Experiências usando PERT-CPM para desenvolvimento ágil de software

Thiago Jabur Bittar[1], Patrícia Duarte Silva[1], Larissa P. C. dos Santos[1], Luanna Lopes Lobato[1]

[1]Departamento de Computação – Universidade Federal de Catalão (UFCat)
Av. Dr. Lamartine Pinto de Avelar, 1120 - CEP 75704-020 – Catalão – GO – Brazil

{thiagojabur, luannalobato}@ufg.br

{patricia, larissap}@cc.grad.ufg.br

Abstract. *Identifying activities, establishing precedence, deadlines, and those responsible are initial tasks for good planning of software development. For this, there are important tools to help managers, such as the application of Critical Path Method (CPM) methods and the development and monitoring of the Program Evaluation and Review Technique (PERT) network diagram. In the combination of these methods we have the PERT-CPM, which, in this work, was applied on a case study about the production of a Web and mobile educational game using agile methods. As a result, with PERT-CPM it was possible to identify experiences and lessons for other projects.*

Resumo. *Identificar as atividades, estabelecer as precedências, os prazos de execução e os responsáveis são tarefas iniciais para o bom planejamento do desenvolvimento de software. Para tanto existem instrumentos importantes para auxiliar os gestores, como a aplicação dos métodos Critical Path Method (CPM) e a elaboração e acompanhamento do diagrama de redes Program Evaluation and Review Technique (PERT). Na combinação desses métodos tem-se o PERT-CPM, que, neste trabalho, foi aplicado sobre um estudo de caso sobre produção de um software do tipo jogo educativo Web e mobile usando métodos ágeis. Como resultados, com o PERT-CPM foi possível identificar experiências e lições para outros projetos.*

1. Introdução

Um produto de qualidade no cenário competitivo atual do sistema industrial mundial é de significativa importância para uma instituição. Manter o nível de qualidade de um produto está diretamente ligado com o controle da produção do mesmo, da sua concepção à entrega e uso pelo cliente. O controle sobre o desenvolvimento está inserido no complexo arcabouço de custos, prazos, posicionamento de mercado e níveis de qualidade desejados. Este controle permite a viabilidade, acompanhamento gerencial, produtividade no desenvolvimento, operação e manutenção de produtos [Vargas, 2009].

Nesse sentido, a partir do momento em que o desenvolvimento de um produto é planejado, com a identificação das atividades de produção e o tempo que cada uma pode levar para ser cumprida, é possível determinar um prazo ideal para a entrega do mesmo.

Mas não basta simplesmente estabelecer datas, é necessário estar atento a imprevistos, analisar diariamente a evolução das atividades e reorganizar as mesmas e/ou a equipe, se necessário. Por exemplo, ao identificar o atraso de uma atividade que afeta outra, pode haver a alocação de mais membros na equipe para a conclusão da atividade em questão. Dessa maneira, a qualidade do produto não precisará ser diminuída para acelerar o processo de entrega das atividades pendentes [Martins e Laugeni, 2005].

Na indústria de software a necessidade da qualidade é a mesma. Um software de qualidade é aquele que satisfaz os requisitos solicitados pelo cliente, é de fácil manutenção, tem boa performance, é confiável e fácil e acessível de se usar. Manter a qualidade com um preço final justo e respeitando prazos de entrega é essencial. Então, para reduzir o custo de venda de um software é feita a distribuição do custo entre vários clientes, até mesmo em âmbito global. Ou seja, o software deve ser planejado para atender o máximo possível de clientes, sendo assim, desenhado uma única vez e distribuído diversas.

Desta maneira o desenvolvimento de software deixou de ter características artesanais e começou a assemelhar-se com produções industriais. As empresas de desenvolvimento de software passaram a focar seus esforços na produção de componentes, objetivando o reuso, com entregas rápidas e com qualidade.

Assim, satisfazer o cliente cumprindo o prazo estabelecido para entrega com todos os atributos prometidos para o produto deve ser o foco de todas instituições. O planejamento e o gerenciamento são as ferramentas necessárias para o alcance e manutenção da qualidade mantendo os prazos previstos.

Na falta de planejamento e gerenciamento dos projetos, os mesmos são cancelados, não terminam no prazo determinado, são encerrados prematuramente ficando sem qualidade, entre outras consequências desastrosas. A quebra dos prazos de entrega de softwares foi uma das características mais identificadas durante a Crise do Software [Pressman e Maxim, 2016], revelando a suma importância de um gerenciamento produtivo eficiente durante o desenvolvimento de softwares.

Em termos de gestão, princípios de controle de atividades foram aplicados, nesta pesquisa, em um estudo de caso referente ao desenvolvimento de um jogo educacional para as plataformas *Web* e *mobile*. As atividades para o desenvolvimento do produto foram estabelecidas de acordo com regras contemporâneas da Engenharia de Software (ES) e de metodologia ágil (incluindo, por exemplo, gerenciamento constante de riscos), juntamente com a determinação do seu prazo de conclusão e suas atividades precedentes. Para a gestão destas atividades foi aplicado o método PERT-CPM para identificar os prazos e o caminho crítico para o desenvolvimento planejado do projeto. Tal método não é novo, sendo proposto inicialmente por Wiest e Levy (1969) e mais tarde com um tutorial prático de Swanson e Woolsey (1974), porém a sua utilização nos moldes aqui mencionados, em abordagem prática voltada para produção de produtos de software com metodologia ágil (geralmente utilizada em pequenas empresas e *startups*) tem aspectos de inovação, podendo os preceitos aqui descritos auxiliarem outras equipes.

Este artigo está organizado da seguinte maneira: na Seção 2 tem-se a apresentação dos conceitos envolvidos; já na Seção 3 os trabalhos relacionados são relatados, com a identificação de diferenciais desta pesquisa; na Seção 4 tem-se as fases de ES com métodos ágeis detalhadas e incluídas na gestão, algo que difere esta pesquisa das demais verificadas na Seção 3 e que possivelmente auxiliará outras equipes. Por fim, na Seção 5 é apresentado o estudo de caso realizado, seus resultados e análise e na Seção 6 tem-se as considerações finais deste trabalho.

2. Conceituação

2.1. Desenvolvimento ágil

A primeira e principal proposição para o desenvolvimento ágil de software foi realizada no período de 11 a 13 de fevereiro de 2001 no chamado "Manifesto Ágil" por um grupo de 17 especialistas em desenvolvimento de software que estavam insatisfeitos com insucessos dos métodos tradicionais. Eles também formaram na ocasião a Aliança Ágil (Beck *et al.*, 2001). Os valores principais defendidos por esses especialistas são:

- 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.

Nas metodologias de desenvolvimento ágil o software é produzido de modo incremental, fazendo com que alterações sejam absorvidas ao longo das sucessivas versões, sempre com partes funcionais [Teles, 2004] [Teles, 2005] [Cockburn, 2002] [Sbrocco, 2012].

Foram estabelecidos 12 princípios para se alcançar a agilidade, que são:

- **Valor:** satisfaça o cliente com entrega adiantada e contínua;
- **Flexibilidade:** aceite os pedidos de alterações, pois os processos ágeis devem aproveitar essas mudanças como vantagem competitiva;
- **Frequência:** entregue software em funcionamento frequentemente, em intervalos custos;
- **União:** a equipe da área comercial e os desenvolvedores devem trabalhar em conjunto;
- **Motivação:** use técnicas e ferramentas para ter a equipe sempre motivada;
- **Comunicação:** tenha conversas abertas para comunicação mais efetiva e eficiente;
- **Funcionalidade:** use como medida de progresso o software em funcionamento;
- **Sustentabilidade:** nunca esgote todos os seus recursos, processos ágeis devem promover o desenvolvimento sustentável;
- **Revisão:** valorize a excelência técnica;
- **Simplicidade:** maximize o trabalho realizando apenas o essencial;

- **Organização:** tenha as equipes auto organizadas, com divisão de tarefas e responsabilidades feitas com auxílio de cada integrante do time;
- **Autoavaliação:** a equipe deve avaliar seu trabalho em intervalos regulares.

A partir desses valores foram desenvolvidas várias metodologias ágeis, sendo que entre as mais populares estão a eXtreme Programming (XP) e a SCRUM, com foco significativo nas atividades práticas de desenvolvimento e na gestão dos projetos [Fernandes e Almeida, 2010].

2.2. Método do Caminho Crítico e o diagrama de redes PERT

O Método do Caminho Crítico (em inglês *Critical Path Method*, CPM) e o diagrama de redes PERT (do inglês: *Program Evaluation and Review Technique*), são técnicas de planejamento baseadas em modelos de rede de atividades [Wiest e Levy, 1969] [Swanson e Woolsey, 1974].

Esses métodos (que geralmente são aplicados em conjunto) têm sido utilizados em programas de gestão nas mais diversas áreas, tais como indústria, agricultura, defesa nacional e pesquisa científica [Swanson e Woolsey, 1974]. A motivação para seu uso advém do fato de que na maioria dos projetos do cotidiano humano tem-se um rol complexo de atividades inter-relacionais com tempos definidos, umas que podem ser executadas em paralelo e outras que devem aguardar o desenvolvimento e conclusão de outras.

O PERT-CPM, então, fornece uma significativa base teórica gráfica e matemática para os gestores de projetos para auxílio ferramental no planejamento: i) organizando e explicitando o progresso das atividades em seus caminhos e ii) lidando com o impacto de fatores incertos e no acompanhamento de custos e atrasos [Swanson e Woolsey, 1974] [Christoph, 2001].

No CPM, o caminho é a ordem em que as tarefas são realizadas de acordo com a sequência a ser seguida. O chamado Caminho Crítico é, então, a sequência que leva mais tempo para ser finalizada, indicando o tempo máximo total que um projeto levará. É importante ressaltar que um projeto pode ter vários caminhos críticos, com um empate na soma de tempos totais.

3. Trabalhos Relacionados

Para a busca de trabalhos relacionados foram utilizadas inicialmente três bibliotecas digitais científicas mundiais, a ACM DL[1], a IEEE Xplore[2] e a Springer Link[3], com a meta específica de encontrar publicações com aderência total ao foco desta pesquisa, envolvendo ES, PERT-CPM e desenvolvimento ágil. No entanto tais bases não retornaram resultados substanciais considerando a *string* definida de busca em todos os metadados que endereçam essa meta: pert AND cpm AND software AND agile. O uso

[1] https://dl.acm.org

[2] https://ieeexplore.ieee.org

[3] https://link.springer.com

de uma variação dessa *string* excluindo-se o termo "cpm" se mostrou ineficaz por conta desse item ser utilizado para outras definições.

Uma possível explicação para o resultado encontrado nestas bases é que o método buscado, PERT-CPM, é genérico, servindo para várias áreas do conhecimento e não ter uma maior exploração específica em desenvolvimento ágil após seu lançamento por Swanson e Woolsey (1974) na área da programação matemática no veículo ACM Special Interest Group on Mathematical Programming, SIGMAP. Na verdade, é possível perceber que o método nasceu de um rigoroso formalismo matemático, o que pode, numa primeira e rasa análise, distanciar seu uso nos métodos ágeis, que procuram desburocratizar ao máximo o desenvolvimento. Análise essa que é discutida nas próximas seções deste artigo.

Como modo de ampliar a busca e enriquecer esta investigação foi aberta, então, a possibilidade de uso da pesquisa do Google considerando publicações científicas e também incluída a análise do uso do método fora do desenvolvimento de software. O resultado pode ser visto na discussão do restante desta seção.

Göksu e Catovic (2012) relatam que devido aos efeitos crescentes da globalização em vários ambientes de negócios, espera-se que a indústria de manufatura seja eficiente e eficaz, com entregas de qualidade no prazo correto. De acordo com isso, no planejamento, programação e controle de um projeto, que é uma combinação de várias atividades, são testadas técnicas de gerenciamento de projetos PERT-CPM. Para esse teste empírico, a pergunta de pesquisa que é feita foi: "Como a implementação do PERT-CPM influencia na eficácia e a eficiência de uma empresa de móveis chamada Dallas".

A resposta para essa pergunta é relevante para indicar a importância desses métodos na redução do tempo e dos custos de conclusão do projeto. Os dados retirados da empresa de móveis foram combinados com revisões de literatura. Esse estudo verificou como determinar as atividades envolvidas nos processos de fabricação da empresa selecionada e como demonstrar possíveis benefícios e inconvenientes que esses métodos de planejamento podem criar na organização. Implicações dessa pesquisa são a avaliação do tempo de conclusão do projeto e o controle dos recursos, para que o projeto fosse concluído dentro do tempo e custo planejados usando os métodos mencionados. No final do estudo, tem-se relatos para ajudar outros indivíduos, bem como as empresas, a entender melhor o conceito do método PERT-CPM na redução do tempo e dos custos de conclusão do projeto.

Tal trabalho é análogo ao realizado nesta pesquisa, no entanto o foco aqui está na indústria de software ágil, que apresenta complexidades próprias do domínio, como uma severa e constante análise de riscos, feita em paralelo com as demais. Outro diferencial desta pesquisa apresentada aqui é que são apresentadas as atividades e todo arcabouço gráfico feito, pois Göksu e Catovic (2012) focam na parte inicial teórica, não mostrando visualmente as atividades e gráficos relacionados.

Lu *et al.* (2011), por sua vez, apresentam a combinação das técnicas PERT e CPM para o planejamento do projeto "Pré-emprego" para graduandos universitários. O projeto "Pré-emprego" consiste em combinar atividades desde a elaboração e distribuição de currículos até a preparação para as entrevistas. O foco do trabalho foi,

então, o de resolver questões comuns a todos projetos como: “Quando o projeto inteiro pode ser concluído?”, “Qual é o melhor método para concluir o projeto com custo mínimo?”, “Os recursos são suficientes para concluir o projeto no prazo?”, entre outras questões de mesma importância.

O trabalho mais alinhado ao tema desta pesquisa foi o de Ashkenazi *et al.* (2010), que realizaram um estudo de caso em uma empresa de desenvolvimento de software utilizando o método ágil SCRUM o a técnica COCOMO II para estimar o esforço de atividades. Apesar do foco da pesquisa em questão estar no uso do SCRUM e não existir muitos detalhes na publicação (é um resumo estendido de apenas 4 páginas sem a imagem da rede utilizada) a conclusão é de que o PERT-CPM trouxe benefícios no *design* de cada fase do projeto, permitindo a detecção do(s) caminho(s) crítico(s) e chamando a atenção para o progresso das atividades ali presentes. No entanto os autores indicam que o PERT-CPM não é estritamente necessário no SCRUM, pois são utilizados projetos bem particionados em “semi independentes” subprojetos com pequenas equipes de desenvolvedores com grande flexibilidade de execução de atividades.

Ashkenazi *et al.* (2010), terminam relatando que os líderes de projeto devem saber sim fundamentalmente o caminho crítico, usando ou não um método formal. Ou seja, PERT-CPM pode ser um bom instrumento para o acompanhamento de projeto por estes líderes, especialmente quando se tem complexos caminhos de atividades. Atrasos são péssimos em todos os projetos, incluindo os ágeis e todos os métodos devem ser utilizados para ajustar os desenvolvimentos.

4. Fases da Engenharia de Software com Métodos Ágeis

Na ES, com uso de métodos ágeis, o processo de desenvolvimento de sistema computacional é constituído de um conjunto de atividades com entregas de partes do software em períodos curtos de tempo, chamados genericamente de iterações. Essas atividades são agrupadas em fases, como: levantamento de requisitos, análise de riscos, definição sobre a arquitetura do projeto, elaboração de testes e implementação do código. Em cada fase são definidas, além das suas atividades, as funções e responsabilidades de cada membro da equipe, e como produto resultante, os artefatos que são incrementalmente melhorados.

As fases descritas nas próximas subseções são um apanhado dos conceitos de metodologias ágeis que foram usados neste projeto de desenvolvimento e que deram azo para a criação das tarefas gerenciadas com os métodos ágeis. Tais fases são similares às dos métodos tradicionais, amplamente descritas na literatura; a mudança está no uso de princípios da metodologia ágil.

4.1. Levantamento de requisitos

Nesta fase o problema a ser solucionado pelo software deve ser compreendido, fornecendo aos desenvolvedores e clientes a mesma visão do que deve ser construído. É feito, então, um levantamento e priorizadas as necessidades dos futuros usuários do software. Essas necessidades definidas são denominadas como requisitos. O levantamento e a análise de requisitos irão definir o que o sistema deve fazer, antes de definir como o sistema irá fazer [Pressman e Maxim, 2016].

Os requisitos de software podem ser divididos em dois tipos básicos: os funcionais e os não funcionais. Os primeiros são aqueles que especificam uma função que o sistema ou componente deve ser capaz de realizar. Esses são requisitos de software que definem o comportamento do sistema, ou seja, o processo ou transformação que componentes de software ou hardware efetuam sobre as entradas para gerar as saídas. Tais requisitos capturam as funcionalidades sob o ponto de vista do usuário. Já os não funcionais não estão diretamente relacionados à funcionalidade de um sistema, definindo características do software como método de desenvolvimento, tempo, espaço, sistema operacional, ambiente, usabilidade, ergonomia e acessibilidade.

4.2. Análise de riscos

A análise de riscos consiste na identificação de possíveis eventos que possam afetar o projeto. Esses eventos são denotados como riscos. Os riscos identificados são avaliados e atividades são definidas para reduzir a possibilidade de ocorrência ou o impacto dos riscos mais prejudiciais.

O gerenciamento e monitoramento dos riscos devem ocorrer durante todo o processo de desenvolvimento do projeto com o objetivo de mitigar os riscos encontrados, trazendo o menor impacto negativo ao projeto.

Em uma metodologia ágil a identificação de riscos contínua e a entrega de partes funcionais em pequenos ciclos com grande interação com o cliente auxilia na mitigação e na resolução dos problemas. A tendência é que o risco seja rapidamente controlado numa metodologia ágil. No caso investigado aqui, tem-se a possibilidade de rápidas mudanças no cronograma com ajustes sendo feitos na rede de atividades do projeto.

4.3. Definição sobre arquitetura do projeto

A definição da arquitetura do projeto considera como o sistema funcionará e estará organizado internamente, para que os requisitos do cliente possam ser atendidos. Os artefatos produzidos por essa fase consistem na descrição computacional de como o código-executável deve ser construído.

A arquitetura descreve tanto a estrutura lógica do funcionamento do software quanto a arquitetura física de componentes físicos que pertencem ao software. Na arquitetura lógica é descrito claramente o funcionamento lógico do software em termos de funções, variáveis, componentes e classes. Já na arquitetura física é descrito o conjunto de arquivos fontes, arquivos de dados, bibliotecas, executáveis e outros.

4.4. Elaboração e execução dos testes

A fase de elaboração e execução dos testes busca identificar as situações de uso do software e o comportamento desejável que o software deve entregar. O objetivo dos testes é validar o produto, testando cada funcionalidade de cada módulo levando em consideração as especificações feitas na fase de projeto da arquitetura. Os testes elaborados são executados conforme os componentes do sistema são implementados, produzindo um relatório com as informações relevantes sobre os erros encontrados.

4.5. Codificação

Nesta fase, o sistema é codificado a partir da materialização dos artefatos descritos na fase de definição da arquitetura do projeto. O projeto passa a ser executado computacionalmente em linguagem de programação atendendo os requisitos levantados.

Vale ressaltar que em equipes inexperientes em relação à linguagem utilizada é necessário incluir uma etapa anterior à implementação para haver o treinamento da equipe.

4.5. Manutenção

Mesmo no melhor cenário de identificação de requisitos e desenvolvimento o software precisa software modificações evolutivas e corretivas depois de ser entregue. A manutenção deve ser uma operação viável em que há a aplicação de cada uma das fases descritas anteriormente aos módulos existentes em vez de criar um novo software.

Na utilização de metodologias ágeis é preciso que as equipes pensem na estruturação arquitetural do código fonte do projeto como um todo e utilizem boas práticas, garantindo uma futura manutenção rápida para viabilidade da evolução do software. Não basta então, entregar partes funcionais do software funcionando, elas têm que estarem calcadas numa estrutura sólida e bem projetada.

5. Estudo de Caso

Segundo Yin (2010, p. 19), "O estudo de caso é apenas uma das muitas maneiras de se fazer pesquisa em ciências sociais. Experimentos, levantamentos, pesquisas históricas e análise de informações em arquivos são alguns exemplos de outras maneiras de se realizar pesquisa. Cada estratégia apresenta vantagens e desvantagens próprias, dependendo basicamente de três condições: a) o tipo de questão da pesquisa; b) o controle que o pesquisador possui sobre os eventos comportamentais efetivos; c) o foco em fenômenos históricos, em oposição a fenômenos contemporâneos".

Em geral, os estudos de caso representam a estratégia preferida quando se colocam questões do tipo "*como*" e "*por que*", quando o pesquisador tem pouco controle sobre os eventos e quando o foco se encontra em fenômenos contemporâneos inseridos em algum contexto da vida real. Pode-se complementar esses estudos de casos com dois outros tipos - estudos "*exploratórios*" e "*descritivos*".

Para a investigação proposta neste trabalho foi utilizada a metodologia de Yin (2010) em conjunto com as diretrizes de Runeson e Höst (2009), que são específicas para ES.

Nesse cenário, foi utilizado o seguinte estudo de caso com contexto fictício para testar a eficiência da gestão de atividades usando o PERT-CPM: "Você acabou de entrar em uma equipe de desenvolvimento de software de uma pequena empresa que solicitou o desenvolvimento de um jogo, desde a sua criação até a implementação e testes. Essa empresa não se preocupa com ES, seus gestores alegam que não tem pessoal e recursos para realizarem as atividades de ES. A empresa não faz documentação. O jogo deve ser simples e educativo (abordando conceitos básicos de matemática, biologia ou geografia) e deve ser feito para as plataformas *Web* e *mobile*".

A partir deste contexto, para o desenvolvimento deste jogo foram definidas por dois pesquisadores especialistas em ES as atividades principais, o tempo de execução estimado e a precedência destas atividades conforme apresentado na Tabela 1. Apesar do contexto ser fictício a implementação do jogo e a interação entre membros da equipe e uso de metodologia ágil foi real, com software sendo entregue ao final de uma disciplina "Engenharia de Software II" do curso de Ciência da Computação da Universidade Federal de Catalão (UFCat).

Para efeitos operacionais e para simplificar o estudo de caso foi considerado que após a conclusão de uma atividade a outra subsequente não paralela só começa no dia seguinte.

Tabela 1. Alocação das atividades (Fonte: Dados da pesquisa).

Atividade	Precedência	Duração em dias
Início	-	0
A - Definição do tema	Início	3
B - Pesquisar sobre jogos similares	A	2
C - Pesquisar dados sobre o público alvo	A	3
D - Levantamento de requisitos e *game design*	B, C	3
E - Análise de riscos	Início	2
F - Definição do cronograma	D	2
G - Definição sobre arquitetura do projeto	D	2
H - Escrever documentação	A	21
I - Elaboração dos Layouts	D	2
J - Elaboração dos níveis do jogo	I	3
K - Implementação do código	J	10
L - Elaboração dos testes	K	3
M - Execução dos testes	L	3
N - Escrita do artigo	M	17
O - Escrita do ofício	N	5
P - Apresentação/Entrega	O	1
Fim	O,P,H,F,G,E	-

Com os dados da Tabela 1 é possível identificar que existem 6 caminhos, que são rotas seguindo os arcos a partir do nó "Início" até o nó "Fim". O comprimento de um caminho é a soma das durações das atividades sobre o caminho. Esses caminhos são apresentados na Tabela 2.

Tabela 2. Caminhos e seus comprimentos (Fonte: Dados da pesquisa).

Caminho	Comprimento
Início - A - C - D - I - J - K - L - M - N - O - P - Fim	53
Início - E - Fim	2
Início - A - H - Fim	24
Início - A - C - D - G - Fim	11
Início - A - B - D - I - J - K - L - M - N - O - P - Fim	52
Início - A - B - D - G - Fim	10
Início - A - B - D - F - Fim	10
Início - A - B - D - I - J - K - L - M - N - O - Fim	51
Início - A - C - D - I - J - K - L - M - N - O - Fim	52

Com a somatória dos tempos das atividades foi possível identificar claramente o caminho crítico, que é aquele de maior comprimento. Em nosso caso trata-se do primeiro caminho da Tabela 2, com 53 dias totais para completar o projeto.

Na Figura 1, apresentada a seguir, tem-se o primeiro esboço do diagrama de rede sobre as atividades levantadas, feito com a técnica de lápis e papel. A opção pelo uso dessa técnica foi o de ter maior liberdade de trabalho e para teste dos conhecimentos dos conceitos estudados em sua totalidade. Outro ponto importante e que fica como lição aprendida é de não ter sido encontrado pelos autores desta pesquisa um software simples e gratuito para geração deste gráfico.

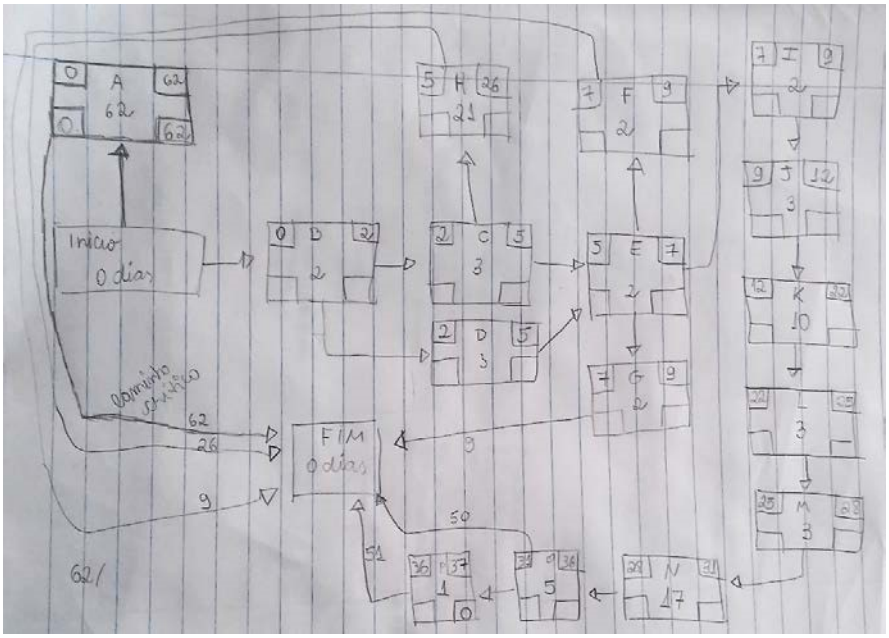

Figura 1. Primeira versão do Gráfico PERT-CPM feito em lápis e papel (Fonte: Autoria própria).

Na Figura 1, cada caixa representa uma atividade e as setas representam a precedência das atividades. A seta é direcionada para a(s) atividade(s) seguinte(s). A leitura do diagrama começa pela atividade "Início", que começa com 0 dias. Em seguida, a atividade "Início" irá apontar para as atividades "A" e "B", que ocorrerão em paralelo após o início do projeto. A atividade "B" aponta para a atividade "C" que só poderá ser iniciada assim que a atividade "B" for concluída.

Para facilitar a identificação das atividades e suas informações referentes, foi definida a estrutura da representação das caixas. Em todas as caixas, no meio é colocada uma letra maiúscula do alfabeto que representa uma atividade. E logo abaixo da atividade, é alocado o tempo de execução da atividade.

Na Figura 2, apresentada a seguir, é descrito o que cada componente da caixa representa.

Figura 2. Descrição do que cada componente da caixa (Fonte: Autoria própria).

Na Figura 3, apresentada a seguir, tem-se a finalização do diagrama de rede, com o uso de ferramenta computacional de editoração gráfica.

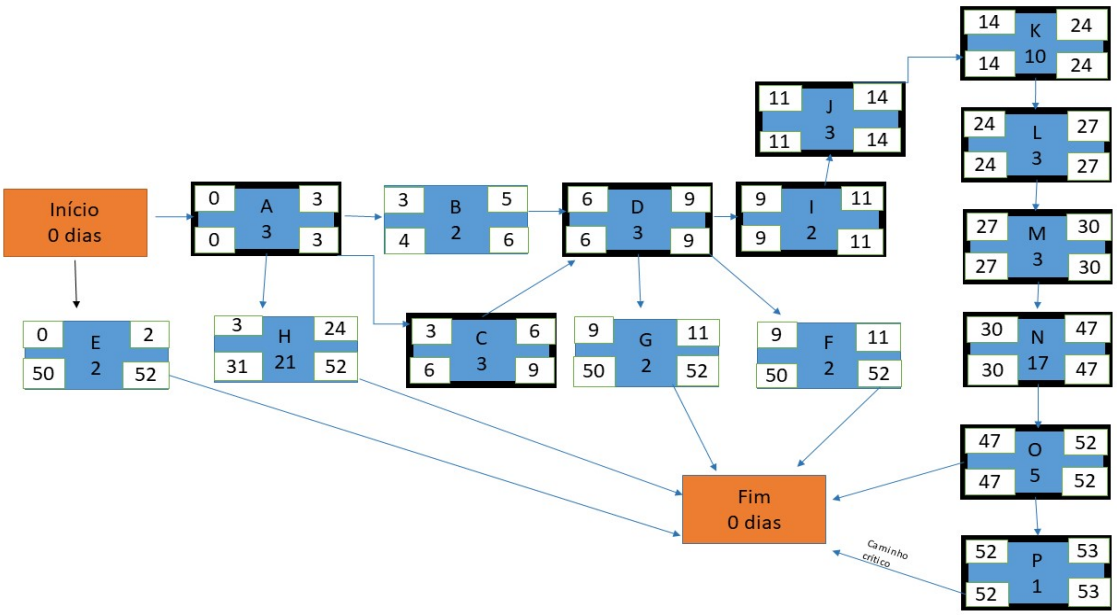

Figura 3. Gráfico CPM final de atividades do desenvolvimento do produto (Fonte: Dados da pesquisa)

Na Figura 3 o gráfico foi refeito e recalculado após um estudo mais aprofundado sobre a aplicação do método PERT-CPM. Este gráfico possui caixas, descritas anteriormente, que representam cada atividade planejada do projeto; as setas, que apontam a sequência das atividades; a representação do caminho crítico, o caminho com as atividades mais longas; e as informações de início e término mais cedo e mais tarde de cada atividade.

Para construir o gráfico da Figura 3 primeiramente foi definido o "Início" com 0 dias para execução. Em seguida, todas as precedências foram ajustadas conforme a Tabela 1, até chegar na atividade "Fim" que também contém 0 dias como tempo de execução. Posteriormente, foi definido o caminho crítico (destacado pela cor preto no gráfico de rede CPM da Figura 3) com 53 dias. É importante ressaltar que essa informação define o tempo total que o projeto leva para terminar.

Após identificar o caminho crítico, foram calculados o início mais cedo, término mais cedo, início mais tarde e término mais tarde de todas as atividades do caminho, exceto das atividades "Início" e "Fim" (que são marcos gerenciais). Estes cálculos foram feitos do seguinte modo:

- **Início mais cedo (IMC):** para calcular o início mais cedo da primeira atividade, sempre é definido o valor da quantidade de dias de execução da atividade "Início". No restante das atividades, é somado o início mais cedo e a quantidade de dias de execução da atividade anterior.
- **Término mais cedo (TMC):** no término mais cedo da primeira atividade, sempre é colocada a quantidade de dias da primeira atividade. Para as demais atividades, é somado o término mais cedo da anterior com a quantidade de dias da atual.
- **Início mais tarde (IMT):** é feito calculando o término mais tarde da atividade atual menos a quantidade de dias da atividade atual.
- **Término mais tarde (TMT):** é calculado da atividade final para a inicial (no inverso da ordem natural de execução). Para as atividades que estão ao lado do fim, é utilizado o valor do dia do início mais tarde da última atividade do caminho crítico.

Depois de identificar o caminho crítico do projeto, é importante identificar também as folgas das atividades. Existem três tipos de folga, a saber:

- **Folga total:** quantidade de tempo que uma atividade pode atrasar sem atrasar a data do projeto.
- **Folga livre:** quantidade de tempo que uma atividade pode atrasar sem atrasar as atividades sucessoras.
- **Folga do projeto:** quantidade de tempo que o projeto pode atrasar sem que atrase a data de entrega imposta pelo cliente.

Com o gráfico concluído é possível identificar com mais clareza as folgas de cada atividade. Na Figura 3 é possível verificar que a atividade H tem **28 dias de folga livre**. Ela pode ser iniciada a partir do terceiro dia ou no máximo no trigésimo primeiro dia de execução do projeto, mas deve ser concluída até no quinquagésimo segundo dia.

6. Conclusão

Entregar um produto de qualidade no prazo é o objeto de desejo da maioria das instituições de bens e serviços, característica essa que agrega maturidade produtiva a essas organizações. O alcance dessa tão sonhada qualidade pode ser facilitado por processos de gerenciamento e planejamento ágil de projetos. Controlar o desenvolvimento de um produto se resume em monitorar custos, prazos e níveis de qualidade desejáveis. O método de controle correto irá permitir a produtividade no desenvolvimento, operação e manutenção de produtos.

A produção de software dos dias atuais apresenta características de processos industriais, portanto os métodos de controle de produção utilizados nas indústrias de grande porte também passam a ser aplicáveis com benefícios no desenvolvimento deste produto, até mesmo no desenvolvimento ágil. Isso pôde ser verificado com sucesso no ensaio do estudo de caso apresentado neste artigo.

O controle de desenvolvimento envolveu a identificação de atividades elementares, suas precedências e seus prazos de execução, o que não burocratizou o trabalho de planejamento e que frequentemente já é feito pelos líderes de projeto de modo não sistematizado. O gerenciamento destas atividades possibilitou determinar o prazo ideal de entrega do produto e as atividades que devem ser tratadas com mais atenção. Com base nisso, foi aplicado o método PERT-CPM envolvendo o planejamento de um software de jogo educacional para as plataformas *Web* e *mobile*.

A partir da construção do gráfico PERT-CPM foi possível determinar com custo de trabalho baixo a melhor estratégia de gerenciamento das atividades. Ao encontrar o caminho crítico e tendo as atividades pré-estabelecidas foi possível determinar os responsáveis e impor datas de entrega com acompanhamento contínuo nas iterações. As atividades identificadas com folga podem ser adiadas e colocadas em iterações finais, priorizada a entrega no prazo de todos os requisitos do produto solicitado.

Não planejar bem as atividades ou contar apenas com o feeling dos líderes agilistas pode dificultar a identificação de gargalos de prazos em atividades e afetar gravemente nas entregas do software. Além disso, atividades executadas ao acaso podem causar atraso em cascata, situação essa em que as atividades mais onerosas são deixadas para o final, estourando o prazo de entrega e afetando a qualidade.

Por fim, verificou-se que o número de horas para se especificar as atividades, responsáveis e para desenvolver o diagrama PERT-CPM é pequeno se comparado ao tempo total estimado para o projeto, sendo uma atividade benéfica mesmo em empresas pequenas que utilizam desenvolvimento ágil.

Referências

Almeida M. e Fernandes J. M. Classification and Comparison of Agile Methods. 2010 Seventh International Conference on the Quality of Information and Communications Technology (QUATIC), Porto, Portugal, 2010, pp. 391-396.

Ashkenazi L., Shmilovici A., Ben-Gal I., Agile Software Development: A Case Study in a Software Company, Proc. of the 16th Industrial Eng. & Management Conference, Tel Aviv, March 23-24, 2010.

Beck, K.; Beedle, M.; Van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R. C.; Mellor, S.; Schwaber, K.; Sutherland, J.; Thomas, D. Manifesto para Desenvolvimento Ágil de Software. 2001. Disponível em: <http://www.agilemanifesto.org/iso/ptbr/>. Acessado em 14 de fevereiro de 2018.

Christoph. S.T.: Job shop scheduling with alternative process plans. 2001. Int. J. Production Economics 74; Páginas 125-134.

Cockburn, A. Agile Software Development. ISBN 0201699699, 978-0201699692. 304 p. Addison-Wesley, 2002.

Göksu, A.; Catovic, S. (2012). Implementation Of Critical Path Method And Project Evaluation And Review Technique. 3rd International Symposium on Sustainable Development. 31th May-1st June 2012. Sarajevo, Bosnia and Herzegovina.

Lu X., Guan S., Tian R., Zhang W. Using PERT/CPM Technology for the Development of College Graduates Seeking Employment in Project Planning. In: Liu C., Chang J., Yang A. (eds) Information Computing and Applications. ICICA 2011. Communications in Computer and Information Science, vol. 244. Springer, Berlin, Heidelberg. 2011.

Martins, P. G.; Laugeni, F. P. Administração da Produção. São Paulo: Saraiva, 5ª Ed., 2005.

Pressman, R. S.; Maxim, B. R. Engenharia de Software. Uma Abordagem Profissional. 2016. 968 páginas. 8a. Edição. Editora AMGH. ISBN: 8580555337.

Runeson, P., Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131.

Sbrocco, J. H. T. C.; Macedo, P. C. de. Metodologias Ágeis: Engenharia de Software sob medida. 1a ed. São Paulo: Érica, 2012. ISBN 978-85-365-0979-2. 254 páginas.

Swanson, H. S.; Woolsey, R. E. D. 1974. A PERT-CPM tutorial. SIGMAP Bull. 16 (Abril de 1974), 54-62. DOI=http://dx.doi.org/10.1145/1217031.1217043.

Teles, V. M. Extreme Programming. São Paulo: Novatec Editora, 2014. 2ª. Ed. ISBN: 978-85-7522-400-7. 328 páginas.

Teles, V. M. Um estudo de caso da adoção das práticas e valores do Extreme Programming. 181 p. Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, UFRJ, 2005.

Vargas, R. Gerenciamento de projetos – Estabelecendo diferenciais competitivos. 7ª Ed. Rio de Janeiro: Brasport, 2009. 276 páginas.

Wiest, J. D. e Levy; F.K. A Management Guide to PERT/CPM. Prentice-Hall, Englewood Cliffs, New Jersey, 1969.

Yin, R. Estudo de caso: planejamento e métodos. Porto Alegre: Bookman, 2ª Ed., 2001. 205 páginas. ISBN 8573078529.