanto, em bancos de dados em memória, o acesso indireto ao *buffer* pode causar impactos de desempenho no sistema. Como não há o gargalo da E/S, evitar os ciclos de CPU causados pelo acesso indireto para cada acesso a um registro pode trazer ganhos de desempenho em sistemas em memória. Esses sistemas evitam o acesso indireto utilizando ponteiros direto para os registros [Faerber et al. 2017]. Além disso, os bancos de dados em memória também tentam otimizar o acesso aos dados utilizando estratégias para organizar os registros, tais como: particionamento, multi-versionamento e leiaute linha/coluna.

Alguns bancos, como o VoltDB [Malviya et al. 2014], particionam os dados fisicamente no sistema. Uma partição é uma divisão de um banco de dados em partes distintas e independentes. Nos bancos de dados em memória particionados, as transações são executadas serialmente em cada partição de dados, ou seja, quando uma transação executa, ela deve ter acesso exclusivo a todas as partições que necessita. A vantagem dessa estratégia é a sua fácil implementação, pois evita os protocolos complexos de controle de concorrência baseados em bloqueios que podem causar sobrecargas nos sistemas em memória, visto que transações podem ter que esperar por desbloqueios. Na Seção 6.2.3 serão explicados os protocolos de controle de concorrência e os problemas que os protocolos baseados em bloqueio podem causar nos sistemas em memória. A desvantagem da estratégia de particionamento é a necessidade do balanceamento de carga em partições muito frequentemente acessadas. Nessa estratégia, as transações podem ser executadas em paralelo desde que em partições diferentes e utilizando núcleos diferentes [Stonebraker and Weisberg 2013, Malviya et al. 2014].

O multi-versionamento não é um conceito novo em bancos de dados, mas tem recebido interesse dos bancos de dados em memória. Nessa estratégia, uma transação faz suas alterações em uma nova versão do dado original e apenas ela pode visualizar essa versão. Ao realizar *commit*, a transação não sobrescreve o dado original, ela apenas marca a nova versão do dado como atual e a original como obsoleta. Essa estratégia permite que transações executem concorrentemente. Se duas transações estiverem alterando o mesmo dados simultaneamente, alguma estratégia de validação é utilizada para escolher qual transação poderá efetivar suas alterações, enquanto que a outra deverá ser abortada. O *abort* de uma transação é feito apenas não efetivando sua versão. A vantagem dessa estratégia é a mesma do particionamento, evitar os protocolos de bloqueio. Outra vantagem é permitir a criação de *snapshots* (imagem do banco de dados em um instante do tempo) que pode dar suporte a análise dos dados próximo de tempo real e recuperação após falhas (essas técnicas serão melhor explicada na Seção 6.2.4). Uma desvantagem do multi-versionamento é a necessidade de um coletor de lixo para excluir versões dos dados obsoletas [Silberschatz et al. 2015, Ramakrishnan and Gehrke 2009]. Os bancos de dados Hyper [Funke et al. 2014] e Sap Hana [Färber et al. 2012b] são multi-versionados e suportam *snapshots*. Na Seção 6.3 esses bancos e suas implementações serão melhor explicadas.

A terceira escolha é se os registros devem ser implementados em linhas ou colunas. Na abordagem orientada a linha, os campos de um registro são armazenados de forma contígua no disco, ou seja, as informações de um registro são mantidos juntas. Bancos baseados em linhas são aplicados em uso intensivo de cargas de trabalho OLTP (*Online Transaction Processing*). As cargas OLTP tem a característica de possuírem muitas transações de curta duração. São exemplos de aplicações OLTP: caixa eletrônico bancário, sistemas de comércio, entre outros. Os bancos colunares são mais apropriados para cargas de trabalho OLAP (*Online Analytical Processing*). Na abordagem colunar, os valores de uma coluna de uma tabela são armazenados contiguamente no disco. As transações OLAP manipulam e analisam um grande volume de dados e possuem uma maior duração do que as transações OLTP. Geralmente, aplicações OLAP são usadas por gestores de organizações para lhes permitir análises comparativas que facilitem a tomada de decisões. Tipicamente, os bancos em memória são utilizados em aplicações OLTP, mas existem alguns bancos em memória aplicados a cargas OLAP e outros com uma abordagem híbrida, ou seja, OLTP e OLAP.

6.2.2. Indexação

Apesar dos bancos de dados em memória serem mais rápidos, em comparação com os em disco, uma indexação eficiente é necessária para suportar consultas sem *scans* (varreduras) em tabelas. O projeto de indexação em bancos em memória é bastante diferente dos bancos em disco. A principal meta dos índices em bancos tradicionais é minimizar a quantidade de acessos ao disco. Os índices nos sistemas baseados em disco mapeiam *IDs* ou chaves primárias dos registos nas páginas gerenciadas pelo *buffer*. Os bancos de dados em memória focam em reduzir o tempo computacional e o uso da memória ao máximo possível. Como os dados residem na memória RAM, os bancos em memória não precisam que um índice armazene valores de atributos. Em vez disso, os índices armazenam ponteiros direto para os registros. Essa estratégia tem as vantagens de: (1) não ser necessário acesso indireto aos registros (descrito na Seção 6.2.1), economizando ciclos de CPU; (2) um único ponteiro prover acesso tanto aos valores dos atributos de uma tupla como à tupla em si, reduzindo o tamanho do índice; (3) eliminar a complexidade de lidar com campos longos, campos de comprimento variável e técnicas de compactação no índice e; (4) atualização em ponteiros ser menos custosa do que em atributos [Lehman and Carey 1986, Hazenberg and Hemminga 2011].

Existem dois aspectos importantes a considerar quando a estrutura de índices cabe inteiramente na memória: utilização eficiente de *cache* da CPU e paralelismo *multicore*. A velocidade dos processadores tem crescido mais rápido do que a velocidade das memórias RAM. Esse descompasso de velocidade pode levar ao aumento de *cache miss* (quando o conteúdo requisitado pela CPU não está presente na *cache*) tornando a memória RAM um gargalo para os sistemas em memória [Faerber et al. 2017]. Para resolver esse problema, pesquisas em índices têm focado em otimizar o uso de *cache line*, considerada a unidade de transferência entre a memória e a *cache*. Técnicas, como *Prefetching B+-Trees* (pB+-Trees) [Chen et al. 2001], exploraram a pré-busca (*prefetching*) de nós em árvores B+ para prover buscas mais eficientes. A pré-busca é uma técnica para acelerar as operações de busca, iniciando uma operação de busca cujo resultado é esperado em breve. Normalmente, a pré-busca ocorre antes de ser necessária, portanto existe o risco

de perder tempo buscando dados que não serão usados.

Atualmente, os servidores possuem dezenas de processadores com dezenas ou centenas de núcleos. Explorar paralelismo tornou-se muito importante, principalmente em sistemas OLTP cujos índices são estruturas frequentemente acessadas e/ou atualizadas. Em geral, existem duas maneiras de atingir altos níveis de concorrência em índices de sistemas em memória: particionada e compartilhada [Faerber et al. 2017]. A abordagem particionada particiona a estrutura de índices e atribui uma única *thread* a cada partição. A vantagem dessa abordagem é a sua fácil implementação, porém ela requer rebalanceamento para remover partições frequentemente acessadas/alteradas. Como exemplo, a técnica PLP [Pandis et al. 2011] utiliza árvore *multi-rooted B+* (MRBTree) em que uma sub-árvore corresponde a uma partição lógica de dados. A abordagem compartilhada permite que qualquer *thread* leia/atualize qualquer parte da estrutura de índices. Para prover altos níveis de concorrência e escalabilidade, essa abordagem procura evitar bloqueios (*lock free*) entre *threads*. As estruturas de índices livres de bloqueios utilizam operações atômicas de CPU para atualizar o estado de um índice. Esse método não requer balanceamento, entretanto há uma complexidade na sua implementação. Bw-tree [Levandoski et al. 2013] é um exemplo de estrutura de índices com alto nível de concorrência livre de bloqueios aplicada em bancos de dados em memória.

6.2.3. Controle de Concorrência

A maioria dos sistemas de bancos de dados comerciais utilizam alguma forma de controle de concorrência em duas fases (*Two-Phase Locking* - 2PL). O 2PL é pessimista e requer que uma transação adquira os bloqueios aos dados que necessita antes de executar e mantenha esses bloqueios até não precisar mais dos dados. Uma transação que queira utilizar dados bloqueados por outra transação deve esperar o desbloqueio desses dados. O Gerenciador de Bloqueios é o componente do SGBD responsável por implementar o protocolo 2PL. Apesar das implementações do gerenciador de bloqueios diferir entre os sistemas, elas comumente usam as seguintes estruturas de dados para manusear os bloqueios: tabela de bloqueios e tabela de transações. A tabela de bloqueios guarda informações sobre o recurso bloqueado, como: *id* do recurso do dado (identificador de um registro, por exemplo); modo de bloqueio (compartilhado ou exclusivo); e fila de transações esperando pelo desbloqueio do recurso. A tabela de transações armazena informações sobre as transações em execução no sistema, como: *id* da transação; estado da transação (em execução ou em espera por desbloqueio, por exemplo); e lista de bloqueios realizados pela transação [Ramakrishnan and Gehrke 2009, Elmasri and Navathe 2010].

As atividades realizadas pelo gerenciador de bloqueios requerem a manipulação de várias variáveis, que não é um problema para bancos de dados em disco devido ao gargalo da E/S, mas podem levar a sobrecargas em sistemas em memória. Assim, muitos bancos de dados em memória evitam implementar gerenciador de bloqueios, preferindo incorporar metadados de controle de concorrência em registros. Por exemplo, um *byte* pode ser utilizado em um registro para identificar se esse recurso está bloqueado ao não. Uma tendência dos bancos de dados em memória modernos é explorar o paralelismo através de alguma forma de controle de concorrência multi-versionado otimista ou execução serial particionada [Faerber et al. 2017].

O Controle de Concorrência Multi-Versionado (*Multi-Version Concurrency Control* - MVCC) utiliza um modelo de dados multi-versionado, discutido na Seção 6.2.1. No MVCC, uma transação faz atualizações em novas versões dos dados originais sem sobrescrevê-los. Antes de efetivar, uma transação deve ser validada para verificar se suas versões conflitam com versões de outra transação. Se a transação for validada, ela marca suas novas versões como atuais e as versões originais como obsoletas. Se a validação falhar, a transação é abortada descartando suas versões. Assim, uma transação nunca precisa esperar por desbloqueios, entretanto essa abordagem possui as desvantagens de sobrecarga para (1) criação de novas versões e; (2) coleta de lixo de versões obsoletas. Essa abordagem é otimista e permite a utilização de um grande número de núcleos, pois cada transação pode manipular suas versões independentemente até o fim de sua execução. A desvantagem dessa abordagem é que dados muito concorridos podem levar a sucessivos *aborts* e *restarts* de transações várias vezes, visto que a validação da transação é feita apenas no final de sua execução [Mühe et al. 2011, Kemper and Neumann 2011, Diaconu et al. 2013].

A execução serial particionada faz uso do modelo de dados particionado, mostrado na Seção 6.2.1, que divide o banco em partes disjuntas de dados. Nessa abordagem, cada transação deve executar serialmente uma após a outra dentro de uma partição, não necessitando de bloqueios e nem de validação. Essa estratégia provê uma execução muito rápida de uma transação dentro de uma partição; e permite a execução paralela de transações que executam em partições diferentes através de núcleos diferentes. Entretanto, uma transação com múltiplas partições precisa adquirir acesso exclusivo a todas as partições que precisa antes de começar a executar. Além disso, essa abordagem não é adequada a transações longas, pois estas podem fazer as outras transações esperar por um longo período de tempo para começar a executar [Kallman et al. 2008, Stonebraker and Weisberg 2013, Malviya et al. 2014].

6.2.4. Durabilidade e recuperação após falhas

Os bancos de dados relacionais em disco tipicamente utilizam alguma forma de recuperação a falhas baseada no protocolo ARIES. Nesse protocolo, informações sobre modificações no banco são escritas sequencialmente em um arquivo de *log* no disco antes que a modificação seja confirmada (esquema *write-ahead logging* - WAL). Cada registro no *log* possui um número sequencial (*Log Sequence Number* - LSN) que determina a ordem em que cada ação foi executada no banco. O arquivo de *log* armazena informações físicas (chamado de *log* físico); como tupla, bloco ou estrutura de dados de um índice, por exemplo. O *log* possui informações suficientes para refazer (REDO) ou desfazer (UNDO) uma atualização no banco. Após uma falha, ARIES (1) analisa o que deve ser recuperado através das informações contidas no *log*; (2) refaz as ações de transações que não foram descarregadas para o disco até o momento da falha e (3) desfaz as transações que não efetivaram. Para reduzir a quantidade de informação a ser analisada no *log* e, consequentemente, o tempo de recuperação, *checkpoints* periódicos identificam porções do *log* em que transações já foram descarregadas para o disco [Mohan and Levine 1992, Ramakrishnan and Gehrke 2009, Silberschatz et al. 2015].

Os bancos de dados em memória também utilizam técnicas de *log* e *checkpoint* para atingir durabilidade e recuperação após falhas. Entretanto, eles evitam abordagens

baseadas no protocolo ARIES por questões de desempenho. Os sistemas em memória preferem utilizar *log* lógico em vez de físico para garantir alto *throughput* e baixa latência durante o processamento normal das transações. O *log* lógico armazena informações sobre as modificações em alto nível (*insert*, *delete* e *update*, por exemplo) para diminuir o volume de dados enviado para o disco. Como desvantagem, o log lógico torna o processo de recuperação mais lenta, pois as ações de suas transações devem ser reexecutadas. Além disso, apenas informações de REDO são armazenadas em *group commit*, ou seja, informações para refazer modificações são acumuladas para envio em lote (até o tamanho de uma página, por exemplo) em uma única E/S [Garcia-Molina and Salem 1992, Kim et al. 2012]. Para reduzir ainda mais o tráfego de *log*, alguns sistemas utilizam um *log* lógico chamado de *Command Logging*, ou *Transaction Logging*, que armazena apenas o nome da *store procedure* de uma transação e seus parâmetros. Assim, todas as ações de uma transação podem ser gravadas em um único registro. Como desvantagem de *Command Logging*, todas as transações devem ser definidas em forma de *store procedure*, ou seja, precisam ser definidas previamente [Malviya et al. 2014, Wu et al. 2017].

Os checkpoints são bastante diferentes nos bancos em memória. Eles são considerados o *backup* mais recente do banco de dados, pois periodicamente propagam as informações lógicas contidas no *log* para sua forma física no disco. Alguns sistemas, em vez de propagar o *log*, fazem *snapshots*, que são arquivos persistentes equivalentes a um estado materializado do banco através de uma técnica chamada de *copy-on-update*. Além de tolerância a falhas, *copy-on-update* provê a possibilidade de análise de dados próximo do tempo real, pois produz uma cópia do banco em um instante do tempo que pode ser utilizada por aplicações OLAP, enquanto os dados originais são modificados por aplicações OLTP. Entretanto, *copy-on-update* pode causar sobrecargas ao sistema, pois pode precisar de uma grande quantidade de memória em sistemas frequentemente modificados, visto que cópias dos dados antes de modificações devem ser guardadas enquanto o mecanismo *copy-on-update* está executando. *Checkpoints* diminuem o tempo de recuperação, pois carregar para a memória informações físicas de *checkpoint* é mais rápido do que executar novamente informações lógicas de *log*. Após uma falha, o gerenciador de recuperação carrega o último *checkpoint* na memória e, em seguida, executa as ações gravadas no *log* a partir do *checkpoint*. [DeWitt et al. 1984, Diaconu et al. 2013, Funke et al. 2014].

6.2.5. Processamento de consultas e compilação

O Processador de Consultas é o componente do SGBD responsável por extrair os dados de um banco da forma mais eficiente possível. Em geral, o processamento de consultas possui as atividades de (1) análise, que consiste em converter uma consulta em alto nível (SQL, por exemplo) para uma representação interna melhor entendida pelo sistema (álgebra relacional, por exemplo); (2) otimização, em que é escolhido um plano de execução eficiente para consulta e (3) execução, efetuar as operações do plano de execução da consulta [Silberschatz et al. 2015, Ramakrishnan and Gehrke 2009]. Apesar das estratégias de processamento de consultas dos bancos de dados em disco focarem em minimizar as E/Ss, enquanto que as estratégias de processamento de consultas nos bancos de dados em memória focam em minimizar o tempo computacional; esses dois tipos de sistema possuem implementações semelhantes nas etapas de análise e otimização, entretanto são muito diferentes quanto a execução [Hazenberg and Hemminga 2011,

Faerber et al. 2017].

Os bancos em disco utilizam comumente um modelo de interação (*iterator model*) para executar um plano de consulta. No modelo de interação, cada operador do plano de consulta implementa um função genérica *get-next()* cuja função é retornar uma tupla para quem invocou o operador. Esse modelo é bem simples e permite uma combinação variada de operadores. Nos sistemas em disco, em que a E/S é sobrecarga dominante, o modelo de interação é muito poderoso, entretanto essa abordagem é muito ineficiente nos sistemas em memória. A execução de uma consulta invoca uma operação *get-next()* para cada tupla produzida, ou seja, essa função pode ser chamada várias vezes, milhões de vezes, por exemplo. Além disso, devido a natureza genérica de *get-next()*, o operador deve chamar outra função para interpretar e fazer o *cast* dos *bytes* de uma tupla para o tipo apropriado em tempo de execução. O modelo de interação leva a sobrecargas desnecessário nos bancos de dados em memória. Assim, esses sistemas preferem compilar as consultas e transações em código de máquina para evitar a interpretação em tempo de execução e, consequentemente, poupar ciclos de CPU. Alguns sistemas até restringem a implementação de transações a *store procedures* para que elas possam ser sempre compiladas e nunca interpretadas. Essa estratégia tem a desvantagem da obrigação das transações serem definidas previamente [Graefe 1993, Freedman et al. 2014, Faerber et al. 2017].

6.3. Bancos de dados em Memória

Essa seção explana os aspectos de implementação de quatro sistemas de gerenciamento de bancos de dados em memória modernos: Hekaton, H-Store/VoltDB, Hyper e SAP-HANA. As discussões serão feita com base nas questões de projeto e escolhas arquitetônicas mostradas na seção 6.2. Contudo, antes disso, será dado um breve histórico da pesquisa e desenvolvimento em bancos de dados em memória.

6.3.1. Histórico

As primeiras pesquisas na tecnologia de bancos de dados em memória começaram objetivando melhorar o desempenho de acesso a memória principal nos bancos de dados tradicionais [Garcia-Molina and Salem 1992]. O desenvolvimento e pesquisa em sistemas de bancos de dados em memória não é recente, começaram ainda no início da década de 80 com IMS/Fast Path [Strickland et al. 1982], MM-DBMS [Lehman and Carey 1987], MARS [Gruenwald and Eich 1991], System M [Salem and Garcia-Molina 1990], TPK [Li and Naughton 2000], OBE [Bitton et al. 1987] e HALO [Eich 1987]. Entretanto, a pouca capacidade de armazenamento e o custo elevando das memórias nessa época inviabilizaram a ampla adoção dos sistemas em memória.

Na metade da década de 90, os avanços tecnológicos no *hardware* (principalmente em memória RAM e processamento *multicore*) e o seu barateamento impulsionaram o desenvolvimento e a pesquisa em sistemas de bancos de dados em memória. O bancos de dados desenvolvidos na década de 90 (por exemplo: ClustRa [Hvasshovd et al. 1995], P*Time [Cha and Song 2004], Dalí [Bohannon et al. 1997], DataBlitz [Baulier et al. 1999], System K [Whitney et al. 1997], TimesTen [TimesTen Team 1999]) já começaram a refletir as tendências e escolhas arquiteturais utilizadas nos bancos de dados em memória mais modernos, como: Hekaton, VoltDB, HyPer e SAP HANA.

6.3.2. Hekaton

Hekaton é uma *engine* do Microsoft SQL Server (a partir da versão 2014) cujos dados residem diretamente na memória. Hekaton foi projetado para aplicações OLTP com altos níveis de concorrência e seu nome oficial é *OLTP in-memory*. As tabelas em Hekaton são armazenadas inteiramente na memória e são duráveis, embora tabelas não duráveis também sejam suportadas. Um banco de dados pode conter tabelas em memória no Hekaton e tabelas em disco no SQL Server. Uma *store procedure* pode acessar ambas as tabelas, em memoria e em disco. Contudo, *store procedures* que referenciam apenas tabela Haketon são compiladas em código nativo de máquina através do compilador Microsoft C. O código gerado contém exatamente o que é necessário para executar a *store procedure* com o melhor desempenho possível. A maioria das decisões são feitas em tempo de compilação da *store procedure* para reduzir sobrecargas em tempo de execução [Diaconu et al. 2013, Faerber et al. 2017].

Hekaton utiliza o multi-versionamento de dados, exemplificado na Figura 6.1. A Figura 6.1 representa registros de uma conta bancária em que cada registro contém os campos de cabeçalho *Begin* e *End* (*timestamps* de início e fim da versão, respectivamente); ponteiros para outras versões; e os campos definidos pelo usuário *Name*, *City* e *Amount*. No exemplo, existem seis versões de registros, sendo três delas provenientes da tupla que representa o cliente de nome John. A versão mais antiga de John contém os valores (John, London, 100) e estava válida do tempo 10 ao 20, quando foi atualizada. Essa atualização criou a versão (John, London, 110). A única versão da tupla da cliente Jane possui os *timestamps* de inicio e fim com os valores 15 e *inf* (fim ainda não definido), respectivamente. Ainda nesse exemplo, a transação de *ID* 75 está fazendo uma transferência de $20,00 da conta de Larry para John. Por esse motivo, a versão atual de John (John, London, 110) possui o *ID* da transação em seu campo *End* (indicando que a tupla está sendo modificada) e a nova versão manipulada pela transação (John, London, 130) possui o *ID* da transação em seu campo *Begin* (indicando que a versão ainda não é válida). Quando a transação for validada, a versão antiga de John receberá o *timestamp* 100 (quadrado tracejado) em seu campo *End*, tornando-a obsoleta, e a nova versão receberá o *timestamp* 100 em seu campo *Begin*, tornando-a como a versão atual no banco. As versões da tupla de Larry serão modificadas de forma semelhante ao que aconteceu em John. Periodicamente, um coletor de lixo exclui versões obsoletas [Diaconu et al. 2013].

O controle de concorrência multi-versionado é utilizado em Hekaton com o objetivo de prover isolamento entre transações sem bloqueios. As transações fazem modificações em suas versões dos dados sem a interferência de outras transações. Antes de uma transação efetivar, ela deve ser validada onde são verificados: se o dado original de alguma de suas versões foi modificado por outra transação; e, em *scans*, se alguma versão adicional foi obtida durante o processo. Uma transação pode estar em quatro estágios: (1) ativa, a transação faz seu processamento normal; (2) validando, a transação passa pelo processo de validação para saber se é efetivada ou forçada a abortar; (3) efetivada, estágio em que a transação terminou seu processamento e suas versões, se possuir, tornaram-se os dados atuais no banco e; (4) abortada, estágio em que a transação foi forçada a parar e suas versões, se possuir, são simplesmente descartadas [Diaconu et al. 2013, Faerber et al. 2017].

Os índices em Hekaton são livres de bloqueio e podem ser de três tipos: *hash*,

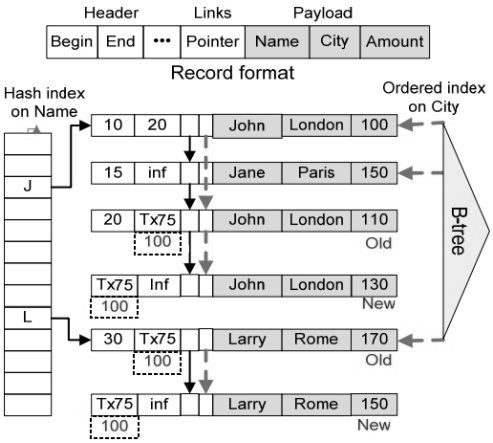


Figura 6.1. Múltiplas versões e indexação no Hekaton. Fonte: [Diaconu et al. 2013]

Bw-tree e *columnstore*. Os índices *hash* suportam buscas por um valor específico. Hekaton implementa um *hash* de tamanho estático com *buckets* de *overflow*. O exemplo da Figura 6.1 possui um índice *hash* no campo *Name* em que, por questões didáticas, o índice é aplicado apenas a primeira letra do nome. O *bucket* J (nomes começados com J) possui uma lista de quatro registros, três para John e um para Jane. Uma consulta que percorra o *bucket* J irá ignorar as versões inválidas. *Bw-tree* é uma estrutura de índices livre de bloqueios baseada em índices *B-tree* e projetada para o *hardware* moderno: processadores *multicore* e hierarquia de memória. Os índices em árvores permitem buscas por valores específicos ou faixa de valores. No exemplo da Figura 6.1 há uma estrutura de índices *Bw-tree* aplicada ao campo *City* cujos nós folha possuem chaves e ponteiros para registros. Tuplas com o mesmo valor de chave são ligadas através de ponteiros e a árvore Bw aponta para a primeira tupla da cadeia. Por exemplo, a chave "Londom"aponta para uma lista com três registros, sendo o primeiro referente a versão (John, London, 100). Igualmente a *hash*, uma consulta utilizando *Bw-tree* ignora as versões inválidas. Os índices *columnstore* são projetados para armazenamento e processamento de dados baseados em coluna [Diaconu et al. 2013, Levandoski et al. 2014].

Haketon utiliza *log* e *checkpoints* para prover durabilidade e recuperação após falhas. Apenas dados são enviados para *log*. Índices são reconstruídos durante o processo de recuperação. Quando uma transação é validada e, consequentemente, está pronta para efetivar, informações sobre novas versões criadas e/ou deletadas são enviadas para o *log* em *group commit*. Informações sobre transações abortadas não são enviadas para o *log*, tornando o processo de *logging* mais rápido. Periodicamente, *checkpoints* são realizado em paralelo ao processamento das transações no sistema. O processo de *checkpoint* produz dois arquivos a partir do *log*: (1) *data*, com versões criadas (*inserts* e *updates*) e; (2) *delta*, com informações sobre versões deletadas. Após uma falha, o gerenciador de recuperação carrega as versões contidas nos arquivos *data* utilizando os arquivos *delta* como filtro para versões deletadas. O par de arquivos *data*/*delta* é considerado a unidade de recuperação e permite uma recuperação altamente paralelizada. Após todos os arquivos *data* serem carregados na memória, os registros de *log* gravados após o último *checkpoint* são executados [Diaconu et al. 2013, Faerber et al. 2017].

6.3.3. H-Store/VoltDB

H-Store é um banco de dados em memória projetado para cargas de trabalho OLTP que pode ser implantado em um *cluster*. VoltDB [Malviya et al. 2014] é a versão comercial de H-Store. H-Store organiza seus dados em partições e cada partição não compartilha seus dados e índices com outras partições. A área de armazenamento das tabelas na memória em uma partição é dividida em *pools* de blocos de tamanho fixo e de blocos de comprimento variado, como ilustrado na Figura 6.2. O *pool* de blocos de tamanho fixo é o espaço de armazenamento primário das tuplas de uma tabela, onde cada tupla possui um tamanho fixo por tabela e cada campo de uma tupla tem o tamanho máximo de 8 *bytes*. Campos maiores que 8 *bytes* são armazenados no *pool* de blocos de comprimento variável. Uma tupla armazena os campos do *pool* de tamanho fixo agrupados, enquanto que guarda apenas a localização na memória dos campos que estão no *pool* de comprimento variado. O sistema possui um tabela de pesquisa com *IDs* dos blocos e suas respectivas localizações na memória para permitir a referência a tuplas individualmente. Para cada tabela, o SGBD mantém uma tabela de *pool* de tuplas livres. Quando uma transação insere uma tupla, primeiro o sistema verifica se há um tupla disponível na tabela de *pool*. Caso a tabela de *pool* esteja vazia, o sistema aloca novos blocos de tamanho fixo para inserir a nova tupla. As tuplas alocadas adicionalmente serão utilizadas em inserções posteriores. Quando uma tupla é excluída, sua referência é enviada para a tabela de *pool* de tuplas livres [Kallman et al. 2008, Faerber et al. 2017].

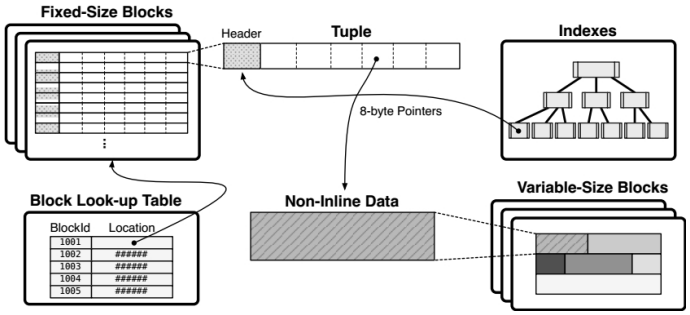


Figura 6.2. *Leiaute* de dados e índices no H-Store.
Fonte: [Faerber et al. 2017]

H-Store faz execução serial das transações em cada partição, ou seja, uma transação deve ter acesso exclusivo a todos os dados e índices nas partições que precisa para poder executar. Cada partição é atribuída a um *site* e cada nó do *cluster* pode possuir múltiplos *sites*. As partições também podem ser replicadas. H-Store implementa um escalonador de transações baseado em *timestamp*. Quando uma requisição de uma transação chega a um nó, o seu coordenador atribui à requisição um *ID* baseado em seu tempo de chegada que é único para todos os nós. Cada nó possui um coordenador que é responsável por direcionar cada transação aos *sites* apropriados. Transações podem executar em paralelo, desde que em partições diferentes. Se uma transação não conseguir acesso a uma das partições que precisa, o SGBD a aborta e então a reinicia. Essa abordagem evita a implementação de detecção de *deadlocks* em transações distribuídas, resultando em um maior *throughput* em transações OLTP, tipicamente curtas. Transações que executam em uma única partição são potencialmente mais rápidas do que as que executam em múltiplas partições. O projeto físico de banco de dados visa definir uma configuração que divida

as tabelas em partições tal que as transações das aplicações executem em apenas uma partição [Kallman et al. 2008, Faerber et al. 2017].

H-Store suporta estruturas de índices *hash* e *B-tree*. Como uma *thread* executa serialmente dentro de uma partição, a implementação de índices é mais simples do que em sistemas concorrentes. Quando uma consulta tenta acessar uma tabela usando uma coluna que não possui o atributo do particionamento dessa tabela, ela é enviada via *broadcast* para todas as partições. Isso é necessário porque o sistema não sabe qual partição possui a(s) tupla(s) que a consulta precisa. Ao invés de enviar comandos SQL em tempo de execução, as aplicações devem executar *store procedures* no sistema. Cada transação deve ser definida em forma de *store procedure*. Para invocar uma transação, uma aplicação precisa executar sua *stores procedure* passando os parâmetros necessários. Essa abordagem requer que todas as transações sejam conhecidas com antecedência, o que não é tão problemático em aplicações OLTP. Encapsular uma transação em uma *store procedure* evita as sobrecargas de processamento de consulta em tempo de execução, como *parsing*, por exemplo. Além disso, como as transações OLTP tipicamente são curtas e manipulam poucos dados, executá-las serialmente é mais eficiente do que utilizar protocolos de controle de concorrência baseados em bloqueios que introduzem uma sobrecarga significante [Kallman et al. 2008, Faerber et al. 2017].

Para prover durabilidade e tolerância a falhas, H-Store produz *log* e *snapshots*. Esse banco de dados utiliza *Command Logging* cujos registros armazenam o nome da *store procedure* da transação, os parâmetros de entrada enviados pela aplicação e o *ID* da transação. Índices não são enviados para o *log* e são reconstruídos no processo de recuperação. Como um registro de *log* não armazena as ações de uma transação, *save points* não são suportados. Isso não é significantemente limitante para transações OLTP, pois elas geralmente são curtas. Os *logs* são produzidos em uma *thread* diferente da *thread* de execução das transações, não implicando em bloqueios das transações pelo processo de *logging*, e vice-versa. Uma transação executada em uma única partição grava seu registro no arquivo de *log* de seu nó. Em uma transação distribuída, apenas o nó que coordenada a execução da transação grava os registros de *log*. Informações sobre a troca de mensagens entre os nós também são gravados no *log*. Os *logs* são escritos também para as replicas, se elas existirem. O SGBD escreve as informação de *log* quando a transação está prestes a efetivar, mas antes de retornar o resultado para aplicação. O sistema tenta escrever registos de *log* de múltiplas transações em *group commit* [Malviya et al. 2014, Faerber et al. 2017].

Periodicamente, são realizados *checkpoints* para copiar os dados efetivados por transações para o disco. O processo de *checkpoint* é chamado de *snapshot* no H-Store. Quando um *checkpoint* começa, o sistema é colocado no modo *copy-on-write* (COW). Durante esse modo, o processo de *snapshot* copia todos os dados de todas as tabelas para o disco durante a execução normal do sistema. Após o COW ter iniciado, novas tuplas inseridas são desconsideradas pelo *checkpoint* e são feitas cópias do conteúdo original de tuplas atualizadas ou deletadas para que possam ser copiadas pela *checkpoint* e, em seguida, deletadas. Após uma falha, o sistema copia o último *snapshot* do disco para a memória. Em seguida, o coordenador de cada nó executa o *log* enviado as transações para seus *sites* apropriados [Malviya et al. 2014, Faerber et al. 2017].

6.3.4. Hyper

Hyper é um sistema de banco de dados em memória capaz de manipular cargas de trabalho OLTP e OLAP simultaneamente no mesmo banco. Tipicamente bancos de dados OLTP e OLAP são separados em dois sistemas dedicados: o banco de dados tradicional e o *data warehouse*. Hyper isola tarefas OLTP e OLAP uma da outra para acomodá-las no mesmo banco. Ele separa os dados entre os que contêm as mais recente modificações e os que estão imutáveis. Esse sistema cria uma *snapshot* de memória virtual duplicando um processo OLTP que é utilizado em uma sessão OLAP. Por exemplo, no Unix a chamada de sistema fork() pode criar um processo filho através de um processo pai. Assim, um processo filho (OLAP) pode ser criado através de um processo pai (OLTP) obtendo a cópia exata do espaço de endereçamento do pai, como ilustrado a esquerda da Figura 6.3). Quando uma sessão OLAP começa, inicialmente, os processos pai e filho possuem o mesmo endereçamento virtual, ou seja, as mesmas páginas na memória representadas pelos quadrados não tracejados da Figura 6.3. Quando algum objeto é atualizado, o mecanismo *copy-on-update* faz uma cópia do conteúdo original do objeto para a memória virtual destinada apenas a sessão OLAP (quadrados tracejados da Figura 6.3). Por exemplo, ainda na Figura 6.3, após a modificação de um objeto, o novo estado do objeto (a") é acessível apenas por transações OLTP e o antigo estado do objeto (a') é acessível apenas por uma sessão OLAP. O mecanismo *copy-on-update* faz a cópia da página do objeto e não apenas a cópia do objeto, assim a' e b serão acessados pela mesma consulta OLAP na mesma página. Essa técnica permite que uma aplicação OLAP possa fazer consultas próximo ao estado mais recente do banco de dados. Adicionalmente, podem existir múltiplas sessões OLAP em diferentes pontos do tempo, cada uma com seu próprio *snapshot* [Funke et al. 2014, Kemper and Neumann 2011, Mühe et al. 2011].

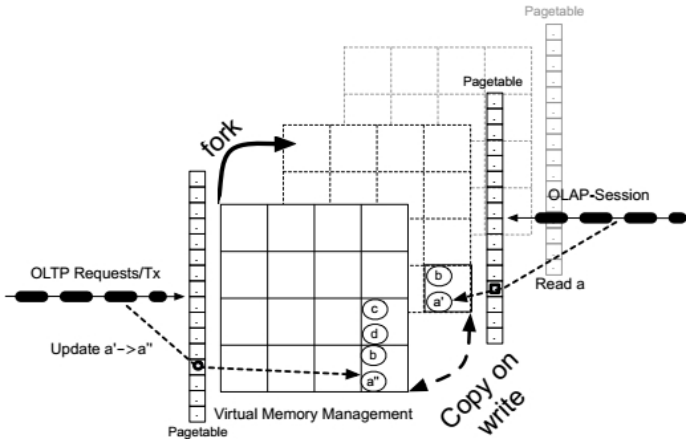


Figura 6.3. *Snapshots* no Hyper.
Fonte: [Mühe et al. 2011]

O mecanismo MVCC foi utilizado para prover isolamento entre transações. As transações OLTP executam serialmente e criam um conjunto de atualizações de páginas (algo similar a páginas sombra) em uma área separada do *buffer* (delta). Periodicamente, o armazenamento delta é copiado para o armazenamento principal. Através de MVCC é possível fazer fork() para sessões OLAP atenderem a análise de dados. Esse modelo tem a vantagem de evitar as sobrecargas do controle de concorrência baseados em bloqueios.

Como desvantagem esse mecanismo impõe uma sobrecarga proporcional ao número de atualizações de processos OLTP. Além disso, se a quantidade de memória disponível for pouca, consultas OLAP podem ter que usar armazenamento secundário para manter suas estruturas de dados temporárias [Funke et al. 2014, Kemper and Neumann 2011, Mühe et al. 2011].

Hyper não utiliza o modelo de processamento de consultas interpretativo. Em vez disso, compila planos de consulta inteiramente em código de máquina. Hyper utiliza massivamente o processamento *multicore* para prover paralelismo entre operadores em *joins*. Por exemplo, esse sistema utiliza Massively Parallel Sort-Merge Joins (MPSM) que é projetado para utilizar a arquitetura NUMA [Kemper and Neumann 2011, Faerber et al. 2017]. NUMA (*Non-Uniform Memory Access*) [Li et al. 2013] é uma arquitetura para projeto de memória principal de computadores multiprocessados. Cada processador, em um sistema NUMA, possui uma memória local cuja latência é mínima, mas o processador também pode fazer acessos remotos a outras memórias cuja latência é maior. Hyper também utiliza Adaptive Radix Tree (ART) [Leis et al. 2013] que é uma estrutura de índices Radix (Trie) adaptativa. ART é uma estrutura de índices para bancos de dados em memória que utiliza o *hardware* moderno para resolver o *trade off* entre altura e espaço das árvores radix. O desempenho de ART é comparado ao de tabelas *hash*.

A durabilidade de transações é feita através de *Command Logging* para um dispositivo de armazenamento não volátil onde são enviadas apenas informações de *REDO*. Um *log* de UNDO é mantido em memória volátil para prover atomicidade das transações, ou seja, suportar *rollbacks*. O mecanismo de *snapshots* também pode ser utilizado para criar *backups* do banco de dados inteiro para um servidor de armazenamento dedicado. No processo de recuperação após uma falha, o arquivo de *snapshot* mais recente é carregado para a memória e, em seguida, os registros de *log* gravados a partir do *snapshot* são executados. Hyper também pode utilizar replicas do servidor para garantir tolerância a falhas [Funke et al. 2014, Kemper and Neumann 2011, Mühe et al. 2011].

6.3.5. SAP HANA

SAP HANA é um banco de dados em memória que possui múltiplas *engines* de processamento de dados em um mesmo sistema: relacional híbrido (transacional e analítico no mesmo banco); para grafos e; processamento de textos para gerenciamento de dados semi e não estruturados [Färber et al. 2012a]. SAP HANA implementa uma estrutura de tabela unificada (esquematizada na Figura 6.4) e um mecanismo de propagação para mover registros através do sistema em diferentes estágios de representação física em um processo de gerenciamento de ciclo de vida controlado: registros orientados a linhas para aplicações OLTP e; registros orientados a colunas para aplicações OLAP. Uma configuração usual consiste em três estágios para registros dentro de uma tabela: *L1-delta*, *L2-delta* e *Main store*. *L1-delta* implementa o formato baseado em linhas e sua estrutura é otimizada para rápidas inserções, atualizações, deleções e projeções. L2-delta é uma estrutura intermediária organizada no formato colunar e aplica compressão de dados (*dictionary encoding*) para atingir um melhor uso da memória de trabalho (*footprint*). Entretanto, os dados não são ordenados, requerendo estruturas índices, se necessário. *Main store* implementa o formato colunar com compressão e ordenação de dados utilizando vários esquemas para atingir níveis de compressão altos, como: *prefix coding*, *cluster coding*,

indirect coding e *run length encoding* [Faerber et al. 2017, Sikka et al. 2012].

O banco de dados SAP HANA foi originalmente projetado para cargas de trabalho OLAP. Essa característica faz o sistema ser bastante eficiente em consultas e análise de dados, mas faz operações de atualização mais caras. Para evitar esse problema, as operações de atualização são feita apenas no armazenamento delta, *L1-delta* para operações regulares de *insert/update/delete* ou *L2-delta* para operações de inserção em lote. Atualizações apenas na parte delta evitam reorganizações nos dados. Periodicamente, o sistema propaga delta para *main store* em operações de *merge*: L1-to-L2-delta Merge e L2-delta-to-main Merge. Essas duas operações são propagadas assincronamente através das estruturas de armazenamento e não afetam significantemente o processamento normal das transações e são independentes das operações de *log* e *checkpoint*. Para assegurar a consistência dos dados, SAP HANA utiliza MVCC [Faerber et al. 2017, Sikka et al. 2012].

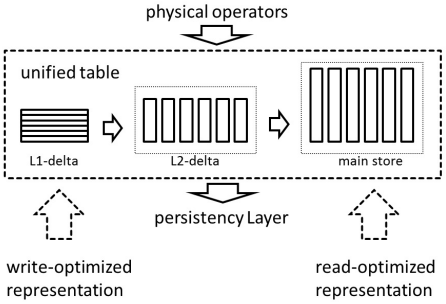


Figura 6.4. Esquema da tabela unificada no SAP HANA. Fonte: [Sikka et al. 2012]

A persistência é implementada através da combinação de *log* de REDO e de *save pointing* (*snapshot*). Registros são escritos no *log* apenas quando modificações são feitas em L1-delta ou L2-delta. Mudanças feitas durante as operações de *merge* não são enviadas para o *log*, mas o evento de *merge* é escrito no *log* para assegurar a consistência dos dados após uma reinicialização do sistema. Periodicamente, *L2-delta* e *Main store* são persistidos em *savepoint*. Durante a recuperação após uma falha, o sistema carrega na memória o último *savepoint* e executa os registros de *log* [Faerber et al. 2017, Sikka et al. 2012].

6.4. Desafios e problemas

Essa seção sumariza algumas áreas sobre pesquisa em sistemas de bancos de dados em memória com tecnologias emergentes: duas tecnologias não novas, mas que se tornaram viáveis a partir do surgimento do *hardware* moderno contemporâneo; e uma tecnologia é especulativa para um futuro próximo.

6.4.1. Gerenciamento de dados frios e quentes

Em muitas cargas de trabalho OLTP existem dados que são frequentemente acessados (*hot*) e dados que são raramente acessados (*cold*). Por exemplo, *sites* de *e-comerce* possuem produtos que são mais populares que outros. Além disso, essa preferência pode mudar durante períodos do ano. Agrupar um banco de dados em dados quentes (multáveis) e frio (imutáveis) tem alguns benefícios, como [Funke et al. 2014]:

1. Possibilidade de escolher representações diferentes para dados frios e quentes. Para dados frios são indicadas técnicas de compressão e armazenamento em disco.

2. Reduzir a quantidade de E/Ss para produção de *snapshots*, pois os dados frios podem ser persistidos apenas uma vez.

Implementar agrupamento de dados quentes/frios requer bastante cuidado para não produzir sobrecargas no sistema. Hyper monitora o acesso aos itens de dados e só "congela"partes do banco de dados que têm pouca ou nenhuma atividade. Abordagens baseadas em LRU (*Least Recently Used*) ou contadores de acesso não são adequadas a sistemas de alto desempenho, pois elas constituem uma carga adicional para transações OLPT. Hyper utiliza uma abordagem livre de sobrecargas baseada em informações geradas pela Unidade de Gerenciamento de Memória (*Memory Management Unit* - MMU) da CPU. As MMUs modernas definem sinalizadores para cada página de memória física para indicar se os dados nessa página foram lidos ou modificados. O sistema utiliza as informações dos sinalizadores para saber se uma página é frequentemente utilizada ou deve ser "congelada"[Funke et al. 2014].

H-Store implementa uma técnica chamada de *Anti-caching* que permite o gerenciamento de dados quentes/frios em sistemas cujo banco de dados é maior que a memória principal disponível sem levar a quedas no desempenho. Quando o banco de dados atinge uma quantidade de memória definida pelo Administrador, o sistema *anti-cache* envia dados frios para o disco utilizando LRU. As tuplas consideradas como frias são mapeadas em uma tabela. Quando uma transação tenta acessar uma dessas tuplas frias, H-Store verifica quais tuplas a transação necessita e, em seguida, a aborta. Em *background*, durante o processamento das outras transações, as tuplas frias da transação abortada são carregadas na memória. Depois de todas as tuplas necessárias estarem disponíveis na memória, a transação é reiniciada [DeBrabant et al. 2013].

6.4.2. Memória Transacional

Memória Transacional (*Transactional Memory* - TM) é um mecanismo para controle de concorrência a dados compartilhados em sistemas concorrentes, análogo ao controle de concorrência em transações de banco de dados. Basicamente, TM é responsável por: identificar acessos a memória dentro de transações, gerenciar os conjuntos de leitura e escrita da transação, detectar e resolver conflitos, gerenciar o estado dos registradores da arquitetura, efetivar e abortar transações. Memória transacional não é um conceito novo, suas pesquisas são datadas desde os anos setenta. Entretanto, essa tecnologia não foi implantada em sistemas de bancos de dados devido ao seu baixo desempenho, visto que era implementada via *software*. Recentemente, o uso de Memória Transacional em *Hardware* (*Hardware Transactional Memory* - HTM) tem atraído a atenção de pesquisas em bancos de dados por: (1) oferecer quase nenhuma sobrecarga a execução das transações e (2) ser menos invasiva aos sistemas atuais. A ideia por trás de HTM é a mesma de TM, entretanto HTM faz suas implementações na CPU. Como desvantagem, o uso de HTM é limitado a transações cujo conjunto de leituras e escritas caiba na *chace*. [Lintzmayer et al. , Cascaval et al. 2008]. Já existem implementações de HTM nos processadores modernos que estão sendo utilizados nos bancos de dados em memória, como o Haswell da Intel [Leis et al. 2014].

Hyper utiliza uma técnica para aumentar concorrência através de HTM. Essa estratégia provê paralelismo em uma *thread* quebrando uma transação em acessos a registros individuais (leitura ou escrita) em que cada acesso é executado dentro de uma memória transacional. Essa abordagem vai ao encontro das limitações de HTM, pois operações simples de registros são curtas e predicáveis. *Timestamps* são utilizados para ordenar os acessos de uma transação [Wang et al. 2014, Faerber et al. 2017]. No trabalho de [Karnagel et al. 2014], os autores utilizam HTM para suprimir bloqueios como uma maneira de melhorar a concorrência ao nível de *thread* em índices B++ no armazenamento delta de SAP Hana. Em [Makreshanski et al. 2015], é feito um estudo sobre alta desempenho em árvores B+ livre de bloqueios e árvores B+ baseadas em HTM. Em [Wei et al. 2015], é apresentada a ferramenta DrTM, um sistema de processamento de transações em memória que explora recursos de RDMA e HTM para melhorar a latência e o *throughput*.

6.4.3. Memória RAM não volátil

A Memória RAM não Volátil (*Non-Volatile Random Access Memory* - NVRAM) é uma tecnologia em desenvolvimento que promete ser uma intercessão das melhores características da DRAM e do disco rígido: alto desempenho, endereçamento por *byte*, persistência, grande volume de armazenamento e baixo consumo de energia [Faerber et al. 2017]. Já existem algumas implementações de NVRAM; como *Phase-Change Memory* - PCM [Raoux et al. 2008], Memristors [Strukov et al. 2008] e *Spin-Transfer Torque Magnetic RAM* - STT-MRAM [Driskill-Smith 2010]; que podem prover um desempenho semelhante a DRAM, mas com persistência. Entretanto, atualmente, NVRAM está disponível apenas em pequenas quantidades, mas especula-se que na próxima década haverão memórias PCM de 1TB e Memristors de 30TB ao preço dos discos rígidos atuais [Zhang et al. 2015].

Em [Chatzistergiou et al. 2015] é proposto REWIND, uma biblioteca em que o usuário pode implementar estruturas de dados persistentes arbitrárias em NVRAM. Os usuários atualizam as estruturas de dados REWIND usando uma *interface* de transação. SOFORT [Oukid et al. 2014] é um mecanismo de armazenamento híbrido projetado em dois níveis de hierarquia NVRAM e DRAM. Esse mecanismo utiliza MVCC para criar novas versões dos dados em NVRAM. O objetivo de SOFORT é atingir recuperação instantânea após falhas. Em [Arulraj et al. 2015] são avaliados três mecanismos diferentes de armazenar em NVRAM: *in-place updates*, *copy-on-write*, and *log-structured updates*. Os resultados dessa avaliação sugeriram que o armazenamento *in-place* é o que provê melhor desempenho em NVRAM. Também foram feitos trabalhos sobre *logging* e recuperação após falhas com NVRAM, como [DeWitt et al. 1984], [Coburn et al. 2013] e [Wang and Johnson 2014]. Devido a inviabilidade atual de uso de NVRAM, muitos dos trabalhos com NVRAM utilizam simuladores para fazer seus experimentos, como o Intel Lab's NVM Hardware Emulator [Dulloor et al. 2014].

6.5. Conclusões

Muitas aplicações contemporâneas, como aplicações *Big Data*, têm requerido serviços de latência muito baixa e com possibilidades de análise em tempo real. Os sistemas convencionais baseados em disco não têm conseguido atender aos requerimentos dessas

aplicações. Assim, sistemas em memória têm sido utilizados como alternativa para atender aplicações que demandam alto desempenho de acesso a dados, visto que os bancos de dados em memória armazenam seus dados diretamente na memória proporcionando altas velocidades de acesso. Entretanto, devido a grande diferença entre o disco e a memória, todos os aspectos de gerenciamento em banco de dados em memória devem ser repensados. Este trabalho discutiu os principais conceitos e técnicas que diferenciam os bancos residentes em memória dos bancos residentes em disco e apresentou uma amostra representativa dos sistemas de bancos de dados em memória mais modernos, auxiliando na tomada de decisão de que tipo de banco é mais adequado a um determinado sistema.

Referências

[Arulraj et al. 2015] Arulraj, J., Pavlo, A., and Dulloor, S. R. (2015). Let's talk about storage & recovery methods for non-volatile memory database systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 707–722. ACM.

[Baulier et al. 1999] Baulier, J., Bohannon, P., Gogate, S., Gupta, C., and Haldar, S. (1999). Datablitz storage manager: main-memory database performance for critical applications. In *ACM SIGMOD Record*, volume 28, pages 519–520. ACM.

[Bitton et al. 1987] Bitton, D., Hanrahan, M. B., and Turbyfill, C. (1987). Performance of complex queries in main memory database systems. In *Data Engineering, 1987 IEEE Third International Conference on*, pages 72–81. IEEE.

[Bohannon et al. 1997] Bohannon, P., Lieuwen, D., Rastogi, R., Silberschatz, A., Seshadri, S., and Sudarshan, S. (1997). The architecture of the dali main-memory storage manager. In *Multimedia Database Management Systems*, pages 23–59. Springer.

[Cascaval et al. 2008] Cascaval, C., Blundell, C., Michael, M., Cain, H. W., Wu, P., Chiras, S., and Chatterjee, S. (2008). Software transactional memory: Why is it only a research toy? *Queue*, 6(5):40.

[Cha and Song 2004] Cha, S. K. and Song, C. (2004). P* time: Highly scalable oltp dbms for managing update-intensive stream workload. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1033–1044. VLDB Endowment.

[Chatzistergiou et al. 2015] Chatzistergiou, A., Cintra, M., and Viglas, S. D. (2015). Rewind: Recovery write-ahead system for in-memory non-volatile data-structures. *Proceedings of the VLDB Endowment*, 8(5):497–508.

[Chen et al. 2001] Chen, S., Gibbons, P. B., and Mowry, T. C. (2001). *Improving index performance through prefetching*, volume 30. ACM.

[Coburn et al. 2013] Coburn, J., Bunker, T., Schwarz, M., Gupta, R., and Swanson, S. (2013). From aries to mars: Transaction support for next-generation, solid-state drives. In *Proceedings of the twenty-fourth ACM symposium on operating systems principles*, pages 197–212. ACM.

[DeBrabant et al. 2013] DeBrabant, J., Pavlo, A., Tu, S., Stonebraker, M., and Zdonik, S. (2013). Anti-caching: A new approach to database management system architecture. *Proceedings of the VLDB Endowment*, 6(14):1942–1953.

[DeWitt et al. 1984] DeWitt, D. J., Katz, R. H., Olken, F., Shapiro, L. D., Stonebraker, M. R., and Wood, D. A. (1984). *Implementation techniques for main memory database systems*, volume 14. ACM.

[Diaconu et al. 2013] Diaconu, C., Freedman, C., Ismert, E., Larson, P.-A., Mittal, P., Stonecipher, R., Verma, N., and Zwilling, M. (2013). Hekaton: Sql server's memory-optimized oltp engine. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1243–1254. ACM.

[Driskill-Smith 2010] Driskill-Smith, A. (2010). Latest advances and future prospects of stt-ram. In *Non-Volatile Memories Workshop*, pages 11–13.

[Dulloor et al. 2014] Dulloor, S. R., Kumar, S., Keshavamurthy, A., Lantz, P., Reddy, D., Sankaran, R., and Jackson, J. (2014). System software for persistent memory. In *Proceedings of the Ninth European Conference on Computer Systems*, page 15. ACM.

[Eich 1987] Eich, M. H. (1987). A classification and comparison of main memory database recovery techniques. In *Data Engineering, 1987 IEEE Third International Conference on*, pages 332–339. IEEE.

[Eich 1988] Eich, M. H. (1988). Mars: The design of a main memory database machine. *Database Machines and Knowledge Base Machines*, 43:325–338.

[Elmasri and Navathe 2010] Elmasri, R. and Navathe, S. (2010). *Fundamentals of database systems*. Addison-Wesley Publishing Company.

[Elmasri and Navathe 2015] Elmasri, R. and Navathe, S. B. (2015). *Fundamentals of database systems*. Pearson.

[Faerber et al. 2017] Faerber, F., Kemper, A., Larson, P.-Å., Levandoski, J., Neumann, T., Pavlo, A., et al. (2017). Main memory database systems. *Foundations and Trends® in Databases*, 8(1-2):1–130.

[Färber et al. 2012a] Färber, F., Cha, S. K., Primsch, J., Bornhövd, C., Sigg, S., and Lehner, W. (2012a). Sap hana database: data management for modern business applications. *ACM Sigmod Record*, 40(4):45–51.

[Färber et al. 2012b] Färber, F., May, N., Lehner, W., Große, P., Müller, I., Rauhe, H., and Dees, J. (2012b). The sap hana database–an architecture overview. *IEEE Data Eng. Bull.*, 35(1):28–33.

[Freedman et al. 2014] Freedman, C., Ismert, E., and Larson, P.-Å. (2014). Compilation in the microsoft sql server hekaton engine. *IEEE Data Eng. Bull.*, 37(1):22–30.

[Funke et al. 2014] Funke, F., Kemper, A., Mühlbauer, T., Neumann, T., and Leis, V. (2014). Hyper beyond software: Exploiting modern hardware for main-memory database systems. *Datenbank-Spektrum*, 14(3):173–181.

[Garcia-Molina and Salem 1992] Garcia-Molina, H. and Salem, K. (1992). Main memory database systems: An overview. *IEEE Transactions on knowledge and data engineering*, 4(6):509–516.

[Graefe 1993] Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Computing Surveys (CSUR)*, 25(2):73–169.

[Gruenwald and Eich 1991] Gruenwald, L. and Eich, M. H. (1991). *MMDB reload algorithms*, volume 20. ACM.

[Hazenberg and Hemminga 2011] Hazenberg, W. and Hemminga, S. (2011). Main memory database systems: Opportunities and pitfalls. *SC@ RUG 2011 proceedings*, page 113.

[Hvasshovd et al. 1995] Hvasshovd, S.-O., Torbjørnsen, Ø., Bratsberg, S. E., and Holager, P. (1995). The clustra telecom database: High availability, high throughput, and real-time response. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 469–477. Morgan Kaufmann Publishers Inc.

[Kallman et al. 2008] Kallman, R., Kimura, H., Natkins, J., Pavlo, A., Rasin, A., Zdonik, S., Jones, E. P., Madden, S., Stonebraker, M., Zhang, Y., et al. (2008). H-store: a high-performance, distributed main memory transaction processing system. *Proceedings of the VLDB Endowment*, 1(2):1496–1499.

[Karnagel et al. 2014] Karnagel, T., Dementiev, R., Rajwar, R., Lai, K., Legler, T., Schlegel, B., and Lehner, W. (2014). Improving in-memory database index performance with intel® transactional synchronization extensions. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, pages 476–487. IEEE.

[Kemper and Neumann 2011] Kemper, A. and Neumann, T. (2011). Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 195–206. IEEE.

[Kim et al. 2012] Kim, J.-J., Kang, J.-J., and Lee, K.-Y. (2012). Recovery methods in main memory dbms. *International journal of advanced smart convergence*, 1(2):26–29.

[Lehman and Carey 1986] Lehman, T. J. and Carey, M. J. (1986). A study of index structures for main memory database management systems. In *Proc. VLDB*, volume 1.

[Lehman and Carey 1987] Lehman, T. J. and Carey, M. J. (1987). *A recovery algorithm for a high-performance memory-resident database system*, volume 16. ACM.

[Leis et al. 2013] Leis, V., Kemper, A., and Neumann, T. (2013). The adaptive radix tree: Artful indexing for main-memory databases. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 38–49. IEEE.

[Leis et al. 2014] Leis, V., Kemper, A., and Neumann, T. (2014). Exploiting hardware transactional memory in main-memory databases. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 580–591. IEEE.

[Levandoski et al. 2014] Levandoski, J., Lomet, D., Sengupta, S., Birka, A., and Diaconu, C. (2014). Indexing on modern hardware: Hekaton and beyond. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 717–720. ACM.

[Levandoski et al. 2013] Levandoski, J. J., Lomet, D. B., and Sengupta, S. (2013). The bw-tree: A b-tree for new hardware platforms. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 302–313. IEEE.

[Li and Naughton 2000] Li, K. and Naughton, J. F. (2000). Multiprocessor main memory transaction processing. In *Proceedings of the first international symposium on Databases in parallel and distributed systems*, pages 177–187. IEEE Computer Society Press.

[Li et al. 2013] Li, Y., Pandis, I., Mueller, R., Raman, V., and Lohman, G. M. (2013). Numa-aware algorithms: the case of data shuffling. In *CIDR*.

[Lintzmayer et al.] Lintzmayer, C. N., Theodoro, E., and Sambinelli, M. Memória transacional.

[Makreshanski et al. 2015] Makreshanski, D., Levandoski, J., and Stutsman, R. (2015). To lock, swap, or elide: on the interplay of hardware transactional memory and lock-free indexing. *Proceedings of the VLDB Endowment*, 8(11):1298–1309.

[Malviya et al. 2014] Malviya, N., Weisberg, A., Madden, S., and Stonebraker, M. (2014). Rethinking main memory oltp recovery. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 604–615. IEEE.

[Mohan and Levine 1992] Mohan, C. and Levine, F. (1992). *ARIES/IM: an efficient and high concurrency index management method using write-ahead logging*, volume 21. ACM.

[Mühe et al. 2011] Mühe, H., Kemper, A., and Neumann, T. (2011). How to efficiently snapshot transactional data: Hardware or software controlled? In *Proceedings of the Seventh International Workshop on Data Management on New Hardware*, pages 17–26. ACM.

[Oukid et al. 2014] Oukid, I., Booss, D., Lehner, W., Bumbulis, P., and Willhalm, T. (2014). Sofort: A hybrid scm-dram storage engine for fast data recovery. In *Proceedings of the Tenth International Workshop on Data Management on New Hardware*, page 8. ACM.

[Pandis et al. 2011] Pandis, I., Tözün, P., Johnson, R., and Ailamaki, A. (2011). Plp: page latch-free shared-everything oltp. *Proceedings of the VLDB Endowment*, 4(10):610–621.

[Ramakrishnan and Gehrke 2009] Ramakrishnan, R. and Gehrke, J. (2009). *Database management systems*. McGraw Hill.

[Raoux et al. 2008] Raoux, S., Burr, G. W., Breitwisch, M. J., Rettner, C. T., Chen, Y.-C., Shelby, R. M., Salinga, M., Krebs, D., Chen, S.-H., Lung, H.-L., et al. (2008). Phase-change random access memory: A scalable technology. *IBM Journal of Research and Development*, 52(4.5):465–479.

[Salem and Garcia-Molina 1990] Salem, K. and Garcia-Molina, H. (1990). System m: A transaction processing testbed for memory resident data. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):161–172.

[Sikka et al. 2012] Sikka, V., Färber, F., Lehner, W., Cha, S. K., Peh, T., and Bornhövd, C. (2012). Efficient transaction processing in sap hana database: the end of a column store myth. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 731–742. ACM.

[Silberschatz et al. 2015] Silberschatz, A., Korth, H. F., Sudarshan, S., et al. (2015). *Database system concepts*, volume 4. McGraw-Hill New York.

[Stonebraker and Weisberg 2013] Stonebraker, M. and Weisberg, A. (2013). The voltdb main memory dbms. *IEEE Data Eng. Bull.*, 36(2):21–27.

[Strickland et al. 1982] Strickland, J. P., Uhrowczik, P. P., and Watts, V. L. (1982). Ims/vs: An evolving system. *IBM Systems Journal*, 21(4):490–510.

[Strukov et al. 2008] Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *nature*, 453(7191):80.

[TimesTen Team 1999] TimesTen Team, C. (1999). In-memory data management for consumer transactions the timesten approach. In *ACM SIGMOD Record*, volume 28, pages 528–529. ACM.

[Wang and Johnson 2014] Wang, T. and Johnson, R. (2014). Scalable logging through emerging non-volatile memory. *Proceedings of the VLDB Endowment*, 7(10):865–876.

[Wang et al. 2014] Wang, Z., Qian, H., Li, J., and Chen, H. (2014). Using restricted transactional memory to build a scalable in-memory database. In *Proceedings of the Ninth European Conference on Computer Systems*, page 26. ACM.

[Wei et al. 2015] Wei, X., Shi, J., Chen, Y., Chen, R., and Chen, H. (2015). Fast in-memory transaction processing using rdma and htm. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 87–104. ACM.

[Whitney et al. 1997] Whitney, A., Shasha, D., and Apter, S. (1997). High volume transaction processing without concurrency control, two phase commit, sql or c++.

[Wu et al. 2017] Wu, Y., Guo, W., Chan, C.-Y., and Tan, K.-L. (2017). Fast failure recovery for main-memory dbmss on multicores. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 267–281. ACM.

[Zhang et al. 2015] Zhang, H., Chen, G., Ooi, B. C., Tan, K.-L., and Zhang, M. (2015). In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1920–1948.

7. Capítulo 7

Autores:

César Alberto Collazos
Universidad del Cauca (Colombia)
email: ccollazo@unicauca.edu.co

Sandra Cano
Universidad San Buenaventura de Cali (Colombia)
email: sandra.cano@gmail.com

Vanessa Agredo-Delgado
Corporación Universitaria Comfacauca – Unicomfacauca (Colombia)
email: vagredo@unicomfacauca.edu.co

Chapter 7

Interactive System Design

César Alberto Collazos, Sandra Cano and Vanessa Agredo-Delgado

Abstract

As Computer Science has developed and evolved in the search for research, design, and development of new technologies to help improve the interaction between people (users) and computing devices, the Human-Computer Interaction (HCI) has gradually become part of its curriculum. Human-Computer Interaction is one of the most promising research areas in Computer Science field, focusing its activities ranging from design to the development of the evaluation of computer systems, with the aim of understanding how computer devices influence people and society. In spite of its growth, it is hard to find in Iberoamerican Computer Science programs the inclusion of HCI in an effective way into its curricula. In this paper we highlight some of the elements need to be considered when HCI courses need to be included in curricula. In particular, we emphasized in aspects related to Interactive System Design, implementation and evaluation.

1. General Information

There is still no concrete definition for the concepts set that makes up the area of human-computer interaction. In general terms, it is the discipline that studies the exchange of information through software between people and computers [Montaño, et al., 2005]. This discipline is responsible for the design, evaluation and implementation of interactive technological devices, studying the largest number of cases that may affect them. The objective is to make the exchange more efficient: minimize errors, increase satisfaction, reduce frustration and, ultimately, make the tasks that surround people and computers more productive, furthermore, Human Computer Interaction (HCI) is a research, formative and practical discipline related with the design of interactive systems. It lies at the crossroads of many scientific areas like Psychology, Computer Vision, Artificial Intelligence, Face Recognition, Motion Tracking [Blackwell, 2010], etc. In recent years there has been a growing interest in improving all aspects of the interaction between humans and computers. It is argued that for truly achieving effective

Human-Computer Intelligent Interaction (HCII), there is a need for the computer to be able to interact naturally with the user, like the way human-human interaction takes place [Sebe, et al., 2004]. Interactive systems are now a ubiquitous part of people's lives - from web applications to games to embedded devices - and the design and usability of these systems are having an increasingly large effect on the quality of people's relationship to technology.

As we see, in recent years, subjects related to Computer Science have evolved rapidly and are also constantly being revised to reflect the nature of the changes that occur, among others, in the technological field that directly affects it [Martig, et al., 2001]. Computing has changed dramatically in recent years, having a profound effect on both the design of the subjects and pedagogy. The rapid evolution of the discipline has a profound effect on education in Computer Science and affects both content and pedagogy: there have been evolutionary changes and other revolutionaries; both undoubtedly affect the knowledge required for an undergraduate subject in their educational process. As these accelerated changes occur both in the academic and cultural context in which education is developed and in the professional activity that develops in different topics related to Computer Science, it is evident that certain topics become more relevant. The technological advances of the past decade have increased the importance of some curricular topics such as the WWW and its applications, graphics and multimedia and human-computer interaction, among others [Martig, et al., 2001], for example, the Bachelor of Arts & Science degree (B.A. & Sc.) program combines courses in Art and Art History, Psychology, and Computer Science, and these courses will provide knowledge and skills in several critical areas: principles of visual communication; critical approaches to visual systems; fundamentals of human perception, memory, and cognition; and the principles of computation and programming needed to design, build, and evaluate games and interactive systems.

Since the publication of the ACM SIGCHI Curricula for Human-Computer Interaction back in 1992 [Hewett, et al., 1992], Computer Science educators have evolved diverse implementations of the above-mentioned guidelines. Most of them were offered as elective courses or modules within other courses, such as Graphics and Multimedia, Software Engineering, or even as part of the introductory courses sequence. In the years 92, only 3% of Computer Science accredited degree programs include an HCI course at the upper level [McCauley & Manaris, 2002]. In Iberoamerica the situation is even worst, there are many reasons why HCI and courses related to Interactive system Design still have not considered a basic part of the Computer Science Curricula, nevertheless, in the last decade, HCI has acquired great importance, constantly emerging technological changes and therefore there is a need for educational content and more research in the field of IHC. One of the main complications in Latin America is the access, availability, and cost of books in the Spanish language [Muñoz Arteaga , et al., 2014].

In Latin America region today already have active research professors in IHC, many of them associated with local committees IHC, even with spaces disclosure. Chile, Mexico, Brazil are the countries which have more committees and conferences related to the field of IHC [Collazos, et al., 2010], some of them are SIGCHI ACM, Professional Association for User Experience (UxPA), Mexican Association of Human-Computer Interaction AC (AMexIHC) and various forums such as Interaction, MexIHC

(IHC Mexican Congress), Latin American Conference on Human Computer Interaction (CLIHC), among others [Muñoz Arteaga , et al., 2014].

There is no doubt that the Human-Computer Interaction, is an essential topic in all areas of Computing. The interfaces are the visible face of the systems, constituting the communication means between the interactive process actors. Undoubtedly, students in all computing branches need to know how to design, develop and maintain interfaces, each for the systems types inherent to their specialty. The interfaces creation is an essential topic with its own life cycle, its own techniques and its own methodologies. Any interaction with the systems will be done through the interfaces and that is why its design should be adjusted according to the different areas involved, always considering the human characteristics, both in terms of capabilities and limitations [Jacko, 2012]. Human-computer interaction can be defined as the study of the interaction between the user and computer [Dix, et al., 2003]. Therefore, the interaction is mainly done at the user interface. HCI design involves some questions, as: What are the challenges of designing interactive? What are the possible solutions to such problems in developing a good user-centered design? What are the principles of user interface design? what are the new trends?

Another relevant aspect is that, in Iberoamenican countries, the HCI teaching-learning process has been immersed in academic programs in the Social Sciences field, where communication, coordination and cooperation with Computer Science academics are rather difficult. It is also common that HCI topics are not considered consistent enough to justify a complete course; therefore, many times they have been included within the Software Engineering syllabus [Collazos, 2005, Granollers, et al., 2008].

However, in countries like Spain, Brazil and Mexico the situation is better. There are associations like AIPO –Asociación Interacción Persona Ordenador (www.aipo.es) and ACM–SIGCHI chapters, which promote HCI teaching in these countries. Next, some aspects about HCI and Interactive Systems Design are going to be explained, concepts that should be included in the curriculum of HCI courses, that allow research and training in this area that is currently booming and is so necessary to compete in the work world to which they will face.

2. Usability

The concept of usability in general terms is defined as "ease of use" whether it is a web page, a computer application or any other system that interacts with a user [ISO, 2001], moreover, Usability Engineering is a method in the progress of software and systems, which includes user contribution from the inception of the process and assures the effectiveness of the product through using a usability requirement and metrics [Nielsen, 1994].

It thus refers to the *Usability Function* features of the entire process of abstracting, implementing and testing hardware and software products. Requirements gathering stage to installation, marketing and testing of products, all fall in this process.

Currently, there are several definitions for this term, which is why we will present a series of definitions that will help us establish a general idea of this concept.

Usability is a quality attribute that assesses how easy user interfaces are to use. The word "Usability" also refers to methods for improving ease-of-use during the design process.

"Usability is one of the most important quality characteristics for Web applications together with others such as reliability and security" [Offutt, 2002]. On the other hand, software usability is defined as the quality of a user's experience when interacting with software products or systems, including websites, devices or software applications. Usability determines efficiency, efficiency and overall user satisfaction. The ISO 25010 standard [ISO, 2011] determines usability as the "ability of the software product to be understood, learned, used and attractive to the user, when used under certain conditions".

Usability is defined by 5 quality components:

- **Learnability**: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency**: Once users have learned the design, how quickly can they perform tasks?
- **Memorability**: When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- **Errors**: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction**: How pleasant is it to use the design?

In order to evaluate Usability Nielsen [Nielsen, 1994] have proposed a set of 10 Usability Heuristics which are general principles for interaction design. These heuristics are:

- **Visibility of system status: T**he system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- **Match between system and the real world:** The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- **User control and freedom**: Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- **Consistency and standards: U**sers should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- **Error prevention:** Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

- **Recognition rather than recall:** Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- **Flexibility and efficiency of use:** Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **Aesthetic and minimalist design: D**ialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- **Help users recognize, diagnose, and recover from errors**: Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- **Help and documentation:** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

3. HCI and Software Engineering

Software Engineering is the study of designing, development and preservation of software, in addition, is a discipline that establishes the use of robust engineering principles, oriented to obtain economic software that is reliable and functions efficiently. The Software Engineering comes in contact with HCI to make the human and machine interaction more vibrant and interactive. To develop an interactive system, software engineering must be considered, through the study of design techniques, software development and engineering procedures, which will allow obtaining quality software. The Software Engineering and Human-Computer Interaction group is a multi-perspective group focusing on a single problem: how to help people develop software that is effective and accurate [Seffah, et al., 2005]. The people we are trying to help range from professional programmers to end users who use special-purpose tools to create their own software. Let us see the following model in software engineering for interactive designing.

3.1The Waterfall Method:

The waterfall model (Fig. 1.) is a relatively linear sequential design approach for certain areas of engineering design, in which the software development is conceived as a stages set that are executed one after the other and it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction ("downwards" like a waterfall) through the phases of conception, initiation, analysis, design, construction,

testing, deployment and maintenance. The waterfall development model originated in the manufacturing and construction industries; where the highly structured physical environments meant that design changes became prohibitively expensive much sooner in the development process [Benington, 1983]. When first adopted for software development, there were no recognized alternatives for knowledge-based creative work.

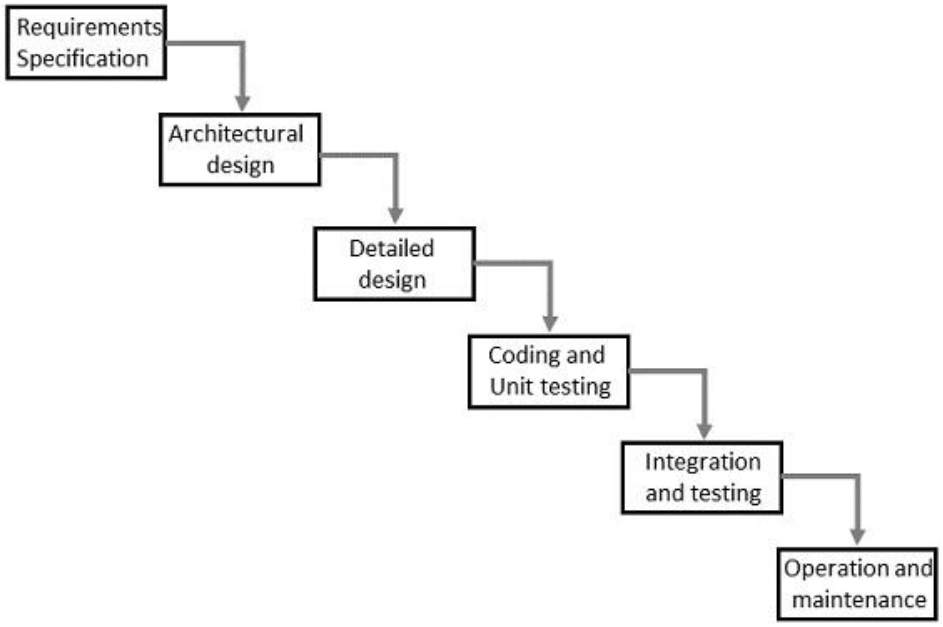


Fig. 1. Waterfall method diagram

This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on. Because of the cascade from one phase to another, this model is known as the 'waterfall model' or software life cycle. The waterfall model is an example of a plan-driven process in principle, you must plan and schedule all the process activities before starting work on them [Sommerville, 2011].

3.2Interactive System Design

The uni-directional movement of the waterfall model of Software Engineering shows that every phase depends on the preceding phase and not vice-versa. However, this model is not suitable for the interactive systems design.

In the interactive system design, every phase depends on each other to serve the purpose of designing and product creation. It is a continuous process as there is so much to know and users keep changing all the time. An interactive system designer should recognize this diversity [Rudisill, et al., 1996].

3.3 Prototyping

Prototyping is another type of software engineering model that can have a complete range of functionalities of the projected system, modeling the final product and allowing a test on certain attributes.

In HCI, prototyping is a trial and partial design that helps users in testing design ideas without executing a complete system [Preece, 1994].

The main purpose of prototyping is to involve the users in testing design ideas and get their feedback in the early stage of development, thus to reduce the time and cost. It provides an efficient and effective way to refine and optimize interfaces through discussion, exploration, testing and iterative revision [Rudd, et al., 1996]. Early evaluation of an interactive application can be based on faster and cheaper prototypes before the start of a full-scale implementation. The prototypes can be changed many times until a better understanding of the user interface design has been achieved with the joint efforts of both the designers and the users.

Prototyping can be divided into *Low-Fidelity Prototyping*, *Medium-Fidelity Prototyping* and *High-Fidelity Prototyping* [Walker, et al., 2002]. In some literature, it is only simply classified as *Low-Fidelity Prototyping* (also called Lo-Fi) and *High-Fidelity Prototyping* (also called Hi-Fi) [Rudd, et al., 1996], where *Low-Fidelity* is mainly about paper-based mock-up [Reinhard, et al., 2003], and *High-Fidelity* [Palanque, et al., 2009] is mainly about computer-based simulation. The determining factor in prototype fidelity is the degree to which the prototype accurately represents the appearance and interaction of the product, not the degree to which the code and other attributes invisible to the user are accurate. Other prototypes will be divided into *Low-Fidelity* and *Medium-Fidelity* prototypes. We will focus on the *Low-Fidelity* and *Medium-Fidelity* prototyping techniques. *Medium-Fidelity Prototyping* and *High-Fidelity Prototyping* are discussed together on some attributes indicated as medium(high)-fidelity prototyping.

Low-Fidelity Prototyping are quickly constructed to depict concepts, design alternatives, and screen layouts, rather than to model the user interaction with a system. *Low-Fidelity Prototyping* provide limited or no functionality. They are intended to demonstrate the general look and the feel of the interface, but not the detail how the application operates. They are created to communicate and exchange ideas with the users, but not to serve as a basis for coding and testing. A facilitator who knows the application thoroughly is generally needed to demonstrate the prototype to the users [Rudd, et al., 1996].

Example of a prototype can be Sketches, which is the first concrete element of an interactive design can later be produced into graphical interface. See the following diagram.

Fig. 2. Low Fidelity prototype (Sketches)

The Fig 2 shows a *Low-Fidelity Prototyping* as it uses manual procedures like sketching in a paper.

A *Medium-Fidelity Prototyping* involves some but not all procedures of the system. E.g., first screen of a GUI.

Finally, a *High-Fidelity Prototyping* simulates all the functionalities of the system in a design. This prototype requires, time, money and work force.

4. User Centered Design (UCD)

User centered design (UCD) is a design process that focuses on user needs and requirements. The consistent application of human factors, ergonomics, usability engineering, and other techniques is what keeps UCD revolving around the users. The aim is to produce highly usable and accessible systems, aiming for user satisfaction while averting negative effects on health, safety, and performance [Pea, 1987].

UCD is an iterative design approach that aims to develop an understanding of user needs, doing so through a mixture of investigative (e.g., surveys and interviews) and generative (e.g., brainstorming) methods and tools. Crucially, UCD heavily involves users in all design and evaluation phases. In general, each iteration of the UCD approach involves four distinct phases. First, designers attempt to understand the context in which a system may be used. Subsequently, the users' requirements are specified. A design phase then follows, which is then succeeded by an evaluation phase. The outcomes of the evaluation are assessed against the users' context and requirements to check how well a design is performing—namely, how close it is to a level that matches the users' specific context and satisfies all their relevant needs. From here, further iterations of these four phases are made, until the evaluation results are satisfactory.

During UCD, the design is based upon an explicit understanding of the users, tasks, and environments. The process aims to capture and address the whole user

experience; therefore, the design team must include professionals across multiple disciplines (e.g., ethnographers, psychologists, software and hardware engineers), as well as domain experts, stakeholders, and the users themselves. Evaluations of the produced designs may be carried out by experts using design guidelines and criteria. However, a crucial matter is that UCD must—at some point—involve the users, and it should also involve long-term monitoring of use.

The process of collecting feedback from users to improve the design is known as user centered design or UCD (See Fig 3).

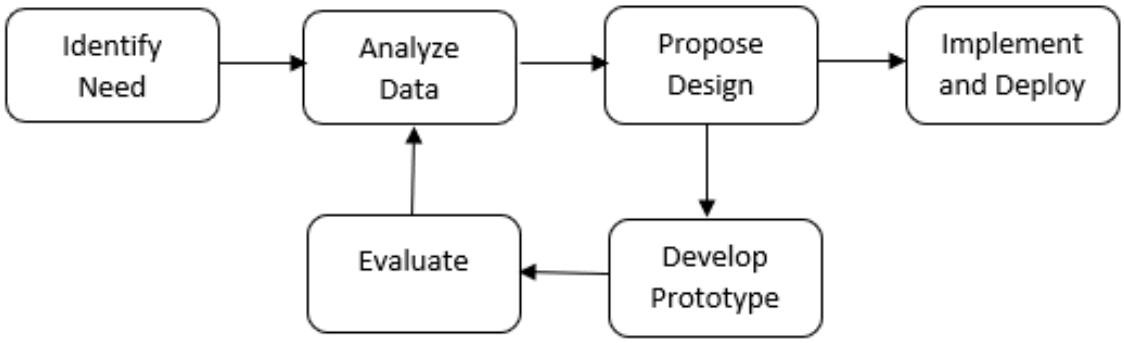


Fig. 3. User Centered Design Process

Therefore, the interactive system design can be defined as a user-oriented field of study that focuses on the communication relationship between the user and the system. The interactivity level is acquired by the existing resources allowing the user to establish a participation and communication process with the system [Cano, et al., 2017a].

4.1 UCD is not [Lowdermilk, 2013]

Is not Usability: It is necessary to define the differences between concepts, for example, UCD is not usability, Usability referred to as human factors, is the study of how humans relate to any products. Usability practice could be implemented in everything from a toaster to a doorknob, and even the packaging of both. Human-computer interaction is rooted, but it focuses on how humans relate to computing products. UCD emerged from HCI and is a software design methodology for developers and designers. Essentially, it helps them make applications that meet the needs of their users.

Is not Subjective: The entire discipline of usability and all its underlying methodologies are a conglomeration of many scientific disciplines, usability is rooted in scientific knowledge. It's far from subjective thinking or conjecture. The user-centered design process works against subjective assumptions about user behavior. It requires proof that your design decisions are effective. If the user-centered design is done correctly, your application becomes an outcome of actively engaging users. Therefore, any design decisions that were made by observing and listening to them will not be based on whims or personal preferences. By observing users directly, we remove assumptions and statistically prove what is actually happening. This gives us a more stable foundation for the direction of our development.

Is not just design: Some people believe that user-centered design practitioners are only focused on aesthetics or making things look pretty. While an application's aesthetic can be important, it's not the whole picture.

Being user-focused is more than just deliberating on how things look or creating flashy animation and slick transitions. The user-centered design ensures that we examine how effective an application is achieving its designed purpose.

5. Study Case

We present a study case designing an interactive system for children with hearing impairment, where UCD helps to obtain evidences of the children through the evaluation methods about characteristics, behaviors in children in the Institute of blind and deaf children in Cali-Colombia. Since 2012, the institute has been working on an alternative method for the teaching of reading and writing in the early stages of the implant known as the Invariant Method [Solovieva & Quintanar, 2012]. Our participants were children with a cochlear implant with ages 7 to 11 years. The work was done with 8 Transitional children, where 4 have a cochlear implant, 2 have some hearing aid and 2 children are listeners since the institute has an inclusion program for children without hearing limitations. Process that can be seen in [Cano, et al., 2016a]

Therefore, to design a system, we need the participation of a multidisciplinary development team made up of professionals in Special Education, Occupational Therapy, Educational Technology, Physiotherapy, Ergonomics and Computer Science.

5.1 Hearing Impairment

Hearing Impairment is an obstacle preventing to process information linguistically through the ear. Generally, it is known as deafness. A deaf child, who does not have hearing aids, will communicate with the society through sign language or lip-reading. Further, they have difficulties developing concepts in several areas. However, some deaf children have been benefited from hearing aids such as cochlear implants (is a transducer that transforms the acoustic signals into electrical signals that stimulate the auditory nerve. These electrical signals are processed through the different parts of the Cochlear Implant, which are divided into External and Internal). These children can go on to communicate verbally and must learn to receive information by means of sounds, so they need to learn to recognize the sounds via the cochlear implant. Schools are choosing pedagogical models that fit the characteristics of the child to promote literacy learning in deaf children [Ruiz Linares, 2009].

Children with *Cochlear Implant* have problems in literacy learning, because they must learn to listen to that they can speak. Therefore, the teachers working an initial stage a method called invariant [Solovieva & Quintanar, 2012], where children must learn to identify each sound corresponding with a word, they begin identifying vowel sounds and then consonants.

5.2 Identifying Needs

A multidisciplinary team collaborated in the design of the interactive system and we designed and developed using a prototyping model to introduce at early stages the end-user, which allowed us to implement the new requirements as they appeared. We carried out interviews with the professionals of the center where children with hearing impairment were attended, we employed direct observation of the users in their context (Fig. 4.) and analyzed session logs and documentation on the users/activities/teaching strategies provided by the center.

Fig. 4. Applying an evaluation method called Drawing Intervention from 7 – 11 years

The method called drawing intervention communicates the user experience through drawings, where the children are not required to speak or comment. This method is based on the observation about what the children draw and has been used to understand children's thinking. The activity consisted in that they were to construct an animated character with the aim to know their interest and to design the character of the game.

Other kinds of activities are also done on paper, such as the children drawing themselves, or putting together a character from pieces handed out to them, where they are required to choose parts of the body, face and go on to piece them together [Cano, et al., 2017b].

5.3 Prototyping

The proposal is creating a board game called Phonomagic, which is formed as an element of user participation where is integrated a physical board that interacts with a digital device such as RFID sensor and Bluetooth to communicate with an application in

Arduino. Therefore, children are constantly interacting within a real environment and same time digital.

Once collected information about children with a cochlear implant is beginning designing the characters and board in low fidelity (Fig. 5. and Fig. 6.).

Fig. 5. Prototype in low fidelity Phonomagic in [Galves & Giraldo, 2016]

Fig. 6. Proposal of principal characters in [Galves & Giraldo, 2016]

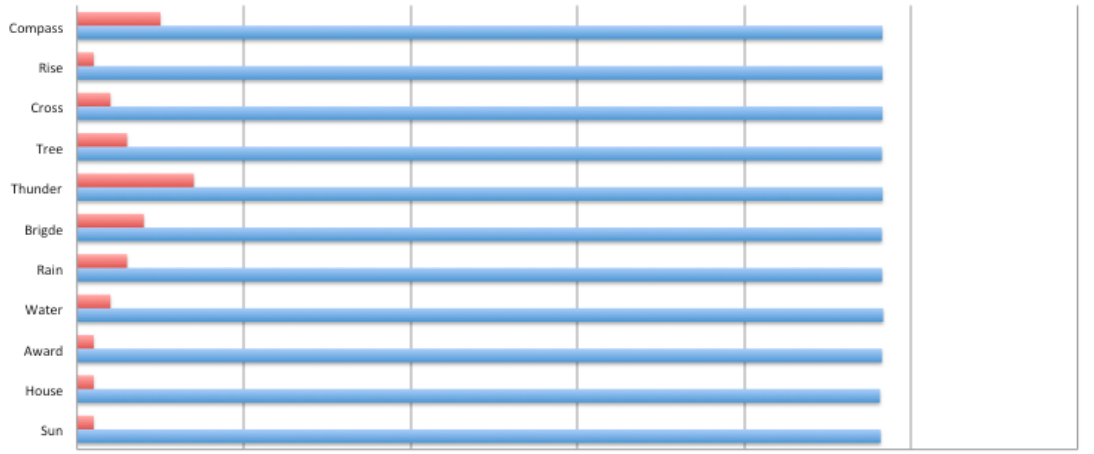
To analyze each of the objects of communication with the children, a group activity was carried out, in which they were told the story. They were then shown physical cards of each of the characters to be able to recreate the story. While there were some words they did not understand, the need was also identified to improve the story with a simpler writing style and using a more visual support of the scenarios.

Finally, prototype design is proposed in Fig. 7., which it integrates a physical board and interaction with Tablet by means a system communication using RFID and Bluetooth to interconnect the data by each card read.

Fig. 7. Functionality with physical card and mobile application. In [Cano, et al., 2016a]

5.4 Prototype evaluation

The final phase of the methodology involves validation. The evaluation is made with the interest of consolidating different aspects that were considered in the previous phases. An activity was carried out with children their performance in interacting with the **Phonomagic** game was evaluated. Three factors were measured: (1) fulfillment of the objective, (2) errors committed, and (3) activities carried out. It is also important to mention that even though the children had a poor vocabulary of Spanish, they knew how to communicate with each other, not with very clear oral language, but they did understand every word their classmate spoke or communicated using gestures.



	Sun	House	Award	Water	Rain	Brigde	Thunder	Tree	Cross	Rise	Compass
Tries	1	1	1	2	3	4	7	3	2	1	5
Average	48.2	48.2	48.3	48.3	48.3	48.3	48.3	48.3	48.3	48.3	48.3

Fig. 8. Evaluating the Fonomagic Tool. In [Cano, et al., 2016b].

In the same way, the validation considered the results obtained in the research and the way in which the children have been involved, both in the creation and in the evaluation. The final prototypes pointed to the invariant method optimization at the

level of support tools, of exercises based on the pedagogical objectives within the literacy area and the learning motivation of children with cochlear implants through play. In this validation stage, the child's experience evaluation was considered when interacting with these prototypes that allowed generating a new version with the obtained results.

The evaluation must be considered in any methodology since it serves to validate the compliance of the aspects that have been defined in the subsequent phases and guarantees the reach of the pedagogical / recreational objectives.

Also, a set of activities are defined where were worked on among several children with the same role, or with different roles. Certain variables were defined to monitor the level of participation of each child within the activities. The aim is thus to determine indicators that make it possible to measure the level of participation of each child during the activities. The following metrics were therefore considered, taking account of those proposed by [Collazos, et al., 2007]: number of errors, solution to the problem and movements (number of attempts). Consequently, it is proposed that these data be collected using the mobile application as strategies for a quantitative evaluation of each child, since the performance results captured are quantitative. The evaluation carried out, however, was qualitative.

Conclusions

In this document we have presented a methodology for the interactive ystems development based on the user description as the center of the process, through a case study, in such a way that shows the importance of its use in any context and in general of the HCI in to training students in the computer area.

The user-centered design is a topic that is part of HCI, and as seen in the case study is important for the development of any system type, since it allows users to be part of the entire process, and the result obtained satisfy their needs, so that changes are made before its implementation and the system can be done as the user requires. In this way, they are concepts that must be inculcated to the students, being a tendency that has been increasing its use, for which it is important to include it in the computer science curriculum, besides the other topics that are part of the HCI area and that should be considered in the training.

The prototyping serves to include the necessary modifications in the development phase. This allows us to provide new and better functionalities, achieve unbeatable designs, detect the necessary changes so that the product has a better acceptance in the market and even discard when the project is not viable. They also serve to analyze and evaluate, detecting errors and possible improvements, confirm that it has the desired characteristics and ensure that it can be adapted to the production process. This saves large costs and avoids detecting a fault when it is already being manufactured in series, which will serve us to design interactive systems in any context you want.

The different experiences published up to now coincide in pointing out the need to orientate the interactive systems towards the user needs, considering their social

context. This approach focused on the user allows us to know their responsibilities in the system and the possible role changes that may be made in the future. Considering that all large software companies and interactive products have departments of user experience, usability, user research, etc. It is necessary to train in these topics and instill a community that analyzes all these aspects.

References

Benington, H. D., 1983. Production of Large Computer Programs. *Annals of the History of Computing,* 5(4), pp. 350-361.

Blackwell, A., 2010. Human Computer Interaction–Lecture Notes. En: *Cambridge Computer Science Tripos.* s.l.:s.n.

Cano, S. y otros, 2017a. Interactive Systems Design Oriented to Children with Special Needs. En: *HCI for Children with Disabilities.* s.l.:Springer, Cham, pp. 73-89.

Cano, S. y otros, 2017b. Assessing User Experience for Serious Games in Auditory-Verbal Therapy for Children with Cochlear Implant. En: *World Conference on Information Systems and Technologies.* s.l.:Springer, pp. pp. 861-871.

Cano, S. y otros, 2016b. Toward a methodology for serious games design for children with auditory impairments. *IEEE Latin America Transactions,* 14(5), pp. 2511-2521.

Cano, S. P. y otros, 2016a. Sistema Interactivo para la Enseñanza de la Lectoescritura para niños con Implante Coclear. *IE Comunicaciones: Revista Iberoamericana de Informática Educativa,* Volumen 24, pp. 21-29.

Collazos, C. A., 2005. La enseñanza de HCI en Colombia. *I Jornadas de Trabajo sobre Enseñanza de HCI, Puertollano (Ciudad Real),* pp. 81-91.

Collazos, C. A., Granollers, T. & Ortega, M., 2010. Hacia una Integración de la Interacción Humano-Computador en las Estructuras Curriculares a Nivel Iberoamericano. *Revista Internacional de Educación en Ingeniería,* 3(1), pp. 7-16.

Collazos, C. A. y otros, 2007. Evaluating Collaborative Learning Processes using System-based Measurement. *Educational Technology & Society,* 10(3), pp. 257-274.

Dix, A., Finlay, J., Abowd, G. & Beale, R., 2003. *Human-Computer Interaction.* Third ed. s.l.:Pearson.

Galves, L. & Giraldo, P., 2016. *Diseño de un Sistema de Comunicación Interactivo para Apoyar el Aprendizaje de la Lectoescritura de Niños con Limitación Auditiva en el Instituto de Niños Ciegos y Sordos del Valle del Cauca,* Popayan: Tesis de pregrado Universidad del Cauca.

Granollers, T., Collazos, C. A. & González, M. P., 2008. The State of HCI in Ibero-American Countries. *Journal of Universal Computer Science,* 14(16), pp. 2599-2613.

Hewett, T. y otros, 1992. *ACM SIGCHI Curricula for Human-Computer Interaction,* New York: ACM Special Interest Group on Computer Human Interaction .

ISO, I. S. I. 9., 2001. *In Software engineering-Product Quality.* s.l.:s.n.

ISO, I. S. Q. S. I. 2., 2011. *in Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Systems and software quality models, ed,.* s.l.:s.n.

Jacko, J., 2012. *Human computer interaction handbook: Fundamentals, evolving technologies, and emerging applications..* Third ed. s.l.:CRC press.

Lowdermilk, T., 2013. *User-centered design: a developer's guide to building user-friendly applications.* United States of America: O'Reilly Media, Inc..

Martig, S., Castro, S. & DiLuca, S., 2001. *Un curso de Interacción Humano Computadora en las currículas de Ciencias de la Computación.* El Calafate Santa Cruz Argentina, s.n.

McCauley, R. A. & Manaris, B., 2002. Comprehensive report on the 2001 survey of departments offering CAC-accredited degree programs. *Computer & Information Science,* 3(7).

Montaño, N., Michinel, J. L. & Soriano, A., 2005. Lo significativo en la Interacción Humano-Computador: una perspectiva educativa del diseño de software. *Revista de Pedagogía,* pp. 375-395.

Muñoz Arteaga , J., Collazos Ordóñez, C. A., Bustos Amador, V. & Alvarez Rodríguez, F., 2014. Collaborative Production in Latin America of the Open Text Books in Human Computer Interaction. *Proceedings of the XV International Conference on Human Computer Interaction,* p. 88.

Nielsen, J., 1994. *Usability engineering.* Montain View, California: Elsevier.

Offutt, J., 2002. *Quality Attributes of Web Software Applications. IEEE Software: Special Issue on Software Engineering of Internet Software.* s.l.:s.n.

Palanque, P., Ladry, J.-F., Navarre, D. & Barboni, E., 2009. High-Fidelity Prototyping of Interactive Systems Can Be Formal Too. En: *International Conference on Human-Computer Interaction.* Berlin, Heidelberg: Springer, pp. 667-676.

Pea, R. D., 1987. User centered system design: New perspectives on human-computer interaction.. *Journal educational computing research,* 3(1), pp. 129-134.

Preece, J., 1994. Extract-Chapter 22: Envisioning Design. En: *Human-Computer Interaction.* s.l.:Addison-Wesley, pp. 451-465.

Reinhard, S., Tscheligi, M. & Giller, V., 2003. Paper prototyping-what is it good for?: a comparison of paper-and computer-based low-fidelity prototyping. *CHI'03 extended abstracts on Human factors in computing systems ,* pp. 778-779.

Rudd, J., Stern, K. & Isensee, S., 1996. Low vs. high-fidelity prototyping debate. *Interactions ,* 3(1), pp. 76-85.

Rudd, L., Stern, K. & Isensee, S., 1996. Low vs. high-fidelity prototyping debate. *Interactions,* 3(1), pp. 76-85.

Rudisill, M., Lewis, C., Polson, P. & McKay, T., 1996. *Human-Computer Interface Design: Success stories, emerging methods, real-world context,* San Francisco: Morgan Kaufmann Pub.

Ruiz Linares, E., 2009. El aprendizaje de la lectoescritura en los niños y niñas sordos. *Caleidoscopio, Revista digital de contenidos educativos,* 2(1).

Sebe, N., Lew, M. & Huang, T., 2004. The State-of-the-Art in Human-Computer Interaction. En: *International Workshop on Computer Vision in Human-Computer Interaction.* Berlin, Heidelberg: Springer, pp. 1-6.

Seffah, A., Desmarais, M. & Metzker, E., 2005. HCI, Usability and Software Engineering Integration: Present and Future. En: *Human-Centered Software Engineering — Integrating Usability in the Software Development Lifecycle.* Dordrecht: Springer, pp. 37-57.

Solovieva , Y. & Quintanar, L., 2012. *Método de formación de lectura para la correction de dificultades en el desarrollo.,* Universidad Autonoma de Puebla, México: Tesis Doctoral. Tesis Maestria de Diagnóstico y Rehabilitación Neuropsicológica..

Solovieva, Y. & Quintanar, L., 2012. *Método de formación de lectura para la correction de dificultades en el desarrollo.,* Puebla: Universidad Autonoma de Puebla, México. Tesis Maestria de Diagnóstico y Rehabilitación Neuropsicológica,.

Sommerville, I., 2011. *Software Engineering.* Ninth ed. Boston: Pearson.

Walker, M., Takayama, L. & Landay, J., 2002. High-fidelity or low-fidelity, paper or computer? Choosing attributes when testing web prototypes. *Proceedings of the human factors and ergonomics society annual meeting ,* 46(5), pp. 661-665.

8. Capítulo 8

Autores:

Ana Grasielle Dionísio Corrêa
Universidade Presbiteriana Mackenzie
email: ana.correa@mackenzie.br

Leandro Pupo Natale
Universidade Presbiteriana Mackenzie
email: leandro.natale@mackenzie.br

Israel Florentino dos Santos
Universidade Presbiteriana Mackenzie
email: israel.santos@mackenzie.br

Capítulo 8

Game Design como Estratégia para o Ensino de Programação de Computadores: exemplo de uso da biblioteca *Pygame*

Ana Grasielle Dionísio Corrêa, Leandro Pupo Natale, Israel Florentino dos Santos

Abstract

Contemporary pedagogical approaches such as game design can motivate students to learn subjects related to programming languages as students become interested in the challenges imposed by game design. In this chapter, a mini-course proposal on game design is presented to support programming learning. The chapter presents a step-by-step guide on how to specify a digital game based on the game design elements and how to encode it using the Python language and the Pygame library. It is present a lesson plan containing the methodology for the construction of a presented game example. It is expected that this workshop can be replicated by teachers and researchers interested in new methodologies for programming teaching.

Resumo

Abordagens pedagógicas contemporâneas, como o game design, podem motivar os alunos na aprendizagem de disciplinas relacionadas à linguagens de programação, pois os alunos passam a se interessar pelos desafios impostos pelo design dos jogos. Neste capítulo, é apresentada uma proposta de minicurso sobre o game design para apoiar a aprendizagem de programação. O capítulo apresenta um passo-a-passo de como especificar um jogo digital baseado nos elementos de game design e como codificá-lo utilizando a linguagem Python e a biblioteca Pygame. Também é apresentado um Plano de Aula contendo a metodologia para a construção do exemplo de jogo apresentado. Espera-se que esta oficina possa ser reproduzida por professores e pesquisadores interessados em novas metodologias para o ensino de programação.

1.1. Introdução

Recentes mudanças sociais em razão da inserção das Tecnologias de Informação e Comunicação (TIC) no quotidiano das pessoas têm causado tanto uma necessidade quanto uma oportunidade para uma nova abordagem de ensino. A necessidade é causada pela queda nas matrículas em cursos de Computação e Informática, conforme aponta o *Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira* [1]. Poucos iniciantes parecem considerar o aprendizado de programação uma atividade fácil e prazerosa. Além disso, projetar um ambiente de ensino e aprendizagem que seja envolvente, considerando os desafios impostos pelo ensino de programação, não é uma tarefa simples [2].

A oportunidade de reverter este cenário pode estar entre os estudantes da nova geração, considerados nativos digitais em função do alto índice no uso de computadores, *smartphones*, videogames, entre outros dispositivos eletrônicos. De acordo com uma pesquisa realizada pela Game Brasil [3] a maioria das pessoas, com trinta anos ou menos, joga ocasionalmente ou frequentemente. A pesquisa mostrou também que, ao contrário de percepções equivocadas, as mulheres representam cerca de 53,6% de todos os jogadores, ou seja, há predominância do público feminino nesta área. Assim, esse considerável interesse em jogos pode ser usado para atrair novos estudantes para os cursos de computação, através de disciplinas de programação introdutórias utilizando design de jogos, do inglês *game design*.

Interagir com jogos é uma atividade recreativa muito popular entre os jovens. Muitos sonham em um dia desenvolver seus próprios jogos e seguir carreira nesta área, uma vez que o mercado de jogos digitais possui grande abrangência mundial. Segundo a pesquisa realizada pela Global Games Market Report 2017 [4], o Brasil é o segundo maior produtor/consumidor de jogos digitais da América Latina, ocupando o 13º lugar no ranking das empresas que mais geram receita no mercado mundial. Neste sentido, o design de jogos pode vir a tornar-se um aliado do ensino, pois pode motivar os estudantes para os desafios de programação. Além disso, fatores intrínsecos relacionados à aprendizagem podem ser aprimorados, tais como o raciocínio lógico e matemático, a capacidade de abstração, o processo para decompor problemas, o trabalho em equipe, entre outras [5, 6, 7, 8, 9].

O desenvolvimento de jogos de computador envolve muitos aspectos da computação, incluindo computação gráfica, interação humano-computador, inteligência artificial, engenharia de software, entre outros. Portanto, criar um jogo comercial de última geração é uma tarefa demasiadamente árdua que requer orçamento considerável e uma equipe de desenvolvimento audaciosa. No entanto, há alternativas mais modestas, por exemplo, variações do *Pac-Man*, *Space Invaders* ou simples jogos de plataforma.

Este capítulo apresenta uma proposta de oficina de programação com uso da biblioteca *Pygame*, a qual faz uso de recursos da linguagem *Python* e permite que alunos iniciantes em programação utilizem seus recursos avançados para criar jogos simples, mas completos, considerando o processo de *game loop*. Além desta seção introdutória, o capítulo ainda apresenta o Referêncial Teórico (seção 1.2), o *game designer* (seção 1.3), a Proposta de Plano de Aula para Construção do Jogo "Papa Bolinhas" (seção 1.4) e as Considerações Finais (seção 1.5).

1.2. Referencial Teórico

Esta seção apresenta os conceitos e fundamentos que permeiam este trabalho apresentando os benefícios que os jogos trazem para a área da educação. Em seguida, são apresentados os conceitos de *game design*, a linguagem *Python*, a biblioteca *Pygame*, a definição de *game loop* e a Inteligência Artificial nos jogos.

1.2.1. Benefícios dos Jogos para a Educação

Estudos sobre tendências educacionais têm apontado um crescente interesse pela utilização de jogos digitais como mecanismo influenciador de aprendizagem [10, 11, 12]. Em 2006 a Federação de Cientistas Americanos [13] relatou os resultados de uma pesquisa sobre as vantagens da utilização de jogos em contextos educacionais. Dentre as vantagens, destaca-se o fato de que um jogo permite extender o contexto de aprendizado para fora da sala de aula, pois podem demandar um tempo relativamente maior do que a proposta de uma aula tradicional.

Os jogos também são fundamentalmente baseados em um paradigma que envolve desafios [14], recompensas descobertas e *feedbacks* que contrastam com o paradigma educacional tradicional de um tutor contando uma história e o aprendiz apenas escutando e testando o que foi dito. A revista Science [15] reforçou o mesmo entendimento, acrescentando evidências sistemáticas na utilização de jogos digitais como ferramentas de apoio ao processo de ensino-aprendizagem. A partir destes estudos diversas instituições de ensino passaram a adotar a integração de jogos e conteúdos programáticos de forma regular.

De acordo com Fogaça [16] os jogos permitem desenvolver estruturas cognitivas, facilitando o desenvolvimento de habilidades, tais como observar e identificar problemas, comparar e classificar conceitos, relacionar e inferir resultados, além de desenvolver a criatividade, perseverança e sociabilidade. No âmbito educacional, é possível utilizar os jogos de duas formas distintas [17]: a) tradicionalmente, onde o aluno exerce o papel de jogador com o objetivo de superar, individualmente ou coletivamente, os desafios educacionais apresentados pelo jogo; e b) excepcionalmente, onde o aluno assume a responsabilidade pela construção de um jogo. Neste último, o aluno é engajado nas três atividades básicas relacionadas ao *game design* que compete em: criar, desenvolver e produzir um jogo. Cabe ao professor atuar como responsável pelo gerenciamento de todas essas atividades, verificando e validando o crescimento intelectual de cada aluno e identificando as barreiras individuais e coletivas no aprendizado [18].

1.2.2. Conceitos de Game Design

De acordo com Brathwaite [19], *game design* é um processo para se definir um jogo e sua estrutura como, por exemplo, cenários, personagens, objetivos, pontuações, desafios, entre outros. O *game design* é o que permite que o jogador sinta-se motivado a alcançar objetivos, seguir regras, de forma a obter progresso no jogo. O *game design* é composto de três atividades básicas [19]: criação, desenvolvimento e produção.

A primeira etapa, a de criação, permite determinar e estruturar o funcionamento do jogo, ou seja, definir a mecânica do jogo. Envolve a especificação dos seguintes elementos: descrição geral, objetivos, regras e limites, cenário, pontuações e *feedbacks*. Para Adams [20], fundador do *International Game Developers Association* (IGDA), a história,

os personagens, os gráficos e a narrativa quando existir, além dos aspectos tecnológicos e audiovisuais são relevantes no *game design*. No entanto, a mecânica do jogo é o fio condutor que levará o jogador a ter um maior engajamento no jogo.

Na etapa de desenvolvimento, o desenvolvedor transpõe o *game design* para uma aplicação funcional. Para isso é preciso codificar, ou seja, construir algoritmos e códigos utilizando uma linguagem de programação. Nesta etapa, a partir da criação de um protótipo do jogo, pode-se realizar testes para verificação de possíveis falhas lógicas e de interação. Neste estudo é utilizada a linguagem *Python*, pois além de ser de fácil aprendizado, quando comparado às demais linguagens procedurais, possui uma biblioteca própria para desenvolvimento de jogos (*Pygame*) com vários recursos disponíveis para criação de um jogo completo, tais como carregamento e manipulação de imagens, *game loop*, contador de tempo, colisão, entre outros.

A terceira etapa, a de produção, é a fase de finalização e distribuição do jogo, ou seja, a fase de comercialização. Envolve definir estratégias de marketing e formas de distribuição.

1.2.3. O *Game Loop*

Cada jogo consiste de uma sequência de capturar a entrada do usuário, atualizar o estado do jogo, lidar com a Inteligência Artificial, tocar trilhas sonoras e efeitos sonoros e mostrar o jogo para o jogador. Em outras palavras, o *game loop* é o ciclo de vida do jogo [21], uma estrutura de *loop* infinito que deve variar entre as ações a seguir [22]:

- **Receber comandos**: neste estágio o programa deve estar preparado para tratar os eventos gerados a partir de dispositivos de entrada de dados, como teclado, mouse e *joystick*. Neste momento, também deve ocorrer o filtro destes eventos, de forma a tratar apenas as informações necessárias. Por exemplo, um jogo que utiliza apenas o clique do mouse nao necessita tratar eventos de movimento do *mouse*; ou jogos que utilizam teclas específicas do teclado, podem ignorar as demais teclas, que eventualmente tenham sido clicadas pelo jogador.

- **Simulação**: nesta fase devem ser executadas chamadas de funções responsáveis por realizar os cálculos e simulações dos objetos na tela, ou consequências de ações que tenham sido executadas em loops passados. Desta forma, as atualizações de pontuação de jogadores, verificação de colisões de objetos e novos posicionamentos devem ser realizados antes da próxima etapa, responsável pela exibição dos resultados na tela.

- **Mostrar resultados**: responsável por exibir as atualizações na tela, movimentar os elementos em tela (animações), retirar imagens de objetos atingidos, executar efeitos sonoros, entre outros. Esta etapa está diretamente relacionada ao *feedback*, sendo muito importante para a continuidade do jogo e interesse do jogador.

Estas três etapas se repetem até o jogador finalizar a execução do jogo. O *game loop*, portanto, pode ser considerado o batimento cardíaco do jogo.

1.2.4. A Mecânica do Jogo

De acordo com Tozour [23], a Inteligência Artificial (IA) em jogos, também conhecida como Game IA, preocupa-se em como o sistema deve agir e não em como pensar. Sua importância está nos resultados visando proporcionar diversão e jogabilidade. Apenas as definições e implementações da interatividade do jogo com o jogador, dentro do *game loop*, como a definição das regras e pontuações, não são suficientes para garantir uma boa jogabilidade e manter a atenção do jogador. Mesmo em jogos educacionais e principalmente quando usamos *game design* com a finalidade de ensino de técnicas de programação, como é o caso deste trabalho, torna-se primordial criar técnicas de IA.

Existem diversas técnicas e algoritmos que permitem criar simulações em diferentes situações e condições adversas, dificilmente reproduzidas exaustivamente em ambientes reais [21, 24, 25, 26]. Entre as principais técnicas encontram-se os algoritmos determinísticos e os padrões de movimento, que geralmente são compostos por algoritmos de perseguição e evasão [27]; a máquina de estado também tem o seu uso crescente, pois define onde os personagens podem se encontrar, bem como os comportamentos baseados nestas situações; os sistemas especialistas auxiliam na resolução de problemas com estados globais, abrangentes e complexos, fazendo uso de uma base de regras; algoritmos de busca que utilizam heurísticas de busca inteligente, de maneira a inserir no personagem uma natureza inteligente; algoritmos genéticos, que geram o DNA virtual, ou seja, são capazes de criar uma série de valores que representam os parâmetros de uma espécie modelada, dentre outras técnicas.

De acordo com Silva [27], a Game IA nem sempre é usada para encontrar uma forma de vencer o jogador, afinal neste caso a máquina poderia se tornar invencível. A maior importância de inserir IA nos jogos é dar a impressão de que a máquina realmente está competindo com o jogador, devendo também haver uma margem de erros para que a jogabilidade se torne realista. Resumindo, na Game IA o objetivo principal é usar a IA para obter diversão. Pode-se entender que sua importância está de acordo com o resultado que se espera gerar, e não como o jogo chegará a um determinado resultado.

1.2.5. A Linguagem Python e a Biblioteca Pygame

Segundo relatório publicado pela IEEE Spectrum [28], *Python* é a linguagem de programação mais utilizada por desenvolvedores de todo o mundo, sendo amplamente utilizada por grandes empresas como Google, Facebook, NASA, Twitter, etc. A escolha desta linguagem, para desenvolvimento deste trabalho, ocorreu devido às suas características inerentes que facilitam o aprendizado de conceitos de programação sem conflitar constantemente com os detalhes de sintaxes, como ocorre com outras linguagens como C, C++ e Java.

A linguagem Python foi criada no final da década de 90 e se popularizou rapidamente entre os desenvolvedores de jogos, principalmente por facilitar a interação com uma série de dispositivos gráficos [21]. Os dados coletados do mouse, teclado e *joystick(s)*, por exemplo, precisam estar sincronizados com as imagens exibidas na tela. A complexidade aumenta quando é preciso oferecer suporte a múltiplos sistemas operacionais e hardwares. Nestes casos, o uso de bibliotecas especializadas, como o *Pygame*, minimiza esta complexidade.

O *Pygame* está estruturado em módulos que podem ser utilizados independentemente. A Tabela 1.1 apresenta os principais módulos da biblioteca *Pygame* [21].

Tabela 1.1: Principais módulos da biblioteca *Pygame*.

Módulo	Descrição
pygame.display	Acesso ao display
pygame.draw	Desenho de formas geométricas, linhas e pontos
pygame.event	Gerenciador de eventos
pygame.font	Manipulação e exibição de fontes
pygame.image	Manipulação de arquivos de imagem (jpg, png, dentre outros)
pygame.key	Captura de teclas do teclado
pygame.mouse	Captura de movimento e comandos do mouse
pygame.music	Manipulação de arquivos de som
pygame.rect	Criação e gerenciamento de áreas retangulares
pygame.surface	Criação e gerenciamento de imagens na tela
pygame.time	Monitorar o tempo de execução

Uma das vantagens da biblioteca *Pygame* é a forma de lidar com um dos elementos mais importantes do desenvolvimento de jogos, os eventos. Durante a execução de qualquer programa, com interface gráfica e interação com usuários, como é o caso dos jogos, são gerados eventos responsáveis pela identificação e notificação de ações ocorridas a qualquer momento. Independente do que estiver sendo executado no programa, o usuário pode executar um clique no mouse, movimentar o mouse, apertar uma tecla do teclado ou tudo ao mesmo tempo. O *Pygame* tem a capacidade de armazenar todos os eventos gerados em uma fila de processamento, de forma a permitir o tratamento individual de cada um desses eventos.

A seguir, é apresentado um exemplo de uso desses módulos do Pygame para criar um jogo simples. O exemplo é iniciado com a especificação do *game design* e, na sequência, segue-se com os detalhes de programação do jogo através da linguagem Python e da biblioteca Pygame. Trata-se de um exemplo simples e que pode ser utilizado por professores em oficinas de programação.

1.3. Game designer do jogo “Papa Bolinhas”

Ao planejar um curso ou uma aula, é necessário pensar em como tal atividade será desenvolvida e como o público alvo irá receber e interagir com o conteúdo a ser apresentado. A tarefa de definição do design de um jogo ocorre de forma semelhante, ou seja, é preciso decidir como este jogo será, como o público alvo irá recebê-lo e interagir com ele.

Na proposta deste minicurso, as atividades do *game design* são exercitadas com foco no ensino de conceitos de linguagem de programação. Para isso, é apresentado o detalhamento das etapas de desenvolvimento de um jogo simples, denominado "Papa Bolinhas", utilizando a linguagem Python com a biblioteca Pygame.

1.3.1. Etapa de Criação

O "Papa Bolinhas"é um jogo simples desenvolvido para reforçar o aprendizado de técnicas de linguagem de programação. Os elementos de *game design* utilizados neste exemplo são: descrição geral, objetivo do jogador, regras, cenários e seus elementos, pontuações, elementos responsáveis pela geração de obstáculos ao jogador e *feedbacks*.

Descrição: o jogo "Papa Bolinhas"é composto de um elemento responsável por coletar bolinhas vermelhas pelo cenário. As bolinhas vermelhas surgem aleatoriamente no topo da tela, e vão deslizando verticalmente até sumirem pela parte inferior da tela. O Papa Bolinhas é representado por uma bolinha branca (maior do que as vermelhas) movimentado pelo jogador. Cada bolinha vermelha coletada pelo Papa Bolinhas contabiliza 1 ponto para o jogador. O jogador comanda o Papa Bolinhas por meio de setas do teclado. Um temporizador é apresentado na tela para mostrar o tempo de jogo. O jogo termina quando completar 60 segundos.

Objetivo do jogador: controlar o Papa Bolinhas para coletar o maior número possível de bolinhas vermelhas no intervalo de 60 segundos.

Regras do Jogo: o jogador deve movimentar o Papa Bolinhas por toda a tela, usando as setas direcionais do teclado (direita, esquerda, acima e abaixo), tentando capturar o maior número de bolinhas no tempo determinado pelo jogo (60 segundos).

Cenário: é composto de elementos gráficos que permitam diferenciar os elementos representativos do jogador, os limites de tela e a área em que o personagem pode atuar. Neste caso, nosso personagem é representado por um círculo branco (Papa Bolinhas) e bolinhas vermelhas que serão capturadas pelo Papa Bolinhas, um texto representando a pontuação e outro para o temporizador (Figura 1.1).

Figura 1.1: Exemplos de telas do Jogo Papa Bolinhas

Pontuações: a cada bolinha vermelha capturada pelo Papa Bolinhas, o jogo contabiliza 1 ponto para o jogador.

Feedback: este elemento é importante para que o jogador perceba suas ações no jogo: a) o movimento do Papa Bolinhas acompanhando a sequência de teclas pressionadas; b) o desaparecimento das bolinhas vermelhas à partir da colisão do Papa Bolinhas, sinalizando que a bolinha foi capturada; c) o efeito sonoro executado a partir da colisão do Papa Bolinhas com as bolinhas vermelhas indicando acerto; d) a pontuação incrementada em uma unidade indicando a quantidade de bolinhas capturadas até o momento; e)

o temporizador em contagem regressiva indicando o tempo restante para a atividade do jogador; f) o final de jogo quando o temporizador zera e é apresentada uma nova tela com o resumo do jogo.

Mecânica do Jogo: A mecânica deste jogo é relativamente simples, visto que as bolinhas devem se movimentar de forma descendente do topo da tela até o final da tela. Pode-se usar algoritmos bem simples como um gerador de números aleatórios para posicionar cada bolinha em sua posição inicial, mas sempre no topo da tela. As bolinhas são geradas uma por vez e, enquanto não forem capturadas ou enquanto não desaparecerem no limite inferior da tela, se movimentam no sentido descendente, do topo até o final da tela. No entanto, é possível criar algoritmos mais complexos que proporcionem uma maior interatividade e aumente a dificuldade do jogo, como por exemplo, fazer com que as bolinhas "fujam" do Papa Bolinhas; ou que a velocidade de movimento aumente conforme o passar do tempo de jogo; entre outros e desta forma ter uma real implementação de Inteligência Artificial.

1.3.2. Etapa de Desenvolvimento

O desenvolvimento de um jogo digital contempla sua codificação em linguagem de programação. Para a implementação deste exemplo, foi utilizada a versão 3.4.0 da linguagem de programação Python, integrada ao ambiente de programação IDLE (*Python's Integrated Development Environment*) e a biblioteca *Pygame* versão 1.9.1 no Sistema Operacional Windows (as mesmas versões podem ser utilizadas em diferentes distribuições Linux e no MacOS).

A seguir é apresentado um passo-a-passo para a construção do jogo "Papa Bolinhas". Cabe destacar que o exemplo aqui apresentado faz uso de conceitos de programação como a criação de variáveis, operações matemáticas, uso de estruturas de controle e repetição, listas e tuplas (estrutura presente e comum na linguagem Python), cujos princípios e sintaxes de linguagem não serão abordados a fundo neste capítulo. É necessário que o aluno ou pesquisador que venha a desenvolver este exemplo, tenha conhecimentos básicos da linguagem Python.

Parte 1 - Preparando a Tela do Jogo e seus Elementos

Primeiro é necessário criar um novo arquivo no ambiente IDLE e salvá-lo em um diretório no computador. Neste exemplo, foi utilizado um arquivo chamado "*papaBolinhas.py*" armazenado em um diretório chamado "*Jogo*". Dentro do diretório "*Jogo*", foram criados dois subdiretórios: o diretório "*img*" para armazenar as imagens e o diretório "som" para armazenar os arquivos de áudio.

O código fonte deve iniciar com a importação das bibliotecas necessárias para o desenvolvimento do programa. Neste exemplo, é utilizada a biblioteca para desenvolvimento de jogos em Python (*Pygame*); a biblioteca padrão para eventos do sistema operacional e constantes (*sys* e *locals*) e a biblioteca para geração de números aleatórios do Python (*random*), conforme ilustrado no trecho de Código 1.

```
1 # Importa as bibliotecas utilizadas
2 import sys, pygame
3 from pygame.locals import *
4 from random import *
```

Código 1: Importação das bibliotecas

O Código 2 apresenta a inicialização da biblioteca *Pygame* através da chamada *pygame.init()* (linha 2). Esta tarefa faz com que a biblioteca inicialize diferentes módulos e drivers responsáveis por capturar informações dos dispositivos de entrada/saída, como por exemplo: identificar o movimento de mouse, teclas de teclado pressionadas, acesso aos recursos e drivers de áudio e vídeo, entre outros. Em seguinda, define-se o tamanho (largura e altura) da janela principal do jogo (linha 4), acessado através do módulo *display* do *Pygame* (linha 5). Este módulo retorna um objeto do tipo *surface* (responsável por representar e armazenar atributos de diferentes elementos em tela). Assim, pode-se inserir todos os elementos necessários na tela.

A tela do jogo pode ter uma cor de fundo ou uma imagem de fundo que a torne mais contextualizada ao tema do jogo. Para os casos em que se deseja trabalhar com imagem de fundo, deve-se tomar cuidado com a resolução desta imagem para que não seja nem superior nem inferior ao tamanho da tela. O ideal é editar a imagem em algum editor gráfico, para garantir melhor qualidade, e exportar esta imagem para os formatos PNG ou JPG, com os quais o *Pygame* trabalha melhor. Neste exemplo, é utilizada uma imagem de fundo no formato PNG carregada através do módulo *pygame.image* que permite a manipulação de arquivos de imagens.

```
1 # Inicializa a biblioteca pygame
2 pygame.init()
3 # Cria a surface
4 size = (800, 600)
5 screen = pygame.display.set_mode(size)
6 # Define um título para a janela
7 pygame.display.set_caption("Papa Bolinhas")
```

Código 2: Inicialização de Pygame e configuração da janela do jogo.

O Código 3 mostra como carregar uma imagem para dentro do programa e armazená-la na variável *imagem* (linha 2). Logo mais, será ilustrado como aplicá-la na tela do jogo. Cabe ressaltar a necessidade de especificar o caminho no qual o arquivo da imagem está armazenado. Neste exemplo, a imagem está armazenada dentro do diretório "img" localizada dentro do diretório "Jogo" onde encontra-se o arquivo principal *papaBolinhas.py*.

Uma boa prática de programação é definir, através de variáveis, as cores utilizadas no jogo. O *Pygame* trabalha com o padrão RGB (*Red, Green e Blue* - vermelho, verde, azul). Desta forma, as variáveis podem ser definidas como triplas que armazenam as cores desejadas nesta escala. Cada valor dos elementos da tripla deve então variar entre 0 e 255.

Neste exemplo, será necessária a utilização de quatro cores e, portanto, quatro variáveis para definir os valores preto, branco, amarelo e vermelho, como ilustrado nas linhas 4 a 5 do Código 3.

```
1 #Carrega a imagem de fundo
2 imagem = pygame.image.load("imagem_fundo.png")
3 # Define as cores em RGB
4 BLACK  = (0, 0, 0)
5 WHITE = (255, 255, 255)
6 YELLOW = (255, 255, 0)
7 RED = (255, 0, 0)
```

Código 3: Carregamento da imagem e definição da paleta de cores

Duas variáveis precisam ser declaradas para armazenar e controlar a posição e velocidade de movimento do jogador na tela, ou seja, do Papa Bolinhas (Código 4). O posicionamento dos elementos em tela é definido pelo par ordenado (x, y). Para o *Pygame* a tela do computador é representada por *pixels*, sendo que a contagem se inicia no canto superior esquerdo da tela. Pode-se posicionar o Papa Bolinhas em qualquer posição, por exemplo, na posição x=400 e y=300 (linha 3). Já a velocidade de movimento do Papa Bolinhas pode ser representado por um par ordenado (x, y). A velocidade é dada pela quantidade de *pixels* por unidade de tempo com que o elemento deverá ser redesenhado na tela: quanto maior a quantidade de pixels, maior será a percepção de movimentação rápida. Neste exemplo, está sendo utilizado 5 *pixels* como valor de velocidade, tanto para o Papa Bolinhas quanto para as bolinhas vermelhas a serem capturadas (linha 4). É preciso também definir as posições das bolinhas vermelhas. Elas podem ser declaradas com valores iguais a zero (linha 11 e 12), mas dentro do *game loop* esses valores serão atualizados para os valores gerados pelo gerador de números aleatórios.

```
1 # Declara a lista que controla a
2 #posição X e Y do Papa Bolinhas
3 posicaoPapaBolinhas = [400, 300]
4
5 # Armazena, em uma lista, a velocidade de
6 # movimentação do Papa Bolinhas
7 velocidadePapaBolinhas = [5, 5]
8
9 # Cria variáveis com valores para controlar
10 # o posicionamento do círculo vermelho.
11 X_vermelho = 0
12 Y_vermelho = 0
```

Código 4: Configuração da posição e velocidade de movimento do Papa Bolinhas

Agora é preciso criar uma variável booleana, chamada “criar” (linha 3 do Código

5 para controlar quando a bolinha vermelha será gerada. Por exemplo, quando o valor da variavel for True e o *game loop* for inicializado, gera-se a bolinha vermelha e imediatamente ajusta-se a variável “criar” para False, pois uma nova bolinha vermelha só poderá ser gerada novamente quando a bolinha corrente sumir da tela, seja por meio do Papa Bolinhas, seja pela saída na parte inferior da tela.

```
1  # Cria uma variável booleana para controlar
2  # a geração do círculo vermelho
3  criar = True
```

Código 5: Configuração da posição e velocidade de movimento do Papa Bolinhas

O Código 6 adiciona textos na tela para exibir cronômetro e a atualização dos pontos do jogador, declarando uma fonte (linha 2). Em seguida, é preciso criar a variável "placar"que irá armazenar os pontos do jogador (linha 3). No início do jogo, o placar apresenta valor zero, pois o jogador ainda não capturou nenhuma bolinha.

```
1  # Especificando o formato da fonte
2  font = pygame.font.SysFont('sans',40)
3  placar = 0
```

Código 6: Configuração da posição e velocidade de movimento do Papa Bolinhas

Dentro do *game loop* essa variável será atualizada quando o jogador pontuar no jogo, ou seja, quando o jogador capturar as bolinhas vermelhas. É preciso também definir a variável responsável por controlar a velocidade com que os quadros serão exibidos na tela (linha 14), realizado através da função *pygame.time.Clock()*.

```
1  # variável clock para controlar a velocidade de
2  # quadros por segundo
3  clock = pygame.time.Clock()
4
5  # Cria o objeto Clock para permitir um sincronismo
6  # do tempo em segundos com os frames por segundo
7  CLOCKTICK = pygame.USEREVENT+1
8
9  # configurado o timer para execução a cada 1 segundo
10 pygame.time.set_timer(CLOCKTICK, 1000)
11 temporizador = 60
```

Código 7: Configuração de fonte de texto e tempo de jogo

Parte 2 - Criando o *Game Loop*

O *game loop* é o motor de jogo, ou seja, responsável por checar, quadro a quadro, os eventos ocasionados pelas condições e regras do jogo e as ações do jogador. O *game loop* é constituído de um laço de repetição *"while"* (linha 2 do Código 8), o qual permanece sempre como verdadeiro "True", pois deve ficar checando quadro a quadro, os eventos de teclado e execução de cálculos de simulação do jogo, até que uma das condições de parada seja satisfeita (final de jogo ou final do temporizador).

```
1  # Loop principal do jogo
2  while True:
3    # Verifica se algum evento aconteceu
4    for event in pygame.event.get():
5      # Fecha a aplicação em eventos de saída
6      if event.type == pygame.QUIT:
7        pygame.quit()
8        sys.exit()
9      #capturando evento de relógio a cada 1 segundo
10     if event.type == CLOCKTICK:
11       temporizador = temporizador -1
```

Código 8: *Criando o game loop e tratando os eventos*

No Código 8, a primeira coisa que o *game loop* precisa fazer é checar se há eventos na fila de eventos. Para isso, é utilizado um laço "for" para iterar a fila de eventos gerado pelo sistema operacional, capturados e encapsulados pelo *Pygame* (linha 4). O primeiro evento a ser verificado é o clique no botão de fechar a janela do jogo. Caso positivo, deve-se fazer a chamada da função responsável por fechar a janela "pygame.QUIT" (linha 8), e que deslocará todos os objetos de memória. Em casos de jogos mais complexos, neste ponto é possível gravar informações em arquivos para manter as últimas ações do usuário, para que, quando o jogo for inicializado novamente, o ponto de inicialização possa ser mesmo em que fora interrompido. Ainda dentro do laço "for" é preciso atualizar a variável temporizador (linha13), inicializada com sessenta segundos e que diminuirá em uma unidade a cada segundo.

Após a execução do laço "for" é preciso verificar se o tempo de jogo se esgotou e, em caso positivo, interromper o *game loop* com um comando "break" (linha 3 do Código 9). Esta ação indica o final de jogo.

```
1  #finalizando o jogo
2  if temporizador == 0:
3    break
```

Código 9: Captura de eventos de teclado

Prosseguindo com o *game loop*, deve-se fazer a verificação das teclas pressionadas pelo usuário. A verificação será feita apenas para as teclas válidas dentro da regra do jogo, os demais eventos serão automaticamente descartados, por não terem um tratamento

específico. Para identificar as teclas posicionais do teclados, são utilizadas variáveis definidas pelo próprio *Pygame*, e dentro de cada condicional (linhas 10, 12, 14 e 16 do Código 10).

```
1  # Captura o evento se alguma tecla for pressionada
2  pressed = pygame.key.get_pressed()
3
4  # Verifica qual tecla (seta) foi pressionada e atualiza a
5  # lista de posições de acordo com a velocidade
6  if pressed[pygame.K_UP]:
7    posicaoPapaBolinhas[1] -= velocidadePapaBolinhas[1]
8  if pressed[pygame.K_DOWN]:
9    posicaoPapaBolinhas[1] += velocidadePapaBolinhas[1]
10 if pressed[pygame.K_LEFT]:
11   posicaoPapaBolinhas[0] -= velocidadePapaBolinhas[0]
12 if pressed[pygame.K_RIGHT]:
13   posicaoPapaBolinhas[0] += velocidadePapaBolinhas[0]
```

Código 10: Captura de eventos de teclado

Para cada tecla deve ser realizado o cálculo de posicionamento e velocidade do Papa Bolinhas (linhas 11, 13, 15 e 17, Código 10). Em outras palavras, deve-se ter coerência com as teclas pressionadas, movimentando o personagem de acordo com as teclas correspondentes, de forma a fornecer ao jogador um *feedback* correto. Por exemplo, para movimentar o Papa Bolinhas para a direita, então deve-se incrementar a variável que representa a posição "x" com o valor definido para a velocidade. Desta forma, se o Papa Bolinhas estiver na posição (0,0), após pressionar a tecla da direita, a nova posição será (0,5), ou seja, movimenta-se o Papa Bolinhas cinco *pixels* no eixo horizontal. Da mesma forma, se a tecla da esquerda for pressionada, então o Papa Bolinhas deverá retroceder menos cinco *pixels* a partir da sua posição atual. O mesmo princípio se repete ao eixo y.

Após o tratamento de todos os eventos gerados pelos dispositivos de entrada, deve-se desenhar os elementos na tela em suas novas posições. O *Pygame* carrega os elementos em camadas, ou seja, atualiza a tela de acordo com os elementos que estão sendo gerados. Desta forma, é preciso primeiro criar a camada mais inferior, a de plano de fundo. Aqui será utilizada a imagem carregada e armazenada na variável "imagem". Para apresentar a imagem de fundo, deve-se utilizar o comando "blit" para fazer com que a imagem apareça na tela, conforme ilustrado no Código 11. Esse comando é acessado através da *surface* gerada ou seja, o objeto *screen* criado na Parte 1.3.2, fora do *game loop*.

O personagem do Papa Bolinhas, como mencionado nas sessões anteriores, é representado por um elemento geométrico simples, um círculo branco gerado à partir do módulo de desenho de círculos fornecido pelo próprio *Pygame* "pygame.draw.circle", conforme ilustrado no Código 11 (linha 2). Nesta chamada de função passa-se como parâmetro a *surface* principal (a qual representa a tela do jogo), informando sobre qual *surface* o novo elemento deve ser desenhado, a variável representando a cor do círculo

(*WHITE*), a lista com os valores da posição em que o círculo deverá ser desenhado na tela e o tamanho da raio do círculo determinando o tamanho do círculo em *pixels*.

```
#atualiza a imagem de fundo na tela
screen.blit(imagem, (0, 0))

# Desenha um círculo branco na tela
pygame.draw.circle(screen, WHITE, posicaoPapaBolinhas, 20)

# Aqui é definida a posição inicial da bola vermelha
if criar == True:
  X_vermelho = randint(20,580)
  Y_vermelho = 20
  criar = False
```

Código 11: Gerando o Papa Bolinhas

O próximo passo é implementar a IA do jogo e apresentar os resultados na tela. Neste jogo, a IA é definida como sendo a geração aleatória de bolinhas vermelhas no topo da tela e movimentadas de cima para baixo. Seguindo as regras definidas no *game design*, deve-se apresentar apenas uma bolinha vermelha por vez na tela. Uma nova bolinha vermelha só será gerada e posicionada, caso a bolinha vermelha corrente seja capturada ou desapareça na parte inferior da tela. Para isso, utiliza-se uma variável booleana, definida na Parte 1.3.2, representando computacionalmente a bolinha vermelha, se e somente se, esta variável for *True*. Se a verificação for válida, então a posição x da bolinha vermelha é sorteada podendo variar entre 20 e 780 *pixels*. Qualquer valor sorteado dentro deste intervalo está dentro do parâmetro definido como largura da tela (800), desconsiderando 20 pixels referente ao tamanho da bolinha vermelha (raio 10) para que a mesma não seja desenhada contendo partes dela para fora das dimensões da tela, conforme ilustrado no Código 11 (linhas 6 a 8). O próximo passo é atualizar a movimentação da bolinha vermelha, levando em consideração a velocidade de movimentação na tela (5 *pixels* no eixo y) e a representação visual da bolinha na tela, conforme ilustrado no Código 12.

```
# Velocidade de queda do círculo Vermelho
Y_vermelho += 5

# posicionando bolinha vermelha
posicaoBolasVermelhas = [X_vermelho,Y_vermelho]

# Desenha o círculo vermelho com raio de tamanho 10
pygame.draw.circle(screen, RED, posicaoBolasVermelhas, 10)
```

Código 12: Gerando e posicionando a bolinha vermelha

Após as atualizações de movimentações dos elementos na tela, é preciso verificar

se houve colisão entre o Papa Bolinhas e a bolinha vermelha. Este cálculo de colisão serve para indicar ao jogador se ele conseguiu coletar a bolinha vermelha que estava caminhando sobre a tela. A colisão, neste jogo, é calculada da seguinte forma: se a posição x e y do Papa Bolinhas for a mesma posição x e y da bolinha vermelha, contabilizando 20 *pixels*, é considerada a colisão, conforme ilustrado no Código 13 (linhas 4 a 7). Se esta verificação for verdadeira, então a variável "criar" é atualizada para *True* (linha 8) para que na próxima iteração do *game loop*, uma nova bolinha vermelha seja gerada. Em seguida, o placar é atualizado somando-se um ponto (linha 9). Em seguida, o arquivo de áudio "catch.mp3" é executado (linha 10) a partir do módulo *mixer* do *Pygame*, linha 11.

```
1  # Cálculo de colisão entre as bolinhas
2  # CB: Círculo Branco    CV: Círculo Vermelho
3  if (posicaoPapaBolinhas[1] + 20 >= Y_vermelho - 10  and
4        posicaoPapaBolinhas[1] - 20 <= Y_vermelho + 10) and
5         (posicaoPapaBolinhas[0] +20  >= X_vermelho - 10 and
6            posicaoPapaBolinhas[0] - 20 <= X_vermelho + 20):
7         criar = True
8         placar = placar+1
9         pygame.mixer.music.load('som/catch.mp3')
10        pygame.mixer.music.play(0)
```

Código 13: Detecção de colisão

Agora é preciso renderizar os textos na tela e para isso utiliza-se a variável $timer1$ recebendo o retorno da função "render", a qual recebe como parâmetro a string "Tempo", concatenada com o valor da variável temporizador (convertida para *string*), o valor *True* para mostrar a fonte e a cor da fonte na tela. Em seguida, usa-se o módulo "blit" para exibir o texto na tela na posição $x = 50$ e $y = 50$. Desta forma, o texto do cronômetro aparecerá no canto superior esquerdo da tela, conforme ilustrado no Código 14 (linhas 2 a 4). O mesmo princípio é utilizado para renderizar o texto do Placar na posição $x = 600$ e $y = 500$.

```
1  # Renderizando as fontes do cronômetro na tela do usuário
2  timer1 = font.render('Tempo ' + str(temporizador),
3    True, (YELLOW))
4  screen.blit(timer1, (50, 50))
5
6  # Renderizando as fontes do placar na tela
7  score1 = font.render('Placar '+str(placar), True, (WHITE))
8  screen.blit(score1,(600, 50))
```

Código 14: Renderizando textos de placar e tempo na tela

Todos os elementos do jogo foram apresentados em camadas, seguindo as regras e lógica do jogo. Agora é preciso atualizar a tela para o jogador. Isso é feito por meio do

módulo “display.flip()” a cada iteração do *game loop*, conforme ilustrado no Código 15 (linha 2). É preciso também limitar a taxa de quadros por segundo para 60fps (linha 6). Isso faz com que o jogo seja atualizado a cada sessenta frames por segundo garantindo uma boa velocidade de animação para cada interação do jogador com o jogo.

```
1 # Atualiza a tela com todos os elementos
2 pygame.display.flip()
3 # Taxa de quadros 60 quadros por segundo (60fps)
4 clock.tick(60)
```

Código 15: Atualização da tela do jogo

Parte 3 - Trabalhando o Final de Jogo

Após sair do *game loop* é interessante mostrar para o jogador a sua pontuação total, ou seja o Placar do Jogo. Para isso, é preciso limpar a tela. Neste exemplo, o texto referente ao placar é apresentado em vermelho sobre uma tela branca. Isso pode ser feito desenhando um retângulo branco do tamanho da tela, conforme ilustrado em Código 16 (linhas 2 e 3). Em seguida, deve-se criar um texto contendo uma mensagem de "Fim de Jogo", seguido do valor final da variável placar (linhas 4 a 6). Em seguida, é preciso apresentar o texto através da função “blit” (linha 7). Por último, é necessário atualizar a tela através do módulo “display.flip()” (linha 10).

```
1  #Limpando a tela do jogo
2  frame = pygame.draw.rect(screen, (WHITE),
3           Rect((0, 0), (800, 600)))
4  textofinal = font.render('Fim de Jogo - Placar final: ' +
5                 str(placar), True, (RED))
6  size = font.size(str(textofinal))
7  screen.blit(textofinal, (size[0]/2., size[1]/2.))
8  #Atualiza a tela
9  pygame.display.flip()
10 #Pequeno loop game esperando o usuário fechar a janela
11 while True:
12     for event in pygame.event.get():
13         if event.type == pygame.QUIT:
14             pygame.quit()
```

Código 16: Tela de pontuação final do jogo

Como o *game loop* foi encerrado, não é mais possível recuperar o evento para fechar a janela, caso o jogador queira encerrar o jogo. Deste modo, é preciso criar um pequeno *game loop* apenas com a verificação do evento em questão, conforme ilustrado no Código 16, linha 12. Ao clicar no botão fechar da janela, a aplicação é encerrada.

Este foi o passo a passo para a criação do jogo “Papa Bolinhas” contruído na linguagem de programação *Python* e com a biblioteca *Pygame*. Como se trata de um jogo dedicado apenas ao ensino de programação, não será apresentada a etapa de produção. A seguir, é apresentado um Plano de Aula, com a proposta para utilização deste tutorial em oficinas de programação.

1.4. Proposta de Plano de Aula para Construção do Jogo “Papa Bolinhas”

Esta seção apresenta uma proposta para a aplicação da metodologia de *game design*, considerando a especificação e o desenvolvimento do jogo "Papa Bolinhas". Espera-se que este Plano de Aula possa ser reproduzido por professores e pesquisadores interessados em novas metodologias de ensino e aprendizagem de programação.

1.4.1. Plano de Aula

Um plano de aula tem como objetivo principal apresentar a distribuição do conteúdo programático que será trabalhado durante uma disciplina ou curso. Nele deve constar os objetivos pedagógicos do curso, o conteúdo programático, o número de aulas e o tempo necessário para trabalhar cada tópico abordado dentro da disciplina. A Tabela 1 apresenta o Plano de Aula da oficina intitulada “Construção de Jogos Utilizando *Python* e *Pygame*”.

Tabela 1.2: Plano de Aula

PLANO DE AULA	
Nome da Oficina	Construção de Jogos Utilizando Python e *Pygame*
Conteúdo Programático	1. Benefícios dos jogos para a educação 2. Desenvolvimento de jogos como estratégia de ensino de programação 3. Conceitos de game design 4. Construção de um documento de game design 5. Implementação de um jogo em *Pygame* 6. Testes de funcionalidade
Duração da oficina	A oficina terá duração de 6h sendo duas aulas de 3h. Uma aula oferecida em um período e outra aula em outro período ou dia.
Objetivos pedagógicos	Apresentar uma proposta de ensino dos conceitos de linguagem de programação através do desenvolvimento de um jogo, utilizando a linguagem Python.
Materiais necessários	Sala com computadores, internet, projetor e quadro. Softwares: python 3.4 ou superior, *Pygame* versão 1.9.1 e IDLE versão 3.4.0. Cada aluno deverá ter o seu computador de trabalho. Será utilizada uma apresentação com os conceitos que serão apresentados ao longo da oficina.
Avaliação da oficina	Ao final da oficina, será solicitado que os participantes respondam um questionário on line de satisfação.

Tabela 1.3: Aula 1

Descrição da atividade	Tempo
Apresentação dos professores e objetivos da oficina.	10 min
Apresentação dos participantes e preenchimento do questionário de perfil.	15 min
Benefícios dos jogos para a educação	10 min
Demonstrações de jogos criados por estudantes de oficinas anteriores	10 min
Conceitos de game design	10 min
Construção de um documento de *game design*	20 min
Intervalo	15 min
Implementação do jogo (parte 1): Criação dos elementos visuais do jogo (textos, desenhos e imagens)	60 min
Implementação do jogo (parte 2): Iniciando o *game loop* e capturando eventos do teclado.	30 min
TOTAL	180 min

Tabela 1.4: Aula 2

Descrição da atividade (Aula 1)	Tempo
Recapitulando a aula anterior	20 min
Desenvolvimento do jogo (parte 3): criando os desafios do jogo.	35 min
Intervalo	20 min
Desenvolvimento do jogo (parte 4): atualizando a pontuação do jogo.	35 min
Desenvolvimento do jogo (parte 5): testes de funcionalidade e feedbacks dos outros participantes.	40 min
Aplicação do questionário de satisfação.	20 min
Encerramento.	10 min
TOTAL	180 min

1.5. Considerações Finais

Este capítulo teve como objetivo apresentar um estudo sobre a aplicação da metodologia de *game design* para apoiar o ensino de programação de computadores. Foi apresentado um passo-a-passo do jogo "Papa Bolinhas"utilizando a biblioteca Pygame. Espera-se que este tutorial possa ser reproduzido por pesquisadores e professores interessados na utilização de novas metodologias para o ensino de programação. Para apoiar esta estratégia,

propôs-se um Plano de Aula contendo objetivos, proposta pedagógica, materiais necessários e cronograma de execução. Foi realizado um estudo piloto com estudantes de cursos de computação, e foi possível concluir que a metodologia de *game design* aqui apresentada motivou os estudantes ao aprendizado de programação.

Espera-se que este trabalho possa vir a contribuir com as pesquisas relacionadas ao ensino de programação, via desenvolvimento de jogos, pois notadamente é uma abordagem que motiva os estudantes a aprender além do conteúdo proposto. Isso porque durante o desenvolvimento dos jogos os alunos se deparam com diversos problemas de programação que eles próprios concebem, dependendo do jogo que eles querem desenvolver, como, por exemplo, como animar um personagem com tiros, como construir e navegar entre várias telas, como criar um ranking das últimas jogadas, etc. Esses problemas os estimulam a pesquisar, construir algoritmos, testar as várias soluções e ajustar possíveis erros de lógica surgidos a partir da modificação das versões de jogos anteriores. Isso faz com que o aluno desenvolva o raciocínio lógico e matemático e que desperte nele a curiosidade e o fascínio que provocam importantes mudanças na atitude e no empenho acadêmico.

É importante reconhecer que os jovens universitários estão cada vez mais imersos em diversas fontes de informação como a Internet, fazendo uso de dispositivos de interação como computadores, smartphones, tablets etc. Encontrar formas de motivar o alunos para o estudo de conceitos e técnicas teóricas é de suma importância para o bom aproveitamento desse aluno. Como próximas etapas, espera-se desenvolver um projeto de construção de um jogo completo e funcional que envolva a interdisciplinariedade de forma efetiva.

Referências

[1] da Educação Superior, I. C. (2016). Divulgação dos Principais Resultados. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). Ministério da Educação (MEC).

[2] Scott, M. J., & Ghinea, G. (2013). Educating programmers: A reflection on barriers to deliberate practice. arXiv preprint arXiv:1311.0390.

[3] Pesquisa Game Brasil (2018). Comportamento, Consumo e Tendências do Gamer Brasileiro. Game Brasil. 5ª Ed. PGB.

[4] Global Games Market Report (2017). Trends, Insights, and Projections Toward 2020. New Zoo.

[5] KAFAI, Yasmin B. Minds in play: Computer game design as a context for children's learning. Routledge, 2012.

[6] Leutenegger, S., & Edgington, J. (2007, March). A games first approach to teaching introductory programming. In ACM SIGCSE Bulletin (Vol. 39, No. 1, pp. 115-118).

[7] Overmars, M. (2004). Teaching computer science through game design. Computer, 37(4), 81-83.

[8] Marques, D. L., Costa, L. F. S., de Azevedo Silva, M. A., & Rebouças, A. D. D. S. (2011, November). Atraindo alunos do ensino médio para a computação: Uma Experiência Prática de Introdução à Programação utilizando Jogos e Python. In Anais do Workshop de Informática na Escola (Vol. 1, No. 1, pp. 1138-1147).

[9] Rebouças, A. D. D. S., Marques, D. L., Costa, L. F. S., & de Azevedo Silva, M. A. (2010). Aprendendo a ensinar programação combinando jogos e Python. In Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE) (Vol. 1, No. 1).

[10] Ke, Fengfeng. Designing and integrating purposeful learning in game play: A systematic review. Educational Technology Research and Development, v. 64, n. 2, p. 219-244, 2016.

[11] , Mansureh; Hirumi, Atsusi; Bai, Haiyan. The effects of modern math computer games on learners? math achievement and math course motivation in a public high school setting. British Journal of Educational Technology, v. 38, p. 49-259, 2008.

[12] WU, Wen-Hsiung et al. Re-exploring game-assisted learning research: The perspective of learning theoretical bases. Computers & Education, v. 59, n. 4, p. 1153-1161, 2012.

[13] FEDERATION OF AMERICAN SCIENTISTS. Summit on educational games: Harnessing the power of video games for learning. Whashington, DC. 2006. http://informalscience.org/sites/default/files/Summit_on_Educational_Games.pdf

[14] Henrique, B., & Armando, J. (2016). Digital Games And Education: A Possibility To Change Pedagogical Approach In Formal Teaching. Revista Iiberoamericana de Educacion.

[15] Miller, L. M., Chang, C. I., Wang, S., Beier, M. E., & Klisch, Y. (2011). Learning and motivational impacts of a multimedia science game. Computers & Education, 57(1), 1425-1433.

[16] , T. D. O. (2016). A utilização das tecnologias digitais no espaço escolar: perspectivas e desafios para o ensino.

[17] Martins, V.F.; Eliseu, M.A.; Omar, N.; de Castro, M.L.A; Correa, A.G.D. Using Game Development to Teach Programming. In: Handbook of Research on Immersive Digital Games in Educational Environments, IGI Global, 2018.

[18] Simkova, Monika; Using of computer games in supporting education. Procedia-Social and Behavioral Sciences 141 (2014): 1224-1227.

[19] Brathwaite, B., & Schreiber, I. (2009). Challenges for game designers. Nelson Education.

[20] ADAMS, Ernest, 2010. Fundamentals of Game Design. Second Edition. Berkeley: New Riders.

[21] McGugan, W., & Kinsley, H. (2015). Beginning Python Games Development: With Pygame. Apress.

[22] Craven, P. (2015). Program Arcade Games: With Python and Pygame. Apress.

[23] Tozour, P. (2002). The evolution of game AI. AI game programming wisdom, 1, 3-15.

[24] J. R. Koza, “Genetic programming: on the programming of computers by means of natural selection (complex adaptive systems)", Cambridge, MA, 1992.

[25] S. M. Lucas, “Ms pac-man competition,"ACM SIGEVOlution, vol. 2, no. 4, pp. 37?38, 2007.

[26] Togelius, Julian. How to run a successful game-based AI competition. IEEE Transactions on Computational Intelligence and AI in Games, v. 8, n. 1, p. 95-100, 2016.

[27] Silva, M.P. (2015). Aplicação da Inteligência Artificial em Jogos: Uma Abordagem Histórica (Trabalho de Conclusão de Curso, Faculdade de Computação de Montes Carlos.

[28] Cass, S. (2018). Interactive: The 2018 top ten programming languages. IEEE Spectrum, July, 18.

[29] HINES, Pamela J.; JASNY, Barbara R.; MERVIS, Jeffrey. Adding a T to the three R’s. 2009. doi:10.1126/science.323.5910.53a.

[30] CHELL, Jesse. The Art of Game Design: A book of lenses. AK Peters/CRC Press, 2014.

9. Capítulo 9

Autores:

Flávio Eduardo Aoki Horita
Universidade Federal do ABC (UFABC)
email: flavio.horita@ufabc.edu.br

Valdemar Vicente Graciano Neto
Universidade Federal de Goiás (UFG)
email: valdemarneto@inf.ufg.br

Rodrigo Pereira dos Santos
Universidade Federal do Rio de Janeiro (UNIRIO)
email: rodrigo.pereira@uniriotec.br

Capítulo 9

Design Science Research em Sistemas de Informação e Engenharia de Software: Conceitos, Aplicações e Trabalhos Futuros

Flávio Eduardo Aoki Horita, Valdemar Vicente Graciano Neto e Rodrigo Pereira dos Santos

Abstract

Artifacts are elements employed to describe knowledge obtained from research, which may support the development of new products, either organizational or computational. In this context, the elaboration and evaluation of such artifacts become an essential element in new research works in informatics, particularly those focused on information systems and software engineering. This chapter aims at presenting the foundations of Design Science Research (DSR) by explaining its essential elements, as well as by introducing supporting frameworks and research methodologies. Three application cases that applied DSR in the areas of information systems and software engineering were used as objects of analysis. Results indicated that these areas still lack research works that explore the potential use of DSR.

Resumo

Artefatos são elementos empregados para descrever o conhecimento obtido em pesquisas e, assim, apoiar o desenvolvimento de novos produtos organizacionais e/ou computacionais. Neste contexto, a elaboração e avaliação desses artefatos torna-se imprescindível para novas pesquisas em computação, principalmente, em sistemas de informação e engenharia de software. Este capítulo busca explorar o conhecimento acerca de Design Science Research (DSR), compilando seus elementos fundamentais, apresentando frameworks de suporte e metodologias de pesquisa. Três casos de aplicação em pesquisas nas áreas de sistemas de informação e engenharia de software são utilizados como objetos de análise. Os resultados obtidos apontam que a literatura nestas áreas ainda carece de estudos para explorar o potencial da DSR.

1.1. Introdução

O avanço científico é pautado em um processo lento e sucessivo de passos bem definidos, quais sejam: (1) observação de um fenômeno, (2) elaboração de hipóteses, (3) criação de um modelo ou solução para descrever o fenômeno observado, (4) avaliação do modelo/solução concebido (usualmente composta por aplicação do modelo/solução em um cenário, coleta e análise dos resultados, confirmação ou refutação das hipóteses, e criação de teorias e leis pela generalização dos resultados, caso possível) e (5) criação de conhecimento a partir da experiência vivenciada e indicação de novas linhas de investigação. O conhecimento construído ao final desse processo é composto por uma ou mais contribuições científicas, que podem ser classificadas em dois grupos distintos [Gregor e Hevner, 2013]: (a) descritivas: caracterizadas por descrever um fenômeno de interesse e todas as variáveis de influência; e (b) prescritivas: compreendem o conhecimento estruturado e elaborado para entendimento do fenômeno.

De maneira análoga, a literatura tem adotado o termo "*o que?*" para indicar estudos com enfoque nas contribuições descritivas, ao passo que o termo "*como?*" é atribuído para as contribuições prescritivas. Destaca-se ainda que ambas as categorias de contribuições são relacionadas entre si ao ponto que avanços no campo descritivo são representados por elementos no âmbito prescritivo e, progressivamente, elas contribuem para compor o conhecimento adquirido. Com base na contribuição almejada, diversos métodos científicos podem ser empregados para a condução de pesquisas nas áreas de Sistemas de Informação (SI) e Engenharia de Software (ES) [Runeson e Höst, 2009; Araújo et al., 2017]. Contribuições descritivas, pelo seu foco exploratório, descritivo e analítico, são frequentemente alcançadas por diversos métodos. Dentre eles, destacam-se os estudos de caso, originalmente, empregados em pesquisas das ciências sociais cujo objetivo é analisar um fenômeno de interesse em seu contexto real; os experimentos (quasi-) controlados usualmente empregados para testar uma ou mais hipóteses associadas a variáveis de interesse; e as análises da literatura utilizadas para sintetizar um conhecimento de interesse presente na literatura.

Contribuições prescritivas, por outro lado, buscam sintetizar e consolidar o conhecimento obtido por meio de artefatos, em geral, definidos como "uma coisa" possivelmente transformada em um objeto (p. ex., modelos) ou processo (p. ex., software) [Goldkuhl, 2002]. Nesse contexto, *Design Science Research* (DSR)[1] tem se destacado como um dos métodos de pesquisa adotado para solução de problemas por meio da elaboração e avaliação de tais artefatos [Peffers et al., 2007]. Sob tal prisma, DSR configura-se como uma metodologia útil e complementar aos métodos empíricos clássicos utilizados para validar hipóteses levantadas sobre um fenômeno observado [Wohlin et al., 2003]. DSR apoia-se na premissa de que, para criar soluções inovadoras, é necessário avançar o estado da arte. Logo, a aplicação sistemática de DSR para solução de um problema produz avanço no estado da arte como efeito colateral da criação de artefatos inovadores.

Pela sua natureza aplicada, DSR tem sido utilizada com sucesso em pesquisas conduzidas no domínio de SI. Por outro lado, DSR ainda tem sido pouco explorada no domínio de ES. Neste contexto, este capítulo tem como objetivo principal prover uma

[1] Diversas traduções têm sido empregadas em referência ao termo *Design Science Research* na língua portuguesa; p. ex., Ciência de Projeto e/ou Ciência do Artificial. Contudo, nenhuma delas ainda encontrou espaço na literatura brasileira e, por conta disso, optou-se por manter o termo original neste trabalho.

base de conhecimento visando permitir uma inserção e/ou aprimoramento de pesquisas em SI e ES que adotam (ou queiram adotar) a DSR. Os fundamentos acerca do método são definidos e introduzidos para homogeneizar a terminologia utilizada, bem como para apoiar a utilização de *frameworks* e metodologias de desenvolvimento. A aplicação desses conceitos é demonstrada por meio de duas pesquisas que adotaram a DSR como método de pesquisa nas áreas de SI e ES. Dessa forma, as principais contribuições deste capítulo são as seguintes: (a) compilar o conhecimento existente sobre os fundamentos de DSR em SI; (b) expandi-los para o domínio de ES; (c) demonstrar a aplicabilidade dos conceitos em pesquisas reais; e (d) estabelecer tal conhecimento para permitir o aperfeiçoamento das pesquisas e amadurecimento da comunidade latino-americana de informática.

Para tanto, este capítulo está organizado da seguinte forma: A Seção 1.2 sumariza o estado da arte sobre DSR descrevendo os fundamentos, *frameworks* e métodos para auxiliar na condução de trabalhos. A Seção 1.3 detalha a aplicação da DSR em duas pesquisas das áreas de SI e ES. Por fim, a Seção 1.4 apresenta algumas discussões e as considerações finais deste capítulo.

1.2. Fundamentação Teórica

Esta seção apresenta uma revisão quanto aos conceitos relacionados à DSR, bem como processos, *frameworks* e técnicas que dão suporte ao método. Além disso, conceitos nas áreas de SI e ES associados à DSR serão apresentados.

1.2.1. *Design Science Research* (DSR)

DSR é um método empírico (baseado em evidência) para criação sistemática de soluções inovadoras [Hevner et al., 2004]. Neste sentido, DSR prescreve um conjunto rigoroso de etapas para levar a avanços no estado da arte. Para tantos, o principal conceito em DSR é o artefato, isto é, soluções para problemas materializadas em modelos, métodos, processos e ferramentas. DSR vale-se da premissa de que não é possível conceber um artefato inovador para solucionar um problema em aberto sem que isso necessariamente passe pelo avanço do estado da arte no domínio em questão. Como o próprio nome diz, trata-se de uma "ciência de projeto", isto é, conhecimento sistemático e repetível sobre um fenômeno observado necessariamente criado como fruto da concepção de soluções.

A Figura 1.1 apresenta uma estrutura conceitual com os elementos essenciais para auxiliar no entendimento, criação e avaliação de pesquisa que empregam a DSR [Hevner et al., 2004]. O domínio de aplicação, representando o ambiente, determina o fenômeno de interesse composto por pessoas, organizações, tecnologias, objetivos e atividades. Este domínio fornece então as necessidades de negócio que irão nortear a definição do problema de pesquisa. Enquanto, por outro lado, a base de conhecimento provê a fundamentação teórica para a condução da pesquisa; p. ex., as variáveis de análise e métodos de avaliação, bem como modelos e *framework* existentes. Esta base será elemento fundamental para compor o conhecimento aplicável durante a pesquisa. Dessa forma, a DSR utiliza as necessidades adjacentes do meio prático na definição de problemas, que foram solucionados empregando artefatos construídos e avaliados com conhecimento científico. As contribuições propostas a partir da DSR serão relevantes e importantes somente quando atenderem às necessidades do domínio de aplicação e também apoiarem na condução de novas pesquisas [Hevner et al., 2004].

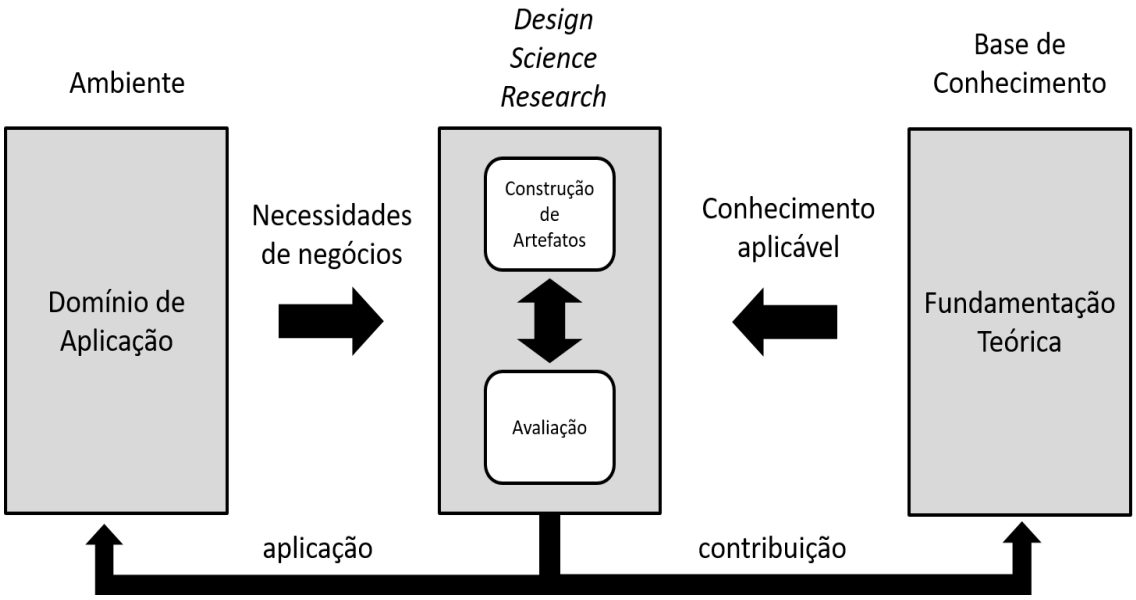


Figura 1.1. Estrutura Conceitual de DSR. Adaptado de [Hevner et al., 2004].

1.2.2. Artefatos x Teorias

DSR tem o(s) artefato(s) como resultado-alvo após a condução do processo metodológico, principalmente, porque eles ajudam a registrar o conhecimento obtido para apoiar o desenvolvimento de novos produtos, organizacionais e/ou computacionais. Esses artefatos podem ser categorizados nos seguintes tipos: construtos, modelos, métodos, ou instâncias [Hevner et al., 2004; Sein et al., 2011]. *Construtos* são compreendidos como vocabulários e símbolos essenciais para a descrição e análise de fenômenos de interesse; p. ex., taxonomias. *Modelos* são as representações de um problema e algumas das possíveis soluções; p. ex., arquiteturas conceituais de sistemas. *Métodos* compreendem um conjunto de atividades para possibilitar a execução de uma tarefa. *Instâncias* representam realizações dos elementos (p. ex., modelos, métodos, construtos) em um ambiente natural. Estas instâncias também são especializações de conhecimento adquirido para domínios com características similares ou diferentes; p. ex., sistemas para o gerenciamento de vendas. Dessa forma, estes artefatos são usualmente convertidos e traduzidos em algo material e concreto [Gregor e Hevner, 2013]; p. ex., um conjunto de algoritmos torna-se um sistema de informação.

Em um nível mais abstrato do conhecimento, as teorias são compostas por elementos não materiais importantes para prover um conhecimento consolidado quanto à existência do artefato. Em outras palavras, teoria refere-se a "um conjunto de ideias e declarações abstratas empregadas para entender, explicar, descrever e prever o comportamento de um fenômeno de interesse" [Gregor, 2006]. A Tabela 1.1 apresenta a relação entre contribuições e níveis de abstração (indicada pelas setas bidirecionais).

Uma teoria envolve o uso de artefatos em sistemas sociotécnicos, ou seja, essa teoria foca em analisar tanto os sistemas computacionais e os aspectos sociais atrelados, quanto a interação entre esses dois elementos. Assim, a taxonomia de teorias em SI tem cinco tipos [Gregor, 2006]: (I) teorias de análise, (II) teorias de explicação, (III) teorias de predição, (IV) teorias de explicação e predição, e (V) teorias de projeto e ação. Tabela 1.2 apresenta esses tipos com sua descrição e atributos.

Tabela 1.1. Relação entre as contribuições e o nível de abstração do conhecimento. Adaptado de [Gregor e Hevner, 2013].

Estado Alcançado	**Nível de abstração**	**Contribuição**	**Exemplos**
Abstrato, completo e maduro	*Nível 3*	Conhecimento generalizado e consolidado de um fenômeno.	Teorias
↕ ↕ ↕ ↕	*Nível 2*	Conhecimento emergente com algum detalhamento.	Construtos, métodos, modelos e princípios de projeto.
Específico, limitado e imaturo.	*Nível 1*	Detalhamento específico a um domínio.	Instâncias específicas.

Tabela 1.2. Taxonomia dos tipos de teorias. Adaptado de [Gregor, 2006].

Tipo	***Descrição e Atributos***
I. Análise	Responde à pergunta: "O que é?". Ou seja, a teoria não aprofunda para além da análise e descrição. Sem relação de causa e nem predições.
II. Explicação	Responde às perguntas: "O que é? Quem? Por quê? Quando? e Onde?". Ou seja, a teoria fornece esclarecimentos, mas não tem o objetivo de realizar previsões com precisão. Não existem proposições testáveis.
III. Predição	Responde às perguntas: "O que é? e O que será?". Ou seja, a teoria fornece previsões e proposições testáveis; contudo, não fornece maior explicações sobre os fenômenos.
IV. Explicação e predição	Responde às perguntas: "O que é? Quem? Por quê? Quando? Onde? e O que será?". Ou seja, a teoria fornece previsões, proposições testáveis e explicações acerca dos fenômenos.
V. Projeto e ação	Responde à pergunta: "Como fazer algo?". Ou seja, a teoria fornece prescrições explícitas (p. ex., métodos, técnicas e funções) sobre como construir um artefato.

Como o desenvolvimento de uma teoria pode ter diversas etapas, os tipos de teorias possuem uma inter-relação, os artefatos elaborados por uma teoria são subsídios para a concepção (e desenvolvimento) de uma outra teoria. Por exemplo, as teorias I e II sempre podem fornecer artefatos que contribuam para o desenvolvimento das demais teorias. A Figura 1.2 apresenta as possíveis inter-relações entre os tipos de teorias.

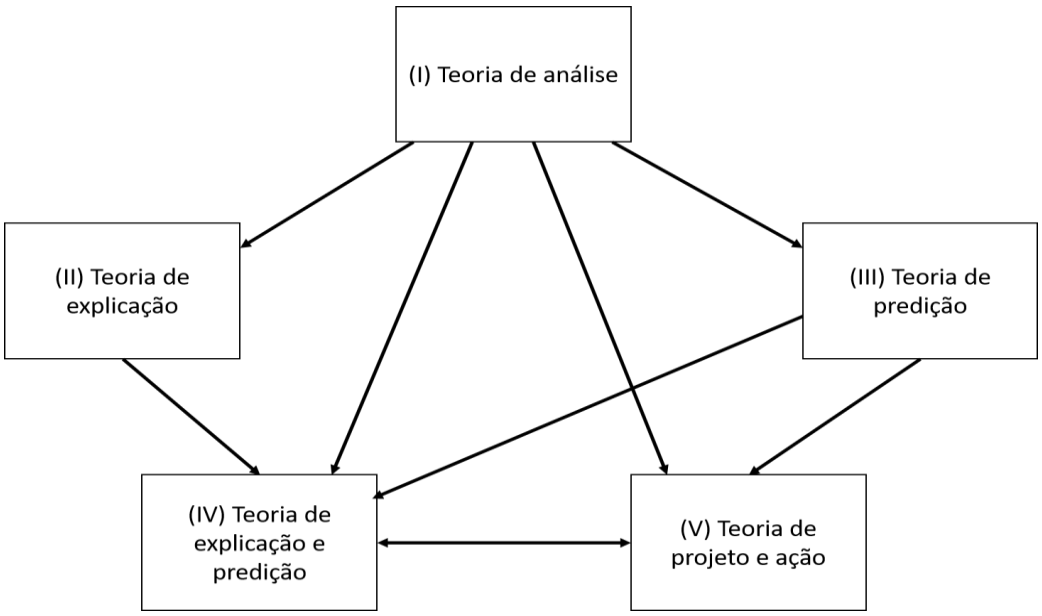


Figura 1.2. Inter-relação entre os tipos de teorias. Adaptado de [Gregor, 2006].

1.2.3. Processo Metodológico

Como descrito, DSR tem como objetivo elaborar artefatos para solucionar necessidades identificadas em domínios de aplicação específicos e, para isso, é empregado o conhecimento científico. Diversos fatores, metodológicos e científicos impactam na forma como a pesquisa será conduzida; por exemplo, quando definir as variáveis de avaliação do artefato, como garantir o rigor da pesquisa, ou, até mesmo, como realizar uma DSR. Vários são os processos metodológicos adotados pela literatura, sendo o proposto por Peffers et al. (2007) o mais utilizado.

Este processo é composto por seis atividades fundamentais, executadas, preferencialmente, de maneira sequencial com dois pontos de aprimoramento dos resultados obtidos, conforme apresentado na Figura 1.3.

A Atividade 1 (identificar o problema) utiliza as necessidades de domínio para estabelecer o problema de pesquisa a ser trabalhado. O valor da solução desenvolvida também é identificado. Com base neste problema, a Atividade 2 (definir o objetivo) tem como foco principal detalhar o objetivo da pesquisa. Viabilidade da proposta e granularidade no detalhamento são dois fatores analisados durante essa fase da pesquisa. Na Atividade 3 (projetar e desenvolver o artefato), o artefato propriamente dito é desenvolvido por meio de funcionalidades, componentes, arquiteturas, bem como relacionamentos e associações entre esses elementos. Em seguida, a utilidade do artefato é demonstrada em uma ou mais instâncias do domínio de aplicação e avaliada utilizando variáveis de análise qualitativa e/ou quantitativas, ambos objetivos das Atividades 4 (demonstrar a aplicabilidade) e 5 (avaliar o artefato), respectivamente. Por

fim, a Atividade 6 (comunicar os resultados) determina que os resultados alcançados sejam comunicados em artigo científicos, relatórios técnicos e trabalhos acadêmicos.

A partir dos resultados obtidos nas atividades de avaliação e comunicação dos resultados (Atividades 5 e 6, respectivamente), tanto o objetivo quanto o artefato desenvolvido podem ser refinados e aprimorados em novas pesquisas e estudos. Dessa forma, espera-se garantir um progresso contínuo dos resultados alcançados.

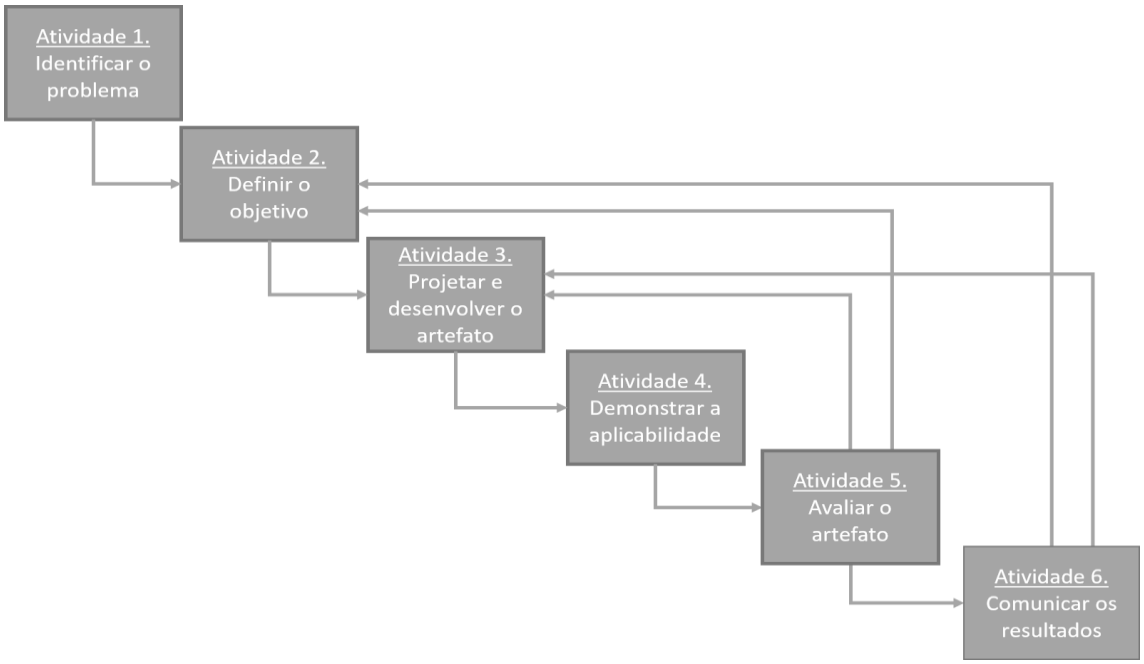


Figura 1.3. Processo metodológico para DSR. Adaptado de [Peffers et al., 2007].

1.2.4. Princípios-Guia

De maneira complementar ao processo metodológico definido por Peffers et al. (2007), um conjunto de princípios-guia foram propostos por Hevner et al. (2004) para garantir a qualidade e rigor do trabalho desenvolvido. Para isso, tais princípios-guia norteiam a análise e definição tanto do problema a ser resolvido e o artefato proposto quanto do contexto empregado para avaliação e análise. A Tabela 1.3 resume os princípios-guia.

Em conjunto, o processo metodológico e os princípios-guia são importantes ferramentas para auxiliar na condução de DSR com qualidade, rigor e relevância científica e prática. A fim de servirem como exemplos da aplicação da DSR em SI e ES, na seção seguinte duas pesquisas que empregaram essas ferramentas serão detalhadas.

1.3. Aplicações da DSR em SI e em ES

Esta seção detalha três exemplos de aplicação da DSR para pesquisas nas duas áreas de enfoque deste trabalho: SI e ES.

1.3.1. DSR em SI

A área de SI está situada na intersecção de duas outras áreas de conhecimento: ciências sociais aplicadas (p. ex., Administração) e ciência da computação (p. ex., ES). Pesquisas em SI buscam examinar o uso de sistemas computacionais no auxílio das questões

organizacionais e, com base nisso, gerar artefatos que possibilitem o aperfeiçoamento (e desenvolvimento) de melhores soluções.

Tabela 1.3. Princípios-guia para DSR. Adaptado de [Hevner et al., 2004].

#	***Princípio***	***Descrição***
1	Projeto como um artefato	DSR deve produzir um artefato.
2	Relevância do problema	O objetivo da pesquisa deve estar fundamentado em problemas identificados na prática.
3	Avaliação do projeto	A qualidade e utilidade do artefato deve ser avaliado com métodos rigorosos em cenários relacionados com o fenômeno de interesse.
4	Contribuições da pesquisa	DSR deve prover relevantes contribuições para auxiliar na elaboração de artefatos.
5	Rigor da pesquisa	Métodos rigorosos devem ser empregados na elaboração e avaliação dos artefatos.
6	Projeto como um processo de aprimoramento	Projeto de artefato representa um processo iterativo na elaboração de artefatos para resolver um problema.
7	Comunicação da pesquisa	Resultados da DSR devem ser compartilhados tanto com a literatura quanto com a prática.

Em virtude dessa interdisciplinaridade inerente a área, diversos métodos são empregados na condução de pesquisas [Araújo et al., 2017]. Quando o escopo tende a uma análise mais qualitativa do fenômeno, destacam-se os estudos de caso e as pesquisas etnográficas [Runeson e Höst, 2009]. Em contraste, experimentos e simulações têm sido mais empregados quando o objetivo é avaliar um sistema computacional. Na intersecção destes dois grupos de métodos, encontram-se pesquisas que empregam o método de pesquisa-ação [Sein et al., 2011].

Embora a DSR seja um método adotado em pesquisas de SI – o quarto método mais utilizado pela comunidade [Araújo et al., 2017] –, a literatura brasileira na área de SI ainda carece de pesquisas que não apenas ampliem o campo de conhecimento, mas também disseminem o entendimento quanto a diferenciação entre o desenvolvimento e artefatos, a concepção de teorias e a relação entre eles. Isso porque pesquisas são ainda centradas no caráter computacional onde não existe uma problematização, no caráter nominal sem criação de artefatos de TI, ou no projeto de ferramentas de TI. Na seção a seguir, é apresentado um exemplo da aplicabilidade dos fundamentos da DSR em SI.

1.3.1.1. Caso 1: Tomada de Decisão com Fontes Heterogêneas de Dados

Uma tomada de decisão precisa exige informações mais atualizadas para estabelecer a realidade da situação geral. Novas fontes de dados (p. ex., tecnologias vestíveis e

sistemas colaborativos) tem aumentado a quantidade de dados úteis disponíveis [Horita et al., 2013; Horita et al., 2015; De Assis et al., 2018; Horita et al., 2018a, Horita e De Albuquerque, 2018], a chamada "big data". Isso tem grande potencial para transformar todo o processo de negócio e melhorar a precisão na tomada de decisão [Wamba et al., 2015]. Quando combinadas, essas fontes de dados podem ser consideradas grandes em *volume* (p. ex., terabytes, zetabytes), com diferentes *velocidades* (p. ex., a cada 10 minutos) e uma *variedade de formatos* (p. ex, números, mídias e textos). Além disso, uma análise com relação à sua *veracidade* é necessária, uma vez que essas fontes de dados são desconectadas e propensas a erros. Esses são os "4Vs" que caracterizam "big data" [Hashem et al., 2015].

Embora todos esses dados abram novas oportunidades, o seu grande volume em conjunto com uma integração inapropriada e uma visualização inadequada geram problemas que tornam as informações ignoradas por tomadores de decisão [Horita et al., 2016]. Isso ocorre pois não existe uma clara compreensão sobre as necessidades dos tomadores de decisão ou sobre como os dados disponíveis podem ser usados para atender essas necessidades.

Com base nesse problema, uma DSR foi conduzida com o intuito de propor um artefato para estabelecer uma relação entre requisitos de informações em decisões com as fontes de dados existentes. A pesquisa adotou o processo metodológico definido por Peffers et al. (2007), descrito na Seção 2, e iniciou pela definição do objetivo da pesquisa, ou seja, elaborar um modelo para descrever o relacionamento entre elementos de tomada de decisão com fontes de dados existentes. Essa relação foi descrita na Atividade 3 do processo utilizando elementos abstratos associativos em um modelo e notação multi-camadas, chamado oDMN+ (do Inglês, *observation-aware Decision Model and Notation*). Este modelo teve como bases teóricas fundamentais os seguintes modelos: Modelo e Notação de Processos de Negócios (BPMN, do Inglês, *Business Process Model and Notation*), Modelo e Notação de Decisões (DMN, do Inglês, *Decision Model and Notation*) e Modelo de Observações e Medidas (O&M, do Inglês, *Observation & Measurements*) [OMG, 2013; OMG, 2014; OGC, 2013]. A Figura 1.4 apresenta o oDMN+ utilizando o Diagrama de Classes da Linguagem UML.

Em seguida, na Atividade 4, a aplicabilidade do modelo foi demonstrada no contexto do monitoramento e alertas contra desastres naturais no Centro Nacional de Monitoramento e Alertas de Desastres Naturais (Cemaden) [Horita et al., 2018]. Entrevistas e sessões de grupos focais também foram conduzidas a fim de avaliar a eficácia do modelo, ambas no escopo da quinta atividade. Para isso, as abordagens definidas por Seaman (1999), Sobreperez et al. (2008) e Saldaña (2015).

Para garantir a qualidade e rigor científico na condução da DSR, os sete princípios-guias definidos por Hevner et al. (2004) foram identificados na pesquisa. Em aderência ao primeiro princípio-guia, como mencionado na Seção 1.2.4, o problema de pesquisa foi identificado tanto na revisão da literatura quanto em trabalhos conduzidos na prática. Em seguida, o artefato foi elaborado em forma de modelo, evidência em concordância com o segundo princípio-guia. A demonstração e avaliação conduzida no âmbito do Cemaden comprovaram a aderência aos terceiro e quarto princípios-guia. O embasamento teórico adotado para definir o modelo - BPMN, DMN e O&M - fornecem indícios para o quinto e sexto princípios-guia. Também em linha com o sexto princípio-guia, o oDMN+ representa uma extensão do modelo – chamado oDMN [Horita et al.,

2016] – em busca do seu aprimoramento e refinamento. Por fim, a própria publicação no periódico quanto este trabalho cumprem o último princípio-guia [Horita et al., 2017].

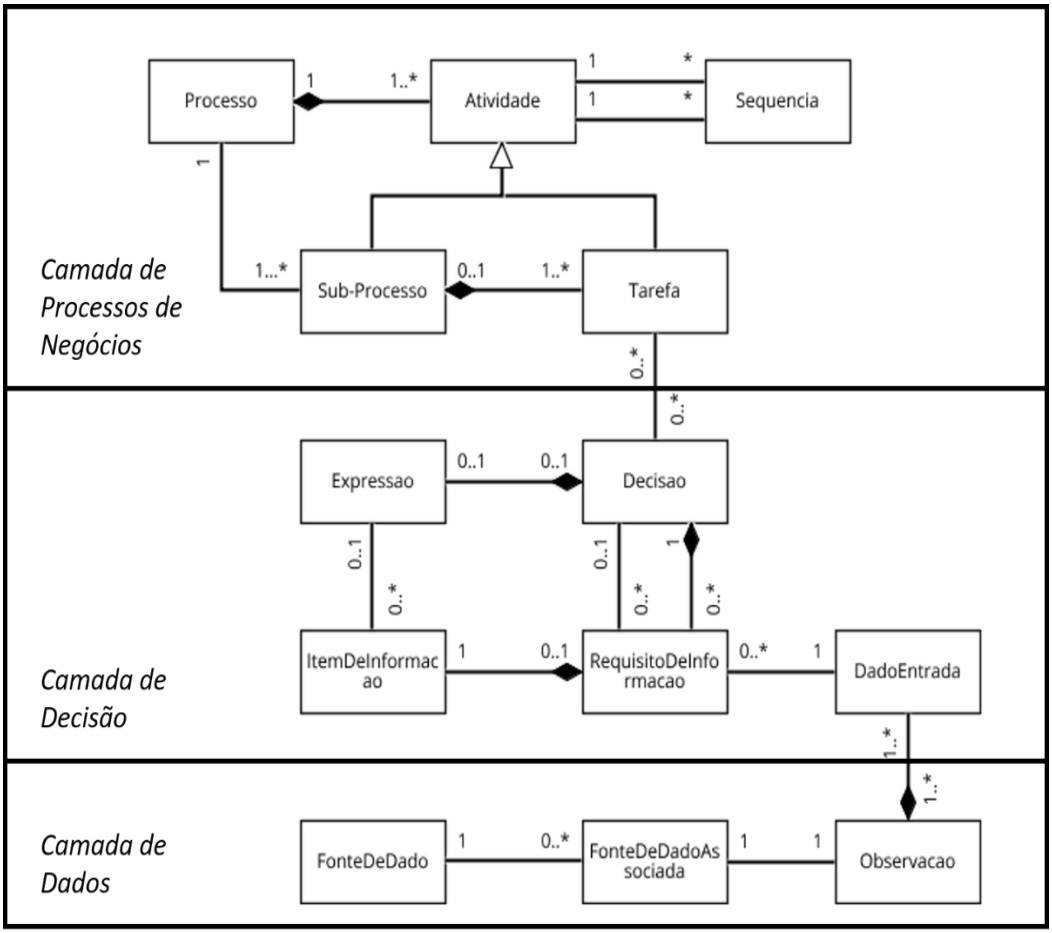


Figura 1.4. Artefato oDMN+ para estabelecer uma relação entre fontes requisitos de informações em decisões com as fontes de dados existentes.

1.3.2. DSR em Engenharia de Software

A área de ES dispõe de um rico acervo de métodos experimentais com diversos graus de rigor para serem aplicados em estudos científicos. Tais métodos são aplicados para teorizar sobre fenômenos observados, partindo de estudos piloto e provas de conceito, passando por pesquisas de opinião, estudos de caso e experimentos controlados. Nesse contexto, DSR atua em um momento anterior à aplicação da experimentação e demais métodos para confirmação ou refutação de hipóteses sobre um fenômeno. Sob a perspectiva exploratória, DSR é um método para desenvolvimento de soluções; portanto, compatível com o procedimento científico largamente aceito em ES.

Entretanto, diferentemente de SI, o objeto de estudo fim (não meio) em ES é o software, bem como os processos e métodos para a sua concepção, manutenção e evolução. Logo, a inovação esperada como resultado da aplicação de DSR perpassa a demanda de soluções de software e pela produção dos artefatos mencionados anteriormente (Construtos, Métodos, Modelos, Instâncias e Teorias). Protótipos de

software propriamente ditos tornam-se artefatos (do tipo instância) proeminentes e recorrentes quando DSR é aplicada a ES.

Uma vez que ES é uma ciência jovem, teorias são ainda escassas na área. Em contrapartida, modelos e métodos são largamente aceitos e adotados. A seguir são apresentados três casos reais que utilizaram DSR na pesquisa em SI e ES.

1.3.2.1. Caso 2: Projeto de Arquiteturas de Software para Sistemas-de-Sistemas

Sistemas de software têm se tornado cada vez mais especializados para atender domínios específicos. Em virtude disso, a construção de soluções holísticas de larga escala que envolvam uma cidade inteira, por exemplo, frequentemente exige a combinação de múltiplos sistemas independentes provenientes de diferentes domínios. Neste contexto, surgem os Sistemas-de-Sistemas (SoS, do Inglês, *Systems-of-Systems*), que compreendem conjuntos de sistemas de software operacionalmente e gerencialmente independentes que interoperam para prover funcionalidades mais complexas do que aquelas ofertadas individualmente por seus sistemas constituintes [Graciano Neto et al., 2017]. Devido à independência de seus constituintes, a arquitetura de um SoS é altamente dinâmica, com constituintes podendo entrar e sair do SoS em tempo de execução [Graciano Neto et al., 2018].

Nesse sentido, DSR foi utilizada para sistematizar a criação de um artefato que permitisse antecipar o impacto da arquitetura dinâmica do SoS nas funcionalidades ofertadas como resultado da interoperabilidade entre os seus constituintes. Seguindo o processo proposto por Peffers et al. (2007), iniciou-se o desenvolvimento do artefato. A Atividade 1 teve como objetivo identificar o problema de pesquisa. SoS são sistemas de domínio crítico, uma vez que falhas podem causar prejuízos e perdas consideráveis, incluindo apagões, acidentes, enchentes e perdas humanas. Em virtude de sua natureza inerentemente dinâmica, era necessário prover em tempo de projeto uma solução que permitisse prever o impacto da arquitetura dinâmica nos comportamentos sendo ofertados pelo SoS.

Na Atividade 2, estabeleceu-se como objetivo da pesquisa a criação de um modelo que fosse capaz de registrar de modo verossímil a estrutura e o comportamento de SoS. Na Atividade 3, um contexto foi escolhido para viabilizar a criação do artefato. Tal contexto foi um SoS de Monitoramento Urbano de Enchentes (SoSMUE) [Graciano Neto, 2016], cuja funcionalidade principal consistia em utilizar sensores inteligentes, drones e sistemas de *crowdsourcing* para monitorar rios que atravessam áreas urbanas, além de disparar alarmes de possíveis enchentes durante estações chuvosas, como ilustrado na Figura 1.5. Durante a condução do estudo (Atividade 4), uma linguagem chamada SoSADL [Oquendo, 2016] foi escolhida para realizar a modelagem da arquitetura do SoSMUE, uma vez que ela permitia a modelagem fiel da estrutura do SoS. Além disso, foram desenvolvidos também um mecanismo transformador de modelos capaz de converter automaticamente modelos de arquitetura documentados em SoSADL em modelos de simulação documentados em DEVS [Zeigler, 2000], viabilizando a predição, em tempo de projeto, do comportamento do SoSMUE durante sua execução e considerando o impacto da arquitetura dinâmica na funcionalidade.

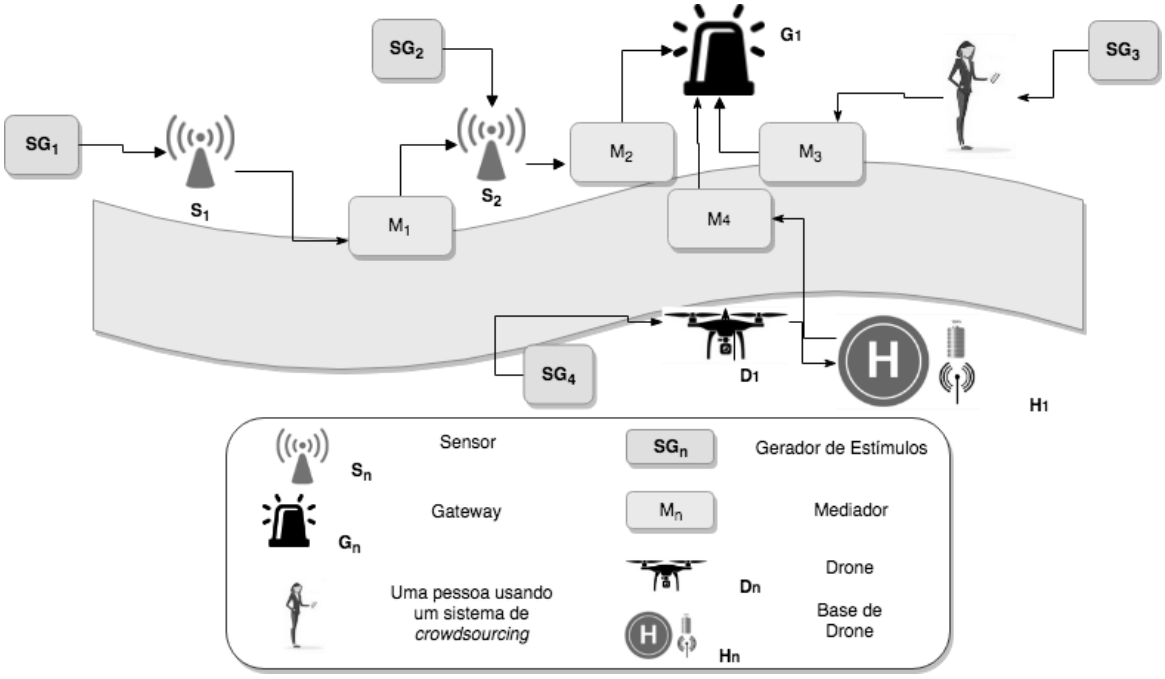


Figura 1.5. Artefato informal em pequena escala de uma arquitetura de SoS para Monitoramento Urbano de Enchentes.

Na Atividade 5, resultados foram divulgados em conferências especializadas [Graciano Neto et al., 2017a; Graciano Neto et al., 2018a; Graciano Neto et al., 2018b]. Tais resultados permitiram concluir que a associação dos formalismos SoSADL e DEVS viabilizam uma modelagem fidedigna e predição de estrutura e comportamento de um SoS real. Na Atividade 6, houve a criação de conhecimento e a produção de artefatos. Como resultado dos artefatos criados, foi concebido um método de avaliação de arquiteturas de SoS. Logo, os artefatos criados foram do tipo *modelo* e *método*; além de um elemento de software necessário para viabilizar a condução do estudo.

Os sete princípios-guias definidos por Hevner et al. (2004) foram seguidos com rigor para sistematizar a condução da DSR. O problema de pesquisa foi identificado a partir da literatura e confirmados durante a execução do estudo e *feedback* da comunidade (primeiro princípio). Um conjunto de artefatos foi elaborado, em particular um artefato de software que viabilizou o mapeamento entre dois modelos, materializando o segundo princípio-guia. Dois cenários foram utilizados para avaliar o artefato criado, alinhando-se aos terceiro e quarto princípios-guia. O embasamento teórico adotado para a especificação de modelos e a implementação do transformador revelam aderência aos quinto e sexto princípios-guia. Ao final, as publicações derivadas comunicaram os resultados à comunidade pertinente, cobrindo o último princípio-guia.

1.3.2.2. Caso 3: Análise de Demandas e Soluções em Ecossistemas de Software

Requisitos de software têm surgido de complexas redes de *stakeholders* oriundos de diferentes departamentos ou unidades organizacionais de uma organização consumidora de software, que se juntam para dividir custos e compartilhar os benefícios de desenvolver ou adquirir produtos de interesse [Santos, 2014]. Uma das principais motivações está no fato de que os produtos são adquiridos de uma rede de fornecedores que os evoluem de forma independente [Santos et al., 2014]. Além disso, as demandas

de sistemas que emergem das unidades organizacionais e as características dos produtos do mercado também evoluem independentemente, como acontece em linhas de produto de software [Bosch, 2009]. Isso tem afetado o processo de aquisição de software na indústria global, considerando a formação de redes de produção de software compostas por um conjunto de empresas que fornecem produtos e/ou serviços similares e que colaboram e/ou disputam nichos de mercado [Santos et al., 2017]. Essas redes vêm sendo chamadas de Ecossistemas de Software ou ECOS [Santos e Werner, 2011].

Nesse cenário, a DSR foi utilizada para sistematizar a criação de um artefato que permitisse analisar demandas de sistemas e soluções de mercado a fim de apoiar gestores e arquitetos de TI a anteciparem o impacto de aquisições na base de ativos de software de uma organização consumidora de software, sendo esta base considerada como plataforma deste tipo de ECOS [Santos, 2016]. Seguindo o processo proposto por Peffers et al. (2007), iniciou-se o desenvolvimento do artefato. A Atividade 1 teve como objetivo identificar o problema de pesquisa. ECOS são sistemas fortemente afetados por redes de interoperabilidade e alianças organizacionais, em que decisões de gestão e de arquitetura de TI afetam diretamente a comunidade de desenvolvedores e de usuários dos sistemas que derivam (ou formam) a plataforma. Nesse sentido, era preciso prover uma solução que permitisse avaliar o impacto da seleção de demandas e soluções no ECOS. Na Atividade 2, estabeleceu-se como objetivo da pesquisa a criação de um modelo que fosse capaz de registrar a estrutura de um ecossistema a partir de seus conceitos-chave e relações, o que gerou o *framework* para apoiar análise de ECOS denominado ReuseECOS [Santos e Werner, 2012], conforme ilustrado na Figura 6.

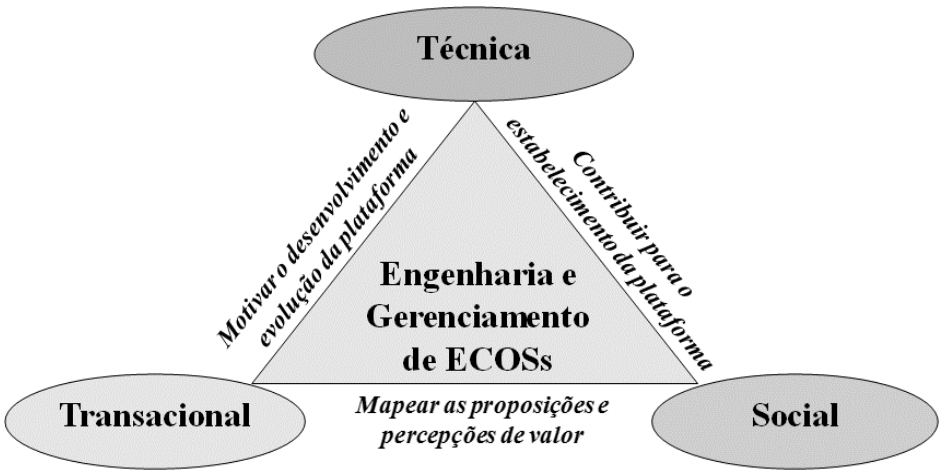


Figura 1.6. Artefato informal de um *framework* para apoiar análise de ECOS.

Na Atividade 3, um contexto foi escolhido para viabilizar a criação do artefato. Tal contexto foi um ECOS centrado em uma organização (instituição pública no Brasil) [Santos et al., 2013], cujo objetivo principal consistia em analisar as dimensões técnicas, transacional e social na preparação da aquisição de software. Durante a condução do estudo (Atividade 4), uma biblioteca de componentes e serviços de software chamada Brechó-EcoSys [Santos e Werner, 2011a] foi escolhida como base para construir um mecanismo de gestão de demandas e soluções em ECOS [Lima et al., 2016]. Esse mecanismo viabiliza a predição de impactos de aquisição no ECOS [Santos, 2016].

Na Atividade 5, resultados foram divulgados em conferências e periódicos especializados [Lima et al., 2016; Santos et al., 2017; Santos et al., 2018]. Tais resultados permitiram concluir que a análise de demandas e soluções permite antecipar ao time de gestão e de arquitetura de TI o nível de dependência de tecnologia e de sinergia de objetivos do ECOS de uma organização consumidora de software. Na Atividade 6, houve a criação de conhecimento e a produção de artefatos. Como resultado dos artefatos criados, foi concebida uma ferramenta de visualização das redes socio-técnicas formadas por objetivos de negócio, demandas, sistemas, tecnologias e fornecedores, denominada SECO2M. Logo, o artefato criado foi do tipo *modelo*, além de um elemento de software necessário para viabilizar a condução do estudo.

Os sete princípios-guias definidos por Hevner et al. (2004) foram seguidos com rigor para sistematizar a condução da DSR. O problema de pesquisa foi identificado a partir da literatura e confirmados durante a execução do estudo e *feedback* da comunidade (primeiro princípio). Um conjunto de artefatos foi elaborado, em particular um artefato de software que viabilizou a análise de demandas e soluções em ECOS, materializando o segundo princípio-guia. Dois cenários foram utilizados para avaliar o artefato criado, alinhando-se aos terceiro e quarto princípios-guia. O embasamento teórico adotado para a análise de ECOS e a implementação do ferramental de apoio revelam aderência aos quinto e sexto princípios-guia. Ao final, as publicações derivadas comunicaram os resultados à comunidade pertinente, cobrindo o último princípio-guia.

1.4. Discussões e Considerações Finais

Este capítulo apresentou uma base de conhecimento essencial para permitir não apenas a aplicabilidade da DSR, mas também seu aprimoramento em pesquisas nas áreas de SI e ES. Além da introdução dos fundamentos relacionados com o método, três casos reais, presentes na literatura, foram detalhados com o intuito de demonstrar sua aplicabilidade nas áreas de pesquisa de interesse. A partir deste estudo, pode-se concluir que a DSR é um método amplamente difundido e consolidado para estudos na área de SI. Tanto a sistemática do processo metodológico detalhado em Peffers et al. (2007) quanto os princípios-guia de Hevner et al. (2004) oferecem um arcabouço relevante para auxiliar na condução de projetos e pesquisas nessa área.

Literatura associada a DSR em ES. Em contraste, a literatura de ES apresenta poucos estudos que definem, conceitualmente, as contribuições esperadas em pesquisas conduzidas nessa área [Wieringa et al., 2007]. Além disso, são escassos os trabalhos focados em analisar o desenvolvimento de teorias e/ou processos metodológicos [Wieringa et al., 2011; Hall e Rapanotti, 2017]. As justificativas levantadas pela literatura, inicialmente, são três [Johnson et al., 2012]: a) a ES não precisa de teorias, b) a ES dispõe de teorias suficientes e c) a ES não pode ter uma teoria. Contudo, o desenvolvimento de teorias é importante e relevante, pois elas ajudariam a generalização e aprimoramento dos resultados obtidos, deixando de atuar em um ciclo vicioso de tentativas e erros em domínios específicos [Johnson et al., 2012].

Casos práticos aplicando DSR em ES. Nesse sentido, devido à característica de inovação, DSR é um método de criação com grande potencial de aplicação em ES, tanto para domínios emergentes, tais como cidades inteligentes e seus sistemas constituintes; quanto para problemas clássicos, mas ainda persistentes, tais como soluções para síntese totalmente automática (sem intervenção humana) de arquiteturas de software a partir de

um conjunto de requisitos e de análise de demandas e soluções durante a preparação de aquisição de software. Em ambos os cenários, a criação de modelos, métodos, processos e ferramentas necessariamente exige avanços no estado da arte. Logo, a identificação de problemas importantes para serem solucionados em ES pode ter solução impulsionada pela aplicação de DSR, uma vez que tal abordagem motiva, indireta e colateralmente, o avanço do estado da arte para efetiva concepção de solução viável e útil. Ademais, o conjunto de artefatos criados como resultado da aplicação de DSR em ES enseja e materializa o conhecimento produzido durante o processo criativo e inovador necessário para soluções desta natureza.

Artefatos e as contribuições científicas-práticas em ES. Outra linha de pesquisa deve focar no estudo e definição dos artefatos necessários para a consolidação dos conhecimentos obtidos na área de ES. Isso torna-se necessário, principalmente, pela tensão imposta pelos os trabalhos desenvolvidos no âmbito da Engenharia de Software Baseada em Evidências (ESBE). Esta disciplina da ES tem como objetivo prover evidências para caracterizar uma tecnologia de interesse e, para isso, devem ser conduzidos estudos primários - em contexto específico - e secundários - em contexto mais abrangente. Dessa forma, trabalhos futuros devem analisar sob o olhar da ES o quanto uma teoria desenvolvida por meio da DSR provê evidências suficientes para atender as necessidades impostas pela ESBE. Esta análise também pode contemplar a definição de uma metodologia de avaliação, bem como uma investigação sobre a adequação da taxonomia de artefatos frente às exigências da disciplina. Por exemplo, dependendo do domínio, DSR demanda não somente um único artefato, mas um conjunto que inclui modelos, métodos e protótipos para materializar e viabilizar a ideia.

Avaliação em DSR para ES. Destaca-se ainda o desenvolvimento e/ou adequação de métodos consolidados para avaliação de DSR em SI [Peffers et al., 2012] no âmbito da ES. Dentre eles, Pries-Heje et al. (2008) apresenta um *framework* que delimita a avaliação em três aspectos: "quando" a avaliação será realizada (p. ex., artefatos são analisados antes ou depois do desenvolvimento), "o que" será avaliado (p. ex., o artefato ou a metodologia) e "como" será avaliado (p. ex., em um contexto natural ou artificial). Este *framework* é complementado por outro trabalho, desenvolvido por Venable et al. (2016), no qual é definido um processo com etapas bem estabelecidas para auxiliar na execução da avaliação. Pesquisas em ES deveriam investigar a adequação de ambos os trabalhos como uma maneira de avaliar os artefatos propostos e desenvolvido na área.

1.5. Referências

Assis, L. F. F. G., Horita, F. E. A., Freitas, E. P., Ueyama, J. De Albuquerque, J. P. (2018). "A Service-Oriented Middleware for Integrated Management of Crowdsourced and Sensor Data Streams in Disaster Management". Sensors 18, 6, pp. 1689:1-1689:27.

Araújo, R., Fornazin, M., Pimentel, M. (2017). "Uma Análise sobre a Produção de Conhecimento Científico nas Pesquisas Publicadas nos Primeiros 10 anos da iSys (2008-2017)". iSys: Revista Brasileira de Sistemas de Informação 10, 4, pp. 45-65.

Bosch, J. (2009). "From Software Product Lines to Software Ecosystem". In: Proceedings of 13th International Software Product Line Conference (SPLC), San Francisco, USA, pp. 1-10.

Goldkuhl, G. (2002). “Anchoring Scientific Abstractions - Ontological and Linguistic Determination Following Socio-Instrumental Pragmatism”. In: Proceedings of the European Conference on Research Methods in Business and Management, Reading, UK, pp. 29-30.

Graciano Neto, V. V. (2016). “Validating Emergent Behaviors in Systems-of-Systems through Model Transformations”. In: Proceedings of the ACM PhD Student Research Competition at MODELS 2016 co-located with the 19th International Conference on Model Driven Engineering Languages and Systems, St. Malo, France, pp. 1-6.

Graciano Neto, V. V., Santos, R. P. dos, Araújo, R. (2017). “Sistemas de Sistemas de Informação e Ecossistemas de Software: Conceitos e Aplicações” In: Tópicos em Sistemas de Informação: Minicursos XIII Simpósio Brasileiro de Sistemas de Informação, Lavras, Brasil, pp. 22-41.

Graciano Neto, V. V., Paes, C. E. B., Garcés, L., Guessi, M., Manzano, W., Oquendo, F., Nakagawa, E. Y. (2017a) “Stimuli-SoS: A model-based approach to derive stimuli generators in simulations of software architectures of systems-of-systems”. Journal of the Brazilian Computer Society 23, 2017, pp. 13:1-13:22.

Graciano Neto, V. V. (2018). A simulation-driven model-based approach for designing software-intensive systems-of-systems architectures. Ph.D. Dissertation in Computer Sciences and Computational Mathematics, ICMC/USP, São Carlos, Brazil, 217p.

Graciano Neto, V. V., Garcés, L., Guessi, M., Paes, C. E. B., Manzano, W., Oquendo, F., Nakagawa, E. Y. (2018a). “ASAS: An approach to support simulation of smart systems”. In: Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS), Big Island, Hawaii, USA, pp. 5777-5786.

Graciano Neto, V. V. (2018b). “Externalizing patterns for simulation in software engineering of systems-of-systems”. In: Proceedings of the ACM/SIGAPP Symposium on Applied Computing (SAC), Pau, France, pp. 1-8.

Gregor, S. (2006). “The Nature of Theory in Information Systems”. MIS Quarterly 30, 3, pp. 611-642.

Gregor, S., Hevner, A. R. (2013). “Positioning and Presenting Design Science Research for Maximum Impact”. MIS Quarterly 37, 2, pp. 337-355.

Hall, J. G., Rapanotti, L. (2017). “A design theory for software engineering”. Information and Software Technology 87, 2017, pp. 46-61.

Hannay, J. E., Sjoberg, D. I. K., Dyba, T. (2007). “A Systematic Review of Theory Use in Software Engineering Experiments”. In: IEEE Transactions on Software Engineering 33, 2, pp. 87-107.

Hahsem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., Khn, S. U. (2015). “The rise of ‘big data’ on cloud computing: Review and open research issues”. Information Systems 47, 2015, pp. 98-11.

Hevner, A. R., March, S. T., Park, J., Ram, S. (2004). “Design science in information systems research”. MIS Quarterly 28, 1, pp. 75-105.

Horita, F. E. A., de Albuquerque, J. P. (2018) “The use of heterogeneous geospatial big data for decision-making”. In: Proceedings of the PhD Theses Award, Natal, Brazil, pp. 1-6.

Horita, F. E. A., de Albuquerque, J. P. Marchezini, V. (2018) “Understanding the decision-making process in disaster risk monitoring and early-warning: A case study within a control room in Brazil”. International Journal of Disaster Risk Reduction 28, 2018, pp. 22-31.

Horita, F. E. A., Vilela, R. B., Martins, R. G., Bressiani, D. A., Fernandes, G. P., de Albuquerque, J. P. (2018a) “Determining flooded areas using crowd sensing data and weather radar precipitation: a case study in Brazil”. In: Proceedings of the 13th nternational Conference on Information Systems for Crisis and Response Management (ISCRAM), Rochester, New York, USA, pp. 1040-1050.

Horita, F. E. A., de Albuquerque, J. P. Marchezini, V., Mendiondo, Eduardo M. (2017) “Bridging the gap between decision-making and emerging big data sources: An application of a model-based framework to disaster management in Brazil”. Decision Support Systems 97, 2017, pp. 12-22.

Horita, F. E. A., Link, D, Albuquerque, J. P., Hellingrath, B. (2016). “oDMN: An integrated model to connect decision-making needs to emerging data sources in disaster management”. In: Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS), Kauai, Hawaii, USA, pp. 2882-2891.

Horita, F. E. A., Albuquerque, J. P., Degrossi, L. C., Mendiondo, E. M., Ueyama, J. (2015). “Development of a spatial decision support system for flood risk management in brazil that combines volunteered geographic information with wireless sensor networks”. Computers & Geosciences 80, 2015, pp. 84-94.

Horita, F. E. A., Degrossi, L. C.; Assis, L. F. F. G.; Zipf, A., Albuquerque, J. P. (2013) “The use of volunteered geographic information (VGI) and crowdsourcing in disaster management: a systematic literature review”. In: Proceedings of the 19th Americas Conference on Information Systems (AMCIS), Chicago, USA, pp. 1–10.

Johnson, P., Ekstedt, M., Jacobson, I. (2012). “Where’s the theory for software engineering?”. IEEE software 29, 5, pp. 94-96.

Lima, T. M. P., Santos, R. P., Oliveira, J., Werner, C. M. L. (2016). “The importance of socio-technical resources for software ecosystems management”. Journal of Innovation in Digital Ecosystems 3, 2, pp. 98-113.

OGC. (2013). “Observations and Measurements (O&M)”. <http://www.opengeospatial.org/standards/om>.

OMG. (2013). “Business Process Model and Notation (BPMN), Version 2.0”.. <http://www.omg.org/spec/BPMN/2.0/>. Citations on pages 67 and 128.

OMG. (2014). “Decision Model and Notation (DMN)”. <http://www.omg.org/spec/DMN/>.

Oquendo, F. (2016). “Formally describing the software architecture of systems-of-systems with SosADL”. In: Proceedings of the 11th System of Systems Engineering Conference (SoSE), Kongsberg, Norway, pp. 1-6.

Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S. (2007). "A Design Science Research Methodology for Information Systems Research". Journal of Management Information System 24, 3, pp. 45-77.

Peffers, K., Rothenberger, M., Tuunanen, T., Vaezi, R. (2012). "Design science research evaluation". In: Proceedings of the 7th International Conference on Design Science Research in Information Systems, Las Vegas, USA, pp. 398-410.

Pries-Heje, J., Baskerville, R., Venable, J. R. (2008). "Strategies for Design Science Research Evaluation". In: Proceedings of the European Conference on Information Systems (ECIS), Galway, Ireland, pp. 255-266.

Runeson, P., Höst, M. (2009). "Guidelines for conducting and reporting case study research in software engineering". Empirical Software Engineering 14, 2, pp. 131-164.

Santos, R. P., Werner, C. M. L. (2011). "A Proposal for Software Ecosystems Engineering". In: Proceedings of the Third International Workshop on Software Ecosystems (IWSECO), Brussels, Belgium. Frankfurt: CEUR-WS, v. 746, pp. 40-51.

Santos, R. P., Werner, C. M. L. (2011a). "Brechó-EcoSys: From a Component Library to a Software Ecosystems Platform". In: Proceedings of the 12th International Conference on Software Reuse (ICSR), Pohang, South Korea, pp. 1-4.

Santos, R. P., Werner, C. M. L. (2012). "ReuseECOS: An Approach to Support Global Software Development through Software Ecosystems". In: Proceedings of the 2012 Seventh IEEE International Conference on Global Software Engineering Workshop (ICGSEW), Porto Alegre, Brazil, pp. 60-65.

Santos, R. P., Werner, C. M. L., Alves, C. F., Pinto, M. J. S., Cukierman, H. L., Oliveira, F. M. A., Egler, T. T. C. (2013). "Ecossistemas de Software: Um Novo Espaço para a Construção de Redes e Territórios Envolvendo Governo, Sociedade e a Web". In: Werner, C. M. L., Oliveira, F. J. G., Ribeiro, P. T. (Org.). Políticas Públicas: Interações e Urbanidades. Rio de Janeiro: Letra Capital, pp. 337-366.

Santos, R. P. (2014). "ReuseSEEM: an approach to support the definition, modeling, and analysis of software ecosystems". In: Proceedings of the 36th International Conference on Software Engineering (ICSE), Hyderabad, India, pp. 650-603.

Santos, R. P., Esteves, M. G. P., Freitas, G., Souza, J. M. (2014). "Using Social Networks to Support Software Ecosystems Comprehension and Evolution". Social Networking 2014, 3, pp. 108-118.

Santos, R. P. (2016). "Managing and Monitoring Software Ecosystem to Support Demand and Solution Analysis". PhD Thesis in Computer Science and Systems Engineering, COPPE/UFRJ, Rio de Janeiro, Brazil, 228p.

Santos, R. P., Lima, T. M. P., Werner, C. M. L., Tostes, L. R. M., Barbosa, G. S. (2017). "A Sociotechnical Negotiation Mechanism to Support Component Markets in Software Ecosystems". CLEI Electronic Journal 20, 3, pp. 4:1-4:25.

Santos, R. P., Werner, C. M. L., Finkelstein, A. (2018). "Ecosystems Effects on Software-Consuming Organizations: an experience report on two observational studies". In: Proceedings of 2018 12th European Conference on Software Architecture Workshops, Madrid, Spain.

Saldaña, J. (2015) “The coding manual for qualitative researchers”. Ed. 2. London: Sage Publications Ltda.

Sobreperez, P. (2008). “Using plenary focus groups in information systems research: More than a collection of interviews”. Electronic Journal of Business Research Methods 6, 2, pp. 181-188.

Seaman, C. B. (1999). “Qualitative methods in empirical studies of software engineering”. IEEE Transactions on Software Engineering 25, 4, pp. 557–572.

Venable, J., Pries-Heje, J., Baskerville, R. (2016). “FEDS: a framework for evaluation in design science research”. European Journal of Information Systems 25, 1, pp.77-89.

Wamba, S. F., Akater, S., Edwards, A., Chopin, G., Gnanzou, D. (2015). “How ‘big data’ can make big impact: Findings from a systematic review and a longitudinal case study”. International Journal of Production Economics 165, 2015, pp. 234-246.

Wieringa, Roel J. (2014). Design Science Methodology for Information Systems and Software Engineering. Berlin: Springer-Verlag Berlin Heidelberg.

Wieringa, R., Daneva, M., Condori-Fernandez, N. (2011). “The Structure of Design Theories, and an Analysis of their Use in Software Engineering Experiments”. In: Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM), Banff, Canada, pp. 295-304.

Wohlin, C., Höst, M., Henningsson, K. (2003). “Empirical research methods in software engineering”. In: Conradi R., Wang A.I. (eds) Empirical Methods and Studies in Software Engineering. Lecture Notes in Computer Science, vol 2765. Springer, Berlin, Heidelberg.

Zeigler, B. P., Praehofer, H., Kim, T. G. (2000). “Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems”. Academic Press.

10. Capítulo 10

Autores:

Luana Almeida Martins
Universidade Federal de Lavras (UFLA)
email: luana.martins1@posgrad.ufla.br

André Pimenta Freire
Universidade Federal de Lavras (UFLA)
email: apfreire@dcc.ufla.br

Paulo Afonso Júnior
Universidade Federal de Lavras (UFLA)
email: pauloa.junior@dcc.ufla.br

Heitor Costa
Universidade Federal de Lavras (UFLA)
email: heitor@dcc.ufla.br

Capítulo 10

Sistemas de Tecnologia Assistiva

Luana Almeida Martins, André Pimenta Freire, Paulo Afonso Júnior e Heitor Costa

Abstract

The difficulties of using Information and Communication Technologies by people with disabilities can be overcome or reduced by using appropriate Assistive Technology resources. There are several software systems that address accessibility issues while using such technologies. Those systems can help promote greater independence and digital inclusion of people with disabilities. In order to show the main accessibility aspects related to these people, this chapter introduces disabilities in a historical Brazilian context. In order to overcome such difficulties, Assistive Technology features are presented and several software systems are compared in terms of their functionality. With this, we aimed to present a guide to software as an Assistive Technology resource often used by people with disabilities.

Resumo

As dificuldades de utilização de Tecnologias da Informação e Comunicação por pessoas com deficiência podem ser amenizadas ou superadas por meio recursos de Tecnologia Assistiva. Existem diversos sistemas de software que abordam questões de acessibilidade para a utilização dessas tecnologias, promovendo maior independência e inclusão digital de pessoas com deficiência. Para mostrar as limitações que essas pessoas podem enfrentar no uso de sistemas computacionais, o capítulo conceitua deficiência em um contexto histórico brasileiro. Visando superar tais dificuldades, os recursos de Tecnologia Assistiva são apresentados e diversos sistemas de software são comparados quanto a sua funcionalidade. Com isso, espera-se apresentar um guia de sistemas de software de Tecnologia Assistiva frequentemente utilizados por pessoas com deficiência.

1.1. Introdução

As Tecnologias da Informação e de Comunicação são utilizadas como meio de acesso às informações. Em um cenário ideal, todas as pessoas possuiríam acesso a essas tecnologias, o que proporcionaria mais oportunidades de interação e de aprendizado a todos os seus usuários, de forma igualitária. Contudo, muitas pessoas não possuem acesso a essas tecnologias, seja por aspectos sociais, econômicos, culturais ou outros aspectos mais técnicos de acessibilidade. A distribuição desigual do acesso a essas tecnologias gera a exclusão digital, que pode ser um fator agravante para a exclusão social.

As pessoas com deficiência compõem parte do grupo de pessoas que podem ser excluídas de diversos aspectos da sociedade, por causa da falta de acessibilidade para a utilização de tais tecnologias. Segundo o Censo de 2010, o mais recente divulgado pelo IBGE (Instituto Brasileiro de Geografia e Estatística), a população brasileira era composta por 45,6 milhões de pessoas com deficiência, o que representava 23,9% da população. Similarmente, os idosos também podem ser afetados pela a exclusão digital. Eles correspondem a 20,9% do grupo de pessoas com deficiência [IBGE 2010].

De modo a promover a inclusão social de pessoas com deficiência, podem ser utilizados recursos de Tecnologia Assistiva. Esses recursos variam de uma simples bengala a software e hardware especiais, utilizados com a finalidade de superar / reduzir as limitações funcionais de pessoas com deficiência, por meio da abordagem de aspectos de acessibilidade. Com isso, os recursos de Tecnologia Assistiva visam promover maior independência, qualidade de vida e inclusão de pessoas com deficiência.

A inclusão social deve acontecer nos mais diversos aspectos da sociedade, como no ambiente escolar. Para a educação de pessoas com deficiência, é necessária a adequação física do ambiente e, principalmente, o fornecimento de recursos pedagógicos e de profissionais capacitados. Nesse ambiente, a utilização de recursos de Tecnologia Assistiva é feita com o mesmo intuito, de superar / reduzir as limitações funcionais, não sendo suficientes para ocorrer a educação inclusiva. Portanto, para a educação inclusiva, é necessário que haja métodos diferenciados de ensino, sem que os objetivos educacionais sejam modificados.

Dessa forma, promover a inclusão de pessoas com deficiência na sociedade é um processo que visa oferecer oportunidades iguais de acesso a bens e serviços para essas pessoas, sendo aplicada nos mais diversos ambientes, como no trabalho, no lazer e na educação. Nesses ambientes, é necessário que sejam reconhecidas e respeitadas as diferentes habilidades funcionais das pessoas com deficiência e, em particular, no ambiente escolar é necessário que as necessidades dos alunos com deficiência sejam percebidas e atendidas, a fim de promover a educação inclusiva. Além disso, muitas barreiras físicas são encontradas nesses ambientes, as quais impedem a liberdade e autonomia das pessoas com deficiência. Por esse motivo, os recursos de Tecnologia Assistiva são utilizados. Esses recursos facilitam ou possibilitam a interação da pessoa com deficiência com o ambiente em que se encontra, viabilizando sua autonomia e independência, por meio da abordagem de questões de acessibilidade.

Para mostrar a importância da abordagem de aspectos de acessibilidade, o restante deste capítulo está estruturado da seguinte forma. Na Seção 1.2, são abordados aspectos históricos da pessoa com deficiência. Na Seção 1.3, são apresentadas a definição e a classificação das deficiências em: i) Deficiência Visual; ii) Deficiência Auditiva;

iii) Deficiência Física, iv) Deficiência Cognitiva e v) Deficiência Múltipla. Além disso, são apresentados dados demográficos de pessoas residentes no Brasil por tipo de deficiência. Na Seção 1.4, são apresentados a definição e os objetivos da Tecnologia Assistiva, além de mostrar a classificação dos recursos de Tecnologia. Na Seção 1.5, são apresentados recursos de software para pessoas com deficiência visual, como leitores e ampliadores de tela. Na Seção 1.6, são mostrados recursos de software utilizados por pessoas com deficiência auditiva, como intérpretes em 3D de língua portuguesa para língua de sinais. Na Seção 1.7, são apresentados recursos de software utilizados por pessoas com deficiência física, que facilitam o processo de escrita, como teclados virtuais e preditores de texto. Na Seção 1.8, são apresentados recursos de software para pessoas com deficiência cognitiva, que visam facilitar a leitura e a compreensão do conteúdo disponibilizado. Na Seção 1.9, é abordado o tema educação mediada pela tecnologia. A Seção 1.10 apresenta as considerações finais do capítulo.

1.2. Aspectos Históricos da Pessoa com Deficiência

As pessoas com deficiência nem sempre foram vistas ou referidas como iguais às pessoas sem deficiência pela sociedade. Em diferentes épocas, foram cunhados termos relacionados à deficiência que expressam a forma de tratamento e o valor atribuído a essas pessoas. Os próximos parágrafos relatam os paradigmas excludentes de pessoas com deficiência e a luta dessas pessoas para romper com tais paradigmas. A compreensão das formas de percepção sobre pessoas com deficiência é importante para o posicionamento de desenvolvedores e de profissionais na área de software para o entendimento das perspectivas de independência e de autonomia vigentes na visão atual sobre pessoas com deficiência.

No decorrer da história antiga e medieval, são percebidas duas vertentes relacionadas ao tratamento de pessoas com deficiência: a rejeição ou a proteção social [Ozawa e Amaral 2017]. As sociedades escravistas grega e romana foram marcadas por guerras destinadas à conquista de escravos e à conservação da ordem vigente. Por esse motivo, atributos como força física, beleza e corpo perfeito eram supervalorizados, de modo que as pessoas com deficiência não possuíam valor para a sociedade e eram simplesmente descartadas.

Contrariamente, o modelo de institucionalização introduzido na Idade Média concentrava-se na proteção social de pessoas com deficiência. As instituições (hospitais, hospícios e asilos), geralmente mantidas pela Igreja Católica, recebiam pessoas idosas, doentes e com limitações funcionais. Nessas instituições, a pessoa com deficiência era mantida e cuidada distante da sociedade [PEE 2006]. Tal modelo somente passou a ser contestado após os primeiros avanços científicos verificados na Sociedade Moderna. Com os avanços ocorridos, surgiu a primeira forma de compreensão da pessoa com deficiência, que consistia na crença do funcionamento dos órgãos em pares [PDE 2016] denominada concepção biológica ingênua por Vigotski. Em outras palavras, a pessoa com deficiência apresentaria um “defeito” em um órgão, para o qual, um segundo órgão, sem defeito, supre sua função. Essa filosofia é exemplificada pelo caso da crença que uma pessoa cega possui a audição melhor que uma pessoa sem deficiência para compensar a cegueira.

Apesar de inverídica, tal teoria permitiu avanço na forma como as pessoas com deficiência eram vistas. Durante muitos séculos, a visão de que as pessoas com deficiência não eram humanas e nem cidadãos prevaleceu. Porém, a deficiência foi colocada no

âmbito da ciência, por meio da qual surgiram os primeiros pressupostos para a educação de pessoas com deficiência [PEE 2006].

A princípio, a educação restringia-se à nobreza e à burguesia enriquecida. As primeiras instituições para a educação de surdos e cegos surgiram na França e, posteriormente, em diversas outras, em diferentes países [PEE 2006]. Nesse período, o principal problema para a inserção da pessoa com deficiência na sociedade era considerada sua limitação física ou sensorial. Por esse motivo, as instituições focavam em modificar as pessoas com deficiência para que pudessem ser inseridas na sociedade. Como exemplo, tem-se a forma como as pessoas com deficiência auditiva eram educadas. Em uma escola específica para surdos, as crianças aprendiam a reproduzir sons e fazer leitura labial, sendo reprimidas suas tentativas de comunicação por sinais [Beltrami e Moura 2015].

Esse quadro se estendeu durante o início do Século XX. Durante esse período, a sociedade utilizava o termo "os inválidos" para fazer referência às pessoas com deficiência [Sassaki 2003]. Tal termo significava que as pessoas com deficiência não possuíam valor para a sociedade, sendo consideradas um fardo para a família. Após a Segunda Guerra Mundial, houve mais avanços na forma como as pessoas com deficiência eram tratadas e vistas pela sociedade. Com o advento da Declaração Universal dos Direitos Humanos, foram consagrados um conjunto de direitos humanos a todo e qualquer cidadão, o que levou ao questionamento da forma de tratamento de pessoas com deficiência [Damasceno 2014].

Nessa época, termos como "os incapacitados" e "os defeituosos" ainda eram utilizados [Sassaki 2003]. No entanto, o primeiro termo passava a reconhecer a capacidade residual da pessoa com deficiência, enquanto o segundo termo focava apenas na deficiência e não na capacidade que a pessoa com deficiência possuía para realizar uma tarefa. Em paralelo, questões como o modelo de institucionalização começaram a ser criticadas como uma prática que violava os direitos humanos.

Como resultado da luta constante desse grupo de pessoas, a ONU (Organização das Nações Unidas) especificou o termo "pessoas deficientes", o que igualou as pessoas com deficiência em direito e dignidade às pessoas sem deficiência [Sassaki 2003]. Tal termo foi evoluído para "pessoas com deficiência", definido no art. 1º da Convenção Internacional dos Direitos da Pessoa com Deficiência [CDPD 2008] da forma:

> "*Pessoas com deficiência são aquelas que têm impedimentos de natureza física, mental, intelectual ou sensorial, os quais, em interação com diversas barreiras, podem obstruir sua participação plena e efetiva na sociedade em igualdades de condições com as demais pessoas*". (p. 27).

Diferente dos termos anteriores, este último considera que a deficiência não está na pessoa, mas na relação entre a pessoa e o meio. Isso significa que, para propiciar às pessoas com deficiência a plena participação na sociedade, devem ser abordadas questões de acessibilidade de modo a apoiar a relação entre a pessoa e o meio.

Essa nova definição impactou em todos os aspectos relacionados a forma de tratamento e o valor atribuído às pessoas com deficiência. A partir dessa definição, houve o empoderamento das pessoas com deficiência, visto que elas são responsáveis pelas suas

escolhas, além de contribuir de forma a mudar a sociedade rumo à inclusão de todas as pessoas com ou sem deficiência.

1.3. Conceitos e Tipos de Deficiência

O conceito relativo à deficiência tem evoluído de uma perspectiva individual e médica para uma perspectiva estrutural e social. Diante dessa nova perspectiva, diferentes conceitos podem ser encontrados para deficiência (Tabela 1.1), os quais destacam a interação da pessoa com deficiência e o meio em que está inserida, ao invés de focar em suas limitações.

Tabela 1.1: Conceitos de Pessoas com Deficiência

Órgão	Classificação
Classificação Internacional de Funcionalidade, incapacidade e saúde - CIF	*"Deficiências são problemas na função ou estrutura corporal, tais como um desvio ou perda significativos. Incapacidade é um termo abrangente para deficiências, limitações de atividade e restrições de participação. Ela denota os aspectos negativos da interação entre um indivíduo (com uma condição de saúde) e os fatores contextuais daquele indivíduo (fatores ambientais e pessoais)."* [CIF 2004].
Estatuto da Pessoa com deficiência Lei 13.146/ 20015	*"Considera-se pessoa com deficiência aquela que tem impedimento de longo prazo de natureza física, mental, intelectual ou sensorial, o qual, em interação com uma ou mais barreiras, pode obstruir sua participação plena e efetiva na sociedade em igualdade de condições com as demais pessoas."* [Brasil 2015].
Convenção dos Direitos da Pessoa com Deficiência - ONU	*"Pessoas com deficiência são aquelas que têm impedimentos de natureza física, mental, intelectual ou sensorial, os quais, em interações com diversas barreiras, podem obstruir sua participação plena e efetiva na sociedade com as demais pessoas."* [CDPD 2008].

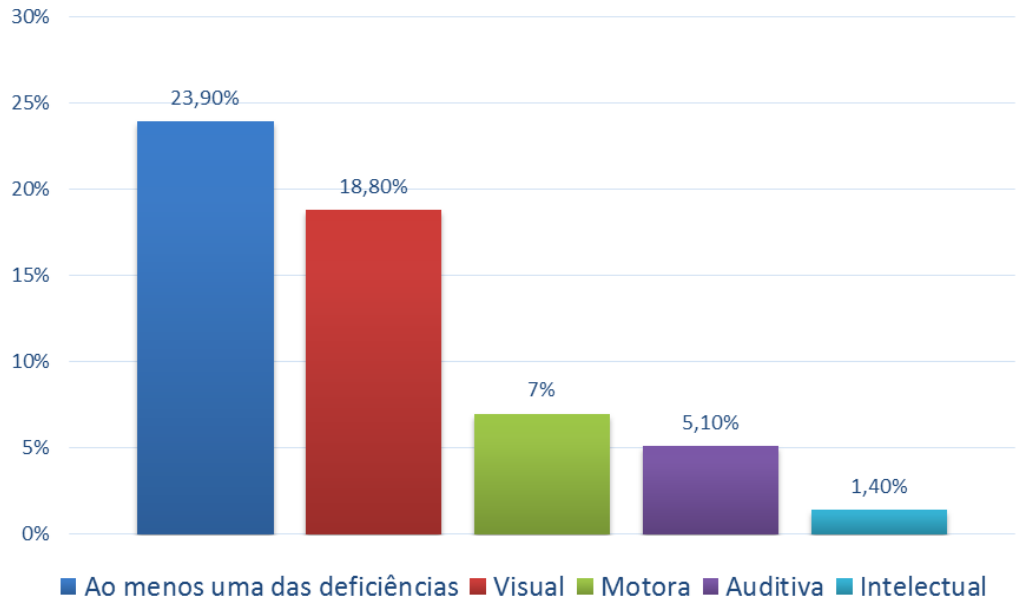
Quanto a natureza, as deficiências podem ser agrupadas em [PEE 2006]:

- **Deficiência Visual.** Caracteriza-se pela perda de parte da percepção visual (baixa visão) ou pela ausência total de resposta visual (cegueira). A acuidade visual de pessoas com baixa visão varia muito, e possui diferentes tipos de efeitos na visão de uma pessoa, como perda da visão central (degeneração macular) ou periférica (glaucoma), maior opacidade (catarata), dificuldade em distinguir cores (daltonismo) e vários outros;
- **Deficiência Auditiva.** Caracteriza-se pela perda parcial ou total da capacidade auditiva. Em relação a deficiência auditiva, é comum ocorrer o equívoco de que toda pessoa surda é muda. Em geral, a pessoa que não ouve pode não desenvolver a fala, pois não aprendeu como emitir tais sons. Contudo, existem pessoas surdas que conseguem sonorizar palavras e compreendem palavras por meio da leitura labial. Note que o mais comum é a utilização de línguas de sinais para estabelecer comunicação com pessoas com deficiência auditiva;

- **Deficiência Cognitiva.** Caracteriza-se pela limitação intelectual. Algumas das principais categorias de deficiências cognitivas incluem *deficit* ou dificuldades com memória, solução de problemas, atenção, leitura, compreensão linguística e verbal, compreensão matemática e compreensão visual;
- **Deficiência Física.** Caracteriza-se pela alteração completa ou parcial de uma ou mais partes do corpo. Essa alteração pode ser caracterizada pela (i) paralisia parcial de um membro do corpo (monoparesia), metade do corpo (hemiparesia), dos membros inferiores (paraparesia) e paralisia dos membros inferiores e superiores (tetraparesia), (ii) paralisia total um membro do corpo (monoplegia), paralisia da metade do corpo (hemiplegia), membros inferiores (paraplegia), de três membros (triplegia) e paralisia dos membros inferiores e superiores do corpo (tetraplegia), (iii) amputação ou ausência de membros; e (iv) paralisia cerebral;
- **Deficiência Múltipla.** Indica que uma pessoa possui duas ou mais deficiências associadas. Esse tipo de deficiência pode ocorrer como deficiência física / cognitiva, deficiência visual/auditiva, deficiência visual / física, deficiência cognitiva / auditiva, deficiência cognitiva / visual e deficiência física / auditiva.

As pessoas com deficiência compõem uma parcela da população que pode ser excluída de vários aspectos da sociedade por causa da falta de acessibilidade. Nos dados dos últimos censos, pode-se perceber que elas correspondem a um percentual importante da população brasileira. Segundo o Censo de 2010, o mais recente divulgado pelo IBGE, a população brasileira era composta por 45,6 milhões de pessoas com deficiência, o que representava 23,9% da população [IBGE 2010]. Como mostrado no Gráfico 1.1, esse número indica a quantidade de pessoas que declararam ter uma ou mais deficiências (23,9%), que podem ser classificadas em deficiência intelectual (1,4%), motora (7%), auditiva (5,1%) e visual (18,8%).

Gráfico 1.1: Distribuição percentual da população brasileira por tipo de deficiência. Adaptado de IBGE (2010).



Ainda, conforme esse censo, a população brasileira era composta por 14,1 milhões de pessoas com idade igual ou superior a 65 anos, o que representava 7,4% da população. Na parcela de 7,4% da população composta por idosos, têm-se que 9,5 milhões de pessoas apresentam redução das capacidades funcionais [IBGE 2010], particularmente em termos visuais e auditivos [MS 2006]. Isso significa que 67,7% da população idosa possui algum tipo de deficiência. Assim, na parcela de 23,9% da população que possui deficiência, há 20,9% de pessoas idosas. Em questões de acessibilidade, esses números aumentam quando as pessoas que vivem direta ou indiretamente com idosos e / ou pessoas com deficiência são consideradas [Rodrigues e Alves 2013]. Eles também deparam-se com situações limitantes.

1.4. Tecnologia Assistiva

Conforme o Comitê de Ajudas Técnicas [CAT 2009], a Tecnologia Assistiva é uma área de conhecimento que visa maximizar as habilidades funcionais de pessoas com deficiência, por meio da utilização de produtos, recursos, estratégias e serviços que atendem às necessidades específicas dessas pessoas.

Dessa forma, a Tecnologia Assistiva proporciona à pessoa com deficiência mais independência, qualidade de vida e inclusão social, por meio da ampliação de sua comunicação, mobilidade, controle de seu ambiente, habilidades de seu aprendizado, trabalho e integração com a família, amigos e sociedade. Diferentes tipos de recursos de Tecnologia Assistiva são apresentados na Figura 1.1, tais como recursos para leitura, visão, audição e comunicação, recursos para mapeamento mental e *brainstorming*, recursos para potencializar as habilidades e auxiliar no estudo, recursos para gerenciamento de tempo, organização e gerenciamento de tarefas, recursos para reconhecimento de fala, conversor de fala para texto, escrita e ferramentas de pesquisa.

Ainda, quando recursos de Tecnologia Assistiva são utilizados por pessoas com e sem deficiência, a fim de realizar uma mesma tarefa, isso descaracteriza esse recurso como Tecnologia Assistiva [Rodrigues e Alves 2013]. Em outras palavras, esses recursos devem atender especificamente as pessoas com deficiência, por meio da abordagem de questões de acessibilidade.

Como existem diversos grupos de usuários com diferentes capacidades funcionais, os recursos de Tecnologia Assistiva devem satisfazer às necessidades específicas de cada grupo. Esses recursos são classificados em [Bersch 2008]:

- **Auxílio para Vida Diária e Prática**. Promove independência durante a realização de tarefas básicas diárias. Por exemplo, podem ser utilizados talheres especiais para auxiliar no momento da alimentação, roupas com velcros para facilitar sua vestimenta, barras de apoio para facilitar a locomoção;
- **Comunicação Alternativa e Aumentativa.** Para pessoas sem fala e / ou escrita ou com lacuna de comunicação entre suas necessidades e sua capacidade de falar e / ou escrever, podem ser utilizadas placas de comunicação e vocalizadores;
- **Recursos de Acessibilidade para o Computador**. Hardware / software para facilitar o uso do computador por pessoas com limitações físicas e sensoriais.

Por exemplo, podem ser utilizados teclados adaptados, software de reconhecimento de voz e impressoras Braille;

- **Sistemas de Controle de Ambientes.** Ao fazer o uso de um controle remoto, as pessoas com deficiência podem ligar, desligar e ajustar dispositivos elétricos e eletrônicos. Por exemplo, luz, som e abertura e fechamento de portas e janelas;
- **Projetos Arquitetônicos para Acessibilidade.** Assegura o acesso, a funcionalidade e a mobilidade a todas as pessoas, independentemente da sua condição física e sensorial, facilitando a locomoção pelo meio ambiente. Por exemplo, rampas, elevadores e banheiros;
- **Órteses e Próteses.** As peças artificiais (prótese) substituem as partes em falta (peças humanas) ou são colocadas ao lado de um segmento do corpo (chaves), garantindo melhor posicionamento, estabilização e funcionalidade;
- **Adaptação Postural.** Recursos que promovem adaptações na postura (deitada, sentada e em pé), por exemplo, travesseiros na cama ou estabilizadores ortostáticos;
- **Auxílio de Mobilidade.** Equipamentos ou estratégias utilizadas na melhoria da mobilidade pessoal. Por exemplo, bengalas, muletas, cadeiras de rodas e *scooters*;
- **Ajuda para Pessoas Cegas ou de Baixa visão.** Favorece o desempenho autônomo e independente de pessoas com deficiência visual. Por exemplo, lentes, interfaces de toque, impressoras Braille, software de reconhecimento de voz, ampliadores e leitores de tela;
- **Auxílio para Perda Auditiva ou Pessoas Surdas.** Favorece o desempenho autônomo e independente de pessoas com deficiência auditiva. Por exemplo, aparelhos auditivos, telefones com teclado de teletipo (TTY - *Teletypewriter*) e sistemas de alerta visual de toque;
- **Adaptadores de Veículos.** Adaptações de veículos a motor a serem utilizados para o transporte de pessoas com deficiência. Por exemplo, facilitadores de embarque e desembarque;
- **Esporte e Recreação.** Recursos utilizados para a prática de esportes e de atividades de lazer.

É importante destacar que o agrupamento dos recursos de Tecnologia Assistiva por tipo de deficiência não são mutuamente exclusivos. Por exemplo, um recurso utilizado por pessoas com deficiência visual também pode ser utilizado por pessoas com limitações físicas e sensoriais, como a impressora Braille e softwares de reconhecimento de voz (recursos descritos nos tópicos: Ajuda para Pessoas Cegas ou de Baixa visão e Recursos de Acessibilidade para o Computador).

Sistemas de software, como recurso de Tecnologia Assistiva, permitem ou facilitam a interação entre as pessoas com deficiência e o computador. Por exemplo, as pessoas com deficiência visual podem utilizar os leitores e / ou os ampliadores de tela, recursos de software que tornam o conteúdo disponibilizado na tela do computador acessível para pessoas com deficiência visual. Outros exemplos são as placas de

comunicação e os vocalizadores para a comunicação aumentativa e alternativa, recursos de software utilizados por pessoas com deficiência na fala. Esses recursos permitem também a utilização de tecnologias da informação e comunicação por pessoas com deficiência, de modo a incluí-las na sociedade atual em que vive, marcada pelo acesso constante a diversas informações, o que cria possibilidades comunicativas, cognitivas, sociais e culturais.

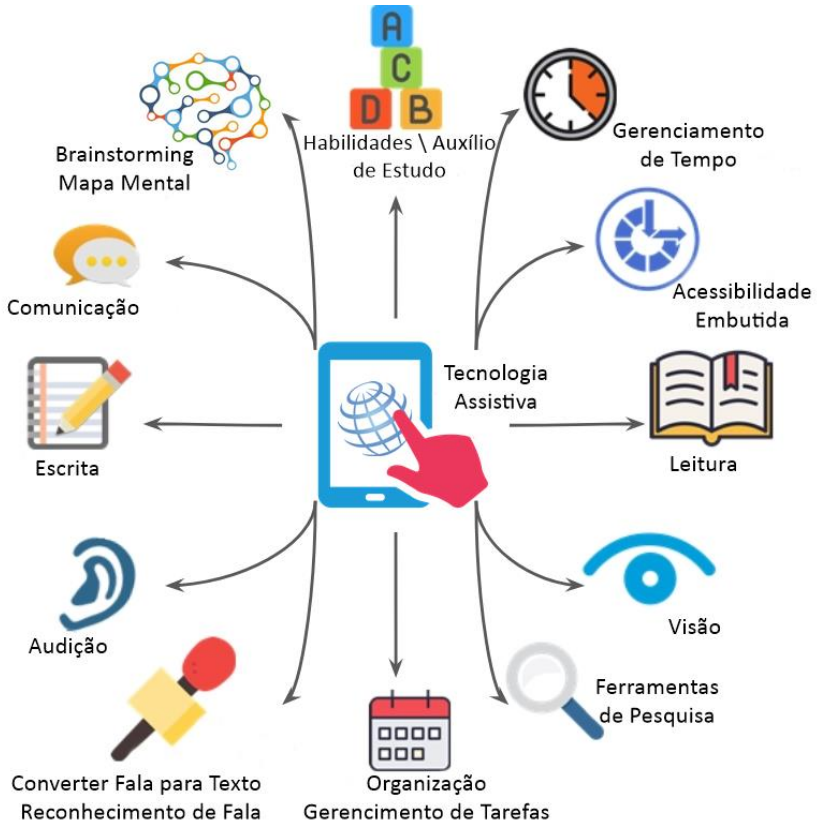


Figura 1.1: Exemplos de recursos de Tecnologia Assistiva. Adaptado de Augsburg University (2018).

1.5. Recursos de Software para Pessoas com Deficiência Visual

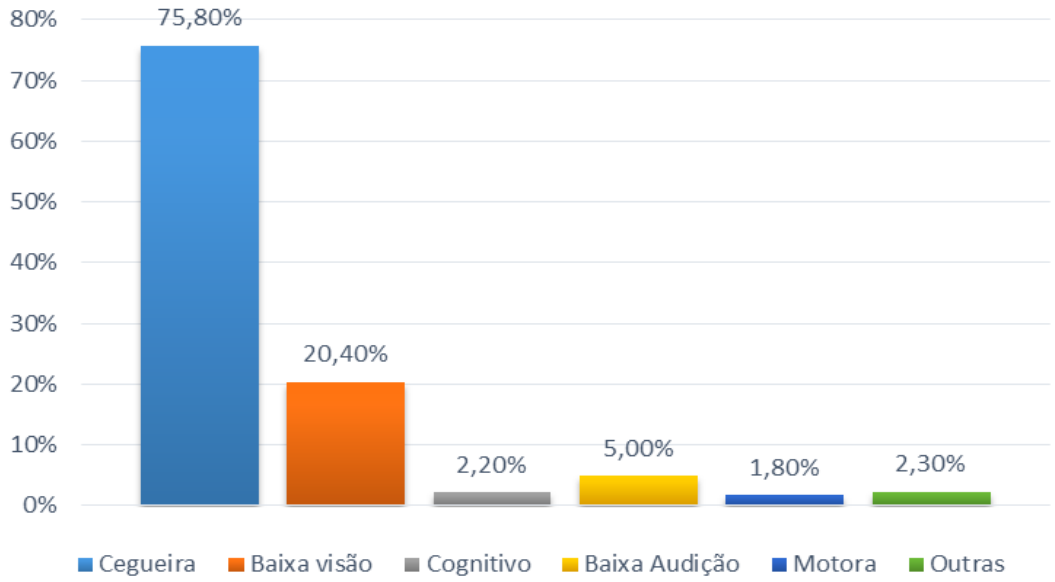
Dentro do grupo de pessoas com deficiência visual, existem diferentes necessidades a serem atendidas. Por exemplo, as pessoas com baixa visão podem utilizar recursos como os ampliadores de tela, combinados (ou não) com leitores de tela, enquanto que os ampliadores não teriam utilidade para pessoas cegas sem visão residual. Diversos recursos são utilizados por pessoas com baixa visão, mas não por pessoas cegas, como as opções de tamanhos e fontes de letras, espaçamento entre linhas, parágrafos e caracteres e alto contraste.

Dessa forma, para a diversidade de necessidades entre o grupo de pessoas com deficiência visual (baixa visão, cegueira, daltonismo e outras), existem diversos tipos de software que fornecem funções específicas para os grupos de pessoas com deficiência visual. Para explorar os recursos de Tecnologia Assistiva voltados para a deficiência visual, no restante da seção, são apresentados recursos de software mais utilizados, separados nas categorias (i) leitores de telas e (ii) ampliadores de telas.

1.5.1. Leitores de Telas

Os leitores de tela, usualmente utilizados por pessoas com deficiência visual, são recursos de software que acessam os conteúdos disponibilizados por meio de tecnologias como computadores, celulares e televisores. Conforme a pesquisa mais recente realizada pela *Web Accessibility In Mind* - WebAIM (2017), dentre as pessoas que utilizam leitores de tela, 95,4% das pessoas têm deficiência visual (75% são cegas, 20,4% têm baixa visão), 2,2% têm deficiência cognitiva, 5% deficiência auditiva, 1,8% deficiência motora e 2,3% têm outros tipos de deficiência (Gráfico 1.2). Como abordado na Seção 1.3, a deficiência múltipla é o resultado da combinação de uma ou mais deficiências que atingem uma pessoa. Dessa forma, pessoas com deficiência visual (cegueira ou baixa visão) podem ter deficiência física, auditiva ou cognitiva.

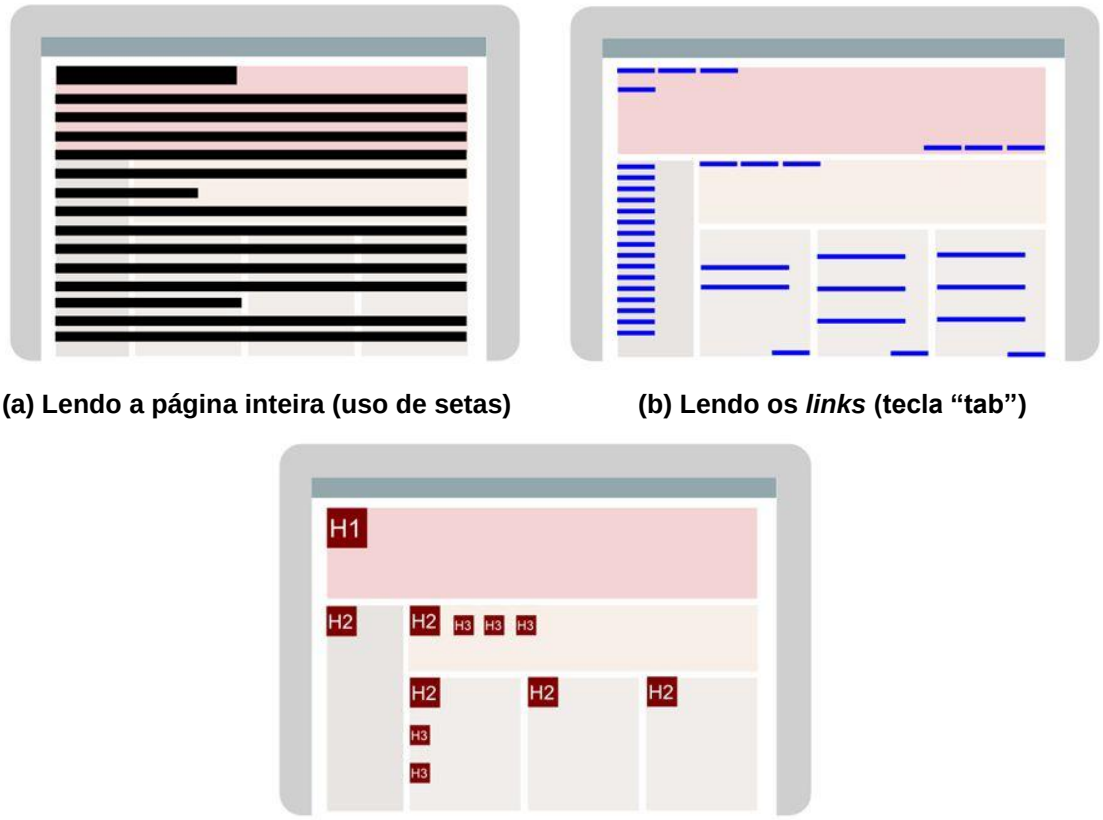
Gráfico 1.2: Utilização de Leitores de Telas por tipo de deficiência. Adaptado de WebAIM (2017).



Os leitores de tela fornecem ao usuário um sintetizador de voz que "lê" o conteúdo mostrado na tela. Existem três maneiras de navegar em um sítio Web ou sistema ao utilizar leitores de telas [EMAG 2014]: i) ler a página completa, uma linha por vez, ao mover as setas do teclado (Figura 1.2a); ii) navegar pelos links usando a tecla "tab" (Figura 1.2b); ou iii) navegar pelos cabeçalhos usando a tecla "h" (Figura 1.2c) ou outra equivalente disponibilizada pelo software.

Ainda, na pesquisa da WebAIM (2017) foram levantados os principais leitores de tela utilizados (Gráfico 1.3), na qual destacam-se o JAWS[1] (*Job Access With Speech*), o NVDA[2] (*NonVisual Desktop Access*) e o Voice Over[3]. O JAWS é o principal leitor de telas utilizado por 46,6% das pessoas que fazem o uso desse tipo de recurso. Em seguida, os leitores NVDA e Voice Over, preferidos por 31,9% e 11,7% das pessoas que utilizam tal recurso, respectivamente.

[1] Disponível em: <http://www.freedomscientific.com/Products/Blindness/JAWS>
[2] Disponível em: <https://www.nvaccess.org/>
[3] Disponível em: <https://www.apple.com/br/accessibility/iphone/vision/>

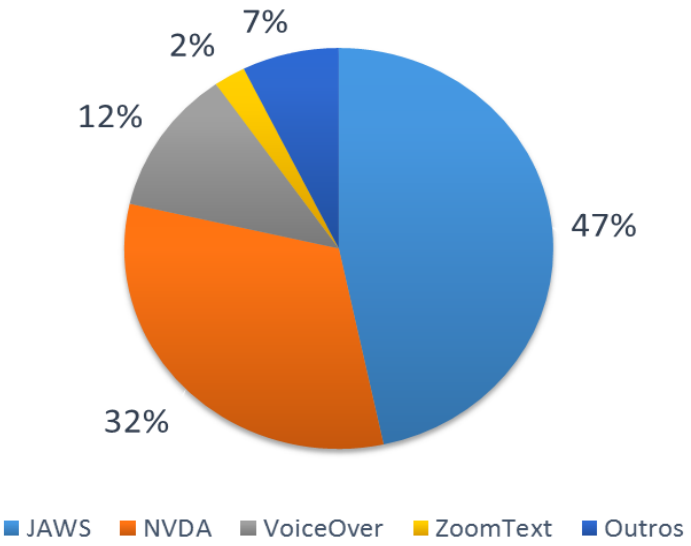


(a) Lendo a página inteira (uso de setas) **(b) Lendo os *links* (tecla "tab")**

(c) Lendo os cabeçalhos (tecla "h")

Figura 1.2: Navegação em Portal / Sistema Utilizando Leitores de Telas. Extraído de eMAG (2014).

Gráfico 1.3: Principais Leitores de Telas (WEBAIM, 2017).



O leitor de telas JAWS é um software proprietário para computadores que executam o sistema operacional Windows. Desenvolvido pela Freedom Scientific, o software abrange diversas funções, como OCR (*Optical Character Recognition*, é uma

tecnologia capaz de reconhecer caracteres a partir de um arquivo de imagem) para arquivos de imagem e documentos PDFs inacessíveis, suporte a câmera para acesso direto a documentos ou livros impressos, sintetizadores multilíngues, conjunto de livros em DAISY (*Digital Accessible Information System*, é o padrão mundial para livros em áudio) e suporte a Braille. Apesar de ser o leitor mais utilizado e fornece um conjunto diverso de funções, o software apresenta alto custo de aquisição, o que muitas vezes impossibilita sua compra e sua utilização por pessoas com deficiência visual.

Visando superar a limitação referente ao alto custo de aquisição, Michael Curran e James Teh deram início ao desenvolvimento do software leitor de telas livre NVDA. O software disponibilizado para computadores que executam o sistema operacional Windows tem conquistado grande número de usuários e, especialmente, usuários que colaboram no processo de desenvolvimento contínuo do software. Em relação às funções, o software fornece o sintetizador de voz em 43 idiomas e suporte a Braille.

Diferente dos leitores anteriores, Voice Over é um leitor de telas nativo do sistema operacional iOS. Além de ser um leitor de telas, o software abrange outras funções voltadas para pessoas com baixa visão, como um ampliador de telas, opções de contraste e tamanho do cursor.

1.5.2. Ampliadores de Telas

Os ampliadores de telas, usualmente utilizados por pessoas com baixa visão, são recursos de software que ampliam a área selecionada na tela. Normalmente, outros recursos são combinados com os ampliadores de telas, de modo a permitir que as pessoas com baixa visão percorram o texto disponibilizado com maior clareza. Os recursos utilizados com mais frequência com os ampliadores de telas são os recursos de dimensionamento de texto, alto contraste e personalização de cores.

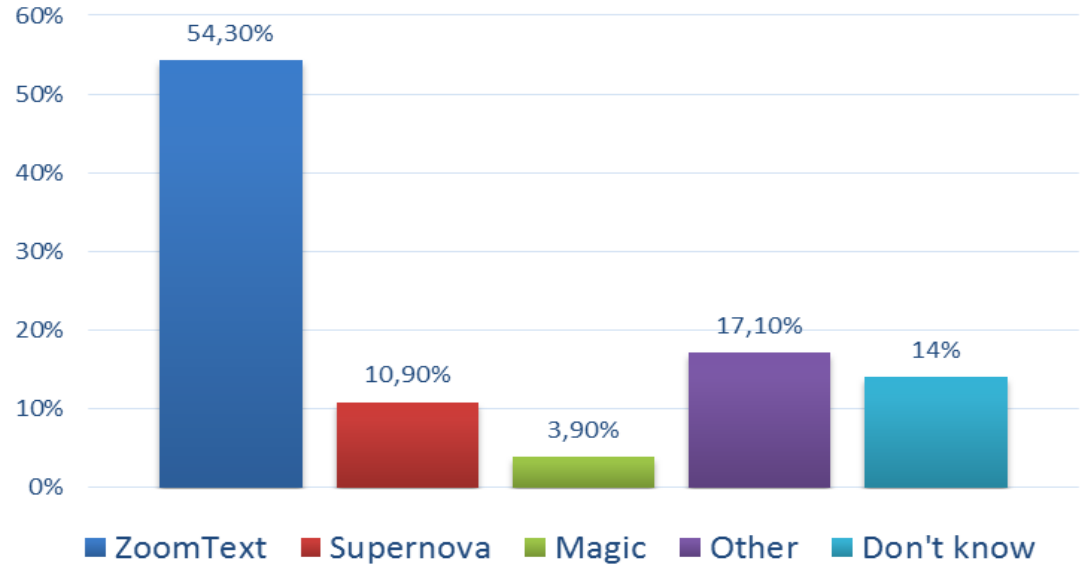
Uma pesquisa realizada no Reino Unido [Moore 2016] destaca o ZoomText[4] como principal ampliador de telas, correspondendo a 54,3% dos usuários de tal recurso. Outros sistemas de software como o Supernova[5] e o MAGIc[6] ganharam destaque, sendo utilizados respectivamente, por 17,1% e 3,9% dos usuários (Gráfico 1.4). Similarmente, na pesquisa realizada pela WebAIM (2013), são descritos os sistemas de software ZoomText e MAGic como os ampliadores mais utilizados.

O ZoomText é um ampliador e leitor de telas desenvolvido pela *AI Squared*. Esse software é proprietário, disponibilizado para dispositivos que executam o sistema operacional Windows. Nesse software, há suporte para *touch screen*, ampliação até 64x, inversão de cores, ajuste de contraste e brilho, acesso a tela de *logon* do Windows, aprimoramentos de cursor, compatível com CITRIX e suporte para dois monitores.

O ampliador de telas MAGic, desenvolvido pela *Freedom Scientific*, é disponibilizado para computadores que executam o sistema operacional Windows. É um software proprietário que fornece funções de ampliação de tela que pode (ou não) ser comprada juntamente com o leitor de telas, que se beneficia com o desenvolvimento do JAWS. O MAGIc permite a ampliação da tela em até 60x, fornece opções de aprimoramentos de cursor, inversão de cores, ajuste de brilho e contraste, compatível com CITRIX e suporte para dois monitores.

[4] Disponível em: <http://www.zoomtext.com/products/zoomtext-magnifierreader/>
[5] Disponível em: < https://yourdolphin.com/supernova-magnifier>
[6] Disponível em: < http://www.freedomscientific.com/Products/LowVision/MAGic>

Gráfico 1.4: Principais Ampliadores de Telas no Reino Unido. Adaptado de Moore (2016)



O SuperNova é um software proprietário desenvolvido pela Dolphin em quatro edições. No geral, o software oferece as funcionalidades de ampliação da tela em até 64x, leitura de tela, suporte para *touch screen* e para Braille, ajuste de contraste e brilho, inversão de cores, acesso a tela de *logon* do Windows, e suporta dois ou mais monitores.

1.6. Recursos de Software para Pessoas com Deficiência Auditiva

Em software, como recurso de Tecnologia Assistiva para pessoas com deficiência auditiva, há legendas em conteúdos que possuem áudio. Normalmente, as legendas em vídeos na internet são disponibilizados em língua escrita, como na língua portuguesa. Contudo, a língua portuguesa não é a primeira língua dos surdos, o que pode levar a dificuldades de compreensão do texto. Com isso, soluções como o HandTalk[7] e o ProDeaf[8] recebem destaque por traduzir o conteúdo para a língua de sinais, como a LIBRAS (Língua Brasileira de Sinais).

O ProDeaf Móvel[9], desenvolvido pela ProDeaf, é um aplicativo disponibilizado de forma gratuita para dispositivos Android, iOS e Windows Phone. O aplicativo possui um dicionário com palavras e expressões em Português, que, ao serem selecionadas, são traduzidas pelo intérprete 3D. Ainda, o aplicativo é capaz de transformar texto e áudio em língua de sinais. Além do ProDeaf Móvel, a empresa disponibiliza o ProDeaf WebLibras[10], um plug-in proprietário, que realiza a tradução de sites em língua portuguesa para a língua de sinais, e o ProDeaf Web[11], uma página em que é disponibilizado, de forma gratuita, um dicionário Português-LIBRAS, um intérprete 3D e a opção de criar novos sinais.

O HandTalk, um aplicativo brasileiro, ganhador do prêmio WSA-Mobile (*World Summit Award Mobile*), é disponibilizado gratuitamente para dispositivos Android, iOS e Windows Phone, capaz de transformar áudio, imagens e textos em língua de sinais.

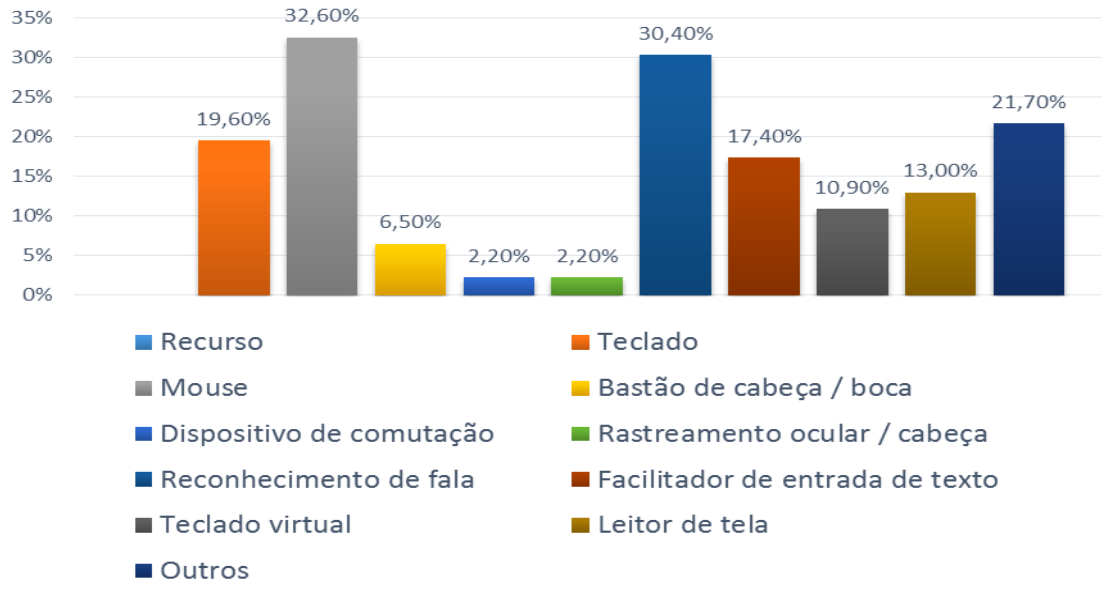
[7] Disponível em: <https://www.handtalk.me/app>
[8] Disponível em: <http://www.prodeaf.net/>
[9] Disponível em: <http://www.prodeaf.net/pt-br/prodeaf-movel>
[10] Disponível em: <http://www.weblibras.com.br/>
[11] Disponível em: < http://web.prodeaf.net/>

1.7. Recursos de Software para Pessoa com Deficiência Física

Por causa de diversos tipos de deficiência física, existem diferentes recursos de Tecnologia Assistiva, voltados para as limitações que podem ser enfrentadas por pessoas que têm essa deficiência. Ao utilizar o computador, as pessoas com deficiência física podem ter limitações referentes ao uso do *mouse* ou do teclado, em que componentes pequenos presentes na tela podem ser de difícil acesso.

Desse modo, os recursos para pessoas com deficiência física dividem-se em *mouse* e teclados especiais, utilizados por 32,6% e 19,6% de pessoas com deficiência física, respectivamente. Além disso, são utilizados bastões na cabeça / boca e botões adaptados, por 6,5% e 2,2% de pessoas com deficiência física, respectivamente. Como recursos de software, são utilizados, Rastreamento ocular / cabeça por 2,2% de pessoas com deficiência física, Reconhecimento de fala por 30,4%, Preditor de texto por 17,4%, teclado virtual por 10,9%, leitores de tela por 13%, e 21,7% de pessoas com deficiência física utilizam outros recursos (Gráfico 1.5) [WebAIM 2013].

Gráfico 1.5: Recursos utilizados por pessoas com deficiência física. Adaptado de WebAIM (2013)



Desse modo, foram selecionados alguns dos recursos mais populares, conforme sua classificação por estrelas, disponibilizados gratuitamente no ATHENA - *Free AT Software Inventory* [ATHENA 2018]. Dentre os recursos de software de Tecnologia Assistiva para pessoas com deficiência física, destacam-se, Click-N-TypePortable[12] e o LetMeType[13].

O Click-N-TypePortable é um teclado virtual que não precisa ser instalado para ser utilizado em computadores que executam o sistema operacional Windows. Além disso, o software disponibiliza os recursos de previsão de palavras, conclusão de palavras, modos de verificação, criação de layouts e vários idiomas.

O aplicativo LetMeType, disponibilizado para computadores com sistema operacional Windows, amplia o conceito de *autocomplete* no nível do sistema

[12] Disponível em: <http://access.uoa.gr/ATHENA/eng/applications/view/12>
[13] Disponível em: <http://access.uoa.gr/ATHENA/eng/applications/view/255>

operacional. O aplicativo monitora o que está sendo digitado em qualquer caixa de entrada e cria um banco de dados de palavras frequentes para, posteriormente, exibir uma lista das palavras mais prováveis que começam com as letras digitadas.

1.8. Recursos de Software para Pessoa com Deficiência Cognitiva

Normalmente, os subgrupos como as pessoas com dificuldades de aprendizagem (e.g. dislexia e disgrafia), transtornos de atenção (e.g. TDAH - Transtorno do Déficit de Atenção com Hiperatividade), e outros, possuem similaridade nos desafios enfrentados por pessoas com deficiência cognitiva. Por esse motivo, os subgrupos são colocados na categoria deficiência cognitiva. As dificuldades enfrentadas estão relacionadas à percepção e ao processamento das informações disponibilizadas, à capacidade de memória, à solução de problemas e à atenção [WebAIM 2013]. Alguns cuidados que facilitam o entendimento do conteúdo disponibilizado à pessoa com deficiência intelectual incluem utilização da linguagem mais simples, possibilidade de configurar o formato do texto, como tamanho e família da fonte, cor de fundo e fonte, estilo de texto como negrito, itálico e sublinhado, e diversos outros. Visando a esses cuidados, exemplos de software voltados para pessoas com deficiência cognitiva são WebHelpDislexia[14] e o ATbar[15].

O WebHelpDyslexia, desenvolvido por alunos da Universidade Federal de Lavras (UFLA), é uma extensão do Google Chrome™ que visa facilitar a leitura de páginas Web por pessoas com dislexia [Avelar et al. 2015]. Note que as dificuldades encontradas por pessoas com dislexia assemelham-se às dificuldades encontradas por pessoas com síndrome de Irlen e baixa visão [Rossini 2014]. A ferramenta fornece um conjunto de funções que permitem mudanças em páginas Web, como diferentes famílias de fonte, tamanho, alinhamento e cor do texto, remoção de itálico, sublinhado e negrito, ajustes de espaçamento entre linhas, caracteres e parágrafos, mudança na cor de fundo da página e destaque de trechos do texto, além de uma ferramenta que funciona como “régua de leitura” que destaca partes do texto de modo a auxiliar a concentração, e um dicionário de sinônimos para auxiliar no entendimento do texto. Na Figura 1.3, é apresentada a página web antes de alterar tamanho e família da fonte, cor de fundo e régua de leitura e, na Figura 1.4, é representada a página Web com o novo estilo. Dessa forma, ao fornecer esse conjunto de funções, WebHelpDyslexia tem se mostrado útil, não apenas para pessoas com dislexia, mas para pessoas com as mais diversas limitações visuais

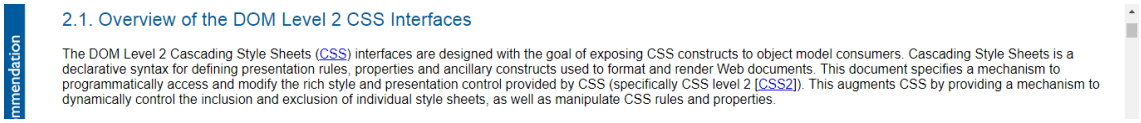


Figura 1.3: Antes de aplicar configurações com o WebHelpDyslexia.

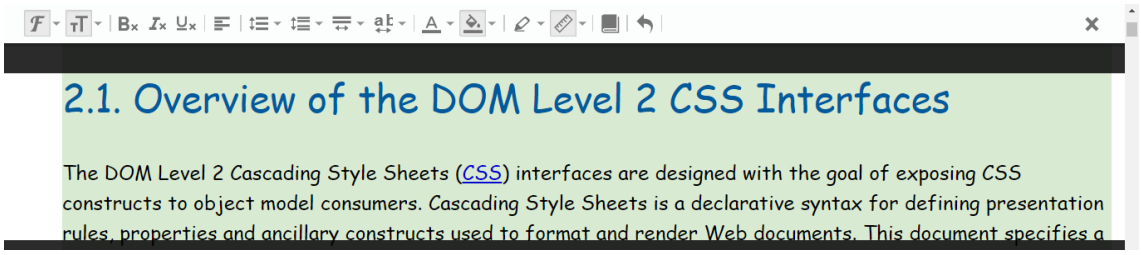


Figura 1.4: Antes de aplicar configurações com o WebHelpDyslexia.

[14] Disponível em: <https://chrome.google.com/webstore/detail/webhelp/pjnhjelpkdoihfjeeemmahpdbmgliboo>
[15] Disponível em: <https://chrome.google.com/webstore/detail/atbar/lihjlachbdicbhpalgegcgknkbmjhicl>

O ATbar é uma barra de ferramentas gratuita, que suporta múltiplos navegadores. Ela é utilizada para ajudar usuários a customizar o modo que eles veem e interagem com páginas da web. Para isso, são fornecidas funções para aumentar e diminuir o tamanho das fontes, leitura de texto em voz alta, cor do texto e um dicionário que auxilie a leitura. Além disso, na ATbar é verificada a ortografia de formulários e há tentativa para prever palavras quando o usuário está escrevendo.

1.9. Educação Mediada pela Tecnologia Assistiva

A educação é um processo contínuo de integração do ser humano na sociedade, visando a sua formação integral. Conforme estabelecido na Declaração dos Direitos Humanos [Brasil 1998], no Art. 26º, a educação é um direito fundamental, para o qual deve-se considerar:

> 1. *Todo ser humano tem direito à instrução. A instrução será gratuita, pelo menos nos graus elementares e fundamentais. A instrução elementar será obrigatória. A instrução técnico-profissional será acessível a todos, bem como a instrução superior, esta baseada no mérito.*
>
> *2. A instrução será orientada no sentido do pleno desenvolvimento da personalidade humana e do fortalecimento do respeito pelos direitos do ser humano e pelas liberdades fundamentais. A instrução promoverá a compreensão, a tolerância e a amizade entre todas as nações e grupos raciais ou religiosos e coadjuvará as atividades das Nações Unidas em prol da manutenção da paz.*

As pessoas com deficiência também têm direito à educação, a qual pressupõe que o aluno com deficiência participa efetivamente do ambiente escolar, como atividades pedagógicas, esportivas ou de lazer. Para a educação de pessoas com deficiência, não basta a utilização de recursos de Tecnologia Assistiva em sala de aula. Esses recursos passam a ser utilizados como suporte para a educação inclusiva, de modo a atender as necessidades específicas dos alunos com deficiência [Oliveira 2016].

De modo a atender às demandas educacionais dos alunos com deficiência, a educação inclusiva apoia-se na crescente expansão do uso de tecnologias no ambiente escolar [Macedo e Mara 2013]. Nesse âmbito, as Tecnologias da Informação e Comunicação têm grande potencial para favorecer o processo ensino-aprendizagem, pois são recursos que possibilitam o estímulo dos sentidos, permitem o compartilhamento de informações e promovem o relacionamento entre os alunos [Souza 2016].

Os materiais digitais utilizados como apoio no processo ensino aprendizagem permitem diferentes graus de interatividade e incluem todos os tipos de mídias digitais, como texto, vídeo, jogos e outros [Macedo e Mara 2013]. Estes materiais são elaborados por profissionais de educação, pois a capacidade de inspirar os alunos, a criatividade e a sensibilidade são únicas do profissional [Oliveira e Sousa 2016]. Nesse sentido, cabe ressaltar a importância da capacitação dos profissionais de educação, para que consigam utilizar essas tecnologias em favor do processo ensino-aprendizagem. A capacitação desses profissionais permite sua conscientização a respeito da importância da abordagem de questões de acessibilidade para a criação de materiais digitais.

De modo a auxiliar esses profissionais na incorporação de princípios de acessibilidade na criação de materiais digitais de aprendizagem, existem diversas diretrizes que constituem um conjunto de recomendações para tornar o conteúdo acessível. As questões de acessibilidade são discutidas principalmente pela *World Wide Web Consortium* (W3C), que disponibiliza um conjunto de princípios, recomendações e critérios de sucesso voltados para conteúdos *on-line*, por meio da disponibilização das diretrizes *Web Content Accessibility Guidelines* (W3C-WCAG).

1.9.1 *Web Content Accessibility Guidelines*

O *Web Content Accessibility Guidelines* (WCAG) define um conjunto de diretrizes de modo a tornar o conteúdo web mais acessível para pessoas com limitações funcionais. No entanto, essas diretrizes não cobrem todas as necessidades das pessoas com todos os tipos, graus e combinações de limitações [W3C 2008].

Esse padrão está organizado em quatro princípios que constituem a base da acessibilidade na Web (Tabela 1.2). Cada princípio possui um conjunto de diretrizes, para as quais são fornecidos, o total de 62 critérios de sucesso, distribuídos em três níveis de conformidade (A, AA, e AAA - do nível mais baixo para o nível mais alto). Ainda, cada diretriz e cada critério de sucesso possui um conjunto de técnicas que podem ser utilizadas.

Os princípios são:

- O princípio **Perceptível** refere-se a forma de apresentação da informação e dos componentes da interface de forma a serem percebidas pelo usuário. Esse princípio é composto pelas diretrizes:
  - **Alternativas em texto** (1 critério). Consiste em fornecer alternativas em texto para o conteúdo não textual conforme as necessidades dos usuários;
  - **Mídias baseadas no tempo** (9 critérios). Consiste em fornecer alternativas para conteúdo em multimídia;
  - **Adaptável** (3 critérios). Consiste em criar conteúdo a ser apresentado em diferentes formas, mantendo sua estrutura e formatação;
  - **Distinguível** (9 critérios). Consiste em separar o primeiro plano do plano de fundo, de modo a facilitar a audição e a visão dos conteúdos disponíveis;
- O princípio **Operável** refere-se a facilidade de navegação entre os componentes da interface do usuário. Esse princípio é composto pelas diretrizes:
  - **Teclado acessível** (3 critérios). Consiste em deixar toda a informação disponível a partir do teclado;
  - **Tempo suficiente** (5 critérios). Consiste em fornecer tempo suficiente para os usuários lerem e utilizarem o conteúdo disponibilizado;
  - **Convulsões** (2 critérios). Consiste em criar o conteúdo respeitando limites de *flashes* de modo a não causar convulsões;

- **Navegável** (10 critérios). Consiste em fornecer ajuda aos usuários para localizar conteúdos durante a navegação;

- O princípio **Compreensível** refere-se a forma de apresentação das informações e a organização da interface de modo que seja compreensível. Esse princípio é composto pelas diretrizes:
  - **Legível** (6 critérios). Consiste em elaborar o conteúdo textual de modo a ser facilmente compreendido;
  - **Previsível** (5 critérios). Consiste em sequenciar e apresentar o conteúdo de forma previsível;
  - **Assistência de entrada** (6 critérios). Consiste em ajudar os usuários a prevenir e corrigir os erros ocorridos;
- O princípio **Robusto** refere-se a robustez do conteúdo para ser interpretado por diversas tecnologias de apoio. Esse princípio contém a diretriz:
  - **Compatível** (2 critérios). Consiste em maximizar a compatibilidade com diferentes versões de tecnologias de apoio.

Tabela 1.2: Organização das diretrizes WCAG

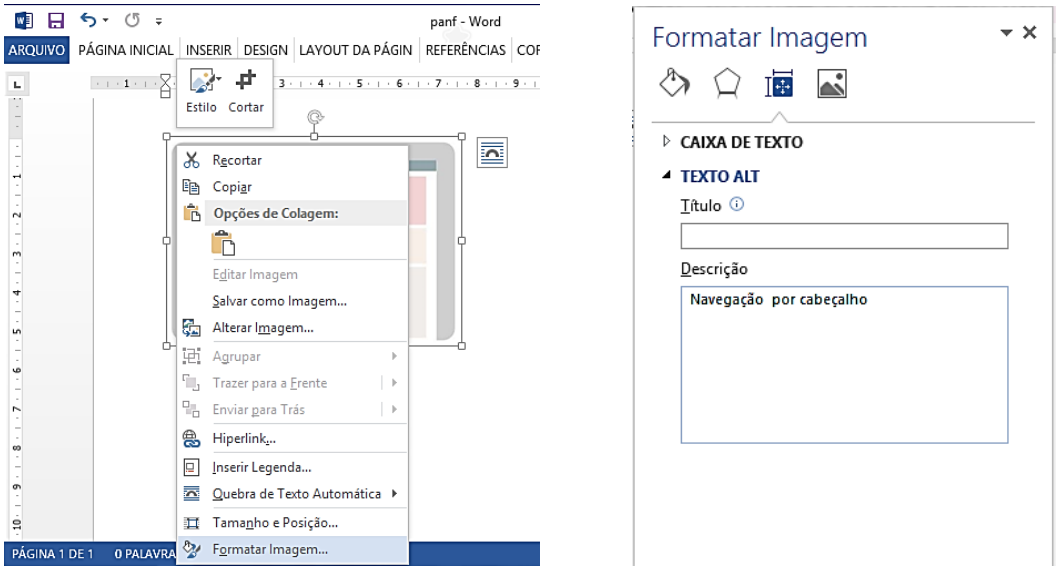
Princípio	Diretriz	Nível A	Nível AA	Nível AAA
1. Perceptível	1.1 Alternativas em texto	1.1.1	-	-
	1.2 Mídias baseadas no tempo	1.2.1 - 1.2.3	1.2.4, 1.2.5	1.2.6 - 1.2.9
	1.3 Adaptável	1.3.1 - 1.3.3	-	-
	1.4 Distinguível	1.4.1, 1.4.2	1.4.3 - 1.4.5	1.4.6 - 1.4.9
2. Operável	2.1 Teclado acessível	2.1.1, 2.1.2	-	2.1.3
	2.2 Tempo suficiente	2.2.1, 2.2.2	-	2.2.3 - 2.2.5
	2.3 Convulsões	2.3.1	-	2.3.2
	2.4 Navegável	2.4.1 - 2.4.4	2.4.5 - 2.4.7	2.4.8 - 2.4.10
3. Compreensível	3.1 Legível	3.1.1	3.1.2	3.1.3 - 3.1.6
	3.2 Previsível	3.2.1, 3.2.2	3.2.3, 3.2.4	3.2.5
	3.3 Assistência de Entrada	3.3.1, 3.3.2	3.3.3, 3.3.4	3.3.5, 3.3.6
4. Robusto	4.1 Compatível	4.1.1, 4.1.2		

1.9.2 Exemplos de aplicação

Nesta seção, são explorados alguns exemplos, considerando algumas diretrizes WCAG, que podem ser aplicadas em documentos Microsoft Word, de modo a tornar seu conteúdo acessível.

Ao considerar um documento com imagens, as pessoas com deficiência visual podem não conseguir compreender todo o contexto por causa da falta de acessibilidade da imagem. Na diretriz **Alternativas de Texto**, é sugerido o fornecimento de alternativas de texto para qualquer conteúdo não textual. Dessa forma, as imagens podem ser descritas textualmente de modo a fornecer informações a respeito de seu conteúdo. Uma técnica que pode ser utilizada, consiste no preenchimento da entrada "Alt" nas propriedades da imagem, no momento em que o documento está sendo criado. Na Figura 1.5, é

apresentado como aplicar essa técnica em um documento Word. Basicamente, deve-se selecionar a opção “Formatar imagem” (Figura 1.5a) e, em seguida, a opção “Todo texto”, na qual o campo “Texto alternativo” está disponível (Figura1.5b). Essa técnica possibilita que a descrição adicionada seja lida por leitores de telas, podendo ser aplicadas a tabelas, *links* e outros elemento os não textuais.



(a) Formatar imagem **(b) Texto alternativo**

Figura 1.5: Textos alternativos

Em um documento longo, os usuários com visão podem identificar rapidamente o que procuram ao observar a sua estrutura. Ao utilizar leitores de telas, por exemplo, os usuários também são capazes de identificar o que procuram por meio da estrutura do documento, desde que o padrão de marcação e a semântica de formatação sejam utilizados. Nesse contexto, a diretriz **Informações e Relacionamentos**, é sugerido que as informações, os relacionamentos e a estrutura sejam determinados. Para tanto, podem ser utilizadas convenções de formatação de texto. Na Figura 1.6, é possível perceber que esse padrão de formatação pode ser aplicado ao selecionar a opção “Estilo” para o texto, dividida em Título, Subtítulo, Referências, Parágrafos e outros.

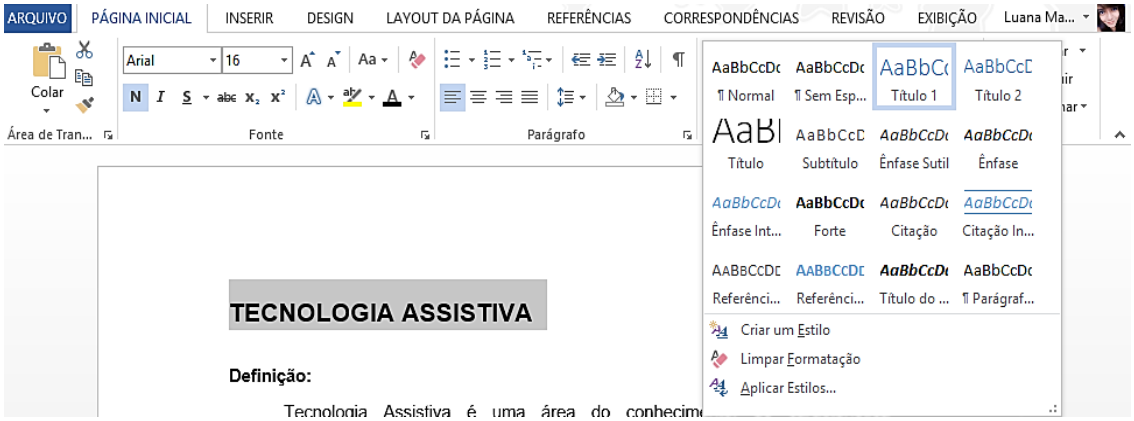
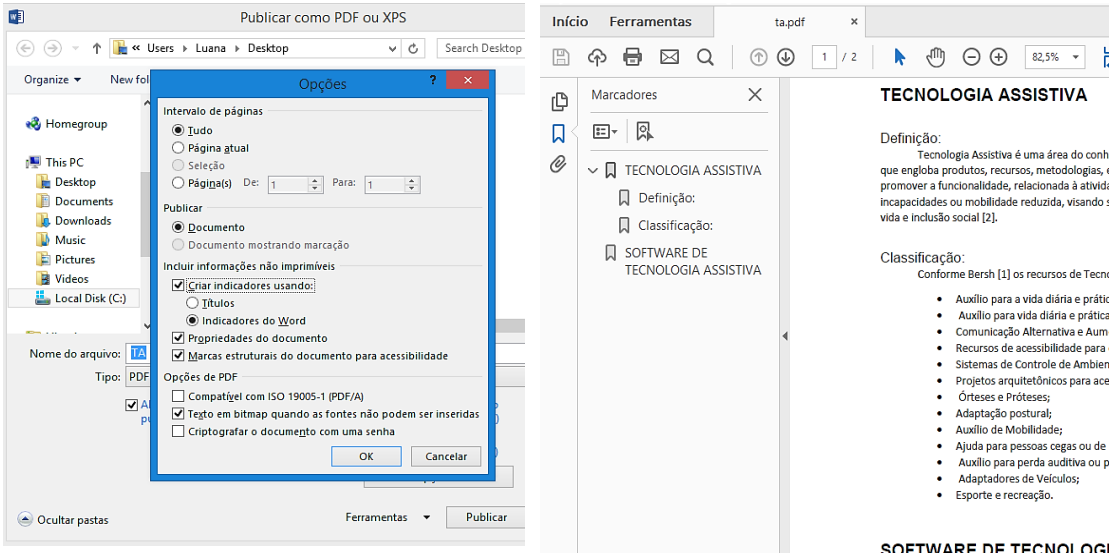


Figura 1.6: Padrão de formatação

Da mesma forma, os usuários podem localizar o conteúdo usando marcadores em documentos, disponibilizados como uma estrutura hierárquica de tópicos. Essa estrutura auxilia pessoas com deficiências cognitivas, que podem sentir-se perdidas com a quantidade de informações disponíveis no texto. Nesse contexto, com relação a diretriz **Navegável**, pode-se utilizar técnicas como a criação de marcadores em documentos. Tal estrutura pode ser dada utilizando a técnica descrita anteriormente. Ao padronizar a formatação, é possível criar uma estrutura hierárquica, que pode ser visualizada ao exportar o documento para PDF. Na Figura 1.7, pode ser identificada, no lado (a) a marcação das opções "Criar indicadores usando: títulos" e "Marcas estruturais do documento para acessibilidade" no momento de exportar o documento para PDF. No lado (b), a estrutura hierárquica criada é apresentada.



(a) Criar indicadores e marcas estruturais **(b) Resultado**

Figura 1.7: Padrão de formatação

Outro fator importante é a navegabilidade em tabelas. É preciso fornecer o contexto das células da tabela, o que inclui as células de cabeçalho da tabela associada a cada célula. Isso permite que os leitores de telas, por exemplo, forneçam o conteúdo da tabela de forma que o usuário saiba a que conteúdo as células estão associados. Na Figura 1.8, é mostrado que, para associar o cabeçalho às células da tabela, basta selecionar a tabela e, em opções de *layout*, selecionar a opção "Repetir linhas de cabeçalho".

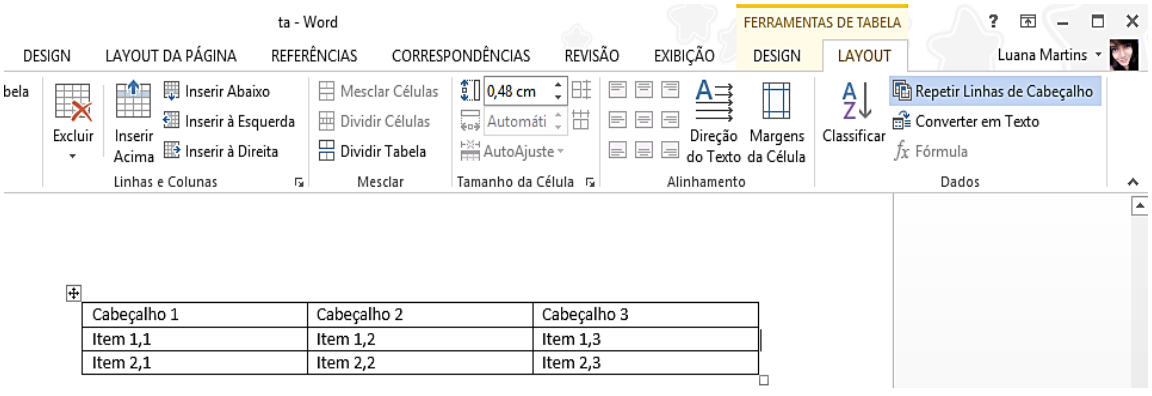
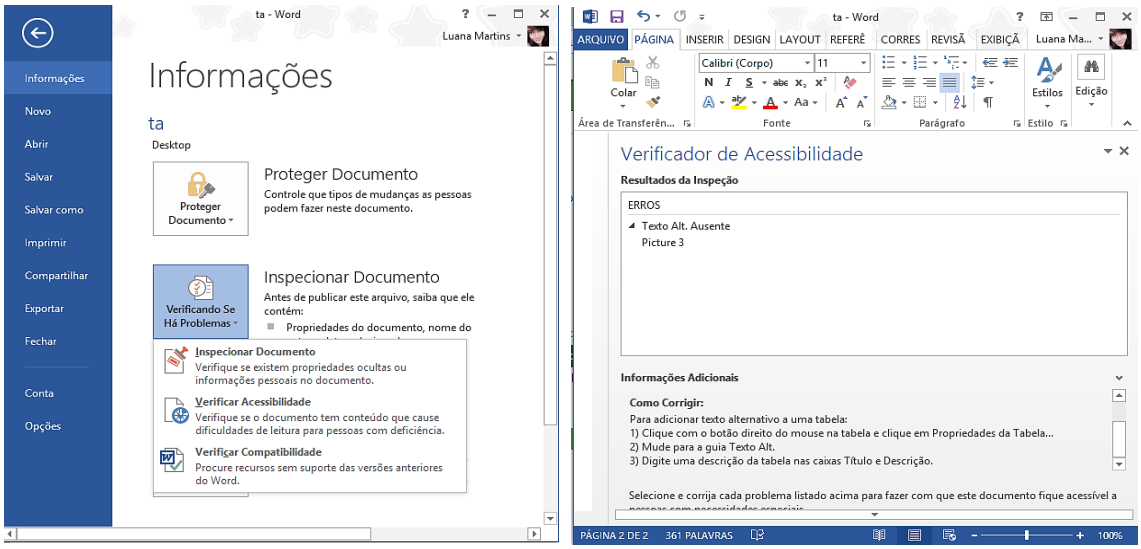


Figura 1.8: Navegação em tabelas

Nos exemplos abordados, são apresentadas algumas diretrizes essenciais para a acessibilidade do documento. Conforme definido pela WCAG, existem muitas outras que podem ser aplicadas, algumas com nível maior de dificuldade. Uma forma de conferir se o documento atende ao básico de acessibilidade é utilizar a opção "Verificar Acessibilidade", disponível para documentos Word. Na Figura 1.9a, é apresentado como selecionar a opção, em que a opção pode ser selecionada ao ir em "Informações do Documento" e, em seguida, "Inspecionar Documento". Os problemas encontrados são relatados conforme indicado na Figura 1.9b, juntamente com uma sugestão de como contorná-los.



(a) Verificar acessibilidade **(b) Problemas encontrados**

Figura 1.9: Verificar acessibilidade

1.10. Conclusão

As pessoas com deficiência compõem parte da população que sofre com a exclusão social, como resultado do preconceito sofrido ao longo da história, em que essas pessoas eram vistas como pessoas dependentes e incapazes de realizar suas atividades. Essa visão tem sido modificada ao longo dos anos, por meio da constante luta de pessoas com deficiência pela igualdade social. Atualmente, é considerado que a deficiência não está na pessoa, mas na relação entre a pessoa e o meio em que está inserida. Dessa forma, as limitações enfrentadas pelas pessoas com deficiência para interação com o meio devem ser facilitadas e superadas por meio da abordagem de questões de acessibilidade. Essas questões de acessibilidades podem ser abordadas por meio de recursos de Tecnologia Assistiva.

Existem diversos tipos de deficiência, como a deficiência visual, a auditiva, física, a cognitiva e a deficiência múltipla. As pessoas com deficiência são agrupadas conforme o tipo de deficiência, que podem possuir diferentes capacidades funcionais. Por este motivo, elas necessitam de recursos apropriados a suas capacidades. Então, os recursos de Tecnologia Assistiva visam atender as necessidades específicas de cada grupo ou pessoa com deficiência. Esses recursos incluem desde uma simples bengala, a software e hardware especiais que abordam questões de acessibilidade, a fim de proporcionar às pessoas com deficiência, mais independência, qualidade de vida e inclusão social.

Sistemas de software, como recurso de Tecnologia Assistiva, permitem / facilitam a interação entre pessoas com deficiência e tecnologias da informação e comunicação, de modo a permitir que as pessoas com deficiência tenham acesso as diversas informações disponibilizadas por esses meios, o que cria oportunidades comunicativas, cognitivas, sociais e culturais. Diversos recursos de software de Tecnologia Assistiva foram apresentados nesse capítulo, de modo a proporcionar uma visão geral dos recursos mais utilizados considerando os diferentes tipos de deficiência. Para a deficiência visual, foram apresentados recursos que tornam o texto disponibilizado na tela do computador acessível por meio de leitores de tela, que fornece um sintetizador de voz que realiza a leitura do conteúdo; e ampliadores de tela, que fornecem recursos de ampliação de texto, bem como recursos de dimensionamento de texto, opções de contraste, tamanho do cursor, e outros. Para a deficiência auditiva, foram apresentados recursos que tornam áudios, textos e imagens disponibilizadas em língua portuguesa acessíveis, por meio da tradução do conteúdo para a língua de sinais. Para a deficiência física, foram apresentados recursos que facilitam a escrita de textos, por meio de teclados virtuais e predição de texto. Para a deficiência cognitiva, foram apresentados *plug-ins* que visam tornar o conteúdo da Web mais fácil de ser lido e compreendido por pessoas que têm essa deficiência.

Esses recursos de Tecnologia Assistiva podem auxiliar no processo de educação de pessoas com deficiência. O direito a educação considera a inclusão de pessoas com deficiência no ambiente escolar, para as quais devem ser oferecidos métodos diferenciados de ensino, sem modificar os objetivos educacionais propostos. Para tanto, não basta apenas a utilização de recursos de Tecnologia Assistiva, é necessário a adequação de espaços físicos, recursos pedagógicos e a formação e capacitação de professores.

Portanto, percebe-se por meio do conteúdo apresentado nesse capítulo, que a inclusão social está estritamente relacionada a abordagem de questões de acessibilidade. Existem diversas políticas públicas e recursos de Tecnologia Assistiva, que visam a inclusão de pessoas com deficiência nos mais diversos aspectos da sociedade, como a inclusão nas escolas regulares. Apesar de ser um grande avanço, quando comparado ao histórico de exclusão sofrido por essas pessoas, ainda há muito o que ser trabalhado para que essas pessoas sejam completamente incluídas na sociedade.

Referências

ATHENA - Free AT Software Inventory. (2018). Free AT Software. Disponível em: http://access.uoa.gr/ATHENA/eng/pages/home. Acessado em: 26/04/2018.

Avelar, L. O., Rezende, G. C., & Freire, A. P. (2015). WebHelpDyslexia: A browser extension to adapt web content for people with dyslexia. Procedia Computer Science, 67, 150-159.

Beltrami, C. M., & de Moura, M. C. (2015). A educação do surdo no processo de inclusão no Brasil nos últimos 50 anos (1961-2011). Revista Eletrônica de Biologia (REB). ISSN 1983-7682, 8(1), 146-161.

Bersch, R. (2008). Introdução à tecnologia assistiva. Porto Alegre: CEDI, 21.

Brasil. (2015). Lei nº 13.146, de 6 de julho de 2015. Institui a Lei Brasileira de Inclusão da Pessoa com Deficiência (Estatuto da Pessoa com Deficiência). Diário Oficial da União.

CAT - Comitê de Ajudas Técnicas. (2009). Tecnologia assistiva. Brasília: Corde

CDPD - Resende, A. P. C., & VITAL, F. M. D. P. (2008). A convenção sobre Direitos das Pessoas com Deficiência Comentada. Brasília: Secretaria Especial dos Direitos Humanos. Coordenadoria Nacional para Integração da Pessoa Portadora de Deficiência, 164.

Damasceno, L. R. D. S. (2014). Direitos humanos e proteção dos direitos das pessoas com deficiência: evolução dos sistemas global e regional de proteção. Conteúdo Jurídico, Brasília-DF, v. 29.

eMAG. (2014). Modelo de Acessibilidade em Governo Eletrônico – Versão 3.1. Disponível em: <http://emag.governoeletronico.gov.br/>. Acessado em: 26/04/2018.

IBGE – Censo Demográfico. (2010). Características gerais da população, religião e pessoas com deficiência. Rio de Janeiro: Instituto Brasileiro de Geografia e Estatística.

Macedo, S., & Mara, C. (2013). Diretrizes de acessibilidade em conteúdos didáticos. InfoDesign: Revista Brasileira de Design da Informação, 10(2).

Ministério da Saúde (MS). Secretaria de Atenção à Saúde. (2006). Envelhecimento e saúde da pessoa idosa. Normas e manuais técnicos.

Moore, C. (2016). Results of the 2016 GOV.UK assistive technology survey - Low Vision. Disponível em: <https://accessibility.blog.gov.uk/2016/11/01/resu lts-of-the-2016-gov-uk-assistive-technology-survey/>. Acesso em: 30/04/2018.

No Brasil, Representação da UNESCO. (1998). Declaração universal dos direitos humanos.

Oliveira, A. (2016). Tecnologia Assistiva - Um Tema Em Ascenção: Aplicação de Recursos de Tecnologia Assistiva na Educação. Seminário de Pesquisa, Pós-Graduação e Inovação.

Oliveira, L. A., & Sousa, C. N. B. (2016). As tecnologias da informação no processo educacional. SIED: EnPED-Simpósio Internacional de Educação a Distância e Encontro de Pesquisadores em Educação a Distância.

OMS - Organização Mundial da Saúde. (2004). CIF: Classificação Internacional de Funcionalidade, Incapacidade e Saúde.

Ozawa, M. G. M., & Amaral, S. T. (2017). Breve Explanação sobre os Direitos Humanos em Relação à Pessoa com Deficiência Mental. ETIC - Encontro de Iniciação Científica do Centro Universitário Antônio Eufrásio De Toledo De Presidente Prudente, v. 13, n. 13.

PDE, Cadernos. (2016). Os Desafios da Escola Pública Paranaense na Perspectiva do Professor. Disponível em: <http://www.diaadiaeducacao.pr.gov.br/portals/cadernospde/pdebusca/producoes_pde/2016/2016_pdp_edespecial_unioeste_sueliferreirarocha.pdf>.

PEE - Programa Institucional de Ações Relativas às Pessoas com Necessidades Especiais. (2006). Pessoa com Deficiência: Aspectos Teóricos e Práticos. EDUNIOESTE, 20 ed., 143 p., 15-56.

Rocha Rodrigues, P., e Gama Alves, L. R. (2013). Tecnologia assistiva–uma revisão do tema. Holos, v. 6.

Rossini, D. (2014). WebHelpDislexia: O que é isso? Disponível em: <http://sopadenumerosecalculos.blogspot.com.br/2014/12/webhelp-o-que-e-isso.html>. Acessado em: 05/05/2018.

Sassaki, R. K. (2003). Terminologia sobre deficiência na era da inclusão. Mídia e deficiência. Brasília: andi/Fundação banco do brasil, 160-165.

Souza, A. M. (2016). As Tecnologias da Informação e da Comunicação (TIC) na educação para todos. Educação em Foco, v.21, n. 3, 349-366.

W3C - World Wide Web Consortium. (2008). Web content accessibility guidelines (WCAG). Disponível em: <https://www.w3.org/WAI/WCAG20 /quickref/#media-equiv>. Acessado em: 15 de maio de 2018.

WebAIM - Web Accessibility In Mind. (2013). Cognitive Disabilities Part 2 - Conceptualizing Design Considerations. Disponível em: <https://webaim.org/articles/cognitive/conceptualize/>. Acessado em: 28/04/2018.

WebAIM - Web Accessibility In Mind. (2013). Survey of Users with Motor Disabilities. Disponível em: <https://webaim.org/projects/motordisabilitysurvey/>. Acessado em: 28/04/2018.

WebAIM - Web Accessibility In Mind. (2013). Visual Disabilities - Low Vision. Disponível em: <https://webaim.org/articles/visual/lowvision>. Acessado em: 28/04/2018.

WebAIM - Web Accessibility In Mind. (2017). Screen Reader User Survey \#7 Results. Disponível em: <https://webaim.org/projects/screenreadersurvey7/>. Acessado em: 28/04/2018.

11. Capítulo 11

Autores:

Rosangela de Fátima Pereira Marquesone
Escola Politécnica da Universidade de São Paulo (Poli-USP)
email: rpereira@larc.usp.br

Francisco Pereira Junior
Universidade Tecnológica Federal do Paraná (UTFPR) - Cornélio Procópio
email: fpereira@utfpr.edu.br

Tereza Cristina Melo de Brito Carvalho
Escola Politécnica da Universidade de São Paulo (Poli-USP)
email: terezacarvalho@usp.br

Capítulo 11

Análise de Dados com R: uma Visão Inicial das Atividades de um Cientista de Dados

Rosangela de Fátima Pereira Marquesone, Francisco Pereira Junior e Tereza Cristina Melo de Brito Carvalho

Abstract

The unprecedented volume, variety, and velocity of data has provided significant changes in how data is currently used. Different areas such as medicine, marketing, telecommunication and retail are benefiting from data analysis strategy, generating results such as pattern discovery, process automation, creation of new customer connections, and development of new business models. In this context, data from various sources are used, much of this unstructured data obtained from external sources, such as sensors from devices of the Internet of Things, server logs and data from online social networks. To obtain value extraction capability, data analysis requires knowledge to capture, store, select, examine, prepare, model and visualize the data. Several solutions are currently being proposed, and the R language is very used in this process. One of the great features that make this language so popular is the vast set of libraries for different analysis strategies, from capture to data visualization. This course presents an introduction to the process of data analysis using the R language, besides presenting aspects of one of the professionals currently on the rise in IT: the data scientist.

Resumo

O volume, a variedade e a velocidade sem precedentes dos dados tem proporcionado mudanças significativas no modo como esses dados são utilizados atualmente. Diferentes áreas como a medicina, o marketing, telecomunicação e varejo estão se beneficiando da estratégia de análise de dados, gerando resultados tais como descoberta de padrões, automatização de processos, criação de novas conexões com clientes e desenvolvimento de novos modelos de negócios. Nesse contexto, dados de diversas fontes são utilizados, sendo muito destes dados não-estruturados, obtidos de fontes externas, tais como sensores de dispositivos da Internet das Coisas, logs de servidores e dados de redes sociais

online. Para obter a capacidade de extração de valor, a análise de dados requer conhecimentos para capturar, armazenar, selecionar, examinar, preparar, modelar e visualizar os dados. Diversas soluções estão sendo propostas atualmente, e a linguagem R é muito utilizada nesse processo. Uma das grandes características que tornam essa linguagem tão popular é o vasto conjunto de bibliotecas para diferentes estratégias de análise, desde a captura até a visualização de dados. Esse minicurso apresenta uma introdução ao processo de análise de dados utilizando a linguagem R, além de apresentar habilidades exigidas do profissional mais em alta no mercado de TI: o cientista de dados.

1.1. Introdução

Estamos na era dos dados. Essa afirmação é comprovada por diversos estudos e é reafirmada por inúmeros pesquisadores da área de negócios e tecnologia [Goldman et al. 2012, De Mauro et al. 2016, Marquesone 2016]. Uma das evidências é o crescimento do valor de mercado das companhias que detêm uma vasta quantidade de dados da população, como Facebook e Google. Porém, atualmente, não somente grandes empresas, mas também *startups*, pequenas e médias empresas estão gerando modelos de negócios guiados pela avalanche de dados, e extraindo informações capazes de identificar tendências de mercado, comportamento de consumidores e alavancar novas oportunidades de negócios.

Diferente do cenário tradicional, no qual os dados eram coletados em sua maioria de sistemas transacionais, tais como sistemas ERP (*Enterprise Resource Planning*) e CRM (*Customer Relationship Management*), atualmente os dados utilizados nos negócios são oriundos de uma diversidade de fontes, como registros de *logs* de servidores e sistemas, planilhas eletrônicas, documentos de texto e e-mails. Não fosse o bastante, o crescimento do volume de dados também está sendo impulsionado pelo uso massivo de sensores em sistemas de automação e de segurança, pela navegação na Internet e principalmente pelo uso expressivo das redes sociais online e dos aplicativos nos *smartphones*.

Diante deste cenário, surgiu o conceito de Big Data. A definição sobre esse termo é dada pela consultoria Gartner[1], descrevendo que "Big Data faz referência ao volume, variedade e velocidade de dados, necessitando de estratégias inovadoras e rentáveis para extração de valor dos dados e aumento da percepção". Essa definição demonstra a necessidade de uma quebra de paradigmas e uma mudança cultural para se inserir nesse contexto [Marquesone 2016].

O principal objetivo para o uso dos dados no contexto de Big Data é gerar conhecimento a partir deles. Diante a possibilidade de coletar, processar e analisar uma vasta quantidade de dados, extraindo percepções sobre eles, cresce também o interesse das corporações em tomar decisões orientadas por dados, no intuito de torná-las mais assertivas para seus negócios. Em busca de respostas à manipulação de dados e à extração de conhecimento sobre Big Data, nasceu a ciência de dados, e com ela a profissão de cientista de dados. Considerada uma atividade multidisciplinar, profissionais da área de ciência de dados agregam conhecimentos de diferentes disciplinas, tais como matemática, estatística e ciência da computação, para propor soluções adequadas e eficientes nesse contexto.

Conforme será apresentado, são inúmeras as etapas para o desenvolvimento de

[1] https://www.gartner.com/it-glossary/big-data

uma solução em Big Data. Todavia, grande parte da inteligência dada nesta solução está na etapa de análise de dados. Nessa área, também denominada de *data analytics*, destacam-se 4 categorias de análise: descritiva, diagnóstica, preditiva e prescritiva, conforme explicado a seguir e demonstrado na Figura 1.1.

- Análise descritiva: é a forma mais básica de análise cujo objetivo principal é a sumarização dos dados históricos de uma instituição para descrever os acontecimentos passados e responder à seguinte pergunta: "**o que aconteceu?**".
- Análise diagnóstica: ainda relacionada à análise de dados históricos (passado), busca-se com a análise diagnóstica identificar a causa dos acontecimentos ocorridos na instituição, e assim responder a pergunta: "**por que isso aconteceu?**".
- Análise preditiva: além de compreender o passado, esta categoria de análise é dotada de uma inteligência capaz de obter uma percepção do futuro. Ou seja, a análise preditiva tenta responder a pergunta: "**o que pode acontecer?**".
- Análise prescritiva: trabalhando também com a percepção de futuro, a análise prescritiva deve ter a capacidade de sugerir ações estratégicas e tomar decisões automáticas que se beneficiem das predições, e assim, indicar os meios para saber "**como fazer acontecer?**".

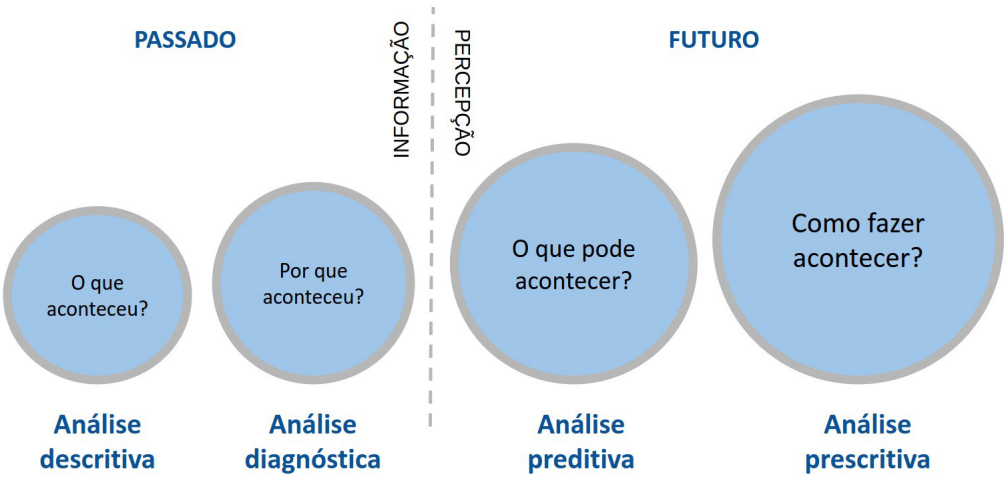


Figura 1.1. Categorias de *analytics* [Marquesone 2016].

Diversas tecnologias são utilizadas na ciência dos dados e, entre elas, pode-se destacar a linguagem R, por seu vasto conjunto de funcionalidades para explorar, visualizar, modelar e descobrir padrões nos dados. Nesse minicurso será apresentado como a linguagem R é utilizada atualmente por cientistas de dados durante todo esse processo de desenvolvimento de uma solução em Big Data. Para isso, são apresentados os cenários de aplicação de análises de dados e é realizada uma discussão sobre o papel do cientista de dados, com informações sobre as habilidades necessárias e a descrição do perfil desejado para esse profissional. Na sequência, são apresentadas as principais etapas do processo de análise de dados, demonstrando exemplos práticos de uso da linguagem R. Boa leitura!

1.2. As Etapas do Processo de Análise de Dados

O objetivo central de um projeto de análise de dados é extrair informações e novas percepções úteis para um determinado problema. Esse objetivo, porém, pode ocasionar diferentes resultados, de acordo com os dados utilizados, a informação e a percepção desejada. Por tal motivo, o processo de análise de dados varia de acordo com a definição do que se deseja obter com os dados. Por exemplo, um projeto com foco em analisar dados para gerar um modelo de risco de cartão de crédito contém atividades distintas de um projeto com foco em análise de sentimento de redes sociais. Entretanto, a literatura aponta um conjunto de etapas em comum no processo de análise de dados, sendo elas: seleção de dados, pré-processamento, transformação, mineração, interpretação e avaliação [Spector 2008]. Os tópicos a seguir apresentam detalhes de cada etapa.

1.2.1. Seleção de Dados

Ao iniciar um processo de análise de dados, uma das primeiras atividades a se realizar é a seleção dos dados. Um dos desafios existentes nesse processo refere-se à identificação de fontes de dados apropriadas para responder questões feitas aos dados. Ou seja, antes de selecionar os dados, é importante ter uma definição clara do que se deseja extrair de informação, pois é essa clareza que irá guiar a etapa de seleção, bem como as etapas seguintes.

Para realizar a seleção de dados é importante compreender suas categorias, bem como suas características específicas. Essa compreensão permitirá identificar e priorizar as estratégias de aquisição, gerenciamento e extração dos dados selecionados. Com base nessa afirmação os dados podem ser classificados a partir de diferentes perspectivas, tais como as descritas a seguir.

Dados internos são aqueles gerados, mantidos e controlados internamente no escopo de uma empresa. São provenientes dos próprios sistemas locais (produção, estoque, vendas, financeiro e RH); da interação dos colaboradores (documentos de texto, planilhas eletrônicas, mensageiros e e-mails); e das imagens de circuito de segurança; a característica primordial para dados internos é que eles sejam controlados pela própria empresa.

Dados externos são aqueles adquiridos de terceiros. Postagens em redes sociais online, dados de domínio público e de banco de dados abertos, publicações de satisfação e/ou reclamação do cliente final, projeções do mercado financeiro e das finanças públicas, são exemplos de dados externos.

Dados estruturados possuem como característica uma estrutura rígida, predefinida e bem organizada. Normalmente estão armazenados em um banco de dados relacional e respeitam as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Como exemplo, pode-se citar as tabelas de banco de dados relacional.

Dados não-estruturados possuem características opostas aos dados estruturados: não possuem um esquema predefinido para os dados; não atendem às exigências formais dos bancos de dados tradicionais; e, cada um dos registros pode ser organizado de forma diferente. Normalmente são armazenados na sua forma original, em representação binária ou textual. Como exemplos, podem-se citar: documentos de texto; arquivos em formatos de áudio e vídeo.

Dados semiestruturados entremeiam os dois anteriores, pois possuem uma estrutura predefinida, porém ela não é rígida. Geralmente é baseada em *tags*, cujo objetivo é marcar os atributos e indicar uma determinada hierarquia. Assim, o esquema de dados torna-se auto-descritivo, mesmo podendo os atributos variar de registro para registro. Como exemplos, podem-se citar: arquivos no formato XML, JSON, RSS e OWL.

Dados gerados por humanos são oriundos da interação das pessoas com o mundo digital e são criados explicitamente a partir do pensamento humano. Criação de documentos de texto, planilhas eletrônicas e apresentações; e-mails; postagens em blogs; postagens, curtidas e compartilhamentos em redes sociais online; avaliações de produtos, sites e serviços, são exemplos de dados gerados por humanos. Ou seja, as pessoas que geraram tais dados estão conscientes de como e quando o dado foi criado, pois a propriedade intelectual está atrelada ao seu conteúdo.

Dados gerados por máquinas são dados digitais criados automaticamente a partir de processos de computadores, de aplicações e da comunicação direta entre dispositivos, e assim, não necessitam da manifestação explícita de um ser humano na sua concepção. Além dos dados gerados por *logs* (de sistemas, de servidores, do monitoramento da navegação na Internet), sistemas de GPS e etiquetas RFID, a criação desse tipo de dado tem sido potencializada pelas tecnologias ligadas ao paradigma de Internet das Coisas (IoT - *Internet of Things*). A IoT que aprimorou ainda mais a interação máquina-a-máquina e tornou possível o monitoramento de situações, objetos e corpos por meio de dados capturados por sensores, atuadores e microchips, sem qualquer participação humana.

Dados biométricos são gerados a partir da identificação automática de uma pessoa, baseando-se em características anatômicas e/ou comportamentais. Como exemplos, podem-se citar: DNA, impressão digital e facial, forma do nariz, padrão de voz e forma de andar.

Dados numéricos referem-se a dados quantitativos, ou seja, dados mensuráveis, expressos em termos numéricos. Esses ainda podem ser classificados como dados discretos ou contínuos. Os dados discretos representam dados contáveis e adotam apenas valores inteiros, como o número de itens em um cesto de compras, números de bactérias em uma amostra e número de filhos. Já os dados contínuos são variáveis numéricas (inteiras ou decimais), normalmente resultantes de uma medição por algum instrumento e podem assumir qualquer valor dentro de um intervalo, como: peso e temperatura. Em geral, dados de séries temporais também utilizam variáveis contínuas ao longo do tempo, como por exemplo, um histórico do valor de uma ação na bolsa de valores.

Dados categóricos são considerados dados qualitativos, ou seja, dados que podem ser divididos em grupos. Nessa categoria os dados ainda podem ser divididos em nominais e ordinais. As variáveis nominais não apresentam uma relação de maior/menor entre elas, como por exemplo as variáveis gênero, raça e nome. Já nas variáveis ordinais há uma relação de ordem que as define, permitindo a realização de comparações entre elas. São exemplos de variáveis ordinais: nível de satisfação e status socioeconômico.

É importante ressaltar que um determinado conjunto de dados pode se enquadrar em mais de uma das categorias descritas anteriormente. Por exemplo, um dado pode ser ao mesmo tempo interno, semiestruturado, gerado por máquinas e conter variáveis

numéricas e categóricas.

Em resumo, algumas das principais atividades de um cientista de dados no processo de seleção de dados, são:

- Identificar perguntas que se deseja responder a partir da análise dos dados.
- Identificar quais fontes de dados podem auxiliar na extração de informação útil.
- Categorizar os dados de acordo com sua estrutura.
- Identificar em qual aspecto os dados selecionados podem contribuir com a análise.

1.2.2. Pré-processamento dos Dados

Um consenso existente no processo de análise de dados é que dificilmente as fontes de dados estarão prontas para serem analisadas. Na maioria das situações, os dados a serem utilizados podem conter as seguintes características:

- **Quantidades significativas de *outliers***. Essa ocorrência aponta para itens de dados que apresentam valores discrepantes do restante do conjunto. Embora a existência de *outliers* possa impactar o resultado da análise, sua manipulação deve ser feita com cautela, verificando se esse representa um comportamento legítimo ou ocorreu devido a uma falha na medição. Em casos onde seja identificado que o *outlier* refere-se a um erro, uma possível prática é corrigi-lo, se possível, ou removê-lo da base de dados. Caso comprovado que o dado é legítimo, sua permanência é valiosa para a análise. Entretanto, a distinção de um dado legítimo de um errôneo, por vezes, necessita da avaliação de profissionais que tenham conhecimento sobre tal domínio dos dados [Runkler 2016].

- **Dados incorretos e/ou irrelevantes**. Esse caso refere-se a dados cujos valores estão incorretos ou não são úteis para a análise desejada. Tais dados podem ser identificados por meio da análise de correlação entre as variáveis e de reconhecimento de padrão entre as observações.

- **Dados duplicados**. Essa ocorrência é comum, podendo ter sido causada por erros humanos ou de máquinas, principalmente em situações em que dados de diferentes sistemas foram integrados. Para evitar que a duplicidade afete o resultado da análise, tais ocorrências devem ser identificadas e removidas.

- **Dados ausentes**. Referem-se a determinadas observações das quais os registros de dados não estão completos. Isso pode ocorrer pela falta de recursos disponíveis no momento em que o dado foi coletado, ou pelo fato de que esses dados não foram medidos corretamente. Para tratar essas ocorrências, deve-se realizar uma avaliação no intuito de identificar o que originou a ausência desse dado, para apontar o melhor tratamento a ser feito, sem necessitar da remoção do registro inteiro.

Considerada uma das etapas primordiais no processo de análise de dados, o pré-processamento compreende um conjunto de técnicas para a limpeza e a preparação dos dados para a análise. Esse processo investigativo, na qual os dados são avaliados e compreendidos com detalhes é chamado de análise exploratória de dados. Nesse processo, inúmeras medidas, tais como média, mediana e desvio padrão são geradas com o objetivo de aumentar a compreensão dos dados [Velleman and Hoaglin 1981]. Além disso, a análise exploratória também faz uso de recursos visuais para auxiliar na investigação. Inúmeros gráficos podem ser gerados nesse processo, tais como o *boxplot*, histograma e *scatterplot*.

Embora seja uma tarefa árdua, a qualidade de execução dessa etapa é um dos fatores preponderantes para a extração de informações corretas. Do contrário, pode ocorrer o que pesquisadores contextualizam como *garbage in* e *garbage out*, referenciando ao fato de que usar dados inapropriados em uma análise vai gerar resultados incorretos, e consequentemente tomadas de decisão equivocadas.

Por fim, algumas das principais atividades de um cientista de dados no processo de pré-processamento, são:

- Explorar e visualizar os dados em busca de conhecê-los em detalhes, identificando necessidades de melhoria.
- Desenvolver e aplicar técnicas de limpeza dos dados, aumentando a confiabilidade dos dados para a análise.
- Gerar e armazenar novos conjuntos de dados mediante ao pré-processamento das fontes de dados anteriormente selecionadas.

1.2.3. Transformação dos Dados

A etapa de pré-processamento dos dados permite aumentar a confiabilidade dos dados selecionados para a análise. No entanto, é comum ocorrer que, mesmo após o pré-processamento, os dados ainda não estejam preparados de acordo com a necessidade da análise a ser realizada. Para estas situações, surge a etapa de transformação dos dados e as principais ações realizadas nessa etapa são:

- **Normalização dos dados**: há casos nos quais diferentes atributos possuem intervalos de valores diferentes. Caso esses sejam utilizados em conjunto, o resultado da análise pode ser afetado devido à distinção dos intervalos. A normalização é uma técnica utilizada para organizar os atributos em um intervalo específico.
- **Discretização dos dados**: técnica que visa o mapeamento do domínio de um atributo contínuo para um discreto. Nessa técnica ocorre um processo de divisão de um conjunto de atributos contínuos em intervalos categóricos.
- **Enriquecimento dos dados**: a partir da análise exploratória dos dados, pode-se descobrir a possibilidade de gerar novos atributos a partir dos existentes. Assim, essa técnica refere-se à complementação e atualização de informações, e até mesmo a inserção de novos atributos em um conjunto de dados. O objetivo principal dessa ação é melhorar o nível de qualidade do dado, objeto da futura análise.

- **Agregação dos dados**: é comum que os dados relevantes de uma análise não sejam coletados somente de um único conjunto, mas sim de diferentes fontes, em diferentes arquivos e sistemas. Portanto, essa técnica visa a combinação de dados de múltiplas fontes em um único conjunto.
- **Redimensionamento dos dados**: denominada análise de componentes principais (PCA - *Principal Component Analysis*), essa técnica estatística tem como objetivo encontrar uma projeção linear de um conjunto de dados correlacionados, gerando assim um conjunto substancialmente menor de variáveis não correlacionadas, mantendo a informação do conjunto de dados original. Algumas variações dessa técnica são a decomposição em valores singulares (SVD - *Singular Value Decomposition*), decomposição de Karhunen-Loève e funções empíricas ortogonais [Runkler 2016].

Em resumo, entre as atividades de um cientista de dados nessa etapa, pode-se destacar:

- Identificar, a partir do modelo que será utilizado na mineração de dados, quais dados necessitam ser normalizados ou discretizados.
- Avaliar e gerar novos atributos a partir dos dados existentes, visando o enriquecimento dos dados.
- Explorar outras fontes de dados existentes, avaliando a possibilidade de integrá-las aos dados selecionados.

Juntas, a seleção, pré-processamento e transformação dos dados são as etapas que consomem a maior parte do tempo no processo de análise dos dados. Estima-se que 80% do tempo de desenvolvimento de um projeto de Big Data seja gasto nessas etapas [Mitra et al. 2002, Marquesone 2016].

1.2.4. Mineração de Dados

Estando os dados pré-processados e transformados, inicia-se a etapa de mineração de dados. Em geral, principalmente no contexto de Big Data, utilizam-se algoritmos computacionais capazes de processar e extrair padrões dos dados, que muitas vezes são difíceis ou impossíveis de se observar somente por um olhar humano [Morabito 2015]. Essa etapa é considerada por [Flinn 2018] como uma fase de aprendizado, que tem como objetivo reduzir a incerteza sobre uma predição, melhorando assim a capacidade de se prever algo por meio dos dados. [Flinn 2018] ainda enfatiza que essa etapa contribui para reduzir as incertezas e não eliminá-las, uma vez que alcançar esse nível é considerado algo atípico.

Uma das primeiras atividades da etapa de mineração de dados consiste em selecionar o algoritmo a ser utilizado para a mineração e análise dos dados. Esses algoritmos podem ser divididos nas seguintes categorias: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço. Em [Wu et al. 2008], são apresentados 10 algoritmos de mineração de dados comumente utilizados.

Uma das estratégias utilizadas para verificar a possibilidade de se utilizar um algoritmo de **aprendizado supervisionado** é a verificação se há uma variável de saída (rótulo)

para todas as observações que serão utilizadas na construção do modelo. Após constatar a possibilidade de se utilizar um algoritmo de aprendizado supervisionado, o próximo passo é identificar o tipo de saída que se objetiva prever.

Diversos algoritmos são disponibilizados para a regressão, tais como a regressão linear simples, regressão linear múltipla e regressão logística. Por outro lado, caso o objetivo seja prever um valor a partir de um conjunto finito de classes, deve-se utilizar modelos de classificação. Exemplos de análises por meio de classificação são: prever o risco de ceder o crédito a um determinado cliente, prever o diagnóstico de um paciente com base nos resultados dos exames e prever a possibilidade de um e-mail ser considerado spam ou não-spam.

Conforme apresentado na Figura 1.2, os algoritmos de classificação necessitam de uma variável alvo, que indica as saídas existentes das observações utilizadas na construção do modelo. Diversos algoritmos de classificação podem ser utilizados, tais como: SVM (*Support Vector Machine*), *Random Forest*, e *Naive Bayes*.

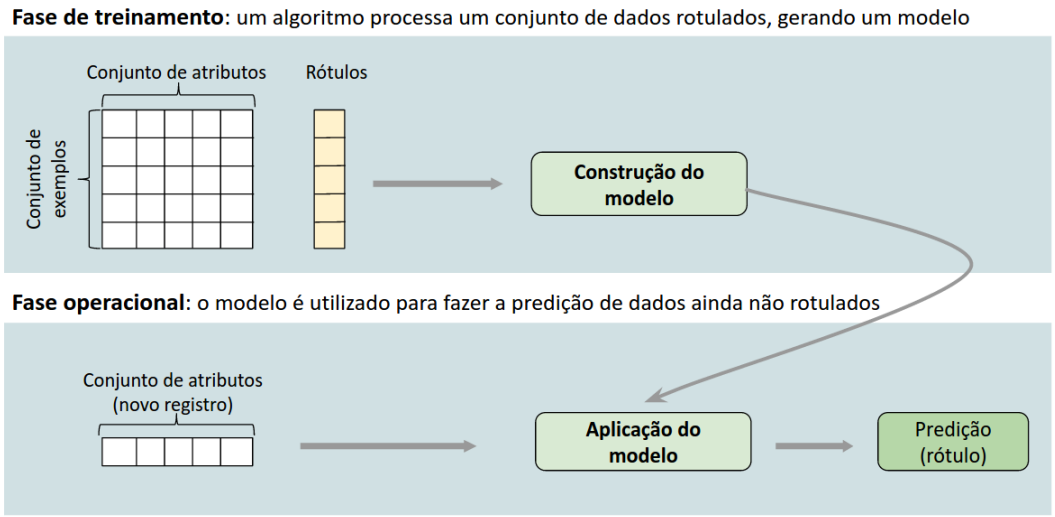


Figura 1.2. Fluxo de algoritmo de classificação [Marquesone 2016].

Na categoria de aprendizado supervisionado, as bases de dados são divididas em duas partes, chamadas de conjunto de treinamento e conjunto de teste. O conjunto de treinamento, representando normalmente entre 70 e 85% dos dados selecionados, é utilizado para treinar o modelo a partir dos dados de entrada e de saída em cada observação. Utiliza-se o conjunto de teste (respectivamente equivalente entre 30 e 15% dos dados selecionados) para avaliar o modelo, baseando-se no que se espera que ele consiga prever a partir de novas observações.

Não havendo a variável de saída discriminada, uma outra estratégia é a utilização de algoritmos de **aprendizado não-supervisionado**. Nessa categoria de aprendizado, não há uma clareza sobre as possíveis saídas do modelo, espera-se que o algoritmo seja capaz de identificar segmentos a partir dos padrões identificados nos dados de entrada. Esse processo é conhecido como agrupamento (*clustering*), que tem como objetivo atribuir rótulos a objetos em dados não rotulados [Runkler 2016].

São exemplos de mineração de dados por meio de agrupamento: segmentar clientes a partir de uma base de dados, para campanhas de marketing; identificar ações fraudulentas de ações legítimas, de acordo com o comportamento de uma compra; identificar imagens similares em bases de dados de imagens médicas.

A Figura 1.3 apresenta um exemplo de um fluxo de construção e utilização de um modelo de agrupamento. Entre as técnicas existentes nesse contexto, pode-se citar: *k-means*, agrupamento *fuzzy* e agrupamento hierárquico [Xu and Wunsch 2005].

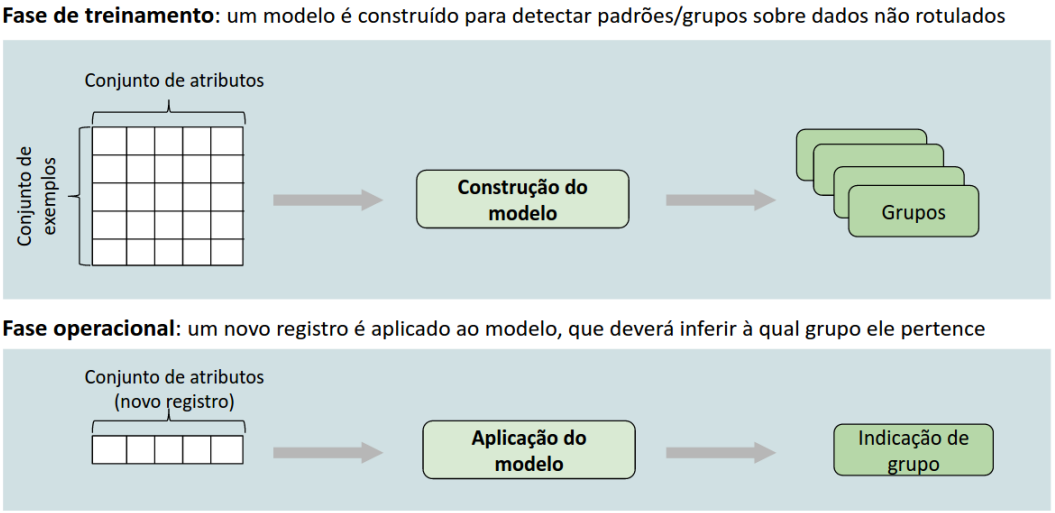


Figura 1.3. Fluxo de algoritmo de agrupamento [Marquesone 2016].

Os algoritmos de **aprendizado por reforço**, por sua vez, são utilizados em cenários nos quais os dados das variáveis de saída não estão disponíveis, porém, o modelo recebe um conjunto de informações ligadas à essa variável. Como exemplo pode-se citar o controle de movimentos de um robô e a definição de passos em uma partida de xadrez.

Cabe ressaltar também que há ainda algoritmos que podem ter o comportamento tanto de aprendizado supervisionado quanto de aprendizado não-supervisionado, como é o caso das redes neurais [Forte 2015].

Para concluir, algumas das principais atividades de um cientista de dados no processo de mineração de dados, são:

- Separar bases de dados entre conjuntos de treinamento e de teste.
- Selecionar o melhor algoritmo para a construção do modelo.
- Construir o modelo a partir do conjunto de treinamento.

1.2.5. Interpretação e Validação

Além de definir antecipadamente o que se deseja obter de resultado a partir de um modelo, uma boa prática é identificar qual o desempenho esperado desse modelo. Uma vez que os modelos não chegam a um nível de acerto de 100%, é necessário identificar qual o limite

mínimo de acerto do modelo que está sendo proposto. Dessa forma, essa etapa refere-se a um conjunto de técnicas para a interpretação e validação do modelo desenvolvido.

O conjunto de dados de teste (dados que não foram utilizados no treinamento do modelo) é empregado nessa etapa de validação. Uma baixa taxa de erro de validação indica que o modelo treinado contém uma boa representação da relação de entrada-saída dos dados [Runkler 2016]. Nessa etapa também insere-se a validação cruzada, um método utilizado para validar os modelos gerados, evitando a ocorrência de *overfitting*[2] [Hawkins 2004].

Algumas das principais atividades de um cientista de dados no processo de interpretação e validação, são:

- Executar o modelo com um conjunto de teste.
- Avaliar as métricas de validação, identificando se a qualidade do modelo está de acordo com o esperado.
- Reajustar o modelo ou os dados e construí-lo novamente, caso a qualidade do modelo não tenha atingido o desempenho esperado.

1.3. Casos de Uso

Embora não seja recente o apoio dos dados na tomada de decisão de negócios, essa prática vem se potencializando nos últimos anos, a partir do crescimento acelerado do volume de dados. Nessa seção será apresentado como a análise de dados no contexto de Big Data está oferecendo benefícios para as áreas de medicina, marketing, varejo e telecomunicações.

1.3.1. Medicina

A área médica é composta por uma variedade de dados, extraídos de diferentes fontes. Um conjunto de dados muito utilizado refere-se aos dados clínicos, extraídos de registros eletrônicos de saúde, compostos por informações como anotações médicas, informações de tratamentos, medicamentos e procedimentos laboratoriais de um paciente. Devido ao formato não estruturado, as análises de tais dados utilizam, em geral, técnicas de processamento de linguagem natural para extrair informações dos dados textuais [He et al. 2017].

De acordo com [Adjekum et al. 2017], o aumento da disponibilidade de dados heterogêneos em larga escala de pacientes foi o fator chave para a geração da área denominada medicina de precisão, que depende da agregação de dados de uma vasta quantidade e variedade de dados. Além dos registros eletrônicos de saúde, as bases de dados utilizadas na medicina de precisão são compostas por dados de sequenciamento do genoma humano, imagens médicas, registros clínicos, dados biométricos, dados de saúde móvel (*m-health*), dados de medicamentos, entre outros.

Como exemplos de como a análise de dados pode proporcionar avanços na medicina, pode-se citar: diagnóstico de câncer, aceleração no desenvolvimento de medicamentos, monitoramento de doenças e apoio na tomada de decisões clínicas.

[2]Termo utilizado em estatística para descrever um modelo estatístico que se ajusta ao conjunto de dados anteriormente observado, mas se mostra ineficaz na predição de novos resultados.

Um projeto notório de análise de dados em medicina é o "*Google flu trends*"[3]. Em busca de proporcionar uma nova alternativa para o monitoramento de doenças infecciosas, este projeto utilizou dados de consulta de pesquisa na Internet para prever tendências de disseminação de doenças. Como resultado, a aplicação desenvolvida foi capaz de identificar surtos de gripe na população cerca de duas semanas antes dos sistemas convencionais de saúde.

1.3.2. Marketing

O marketing enquadra-se na área de ciências sociais, que tem como objetivo estudar o comportamento humano. Muitos desses dados são oriundos atualmente de mídias sociais (redes sociais online, blogs, comunidades, fóruns), por essas produzirem volumes de dados em larga escala. Isso permite aos especialistas realizar um grande número de análises, tais como análises comportamentais, de segmentação e de fidelização, além de gerar um vasto número de indicadores de desempenho. Outra base de dados comum em departamentos de marketing são as que contém registros de avaliações de clientes de um determinado produto ou serviço. Esta área de marketing em que as ações de marketing são apoiadas por dados é atualmente denominada *data-driven marketing* [Sundsøy et al. 2014].

Entre as atividades de um cientista de dados em marketing, pode-se citar:

- Geração de sumarizações que revelem diferenças entre grupos de clientes, utilizando métricas como médias de grupo e tabulações cruzadas.
- Análise de cesto de compras [Kaur and Kang 2016], na qual se identifica a partir de uma base de dados histórica, quais itens são comprados em conjunto em uma única compra.
- Análise de sentimentos, permitindo compreender a atitude ou a reação emocional de um sujeito diante de um tópico ou marca específica.
- Análise comportamental, permitindo compreender o comportamento dos clientes em cada canal e ponto de interação, para assim identificar quais aspectos influenciam suas ações. Tal técnica pode auxiliar na tomada de decisão de campanhas de marketing, gerando conteúdo personalizado de acordo com os comportamentos identificados.

Um dos casos de uso de análise de dados para estratégias de marketing é o do serviço de *streaming* de músicas Spotify. Para compreender melhor seu cliente, oferecer uma melhor experiência, e realizar campanhas de marketing efetivas, a empresa realiza análise a partir de 3 bases principais: (1) de *playlists*, que contém informações de músicas tratadas como similares; (2) histórico de audições individuais, que revela padrões na ordem em que as músicas são tocadas; e, (3) *likes*, *dislikes* e saltos de músicas, revelando as preferências dos usuários [Kazakov 2018]. Em 2016 a empresa analisou dados de milhões de usuários e *playlists* armazenadas em sua base, e por meio de uma equipe formada por profissionais de marketing e cientistas de dados, identificou comportamentos

[3]https://www.google.org/flutrends/about/

considerados relevantes ou engraçados de seus usuários, utilizando tais informações para uma campanha global chamada "*Thanks, 2016. It's been weird*" [Roberts 2016].

1.3.3. Varejo

Em [Kumar et al. 2017], os autores categorizam as estratégias de análises de dados no setor varejista em 4 níveis, sendo eles:

- mercado: análises referentes à precificação, expansão de mercado, *marketing share* e publicidade.
- empresa: análises relacionadas ao marketing multi-canais, alianças e marcas de loja.
- loja: análises e estratégias associadas à localização, incluindo, por exemplo, precificação dinâmica e promoções.
- cliente: análises que permitem identificar aspectos sobre a experiência, fidelização, satisfação e engajamento do usuário.

O varejo é considerado uma das áreas que geram e armazenam bases de dados gigantescas, e uma das empresas notórias nesse contexto é a Walmart. Já em 2012, estimava-se que a empresa coletava mais de 2.5 petabytes de dados a cada hora, referentes às transações de seus clientes [McAfee et al. 2012]. Além desse volume em larga escala, a empresa também revela ser capaz de pesquisar bilhões de documentos de mídias sociais, identificando fatores como tendências, produtos populares, e sentimentos dos clientes, inclusive separado por localização geográfica [Edosio 2014].

1.3.4. Telecomunicações

Os dados coletados no setor de telecomunicações são capazes de evidenciar tendências e comportamentos dos clientes de uma operadora. Exemplos incluem os dados de registro de chamada (CDR - *Call Detail Record*), dados de comportamento do usuário em mídias sociais, dados de histórico de bilhetagem, histórico de compras e de avaliações de serviço.

Além desse objetivo, tais dados também podem ser utilizados internamente nas empresas de telecomunicações, permitindo otimizar atividades de gerenciamento financeiro, de recursos de infraestrutura, cadeia de suprimentos, entre outros. Exemplos de dados incluem os registros de *log* da rede, registros de erros e dados de indicadores do *call center*. A habilidade de analisar e extrair percepções de tais dados podem resultar no aperfeiçoamento dos produtos e serviços prestados [Jose et al. 2017].

Uma característica típica dos dados de telecomunicações é que esses normalmente precisam ser analisados em tempo real ou muito próximo ao tempo real, uma vez que serviços e informações aos clientes desse setor necessitam dessa velocidade de análise. Além dos provedores de infraestrutura de telecomunicação, as empresas provedoras de aplicativos coletam e analisam uma vasta quantidade de informação obtida dos clientes pela rede [Yazti and Krishnaswamy 2014]. Para atender a esse requisito, empresas de telecomunicações precisam lidar com desafios referentes à captura e ao processamento de

dados de grande volume, à redução de latência entre a captura de um evento e sua análise e ao desenvolvimento de algoritmos capazes de extrair valor dos dados obtidos.

Entre as oportunidades existentes a partir da análise de dados nessa área, pode-se citar:

- Detecção de fraude: os inúmeros dados obtidos dos clientes de telecomunicações possibilitam que as empresas identifiquem comportamentos suspeitos que caracterizam uma fraude, e realizem ações que combatam tais eventos.
- Prevenção de falhas: técnicas de análise preditiva permitem a identificação antecipada de potenciais falhas na infraestrutura de rede, para que assim a empresa atue na sua resolução antes que o evento ocorra e aumente a qualidade do serviço prestado aos clientes.

Um exemplo de caso de uso sobre análise de dados nesse setor é o da empresa Telefônica, uma das maiores empresas de telecomunicações em nível global. Com o objetivo de extrair informações referentes à movimentação da população no espaço e no tempo, foi desenvolvida a plataforma estatística chamada *Smart Steps*. Esta plataforma possui funcionalidades para inferir o movimento de um conjunto aglomerado de pessoas a partir do uso da infraestrutura de telefonia móvel. Conforme apresentado em [Feriancic et al. 2015], os dados foram capturados em nível nacional (no Brasil), obtendo informações de registros de celulares de aproximadamente 80 milhões de pessoas. Essa vasta quantidade de informações permitiu identificar padrões e obter novas percepções em cenários como a previsão de congestionamentos de veículos no trânsito e a necessidade de maior demanda de infraestrutura de rede.

1.4. O Papel do Cientista de Dados

À medida que as empresas foram percebendo a necessidade de analisar e extrair informações de um volume cada vez maior de dados, surgiu a necessidade de um profissional com habilidades para tais tarefas. Entretanto, no início, não havia muita clareza sobre quais eram os papeis desempenhados por esse profissional, e quais capacidades esse deveria possuir. Considerada uma profissão recente, o termo **cientista de dados** foi cunhado por DJ Patil e Jeff Hammerbacher em 2008 [Patil and Davenport 2012]. Em busca de fornecer uma melhor compreensão desse termo, essa seção apresenta informações referentes ao perfil do cientista de dados e descreve sugestões para se tornar esse profissional.

1.4.1. Perfil do Profissional

Embora esteja mais consolidado, ainda é considerado um desafio traçar um perfil do profissional cientista de dados. No entanto, em geral, esse profissional possui as seguintes habilidades:

- Pensamento analítico: conforme indicado em [Flinn 2018], muitas empresas estão buscando profissionais com doutorado para exercerem o papel de cientista de dados. Isso porque são treinados em transformar problemas em questões bem definidas, identificar e avaliar metodologias de pesquisa, comparar métodos e tecnologias, identificando a solução mais adequada para obter a resposta mais assertiva.

- Curiosidade: por se tratar de uma profissão que exige o processo criativo, é essencial que o cientista de dados tenha curiosidade para testar hipóteses, levantar perguntas e testar possibilidades de extração de valor dos dados.

- Criatividade: refere-se a criatividade para extrair informações úteis, gerar visualizações e identificar padrões que sejam claros e convincentes.

Em suma, as atividades de um cientista de dados incluem:

- Explorar bases de dados e levantar perguntas.

- Estruturar dados não estruturados.

- Construir modelos.

- Automatizar análises.

- Desenvolver produtos e serviços guiados por dados.

- Coletar métricas.

- Contar histórias a partir dos dados.

- Integrar bases de dados identificando correlações entre elas.

É importante lembrar que um cientista de dados pode se enquadrar em diversos perfis, conforme o relatório "*Analyzing the Analyzers*" [Harris et al. 2013]. A definição desse perfil auxilia na compreensão de qual área um cientista de dados deve se aprofundar, podendo ser, conforme indica o relatório, a área de negócios, criatividade com os dados, pesquisa ou desenvolvimento. Além dessa variedade de perfis, a partir da profissão cientista de dados surgiram outros papéis, que atuam em conjunto na execução de um projeto, tais como o engenheiro de dados, o arquiteto de dados, o engenheiro de *machine learning* e o Diretor Executivo de Dados (CDO - *Chief Data Officer*).

1.4.2. Sugestões para se Tornar um Cientista de Dados

Atualmente há um certo temor em relação ao futuro da profissão cientista de dados, devido à possibilidade de automatização do processo de análise dos dados por meio de avanços da inteligência artificial. Como exemplo, pode-se citar o projeto AutoML do Google [Silver et al. 2017], em que algoritmos são capazes de desenvolver novos algoritmos sem a necessidade de interferência humana. No entanto, as habilidades de raciocínio de um profissional cientista de dados ainda são consideradas fundamentais para o avanço de soluções guiadas por dados [Flinn 2018].

Entre as sugestões para que um profissional possa se tornar um cientista de dados, estão:

- Habilidade com linguagem R ou Python. Segundo DJ Patil, a habilidade essencial de um cientista de dados é saber desenvolver códigos [Patil and Davenport 2012]. Portanto, é imprescindível que um cientista de dados domine uma linguagem de programação. Embora muitas linguagens possam ser utilizadas em um projeto de análise de dados, um requisito comum entre as empresas é que o profissional tenha habilidade com a linguagem R ou Python, consideradas cada vez mais importantes para o cientista de dados.

- Conhecimento sobre tecnologias de Big Data. Além de R e Python, principalmente em projetos que lidam com grandes volumes de dados, o cientista de dados também deve ter domínio sobre tecnologias de Big Data. Entre as tecnologias existentes, recomenda-se o conhecimento em soluções *open source* do ecossistema Hadoop, ecossistema Spark e bancos de dados NoSQL (*Not only SQL*).

- Conhecimento em estatística. Mesmo que o profissional cientista de dados atue no desenvolvimento de código e utilize bibliotecas com funções estatísticas pré-definidas, é imprescindível que esse tenha conhecimento em estatística, de forma a selecionar corretamente as equações, teoremas e métodos necessários para a análise de dados.

Soluções como o Kaggle[4] também podem auxiliar no treinamento de habilidades para se tornar um cientista de dados. Essa plataforma oferece um modelo de competição online no qual empresas e universidades lançam desafios de ciência de dados e tem uma comunidade ativa de pessoas com foco em disseminar o conhecimento sobre essa área.

1.5. Linguagem R

Considerada uma das soluções mais adequadas para projetos de ciência de dados, a linguagem R tem se destacado nessa área. Características como a diversidade de técnicas e métodos disponíveis para importar, preparar, modelar e visualizar dados de diferentes estruturas e fontes, auxiliam a jornada de uma análise de dados.

1.5.1. História

A história do R iniciou em 1992, no departamento de estatística da Universidade de Auckland, na Nova Zelândia. Ross Ihaka e Robert Gentleman, professores desse departamento, iniciaram o desenvolvimento de um projeto de pesquisa com o intuito de desenvolver uma solução para executar testes estatísticos. Neste projeto eles decidiram adotar a sintaxe da linguagem "S", desenvolvida no Bell Laboratories nos anos 70, como base para a linguagem que eles estavam criando. E, por brincadeira, decidiram nomear a linguagem desenvolvida como "R", referente às iniciais do nome dos dois pesquisadores.

A primeira versão da linguagem foi lançada em 1994, despertando o interesse da comunidade acadêmica, que passaram a adotá-la nas aulas de estatística. Incentivado por outros pesquisadores, os criadores do R tornaram a linguagem *open source* e a disponibilizaram por meio da licença GPL (*General Public License*), o que permite que a linguagem receba contribuições de diversos outros colaboradores. Em 1996 o projeto R

[4]https://www.kaggle.com

foi formalmente apresentado ao público no periódico "*The Journal of Computational Statistics and Graphics*", por meio do artigo intitulado: "*R: A Language for Data Analysis and Graphics*"[5].

1.5.2. Vantagens e Desvantagens

Atualmente os cientistas de dados têm à disposição diversas tecnologias e ferramentas para um projeto de análise de dados, o R é uma das mais populares. A seguir são apresentados os benefícios e as limitações dessa linguagem.

- Plataforma única para análise de dados. A partir do R os cientistas de dados podem executar todas as etapas do processo de análise de dados, desde a seleção até a interpretação e validação do modelo. Primeiramente os dados são importados para o R, e a partir dessa ação, pode-se executar operações de limpeza, transformação, modelagem e visualização dos dados.

- Permite integração com diversos bancos de dados. O R oferece mecanismos que possibilitam a análise de dados de diferentes formatos e fontes, tais como arquivos de texto com delimitadores, sistemas estatísticos, bases de dados relacionais, arquivos XML, PDF e até mesmo dado de *streaming*. Além disso, essa característica do R facilita o desenvolvimento de projetos que necessitam da integração de diversas fontes de dados.

- Grande variedade de recursos. Possivelmente um dos fatores mais atrativos do R é a sua variedade de métodos estatísticos e visualização de dados disponíveis por meio de pacotes. Os detalhes dessa variedade é apresentado na subseção 1.5.3.

- Ampla comunidade. Uma das características que tornam o R tão atrativo, assim como a linguagem Python, é o grande número de usuários que fazem parte da comunidade *open source*, contribuindo e auxiliando na evolução e utilização da linguagem.

Entretanto, principalmente no contexto de análises de Big Data, o R apresenta algumas limitações que podem comprometer o processo de análise de dados. A seguir são apresentadas algumas dessas restrições:

- Não possui paralelismo. A linguagem R não possui nativamente soluções para processamento paralelo, como execução de múltiplas *threads*, impedindo-a de aperfeiçoar seu desempenho em computadores com vários núcleos de processamento.

- Dados em memória. Nativamente o R carrega todo o conjunto de dados na memória, para que o usuário possa realizar operações sobre ele. Esse fator pode ser um empecilho caso o tamanho da base de dados utilizada exceda a capacidade de memória.

[5]https://www.tandfonline.com/doi/abs/10.1080/10618600.1996.10474713

Embora essas restrições possam impedir a análise de grande volume de dados, atualmente são propostos pacotes R específicos para superar tais limitações. Uma das possibilidades é a integração do R com o arcabouço Hadoop, por meio do pacote *RHadoop*[6], permitindo que a aplicação em R obtenha a escalabilidade de um ambiente distribuído. Outras iniciativas são o pacote *snow*[7], para execução do R com MPI (*Messaging Parsing Interface*) e o *multicore*[8], para paralelizar o R em ambiente com múltiplos núcleos de processamento.

1.5.3. Pacotes R

Um dos principais fatores que tornam o R uma solução atrativa para cientistas de dados é o vasto conjunto de pacotes que ele oferece. Tais pacotes, desenvolvidos por colaboradores da comunidade de software livre, são coleções de funções, bases de dados e códigos compilados, que auxiliam no processo de descobertas e análises de dados.

O CRAN[9] (*Comprehensive R Archive Network*) é o repositório oficial para o armazenamento dos pacotes R. Até a data da escrita desse texto, o repositório possuía aproximadamente 12.000 pacotes disponíveis. A Tabela 1.1 apresenta alguns exemplos dos pacotes existentes para serem utilizados em projetos de ciência de dados.

Tabela 1.1. Exemplos de pacotes disponíveis no R

Categoria	Pacotes
Importação de dados	Arquivos Excel: **readxl** APIs: **httr** Arquivos XML: **xml**
Preparação dos dados	Agregação dos dados: **aggregate** Manipulação de matrizes: **matrix** Mineração de texto: **tm**
Algoritmos de classificação	Árvores de decisão: **rpart** Random Forest: **randomForest** Regressão: **glm**
Algoritmos de agrupamento	Cluster particionado: **kmeans** Cluster hierárquico: **hclust** Cluster baseado em modelo: **mclust**
Visualização de dados	Gráficos estáticos: **ggplot2** Gráficos interativos: **shiny** Gráficos de séries temporais: **dygraphs**

1.6. Atividade Prática

Para complementar o conteúdo teórico observado anteriormente, nessa seção será apresentado um exemplo prático do uso do R para análise de dados. O código-fonte dessa

[6]https://github.com/RevolutionAnalytics/RHadoop/wiki
[7]https://cran.r-project.org/web/packages/snow/index.html
[8]https://cran.r-project.org/web/packages/multicore/index.html
[9]https://cran.r-project.org/web/packages/

atividade pode ser encontrado no repositório GitHub, por meio do seguinte link: `https://github.com/jolai-r/minicurso`

Para auxiliar o leitor no conhecimento da linguagem R e apresentar um exemplo prático de análise de dados, o repositório de código contém duas atividades principais:

- Introdução ao R: essa atividade tem como objetivo apresentar aspectos básicos da linguagem, tais como a chamada de funções, operações matemáticas e manipulação de strings, para auxiliar o leitor na familiarização com o R.
- Análise de dados com R: o objetivo dessa atividade será analisar uma base de dados de usuários, gerando recomendações de acordo com as preferências de outros usuários com gostos similares, conforme descrito na subseção 1.6.1.

Para tais atividades, as seguintes ferramentas foram utilizadas:

- R versão 3.4.1[10]
- RStudio versão 1.1.383[11]

1.6.1. Descrição da Atividade de Análise de Dados com R

A atividade prática de análise de dados com R será baseada no contexto de recomendações de conteúdo para usuários de serviços de *streaming* de músicas. Mais especificamente, o objetivo será responder a seguinte pergunta a partir dos dados: "É possível recomendar uma música a um usuário, baseando-se em uma correlação de suas preferências com a de outros usuários?"

Para responder essa pergunta utilizaremos o arquivo *lastfm.csv*, existente no mesmo repositório *github* descrito anteriormente. Last.fm[12] é uma empresa criada no Reino Unido em 2002, com foco em oferecer uma rádio online para usuários interessados em música. A base de dados utilizada corresponde a uma lista de audições dos usuários da Last.fm de bandas europeias, e foram extraídas do projeto *Million Song Dataset*[13].

Para a análise proposta será utilizada a filtragem colaborativa, uma técnica para sistemas de recomendação da qual torna-se possível prever e recomendar itens a usuários, baseado em preferências similares [Ekstrand et al. 2011]. As etapas sugeridas aqui estão pautadas nas etapas descritas no processo de análise de dados expostas neste texto. Por questões de escopo, não será apresentado o conjunto de ações necessárias para incluir o resultado da análise em produção.

1.6.2. Seleção

O primeiro passo a ser realizado no R é importar a base de dados e explorá-la, para conhecer sua estrutura. O carregamento da base de dados pode ser feito por meio da seguinte instrução de código:

[10]https://cran.r-project.org/bin/windows/base/old/3.4.1/
[11]https://www.rstudio.com/products/rstudio/download/
[12]https://www.last.fm
[13]https://labrosa.ee.columbia.edu/millionsong/lastfm

```
base <- read.csv(file="lastfm.csv")
```

1.6.3. Pré-processamento

Por convenção, o R utiliza o símbolo «-"para a atribuição de valores a uma variável. Na instrução anterior foi criado um objeto *base* do tipo *dataframe*, que conterá o conteúdo textual do arquivo *lastfm.csv*. A verificação do conteúdo do arquivo pode ser feita da seguinte maneira:

```
str(base)
'data.frame':   1257 obs. of  286 variables:
 $ user                     : int  1 33 42 51 62 75 130 ...
 $ a.perfect.circle         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ abba                     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ac.dc                    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ adam.green               : int  0 1 0 0 0 0 0 0 0 0 ...
 $ aerosmith                : int  0 0 0 0 0 0 0 0 0 0 ...
...
```

Neste exemplo foi utilizada a chamada da função *str(<objeto>)*, que apresenta informações sobre a estrutura dos dados a serem analisados. O resultado permite identificar que os dados correspondem a um total de 1257 observações de 286 variáveis, sendo todas as variáveis do tipo inteiro. Por questões de limitação do escopo do texto, foi apresentada somente uma amostra da saída da função.

Uma outra função útil para verificar o conteúdo dos dados é a *head()*, que demonstra os registros de um determinado grupo de observações. Para facilitar a visualização, os argumentos foram configurados para apresentar somente os resultados das colunas 3 a 8, conforme apresentado a seguir.

```
head(base[,c(1,3:8)])
   user abba ac.dc adam.green aerosmith afi air
1     1    0     0          0         0        0
2    33    0     0          1         0        0
3    42    0     0          0         0        0
4    51    0     0          0         0        0
5    62    0     0          0         0        0
6    75    0     0          0         0        0
```

Em algoritmos de filtragem colaborativa, os dados utilizados referem-se às informações de preferências dos usuários, não sendo utilizadas informações sobre o perfil do usuário. Por esse motivo, uma transformação a ser realizada será a remoção da coluna que contém a variável *"user"*, conforme apresentado a seguir.

```
base <- (base[,!(names(base) %in% c("user"))])
```

Dessa forma, a partir da instrução anterior, o objeto *base* foi alterado, contendo somente uma matriz de frequência de músicas ouvidas pelos usuários.

Para gerar a recomendação, será necessário calcular a similaridade de cada banda, com todas as demais. Para isso, será utilizada nessa análise a medida de similaridade

de cosseno, que tem como objetivo gerar uma medida que calcula o cosseno do ângulo entre dois vetores [Sarwar et al. 2001]. Para gerar essa medida, a seguinte função deve ser implementada:

```
calculaCosseno<- function(x,y)
  {
    cosseno <- sum(x*y) / (sqrt(sum(x*x)) * sqrt(sum(y*y)));
    return(cosseno);
  }
```

1.6.4. Transformação

A seguir, será criada uma matriz contendo a estrutura necessária para realizar a comparação entre os itens e armazenar as medidas de similaridade. Para isso, essa matriz deverá ter os itens dispostos em linhas e colunas, conforme descrito a seguir.

```
basecosseno  <- matrix(NA, nrow=ncol(base), col=ncol(base),
dimnames=list(colnames(base), colnames(base)))
```

A partir dessa instrução foi criado o objeto chamado *basecosseno*, até o momento sem valores preenchidos. Os valores serão obtidos a partir da chamada da função *calculaCosseno()*, anteriormente implementada.

1.6.5. Mineração de Dados

A etapa de mineração de dados nessa atividade será restrita à chamada da função *calculaCosseno()*. Essa função é aplicada para cada item da matriz conforme os comandos a seguir:

```
for(i in 1:ncol(base)) {
     for(j in 1:ncol(base)) {
       basecosseno[i,j] <-
         calculaCosseno(as.matrix(base[i]),
         as.matrix(base[j])) }
  }
```

Após a finalização do comando anterior, cada item da matriz [i,j] conterá o valor do cosseno do ângulo. Para visualizar esse conteúdo, a instrução de código a seguir permite que o objeto *basecosseno* seja convertido novamente para um *dataframe*, facilitando a manipulação dos dados.

```
basecosseno <- as.data.frame(basecosseno)
```

1.6.6. Interpretação e Avaliação

A partir do cálculo de similaridade de cosseno realizado, o objeto *basecosseno* contém a medida de similaridade para cada banda existente na base de dados. Os valores são apresentados em um intervalo de 0 a 1, ou seja, normalizados. Quanto mais próximo a 1, o valor é considerado mais similar, e, portanto, é o mais indicado para ser recomendado ao usuário.

A verificação do conteúdo do objeto pode ser feita a partir da chamada da função *head()*. A instrução de código a seguir contém um exemplo de execução dessa função, apresentando os resultados das colunas 2 a 5.

```
head(basecosseno[,c(1,2:5)])
                 a.perfect.circle    ac.dc       adam.green
a.perfect.circle   1.00000000    0.01791723     0.05155393
abba               0.00000000    0.05227877     0.02507061
ac.dc              0.01791723    1.00000000     0.11315371
adam.green         0.05155393    0.11315371     1.00000000
aerosmith          0.06277648    0.17715300     0.05663655
afi                0.00000000    0.06789420     0.00000000
```

É possível perceber a partir dos resultados apresentados que há uma variação entre as similaridades de cada banda. E, embora o cálculo tenha sido gerado para todos os itens de dados, em um cenário de produção, é muito comum a busca pelos *top N* itens mais similares.

O exemplo a seguir apresenta uma redução da base de dados, por meio de um novo objeto, que receberá as informações dos 10 primeiros artistas mais similares. Para isso, será criado um objeto com 11 colunas, uma vez que a primeira coluna será sempre o próprio artista da recomendação.

```
basevizinhos <- matrix(NA, nrow=ncol(basecosseno),
ncol=11,dimnames=list(colnames(basecosseno)))
```

Após esta criação, será necessário ordenar o objeto *basecosseno* em ordem decrescente, para que os artistas com maior similaridade apareçam primeiro. Assim, os 11 primeiros artistas de cada item são carregados para a *basevizinhos*.

```
for(i in 1:ncol(base)) {
     basevizinhos[i,] <- (t(head(n=11,
     rownames(basecosseno[order(basecosseno[,i],
     decreasing=TRUE),][i]))));
}
```

O resultado pode ser observado por meio da seguinte chamada de função:

```
head(basevizinhos[,c(2:3)])
                 [,1]                    [,2]
a.perfect.circle "tool"                  "dredg"
abba             "madonna"               "robbie.williams"
ac.dc            "red.hot.chili.peppers" "metallica"
adam.green       "the.libertines"        "the.strokes"
aerosmith        "u2"                    "led.zeppelin"
afi              "funeral.for.a.friend"  "rise.against"
```

A saída apresentada refere-se aos 2 primeiros artistas recomendados para cada item (não foram apresentadas as 10 primeiras recomendações por limitações de espaço). Por exemplo, de acordo com a similaridade de cosseno, o usuário que gostou da banda Aerosmith, supostamente gostará também das bandas U2 e Led Zeppelin.

Após finalizar a análise, o resultado contendo as recomendações de todos os artistas pode ser armazenado em um arquivo CSV, utilizando o comando:

```
write.csv(file="top10lastfm.csv",x=basevizinhos[,-1])
```

A atividade apresentada é um pequeno exemplo de uma visão inicial dos comandos utilizados por cientistas de dados para analisar dados, com o objetivo de gerar recomendações. Em projetos de análises de dados, no entanto, conforme identificado na literatura, é comum a execução de um conjunto mais extenso de comandos e manipulações de dados, de acordo com o objetivo pretendido [Amatriain et al. 2011, Han et al. 2011, Wu et al. 2014].

1.7. Considerações Finais

A ciência dos dados e as tecnologias de Big Data estão sendo amplamente adotadas nas estratégias de negócios das empresas, em tomadas de decisões governamentais e em projetos de instituições de pesquisa. Esse minicurso apresentou uma visão inicial das atividades realizadas pelo profissional cientista de dados, demonstrando um contexto geral do processo de análise de dados, casos de uso em diferentes setores da indústria, uma discussão do perfil do cientista de dados, as características da Linguagem R no processo de análise de dados e, por fim, um exemplo prático da utilização do R em uma aplicação de recomendação de músicas.

Os atributos volume, variedade e velocidade não se enquadram mais somente no contexto de Big Data, mas sim, no ritmo em que a sociedade vive atualmente. Embora ainda incipiente, características atuais como o crescente volume de dados, a evolução de tecnologias de processamento de dados, a redução de custo de armazenamento e o avanço da inteligência artificial têm gerado oportunidades promissoras, não somente para uma área específica, mas para todas as que estão utilizando os dados como fonte de conhecimento.

Referências

[Adjekum et al. 2017] Adjekum, A., Ienca, M., and Vayena, E. (2017). What is trust? ethics and risk governance in precision medicine and predictive analytics. *Omics: a journal of integrative biology*, 21(12):704–710.

[Amatriain et al. 2011] Amatriain, X., Jaimes, A., Oliver, N., and Pujol, J. M. (2011). Data mining methods for recommender systems. In *Recommender systems handbook*, pages 39–71. Springer.

[De Mauro et al. 2016] De Mauro, A., Greco, M., Grimaldi, M., and Nobili, G. (2016). Beyond data scientists: a review of big data skills and job families. *Proceedings of IFKAD 2016 Towards a New Architecture of Knowledge: Big Data, Culture and Creativity*, pages 1844–1857.

[Edosio 2014] Edosio, U. Z. (2014). Big data analytics and its application in e-commerce. *Proceedings E-Commerce Technologies. University of Bradford*.

[Ekstrand et al. 2011] Ekstrand, M. D., Riedl, J. T., Konstan, J. A., et al. (2011). Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173.

[Feriancic et al. 2015] Feriancic, G., Celeiro, F. R., and Silva, L. N. B. (2015). Planejamento da mobilidade com big data de telefonia móvel. In *20º Congresso Brasileiro de Transporte e Trânsito, Santos*.

[Flinn 2018] Flinn, S. (2018). *Optimizing Data-to-Learning-to-Action*. Springer, Texas.

[Forte 2015] Forte, R. M. (2015). *Mastering predictive analytics with R*. Packt Publishing Ltd.

[Goldman et al. 2012] Goldman, A., Kon, F., Pereira Junior, F., Polato, I., and de Fátima Pereira, R. (2012). Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. In *Anais do XXXII Congresso da Sociedade Brasileira de Computação, XXXI Jornadas de Atualização em Informática (JAI)*, pages 88–136. Curitiba, Paraná, Brasil.

[Han et al. 2011] Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[Harris et al. 2013] Harris, H., Murphy, S., and Vaisman, M. (2013). *Analyzing the Analyzers: An Introspective Survey of Data Scientists and Their Work*. O'Reilly Media, Inc.

[Hawkins 2004] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

[He et al. 2017] He, K. Y., Ge, D., and He, M. M. (2017). Big data analytics for genomic medicine. *International journal of molecular sciences*, 18(2):412.

[Jose et al. 2017] Jose, B., Ramanan, T. R., and Kumar, S. M. (2017). Big data provenance and analytics in telecom contact centers. In *Region 10 Conference, TENCON 2017-2017 IEEE*, pages 1573–1578. IEEE.

[Kaur and Kang 2016] Kaur, M. and Kang, S. (2016). Market basket analysis: Identify the changing trends of market data using association rule mining. *Procedia computer science*, 85:78–85.

[Kazakov 2018] Kazakov, D. (2018). Product breakdown: Spotify. https://towardsdatascience.com/in-this-article-i-provide-a-detailed-analysis-of-spotify-as-a-company-music-industry-direction-eeb945d7257c. Online; Acesso em: 08 jun. 2018.

[Kumar et al. 2017] Kumar, V., Anand, A., and Song, H. (2017). Future of retailer profitability: an organizing framework. *Journal of Retailing*, 93(1):96–119.

[Marquesone 2016] Marquesone, R. (2016). *Big Data: técnicas e tecnologias para extraçao de valor dos dados*. Casa do Código, São Paulo.

[McAfee et al. 2012] McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., and Barton, D. (2012). Big data: the management revolution. *Harvard business review*, 90(10):60–68.

[Mitra et al. 2002] Mitra, S., Pal, S. K., and Mitra, P. (2002). Data mining in soft computing framework: a survey. *IEEE transactions on neural networks*, 13(1):3–14.

[Morabito 2015] Morabito, V. (2015). *Big data and analytics: Strategic and organisational impacts*. Springer.

[Patil and Davenport 2012] Patil, T. and Davenport, T. (2012). Data scientist: the sexiest job of the 21st century. *Harvard Business Review*, 90(10):70–76.

[Roberts 2016] Roberts, H. (2016). Spotify says: 'thanks 2016, it's been weird,' in its largest ad campaign yet. http://www.businessinsider.com/spotify-global-ad-campaign-signing-off-2016-2016-11. Online; Acesso em: 08 jun. 2018.

[Runkler 2016] Runkler, T. A. (2016). *Data analytics: models and algorithms for intelligent data analysis*. Springer.

[Sarwar et al. 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

[Silver et al. 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

[Spector 2008] Spector, P. (2008). *Data manipulation with R*. Springer Science & Business Media.

[Sundsøy et al. 2014] Sundsøy, P., Bjelland, J., Iqbal, A. M., de Montjoye, Y.-A., et al. (2014). Big data-driven marketing: how machine learning outperforms marketers' gut-feeling. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 367–374. Springer.

[Velleman and Hoaglin 1981] Velleman, P. F. and Hoaglin, D. C. (1981). *Applications, basics, and computing of exploratory data analysis*. Duxbury Press.

[Wu et al. 2008] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37.

[Wu et al. 2014] Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2014). Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107.

[Xu and Wunsch 2005] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

[Yazti and Krishnaswamy 2014] Yazti, D. Z. and Krishnaswamy, S. (2014). Mobile big data analytics: research, practice, and opportunities. In *Mobile Data Management (MDM), 2014 IEEE 15th International Conference on*, volume 1, pages 1–2. IEEE.

12. Capítulo 12

Autores:

Leandro A. Silva
Universidade Presbiteriana Mackenzie
email: leandroaugusto.silva@mackenzie.br

Diego C. D. Nogare
Universidade Presbiteriana Mackenzie
email: nogare@ngrsolutions.com.br

Ismar F. Silveira
Universidade Presbiteriana Mackenzie
email: ismarfrango@gmail.com

Capítulo 12

Fundamentos de Big Data com Apache Spark

Leandro A. Silva, Diego C. D. Nogare, Ismar F. Silveira

Abstract

This work presents Apache Spark framework as a set of software packages, with characteristics of parallel processing and memory, to use it in a process of large volume of data. For the presentation of the framework and contextualization of the main fundamentals of Big Data, a case involving the monitoring of an equipment was presented. From this case we present the Big Data architecture used in this type of application and ways to explore data in a type of predictive analysis of equipment.

Resumo

Este trabalho se propõe a apresentar o framework Apache Spark como sendo um conjunto de pacotes de software, com características de processamento paralelo e em memória, para auxiliar no processamento de grande volume de dados. Para a apresentação do framework e contextualização dos principiais fundamentos de Big Data, um case envolvendo o monitoramento de um equipamento foi apresentado. A partir deste case apresentam-se a arquitetura Big Data utilizada neste tipo de aplicação e formas de explorar dados em um tipo de análise preditiva de falha de equipamento.

1.1. Introdução

Atualmente, muito se fala da importância dos dados, a ponto de metaforicamente considera-los o “petróleo” ou a “energia” do futuro, vindo a provocar uma nova revolução industrial [Van der Aalst, 2014]. Isso deve-se, em parte, à grande quantidade de dados gerados atualmente e disponíveis para análise, aliado ao poder computacional existente nos dias de hoje.

Trabalhos embrionários em pesquisa e aplicação de análise de dados surgiram originalmente com o nome de Inteligência de Negócios ou BI (do inglês, *Business Intelligence*) [Chen, Chiang e Storey, 2012]. Em BI, as empresam começaram a dar atenção à integração de diferentes fontes de dados em um tipo de modelagem com foco único na geração de insumos para processos de tomada de decisão. Conforme o volume de dados foi aumentando, em velocidade tal que tornava a análise por um humano inviável, o termo BI foi dando lugar para o que se convencionou chamar de *Big Data* [Chen, Mao e Liu, 2014]. Interessante notar que no Brasil ainda não há um consenso na tradução – não é incomum encontrar o termo "Dados Massivos", entretanto, permanece o termo em inglês sendo ainda o mais comumente utilizado.

Para ilustrar essa evolução terminológica dentro de um contexto temporal, em uma análise ao *Trends* do Google[1], acompanhando a frequência de busca dos termos BI e Big Data neste buscador desde 2004 até meados de 2018 no Brasil, é visível que há queda pela procura do BI e aumento de interesse pelo termo Big Data. Esta análise está ilustrada na Figura 1, vê-se que a partir de aproximadamente Maio/2013 o termo BI deixa de ser o mais buscado, empatando com o Big Data, sendo que esse último começa a crescer e despontar no final de 2012.

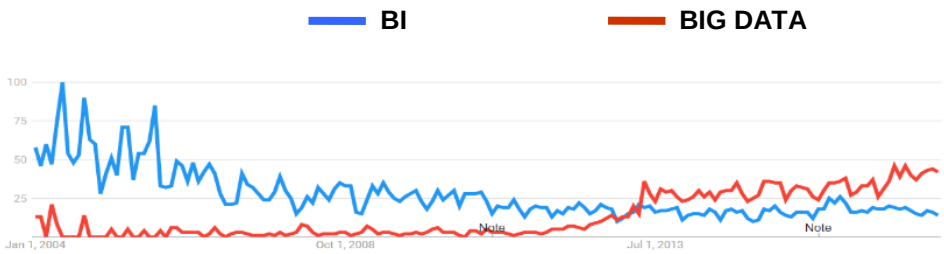


Figura 1 - Comparação de termos no Brasil sobre Business Intelligence e Big Data. Fonte: Google Trends

Complementarmente à Figura 1, é interessante verificar as tendências de tecnologia apresentadas pelo Gartner Group, que é uma empresa mundialmente conhecida por apresentar as tendências em tecnologia a partir de um gráfico chamado *Hype Cycle*. Na Figura 2 apresenta-se um exemplo deste gráfico, por meio do qual pode-se verificar como uma tecnologia emergente pode ser analisada em termos de tempo (*time*) por expectativa (*expectations*), dimensões representadas pelos eixos X e Y respectivamente. O caminho natural para uma tecnologia se consolidar no mercado deve iniciar com um disparo de inovação (*innovation trigger*), atingir o pico de euforia (*peak of inflated expectations*) e ir decaindo, passando pelo vale da desilusão (*trough disilusionment*), vencendo o aclive do esclarecimento (*slope of enlightenment*) até atingir o platô de produtividade (*plateau of productivity*).

Para exemplificar a aplicação do Hype Cycle, em 2014 a tecnologia emergente Big Data está no limiar entre o pico de euforia e o vale da desilusão, como visto na Figura 3. Em 2015 o Gartner retirou Big Data do gráfico. A retirada do gráfico de tecnologias emergentes foi por entender que o Big Data passou por todas as fases

[1] http://trends.google.com

previstas e não faria mais sentido manter no gráfico. Acompanhe na Figura 3 o ponto onde está Big Data no Hype Cycle de 2014 [Gartner, 2014].

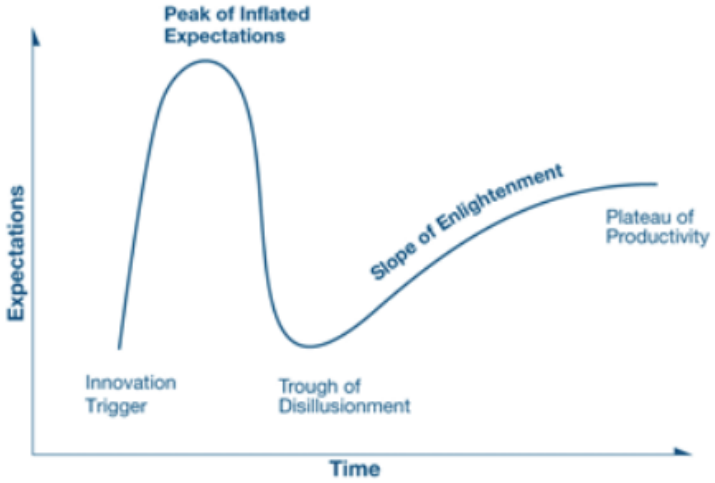


Figura 2 - Forma de analisar o Hype Cycle. Fonte: Gartner Group

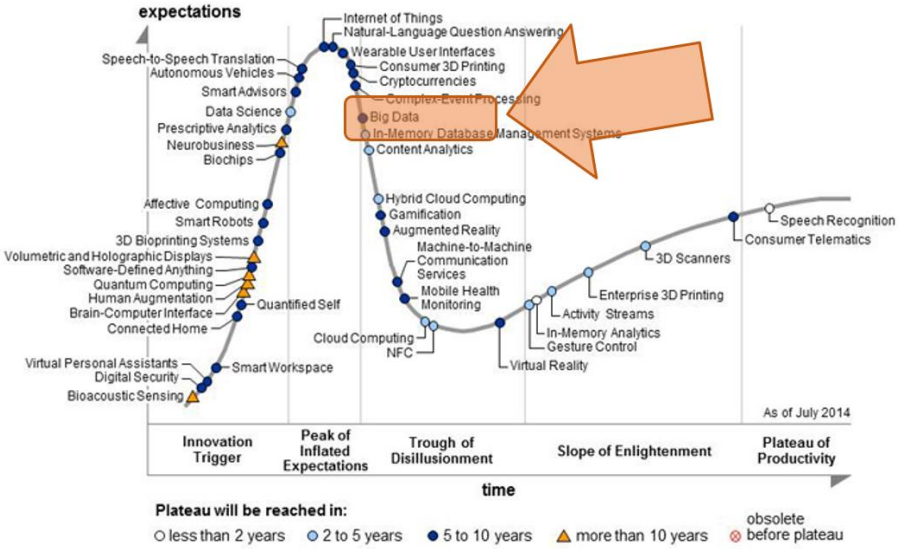


Figura 3 - Hype Cycle do Gartner no último ano que Big Data fez parte do gráfico. Fonte: [Gartner, 2014].

Por fim, outro aspecto importante a motivar sobre Big Data são os dados de buscas das Habilidades "Quentes" (*Hot Skills* em inglês), ou seja, dos empregos mais buscados e as habilidades mais solicitadas em 2015 para o ano de 2016 no Brasil, dentro do LinkedIn (2015). É possível acompanhar na Figura 4 que as habilidades envolvidas em profissionais que atuam em Big Data ocupam não apenas a primeira posição no ranking, mas também a quarta, sétima e décima quarta posição.

Figura 4 - Principais habilidades de 2015 para contratações em 2016 – Estudo Brasil. Fonte: Linkedin

Dado o exposto, é possível notar a relevância de Big Data e a ampla demanda de utilização. Essa necessidade de mercado implica em capacitação profissional e, consequentemente, pela produção de material didático que apresente os principais fundamentos deste assunto. É nesse contexto que este trabalho se apresenta, fornecendo os conceitos principais e ilustrando-os por meio de uma aplicação envolvendo a análise dos dados advindos do monitoramento de um equipamento com a finalidade de observar o momento de falha.

Além da introdução, o trabalho está estruturado da seguinte maneira: inicia-se com a discussão dos conceitos básicos, partindo em seguida para a apresentação do *case* e metodologia envolvida. Seguem os resultados do *case* analisados com uso da Linguagem R. Por fim, algumas conclusões são apresentadas.

1.2. Conceitos Fundamentais de Big Data

Big Data é fundamentado em três pilares principais, conhecidos como os 3 "Vs": Volume, Velocidade e a Variedade [Laney, 2001], [Zikopoulos et al., 2011], [Chen, Mao e Liu, 2014]. Existem estudos em algumas outras literaturas colocando a questão com diversos outros Vs, mas não serão discutidos nesta publicação [Chen, Mao e Liu, 2014]. Para entender os 3 Vs de Big Data, é necessário olhar isoladamente cada um deles, porém, de um ponto de vista pragmático, todos devem ser contemplados.

A velocidade de Big Data está relacionada principalmente ao desempenho de processamento de grandes volumes de dados. Neste contexto, é comum separar os diferentes tipos de processamento: *Batch* (Lote), Periódico, Próximo ao Tempo Real e por fim, em Tempo Real [Kiran et al., 2015].

No caso da variedade, ao observar um banco de dados tradicional, ou uma planilha convencional, sabe-se de antemão sua estrutura, no que se refere aos tipos de dados existente em cada uma das colunas. A mudança de diversos paradigmas tecnológicos dos últimos anos faz com que as estruturas devam ser mais versáteis, chegando a centenas de tipos diferentes [Chen, Mao e Liu, 2014]. Exemplos típicos de variedade de tipos são: textos, fotos, áudios, vídeos, dados de GPS, dados de sensores, documentos, SMS, dados de redes sociais, etc., além dos tipos de dados convencionais.

O volume de dados disponíveis para se trabalhar tem aumentado em uma taxa crescente. Como exemplo, considere que um único pixel em uma imagem preto-e-branco pode ser representada por apenas 1 bit. Quando se deseja armazenar uma imagem não muito grande, o volume passa a alguns kilobytes. Para o caso de uma música, o tamanho passa para alguns megabytes. Já um filme completo necessita de alguns gigabytes. Outro aspecto relacionado ao volume são as fontes de dados [Lynch, 2008]. Antes, tinha-se apenas funcionários e sistemas internos gerando dados para a empresa. Nos dias atuais pode-se pensar que outras fontes, inclusive externas à empresa em questão, são potenciais fontes de dados.

Para suportar a demanda de armazenamento e consulta de grande volume de dados, com ampla variedade, proporcionando velocidade adequada em seu processamento, existem atualmente diversas plataformas de *software*, algumas das quais são descritas a seguir.

1.2.1 Hadoop

Hadoop é uma plataforma de software distribuído, construído em Java e gerenciado atualmente pela Apache Foundation para processamento de grandes massas de dados [Zikopoulos e Eaton, 2011], [White, 2012], [Prajapati, 2013]. Hadoop oferece recursos para o armazenamento e processamento de dados, sendo o armazenamento feito de forma distribuída, usando o padrão *Hadoop Distributed File Systems* (HDFS) e o processamento dos dados distribuídos pelo processo de MapReduce.

O HDFS é baseado no processo de armazenamento da Google, chamado *Google File System* [Chang et al., 2008]. Trata-se de um processo que permite armazenar grande volume de dados de forma distribuída e redundante.

O MapReduce é um paradigma de desenvolvimento desenvolvido para trabalhar em grande escala de dados de forma paralela, dividindo as atividades em tarefas independentes. Este framework foi introduzido pelo Google para suportar a arquitetura MPP (*Massively Pparallel Processing*) em conjunto de soluções computacionais [Dean e Ghemawat, 2008]. O propósito deste framework surgiu para atender um cenário no qual os dados podem estar divididos em diversos servidores, para melhorar performance e escalabilidade para o sistema.

O processo realizado pelo MapReduce é dividido em duas partes: Map e Reduce. O processo de Map faz com que a requisição feita pelo solicitante seja enviada para o nó do cluster que possui aquele dado. Já o processo de Reduce faz o papel de consolidar os resultados processados em diversos nós do cluster e entregar uma saída única. Acompanhe na Figura 5 a arquitetura MapReduce.

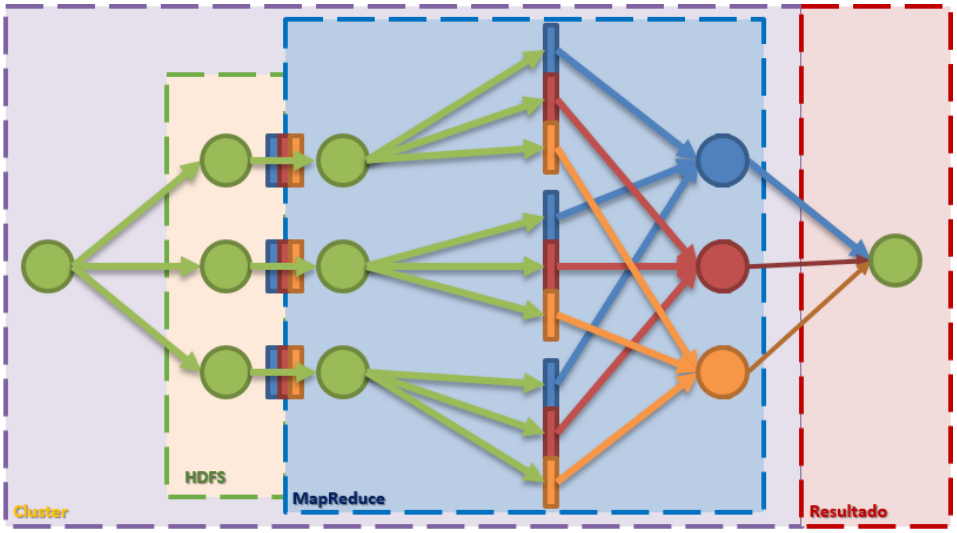


Figura 5 - Visão macro do processo de MapReduce. Fonte: Autores

Para explicar o funcionamento de MapReduce, imagine a situação hipotética de um programa que contabiliza a frequência de ocorrência de termos em determinadas sentenças. O processo se inicia com o cluster (no caso, de Hadoop) definindo quem é o nó Principal (do inglês *Master*) e quem são os nós Trabalhadores (do inglês, *Workers*). O nó principal fará a definição dos trabalhadores de quem terá o papel de Map e quem terá o papel de Reduce. Os nós que foram assinados como Map, recebem os dados que devem ser processados. E os nós assinados como Reduce, receberão os dados processados pelo Map. O processamento inicia, e, o processo do Map começa a contabilizar a frequência da ocorrência do termo e faz a atribuição do valor 1 cada vez que o termo é encontrado. Ao término do seu processamento, o Map pode armazenar em disco no HDFS o seu resultado, para então, notificar o nó principal. O nó principal, por sua vez aciona os nós de Reduce para acessar aquele endereço do HDFS que foi salvo o processamento do Map, para que execute o processo de sumarizar as frequências de cada termo. Ao término de processamento de todos os nós de Map e de Reduce, o principal devolve ao usuário o resultado único.

Ainda que MapReduce seja o núcleo primário de todo o processo, para cada necessidade de ação específica deve haver uma integração com ferramentas auxiliares. Estas ferramentas compõem o ecossistema do Hadoop, permitindo que as tarefas necessárias para um projeto completo sejam realizadas de forma modular.

Em suma, Hadoop é uma plataforma de computação distribuída voltada para clusters horizontais, permitindo o processamento de grandes volumes de dados. Um dos principais destaques deste tipo de tecnologia é a premissa de tolerância a falhas no hardware. Para dentro do Hadoop é comum o desenvolvimento utilizando processos de MapReduce e HDFS. Porém não é apenas com estas duas frentes que é composto o Hadoop. O ambiente, como um todo, é composto pelos seguintes processos integrados, sendo:

- **Hadoop Common** – Sistema de bibliotecas e arquivos necessários para executar todos os processos internos do Hadoop.
- **HDFS - Hadoop Distributed File System** - Sistema de arquivos distribuído de armazenamento físico de dados. O HDFS garante o armazenamento físico dos dados no cluster.
- **Hadoop Yarn** – É a plataforma de gerenciamento de recursos. O Yarn é responsável pelo gerenciamento dos recursos computacionais existente nos servidores apresentados ao cluster, também pelo agendamento e requisições dos recursos.

O Hadoop está presente em grandes empresas como Adobe, eBay, Facebook, Google, LinkedIn, Spotify, e muitas outras [Apache, 2018].

1.2.2 Distribuições Hadoop

Hadoop possui uma série de distribuições que estendem o núcleo primário, aumentando funcionalidades e facilidades de implementações. As mais conhecidas são:

- **Cloudera:** Sua distribuição Hadoop mantém diversos aspectos do projeto original hospedado no Apache Foundation. Ela agrega um grande número de melhorias, entre elas uma ferramenta de gestão e monitoramento chamada *Cloudera Manager* e também o *Cloudera Impala*, um motor SQL que trabalha com dados relacionais no Hadoop.
- **Hortonworks:** Mantém uma distribuição de Hadoop mais próxima do original, talvez até mais do que qualquer outra distribuição. Um dos pontos que levou a Hortonworks a seguir esta abordagem é para construir o ecossistema e manter a evolução dos códigos *open source*, focando em beneficiar os usuários evitando que fiquem dependentes de apenas um fornecedor. A abordagem comercial da Hortonworks permite que, caso um cliente tenha interesse em sair da sua carteira de clientes, conseguirá levar seu ambiente para outros lugares, uma vez que está mais próximo do Hadoop original.
- **IBM:** Em 2013 a IBM comprou a empresa de computação em nuvem chamada Softlayer. O foco foi fazer frente a outros fornecedores de serviço que atendiam em escalas menores, algo que a IBM não se propunha a fazer por ser reconhecida mundialmente nas entregas de grandes projetos, principalmente para grandes empresas. Com este objetivo, a IBM tornou-se um grande player no mundo Hadoop. Em 2014, a IBM possuía mais de 100 instalações de Hadoop, e muitos de seus clientes trabalhando na casa de petabytes de dados.
- **Microsoft:** A Microsoft pôde desenvolver uma distribuição Hadoop que atende a uma grande faixa de profissionais de infraestrutura, conquistando isso ao habilitar o Hadoop para rodar em Windows. Porém, o uso do Linux segue possível, por meio da Azure, a plataforma de computação em nuvem da Microsoft, que possui uma grande quantidade de distribuições Linux para uso. Dentro do Azure, existe um produto chamado HDInsight. Trata-se de uma oferta de Hadoop como serviço baseada na distribuição da Hortonworks, adaptada exclusivamente para rodar no Azure. Ao provisionar o serviço do Hadoop, o ambiente já é criado com diversas soluções do ecossistema Hadoop, como Hive, Sqoop, Mahaout, entre outros.

1.2.3 Ferramentas auxiliares

O ecossistema possui diversas funcionalidades integrando o núcleo do Hadoop. Estas ferramentas auxiliares possibilitam que o trabalho seja realizado de ponta a ponta, permitindo que tarefas complexas sejam desenvolvidas por agentes especializados. Veja na Figura 6 alguns dos principais produtos integradores ao Hadoop.

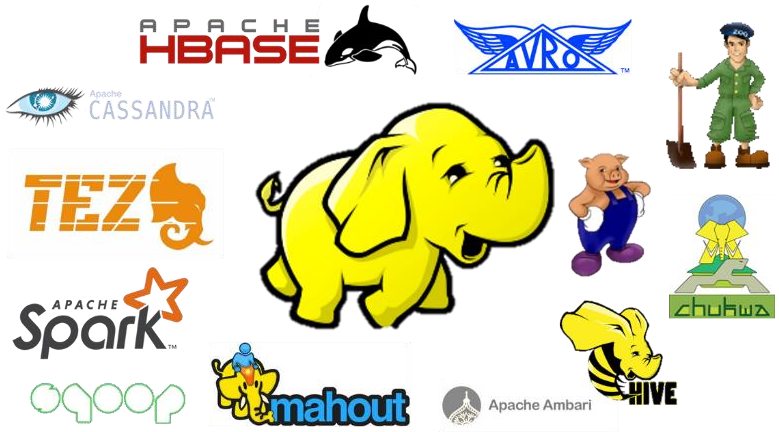


Figura 6 - Algumas ferramentas que se integram ao ecossistema Hadoop. Fonte: Autores

Como cada ferramenta destas possui um objetivo específico, é interessante conhecer, de modo geral, o que elas podem entregar, e se aprofundar nos estudos relativos à sua necessidade real. A seguir, são mencionadas as principais ferramentas que o mercado utiliza para resolver os problemas mais comuns em projetos utilizando Hadoop. É importante lembrar que cada ferramenta, por ter um objetivo específico, possui sua própria linguagem de programação (quando aplicável) e forma de trabalho.

- **Hbase:** É um banco de dados colunar e distribuído, escrito em Java e desenhado a partir do Google BigTable. Integra-se de forma simples e rápida com o Hadoop, podendo utilizar MapReduce para distribuição do processamento [George, 2011].
- **Sqoop:** É a ferramenta para trabalhar com movimentação de dados entre o Hadoop e uma base de dados SQL-like. Pode-se seguir nos conceitos de cargas incrementais, permitindo o acréscimo de dados novos nas tabelas de destino. Normalmente é integrado com o Hive ou HBase, do lado do Hadoop. [Ting e Cecho, 2013].
- **Mahout:** O processo de Aprendizagem de Máquina em um cenário com integração ao Hadoop pode ser feito através do Mahout. Diversos algoritmos de filtragem colaborativa, classificação, agrupamentos (*clustering*) e recomendação já foram implementados, e estão disponíveis para alavancar o reconhecimento de padrões nos projetos [Withanawasam, 2015].
- **Hive:** É uma abordagem de *Data Warehouse* para o Hadoop, permitindo que o mecanismo de MapReduce faça as vezes de processamento distribuído, já contando com processos de *Data Summarization*, Consultas e *Data Analysis*. É bem simples de integrar pois possui uma interface similar à de padrões SQL. A linguagem por trás é o HQL – HiveQL, que é bastante semelhante à MySQL em questão de sintaxe [Thusoo et al., 2010].

- **Pig:** É uma linguagem de programação de alto nível para o Hadoop. Sua sintaxe é o Pig Latin, que é procedural, e é possível abstrair programação avançada em Java para MapReduce usando esta ferramenta. Normalmente os programas escritos em Pig são focados para resolver problemas de SGBDR – Sistemas Gerenciadores de Bancos de Dados Relacionais. O Pig já possui em seu núcleo processos para realizar movimentações de dados (processo conhecido como ETL) e planos de execuções declarados [Gates e Dai, 2016].
- **Avro:** É um sistema de serialização de dados, que usa o formato JSON - *JavaScript Object Notation* – para definição de seus tipos de dados e protocolos. É comum o Avro ser uma fonte de dados para o Hadoop e para o Spark. Os arquivos são compostos por um cabeçalho e blocos de dados. O cabeçalho possui definição de esquema e os blocos de códigos possuem a serialização binária ou o JSON. Dependendo da aplicabilidade, usa-se o formato binário ou JSON - binário é mais usado para movimentação entre aplicativos e JSON para soluções web [White, 2012].
- **Zookeeper:** É uma ferramenta que ajuda na organização de distribuição de recursos do projeto. Pensando em escalabilidade, quando um projeto precisa utilizar diversos servidores em seu parque tecnológico, a chance de problemas acontecerem é quase que inevitável. Com o uso do Zookeeper, que é um centralizador de serviços para manutenção e configuração, o gerenciamento das complexidades existentes no Deploy distribuído do projeto diminui significantemente [White, 2012].

1.2.4. Spark

Apesar de fazer parte do ecossistema Hadoop, o Spark permite uma aceleração do processamento por usar recursos de paralelismo implícito de dados e também processamento em memória ao invés de usar exclusivamente disco.

O Spark foi desenvolvido na Universidade de Berkeley, na Califórnia, em 2009, dentro do AMPLab (AMP é um acrônimo para *Algorithms, Machines and People* – Algoritmos, Máquinas e Pessoas) como um projeto de doutorado, e doado para a fundação Apache em 2013 [Zaharia et al., 2010].

O RDD (do inglês *Resilient Distributed Dataset*) foi criado em 2012 com o objetivo de transpassar as limitações que o MapReduce apresentada à época [Zaharia et al., 2012]. A diferença principal entre o MapReduce e o Spark é que o segundo trabalha com um foco de programa distribuído que oferece uma forma (deliberadamente) restrita de memória compartilhada e distribuída. Este processo paralelo é mais rápido que o processo linear, apresentado no MapReduce, e reduz a latência. A Figura 7 apresenta uma comparação de tempo de execução de uma regressão logística utilizando Hadoop e Spark, e é possível acompanhar que o resultado do Spark é superior a 100 vezes comparado com Hadoop.

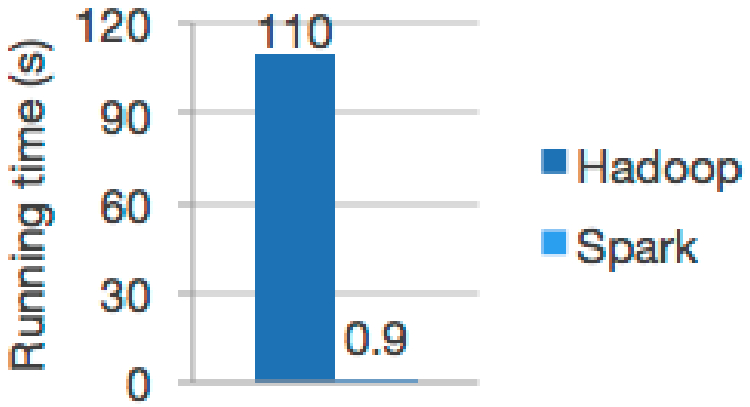


Figura 7 - Comparação de tempo para executar uma Regressão Logística em Spark e Hadoop. Fonte: Apache Spark

Spark Core é o núcleo desta plataforma [Spark, 2018], que fornece a distribuição de tarefas e permite o agendamento de I/O através de APIs. Estas tarefas distribuem as operações no RDD do Spark que faz o agendamento e distribuição das atividades em paralelo no cluster. Uma das linguagens mais comuns de se encontrar no mercado quando está trabalhando com Spark é o Scala. Uma das integrações que vem ganhando bastante atenção é o Sparklyr[2], que permite interagir com o Spark por meio da linguagem de programação estatística R.

Outro componente interno do Spark é o Spark SQL [Spark, 2018], que ajuda a abstrair problemas de âmbito dos dados. É possível trabalhar no Spark SQL tanto com dados estruturados quanto com semi-estruturados. O suporte à linguagem SQL é nativo, mas também é possível manipular os dados com Scala, Java, Python ou R. As integrações com ferramentas externas podem ser feitas utilizando alguns conectores bastante conhecidos, como ODBC ou JDBC.

O segundo pilar do núcleo base é o Spark Streaming [Spark, 2018], que aproveita a velocidade do agendamento de tarefas do Spark Core para executar as rotinas de fluxo da transmissão de dados. O padrão é uma execução de ingestão em mini-lotes e transformações RDDs. O processo de streaming do Spark pode ser executado em paralelo à outras soluções como Kafka ou Flink.

Um outro grande benefício de se trabalhar com este framework é o Mllib – *Machine Learning Library* [Spark, 2018]. Esta estrutura de aprendizagem de máquinas é muito mais rápida na hora de processar e criar modelos preditivos que o Mahout. Estudos apontam para uma taxa de até 100 vezes mais rápido. Algoritmos de classificação e regressão como Support Vector Machine, Naïve Bayes, Regressão Logística, Regressão Linear, Árvores de Decisão são alguns dos algoritmos possíveis para se trabalhar usando Spark MLlib.

Por fim, o GraphX [Spark, 2018] completa os pilares do Spark permitindo que a ferramenta tenha uma abordagem direcionada a grafos. Um ponto de atenção é sobre o funcionamento, pois o GraphX roda em cima dos RDDs, e como os RDDs são imutáveis, o grafo também não pode sofrer mudanças.

[2] https://spark.rstudio.com/

Veja como é a proposição da arquitetura apresentada pelo Spark (2018) na Figura 8.

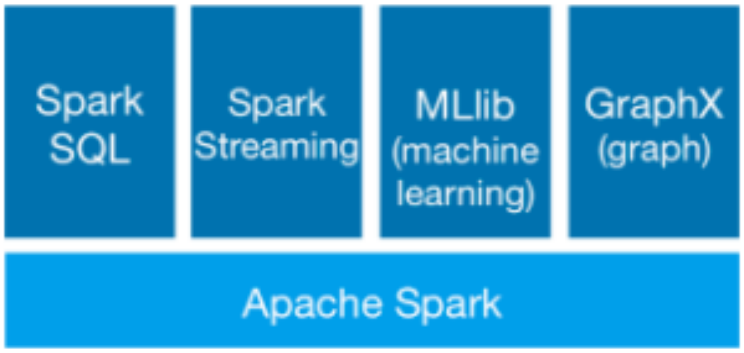


Figura 8 - Arquitetura do Apache Spark. Fonte: Apache Spark

1.3. Estudo Dirigido Prático e Metodologia

Com objetivo de introduzir os fundamentos de Big Data com Apache Spark, um estudo prático e controlado será conduzido através do Sparklyr (2018). Neste estudo será simulado um cenário, com dados reais extraídos de uma linha de produção, geralmente encontrado em fábricas, em que a temperatura tem relação com o funcionamento das máquinas. Para tanto, considera-se que há sensores de temperatura monitorando uma máquina da linha de produção. Por motivos de sigilos assinados através de Acordo de Confidencialidade (do inglês *NDA – Non-Disclosure Agreement*) não pode ser revelada sua origem.

Para este cenário considerou-se a arquitetura de referência de Big Data ilustrada na Figura 9. Esta arquitetura consiste na primeira camada de um conjunto de 2500 sensores de temperaturas e módulos para a coleta dos dados. A cada período de tempo, os dados coletados são encaminhados para a camada de barramento. Este barramento é responsável por garantir que apenas as mensagens autorizadas que chegam é que devem ser encaminhadas para as camadas seguintes. Em seguida, o serviço de *streaming analytics* recebe as mensagens autorizadas (dados) e as processam, direcionando as saídas para dois destinos estabelecidos. Esta arquitetura de referência considera o envio dos dados para uma ferramenta de visualização de dados em tempo real (*online*) e, também, para um repositório de persistência, neste caso será considerado um tipo de banco de dados NoSQL (MongoDB). Após realizar a persistência da temperatura medida pela máquina, o Spark é acionado para que faça os processamentos necessários em lote (*batch*), para ser apresentado também na ferramenta de visualização de dados.

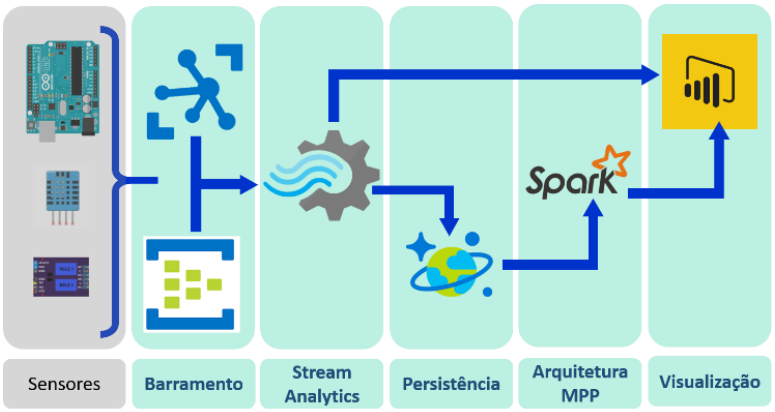


Figura 9 - Arquitetura de referência para o processamento em tempo real. Fonte: Autores

Para este estudo dirigido, será considerado que as medidas de temperatura já foram coletadas e enviadas ao banco de dados. Ou seja, será usado uma amostra que contém dados reais exportados de apenas uma semana de coleta. Neste conjunto de dados a ser exemplificado, apenas dois valores serão considerados, a temperatura e o estado de funcionamento da máquina. Este último é um dado externo à arquitetura, geralmente incorporada a partir de relatórios mantidos pela a equipe de manutenção e operação da fábrica.

Portanto, este estudo dirigido pretende permitir o entendimento do uso do Spark como ferramenta de processamento e de R como ferramenta de exploração e manipulação de dados.

É esperado que ao término do estudo dirigido, seja possível responder:

- Quais horários em que a temperatura da fábrica é mais alta;
- Quais temperaturas normalmente levam a máquina parar de funcionar;
- Estimativa de temperatura de parada da máquina.

As duas primeiras perguntas são possíveis de se responder apenas com Análise Exploratória dos dados, e estão respondidas de forma direta. A terceira pergunta será respondida por meio de Análise Preditiva, que usará Regressão Logística, e sua explicação será mais completa e detalhada.

1.4. Resultados Experimentais

As análises dos desafios propostos no estudo dirigido permitem que os resultados sejam encontrados, e que as respostas para os problemas apresentados tenham desempenho em tempo de resposta perceptível, o que é objetivo do Sparklyr. Para o desenvolvimento utilizou-se o ambiente RStudio, instalado em uma máquina virtual com Sistema Operacional Windows 10 e a versão 2.2.0 do Spark.

Inicialmente é preciso instalar e carregar o pacote Sparklyr. Após o carregamento do pacote, em todos os projetos, deve-se criar uma conexão com Spark que pode ser feito com o seguinte comando:

```
devtools::install_github("rstudio/sparklyr")
library(sparklyr)

sc <- spark_connect(master = "local")
```

Após isto, carregam-se os dados exportados, que serão usados neste estudo em um objeto que será a fonte de dados. Para garantir o funcionamento esperado, é realizado o tratamento para converter os dados nos tipos apropriados. Pode-se carregar e limpar os dados da seguinte maneira (considerando os dados em um arquivo `conjuntoDados.csv` na pasta `Users\UserSpark\Desktop\Temperatura` na raiz do disco C):

```
setwd("C:\\Users\\UserSpark\\Desktop\\Temperatura\\")
dados <- read.csv("conjuntoDados.csv", header=F)

colnames(dados) <- c('id','dataHora','Temperatura','Sinais','Desligou')
dados$id <- NULL
dados$data <- as.Date( dados$dataHora)
dados$dataHora <- as.POSIXct(strptime(as.character(dados$dataHora), "%Y-%m-%d
%H:%M:%S"))
datas <- as.vector( as.character( unique(dados$data) ) )
```

Para melhorar a apresentação visual dos elementos, com cores em tom pastel, pode-se usar o pacote RColorBrewer. O código a seguir instala, carrega e cria um vetor de cores a partir das paletas pré-estabelecidas:

```
install.packages(RColorBrewer)
library(RColorBrewer)

cores <- brewer.pal(7,"Dark2")
```

Para fins didáticos, o conjunto de dados total será segmentado em dias, um para cada dia existente na base de dados original. A segmentação criará sete novos conjuntos de dados armazenado em forma de fonte de dados. Esta segmentação de dados pode ser feita seguindo o código:

```
dia1 <- dados[(dados$data == datas[1]), c("dataHora","Temperatura") ]
dia2 <- dados[(dados$data == datas[2]), c("dataHora","Temperatura") ]
dia3 <- dados[(dados$data == datas[3]), c("dataHora","Temperatura") ]
dia4 <- dados[(dados$data == datas[4]), c("dataHora","Temperatura") ]
dia5 <- dados[(dados$data == datas[5]), c("dataHora","Temperatura") ]
dia6 <- dados[(dados$data == datas[6]), c("dataHora","Temperatura") ]
dia7 <- dados[(dados$data == datas[7]), c("dataHora","Temperatura") ]
```

Com os dados segmentados por dia, é possível gerar gráficos que apresentam o comportamento de horário e temperatura. O código para gerar este gráfico é o seguinte:

```
par(mfrow=c(3,3))
plot( dia1, col=cores[1], main=datas[1] )
plot( dia2, col=cores[2], main=datas[2] )
plot( dia3, col=cores[3], main=datas[3] )
plot( dia4, col=cores[4], main=datas[4] )
plot( dia5, col=cores[5], main=datas[5] )
plot( dia6, col=cores[6], main=datas[6] )
plot( dia7, col=cores[7], main=datas[7] )
```

E a saída visual gerada pode ser acompanhada na Figura 10.

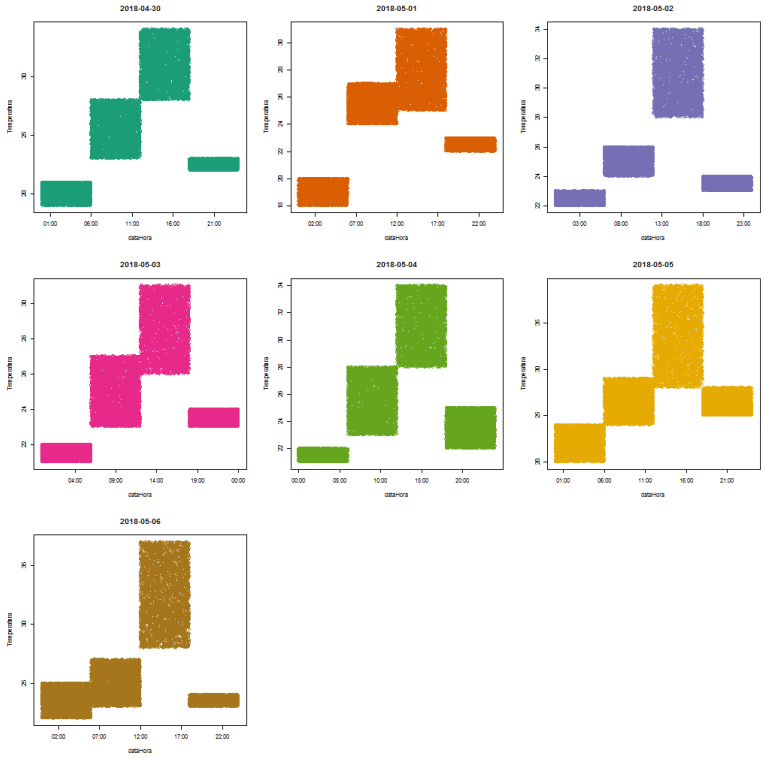


Figura 10 – Apresentação visual do período do dia (eixo X) e as temperaturas coletadas (eixo Y)

O primeiro desafio proposto refere-se a encontrar os horários que a temperatura da fábrica é mais alta. Por meio da observação do gráfico apresentado na Figura 10, esta pergunta pode ser respondida diretamente. O segundo desafio era descobrir quais temperaturas levam a máquina a parar de funcionar. Para responder esta questão, pode-se segmentar os dados entre os subconjuntos que apresentam as temperaturas em conjunto com os dados informativos que informam se a máquina desligou ou não. Uma tabela foi gerada com as combinações de funcionamento e temperatura. Acompanhe o código a seguir para reproduzir este experimento:

```
funcionou <- dados[(dados$Desligou == 0),]
desligou <- dados[(dados$Desligou == 1),]

menorQuebra <- min( dados[ (dados$Desligou == 1)  , c('Temperatura') ] )
maiorQuebra <- max( dados[ (dados$Desligou == 1)  , c('Temperatura') ] )
menorLigacao <- min( dados[ (dados$Desligou == 0)  , c('Temperatura') ] )
maiorLigacao <- max( dados[ (dados$Desligou == 0)  , c('Temperatura') ] )

tabela <- as.data.frame(matrix( c('Menor não Quebra','Maior não Quebra','Menor
Quebra','Maior Quebra'
          ,menorLigacao,maiorLigacao,menorQuebra,maiorQuebra)
          , nrow = 2, ncol = 4, byrow = T))
```

A saída da tabela é como a seguir:

Menor não Quebra	Maior não Quebra	Menor Quebra	Maior Quebra
18	32.99	29	39

Para gerar uma representação visual que ajude no suporte, pode-se criar dois histogramas, um para o conjunto de dados no qual a máquina não desligou e outro para qual a máquina desligou. O código para geração dos histogramas é apresentado a seguir:

```
hist(funcionou$Temperatura, col = cores[1], main='Temperatura que não quebra')
hist(desligou$Temperatura, col = cores[2], main='Temperatura que quebra')
```

E suas propriedades visuais podem ser acompanhadas nas Figuras 11 e 12, sendo que a Figura 11 apresenta a distribuição da frequência de temperaturas na qual a máquina não parou de funcionar, e a Figura 12 apresenta os dados relativos às temperaturas para as quais a máquina parou.

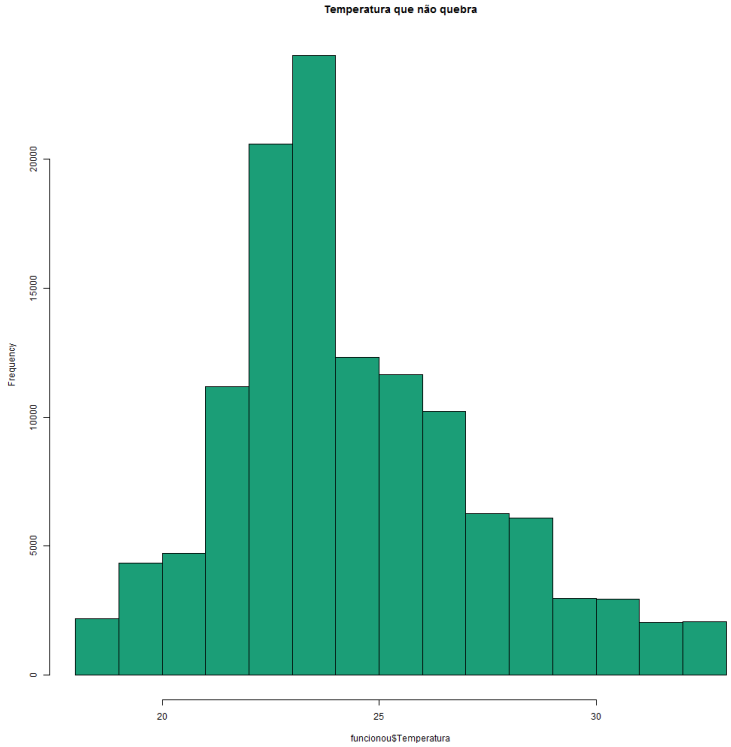


Figura 11 – Histograma de temperaturas na qual a máquina não parou de funcionar

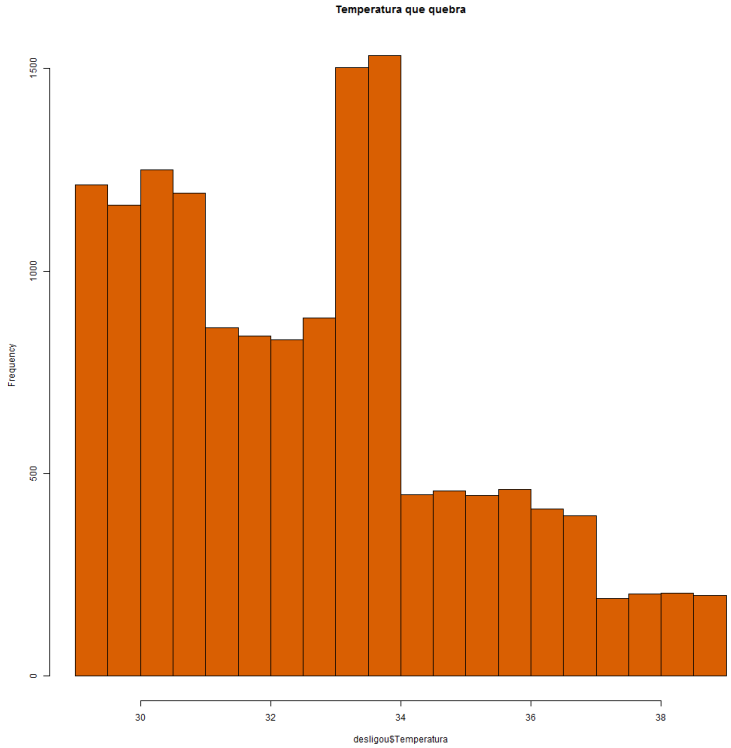


Figura 12 – Histograma de temperaturas na qual a máquina parou de funcionar

Com isso é possível responder a segunda questão proposta no desafio.

Partindo para a solução do terceiro desafio, que é estimar a chance de parada da máquina a partir de uma temperatura, é necessário criar um modelo preditivo que responda esta questão. Como os dados estão distribuídos entre 0 e 1 na variável alvo do nosso estudo com explicação a partir da variável temperatura. Como existe uma sobreposição de valores explicativos para chegar à resposta, um modelo que se adapte a este intervalo e crie uma função que melhor explica a saída é uma regressão logística, que cria uma curva sigmoide. Acompanhe o código a seguir, como estes dados estão distribuídos:

```
plot(dados$Temperatura, dados$Desligou, col=cores[dados$Desligou+1], pch=19)

plot(dados$Temperatura, jitter(dados$Desligou), col=cores[dados$Desligou+1],
pch=19)
```

Repare que existem duas plotagens com os dados, a Figura 13 sendo o eixo Y com dados apenas nas linhas de 0 e 1, e a Figura 14 com dados espalhados em uma variação de 0.2 para mais e para menos também no eixo Y. Esta distribuição é apenas visual, para ajudar no entendimento do processo.

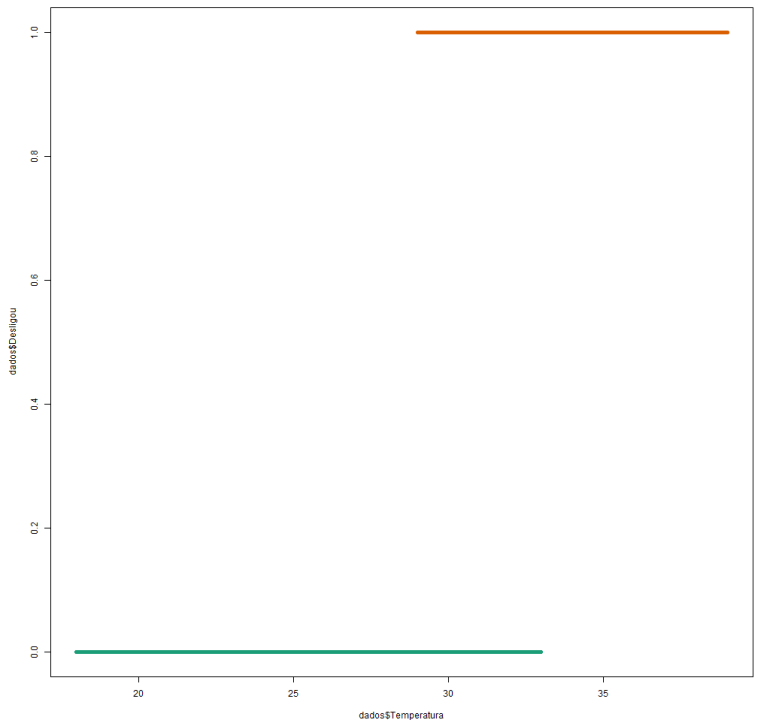


Figura 13 – Representação visual entre a temperatura e o funcionamento da máquina

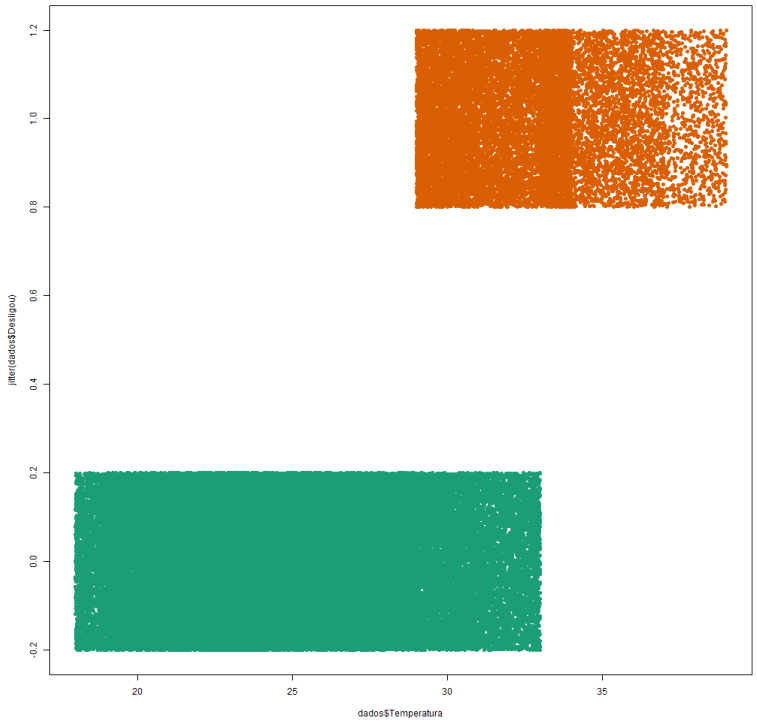


Figura 14 – Representação visual entre a temperatura e o funcionamento da máquina com ruído

Como o conjunto de dados possui mais de 130 mil linhas de observações, será trabalhado com 1500 dados amostrais extraídos de forma aleatória. Este código pode ser acompanhado a seguir:

```
set.seed(111)

exemplo <- sample(nrow(dados), 1500)
amostra <- dados[exemplo,]
```

Para criar a regressão logística no Spark é necessário apresentar o conjunto de dados em um formato que o Spark possa ler. Uma função em R é chamada para converter o *dataframe* que possui os dados amostrais para o formato do Spark. Em seguida, a regressão logística é invocada com estes dados, e informa-se qual é a variável alvo e a variável explicativa.

Na sequência do código, os dados de resposta do modelo gerado são armazenados em um novo objeto que será utilizado para apresentar a curva sigmoide[3] no gráfico. Após este processo, são encontrados os valores de menor temperatura na

[3] Uma curva sigmoide é uma curva em forma de "S" (daí seu nome, visto que a letra S, em grego, tem o nome de sigma). É bastante usada para representar conjuntos de dados que descrevem fenômenos com momentos iniciais com um crescimento lentos, evolução rápida em um segundo momento e estabilização final lenta.

qual a máquina quebrou e de maior temperatura que a máquina continuou funcionando. Por fim, é exibido o gráfico que contempla todas as representações em uma única visão.

```
dadosSpark <- copy_to(sc, amostra, overwrite = T)

glm_spark  <- ml_logistic_regression(dadosSpark, Desligou ~ Temperatura)

fitted_spark <- sdf_predict(dadosSpark, glm_spark)
fitted_spark <- as.data.frame(fitted_spark)

menorQuebraAmostra <- min( amostra[ (amostra$Desligou == 1)  ,
c('Temperatura') ] )
maiorLigacaoAmostra <- max( amostra[ (amostra$Desligou == 0)  ,
c('Temperatura') ] )

plot(amostra$Temperatura, jitter(amostra$Desligou),
col=cores[amostra$Desligou+1], pch=19, main='Amostra')

abline(v = menorQuebraAmostra, col = cores[3], lwd=3)
abline(v = maiorLigacaoAmostra, col = cores[3], lwd=3)

points(amostra$Temperatura, fitted_spark$probability_1, pch=19, col=cores[4])
```

A representação visual da regressão logística binomial com a função sigmoide, e os intervalos de sobreposição da temperatura entre o funcionamento ou não da máquina, pode ser vista na Figura 15.

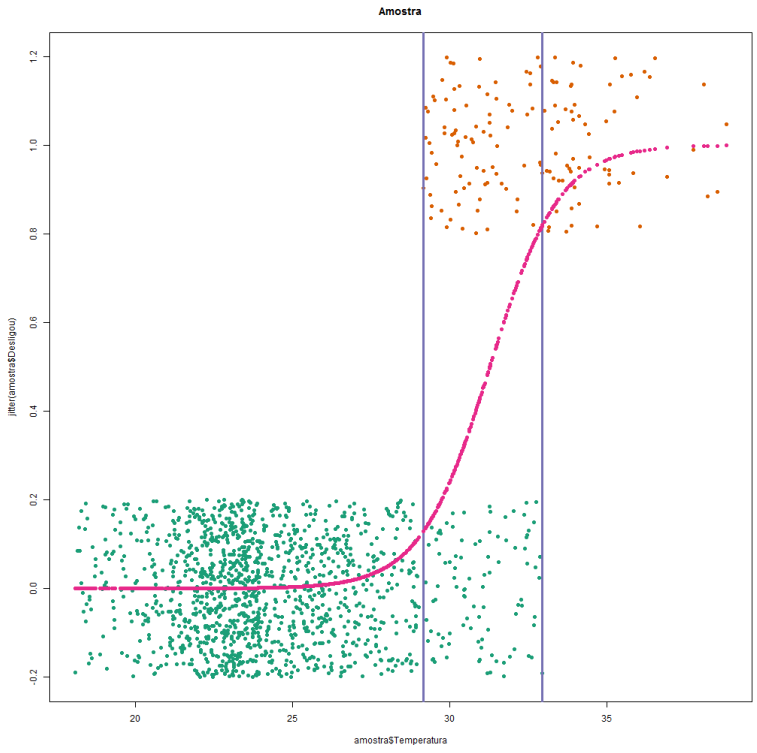


Figura 15 – Distribuição da amostra entre temperatura e funcionamento da máquina, com os itens de sobreposição da temperatura e a curva sigmoide

Na Figura 5, as linhas na vertical apresentam o intervalo de sobreposição das temperaturas. A curva magenta em formato de “S” é a curva sigmoide que explica o comportamento de chance de quebra da máquina a partir de determinada temperatura.

Para finalizar o estudo dirigido, um novo conjunto de temperaturas para interagir com o modelo será criado. São 5 novas temperaturas que simulam um processo em lote (*batch*) que será apresentado ao modelo. O código para criar o conjunto de dados pode ser acompanhado a seguir:

```
NovaTemperatura <-
as.data.frame(matrix(c('22.12','25.52','28.69','31.02','35.14'), ncol = 1,
nrow = 5, byrow = T))
colnames(NovaTemperatura)<- c('Temperatura')
NovaTemperatura$Temperatura <-
as.numeric(as.character(NovaTemperatura$Temperatura))
```

A partir destes dados, será chamado o modelo da regressão logística e recuperada a saída do modelo, que representará a chance de quebra da máquina a partir das temperaturas apresentadas.

No código a seguir será chamada a função que copia os novos dados para o Spark; em seguida é chamada a função de predição passando os novos dados e o modelo da regressão logística, e por fim, os dados de resultado são armazenados em um objeto.

```
novosdadosSpark <- copy_to(sc, NovaTemperatura, overwrite = T)

novosDados_fitted <- sdf_predict(novosdadosSpark, glm_spark)

novosDados_fitted <- sdf_separate_column(novosDados_fitted, 'probability',
list('p1'=1, 'p2'=2) )
novosDados_fitted <- as.data.frame(novosDados_fitted)
```

A partir do resultado predito pelo modelo, foi criada uma tabela que representa a temperatura e a chance de quebra. O código desta tabela pode ser visto a seguir:

```
resultado <- as.data.frame(cbind(NovaTemperatura$Temperatura,
novosDados_fitted$probability_1))
colnames(resultado) = c('Temperatura', 'Chance de Quebra')
resultado
```

O resultado desta tabela é assim apresentado:

Temperatura	Chance de Quebra
22.12	0.0002575166
25.52	0.0055178241
28.69	0.0885087778
31.02	0.4432024096
35.14	0.9704552065

Estendendo a regressão logística impressa na Figura 15, este cruzamento entre Temperatura e Chance de Quebra será apresentado por pontos na cor verde e borda preta. Repare que estes pontos novos seguem exatamente a curva sigmoide que foi criada com o modelo, e mesmo estes novos dados não terem sido utilizados para criar a representação da regressão logística, eles se encaixam perfeitamente na curva sigmoide. Veja, na Figura 16, esta representação com os pontos dos novos dados.

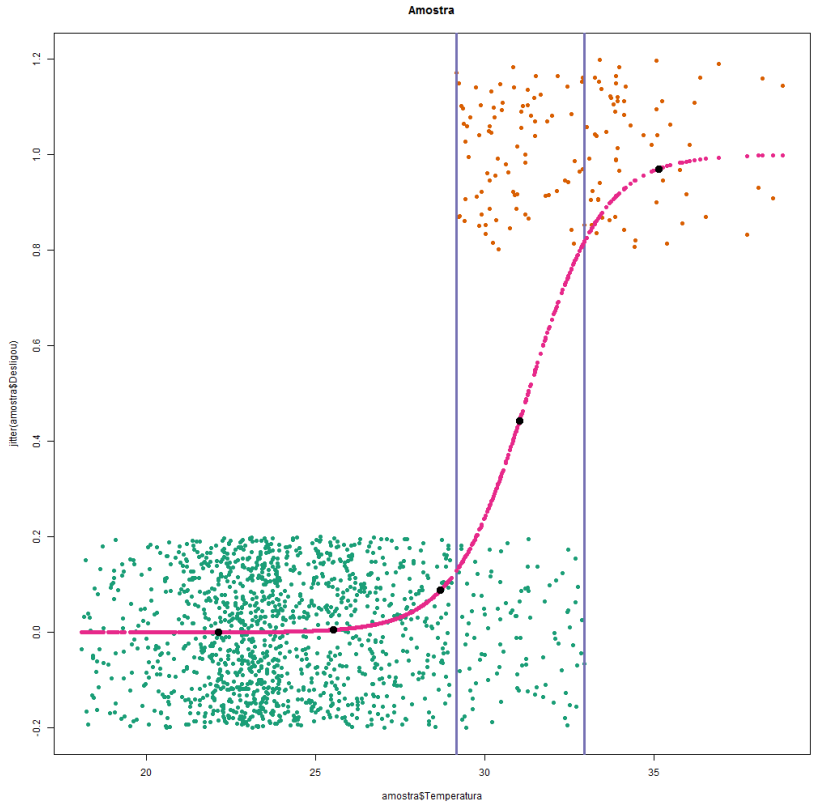


Figura 16 – Curva sigmoide com os novos pontos preditos da temperatura

O código para gerar a Figura 16 pode ser acompanhado a seguir:

```
plot(amostra$Temperatura, jitter(amostra$Desligou),
col=cores[amostra$Desligou+1], pch=19, main='Amostra')
abline(v = menorQuebraAmostra, col = cores[3], lwd=3)
abline(v = maiorLigacaoAmostra, col = cores[3], lwd=3)

points(amostra$Temperatura, fitted_spark$probability_1, pch=19, col=cores[4])

points(NovaTemperatura$Temperatura, novosDados_fitted$probability_1, pch=19,
col='black', lwd=3)
```

1.5. Considerações Finais

O objetivo deste trabalho foi discutir os fundamentos do Big Data com Apache Spark. Para melhor ilustrar tais fundamentos, foi apresentado um *case* em forma de estudo dirigido, em que se apresentou um problema genérico de coleta de dados de sensores que monitoram a temperatura de um equipamento. No estudo, apresentou-se a arquitetura Big Data e uso do Spark com R para a manipulação dos dados.

O foco do presente trabalho foi centrado na análise dos dados, apresentando de forma didática recursos para caracterizar a variável por meio dos seus valores em situações distintas de operações de uma máquina. A discussão da coleta e armazenamento dos dados devem ser vistas a partir das referências apresentadas no trabalho. Espera-se, com isso, que este trabalho possa embasar outras experiências introdutórias e servir como base para iniciantes na área de Ciência de Dados.

Referências Bibliográficas

Apache (2018) Grandes empresas que utilizam Hadoop em seus processos: https://wiki.apache.org/hadoop/PoweredBy

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2), 4.

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business intelligence and analytics: from big data to big impact. MIS quarterly, 1165-1188.

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. Mobile networks and applications, 19(2), 171-209.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. IDC view, 1142(2011), 1-12.

Gates, A., & Dai, D. (2016). Programming Pig: Dataflow Scripting with Hadoop. " O'Reilly Media, Inc."

Gartner's 2014 Hype Cycle for Emerging Technologies Maps the Journey to Digital Business: https://www.gartner.com/newsroom/id/2819918

George, L. (2011). HBase: the definitive guide: random access to your planet-size data. " O'Reilly Media, Inc.".

Ghemawat, S., Gobioff, H., & Leung, S. T. (2003). The Google file system (Vol. 37, No. 5, pp. 29-43). ACM.

Kiran, M., Murphy, P., Monga, I., Dugan, J., & Baveja, S. S. (2015, October). Lambda architecture for cost-effective batch and speed big data processing. In Big Data (Big Data), 2015 IEEE International Conference on (pp. 2785-2792). IEEE.

Laney, D. (2001). 3D data management: Controlling data volume, velocity and variety. META Group Research Note, 6(70).

Linkedin (2015). In The 25 Skills That Can Get You Hired in 2016: http://pt.slideshare.net/linkedin/the-25-skills-that-could-get-you-hired-in-2016

Lynch, C. (2008). Big data: How do your data grow?. Nature, 455(7209), 28.

Meijer E (2011) The world according to linq. Communications of the ACM 54(10):45–51

Prajapati, V. (2013). Big data analytics with R and Hadoop. Packt Publishing Ltd.

SparkR (2018) : https://spark.apache.org/docs/latest/ sparkr. html. Acessado em junho de 2018.

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., ... & Murthy, R. (2010, March). Hive-a petabyte scale data warehouse using hadoop. In Data Engineering (ICDE), 2010 IEEE 26th International Conference on (pp. 996-1005). IEEE.

Ting, K., & Cecho, J. J. (2013). Apache Sqoop Cookbook. O'Reilly Media.

Van der Aalst, W. M. P. Data scientist: the engineer of the future. In: Enterprise Interoperability VI. Springer International Publishing, 2014.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. HotCloud, 10(10-10), 95.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... & Stoica, I. (2012, April). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (pp. 2-2). USENIX Association.

Zikopoulos, P., & Eaton, C. (2011). Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

Withanawasam, J. (2015). Apache Mahout Essentials. Packt Publishing Ltd.

White, T. (2012). Hadoop: the definitive guide: the definitive guide. 2nd ed. O'Reilly Media, Inc.

13. Capítulo 13

Autores:

Marcos Silvano Almeida

Universidade Tecnológica Federal do Paraná (UTFPR) - Campo Mourão

email: marcossilvano@utfpr.edu.br

Lucio Geronimo Valentin

Universidade Tecnológica Federal do Paraná (UTFPR) - Campo Mourão

email: lgvalentin@utfpr.edu.br

André Luiz Satoshi Kawamoto

Universidade Tecnológica Federal do Paraná (UTFPR) - Campo Mourão

email: kawamoto@utfpr.edu.br

Chapter 13

Phaser: a Framework to Create Web Games

Marcos Silvano Almeida, Lucio Gerônimo Valentin and André Luiz Satoshi Kawamoto

Abstract

Videogames represent a habit of millions of people across the globe. By consequence, it is a huge market that extends to almost every computerized device available. Such diversity is supported by a vast selection of game creation tools. Among them, we have the game engine: a collection of tools to aid the creation and execution of games. The Phaser framework is an open source game engine designed to help developers on the task of building web games. This chapter presents a step-by-step tutorial that aims to guide the reader in the creation of a simple platform game by using the Phaser framework.

1. Game Making Tools for Everyone

The digital entertainment is an integral part of millions of people's lives across the globe. Desktop applications, web pages, social networks, smartphones and dedicated game consoles are examples of platforms that offer games and are present on the everyday of the contemporary society (WIJMAN, 2018; ESA, 2018). Each year new technologies expand the horizon to the employment of games. In this regard, games have been used in researches and non-entertainment applications, such as education, training and the promotion of messages. As an entertainment media, the digital games market represent a highly lucrative business that has been ascending since its beginning.

According to Tom Wijman (2018), there are approximately 2.3 billion gamers that are projected to spend $137.9 billion US dollars on games in 2018. This will represent an increase of 13.3% over 2017. This big number is fueled by a large array of companies that develop games and the technologies that support it. Beyond that, with the rise of online digital distribution platforms like Steam, PlayStation Network and Xbox Live, it has become possible to sell games to a worldwide market with little investment. That opened the games market to a large number of small developers that started their own initiatives and launched many successful titles in the mid of 2000s. These small companies are often called independent developers or for short, just indies. The term indie comes from the fact they are independent from a large publisher, which leads a freedom to choose their own creative direction on the design of the game (DUTTON, 2012).

The indies success kindled the creation of free and open source development tools, which has been powered a large number of successful titles since the last decade. There are dozens of tools used in the process of game creation. The main one is called "Game Engine" and consists of a set of tools that aid the creation and execution of a game (ROGERS, 2010). The major features typically found on a game engine include a 2D/3D graphics renderer, physics engine, programing interface and management of sound, network, animations, artificial intelligence and general resource allocation. The wikipedia[1] has a list of common game engines used in the market, though not an absolute one. Among them, we may cite Unity 3D[2], a proprietary engine for 2D and 3D games that uses C#; Cocos2d-x[3], a C++ open source framework for mobile 2D games; and Phaser[4], an open source JavaScript (JS) framework for creation of Web games.

The Phaser framework can be used to create games that run on a web browser or can be packed as a native application for desktop and mobile platforms, such as Android and iOS. Beyond that, the JavaScript is a lightweight language that helps to bring flexibility and productivity to games programming. In this chapter we will discuss an overview of the game development process and present a step by step tutorial on how to create a web game with the Phaser framework. It is important to note that this tutorial is aimed at readers familiar with web development concepts of client-side applications, including notions of HTML (Hypertext Markup Language) and JavaScript languages.

2. The Game Development Process

The game development process borrows techniques from other areas, most notably, software and movies production. It is considered as a multidisciplinary process in the sense that employs roles from different natures. The main roles played by developers on a game development process are listed below:

- Gamer designer: conception of the game;
- Software engineer: design and implementation of the game code and tools;
- Graphics designer: creation of graphical assets for the game;
- Sound designer: creation of sound assets for the game.

Typically on large teams, constituted by dozens or hundreds of workers, these four main roles are subdivides into very specialized tasks. As an example, modern big budget games have team members responsible for producing specific textures, like organic tissue or natural environments. On the other hand, independent projects tend to employ small teams and members usually play many roles at once. Furthermore, if one person develops an entire game by its own, it is understand to plays all the four roles. The developers playing the four aforementioned roles work on a three-step process, which is described in the text below.

A game is typically build by three major sequential steps where people perform the aforementioned roles (BETHKE, 2003). These steps are described in the text below.

[1] Popular game engines: https://en.wikipedia.org/wiki/List_of_game_engines .Accessed on 25/08/2018.
[2] https://unity3d.com/ .Accessed on 25/08/2018.
[3] http://www.cocos2d-x.org .Accessed on 25/08/2018.
[4] https://phaser.io/ .Accessed on 25/08/2018.

2.1. Pre-production

In this phase, the initial concept for the game is established and verified. The concept is typically composed by the description and illustration of various elements, such as game rules, possible interactions and objectives; initial visual concepts, including characters, locations, and animations; and ideas for music and sound effects. This information about the game initial plan is usually recorded in a Game Design Document (GDD), which may vary from a poster to a fully detailed document.

Beyond the conception of the game, the pre-production phase also includes tasks related to software development, including technical feasibility studies, tools research, design and development of the basic software structure for the game code and, most importantly, the creation of one or more game prototypes to evaluate the planned concepts for the game. The prototypes usually employ preliminary versions of the game assets. The feedback from the play tests with the prototypes are recorded on the GDD. In a typical game project, the game concept is built in an incremental and evolving fashion, throughout the first two major phases of the process (CHANDLER, 2009).

2.2. Production

In this phase the game is fully built. The production encompasses the creation of all game software and assets, both visual and audio. If the game was planned for a multiplatform release, a core version is usually first produced with specific tailoring for each platform being made along or after this version is completed. The core version can be developed targeting the most generic platform, which is usually the Personal Computer (PC) or a specific device. Two strategies may apply when choosing which platform to target on the core version: the most limited or the most profitable. Also, functionality and gameplay tests are conducted throughout the entire phase.

2.3. Post-Production

The third and last step on the game development process is comprehended by the tasks centered on concluding and fixing the game, even at post launch. In the post-production phase the developer tasks usually aim at fine tuning, bug fixing and publishing the game on a store.

The progress inside each of the three aforementioned phases commonly occurs in an iterative manner, as the design of the game is gradually established. During the first two phases – pre-production and production – there is a special attention from the team on improving the game concept. Also, it is common to have a lot of rework pushed by feedbacks from gameplay tests, which are frequently employed through the entire process. This changes are related to the fact that the game quality, the one specifically perceived by its users and critics and which may lead to its commercial success, is not defined by a correct implementation of functional requirements. The quality in games is mostly defined by the final experience that emerges from gameplay (ROGERS, 2010).

3. Developing Games for the Web

A web game is made to be used on a web browser. Differently from a standalone game, which is installed on a device to be played, a web game exists inside a HTML[5] page and

[5] https://www.w3.org/html/ .Accessed on 25/08/2018.

can only be played if the user has a web browser installed on his device. However, it is difficult to imagine an operating system that does not come packed with a web browser. In this sense, if we are going to develop a web game, it will probably run on most of the computer devices used across the globe. Also, it is interesting to note that the game can be published outside a digital store. However, we cannot ignore that fact that users commonly prefer to play games that are downloaded and installed from digital stores, like Google Play[6] and App Store[7] (ESA, 2018; WIJMAN, 2018).

There are two publishing options when we are developing web games: embedding the game on a website or packing it inside a standalone application. For the sake of simplicity, this tutorial will aim at the first option. For information related to packing web games as desktop or mobile applications, see Cocoon[8].

4. The Phaser Framework

The Phaser is an open source framework to create 2D web games. It manly aims to aid in the development of games on two platforms:

- Web games: played via web browsers on desktop or mobile devices;
- Mobile games: packed as a native application for smartphones or tablets.

The Phaser provides useful features, common to the majority of 2D games. It can be used to create virtually any kind of 2D game. The main features are list in the framework website[9] and include: graphics rendering; input management; physics simulation; objects, camera and game world manipulation; animations and effects, and device screen scaling.

The framework is built upon JavaScript and uses it as the main programming language. Typescript[10], a superset of JavaScript develop by Microsoft and very popular at the time of this writing can also be used with Phaser. There are two main versions of the framework that are currently maintained: 2.10.5 and 3.9.0. While the latter is newer, the documentation and examples on the official website are all related to the former. Also, the majority of the tutorials and code samples found on the web are target at the version 2. Lastly, the version 3 is still under development and constantly changing. In this tutorial we will use the Phaser 2.10.5.

5. Building a Simple Platform Game

In this section we are going to start the tutorial that aims to guide the reader in a step by step process to create a simple game of the platform genre. Videogame genres are a subject of a broad discussion and there is no definitive list available. According to Rogers (2010), in a platform game the player controls a character to avoid obstacles and to jump between platforms, often suspended on air. Platform games also feature collectibles, enemies, obstacles and the traversing over uneven terrain, which requires jumping and climbing.

---

[6] https://play.google.com/store .Accessed on 25/08/2018.
[7] https://www.apple.com/lae/ios/app-store/ .Accessed on 25/08/2018.
[8] https://cocoon.io/ .Accessed on 25/08/2018.
[9] https://phaser.io/ .Accessed on 25/08/2018.
[10] https://www.typescriptlang.org/ .Accessed on 25/08/2018.

The game we are going to build will allow the player to move the character, jump (and double jump!), collect coins and attack droids by falling on their heads. The player will have three lives and lose each by touching a droid. The game will have two possible outcomes: (1) "Game Over", if the player loses all his lives and (2) "Level Clear" if he catches all coins. There will be only one level. The final game can be played here:

https://marcossilvano.github.io/jolai-platformer/step11_win_or_lose/

Before beginning the tutorial, we need to download the starter template. This and other resources that will be referenced throughout the tutorial are available on the following github repository: https://github.com/marcossilvano/jolai-platformer. The full zip file must be downloaded from the link above and extracted. To begin the tutorial, we need the folder labeled "**step00-starter**". The remaining folders represent the final result of each step of the tutorial. To write the platform game we need a source code editor. While a simple plain text editor can be used, a JavaScript programming editor is preferred. Some examples include VSCode[11], Atom[12] and Sublime[13]. The code for this tutorial was fully written on VSCode.

5.1. The Project Structure

The **step00-starter** is structured as follows:

assets/	Contains the assets for the game. The file **ASSETS-SOURCES.txt** lists the sources for the images.
defs/	Definitions to enable autocomplete on the code editor. VSCode only need this folder and the jsconfig.json file to enable the feature. For other code editor, please refer to its manual.
src/	Contains the JavaScrit source code files. All the code that will be written throughout this tutorial will be placed in files of this folder. All the need files are already created, but contains just code skeletons. Also, the framework file, labelled **phaser.2.10.5.js** is placed inside the subfolder **lib/**. This file contains the source code of the Phaser engine.
index.html	As a web local application, the game needs an html file as a start point. The index.html only declares the basic structure of a html file and adds the reference to every JavaScript file that we are going to write code into. This is the only file that is already fully coded.

To start the tutorial, open the **step00-starter** folder with your code editor. Will be working on the following steps in order to produce the final result: (1) initialize the game, (2) load all assets, (3) create the game screen, (4) add the backgrounds, (5) load a tile map, (6) create the Player, (7) define the game's HUD (see section 5.9), (8) add collectables, (9) add enemies and (10) define conditions for victory and failure. To test the game at any time during this tutorial, open **index.html** file on the **Firefox**[14] browser.

[11] https://code.visualstudio.com/ .Accessed on 25/08/2018.
[12] https://atom.io/ .Accessed on 25/08/2018.
[13] https://www.sublimetext.com/ .Accessed on 25/08/2018.
[14] The Mozilla Firefox is perfect to this tutorial because it allows us to execute local JavaScript code.

5.2. The Game Loop

The Game Loop is the central component of every game. It is responsible for continuously running logic to build every game frame (MADHAV, 2018). A game is basically a sequence of continuous frames being displayed on the screen. For every frame, the game loop calls codes responsible for computing the game logic, which includes checking collisions, moving characters and responding to the player inputs, and drawing everything on the screen. A game commonly processes and displays 30 or 60 frames per second, thus creating the illusion of moving objects on the screen. Game engines and frameworks, such as Phaser, usually hides underneath complexities and exposes only the parts the developer will need to program and specialized in order to achieve the desired behavior and look for the game.

The *Phaser.Game*[15] comprehends the game loop and controls the game execution through scenes. A Phaser game is composed of one or more scenes, which must be coded as subclasses of *Phaser.State*[16]. Thus, *Phaser.States* has the common features that can be used in a game scene, providing a base class that can be extended for the game specific needs. Only one *Phaser.State* can be active at time. The most commonly implemented methods in a subclass of *Phaser.State* are *preload*(), *create*() and *update*(). They are described as follows:

- **preload() -** The *preload*() method is called when a scene is created for loading assets that will be used on the scene, such as images, sounds and game data.
- **create() -** After *preload*(), the method *create*() is invoked to setup the scene, which includes creating game objects, characters, maps and user's input.
- **update() -** When *create*() is completed, the game loop calls the *Phaser.State update*() method to handle the game logic for every frame (30 or 60 times per second). The rendering of the game graphics is done automatically by Phaser for the game objects included in the current *Phaser.State*.

  A representation of the aforementioned steps is presented in Figure 5.1[17].

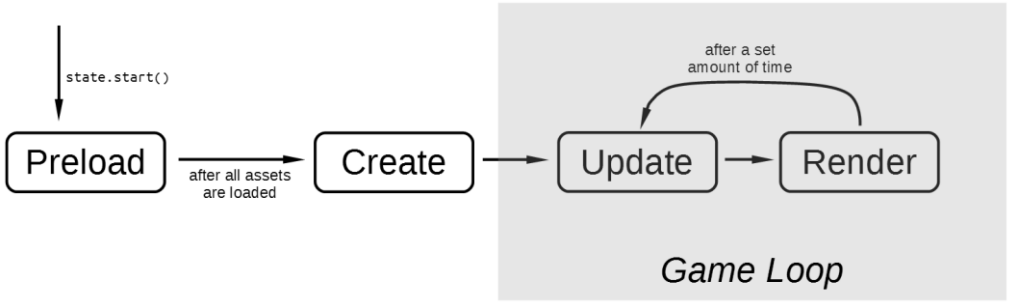


Figure 5.1 - Game loop in Phaser.

5.3. Initializing the Game

Open the src/Game.js file. On the first line, inform the JavaScript engine that we want to run the game code under **strict mode**, a more restricted variant of the language. This

[15] https://phaser.io/docs/2.6.2/Phaser.Game.html .Accessed on 25/08/2018.
[16] https://phaser.io/docs/2.6.2/Phaser.State.html . Accessed on 25/08/2018.
[17] https://leanpub.com/html5shootemupinanafternoon/read . Accessed on 25/08/2018.

mode helps developers to find code mistakes and errors more easily. Information on the strict mode can be found on the Mozilla website[18]. Write this line to enable strict mode:

```
'use strict'
```

Now, let's write down some configuration data for the game. This practice is particularly useful for centering all game properties in one place. Is also helps managing later changes and fine tuning of the game. The code is shown in Figure 5.2.

```
const config = {}
config.RESX = 800  // width in pixels of the game view
config.RESY= 480  // height in pixels of the game view
// the browser's canvas API has better compatibility than WebGL
config.RENDERER = Phaser.CANVAS
config.ROUND_PIXELS = true // improves pixel fidelity
// game view will scale maintaining proportions
config.SCALE_MODE = Phaser.ScaleManager.SHOW_ALL
config.GRAVITY = 1500   // gravity value for physics engine
// general player properties
config.PLAYER_VELOCITY = 200
config.PLAYER_FALL_VELOCITY = 400
config.PLAYER_JUMP_VELOCITY = 500
config.PLAYER_DOUBLE_JUMP_VELOCITY = 600
config.PLAYER_LIVES = 3
```

Figure 5.2 - The config object will store general game properties.

The first batch of properties is related to the game and the purpose of each is described as commentaries directly on the code above. The second batch is player related.

After declaring these general configurations, let's initialize the game. For this purpose, we are going to create a Game class that extends *Phaser.Game*. The code is shown in Figure 5.3. The constructor calls the super constructor and initializes the engine. After that, we will add scenes to the game. Game scenes are code in Phaser as subclasses of *Phaser.State*. In the code bellow, we have two states being registered into the game: a *BootState*, which will load all the game assets prior to the game execution, and a *GameState*, which represents the game itself. Phaser uses string keys to identity resources. In the code below, the 'Boot' string is the key that will be used to identify the *BootState* class being registered. Then, we start the game with the Boot state. New Phaser state classes could be created to accommodate game screens, such as a tittle screen, game over or high score. The last line of code instantiate the Game class, hence starting the game.

```
class Game extends Phaser.Game {
   constructor() {
       super(config.RESX, config.RESY, config.RENDERER, null, null,false)
       // adding scenes (Phaser.State) to the game
       this.state.add('Boot', BootState)
       this.state.add('Game', GameState)
```

[18] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode .Accessed on 25/08/2018.

```
        this.state.start('Boot')
    }
}
const GAME = new Game()  // create the game
```

Figure 5.3 - The Game class initiates the engine and defines the game states.

5.4. Booting Up: Loading All Game Assets

The first state to be executed by the game is the *BootState*. It is responsible for loading all the game assets, such as data, images, sounds and sprite sheets. Again, we use string keys to identity the images being cached. After cached, all the resources can be later accessed by their string keys. A sprite sheet is an image that combines several different frames for a game object, such as the player character. You can check the player sprite sheet by opening the file **assets/player.png**.

The *BootState* is a subclass of *Phaser.State* and has two methods: *preload*() and *create*(). The code for the *preload*() method is shown in Figure 5.4. This method is used by Phaser states for loading and caching game assets. Every specialized *Phaser.State* class can have a *preload*() method but, in this tutorial, we are preloading all the game assets in one state, prior to the game execution. This approach is common to many Phaser games.

```
class BootState extends Phaser.State {
    preload() {
        this.game.load.image('background-sky', 'assets/bg01.png')
        this.game.load.image('background-mountain', 'assets/bg02.png')
        this.game.load.image('life-icon', 'assets/life-icon.png')
        // load characters and effects sprite sheets
        this.game.load.spritesheet('player', 'assets/player.png', 49, 72)
        this.game.load.spritesheet('droid', 'assets/droid.png', 64, 64)
        this.game.load.spritesheet('coin', 'assets/coin.png', 32, 32)
        // load tile map data and image
        this.game.load.tilemap('level1', 'assets/level1.json', null,
                               Phaser.Tilemap.TILED_JSON);
        this.game.load.image('tiles1','assets/tileset-42x42.png');
    }
```

Figure 5.4 - BootState will load all the assets into the game with the preload() method.

The *preload*() method presented in Figure 5.4 loads the images and sprites for the game elements. It also loads the map file and *tileset* – an image containing the building blocks for a tile map. We will discuss tile maps on the *GameState* class.

The *create*() method for the *BootState* is presented in Figure 5.6. It doesn't create any game object but, instead, initializes some engine parameters, such as how game is scaled to fit the available area on the screen and the way images are drawn. In this tutorial we are setting the engine renderer to draw images in a more pixelated manner, as opposed to have blurry characters (see Figure 5.5).

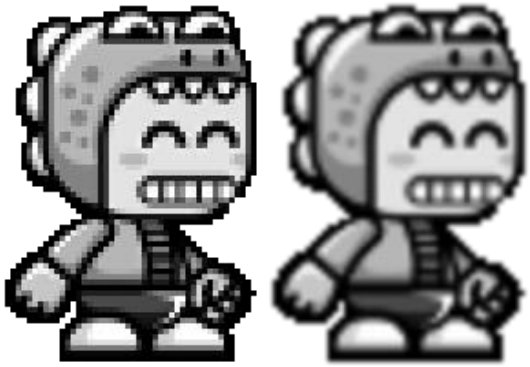


Figure 5.5 - Different methods for presenting images on Phaser: pixelated (left) and smoothed (right)[19].

The code for the *BootState.create*() at Figure 5.6 starts by setting two properties with the values stored in the ***config*** object, previously declared in the Game.js file. Next, we initialize the support for a gamepad in order to allow to the user to play the game with an Xbox 360[20] controller connected to the computer. It is interesting to note that the game will also work on the Xbox One[21] native web browser. Lastly, we ask the engine to change to the *GameState*, where all the real game will run.

```
create() {
    this.game.scale.scaleMode = config.SCALE_MODE;
    // pixel fidelity: render images only at integer positions
    this.game.renderer.renderSession.roundPixels = config.ROUND_PIXELS
    this.game.stage.smoothed = false // "retro", pixelated drawn
    this.game.input.gamepad.start()  // init game pad support
    this.state.start('Game')         // start the game scene
}
```

Figure 5.6 - Create() method for the BootState class. It sets up general game properties.

5.5. The Game State Skeleton

The main class on the game code is the *GameState*. It houses all the game logic and encompasses the three game objects used on the game: the player character, the coins and the droids, which are the enemies. Just like the *BootState* class, the *GameState* is a subclass of *Phaser.State*. Let's start with a basic skeleton for the class. The main methods on the class will be *create*() and *update*(). The *create*() has the same purpose of the one implemented on *BootState*. It is responsible for the creation and setup of all the scene logic and objects. Check the comments on the *create*() method to get an overview of the necessary steps. The two lines already implemented are responsible for starting the physics engine and defining the world's vertical gravity. Again, the global *config* object (Game.js) is being used to define the amount of vertical gravity. The Phaser framework

[19] https://bevouliin.com/crocodile-mascot-game-character-sprites/. Accessed on 25/08/2018.
[20] https://www.xbox.com/en-US/games/xbox-360. Accessed on 25/08/2018.
[21] https://www.xbox.com/en-us/xbox-one. Accessed on 25/08/2018.

has integration with different physics engines[22], including popular ones used in various game frameworks. In this tutorial we will be using the built-in *Arcade Physics*.

The second major method that will be implemented on the *GameState* class is the *update()*. It is the method that houses the game logic on every Phaser state and game object. The code in Figure 5.7 presents a skeleton for the *GameState update()*. The comments are a suggestion of the necessary steps to run its part of the game logic.

```
class GameState extends Phaser.State {
   create() {
      // 1. init physics
      this.game.physics.startSystem(Phaser.Physics.ARCADE)
      this.game.physics.arcade.gravity.y = config.GRAVITY;
      // 2. create and setup objects: map, items, enemies and player
      // 3. create and setup HUD
   }
   update() {
      // 1. update screen game logic for one frame
      // 2. check collisions between game objects:
      // (a) player and map; (b) player and items; (c) player and enemies
   }
}
```

Figure 5.7 - The GameState skeleton. This state (scene) will execute our game logic.

Like an animated cartoon, the perception of movement in a game is created by a sequence of static frames. The *update*() method runs at each frame and represents a step of the game logic. Every game runs in a loop, which means each frame we see on the display must be processed as one step of the "game loop". Considered the heart of a game, the game loop varies from 30 to 60 frames per second, accordingly to the heaviness of the code being executed. Each frame is created by one iteration of the loop, which is comprised by at least three parts: input capture, logic execution and drawing of everything on a single frame.

The Phaser game loop is configured to run at 60 frames per second, which means the code inside the *update*() method at the *GameState* and at all the game objects will execute 60 times per second each. As many game engine available on the market, the drawing part is automatically done by Phaser. This facilitates the process of creating games as we can focus our efforts on creating the logic, and not the underneath algorithms needed to draw a frame on the device.

5.6. First Step: Adding a Background

The first thing we are going to do is add a background to the game. We will be using two sprites in order to create a parallax effect. Also called parallax scrolling (SHANKAR, 2007), it is vastly popular in 2D games and used to create a simple illusion of depth with multiple images. The technique consists of moving images on deeper layers gradually slower than the ones at the foreground. The *GameState* will implement a very simple

[22] https://phaser.io/docs/2.6.2/Phaser.Physics.html .Accessed on 25/08/2018.

version of parallax scrolling, by using only two tilesprites. According to Phaser manual[23], a *TileSprite* is a Sprite that has a repeating texture. The texture can be scrolled and scaled independently of the *TileSprite* itself. Although they have similar names, there are no relation between *TileSprites* and *TileMaps*.

Let's add the backgrounds to the game. First, create the *createBackgroundLayer*(), which will be responsible for adding one background to the game. Then, write a second method, the *createBackgrounds*(), that will simply call the first method twice and assign each background to a separated field. The implementation is shown in Figure 5.8. In order to create each background, we add a *TileSprite* by using an auxiliary method provided by Phaser: *this.game.add.tilesSprite*(). After that, we set the *TileSprite* to be fixed at the camera, which means, it will move together with the camera. Finally, we scale the backgrounds a bit as their textures don't fit the screen size. Note that the images previously cached at the *BootState* are accessible by their string keys. The same could be done at any part of the game after the *BootState* is fully executed.

```
createBackgroundLayer(img) {
    let bg = this.game.add.tileSprite(
        -10, 0, this.game.width+20, this.game.height, img)
    bg.fixedToCamera = true
    bg.tileScale.setTo(2, 3)
    return bg
}
createBackgrounds() {
    this.bgSky = this.createBackgroundLayer('background-sky')
    this.bgMountain = this.createBackgroundLayer('background-mountain')
}
```

Figure 5.8 - Methods responsible for creating and setting up the game backgrounds.

On the *create*() method, call *createBackgrounds*() as the first thing after the physics initialization (Figure 5.9).

```
create() {
   // ...init physics ...
   this.createBackgrounds()
... }
```

Figure 5.9 - Adding the call for createBackground() into the create() method.

With the two backgrounds as *TileSprites* being accessible by fields, we can add game logic in order to make them move. The farthest background, a sky texture, will constantly move left. The second background, a mountains texture, will move accordingly to the camera, at a slower rate (the camera will not move yet). The Figure 5.10 shows the two lines that must be added at the beginning of *update*().

```
update() {
    this.bgSky.tilePosition.x -= 0.2
    this.bgMountain.tilePosition.x = -this.camera.x/5
```

[23] https://phaser.io/docs/2.6.2/Phaser.TileSprite.html .Accessed on 25/08/2018.

```
... }
```

Figure 5.10 - Updating the position of both backgrounds, 60 times per second.

5.7. Creating the Tile map

According to MDN (2018), *Tilemaps* are a very popular technique in 2D game development. It consists of building the game level map as a two dimensional array of small, regular-shaped imagens called tiles. The most common kind of tile map is formed by rectangular images. In this tutorial we are going to use a tile map to create the game level. A very popular free tile map creation tool is the Tiled editor[24]. The level of our game was built using Tiled. The process of creating the level is beyond the scope of this tutorial. It is rather easy to find good tutorials for the tool on YouTube[25]. Also, the official page offers a very comprehensive manual[26].

The tile map level of our game is comprised of two files. You can find both in the **assets/** folder:

- **tileset-42x42.png**: an image containing the tiles. The arrangement works like a sprite sheet: a sequence of rectangular building blocks.
- **level1.json**: a text file with the map description. The map is basically a two dimensional array where each cell indicates the index of a tile. The Tiled editor offers options regarding data formats, including JSON and XML.

Loading a tile map in Phaser is a very straightforward process. For this purpose, we are going to add a method to the *GameState* class. Let's name it *createTileMap*(). The code is shown below. It is important to note that the two aforementioned files were already cached by the *BootState.create*(). In our method, the first line creates the tile map by passing the string key registered in *BootState.create*() and stores a reference to the map object in a field. The second line does the same, but with the tiles image. Inside the map data there are two layers: one for the level map, which is made by tiles; and another for the game objects, which will be *Phaser.Sprites*, such as collectibles and enemies. These two layers were defined inside the Tiled editor. An arbitrary number of layers can be created for the game using Tiled. Keep in mind that there are processing and rendering costs associated with each tiles layer: the more the layers, the bigger the cost to run them on the game.

On the code shown in Figure 5.11 we are creating the 'Tiles Layer 1' and setting the tiles from 1 to 11 as solid (impassible) on this layer. These indexes refer to the tiles image and not the layer. We are also associating a callback event to the spikes tile. Then, when the player sprite overlaps a spike tile on the map, the *hitPlayer*() method will be called. This method is explained in the next section. The last line of code just tells Phaser to resize the game world, originally sized at 800 by 480 pixels, to match the map size.

```
createTileMap() {
    this.map = this.game.add.tilemap('level1');
    this.map.addTilesetImage('tiles1');
    this.mapLayer = this.map.createLayer('Tile Layer 1');
```

[24] https://www.mapeditor.org/ .Accessed on 25/08/2018.
[25] https://www.youtube.com/ .Accessed on 25/08/2018.
[26] http://doc.mapeditor.org/en/stable/ .Accessed on 25/08/2018.

```
    // define solid tiles and a callback event for the spikes tile
    this.map.setCollisionBetween(1, 11, true, 'Tile Layer 1')
    this.map.setTileIndexCallback(29, this.hitPlayer, this)
    // resize game world to fit tile map
    this.mapLayer.resizeWorld();
}
```

Figure 5.11 - Creating the Phaser.TileMap object and setting up the map for the game.

After writing the *createTileMap*() method, let's call it on *create*(), below the lines that add the backgrounds into the game. The code is shown in Figure 5.12.

```
create() {
    // ...background creation...
    this.createTileMap()
... }
```

Figure 5.12 - Calling the createTileMap() on create().

5.8. Creating the Player

In this step we will create the most fundamental part of our game: the player character. The importance of this code is related to the fact that is going to be our main interface with the game or, in other words, we will play the game through this character. The Player will be a subclass of *Phaser.Sprite*[27] and the code will be written in the Player.js file. While Phaser states are subclasses of the base *Phaser.State* class and contain specialized code for game screen, the game objects must be subclasses of the base *Phaser.Sprite* class. Game objects also have an *update*() method to house its own logic. In our game, the Player will manages its logic, including checking inputs from the user and moving the character accordingly. The overall structure for the Player class is presented in Figure 5.13. The two main methods will be the class *constructor*(), to setup everything needed, and the *update*(), which will hold all the character logic. The sequence of setups is described in the comments.

```
class Player extends Phaser.Sprite {
    constructor(game, x, y, img) {
        super(game, x, y, img) // call Phaser.Sprite constructor
        // 1. setup general properties
        // 2. setup animations
        // 3. setup character physics
        // 4. create and setup fields for player logic
        // 5. setup keyboard keys
        // 6. setup xbox 360 controller buttons
    }
    update() {
        // move the player accordingly to user's input, if active
    }
```

[27] https://phaser.io/docs/2.6.2/Phaser.Sprite.html. Accessed on 25/08/2018.

```
}
```

Figure 5.13 - Basic structure for the Player class.

Before starting working on the Player logic, let's add an instance of the Player class to the game screen (*GameState*). To this end, add the lines of code presented in Figure 5.14 to the *GameState.create*() method, after creating the tile map. The first step is to instantiate a Player object by filling the constructor parameters, which are: the game reference, the position on the screen and the string key that identifies the player sprite sheet, previously cached on the *BootState.preload*() method. We are storing the player object in a field and adding to the game on the next line. Then, we set the camera to follow the Player position with some easing.

```
create() {
    // ...creation of the tile map...
    this.player = new Player(this.game, 100, 100, 'player')
    this.game.add.existing(this.player)
    this.game.camera.follow(this.player,
            Phaser.Camera.FOLLOW_LOCKON, 0.05, 0.05)
... }
```

Figure 5.14 - Adding the Player class into the GameState.

The final step consists of adding the collision between the player and the tile map layer, preventing the player from passing through ground and walls. Thanks to the physics engine, it's only a matter of one line. There are two methods for collision checking: *collide*() and *overlap*(). The difference is that *collide*() doesn't allow the sprites to overlap, making them solid. Add the following line to the end of *GameState.update*().The *update*() method will house the collisions checking for all game objects, as the *GameState* contains references to these objects. The code is shown in Figure 5.15.

```
update() {
    // ...background movement...
    this.game.physics.arcade.collide(this.player, this.mapLayer);
... }
```

Figure 5.15 - Checking collisions between Player and Map in GameState.update().

In the last section, we saw that the tile map has some spike tiles and a callback event was associated to them (Figure 5.11). Therefore, when the player sprite overlaps a spike tile on the map, a *GameState.hitPlayer*() method will be called. This method is presented in Figure 5.16 and works as follows: every time the player suffers a hit, it loses one of its three lives and returns to the start position on the level; if all lives are lost, the game is over. We don't have the game's HUD implemented yet as well as the *gameOver*() method. For now, just comment these two lines (but don't forget to remove the comments later). These topics will be addressed respectively in sections 5.9 and 0.

```
hitPlayer() {
    // return to start position
    this.player.damage(1)
    this.player.x = 100
    this.player.y = 100
    this.updateHud()
```

```
    if (!this.player.alive)
        this.gameOver()
}
```

Figure 5.16 - Callback to handle the collisions between Player and hazards.

5.8.1. The Player Constructor

Let's start with the constructor setups (Figure 5.17). First, we define the health property and the anchor for the sprite. We will use the built-in heath property to store the player lives. The anchor is a point X,Y that defines the origin of the sprite, with normalized values. For X, the value 0 means left side of the sprite and 1, the right side. The same applies to Y, where 0 means top and 1 means bottom. In our code, we are putting the player's anchor at the center of the sprite (point 0.5, 0.5). This is particularly useful when we want to rotate and scale the sprite. Next, we setup four animation sequences for the player. We define the animation identifier (a string key), an array with the frames indexes and enable loop, which means the animation will repeat continuously. The frame indexes refer to the sprite sheet cached for the player, on the *BootState* class (file **assets/player.png**).

```
// 1. setup general properties
this.health = config.PLAYER_LIVES
this.anchor.setTo(0.5, 0.5)
// 2. setup animations
this.animations.add('idle', [0, 1], 5, true);
this.animations.add('run', [2, 3, 4, 5], 5, true);
this.animations.add('jump', [6], 5, true);
this.animations.add('fall', [7], 5, true);
```

Figure 5.17 - Player constructor: setting up general properties and animation sequences.

Now we are going to setup the character physics and will create some fields that will be used in the logic for the player. Thanks to the physics engine, working with collisions is really easy on Phaser. We just have to configure the character's body, check for collisions and write callback functions. The body is the representation of the sprite on the physics world[28]. In the code shown in Figure 5.18, we enable the physics body for the player sprite, limit its positioning to the game world bounds and define the size of the collision box that surrounds the sprite. Next, we add some fields to the player object that will be part of our implemented logic.

```
// 3. setup character physics
game.physics.arcade.enable(this)
this.body.collideWorldBounds = true
this.body.setSize(this.width-14, this.height-11, 11, 6)
// 4. create and setup fields for player logic
this.isDoubleJump = false
this.startX = this.x
```

[28] https://phaser.io/docs/2.6.2/Phaser.Physics.Arcade.Body.html. Accessed on 25/08/2018.

```
this.startY = this.y
```

Figure 5.18 - Player constructor: setting up physics and defining logic related fields.

The final part of setups needed in the class constructor regards to input. First we are going to define an object with three fields, each related to a keyboard key[29]. In the code shown in Figure 5.19, we are telling Phaser we want to check the status of the left and right arrows – for movements – as well as the control key – for jump. While the player will move as long as we hold left or right, the jump is a one-time press event: to do a jump the user must press the control key. Holding the key will not affect the jump in anyway. If he wants the character to performance a new jump, he has to release the key and press it again. In the code below, we achieve this by adding a callback anonymous function to the "onDown" event of the jump key. The last part of setups in the constructor is about enabling support for the Xbox 360 controller by using an object of *Phaser.GamePad*[30].

```
// 5. setup keyboard keys
this.keys = this.game.input.keyboard.addKeys({
    left: Phaser.KeyCode.LEFT,
    right: Phaser.KeyCode.RIGHT,
    jump: Phaser.KeyCode.CONTROL
})
// one time press event for jump
this.keys.jump.onDown.add(function () {
    this.jump()
}, this);
// 6. setup xbox 360 controller buttons
this.pad = this.game.input.gamepad.pad1
this.pad.onDownCallback = this.gamepadJump
this.pad.callbackContext = this
```

Figure 5.19 - Final part of the Player constructor: setting up input from keyboard and gamepad.

Differently from the keyboard key events, the gamepad Phaser API only allow us to add a callback to any button press event. In the code shown in Figure 5.20, we set the *Player.gamepadJump*() as the callback method for any button press event on the gamepad. Then, inside the method below, we can check which button was pressed. If it was the jump button, we call the *Player.jump*() method. The *jump*() logic will be explained in the section 5.8.3.

```
gamepadJump() {
    if (this.pad.justPressed(Phaser.Gamepad.XBOX360_A))
        this.jump()
}
```

Figure 5.20 - Callback for every game pad button press.

[29] https://phaser.io/docs/2.6.2/Phaser.Keyboard.html. Accessed on 25/08/2018.
[30] https://phaser.io/docs/2.6.2/Phaser.Gamepad.html. Accessed on 25/08/2018.

5.8.2. Player Movements

Referring to the class skeleton provided in the section 5.8, we are now going to write down the player logic, which starts in the *Player.update*() method (Figure 5.21). First, we check whether the physics body is disabled and, if so, we just set the animation frame to 0 and abort (we will disable the player's body when the level is clear). If the body is enabled, we call the *Player.movePlayer*() method, which runs input and movement logic.

```
update() {
    if (!this.body.enable) { // do nothing if body is disabled
        this.frame = 0
        return
    }
    this.movePlayer()
}
```

Figure 5.21 - Updating the player logic on Player.update() method.

Next, we head to the *Player.movePlayer*() (Figure 5.22). The movement logic is rather simple: if the user hold the left arrow on the keyboard or move the left stick on the Xbox 360 gamepad to the left, we set the player X velocity to a negative value. If the user does the same to the right side, we set the velocity to a positive value. If no direction is hold on the keyboard or gamepad, then the X velocity stays zero. After, we limit the jumping and falling Y speed by using a math utility. Then, we call the *Player.animate*() method which will play the correct animation sequence according to the body velocity.

```
movePlayer() {
    // check keys and define player velocity
    if (this.keys.left.isDown ||
        this.pad.axis(Phaser.Gamepad.XBOX360_STICK_LEFT_X) < -0.1) {
        this.body.velocity.x = -config.PLAYER_VELOCITY
    } else
    if (this.keys.right.isDown ||
        this.pad.axis(Phaser.Gamepad.XBOX360_STICK_LEFT_X) > 0.1) {
        this.body.velocity.x = config.PLAYER_VELOCITY
    } else {
        this.body.velocity.x = 0
    }
    // limit the jump and fall velocity
    this.body.velocity.y = Phaser.Math.clamp(this.body.velocity.y,
        -config.PLAYER_DOUBLE_JUMP_VELOCITY, config.PLAYER_FALL_VELOCITY)
    // turn and animate the player sprite
    this.animate()
}
```

Figure 5.22 - Method containing the Player movement logic.

The *Player.animate*() method defines the sprite animation based on the body velocity and the *body.onFloor*() physics function. This utility checks if there is any solid tile below the sprite. At the end of the method, we use the sprite scale to define which

direction it will face. The logic is very straightforward and can be checked in Figure 5.23. The *Phaser.Animation*[31] class is used to manage and play different player animations.

```
animate() {
    let anim = 'idle' // if has no movement and it's on floor: idle
    // on floor and moving
    if (this.body.velocity.x != 0 && this.body.onFloor())
        anim = 'run'
    // on air and moving up: jumping
    else if (this.body.velocity.y < 0 && !this.body.onFloor())
        anim = 'jump'
    // on midair and falling
    else if (this.body.velocity.y > 0 && !this.body.onFloor())
        anim = 'fall'

    this.animations.play(anim)      // play animation on the sprite

    if (this.body.velocity.x > 0)   // use scale to flip the sprite
        this.scale.x = 1
    else if (this.body.velocity.x < 0)
        this.scale.x = -1
}
```

Figure 5.23 - The method checks the Player movement and sets the animaion.

5.8.3. Jumping

The player already moves and falls, but it doesn't jump. For this purpose, we are going to implement the *Player.jump*() method. There are three situations where the player will perform a jump:

- Normal Jump: the sprite is on the floor and the user presses the jump button;
- Double Jump: the sprite is in midair and the user presses the jump button. This can be performed only once while in midair.
- Attack Jump: if the player falls on the head of a droid, it will jump again automatically. While in midair, a new double jump can be performed.

The three types of jump are implemented into the *Player.jump*() method. This method is attached as a callback event when the user presses the jump key – Phaser.KeyCode.CONTROL – or jump button– Phaser.Gamepad.XBOX360_A. Refer to the section 0 to check how the callback jump() method is attached to these inputs. On the code shown in Figure 5.24, the player will jump if one of the following conditions are satisfied: (1) it is on floor, (2) not in a double jump or (3) has fallen on a droid (an attack jump). The act of jumping is accomplished by setting the Y velocity of the player's body. The rest is done by the gravity provided by the physics engine.

```
jump(isAttack = false) {
    let onFloor = this.body.onFloor() // anything below character?
```

[31] https://phaser.io/docs/2.3.0/Phaser.Animation.html. Accessed on 25/08/2018.

```
    if (onFloor || !this.isDoubleJump || isAttack) {
        // not on floor: can be a double jump or an attack jump
        if (!onFloor) {
            this.body.velocity.y = -config.PLAYER_DOUBLE_JUMP_VELOCITY
            // if is not an attack jump, then is a double jump
            this.isDoubleJump = !isAttack
        } else { // normal jump
            this.body.velocity.y = -config.PLAYER_JUMP_VELOCITY
            this.isDoubleJump = false
        }
    }
}
```

Figure 5.24 - The jump() method controls three types of jumps.

5.9. Head-Up Display

A game HUD (Head-Up Diplay) comprehends the textual and graphical information being displayed to the player as an overlay (ROGERS, 2010). In our game's HUD we are going to display counters for the player lives and coins collected. We will also show text messages for 'Game Over' and 'Level Clear' when any of these conditions are reached. The implementation starts by coding a text creation method. The code for the utility is shown in Figure 5.25. It creates a *Phaser.Text* object and defines several of its properties. They are similar to Cascading Style Sheets (CSS) and can be found on the Phaser documentation[32], if necessary.

```
createText(x, y, string, size=16, color='white', shadow=false) {
    let style = { font: `bold ${size}px Arial`, fill: color }
    let text = this.game.add.text(x, y, string, style)
    if (shadow)
        text.setShadow(3, 3, 'rgba(0,0,0,0.5)', 2)
    text.stroke = '#000000';
    text.strokeThickness = 4;
    text.anchor.setTo(0.5, 0.5)
    text.fixedToCamera = true
    return text
}
```

Figure 5.25 - Text creation utility.

Next, let's write a *createHud*() method that will be called to create and setup all the game HUD. First, it creates an object with fields for every information that will displayed. Textual information are using *Phaser.Text* objects created with the *createText*() utility method. Icons are using *Phaser.Sprites*. The code is in Figure 5.26.

```
createHud() {
    this.hud = {
        gameover: this.createText(this.game.width/2, 200, 'GAME OVER',
                  60, '#ff0000', true),
```

[32] https://phaser.io/docs/2.6.2/Phaser.Text.html .Accessed on 25/08/2018.

```
        gamewin:  this.createText(this.game.width/2, 200, 'LEVEL CLEAR',
                  60, '#00dd00', true),
        lifeIcon: this.game.add.sprite(20, 25, 'life-icon'),
        lifeText: this.createText(75, 50, 'x 0', 20),
        coinIcon: this.game.add.sprite(23, 60, 'coin'),
        coinText: this.createText(75, 80, 'x 0', 20)
    }
    this.hud.gameover.alpha = 0
    this.hud.gameover.scale.setTo(1.5, 1.5)
    this.hud.gamewin.scale.setTo(1.5, 0)
    this.hud.lifeIcon.scale.setTo(0.8, 0.8)
    this.hud.lifeIcon.fixedToCamera = true
    this.hud.coinIcon.fixedToCamera = true
}
```

Figure 5.26 - Creating the game HUD.

The next method is responsible for updating the data on the HUD (Figure 5.27).

```
updateHud() {
    this.hud.lifeText.text = `x ${this.player.health}`
    this.hud.coinText.text = `x ${this.coinCount}`
}
```

Figure 5.27 - Updates the data displayed on the Game HUD.

Lastly, add three lines to the end of the *GameState.create*() method. They are preparing the HUD and adding a nice fade effect when the game starts (Figure 5.28).

```
create() {
    ...
    this.createHud()
    this.updateHud()
    this.game.camera.flash(0x000000)
}
```

Figure 5.28 - Calling the HUD creation and update methods in GameState.create().

5.10. Collectables

Most platform games have collectibles in order to promote the exploration of the environment by the user. In our game, we will use a coin as a collectible item. The game will end in a victory condition – "Level Clear" – if the user catches all the coins. The coin will be a subclass of *Phaser.Sprite* and will be written in the Coin.js file. The class code is pretty simple and extents to only a couple of lines. The whole code will be written inside the class constructor. It repeats some setup from the Player: calls the *Phaser.Sprite* constructor, enables a physics body, disables gravity and defines an animation for the coin sprite. We need to disable gravity as the coin must stay static at its position and not fall with the vertical gravity. The entire class for the coin object is shown in Figure 5.29.

```
class Coin extends Phaser.Sprite {
    constructor(game, x, y, img) {
```

```
        super(game, x, y, img)
        this.game.physics.arcade.enable(this);
        this.body.allowGravity = false
        this.animations.add('spin', [0, 1, 2, 3, 4, 5], 10, true)
        this.animations.play('spin')
    }
}
```

Figure 5.29 - Sprite class for the coin object.

Back to the *GameState* class, write a method (Figure 5.30) that will be responsible for instantiating a *Phaser.Group*[33] of coins. The *GameState.createCoins*() method is a simple method that executes three steps: declare a field for counting the amount of coins collected by the player, create a group for the coins and call a tile map utility that will search the map data file (assets/level1.json) to create every coin defined. The information about the position of every coin and droid is stored into the map file. As already pointed in the section 0, the map of our game contains two layers: one for tiles and another for game objects. To automatically create all the coins from 'Objects Layer 1' defined in the map file, we just have to call *createFromObject*(). This method has a lot of parameters and we will focus on the most significant. The remaining parameters can be check on the Phaser documentation website[34]. The first parameter is the name of the layer where the objects data are stored inside the map file. The third parameter is the string key that was associated with the coin sprite sheet, previously cached in the *BootState.create*() method. The last two parameters are the references to the coins group and the Coin class.

```
createCoins() {
    this.coinCount = 0
    this.coins = this.game.add.group()
    this.map.createFromObjects('Object Layer 1', 71, 'coin', 0, true,
                               true, this.coins, Coin);
}
```

Figure 5.30 - Creating coin objects based on data stored in the map json file.

Let's add a call to the *GameState.createCoins*() inside the *GameState.create*() method. The order in which we add the game objects into the game defines their drawing order: the last one will be in front of all objects. It seems more interesting to put the coins visually between the map and the player sprite. So, let's put the call to *createCoins*() between *createTileMap*() and the Player class instantiation. The code is in Figure 5.31.

```
create() {
    // ...creation of the tile map...
    this.createCoins()
    // ...instantiation of the Player class...
... }
```

Figure 5.31 - Calling createCoins() within GameState.create() method.

[33] A Phaser.Group is a list with added conveniences to use with game objects. For more information, check the Phaser manual at https://phaser.io/docs/2.6.2/Phaser.Group.html. .Accessed on 25/08/2018.
[34] https://phaser.io/docs/2.6.2/index .Accessed on 25/08/2018.

The final step is to make the player collect the coins. To do this we will again rely on the physics engine and check if a collision will occur between the player and any coin from the group. Add the following line at the end of *GameState.update*() (Figure 5.32).

```
update() {
    ...
    this.game.physics.arcade.overlap(this.player, this.coins,
            this.collectCoin, null, this)
}
```

Figure 5.32 - Checking collisions between the player character and the group of coins.

The *overlap*() method in Figure 5.32 checks if an overlap of objects occurs and calls a callback if provided. We provided a callback to make the player collect the coin: *GameState.collectCoint*(). The method shown in Figure 5.33 receives a reference to the player and the coin that has overlapped. We then disable the coin with the *kill*() method, increase the counter of collected coins, update the HUD (see section 5.9) and check for victory condition (all coins were collected). For now, just comment this line as we are going to deal with this on the section 0. Don't forget to remove the comment later.

```
collectCoin(player, coin) {
    coin.kill()
    this.coinCount++;
    this.updateHud()
    if (this.coins.countLiving() == 0)
        this.gameWin()
}
```

Figure 5.33 - Callback to handle the collision between the player and a coin.

5.11. Player vs Droids

Up to now, we have the game world, the HUD, a full controllable player and coins to collect. Now, we need droids! The droids are flying robots that will make the player loose a life if touched. They have a fairly predictable behavior, moving from one position to another, in a loop. The implementation of the droids into the game is going follow a very similar approach to the coins. The steps we are going to make are the following: (1) declare a subclass of *Phaser.Sprite* for the Droid; (2) write a *createDroids*() method and call it inside *GameState.create*(); (3) add an overlap check between the player and the droids group in *GameState.update*() and write a callback to handle it.

The Droid class is shown in Figure 5.34. It must be implemented into the Droid.js file. The movement of the droid is made by a *Phaser.Tween*[35]. *Tweening* is a key process in all kinds of animation and a very common technique of movement and animation interpolation implemented on game engines (MÖLLER et al., 2018). The droid class creates a *Phaser.Tween* object that moves the sprite from its starting position to 200 pixels at left, and then back to start, in an infinite loop. Generally speaking, tweens are useful for moving objects and creating effects.

```
class Droid extends Phaser.Sprite {
```

[35] https://phaser.io/docs/2.6.2/Phaser.Tween.html. Accessed on 25/08/2018.

```
    constructor(game, x, y, asset) {
        super(game, x, y, asset)
        this.game.physics.arcade.enable(this)
        this.body.allowGravity = false      // droids won't fall
        this.body.immovable = true          // player can't push them
        this.body.setSize(25, 14, 3, 15);
        this.anchor.setTo(0.5, 0.5)
        this.animations.add('move', [0, 1, 2, 3], 10, true)
        this.animations.play('move')
        this.smoothed = false               // we want pixelated graphics
        // droid moves from one point to another in a loop
        this.game.add.tween(this)
            .to( { x: this.x - 200 }, 3000, "Quart.easeInOut")
            .to( { x: this.x }, 3000, "Quart.easeInOut")
            .loop(-1)
            .start();
    }
}
```

Figure 5.34 - Class for the Droid NPC character.

Next, we have to write the *GameState.createDroid*() method. Similar to *GameState.createCoins*(), the method will create the droids into a *Phaser.Group*, using the data of the 'Object Layer 1' available in the level map file. We are going to call the method after the coins creation in *GameState.create*(). The code is in Figure 5.35.

```
createDroids() {
    this.droids = this.game.add.group()
    this.map.createFromObjects('Object Layer 1', 73, 'droid', 0, true,
                               true, this.droids, Droid);
}
create() {
    // ...coins creation…
    this.createDroids()
... }
```

Figure 5.35 - GameState.createDroids() method.

Now, we need to check collisions between the players and the group of droids. To achieve this, add the following line to the end of *GameState.update*(). Every time the player overlaps a droid, the *GameState.hitDroid*() is called (Figure 5.36)

```
update() {
    ...
    this.game.physics.arcade.overlap(this.player, this.droids,
            this.hitDroid, null, this)
}
```

Figure 5.36 - Checking collisions between the player and the group of droids.

The collision between player and droids is handled by *GameState.hitDroid*(). The code is shown in Figure 5.37 and is executed only if the player is alive. The method

encompasses both the damage and the attack from the player to a droid. Essentially, if the player is falling and above the droid, it attacks and performs a jump. On the contrary, the player takes a hit. The *GameState.hitPlayer*() was presented in section 5.8.

```
hitDroid(player, droid) {
    if (!player.alive) return;
    // if player is falling and above droid -> bump
    if (player.body.velocity.y > 0 && player.body.y < droid.y) {
        droid.hit()
        player.jump(true) // true indicates it is an attack jump
    } else
        this.hitPlayer()
}
```

Figure 5.37 - Callback to handle a collision between the player and a droid.

The *Droid.hit* () method called in *GameState.hitDroid*() uses a *Phaser.tween* to create a simple effect of scaling up and fading out. When done, it kills the droid instance through the callback on the complete event. The code is shown in Figure 5.38.

```
hit() {
    this.body.enable = false // avoid a new collision while fading out
    this.game.add.tween(this.scale) // scale up
        .to( { y: 1.5, x: 1.5 }, 300)
        .start()
    this.game.add.tween(this)       // fade out
        .to( { alpha: 0 }, 300)
        .start()
        .onComplete.add(this.kill, this)
}
```

Figure 5.38 - Droid.hit() makes the droid fade out and destroy itself.

5.12. Win or Lose

To end our tutorial, we need to create two new states for the game: victory and failure. Victory will be reach if all coins are collected. Conversely, failure will arise if all player lives are lost. The methods that check these conditions were already presented throughout this tutorial. The section 5.8 has shown the method *GameState.hitPlayer*(), which calls *GameState.gameOver*() if the player is not alive. Analogously, the section 0 has presented the code for *GameOver.collectCoin*(), which calls *GameOver.gameWin*() if all coins are collected. The following code shows the aforementioned methods. They are very similar: each one essentially disables the camera from following the player and shows a text message with a simple effect created by tweenings. The tween also controls the duration of the message and calls *GameState.restartGame*() when completed. Then, the *restart*() method defines a fade out effect for the game camera and restarts the *GameState* when completed. With these methods, the game code is finished. The whole code is presented in Figure 5.39.

```
restartGame() {
    this.camera.fade(0x000000)
```

```
    this.camera.onFadeComplete.addOnce(()=>this.state.restart(true),this)
}
gameWin() {
    this.player.body.enable = false
    this.game.camera.follow(null)
    // show 'LEVEL CLEAR' with a scale tweening
    this.game.add.tween(this.hud.gamewin.scale)
        .to( { y: 1.5 }, 150)
        .to( { y: 0 }, 150, 'Linear', false, 3000)
        .start()
        .onComplete.add(this.restartGame, this)
}
gameOver() {
    this.game.camera.shake(0.005, 200)
    this.game.camera.follow(null) // can't follow a non-existing player
    // show 'GAME OVER' with a 'fade in' tweening
    this.game.add.tween(this.hud.gameover)
        .to( { alpha: 1 }, 150)
        .to( { alpha: 0 }, 150, 'Linear', false, 3000)
        .start()
        .onComplete.add(this.restartGame, this)
}
```

Figure 5.39 - Methods of the GameState class to manage victory and failure conditions.

6. The Final Game and Next Steps

To run the final result, just open the index.html on the Firefox browser and you're done. The complete game can be checked in Figure 6.1.

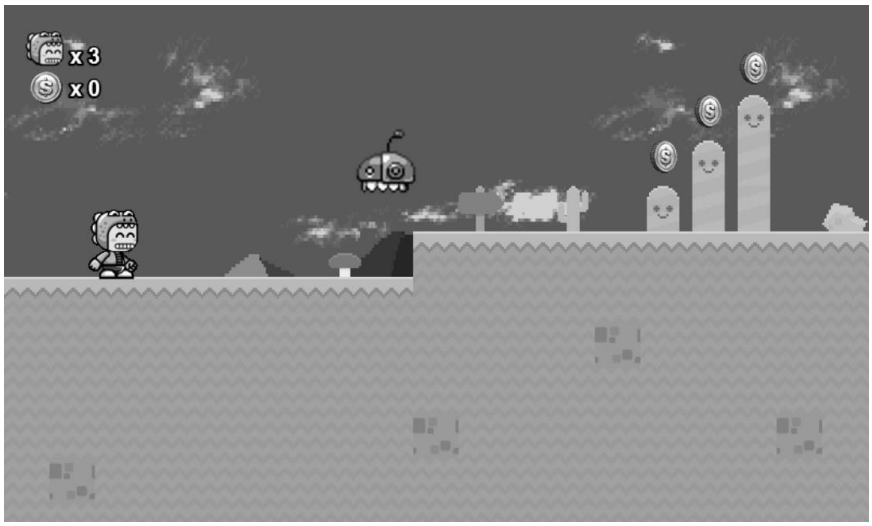


Figure 6.1 - The final game.

Overall, the Phaser framework implements a very easy to use game engine that can be employed to build any kind of 2D game. Facilities such as the ones shown along this tutorial, including the physics engine, tile maps, sprites and tweenings, makes Phaser

a very powerful tool that may be used by both hobbyists and professional game developers. While the performance of the Phaser games may not be comparable to the ones made with engines that employs languages such as C++ or C#, its open source nature and language flexibilities makes Phaser an versatile tool for smaller projects, improving productiveness and lowering production costs. Also, it is important to note that the performance of web games are directly related to the JavaScript engine used to run the code.

This tutorial has shown a gradual implementation of a simple game with the Phaser framework. While the game had the basic elements of a platformer, it's far from a complete commercial title. The size of a full-fledged game is not feasible for a tutorial and the complexity is beyond its scope. However, the game produced can be evolved to a most complex title that can be used as the starting point for a publishable game. YouTube, HTML development websites and, particularly, the Phaser official website are full of tutorials and code samples that can contribute to add great features to the game. Finally, the Cocoon[36] tool is amazing for packing a Phaser web game as a native PC or mobile game.

7. Reference List

BETHKE, Erik. Game development and production. Texas: Wordware Publishing, 2003.

CHANDLER, Heather M. The Game Production Handbook. 2nd ed. Hingham, Massachusetts: Infinity Science Press, 2009.

DUTTON, Fred. What is Indie? Eurogamer, 2012. Accessed in 25/05/2018. Link: https://www.eurogamer.net/articles/2012-04-16-what-is-indie

ESA, Entertainment Software Association. Essential Facts About the Computer and Videogame Industry. 2018. Accessed in 25/08/2018. Link: http://www.theesa.com/wp-content/uploads/2018/05/EF2018_FINAL.pdf

MADHAV, Sanjay. Game Programming in C++: Creating 3D Games. Addison-Wesley Professional, 2018.

MDN, Mozilla Developer Network. Tiles and tilemaps overview. 2018. Accessed in 25/05/2018. Link: https://developer.mozilla.org/en-US/docs/Games/Techniques/Tilemaps

MÖLLER, Tomas Akenine et al. Real-Time Rendering. 4th ed. CRC Press, 2018.

ROGERS, Scott. Level UP: The Guide to Great Videogame Design. Wiley Publication, 2010.

SHANKAR, Aditya Ravi. Pro HTML5 Games: Learn to Build your Own Games using HTML5 and JavaScript. 2nd ed. Bangalore, India: Apress, 2017.

WIJMAN, Tom. Mobile Revenues Account for More Than 50% of the Global Games Market as It Reaches $137.9 Billion in 2018. New Zoo, 2018. Accessed in 01/06/2018. Link: https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/

[36] https://cocoon.io/ .Accessed on 25/08/2018.

14. Capítulo 14

Autores:

Lucio Geronimo Valentin

Universidade Tecnológica Federal do Paraná (UTFPR) - Campo Mourão

email: lgvalentin@utfpr.edu.br

Marcos Silvano Almeida

Universidade Tecnológica Federal do Paraná (UTFPR) - Campo Mourão

email: marcossilvano@utfpr.edu.br

Capítulo 14

URI e HTTP, quase tudo passa por aqui!

Lucio Geronimo Valentin e Marcos Silvanos Almeida

Abstract

Currently on the Internet there is an immense framework of technologies that support both the backend and the frontend of the applications, but a protocol is almost ubiquitous, HTTP. Since its first usable version, the HTTP / 0.9 of 1991, the protocol has evolved to meet the new needs of the WEB. Knowing this protocol is essential to understand one of the main communication links of the WEB and to develop services that exploit its full potential without "reinventing the wheel". In this sense, this mini-course sequentially presents the implementation of a static HTTP server, able to obtain the resources requested by a URI and correctly identified by its MIME Type, following protocol specifications. RFCs are used as the main sources of information for implementation.

Resumo

Atualmente na Internet existe um imenso arcabouço de tecnologias que suportam tanto o backend quanto o frontend das aplicações, mas um protocolo é praticamente onipresente, o HTTP. Desde de sua primeira versão usável, a HTTP/0.9 de 1991, o protocolo evoluiu para atender às novas necessidades da WEB. Conhecer este protocolo é essencial para entender um dos principais elos de comunicação da WEB e desenvolver serviços que explorem todo seu potencial, sem "reinventar a roda". Neste sentido, este mini-curso apresenta de forma sequencial a implementação de um servidor HTTP estático, capaz de obter os recursos solicitados por uma URI e corretamente identificado por seu MIME Type, seguindo as especificações do protocolo. Como principais fontes de informação para a implementação são utilizadas as RFCs.

14.1. Introdução

Como a Internet pode ser definida? Usando ela própria, fazemos uma busca na Wikipedia [Internet, 2018] e encontramos um artigo que define a Internet como uma

rede mundial de computadores. Na verdade, se a Internet fosse somente uma rede de computadores, ela não teria muita utilidade, mas ela tornou-se uma rede de pessoas. Pessoas que fornecem e consomem informações e serviços, e hoje ela permeia o dia a dia das pessoas e de praticamente todas as profissões, sendo um fim ou um meio. Mas do ponto de vista técnico, a Internet realmente é uma rede de computadores que usa uma rígida estrutura de protocolos para troca de pacotes de dados entre os nós da rede. Na camada mais alta do protocolo, a camada de aplicação, estão os protocolos para a troca de mensagens mais específicas. Mais detalhes sobre os aspectos históricos e técnicos da Internet podem ser lido em [Internet, 2018].

Este capítulo apresenta alguns conceitos básicos que regem a infraestrutura da Internet e propõe a implementação de um servidor HTTP, seguindo as especificações da versão 1.1 deste protocolo. Os códigos apresentados utilizam a linguagem de programação Java e seguem uma sequência incremental de dificuldade e de funcionalidades, permitindo ao leitor entender os elementos primordiais que regem um servidor HTTP. O servidor apresentado neste capítulo comporta-se de maneira estática, respondendo a clientes HTTP e obtendo recursos também estáticos disponíveis no servidor. Não serão tratados aqui aspectos sobre recuperação de recursos dinamicamente construídos pelo servidor ou por módulos integrados a ele. Por ser um mini curso básico, as questões de segurança de um servidor HTTP serão somente introdutórias e questões mais complexas não serão integradas ao código do servidor exemplo.

Neste capítulo serão utilizadas principalmente a RFC3986 “Uniform Resource Identifier (URI): Generic Syntax” [Berners-Lee et al. 2005], a RFC2616 “Hypertext Transfer Protocol -- HTTP/1.1” [Fielding et al. 1999] e a RFC2045 “Multipurpose Internet Mail Extensions (MIME)” [Freed et al. 1996] para a implementação de um servidor HTTP básico. A RFC2616 foi posteriormente desmembrada e teve suas partes melhor detalhadas em outras RFCs, porém, ela apresenta o protocolo HTTP de uma forma consolidada e oferece um material inicial mais propício para um exercício pedagógico. O exercício proposto neste capítulo abordará a conexão entre dois nós para criar um fluxo de leitura e escrita que permite a troca de dados (bytes). Dentro deste fluxo, é necessário especificar o formato de mensagens que podem ser trocadas para que os nós se comuniquem efetivamente, neste ponto será utilizado o protocolo HTTP, sua estrutura e seus métodos. Será apresentado um código básico de um servidor HTTP e depois serão discutidos aspectos para tornar o servidor mais dinâmico e seguro.

No decorrer do capítulo, são propostos alguns desafios que oferecem uma maneira para que o leitor aprofunde-se nos detalhes do servidor. Sua implementação exige uma leitura mais detalhada das partes da RFC relacionada ao tema do desafio. Nenhuma restrição técnica é indicada para a conclusão do desafio. O leitor poderá implementar a funcionadade proposta de acordo com seu nível de conhecimento na linguagem de programação e no paradigma orientado a objetos.

14.2. Conectando Dois Nós

Quando se trata de comunicação, a definição de padrões torna-se uma tarefa essencial para se evitar o caos e permitir um desenvolvimento rápido de aplicações ou soluções por diferentes grupos ou fornecedores. Neste contexto, em 1986 um grupo de 21 pesquisadores financiados pelo governo americano deu início às reuniões da *Internet Engineering Task Force* (IETF) [Internet Engineering Task Force, 2018]. Desde então,

esta organização mantém reuniões e grupos de discussões sobre os padrões a serem adotados nas diversas áreas da Internet. Como resultado do trabalho desta organização são compilados documentos técnicos chamados de *Request for Comments* (RFC) [Request For Comments, 2016]. O início deste formato é datado em 1969, quando pesquisadores do projeto ARPANET [ARPANET, 2018], a mãe da Internet, compartilhavam seus documentos em texto sem formatação. As RFCs são numeradas sequencialmente desde de 7 de abril de 1969 e possuem alguns metadados para indicar inclusive, hierarquia e associações entre os documentos.

As RFCs definem todos os detalhes técnicos da Internet, envolvendo aplicações, segurança, transporte, roteamento e outros aspectos que envolvem a rede mundial. Os aspectos da própria IETF foram definidos inicialmente na RFC1391 e agora está na RFC6722. Os documentos podem ser acessados através da página da organização no endeço https://www.ietf.org/ e estão disponíveis nos formatos ASCII, HTML e PDF.

A Internet é uma rede composta por clientes, hospedeiros e roteadores intermediários, além de serviços e protocolos [He et al., 2009]. No entanto, para os nós nos extremos de uma conexão é dada a ilusão que existe um canal direto interligando-os. Desta forma, quando um cliente solicita a conexão com um servidor, toda a infraestrutura da rede ficará transparente e os pacotes de dados serão trocados entre os nós como se eles estivessem um ao lado do outro.

14.2.1. Cliente e Servidor

A conexão entre dois nós só será possível se um nó estiver esperando uma conexão e o outro nó solicitar uma conexão, ou seja, sempre haverá um servidor e um cliente envolvidos.

Desconsiderando alguns aspectos de encapsulamento e roteamento, os pacotes dos protocolos TCP/IP [TCP/IP, 2018] possuem um endereço e uma porta de destino. Assim, para realizar a conexão, o cliente basicamente deverá enviar um pacote para o endereço IP do servidor com a identificação da porta de entrada que o servidor está esperando uma conexão. As camadas intermediárias de transporte do protocolo ficarão encarregadas de entregar o pacote ao destino.

Conhecer os endereços IP instantâneos dos servidores não é uma tarefa fácil. Além do fato que eles podem mudar constantemente, um conjunto numérico não é muito intuitivo e de fácil memorização. Para auxiliar os clientes, uma rede disponibiliza um sistema de nomes de domínio (DNS) [Sistema de Nomes de Domínio, 2018] que pode ser consultado para traduzir um nome amigável em um endereço IP. Desta forma, ao invés de tentar acessar um servidor pelo endereço IP 172.217.30.110, é possível acessar usando “google.com”.

14.2.2. Problemas das Rotas

É importante considerar que no meio desta comunicação possam existir firewalls e outras restrições de segurança ou de estrutura de rede que tornam os nós logicamente invisíveis um ao outro ou inacessíveis [Firewall, 2018]. Estes aspectos não serão tratados neste capítulo. De qualquer forma, quando dois nós conectam-se, dois fluxos de dados são fornecidos para cada nó, um fluxo de entrada e um fluxo de saída. Os fluxos são independentes e é necessário definir algum protocolo de comunicação para

sincronizar a troca de mensagens, pois tecnicamente, os nós podem ler e escrever simultaneamente em uma conexão.

14.2.3. URI, Tudo Passa Por Aqui!

A Internet não é somente uma rede de computadores, mas um conjunto de padrões que permitem a troca de informações e o acesso a sistemas de diversas naturezas. A conexão entre os computadores na Internet é possível devido a identificação que cada nó recebe na rede. Esta identificação é conhecida como endereço IP (Internet Protocol) e é uma informação técnica. Um servidor de nome de domínio (DNS) mantem um serviço que a rede utiliza para traduzir nomes em endereços IP. Desta forma, é possível acessar servidores na rede usando nomes mais amigáveis. Porém, desde o início da Internet houve uma preocupação de como endereçar os diversos recursos disponíveis em diferentes protocolos na rede e possibilitar uma integração padronizada entre aplicações e recursos. Destas discussões iniciais surgiu o identificador universal de recursos (URI) [Berners-Lee, 1994]. Mais tarde, a RFC3986 "Uniform Resource Identifier (URI): Generic Syntax" [Berners-Lee et al., 2005] redefine o significado da sigla URI e comenta sobre seus aspectos conceituais e técnicos, inclusive sobre os significados de uniforme, recurso e identificador. O URI é dividido atualmente em duas categorias, a categoria de localizador uniforme de recursos (URL), que trata os aspectos de localização e endereçamento de um recurso na rede, como por exemplo http://cleilaclo2018.mackenzie.br/index.php, e outra categoria de nome uniforme de recurso (URN), que representa mais um aspecto de padronização para identificar diversos recursos indexados, como por exemplo DOI 10.17487/RFC3986, ISBN, ISSN, entre outros.

Um URI é composto por letras do alfabeto latino básico, dígitos e alguns poucos caracteres especiais. Ele possui uma organização hierárquica com maior significância da esquerda para a direita. E suas partes principais são separadas por dois pontos (":"), arroba ("@"), barra comum ("/"), interrogação ("?") e cerquilha ("#"). A RFC3986 [Berners-Lee et al. 2005] apresenta a linguagem formal que define a sintaxe do URI, usando a forma aumentada de Backus-Naur. Um URI é composto basicamente por um esquema, uma autoridade, um caminho, uma consulta e um fragmento. Uma forma mais reduzida da sintaxe do URI é apresentada a seguir:

```
URI = esquema":"["//"AUTORIDADE] ["/"caminho] ["?"consulta] ["#"fragmento]
```

A Figura 1 mostra o exemplo apresentado na própria RFC3986.

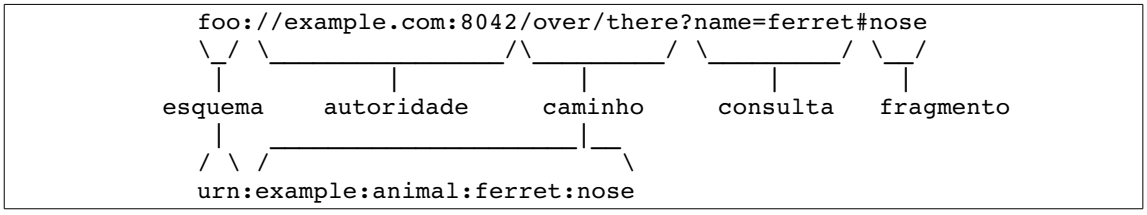


Figura 1: Partes de um URI.

Um esquema define como as demais partes do URI deverão ser interpretadas. Geralmente um esquema está ligado a um protocolo ou a uma aplicação específica que

será capaz de decodificar o restante do URI, e obter o recurso identificado. O esquema não é sensível ao caso, embora recomenda-se usar sempre letras minúsculas. Os esquemas devem ser registrados na IANNA [Internet Assigned Numbers Authority, 2017]. Alguns esquemas bastante conhecidos são http, https, ldap, mailto, tel, telnet, entre outros. Uma lista completa dos schemas atualmente registrados pode ser obtida acessando a página https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml.

Uma autoridade representa o local que detem a guarda do recurso que se deseja acessar. Tecnicamente são representados pelos dados sobre usuário, servidor e porta de acesso. O servidor pode ser expresso pelo seu nome ou pelo seu endereço e a porta de acesso pode ser omitida, uma vez que todos os esquema especificam uma porta padrão em caso de omissão. De forma simplificada, uma autoridade segue a seguinte sintaxe:

```
AUTORIDADE   = [ [usuário ":" senha] "@" ] servidor [ ":" porta ]
```

Um caminho consiste em uma sequência hierárquica de segmentos separados pelo caractere "/". Os segmentos são semelhantes aos diretórios, apesar de não ser uma regra. Inclusive é possível usar os símbolos "." e ".." para referenciar caminhos hierárquicos. Alguns exemplos de caminhos são "/app1/module1/view1.html" e "/module1/actions/".

Uma consulta oferece dados não hierárquicos no formato "chave=valor" que possuam alguma relação com o recurso que está sendo identificado pelo URI. O esquema pode suportar a ausência do valor e também, vários pares separados pelo caracter "&". Um exemplo de uma consulta é "?appId=1&moduleId=1&viewId=1.

Um fragmento permite identificar parte do recurso que foi obtido. Assim, após o recurso ser obtido, o fragmento pode ser localizado e extraído, exibido ou destacado. Um exemplo de fragmento é "#footer".

O URI é um padrão bastante versátil que pode ser adaptado para várias situações. Em gerenciadores de arquivos como Nemo e Nautilus, em suas versões 3 e 1, respectivamente, é possível acessar arquivos em servidores através do URI:

```
ssh://seuUsuario:suaSenha@ipOutroComputador:22/home/seuUsuario
```

O outro computador deve possuir um serviço SSH ativo e com suporte ao protocolo *Secure File Transfer Protocolo* (SFTP). A porta 22 é a porta padrão do esquema ssh, mas foi adicionada ao exemplo para mostrar como pode ser alterada, caso o serviço SSH esteja configurado em outra porta no servidor.

Outro exemplo de utilização da versatilidade do URI, é a conexão entre aplicativos usando as associações de esquemas. No URI a seguir, o esquema é o whatsapp, o sistema operacional poderá abrir o URI com o aplicativo atualmente associado a este esquema:

```
whatsapp://chat?code=BeKDAdJY2ml3e0UW2azNTl
```

Umar forma simples de decodificar as partes de um URL é usando uma expressão regular:

```
^(([^:/?#]+):)?(\/\/([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?
```

```
http://www.ics.uci.edu/pub/ietf/uri/#Related
```

Praticamente todas as linguagens de programação implementam suporte à expressão regular, seja de forma nativa ou por bibliotecas de terceiros. Existem ainda ferramentas

online como o site regexr.com que permitem manipular expressões regulares e verificar seus comportamentos sobre um determinado conteúdo.

14.2.4 Implementando um Modelo Cliente-Servidor

Neste capítulo será utilizada a linguagem Java em sua versão 7, mas qualquer outra linguagem pode ser usada para implementar os exemplos, pois praticamente todas as linguagens fornecem suporte a soquetes de rede TCP/IP e à programação multitarefas [Tanenbaum, 2003].

Os códigos fonte estão disponíveis no GitHub através do link:

```
https://github.com/lgvalent/httpServer
```

EchoServer monotarefa

No modelo cliente-servidor existem duas aplicações rodando. Uma aguardando requisições e outra realizando as requisições. A primeira aplicação a ser implementada neste capítulo faz o papel de servidor. Ele cria um soquete na porta 5555 e fica aguardando uma conexão. Ao receber a conexão, o servidor lê o que o cliente envia e devolve para o cliente a mesma mensagem acrescentando a palavra ECHO no início. A Figura 2 apresenta o código-fonte da classe servidora. Por questões de segurança, alguns sistemas operacionais podem exigir privilégio de administrador para permitir que uma aplicação aguarde por conexões em portas inferiores a 1024.

```
import java.io.*;
import java.net.*;

public class Server {
   public static void main(String args[]) throws Exception {
       System.out.println("DEBUG: Running...");
       ServerSocket ss = new ServerSocket(5555);
       Socket socket = ss.accept();
       System.out.println("DEBUG: Connected...");
       BufferedReader in  = new BufferedReader(
             new InputStreamReader(socket.getInputStream()));
       DataOutputStream out = new DataOutputStream(s.getOutputStream());
       String msg = in.readLine();
       System.out.println("DEBUG: Processing '" + msg + "'...");
       msg = "ECHO: " + msg;
       out.writeBytes(msg + "\n");
       ss.close();
   }
}
```

Figura 2: Código-fonte do servidor monotarefa de eco (em Java).

A aplicação cliente apresentada na Figura 3 deve criar um soquete e conectar-se ao endereço do servidor e à porta 5555. Ao conseguir estabelecer a conexão, o cliente lê alguma entrada do teclado, envia o texto para o servidor processar e exibe o resultado.

```
public class Client {
   public static void main(String argv[]) throws Exception {

      Socket s = new Socket("localhost", 5555);
      BufferedReader in = new BufferedReader(
                                 new InputStreamReader(s.getInputStream()));
      DataOutputStream out = new DataOutputStream(s.getOutputStream());
      System.out.print("DEBUG: Connected... ");
      System.out.println("Write something to receive server ECHO:");
      BufferedReader keyboard = new BufferedReader(
                                       new InputStreamReader(System.in));
      String msg = keyboard.readLine();

      System.out.println("DEBUG: Sending...");
      out.writeBytes(msg + "\n");

      System.out.println("DEBUG: Receiving...");
      msg = in.readLine();
      System.out.println(msg);
      s.close();
   }
}
```

Figura 3: Código-fonte do cliente do servidor de eco (em Java).

Nos códigos anteriores foram inseridas várias instruções para exibição de mensagens de DEBUG na saída. Tanto o servidor quanto o cliente processam apenas uma operação durante a conexão e finalizam sua execução. Nenhum suporte multitarefas foi implementado.

EchoServer multitarefa

A execução do código do servidor fica parada, dormindo, na linha em que aguarda uma nova conexão. Assim que uma nova conexão é recebida, o processo do servidor é acordado para tratar a entrada e saída da conexão e fica ocupado com isto até o fim do processamento. É possível criar uma tarefa para tratar cada requisição e assim permitir que o servidor atenda ao maior número possível de clientes simultâneamente, e aproveite os múltiplos núcleos de processamento disponíveis no hardware. Utilizando o conceito de tarefas, o código será alterado para que o servidor aceite a conexão e a processe fora da tarefa principal (*thread main*) do programa.

Os códigos a seguir serão incrementais e apresentarão somente as partes das classes que estarão em discussão.

Primeiramente é criada uma classe *ServerWorker,* apresentada na Figura 4, que implementa a interface *Runnable* e é composta por um atributo do tipo *Socket* que é fornecido durante sua construção. O código do método *run* é semelhante ao código do método *main* apresentado na Figura 2.

```
public class ServerWorker implements Runnable {
   private Socket socket;

   public ServerWorker(Socket s) {
      this.socket = s;
   }

   public void run() {
      try {
         BufferedReader in = new BufferedReader(new InputStreamReader(
                                                socket.getInputStream()));
         DataOutputStream out = new DataOutputStream(
                                                socket.getOutputStream());
         String msg;
         msg = in.readLine();
         System.out.println("DEBUG: Processing '" + msg + "'...");
         msg = "ECHO: " + msg;
         out.writeBytes(msg + "\n");
         socket.close();
      } catch (IOException e) {
         throw new RuntimeException(e);
      }
   }
}
```

Figura 4: Código-fonte da tarefa geradora de eco (em Java).

A classe Server, apresentada Figura 5, deve ser alterada para que ela aceite uma conexão e delegue o tratamento a outra classe, a *ServerWorker*. O código fica extremamente simples, porém, neste momento não é feito nenhum controle adicional sobre cada tarefa para saber quantas tarefas estão ativas, a quanto tempo uma tarefa está inativa ou outros controles que um servidor poderia fornecer.

Adicionando o protocolo de comunicação

```
public class Server {
   public static void main(String args[]) throws Exception {
      ServerSocket ss = new ServerSocket(5555);
      while(true){
         Socket s = ss.accept();
         new Thread(new ServerWorker(s)).run();
      }
   }
}
```

Figura 5: Código-fonte do servidor multitarefa de eco (em Java).

No exemplo a seguir será implementada uma calculadora simples que suporta as operações de soma e subtração com números inteiros. A calculadora inicialmente possui um resultado igual a zero. O cliente deve enviar uma operação e um operando que serão processados pela calculadora usando o resultado atualmente armazenado. Para cada nova mensagem do cliente, o servidor retorna uma mensagem com o resultado instantâneo. O servidor deve ser capaz de processar várias operações em uma mesma conexão até que ela seja fechada. A cada nova conexão, uma nova instância de cálculos deve ser criada.

Para cumprir com os objetivos do parágrafo anterior, o cliente e o servidor terão que estabelecer um padrão para a troca das mensagens, sem o qual a comunicação seria caótica. Este padrão é chamado de protocolo e especifica o formato e a sequência das

mensagens que deverão ser trocadas entre servidor e o cliente. O protocolo que será usado neste exemplo possui duas mensagens para a requisição e uma mensagem para a resposta. A primeira mensagem de requisição é uma sequência de caracteres com a palavra 'RESET\r\n' que fará com que o servidor inicie uma nova sequência de cálculos. A segunda mensagem de requisição possui o formato '<OP>:<VAL>\r\n', onde '<OP>' pode assumir 'ADD' ou 'SUB', e '<VAL>' representa um valor numérico inteiro. A mensagem de resposta será 'RESULT:<VAL>\r\n', neste caso, o valor resultado pode ser uma sequência de dígitos ou ainda uma mensagem de erro.

Em programação orientada a objetos a correta definição da responsabilidade das classes é uma das atividades mais importantes do paradigma. Assim, uma nova classe deve ser criada para ficar responsável pela manipulação do protocolo, deixando o protocolo desacoplado do restante do código que é responsável pelo gerenciamento das conexões e dos fluxos de entrada e saída. A Figura 6 apresenta o código da classe *CalculatorProtocol*. Ela possui um método que recebe um argumento do tipo String e retorna também uma *String*. Não há referência a nenhuma outra classe. A implementação de testes unitários para esta classe será uma tarefa relativamente fácil, pois não será necessário criar nenhum contexto de Cliente-Servidor. Desta forma, os princípios de alta coesão e baixo acoplamento também são atingidos nesta classe e nas demais.

```
public class CalculatorProtocol {
   private int result = 0;

   public String processMsg(String msg){
      String parts[] = msg.split(":");
      String command = parts[0];
      int value = parts.length > 1 ? Integer.parseInt(parts[1]) : 0;

      switch (command) {
         case "RESET": result = 0; break;
         case "ADD": result += value; break;
         case "SUB": result -= value; break;
         default:
            return "RESULT:ProtocolError. Invalid command. Cannot handle '" +
                                                                  msg + "'\r\n";
      }

      return "RESULT:" + result + "\r\n";
   }
}
```

Figura 6: Código-fonte do protocolo da calculadora básica (em Java).

A classe ServerWorker foi alterada conforme é apresentado na Figura 7, para manipular as questões de conexão, obter uma linha e delegar a interpretação da mensagem à class *CalculatorProtocol*. Como a calculadora suporta várias operações em uma mesma conexão, então o mesmo objeto "cp" é usado para manter o contexto das operações para o cliente atual.

```
public class ServerWorker implements Runnable {
/*... code omitted ... */
   public void run() {
      try {
         CalculatorProtocol cp = new CalculatorProtocol();
         BufferedReader in = new BufferedReader(new InputStreamReader(
                                                socket.getInputStream()));
         DataOutputStream out = new DataOutputStream(
                                                socket.getOutputStream());
         String msg;
         while ((msg = in.readLine()) != null){//Is socket connected?
             System.out.println("DEBUG: Processing '" + msg + "'...");
             msg = cp.processMsg(msg);
             out.writeBytes(msg);
         }
         socket.close();
      } catch (IOException e) {
         throw new RuntimeException(e);
      }
   }
}
```

Figura 7: Código-fonte referenciando a classe CalculatorProtocol (em Java).

Desafio: Acrescente novas operações à calculadora e observe quantas classes e métodos deverão ser alterados para sua aplicação suportar tal funcionalidade nova.

14.3. Protocolo HTTP

O protocolo para transferência de hipertexto (HTTP) foi desenvolvido em 1989 por Tim Berners-Lee [Hypertext Transfer Protocol, 2017]. Ele funciona como um protocolo de requisições e respostas em um ambiente Cliente-Servidor. Suas versões 0.9, 1.0, 1.1 e 2.0 são datadas em 1991, 1996, 1997 e 2015, respectivamente. E atualmente é o principal protocolo para acesso de recursos na Internet. A RFC2616 [Fielding et al. 1999] apresenta a especificação completa da versão 1.1 do protocolo e servirá de base para desenvolvimento do servidor apresentado neste capítulo. A versão 2.0 do protocolo surgiu com o objetivo principal de reduzir a latência e aumentar a velocidade de carregamento dos recursos, incluindo por exemplo, compressão nos dados dos cabeçalhos. Contudo, ela é retrocompatível com a versão 1.1.

Para implementar um servidor HTTP as especificações do protocolo devem ser seguidas à risca e a melhor fonte de informações neste caso, é a própria RFC que define o protocolo. Desta forma, a leitura completa da RFC2616 [Fielding et al. 1999] é indispensável para garantir que todos os aspectos do protocolo estejam sendo satisfeitos pela implementação. Apesar de uma RFC ser bastante técnica e bem organizada, seu texto é de fácil compreensão. A seção dois do documento apresenta as regras básicas utilizadas na especificação das mensagens. Vale ressaltar que existem declarações que são marcadas no texto com termos que indica se um aspectos do protocolo é requerido, recomendado ou opcional. A RFC2119 [Bradner et al. 1997] esclarece como os termos devem ser interpretados corretamente.

14.3.1. Sintaxe de uma Mensagem HTTP

O protocolo HTTP define o mesmo formato para mensagem de requisição e de resposta. Toda mensagem possui uma primeira linha para indicar o tipo da mensagem, porém,

podem haver linhas vazias antes desta linha. A mensagem possuem também um cabeçalho e um corpo que são separados por uma linha em branco, mais especificamente pelos caracteres CARRIER_RETURN (CR ou #13) e LINE_FEED (LF ou #10).

Mensagem de requisição

Uma mensagem de requisição deve indicar em sua primeira linha qual o método HTTP solicitado, o caminho do recurso e a versão do protocolo que será usada no restante da mensagem. O caminho[1] pode incluir a parte de consulta do URI, porém, o fragmento da URI não é enviado para o servidor. Todos os campos da primeira linha são separados por espaços em branco (SP):

```
Requisição-1ªLinha = Método SP Caminho SP Versão-HTTP CRLF
```

Observe o seguinte URI digitado na barra de localização de um navegador WEB:

```
http://blog.valentin.com.br/2017/02/paradigma-de-programacao-clicar-
                        e.html#comment-post-message
```

Ele gerará uma requisição onde a primeira linha é mostrada a segui:

```
GET /2017/02/paradigma-de-programacao-clicar-e.html HTTP/1.1CRLF
```

Os métodos GET, POST e HEAD foram especificados na versão 1.0 do protocolo e os métodos PUT, DELETE, TRACE, CONNECT e OPTIONS, na versão 1.1. Os métodos GET, HEAD e TRACE são métodos considerados seguros, pois não alteram o estado do servidor e devem ser utilizados somente para obter os recursos. Os métodos POST, PUT e DELETE devem ser utilizados para alterar o conteúdo de um recurso no servidor. Apesar deste conjunto de métodos, os clientes HTTP atuais usam basicamente GET e POST para executar suas tarefas. Porém, é importante observar a correta utilização dos métodos para evitar problemas com conteúdos em cache.

Conforme descrito na RFC2616, no cabeçalho de uma requisição podem ser fornecidos os seguintes campos:

```
   Accept, Accept-Charset, Accept-Encoding, Accept-Language, Authorization,
Expect, From, Host, If-Match, If-Modified-Since, If-None-Match, If-Range, If-
   Unmodified-Since, Max-Forwards, Proxy-Authorization, Range, Referer, TE,
                                  User-Agent
```

Mensagem de resposta

A mensagem de resposta será gerada pelo servidor após o processamento da requisição, indicando se o recurso existe e se seu conteúdo pode ser manipulado. A primeira linha de uma resposta contém a versão do protocolo, o código de retorno e uma frase para humanos lerem.

```
Resposta-1ªLinha = Versão-HTTP SP CódigoDeRetorno SP FraseCRLF
```

O código de retorno possui três dígitos que são separados por categorias conforme o primeiro dígito.

Códigos começados por 1xx são respostas informacionais usadas em requisições muito longas para avisar o cliente que está tudo bem até agora com a requisição e que ele pode

1 O caminho pode ser absoluto, incluindo a parte de autoridade do URI, se a requisição estiver sendo realizada através de um proxy. Caso contrário, ele será relativo.

continuar enviando o corpo da mensagem. Ou ainda, para permitir a troca de protocolo da resposta se o cliente indicou tal compatibilidade.

Códigos começados por 2xx indicam que a solicitação foi atendida com sucesso e que o resultado esperado estará no corpo da mensagem. O código 200 é o mais frequente nesta categoria.

Códigos começados por 3xx indicam que a solicitação está correta, mas que o recurso mudou de lugar ou é o mesmo desde a última requisição. Nesta categoria de resposta, o cliente deverá efetuar alguma outra ação para obter o conteúdo do recurso desejado, seja direcionando-se para outra URL ou acessando seu conteúdo em um cache local. Os códigos 301 e 307 são bastante utilizados para redirecionar os clientes de um domínio para outro domínio.

Códigos começados por 4xx indicam que a requisição não pode ser processada, seja por erro na solicitação, seja por restrições do recurso. O código 404 indica, por exemplo, que o recurso ou o caminho não foi encontrado no servidor.

Códigos começados por 5xx indicam problemas com o servidor e não com a requisição. O código 500 é o mais frequente e indica que ocorreu um erro interno. Geralmente exibido quando um site tentar acessar um banco de dados que não está disponível e o programador não implementou a rotina de exceção para este caso. Assim, uma pilha de execução em um HTML geralmente é retornada para o cliente, para ajudar na identificação do erro.

No cabeçalho de uma resposta podem ser fornecidos os seguintes campos:

```
Accept-Ranges, Age, ETag, Location, Proxy-Authenticate, Retry-After, Server,
                            Vary, WWW-Authenticate
```

O HTTP provê mecanismos de autenticação, controle de cache e de sessão que também definem campos específicos no cabeçalho da mensagem. Além disso, o HTTP possui vários metadados sobre a entidade que será transferida no corpo da mensagem, auxiliando o cliente a tratar a mensagem de resposta mesmo antes de recebê-la por completo. Eles são os seguintes:

```
Allow, Content-Encoding, Content-Language, Content-Length, Content-Location,
Content-MD5, Content-Range, Content-Type, Expires, Last-Modified
```

14.3.2 Implementando o Primeiro Servidor HTTP

Inicialmente será desenvolvido um servidor que atende a uma requisição do método GET para qualquer recurso, e responde positivamente sem maiores tratamentos e com um conteúdo fictício.

Olhando os códigos anteriores, parece que uma nova classe *HttpProtocol* deverá ser implementada para substituir a antiga classe *CalculatorProtocol*. Porém, observe que no protocolo da calculadora, cada mensagem era composta por uma linha. Já no HTTP, cada mensagem possui uma estrutura definida com cabeçalho e corpo. Ou seja, a mensagem não será um objeto da classe String, mas de uma classe que abstrai os dados de uma requisição. Um bom nome para esta classe é *HttpRequest* e, conforme o quê foi descrito anteriormente, seus atributos devem ser um método HTTP, um caminho, a versão do protocolo, um cabeçalho com pares nome-valor e um corpo de bytes.

A Figura 8 apresenta a classe *HttpRequest*. Não foram utilizados os métodos acessores para simplificar o exemplo. E como o código inicial trata somente o método GET, que por definição não tem dados no corpo da mensagem, então a classe por enquanto não terá uma atributo para armazenar os dados do corpo da requisição. Também foi implementado um método construtor que possibilita extrair os dados do objeto a partir de um fluxo de entrada de dados. Alguns métodos privados auxiliares foram acrescentados para manter as boas práticas de programação.

```
public class HttpRequest {
   static enum HttpMethods {
      GET, POST, HEAD, DELETE, TRACE, CONNECT, OPTIONS
   };

   public HttpMethods httpMethod;
   public String path;
   public byte httpMajorVersion;
   public byte httpMinorVersion;

   public Map<String, String> headerFields = new HashMap<String, String>();

   public HttpRequest(InputStream input) throws IOException {
      BufferedReader br = new BufferedReader(new InputStreamReader(input));
      readFirstLine(br);
      readHeader(br);
   }

   private void readFirstLine(BufferedReader reader) throws IOException {
      String firstLine = reader.readLine();

      String[] parts = firstLine.split(" ");
      httpMethod = HttpMethods.valueOf(parts[0]);
      path = parts[1];
      // HTTP/1.1
      httpMajorVersion = Byte.parseByte("" + parts[2].charAt(5));
      httpMinorVersion = Byte.parseByte("" + parts[2].charAt(7));
   }

   private void readHeader(BufferedReader reader) throws IOException {
      String line;
      while (!(line = reader.readLine()).equals("")) {
         String[] parts = line.split(":");
         String key = parts[0];
         String value = parts[1].trim(); //Remove SP. LWS not supported yet!
         headerFields.put(key, value);
      }
   }
}
```

Figura 8: Código-fonte que abstrai uma requisição HTTP (em Java).

Os métodos HTTP foram abstraídos em um enumerador chamado HttpMethods. Esta prática facilita futuras verificações sobre o método e a versão do protocolo que estão sendo manipulados na requisição e resposta.

Agora é necessária uma classe que receba um objeto do tipo *HttpRequest* e produza uma resposta, ou seja, produza um objeto do tipo *HttpResponse*. Como foi visto, a mensagem de resposta do protocolo HTTP é bem parecida com a mensagem de requisição, somente a primeira linha é diferente. A Figura 9 apresenta a abstração da resposta em uma classe *HttpResponse*. Assim como na requisição, alguns métodos privados

auxiliares foram implementados para dividir a responsabilidade do método write que possibilita que o conteúdo da resposta seja escrito em algum fluxo de saída.

```
public class HttpResponse {
   public byte httpMajorVersion = 1;
   public byte httpMinorVersion = 1;
   public String statusCode;
   public String reasonPhrase;

   public Map<String, String> headerFields = new HashMap<String, String>();

   public byte[] body;

   public void write(OutputStream output) throws IOException {
      PrintWriter pw = new PrintWriter(output);
      writeFirstLine(pw);
      writeHeader(pw);
      pw.write("\r\n");
      pw.flush();
      if(body != null)
         output.write(body);
   }

   private void writeFirstLine(PrintWriter writer) throws IOException{
      writer.append("HTTP").append("/"+httpMajorVersion)
                                              .append("."+httpMinorVersion);
      writer.write(' ');
      writer.print(statusCode);
      writer.write(' ');
      writer.print(reasonPhrase);
      writer.print("\r\n");
   }

   private void writeHeader(PrintWriter writer) throws IOException{
      for(Entry<String, String> entry: headerFields.entrySet()){
         writer.print(entry.getKey());
         writer.print(": ");
         writer.print(entry.getValue());
         writer.print("\r\n");
      }
   }
}
```

Figura 9: Código-fonte que abstrai uma resposta HTTP (em Java).

As classes HttpRequest e *HttpResponse* possuem atributos semelhantes. Isto se deve pela própria natureza das classes, pois ambas são classes que representam mensagens HTTP. Uma boa prática seria abstrair uma classe *HttpMessage* com os atributos que se repetem, e então, especializar as classes de requisição e resposta para herdarem os atributos comuns e adicionarem seus atributos particulares. Por enquanto, as classes ficarão como estão para seguir o raciocínio de processamento das mensagens e depois, futuras otimizações poderão ser realizadas.

Agora que as classes *HttpRequest* e *HttpResponse* estão prontas, será implementada a primeira versão da classe *HttpProtocol*. A versão mostrada na Figura 10 possui um método que recebe um objeto de requisição e retorna um objeto de resposta. O método *processMsg* verifica se a requisição é do tipo GET e compõe uma resposta sem muitos metadados e dados. Observe que qualquer método diferente de GET resultará em uma resposta 501 indicando que o servidor ainda não suporta a requisição.

```
public class HttpProtocol {

   public HttpResponse processMsg(HttpRequest msg) {
      HttpResponse response = new HttpResponse();

      switch (msg.httpMethod) {
         case GET:
            response.statusCode = "200";
            response.reasonPhrase = "Ok";
            response.body = "I'm almost a HTTP server!".getBytes();
            response.reasonPhrase = "Ok";
            break;
         default:
            response.statusCode = "501";
            response.reasonPhrase = "Not Implemented";
         }

         return response;
   }
}
```

Figura 10: Código-fonte que orquestra o protocolo HTTP (em Java).

Voltando ao código da classe ServerWorker, esta classe será alterada de forma que ela prepare um objeto da classe *HttpRequest* e despache para uma classe *HttpProtocol*, que responsável por tratar uma requisição do protocolo HTTP, como mostra a Figura 11.

```
public class ServerWorker implements Runnable {
   private Socket socket;
   public ServerWorker(Socket s) {
      this.socket = s;
   }

   public void run() {
      try {
         HttpProtocol httpProtocol = new HttpProtocol();

         HttpRequest request = new HttpRequest(socket.getInputStream());

         HttpResponse response = httpProtocol.processMsg(request);

         response.write(socket.getOutputStream());

         socket.close();
      } catch (IOException e) {
         throw new RuntimeException(e);
      }
   }
}
```

Figura 11: Código-fonte da tarefa que trata uma requisição ao servidor (em Java).

É importante observar que os códigos das classes vão ficando extremamente coesos e simples, uma vez que cada método possui uma responsabilidade bem definida. Resta agora implementar um cliente para testar o servidor. A Figura 12 apresenta dois testes, um para o método GET e outro para o método POST, ambos operam sobre o mesmo recurso. Espera-se que a atual implementação do servidor responda 200 para o primeiro teste e 501 para o segundo teste.

```
public class Client {
   public static void main(String argv[]) throws Exception {
      Socket s = new Socket("localhost", 5555);
      BufferedReader in = new BufferedReader(new InputStreamReader(
                                                    s.getInputStream()));
      DataOutputStream out = new DataOutputStream(s.getOutputStream());
      out.writeBytes("GET /test.html HTTP/1.1\r\n");
      out.writeBytes("\r\n");
      String msg;
      while ((msg = in.readLine()) != null)
         System.out.println(msg);
      s.close();

      s = new Socket("localhost", 5555);
      in = new BufferedReader(new InputStreamReader(s.getInputStream()));
      out = new DataOutputStream(s.getOutputStream());
      out.writeBytes("POST /test.html HTTP/1.1\r\n");
      out.writeBytes("\r\n");
      while ((msg = in.readLine()) != null)
         System.out.println(msg);
      s.close();
   }
}
```

Figura 12: Código-fonte do cliente HTTP para teste do servidor (em Java).

A saída esperada para estes testes é apresentada na Figura 13:

```
HTTP/1.1 200 Ok\r\n
\r\n
I'm almost a HTTP server!\r\n
HTTP/1.1 501 Not Implemented\r\n
```

Figura 13: Resultado da execução dos testes no servidor.

Outra forma de testar o GET no servidor é abrindo um navegador e acessando o link:

```
http://localhost:5555/qualquerPasta/qualquerArquivo.html
```

Alguns navegadores podem não exibir o resultado, pois a mensagem da resposta não está com os metadados sobre a entidade retornada. Faltam o tamanho e o tipo do dados retornados.

O próximo passo agora é completar o tratamento do método GET, implementando um método na classe que será responsável por tratar o caminho da requisição e verificar se o recurso existe e está acessível.

Desafio: Adicione no código do servidor instruções que exibam no console os resultados parciais dos processamentos internos da classe, como por exemplo, exiba a primeira linha da requisição, todos os campos do cabeçalho. Assim, será possível observar os dados de uma requisição originada por um navegador.

14.3.3 Entendendo o Que São *MIME Types*

Quando se trata de troca de mensagens em uma rede, existe um desafio em transferir conteúdo binário de forma que ele não seja destruído no transporte. Para facilitar o transporte e o armazenamento de conteúdo binário em sistemas legados ou restrito a caracteres US-ASCII é possível usar uma codificação/decodificação de conteúdo. Este processo transforma o conteúdo binário em uma sequência de caracteres compatível

com o meio de transporte, e posteriormente recupera a sequencia de bits original a partir da decodificação dos caracateres. A RFC4648 [Josefsson, 2006] detalha os padrões de codificação Base16, Base32 e Base64.

O HTTP permite que os dados binários sejam transportados diretamente no corpo da mensagen. No entanto, é requerido que no cabeçalho exista a indicação de *Content-Type* e *Content-Length*. Estes campos orientarão o cliente ou o servidor sobre como os dados do corpo devem ser processados e até onde deve ser feita a leitura por novos bytes. Uma especialização do padrão MIME é utilizada para especificar o tipo do conteúdo. Na seção 19.4 da RFC2616 [Fielding et al. 1999] são detalhadas as diferenças entre o padrão MIME para mensagens de e-Mail e de HTTP.

Basicamente o tipo de conteúdo é definido por uma sequência de caracteres que especifica o tipo e subtipo dos dados, como por exemplo “text/html” e “image/png”. Uma lista quase completa dos tipos é encontrada na página https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Basico_sobre_HTTP/MIME_types/Complete_list_of_MIME_types.

Um bom exercício para entender o padrão MIME e como ocorre a troca de mensagens em texto plano com conteúdo binário é abrir sua conta no GMail, localizar uma mensagem que contenha vários anexos e solicitar no menu de opções da mensagem a opção ‘Mostrar original’. Será aberta uma nova página contendo os dados crus da mensagem. Existe uma opção para baixar a mensagem completa em arquivo texto. Ao abrir a mensagem original apresenta um extenso cabeçalho e várias partes delimitadas por uma sequência de dígitos. A Figura 14 mostra o início de uma das partes do e-mail.

```
--000000000000761186056e41c465
Content-Type: application/pdf; name="tutorial-web-games-v05.pdf"
Content-Disposition: attachment; filename="tutorial-web-games-v05.pdf"
Content-Transfer-Encoding: base64
X-Attachment-Id: f_ji8b3jmq0

JVBERi0xLjUNCiW1tbW1DQoxIDAgb2JqDQo8PC9UeXBlL0NhdGFsb2cvUGFnZXMgMiAwIFIvTGFu
…
```

Figura 14: Trecho de uma mensagem com múltiplas partes. Em destaque um PDF.

Copie o conteúdo que está abaixo da linha em branco até encontrar o sinal ‘=’, que indica final do conteúdo codificado. Salve em um arquivo de texto. Abra o site https://www.base64decode.org/ e na parte de decodificação carregue o arquivo codificado. Se tudo der certo, o arquivo original será baixado para o seu computador.Desafio: Usando as rotinas de arquivos, verifique se o caminho do recurso especificado na requisição GET refere-se a um arquivo no seu sistema de arquivo. Usuários Linux poderão assumir o caminho “/” como sendo a raiz do sistema de arquivos, e os usuários Windows, como sendo a raiz da unidade C:. Caso não queira expor todo o seu sistema de arquivos, defina uma variável BASE_DIR que sempre será concatenada ao caminho para formar um caminho absoluto.

Desafio: Implemente o tratamento do método POST, lembrando que este método possui dados no corpo da mensagem de requisição. Observe os aspectos apresentados na seção 9.5 da RFC2616 [Fielding et al. 1999].

14.3.4 Um Protocolo Sem Estado

Uma das principais vantagens do HTTP é que cada requisição ao servidor é completamente independente das demais. Não existe um procedimento complexo para a comunicação e nenhuma informação do cliente é armazenada pelo servidor para controlar as múltiplas requisições. Simplesmente o cliente requisita um recurso e o servidor responde. Não existe uma máquina de estado que define a sequência de mensagens que devem ser trocadas entre o cliente e o servidor, e em qual estado deste autômato estaria cada cliente. Este tipo de abordagem tornaria o servidor extremamente mais complexo e demandaria mais recursos de hadware.

Apesar de não definir um autômato, o HTTP oferece o recurso de Cookies que permite armazenar alguns dados nos clientes que podem servir para que o servidor entenda em qual estado o cliente se encontra. Este recurso não é implementado diretamente pelo protocolo, mas é um mecanismo padronizado que utiliza os recursos do HTTP para permitir a manipulação de sessão. A RFC2109 [Kristol and Montulli 1997] descreve como funciona este mecanismo. Basicamente o servidor registra no cabeçalho da resposta um campo "Set-Cookie:" com valores que serão armazenados pelos clientes. Todas as vezes que os clientes voltarem a requisitar algum recurso para o mesmo servidor, o cliente registra no cabeçalho da requisição um campo "Cookie:" contendo os dados anteriormente fornecidos pelo servidor. Estes dados não são processados pelo cliente e possuem algumas diretiva para definir validade e escopo dos cookies.

Desafio: Altere o código da classe *ServerWorker* para que antes de escrever a resposta para o cliente, verifica se na requisição houve o registro de um *cookie* chamado *accessCount* no cabeçalho, conforme a especificação da RFC2109. Se houver, incremente o valor e registre ele como um cookie no cabeçalho da resposta. Se não houver, registre o cookie *accessCount*=14.

14.3.5 Segurança no HTTP

Como foi visto até aqui, as mensagens do protocolo HTTP são composta por texto plano e todos os dados necessários para decodificar os dados binários estão no cabeçalho da própria mensagem. Qualquer interceptador poderá abrir as mensagens e recuperar seu conteúdo. Quando uma mensagem tiver informações confidenciais, ela deverá ser enviada para o servidor ou cliente usando o *HTTP Secure* (HTTPS). Este mecanismo simplesmente cria um canal criptografado entres os nós usando o mecanismo de chaves pública e privada por onde as mensagens HTTP podem transitar com segurança. Maiores detalhes sobre as últimas versões de Secure Sockets Layer (SSL) e Transport Layer Security (TLS) podem ser obtidas na RFC6101 [Freier et al. 2011] e na RFC8446 [Rescorla, 2018], respectivamente.

Além do mecanismo de criptografia de mensagens, o HTTP oferece dois esquemas de autenticação de usuários, o *Basic* e o *Digest*, que permitem restringir o acesso a determinados recursos através da exigência de credenciais. Estes mecanismos utilizam campos específicos no cabeçalho para informar o cliente sobre a necessidade de autenticação e para permitir que o cliente solicite ao usuário as credenciais e as envie na próxima requisição. Todas as especificações destes mecanismos são discutidos na RFC2617 [Franks et al. 1999].

Desafio: Usando o esquema *Basic*, permita que seu servidor HTTP exija autenticação para acessar arquivos em determinadas pastas.

14.4 Considerações Finais

Neste capítulo foram apresentadas as RFCs, que formam um rico repositório de informações técnicas sobre praticamente todos os padrões que regem a infraestrutura da Internet. As principais RFCs abordadas foram a RFC3986 “Uniform Resource Identifier (URI): Generic Syntax” [Berners-Lee et al. 2005], a RFC2616 “Hypertext Transfer Protocol -- HTTP/1.1” [Fielding et al. 1999] e a RFC2045 “Multipurpose Internet Mail Extensions (MIME)” [Freed et al. 1996] que fornecem as informações essenciais para a implementação de um servidor HTTP básico. Em um ambiente profissional, deve ser realizada a leitura completa de todas as RFCs indicadas e periféricas. Assim, garante-se que o produto desenvolvido segue as especificações técnicas.

Os códigos apresentados mostraram, de uma forma sequencial e incremental, como construir algo funcional seguindo as especificações técnicas de um servidor HTTP. A linguagem utilizada foi a linguagem Java, mas qualquer outra linguagem de programação com suporte a soquetes pode ser utilizada na implementação dos exemplos apresentados.

O conhecimento aqui apresentado deverá ajudar o leitor a pensar no desenvolvimento de novos serviços e aplicações que podem ser facilmente implementados a partir dos exemplos. Apesar de não ter abordado a parte dinâmica neste mini curso, ela pode ser facilmente integrada no processamento das requisições e na correta interpretação de um URI. Desta forma, é possível criar meios de configuração e acesso em diversas aplicações disponibilizando uma interface HTTP compatível.

Por fim, fica o apelo para que o leitor sempre busque o conhecimento em RFCs e documentos oficiais sobre determinada tecnologia, evitando ter toda sua formação baseada em materiais de fóruns e de blogs.

Referências

ARPANET. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://en.wikipedia.org/w/index.php?title=ARPANET&oldid=842764807>. Acesso em: 7 fev. 2018.

Berners-Lee, T., "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", RFC 1630, DOI 10.17487/RFC1630, June 1994, <https://www.rfc-editor.org/info/rfc1630>.

Berners-Lee, T., Fielding, R., And L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., And T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/info/rfc2616>.

Firewall. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Firewall&oldid=52268207>. Acesso em: 4 jun. 2018.

Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., And L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, DOI 10.17487/RFC2617, June 1999, <https://www.rfc-editor.org/info/rfc2617>.

Freed, N. And N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

Freier, A., Karlton, P., And P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, DOI 10.17487/RFC6101, August 2011, <https://www.rfc-editor.org/info/rfc6101>.

He, Yihua, Georgos Siganos, and Michalis Faloutsos. "Internet topology." In Encyclopedia of Complexity and Systems Science, pp. 4930-4947. Springer, New York, NY, 2009.

Hypertext Transfer Protocol. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Hypertext_Transfer_Protocol&oldid=844811133> . Acesso em: 29 mai. 2018.

Internet Assigned Numbers Authority. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2017. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Internet_Assigned_Numbers_Authority&oldid=49715968>. Acesso em: 29 ago. 2017.

Internet Engineering Task Force. IN:WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://en.wikipedia.org/w/index.php?title=Internet_Engineering_Task_Force&oldid=843477380>. Acesso em: 7 jun. 2018.

Internet. IN:WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Internet&oldid=51418559>. Acesso em: 3 mar. 2018.

Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

Kristol, D. And L. Montulli, "HTTP State Management Mechanism", RFC 2109, DOI 10.17487/RFC2109, February 1997, <https://www.rfc-editor.org/info/rfc2109>.

Request For Comments. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2016. Disponível em: <https://en.wikipedia.org/w/index.php?title=Request_for_Comments&oldid=844599815>. Acesso em: 19 set. 2016..

Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Sistema de Nomes de Domínio. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Sistema_de_Nomes_de_Dom%C3%ADnio&oldid=52241704>. Acesso em: 1 jun. 2018.

TANENBAUM, A. S. – Computer Networks – Prentice Hall, Inc, 2003

TCP/IP. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=TCP/IP&oldid=52104983>. Acesso em: 16 mai. 2018.

15. Capítulo 15

Autores:

Bruno da Silva Rodrigues
Universidade Presbiteriana Mackenzie
email: bruno.rodrigues@mackenzie.br
Bruno Luís Soares de Lima
Universidade Presbiteriana Mackenzie
email: bruno.lima@mackenzie.br
Victor Inácio de Oliveira
Centro Universitário SENAC
email: victorinac@gmail.com

Capítulo 15

Desenvolvimento de Modelo Robótico Controlado por Arduino

Bruno da Silva Rodrigues, Bruno Luís Soares de Lima e Victor Inácio de Oliveira

Abstract

Practical application in various segments of society such as smartliving (eg vacuum cleaners and lawnmowers), military (eg VANT - Unmanned aerial vehicle) and industrial (eg automated cargo transport) and nowadays the humans assist robot it is increasingly useful and common. This diversity of applications has aroused the interest not only of professionals who work in the industry who have economic motivations, but also researchers and teachers interested in multidisciplinarity and the use of robotics as an active learning methodology. The text will deal with basic principles of robotics and embedded systems programming with the goal of building robotic prototypes.

Resumo

Com aplicação em diversos segmentos da sociedade como smartliving (e.g. aspiradores de pó e cortadores de grama), militares (e.g. VANT - Veículo aéreo não tripulado) e industriais (e.g. transporte de cargas automatizado), é cada vez mais comum o uso de robôs para auxiliar os seres humanos. Essa diversidade de aplicações vem despertando o interesse não só de profissionais que atuam na indústria que tem motivações econômicas, mas de pesquisadores e professores interessados na multidisciplinaridade e no uso da robótica como metodologia ativa de aprendizagem. Este texto, tem o objetivo de abordar os princípios básicos da robótica e programação de sistemas embarcados com objetivos de construir protótipos robóticos.

1.1 Introdução a Robótica

Robótica é um ramo multidisciplinar da tecnologia onde engenheiros dos mais diversos segmentos (eletrônicos, mecânicos, bioengenheiros, etc) e profissionais de computação lidam com o design, a construção, a operação e o controle de robôs que além automatizar

tarefas, podem coletar dados sobre o desenvolvimento de suas funções assim como do ambiente onde ele está inserido por meio de sensores.

Definidos como mecanismos eletromecânicos usados para executar tarefas complexas, perigosas ou repetitivas com um alto grau de precisão, diferente de outras máquinas, os robôs são orientados a tarefas e se adaptam ao ambiente tomando decisões que possibilitam corrigir eventuais falhas na execução de sua tarefa.

Desde da criação do primeiro robô comercial instalado na General Motors por George Devol em 1961 [Nof, 1990], a robótica se difundiu para diversos seguimentos dentre os quais pode-se citar educação onde robôs estão sendo usados como metodologia ativa de aprendizagem [Shim, 2016] e na considerada quarta revolução industrial conhecida como indústria 4.0 [Pieroni & Brilli, 2018] com a utilização dos chamados robôs autônomos.

1.1.1 Tipos de Robôs

Não há um consenso na literatura especializada sobre como classificar robôs. Para delimitar e classificar os diferentes tipos de robôs, apresentaremos as classificações segundo os principais institutos especializados. A Associação Japonesa de Robôs Industriais (JIRA), o Instituto de Robôs Americana (RIA) e a Associação Francesa de Robôs Industriais (AFRI) classificam os robôs em função do controle realizado, dividindo os robôs em:

- Robô de controle manual – robôs com diversos graus de liberdade cujas ações são comandas por um operador.

- Robô de sequência fixa - são programados para executar sequencias repetitivas e não necessitam de um operador para desenvolver suas funções.

- Robô de sequência variável - similares aos robôs de sequência fixa, mas podem ter as sequencias configuradas mais facilmente.

- Robô inteligente - possui meios para entender o ambiente onde está inserido e capacidade de completar com sucesso uma tarefa apesar de eventuais mudanças nas condições do ambiente.

Apesar do método de classificação utilizado ser o mesmo, RIA, JIRA e AFRI divergem sobre os nomes e a quantidade de categorias [Coiffet, 2012] onde por exemplo segundo a RIA, robôs de sequência variável e robôs de controle manual são incluídos em uma única classe e são classificados como robôs de sequência variável.

Tsai [Tsai, 1999] classifica os robôs em 5 grupos usando como base a cinemática dos movimentos determinada pela área de trabalho do robô onde temos os seguintes grupos:

- Robô cartesiano – nessa categoria, os robôs possuem 3 eixos lineares prismáticos coincidente com um sistema de coordenadas cartesianas.

- Robô cilíndrico – similar ao robô cartesiano, o robô cilíndrico possui duas juntas prismáticas sobre uma base rotacional que amplia a área de atuação do robô.

- Robô esférico – com dois movimentos rotacionais (base e ombro) e um linear, essa categoria de robô permite desenvolvimento de trabalhos em área esférica.

- SCARA (*Selective Compliance Assembly Robot Arm*) – muito utilizados em linhas de montagem, os robôs do tipo SCARA são compostos por duas juntas de rotação e uma prismática conforme mostrado na Figura 1.

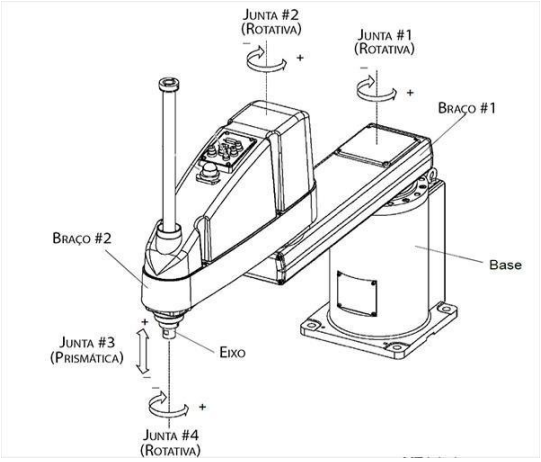


Figura 1. Robô SCARA modelo SCARA G10-851S da Epson (Epson, 2018)

Com a evolução das linhas de produção e a necessidade de flexibilizar os sistemas de manufaturas [Vosniakos, 1990] , além dos robôs de base fixa, uma nova categoria de robôs surgiu com capacidades de locomoção e de operação de modo semi ou completamente autônomo conhecida como Robôs Móveis e/ou Robôs Móveis Autônomos (RMAs) que serão discutidos nesse texto.

1.1.2 Controles e Inteligência.

O sistema de controle de um robô é composto por hardware e por software. A integração dos dois permite que o mecanismo robótico realize as tarefas e tome decisões [Sciavicco 2000]. O sistema de controle normalmente processa informações provenientes de sensores que detectam condições do ambiente e acionam atuadores que são responsáveis por promover a ação programada no robô.

O algoritmo de controle normalmente pode ser executado por um computador, ou por um dispositivo lógico programável ou por um microcontrolador, dependendo da necessidade da aplicação. Dependendo desempenho computacional desejado, custo ou robustez pode ser escolhido o dispositivo eletrônico responsável pelo processamento de dados e execução do algoritmo de controle.

O hardware do sistema de controle é composto, além do dispositivo de processamento, por atuadores como motores, dispositivos de áudio, resistências e dispositivos LASER. E também, o sistema de controle conta com sensores como por exemplo, para detecção de grandezas como presença, luz, ultrassom, GPS (*Global Position System*) e acelerômetros.

O sistema de controle do robô pode ser implementado através de uma malha fechada ou de uma malha aberta [Sciavicco, 2000]. O sistema de controle em malha fechada consiste na verificação por meio de sensores do estado atual do dispositivo (retroalimentação) a ser controlado e a comparação deste com o estado ou valor de medida pré-definida, ou valor de *set-point*. Esta comparação poderá resultar numa diferença (erro). Neste caso, o sistema deverá, através do processo de retroalimentação atuar de

forma a reduzir esta diferença a zero [Ogata, 2011]. Na Figura 2 um diagrama representando um sistema de controle em malha fechada.

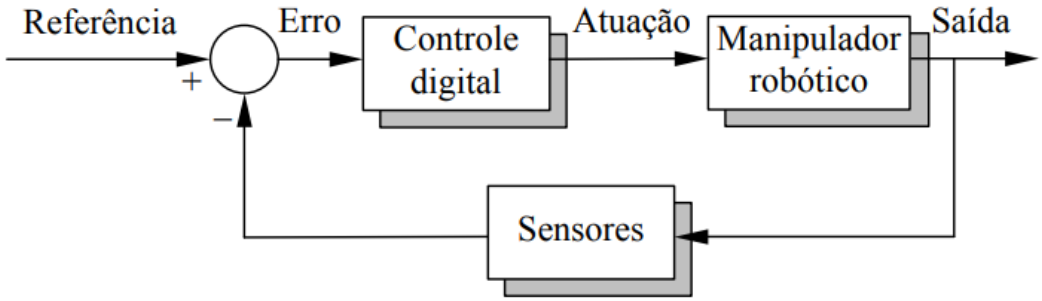


Figura 2. Representação de sistema de controle de malha fechada.

No exemplo da Figura 2 um manipulador robótico poderia ser controlado para realizar um determinado movimento, e sua posição, velocidade e aceleração poderiam ser medidos através dos sensores. A partir do algoritmo do sistema de controle seria realizada a verificação da comparação entre os valores medidos (retroalimentação) e os valores estabelecidos como referência (*set-point*), desta forma, o algoritmo de controle acionaria dispositivos atuadores de forma a manter o manipulador robótico com movimentos constantes e padronizados.

O sistema de controle em malha aberta a saída não tem impacto sobre a ação de controle, ou seja, a decisão é tomada a partir da entrada e não considera a saída. A Figura 3 exemplifica um sistema de controle em malha aberta.

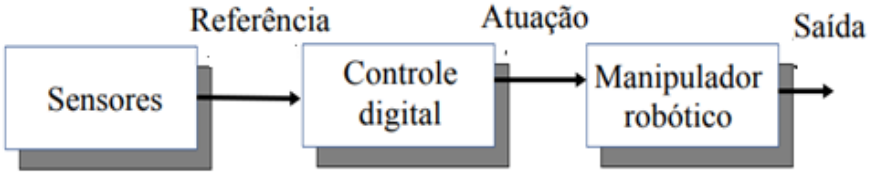


Figura 3. Representação de sistema de controle de malha aberta.

Conforme a Figura 3, notamos que não há necessidade de medir a saída, nem de realizar a retroalimentação. Um exemplo de sistema com controle em malha aberta é o acionamento de um robô via controle remoto, os movimentos do robô são definidos pelos dados enviados pelo controle ao sistema de processamento que realiza o acionamento dos motores de movimentação. Outro exemplo de fácil identificação é o sistema de controle de semáforos de trânsito que opera em malha aberta.

1.1.3 Aplicações

Desde da criação do primeiro robô comercial a robótica evoluiu muito e atualmente dentre as principais aplicações da robótica podemos citar:

- Robôs domésticos geralmente usado para auxiliar na limpeza.
- Robôs de entretenimento projetados para lazer de crianças [Druin, 2000].
- Robôs médicos [Hamet, 2017] que auxiliam profissionais a realizar cirurgias.
- Robôs de reconhecimento e vigilância para uso militar [Amareswar, 2017].

- Robôs industriais e robôs autônomos que são um dos principais elementos da quarta revolução industrial conhecida como indústria 4.0 [Pieroni & Brilli, 2018].

- Robôs na educação sendo usado como metodologia ativa de aprendizagem [Shim, 2016].

1.2 Introdução a Sistemas Embarcados

1.2.1 O que é Arduino?

O Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única projetada com um microcontrolador Atmel AVR [Mcroberts, 2011] com suporte de entrada/saída embutido onde dispositivos de saída (motores, luzes, etc) podem ser controlados através da ação de um operador ou de maneira automática.

Criado pelos professores Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis no *Interaction Design Institute de Ivrea* (Itália) [Severance, 2014] que esperavam tornar mais fácil o desenvolvimento de soluções de automação, o Arduino permite criar projetos de baixo custo, flexíveis de fácil implementação mesmo por profissionais com conhecimentos básicos em eletrônica [Banzi, 2015]. Essas características contribuíram com a popularização do Arduino e atualmente temos uma variedade de plataformas (Figura 4) adaptadas as necessidades especificas que vão desde plataformas de aprendizagem a dispositivos vestíveis e impressões 3D [Arduino, 2016].

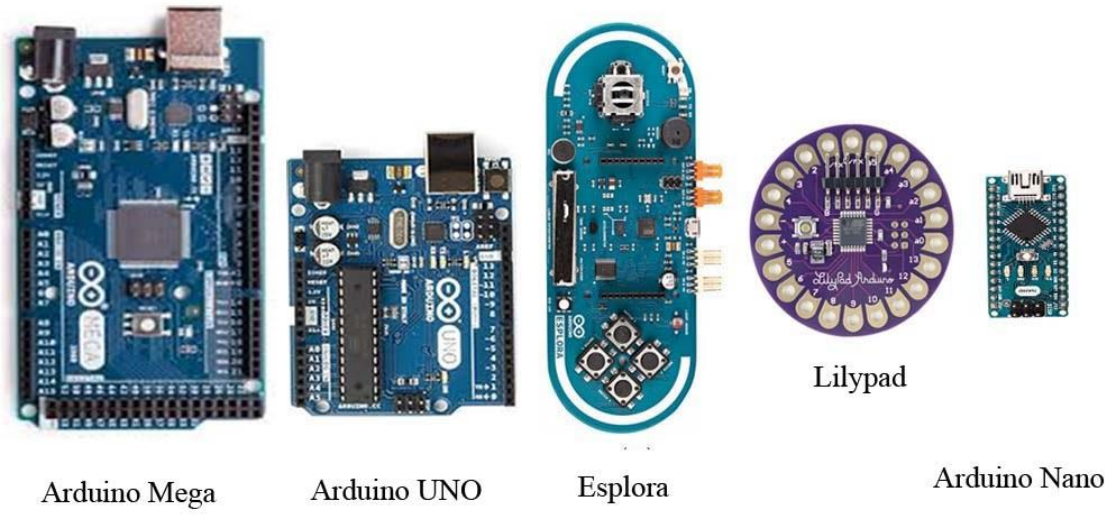


Figura 4. Modelos de Arduino.

Em termos práticos, o Arduino é um pequeno computador que também pode ser chamado de sistema embarcado, pois nesse tipo de dispositivo todo o hardware (unidade central de processamento, memoria, controladores, etc) está incorporado em um único circuito integrado (CI) [Buyya, 2016]. Com capacidade computacional inferior a um computador tradicional, normalmente os sistemas embarcados são programados para realizar um conjunto de tarefas específicas que serão discutidas no item 2.2.

1.2.2 Apresentação de Sistemas: Cenários e Aplicações.

A plataforma Arduino tem potencial aplicação em qualquer sistema de automação e monitoramento onde o desempenho computacional não é fator limitante e onde o objetivo é de um projeto com relativo baixo custo.

Devido a facilidade, a agilidade de programação e gravação do microcontrolador Atmega 328, que é o microcontrolador utilizado no Arduino, a plataforma tem sido muito utilizada para prototipação de projetos. Outro ponto que facilita a utilização do Arduino em larga escala são os Shields, placas eletrônicas contendo sensores, atuadores ou dispositivos de comunicação, que podem ser conectadas diretamente ao Arduino.

As aplicações com Arduino podem variar desde um simples monitoramento da temperatura e umidade de um ambiente [Ferreira, 2017] até aplicações envolvendo robótica ou equipamentos médicos [Mathew, 2016]. Com cada vez mais aplicações voltadas a Internet das Coisas um maior número de desenvolvimentos é voltado para integração com smartphones e ambiente de nuvem. Estes desenvolvimentos focam na automação de atividades comerciais e industriais [Basnayake, 201] ou são focadas em soluções urbanas ou residenciais [Misbahuddin, 2015].

Como por exemplo, poderia ser realizada a contagem de veículos nas vias para determinar o tempo do semáforo ou mesmo o acionamento do ar-condicionado via aplicação celular. No contexto de nuvem, pode se monitorar uma planta fabril remotamente a partir de uma página Web, com sensores instalados na fábrica tendo seus dados coletados por Arduino.

No campo cientifico a plataforma Arduino também se destaca sendo utilizado em diversas publicações, como por exemplo, o trabalho de Rodrigues (2009) onde toda a coleta de dados e controle aplicado a uma matriz de sensores de PH (Potencial de Hidrogenização) é realizado a plataforma Arduino.

1.2.3 Componentes e Arquitetura do Arduino.

A popularização do Arduino se deu ao fato da plataforma Arduino ser composta por hardware e software muito flexíveis, fáceis de usar onde as interfaces (conectores) de entrada e saída (I/O) são expostas, permitindo integrar a CPU à sensores, atuadores e/ou outros módulos expansivos, conhecidos como shields.

Considerada o melhor modelo para quem quer começar no mundo dos sistemas embarcados, o Arduino UNO é o modelo mais utilizado da família Arduino devido a sua simplicidade e robustez. Na figura 5 temos uma foto do Arduino UNO e seus principais componentes.

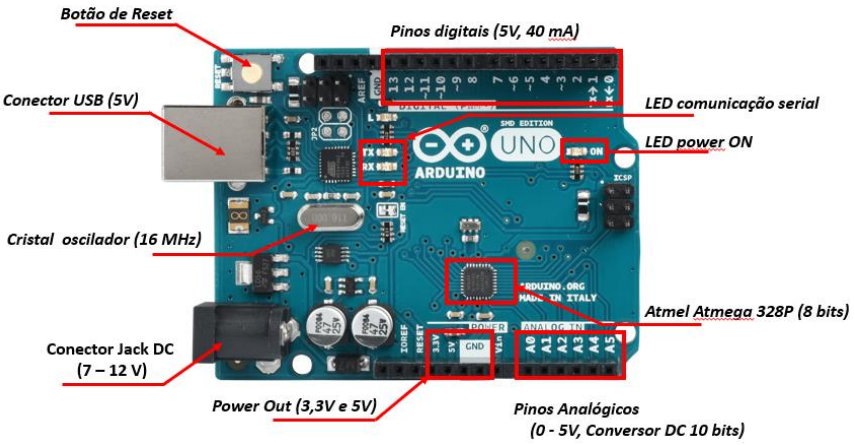


Figura 5. Foto do Arduino modelo Uno.

Baseado no microcontrolador ATmega328 de 8 bits, na tabela 1, temos as principais características do ATmega328 e consequentemente do Arduino Uno.

Tabela 1. Principais características do Arduino Uno

Componente	Característica
Microcontrolador	ATMega328
Tensão de alimentação	6V – 12 V
Entradas e saídas (I/O) digitais	14 pinos (dos quais 6 podem ser saídas PWM)
Entradas Analógica	6 pinos
Memória Flash	32 KB (2 KB usados para o Bootloader)
SRAM	2 KB
EEPROM	1KB
Frequência de clock	16 MHz

Alimentação

O Arduino UNO pode ser alimentado pela conexão USB ou por qualquer fonte de alimentação externa (pilhas e baterias) via conector Jack DC onde a fonte de alimentação é selecionada automaticamente pelo circuito do Arduino. Apesar de suportar alimentação externa entre 6 e 20 V, a alimentação recomendada é de 7 a 12 V (Arduino, 2016).

Memória

Baseado na arquitetura Harvard o ATmega328 possui 32 KB de memória flash (memória de instruções) e 2 KB de SRAM usada como memória de dados. Além da memória de dados e instruções independentes, o Arduino UNO possui 1 KB de EEPROM (Electrically-Erasable Programmable Read-Only Memory – memória de leitura

programável e regravável eletronicamente) que pode ser utilizada pelas aplicações desenvolvidas pelo usuário.

Entradas e saídas

Conforme apresentado na figura 5 o Arduino UNO possui 14 pinos digitais que podem ser utilizados como uma entrada ou uma saída em conformidade com a necessidade e programação do usuário. Operando com 5V, cada pino pode fornecer ou receber um máximo de 40mA. Além disso, alguns pinos possuem funções especificas:

Comunicação serial: Os pinos 0 e 1 são usados para comunicação serial do Arduino onde o pino 0 é usado para receber dados (RX) e o pino 1 transmitir dados (TX).

PWM (Pulse Width Modulation): Os pinos 3, 5, 6, 9, 10, e 11 fornecem uma saída PWM. O PWM é uma técnica utilizada por microcontroladores que permite simular um sinal analógico através da modulação da largura de pulso de um sinal digital [Lathi, 2012].

Interruptores Externos: Os pinos 2 e 3 podem ser configurados para disparar interrupções em função da variação de sinal em sua entrada.

Além das entradas e saídas digitais, o Arduino UNO possui 6 entradas analógicas, onde cada uma delas está ligada a um conversor analógico-digital de 10 bits, que transformam a leitura analógica em um valor entre 0 e 1023. A conversão analógica-digital será novamente abordada no item 1.3.2

Programação

A plataforma Arduino possui um Ambiente de Desenvolvimento Integrado (IDE - *Integrated Development Environment*) que permite ao usuário desenvolver, salvar, compilar os códigos no computador e realizar a gravação do código compilado no Arduino através da porta USB (*Universal Serial Bus*) do computador. A IDE do Arduino é ambiente de desenvolvimento é baseada na linguagem de programação C/C++.

1.3 Introdução a Eletrônica.

1.3.1 Definições (corrente, tensão, resistência)

A corrente elétrica (i) é dada pelo fluxo ordenador de cargas num material num determinado intervalo de tempo conforme mostra a equação 1.

$$i = \frac{dq}{dt} \tag{1}$$

Assim, a corrente elétrica é dada pela derivada da carga elétrica no tempo. Para que ocorra o deslocamento de carga é necessário a aplicação de uma força elétrica esta é proveniente de uma diferença de potencial (d.d.p.), denominada tensão elétrica (V).

A impedância, dificuldade a passagem de corrente elétrica num condutor, circuito ou equipamento é definido como resistência elétrica sendo mensurado em ohms (Ω).

1.3.2 Sinais Elétricos.

Os sinais elétricos podem ser definidos pela diferença de potencial elétrico, ou seja, tensão elétrica imposta entre dois pontos ou pela variação de corrente elétrica num intervalo de tempo.

No estudo da eletricidade e eletrônica as grandezas de tensão elétrica e corrente elétrica são as mais importantes.

A corrente elétrica é caracterizada pela variação de carga elétrica num intervalo de tempo.

A tensão elétrica é a diferença de energia potencial elétrica por unidade de carga entre dois pontos, sendo esta diferença de energia responsável por movimentar as cargas e promover o surgimento da corrente elétrica.

Os sinais elétricos podem ser transmitidos e manipulados pela eletrônica na forma digital e analógica, os itens a seguir discutem as duas formas.

Sinais Digitais

Um sinal digital refere-se a um sinal elétrico com valores discretos descontínuos no tempo. Esses sinais são o princípio de operação dos circuitos digitais como computadores e sistemas embarcados (Arduino) e trabalham com dois níveis de tensão: uma perto de um valor de referência (denominado como terra ou zero volts) e a outra um valor próximo da tensão de alimentação (5 V no caso da figura 6) que correspondem ao nível logico zero e nível logico 1 ou "falso" e "verdadeiro" descritos na lógica booleana [Tocci & Widmer, 2011].

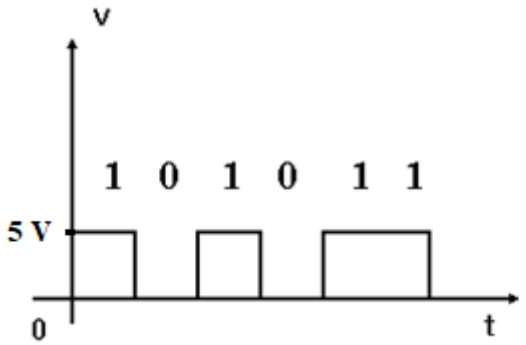


Figura 6. Representação do sinal digital variando em função do tempo.

Embora a maioria das grandezas manipuladas e analisadas na natureza sejam analógicas, as grandezas representadas na forma digital são menos susceptíveis a ruídos, variações indesejadas e mais fáceis de serem integrados a outros sistemas digitais como sistema embarcados.

Sinais Analógicos

Os sinais analógicos são abundantes na natureza [Lathi, 2012] e normalmente podem ser utilizados para representar a variação de uma grandeza física, como por exemplo: a variação da temperatura em um dia, a variação da velocidade em um automóvel, a variação da pressão arterial de uma pessoa, a variação da distância entre um robô e um objeto, entre outros. Esses sinais são de suma importância para o desenvolvimento de robôs autônomos que possuem inúmeros sensores capazes de detectar sinais analógicos diversos a sua volta.

Fisicamente um sinal analógico pode ser representado por uma curva de variação contínua, como a vista na Figura 7, onde é mostrada uma variação da temperatura em °C em função do tempo, medido em horas, nota-se que a variação é continua. Ou seja, para qualquer intervalo fechado de valores existirão infinitos valores possíveis dentro deste intervalo [Fleming, 2006].

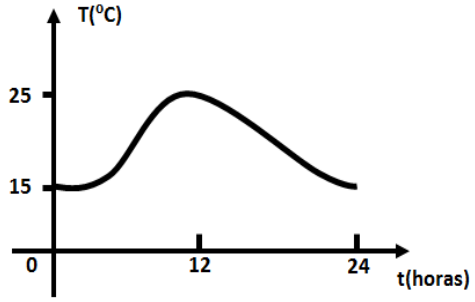


Figura 7. Mostra a variação contínua de uma grandeza analógica, no caso temperatura em função do tempo, dado em horas

Como processar infinitos valores? A resposta para esta questão é impossível. Para que seja possível ler e processar um sinal analógico qualquer é preciso quebra-lo em uma quantidade finita de valores, processo esse conhecido como discretização ou ainda conversão Analógico-Digital (A/D) (Figura 8).

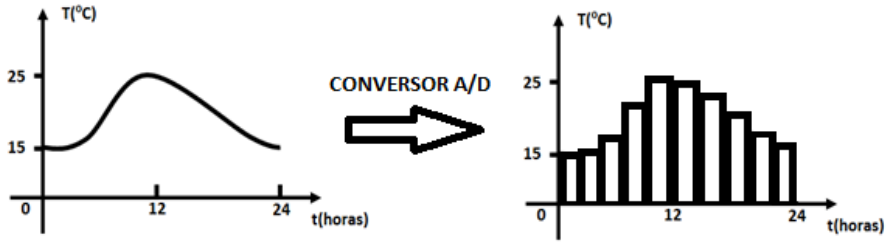


Figura 8. Processo de conversão A/D.

Conforme apresentado na Figura 5, o Arduino UNO possui 6 pinos que podem ser usados como entradas análgicos, pois possuem esse conversor A/D.

A resolução do conversor A/D, que para o Arduino UNO vale 10 bits. Ou seja, a variação analógica em quantidade de bits será de 0 a 1023 bits.

1.4 Dispositivos de Entrada e Saída:

1.4.1 Saída de Dados.

Dentre as portas do Arduino existem as denominadas portas digitais e podem ser configuradas, via software, como entrada ou saída digital [Massimo, 2015]. Uma saída digital tem a função de controlador algum tipo de atuador, que pode ser um *led* de sinalização, um motor elétrico, um relé de contato ou até mesmo um *buzzer* (componente de geração de ruídos sonoros). Como exemplo, na Figura 9 pode-se ver o circuito de acionamento de um motor.

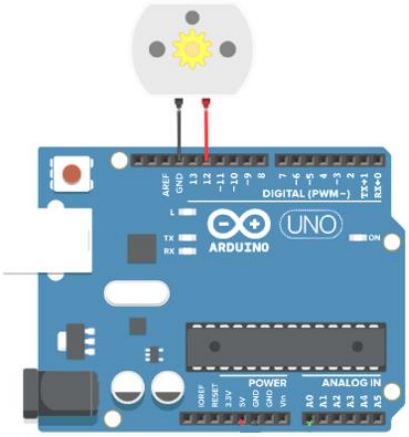


Figura 9. Mostra um motor conectado em uma das entradas digitais do Arduino UNO.

No circuito da Figura 9 temos um exemplo de um motor conectado ao Arduino através dos pinos GND (terra) e ao pino digital 12 que controlará o acionamento do dispositivo. Como citado no 1.2.3, as saídas no Arduino UNO podem ser configuradas como PWM que servem para controlar a intensidade de tensão elétrica fornecida a um atuador. Como por exemplo, através de uma saída PWM será possível controlar a intensidade luminosa de um *led* ou, até mesmo, controlar a velocidade de rotação de um motor DC.

1.4.1.1 Leds, Como Controlar

Um *Led* (*Ligth emited diode*) é um dispositivo amplamente utilizado na sinalização em sistemas embarcados. Basicamente ele possui dois terminais, um deles positivo (anodo) e o outro negativo (catodo) (Figura 10).

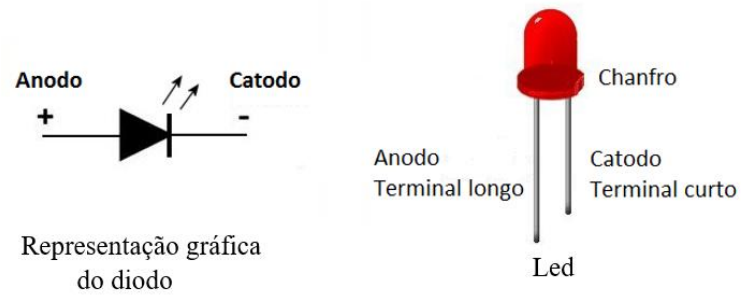


Figura 10. Led (Ligth emited diode)

Para que o *Led* acenda ele deverá estar em um estado denominado diretamente polarizado (Verdonck, 2003), onde a tensão elétrica de acionamento positiva deverá estar ligada ao anodo e a tensão elétrica negativa no catodo, como pode ser visto na Figura 11.

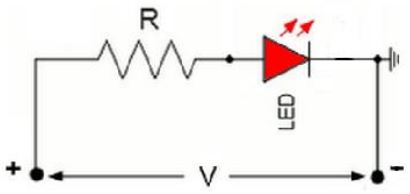


Figura 11. Circuito de polarização direta de um Led. O resistor R serve para limitar a corrente elétrica do circuito.

Para conectar um *Led* ao Arduino pode-se conectar ao terminal GND o catodo e ao pino de saída do Arduino o anodo, porém um resistor de limitação de corrente deve ser ligado em série com o *led* para que o mesmo não venha a queimar. Normalmente usa-se resistores entre 220W e 680W.

Um *led* é um dispositivo que deve ser ligado a uma saída digital, logo o pino ao qual ele deverá ser ligado deve ser configurado como saída digital. Como exemplo de uso, pode-se usar o circuito visto na Figura 12.

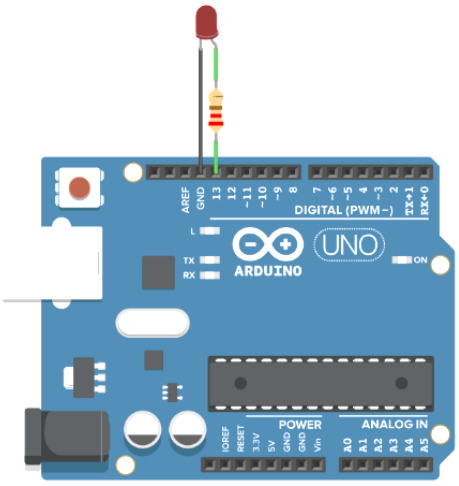


Figura 12. Led ligado ao pino 13 de um Arduino UNO. Neste caso um resistor de 220W foi ligado em série com ele.

Após a implementação do circuito da Figura 12, será necessário configurar o software do Arduino para controlar o *Led*. Para controlar qualquer saída digital no Arduino, primeiro é necessário configurar um pino especifico para se comportar como saída (assim como também pode ser configurado como entrada) através do comando pinMode(), após configurar o pino como saída o programador deverá definir o nível logico da saída, a sintaxe para os comandos citados é dada por:

pinMode(*pin*, *mode*);

digitalWrite(*pin*, *value*);

onde

pin é o número do pino que será controlado

mode define como pino será utilizado onde pode ser usado *INPUT*/*OUTPUT (*entrada e saídas.

Value define o valor logico (*HIGH* e *LOW*) que será aplicado na saída.

Logo para acender o *led* conectado ao Arduino conforme circuito apresentado na figura 13 teremos:

```
pinMode(13, OUTPUT);
digitalWrite(13, HIGH);
```

1.4.1.2 Motores DC, Aplicações e Controle

Um motor DC é uma máquina elétrica capaz de converter energia elétrica em energia mecânica (Martinewski, 2016). Como qualquer motor ele se divide em duas partes: o estator, que recebe a tensão elétrica e é a parte fixa e o rotor que efetivamente executa o movimento (Figura 13).

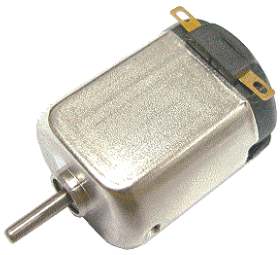


Figura 13. Foto de um motor elétrico DC típico

Dois tipos de controle podem ser utilizados em um motor DC, um deles é a reversão e o outro o controle de velocidade. Para fazer esses controles pode-se inverter o sentido da tensão elétrica aplicada ao motor, nesse caso o sentido de rotação é alterado. E para se controlar a velocidade de rotação é preciso controlar o nível de tensão aplicado ao motor.

Esses dois controles podem ser feitos pelo Arduino UNO com o auxílio de *shields*, como por exemplo, o *shield* L293. Esse *shield* é composto por dois circuitos integrados L293 que permitem a ligação de até 4 motores DC. Um exemplo de utilização pode ser visto no circuito da Figura 14, onde um motor é ligado ao *shield* L293.

Figura 14. Motor DC conectado ao *Shield* L293.

Para controlar um ou mais motores usando o shield L293 é necessário utilizar a biblioteca AFMotor e os comandos *motor1.setSpeed* que definir a velocidade de rotação do motor e *motor1.run* que indicará o sentido de rotação do motor a sintaxe dos comandos é dado por:

setSpeed(*speed*)

run(*mode*)

onde,

speed é a velocidade definida através de PWM

mode é sentido de rotação do motor dado por FORWARD e BACKWARD e RELEASE para parar o motor

1.4.2 Entrada de Dados.

1.4.2.1 Controle Através de Botões e Chaves

Pode-se entender como pinos de entradas digitais como sendo módulos que funcionam com sinais discretos. Como exemplo pode-se citar botões, sensores de presença, sensores de luminosidade, sensores capacitivos e indutivos e teclado matricial ente outros. A figura 15, mostra alguns componentes que podem ser utilizados como entrada digital.

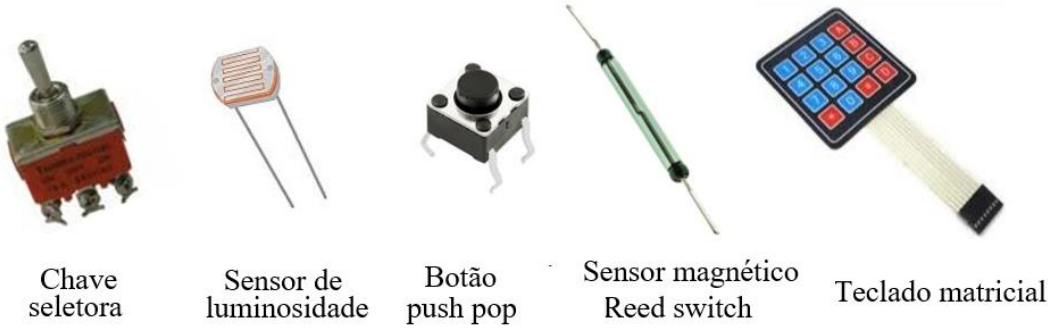


Figura 15. Exemplos de dispositivos usados como entradas digitais

Para controlar a entrada de dados através de um dispositivo como os apresentados na Figura 15, assim como na saída de dados, será necessário configurar o através do comando pinMODE() com a opção INPUT, já a leitura da tensão de entrada (nível logico) será feita através do comando digitalRead( ) onde a sintaxe é dada por:

digitalRead(*pin*);

onde

pin é o número do pino que irá receber um sinal gerado pelos dispositivos de entrada.

1.4.2.2 Comunicação Serial

Transmissão de dados é a transferência dos dados físicos ao longo de um canal de comunicação ponto-a-ponto ou multiponto. Em nosso contexto, realizando comunicação entre o Arduino a outros dispositivos. Todas as placas Arduino realizam transferência de dados pela serial (também conhecida como UART ou USART) através dos pinos digitais

0 (RX) e 1 (TX). Além do padrão UART (RS232), o Arduino também possibilita o uso das comunicações SPI e I^2C, que servem basicamente ao mesmo propósito: transferir dados em alta velocidade para qualquer dispositivo compatível.

Bluetooth

Idealizado pela Ericsson para conectar seus celulares a outros dispositivos sem o uso de cabos (Tanenbaum, 2011), o Bluetooth se tornou uma alternativa sem fio ao protocolo serial RS-232 permitindo conectar o Arduino a outros Arduinos assim como a *smartphones*, *tablets* e *laptops*.

Por ser um dispositivo de baixo custo projetado para operar na faixa de 2,45 GHz com baixo consumo de energia com baixo alcance. Diversos *Shields bluetooth* foram projetados para o Arduino dentre os quais podemos citar o HC-05 (Figura 16), HC-06 e o NRF2401.

Figura 16. Modulo Bluetooth HC-05

Para controlar o modulo *bluetooth*, o desenvolvedor deverá usar a biblioteca SoftwareSerial e os comandos bluetooth.begin() e bluetooth.read() cuja sintaxe é dada por:

bluetooth.begin(*Baud*);

bluetooth.read( );

onde

baud é taxa de bits transmitidos por segundo.

1.5 Construção de Modelo Robótico.

1.5.1 Materiais

Para a montagem do modelo robótico serão necessários os seguintes componentes:

- ✓ Placa Arduino UNO;
- ✓ Shield L293 para controle de Motor;
- ✓ Shield para controle Bluetooth;
- ✓ Chassi robótico 2WD, denominado “chassi mágico”;
- ✓ Conjunto de baterias ou pilhas;
- ✓ Fios de wire-up para conexões;
- ✓ Ferramentas como alicates, chaves de fenda e estrela (Philips).

1.5.2 Métodos.

O robô é constituído por módulos de software e hardware será controlado por um sistema de malha aberta. O programa é desenvolvido para plataforma Arduino em linguagem C sendo composto por dois módulos com as funcionalidades de:

- Controle de movimentação pelo acionamento de motores de CC;

-Comunicação via *bluetooth* para receber os comandos a partir de aplicação executada em dispositivo mobile.

A Figura 17, apresenta a arquitetura do sistema robótico proposto.

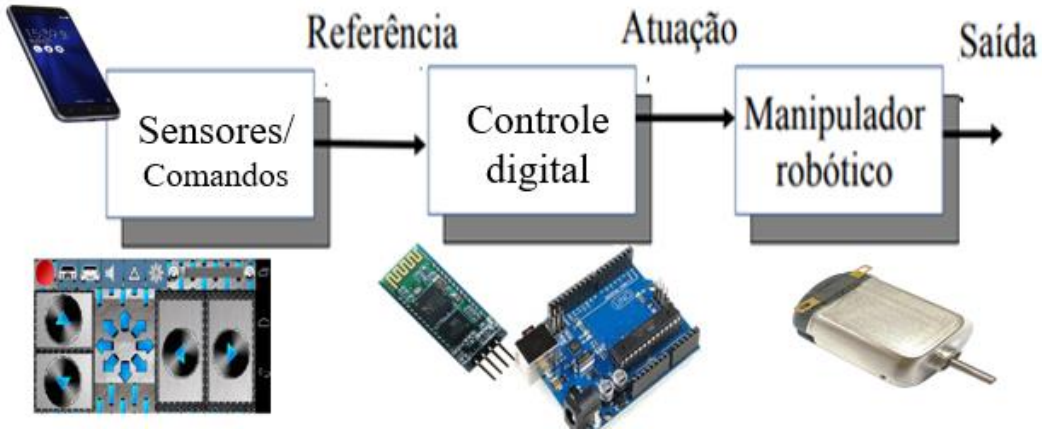


Figura 17. Arquitetura do sistema robótico com Arduino.

A aplicação executada no celular é responsável por enviar comando (referência na malha de controle aberta) via enlace bluetooth. No robô, um *shield Bluetooth*, recebe os comandos e via comunicação serial os dados são coletados pela placa Arduino UNO. A partir dos comandos recebidos via serial o algoritmo desenvolvido em C executado pelo Arduino aciona os motores de corrente contínua (Atuação) de forma a movimentar o robô para frente, para trás, curva a direita ou curva a esquerda.

O módulo de hardware do robô é composto por três partes: comunicação, inteligência e movimentação. A parte de comunicação é composta pelo *shield Bluetooth*. A parte de inteligência é constituída pela placa Arduino UNO que será responsável pela execução do algoritmo a ser desenvolvido. A parte de movimentação é composta por 2 motores de corrente contínua e *shield* ponte H responsável por fornecer a corrente elétrica necessária para acionamento dos motores.

O hardware do controle digital será implementado sobre um Chassi de acrílico comercial. Conhecido como Chassi robótico 2WD, esse modelo chassi é comprado motores DC 3-6v com caixa de redução 48:1 e 2 rodas com pneus de borracha. Na figura 18 temos uma foto do chassi que será utilizado para a montagem do modelo robótico.

Figura 18. Chassi robótico 2WD – modelo utilizado no desenvolvimento do modelo robótico.

A Figura 19, apresenta o fluxograma do código do controle digital (inteligência do robô) que será executado pelo Arduino e tomará a decisão sobre a movimentação do robô.

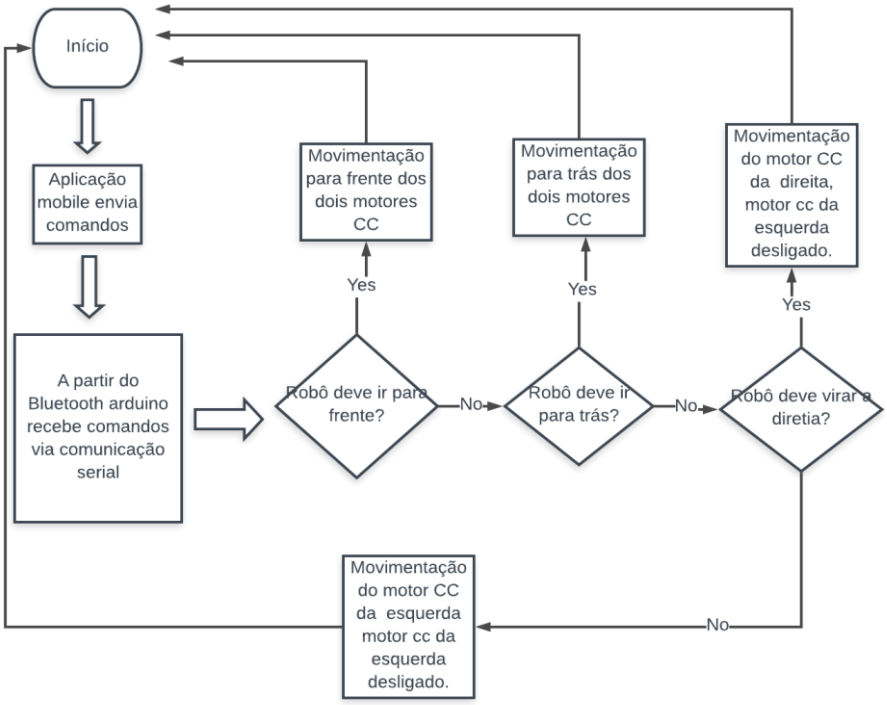


Figura 19. Fluxograma do código da inteligência do robô.

Com base no fluxograma apresentado na figura 19 e nos exemplos de códigos apresentados no tópico 1.4.1.2 Motores DC, aplicações e controle e 1.4.2.2

Comunicação serial, a seguir é apresentado o código que será implementado no robô. Para facilitar o entendimento da codificação, as instruções foram devidamente comentadas.

```
include<AFMotor.h> //biblioteca para utilização da shield ponte h para motores DC
AF_DCMotor motor1 (1); // referenciando os motores de acordo com shield ponte h
AF_DCMotor motor2 (2);
AF_DCMotor motor3 (3);
AF_DCMotor motor4 (4);
char command;

void setup() {
// função que permite configurar o arduino
 Serial.begin(9600);// habilita a comunicação serial que será utilziada para receber os dados via bluetooth
}

void loop() {
// ciclo executivo do arduino

 if(Serial.available() > 0){ // avalia o recebimento de caracter na porta serial
  command = Serial.read();// realiza leitura da porta serial e conteúdo é recebeido pela variável do tipo char
  Stop();
  switch(command){ // tomada de decisão
  case 'F':
   forward(); // executa função para o robô de movimentar para frente (a referência depende da ligação dos motores)
   break;
  case 'B':
   back(); // executa função para o robô de movimentar para trás (a referência depende da ligação dos motores)
   break;
  case 'L':
   left(); // executa função para o robô realizar curva a esquerda
   break;
  case 'R':
    right(); // executa função para o robô realizar curva a esquerda
    break;
    Stop(); // desliga os motores
```

```
    break;
  case 'S':
  break;
    }
  }
}

void forward()
// definição de função

{
  motor1.setSpeed(255); // define velocidade do motor
  motor1.run(FORWARD); // aciona o motor para frente
  motor2.setSpeed(255);
  motor2.run(FORWARD);
  motor3.setSpeed(255);
  motor3.run(BACKWARD);// aciona motor com rotação inversa ao FORWARD
  motor4.setSpeed(255);
  motor4.run(BACKWARD);
}

void back()
{
  motor1.setSpeed(255);
  motor1.run(BACKWARD);
  motor2.setSpeed(255);
  motor2.run(BACKWARD);
  motor3.setSpeed(255);
  motor3.run(FORWARD);
  motor4.setSpeed(255);
  motor4.run(FORWARD);
}

void left()
{
  motor1.setSpeed(255);
  motor1.run(FORWARD);
  motor2.setSpeed(0);
```

```
  motor2.run(RELEASE);
  motor3.setSpeed(255);
  motor3.run(FORWARD);
  motor4.setSpeed(0);
  motor4.run(RELEASE);
}

void right()
{
  motor1.setSpeed(0);
  motor1.setSpeed(RELEASE);
  motor2.setSpeed(255);
  motor2.setSpeed(FORWARD);
  motor3.setSpeed(0);
  motor3.setSpeed(RELEASE);
  motor4.setSpeed(255);
  motor4.setSpeed(FORWARD);
}

void Stop()
{
  motor1.setSpeed(0);
  motor1.run(RELEASE);
  motor2.setSpeed(0);
  motor2.setSpeed(RELEASE);
  motor3.setSpeed(0);
  motor3.run(RELEASE);
  motor4.setSpeed(0);
  motor4.setSpeed(RELEASE);
}
```

1.6 Considerações Finais

Apesar da utilização dos robôs desde a década de 60, os diversos avanços tecnológicos e a necessidade de desenvolvimento de novas aplicações, aumenta o interesse de pesquisadores no desenvolvimento na área. Devido a multidisciplinaridade do tema, a robótica também despertou o interesse dos profissionais de educação que aproveitam a popularização do Arduino e a importância da robótica para desenvolver pratica ativas de aprendizagem.

Neste contexto, neste trabalho buscamos apresentar uma visão geral sobre aspectos teóricos tanto de robótica como sistemas embarcados e através de uma oficina pratica fazer uso do Arduino para o desenvolvimento de um robô de controle manual de malha aberta.

Referências

Amareswar, E. .. (2017). "Multi purpose military service robot". 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), 684 - 686.

Arduino. (15 de 05 de 2016). Fonte: Arduino: https://www.arduino.cc/en/Main/Products

Banzi, M. e. (2015). "Primeiros passos com o Arduíno. 2a. ed". São Paulo: Novatec.

Basnayake, B. A. (2015). "Artificial Intelligence Based Smart Building Automation Controller for Energy Efficiency Improvements in Existing Buildings." International Journal of Advanced Information Science and Technology (IJAIST), 150 -156.

Buyya, R. E. (2016). "Internet of Things: Principles and Paradigms." Cambridge: Elsevier.

Coiffet, P. C. (2012). "An Introduction to Robot Technology." Springer.

da Silva Rodrigues, B. ,.-B. (2009). "Development of suspended gate field effect transistors array-based microsystem for pH measurements." IEEE Sensors, pp. 1623-1627.

Druin, A. H. (2000). "Robots for Kids: Exploring New Technologies for Learning." San Francisco: Morgan Kaufmann Publishers Inc.

Epson. (01 de 05 de 2018). Epson exceed your vision. Fonte: https://www.epson.de/en/products/robot/epson-scara-g10-851s

Ferreira, T. M. (2017). "Monitoramento de temperatura e umidade do acervo de negativos fotgráficos." Boletin técnico da faculdade de tecnologia de São Paulo, 129.

Fleming, D. (2006). "Cálculo - A Funções Limite Derivação Integração", 6ª Ed. São Paulo: Pearson.

Hamet, P. T. (2017). "Artificial intelligence in medicine. Metabolism: Clinical and Experimental", S36–S40.

Lathi, B. D. (2012). "Sistemas de Comunicações Analógicos e Digitais Modernos 4 Ed." São Paulo: LTC.

Martinewski, A. (2016). "Máquinas Elétricas. Geradores, Motores e Partidas, 1ª Ed." São Paulo: Sara'iva.

Mathew, K. C. (2016). "Method for Automating Medicare. International Research Journal of Engineering and Technology (IRJET)", 1952 -1954.

Mcroberts, M. (2011). "Arduíno Básico." São Paulo: Novatec.

Misbahuddin, S. Z.-W.-S. (2015). "IoT based dynamic road traffic management for smart cities. 12th International Conference on High-capacity Optical Networks and Enabling/Emerging Technologies (HONET)", 1-5.

Nof, S. Y. (1990). "Handbook of Industrial Robotics, 2nd Edition." New York City: John Wiley & Sons.

Ogata, K. (2011). "Engenharia de Controle Moderno - 5ª Ed." São Paulo: Pearson Prentice Hall.

Pieroni, A. A., & Brilli, M. (2018). "INDUSTRY 4.0 REVOLUTION IN AUTONOMOUS AND CONNECTED VEHICLE A NON-CONVENTIONAL APPROACH TO MANAGE BIG DATA." Journal of Theoretical & Applied Information Technology, 10-18.

Sciavicco, L. (2000). "Modeling and Control of Robotic Manipulators." London: Springer.

Severance, C. (2014). "Massimo Banzi: Building Arduino. IEEE:Computer, 47(1)", 11 - 12. doi:10.1109/MC.2014.19

Shim, J. K. (2016). "The Effects of a Robot Game Environment on Computer Programming Education for Elementary School Students." IEEE Transactions on Education, 60(2), 164 - 172.

Stallings, W. (2009). "Arquitetura e Organização de Computadores - 8ª Ed." São Paulo: Pearson.

Tanenbaum, A. S. (2011). "Redes de Computadores – 6ª Ed." São Paulo: Pearson.

Tocci, R. J., & Widmer, N. S. (2011). "Sistemas Digitais - Princípios e Aplicações - 11ª Ed." São Paulo: Pearson.

Tsai, L.-W. (1999). "Robot Analysis: The Mechanics of Serial and Parallel Manipulators." Wiley.

Verdonck, P. B. (2003). "Caracterização Elétrica de Tecnologia e Dispositivos Mos." São Paulo: Thomson.

Vosniakos, G. M. (1990). "Automated guided vehicle system design for FMS applications." International Journal of Machine Tools and Manufacture, 85-97.