Definição de Processos de Software baseada em uma Arquitetura de Componentes de Processo

Luiz Carlos M. Ribeiro, Cristiane S. Ramos, Kamilla Crozara, Hilmer R. Neri, Eusyar Alves, Rejane M. C. Figueiredo

Faculdade Gama, Campus Gama – Universidade de Brasília (UnB).

{lcarlos, cristianesramos, hilmer, rejanecosta}@unb.br,
holanda.kamilla@gmail.com, eusyar@hotmail.com

Abstract. *In this work we present a standard process model from a component architecture of Software Process based on Software & Systems Process Engineering Meta-Model Specification (SPEM), that allows defining software and systems development processes and their components. From the standard process model we proceeded to the reuse of the process's components in order to establish a process in use in the context of game development using mechanisms related to the types of process variability.*

Resumo. *Neste trabalho é apresentado um modelo de processo padrão elaborado a partir de uma arquitetura de componentes de Processo de Software baseada no meta-modelo Software & Systems Process Engineering Meta-Model Specification (SPEM), que permite definir processos de desenvolvimento de software e sistemas e seus componentes. A partir do modelo de processo padrão foi realizada a reutilização de componentes de processo para o estabelecimento de um processo em uso no contexto de desenvolvimento de jogos eletrônicos utilizando mecanismos relacionados aos tipos de variabilidade de processo.*

1. Introdução

As organizações de desenvolvimento de software têm buscado cada vez mais aumentar a sua participação e competitividade no mercado, investindo na melhoria de seus processos de software para que possam apoiar tanto a redução dos custos de produção, quanto o aumento da qualidade, produtividade e satisfação do usuário, implantando processos de software segundo modelos, tais como CMMI-DEV [SEI 2006] e MR-MPS [SOFTEX 2009].

Considerando processo de software como um conjunto coerente de atividades, práticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, dispor e manter um produto de software [Fuggeta 2000], a *padronização de processos de software* pode reduzir a diversidade de processos existentes em uma organização, e ainda, promover o aprimoramento da comunicação e reduzir o tempo de treinamento dos envolvidos [Sommerville, 2007], com a utilização de conceitos de reutilização de componentes de processo de software.

Espera-se que, com a reutilização de processos de software, seja possível aumentar a qualidade e produtividade no desenvolvimento de software, minimizando a duplicação do esforço e reaproveitando o máximo possível das experiências de projetos passados.

Assim, os conceitos do desenvolvimento de software baseado em componentes, em que o desenvolvimento do software é realizado pela integração planejada de componentes de software pré-existentes [Brown e Short 1997] vêm sendo aplicados no contexto de processos de software, visando componentes de processos.

A definição de processos de software baseada em uma arquitetura de componentes de processo contribui para que as organizações possam promover a reutilização de processos de forma mais efetiva. Algumas iniciativas de pesquisa têm sido conduzidas no sentido de se estabelecerem métodos para promover a reutilização de processos [Lanna, 2009; Barreto *et al.,* 2008; Barreto *et al.*, 2009, Aleixo *et al.,* 2010]. Muitos desses trabalhos utilizam técnicas e processos de reutilização de software comuns para derivar métodos para reutilização de processos de software. No entanto, Fiorini (2001) propõe uma arquitetura de processos de software para facilitar o acesso e o reaproveitamento dos processos.

Nesse sentido, este trabalho propõe uma arquitetura de processos baseada em conceitos de Arquitetura de Software [Bass *et al*., 2003], definindo elementos que compõe uma arquitetura, tais como: Modelos de Referência; Estilos Arquiteturais; e Arquitetura de Referência.

A Arquitetura de Processos proposta foi baseada em modelos de referência como CMMI-DEV [SEI, 2006] e MR-MPS [SOFTEX, 2009], e possui um estilo arquitetural composto por camadas e uma *Arquitetura de Processos Padrão*, que pode ser estendida para *Arquiteturas de Processos em Uso*, contendo especificidades inerentes ao negócio ou ao tipo de software a ser desenvolvido.

Este artigo está organizado em seções. Na Seção 2 são apresentados os aspectos da definição do modelo de processo como: reutilização de processos de software; conceitos de arquitetura de processos; e o meta-modelo SPEM. Na Seção 3 apresenta-se a proposta do Modelo de Processo Padrão, com a Arquitetura de Processos e o Processo Padrão. Na Seção 4 apresenta-se um exemplo do processo em uso aplicado no desenvolvimento de jogos. Finalizando, na Seção 5, apresentam-se a conclusão e trabalhos futuros.

2. Definição do Modelo de Processo de Software

Esta seção apresenta os conceitos que foram utilizados para definição do processo padrão, quais sejam: reutilização de processos de software (Seção 2.1), arquitetura de software (Seção 2.2) e o meta-modelo SPEM (Seção 2.3).

2.1. Reutilização de Processo de Software

Succi *et al.* (1997) definem reutilização de processo como sendo a réplica de um conjunto de ações de um processo já executado em um novo contexto. Bayer (1999) afirma que é necessário que a definição do processo seja feita de forma flexível para que possa ser adaptado a diferentes situações, que possua diretivas e suporte suficientes, e que não seja muito vago, pois caso contrário, o mesmo não irá atender às necessidades de variadas situações da indústria sem uma forte interpretação e suporte adicionais.

Para Barreto *et al.* (2007), a definição de processo de software é uma tarefa desafiadora, pois além de exigir experiência ela envolve muitos aspectos da Engenharia de Software e é esperado que o conhecimento de profissionais mais experientes possa

ser compartilhado e reutilizado por outros profissionais. Com isso, faz-se necessário estabelecer mecanismos que sejam capazes de representar semelhanças e variabilidade de processos de software, facilitando a reutilização dos processos de software, o que contribuirá para diminuição de custo e esforço da atividade de definição de processos e aumento da qualidade dos processos [Barreto *et al.* 2009].

Segundo Fiorini (2001), o conceito de reutilização de processo é o uso de um modelo de processo, na criação de outro processo. O modelo de processo representa a descrição de um processo na linguagem de modelagem de processos.

Para a criação de processos reutilizáveis é necessário que a sua definição seja feita de forma que a organização possa adaptá-los aos requisitos dos projetos. Para tanto, Hollenbach e Frakes [apud Fiorini 2001] definiram o ciclo de vida para descrição de processo:

(i) Definir o processo reutilizável: é definir um processo propriamente dito e garantir que eles realmente possam ser reutilizados;

(ii) Desenvolver treinamento para o processo reutilizável: definir treinamento;

(iii) Adaptar o processo reutilizável ao projeto: adaptar o processo para atender as necessidades específicas do(s) projeto(s) que irá(ão) utilizá-lo;

(iv) Adaptar o treinamento do processo reutilizável para o processo adaptado ao projeto: adequar o treinamento para as necessidades do projeto que utilizará o processo;

(v) Executar o processo no projeto: significa que o projeto passa a utilizar na prática o processo. Acompanhamento de garantia de qualidade e medições devem ser realizadas;

(vi) Refinar o processo: o processo é avaliado com base nas medições e passos anteriores. Caso o processo não tenha alcançado os seus objetivos, ele deve ser ajustado e novos treinamentos devem ser realizados.

Arquitetura de processo é um conceito que também vem sendo utilizado no contexto de reutilização de processo de software. Para Barreto *et al.* (2007), de forma simplificada, "pode-se considerar que a arquitetura define um "esqueleto" que o processo deve possuir, determinando os principais elementos e como estes se relacionam, sem necessariamente definir como será o detalhamento desses elementos principais".

2.2. Arquitetura de Software

Segundo Larman (2002), uma Arquitetura de Software é um conjunto de decisões significativas sobre a organização do sistema de software, envolvendo a seleção dos elementos estruturais e suas interfaces, o comportamento e como esses elementos interagem e colaboram para alcançar tal comportamento. Tais características auxiliam a obtenção da reusabilidade. Bass *et al.* (2003) e Kruchten *et al.* (2006) definem Arquitetura de Software como sendo a estrutura, ou estruturas, de um sistema, que compreende elementos de software, as propriedades externamente visíveis de tais elementos e os relacionamentos entre os elementos.

Ainda, Bass *et al.* (2003) definem que uma arquitetura de referência consiste em componentes de software e os relacionamentos entre eles que implementam funcionalidades relativas às partes definidas no modelo de referência.

Uma arquitetura de referência serve de base para a elaboração de arquiteturas de software. Estas são definidas a partir de modelos de referência e estilos arquiteturais, que são aplicados a um domínio específico. Os modelos de referência, os estilos arquiteturais e as arquiteturas de referência não podem ser generalizados e confundidos com arquitetura de software, pois nenhum deles é arquitetura de software [Bass *et al.,* 2003], mas sim elementos úteis para a definição da arquitetura de um sistema, que indicam decisões importantes.

Um modelo de referência consiste na decomposição padronizada do problema em partes conhecidas que cooperam entre si em prol de uma solução. Geralmente, estes problemas são de domínio bastante amadurecido e trazem a experiência de analistas de negócio em conjunto com desenvolvedores [Bass *et al.* 2003]. Os estilos arquiteturais expressam esquemas de organização estrutural de sistemas, fornecendo um conjunto de componentes do sistema, suas responsabilidades e a forma de interação entre eles, estabelecendo um padrão de utilização. Por sua vez, uma arquitetura de referência consiste em componentes de software e os relacionamentos entre eles que implementam funcionalidades relativas às partes definidas no modelo de referência. Na Figura 1 apresenta-se o relacionamento entre esses conceitos:

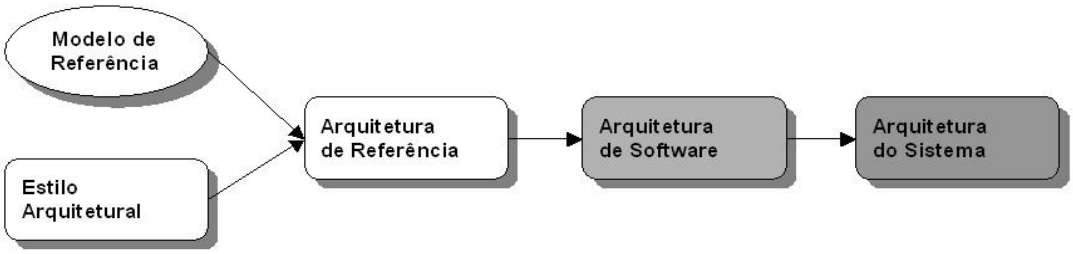

Figura 1. Relacionamento entre modelo de referência, estilo de arquitetura e arquitetura de referência implementando uma arquitetura de software (adaptado de BASS *et al.,* 2003)

*2.3.*O meta-modelo SPEM

O meta-modelo *Software & Systems Process Engineering Meta-Model Specification* (SPEM) [OMG 2008], define os conceitos necessários para modelar, documentar, apresentar, gerenciar e representar métodos e processos de desenvolvimento.

O SPEM é um meta-modelo de processo mantido pelo consórcio W3C [OMG 2008]. Ele permite definir processos de desenvolvimento de software e sistemas e seus componentes. O foco do SPEM são projetos de desenvolvimento, sendo que o objetivo principal é acomodar o maior número possível de métodos e processos de desenvolvimento de diferentes estilos, níveis de formalismo e modelos de ciclo de vida. De maneira geral, é possível representar com o SPEM as atividades e tarefas, papéis e responsabilidades, artefatos produzidos e consumidos e o conhecimento gerado a partir de tais elementos ou a eles relacionados. Este meta-modelo permite que processos de software sejam representados de duas formas:

- **Conteúdo de Método (*Method Content*):** descreve a estrutura dos processos, as tarefas que o compõem, os artefatos produzidos e consumidos, o conhecimento armazenado por meio de guias, conceitos, idéias e lições aprendidas. Todo o conteúdo é definido de maneira independente do ciclo de vida de desenvolvimento.

- **Processos (*Processes*):** a partir da reutilização do conteúdo, são definidos os fluxos de atividades e tarefas e as fases do ciclo de vida de um processo de desenvolvimento, bem como os objetivos de cada etapa do processo.

Outro aspecto importante do SPEM é o conceito de Variabilidade. O meta-modelo permite derivação de elementos de processos a partir de outros elementos pré-existentes. Os tipos de variabilidade permitidos são apresentados no Quadro 1.

Quadro 1 - Tipos de variabilidade (OMG, 2008)

Tipo de variabilidade	Características
Estender	O elemento herda as características do elemento base sem alterar as características deste. A extensão fornece um mecanismo para o *plug-in* de métodos reutilizar elementos, utilizando uma espécie de herança, a partir de uma base de *plug-ins*. Os valores dos atributos e associações também são herdados do elemento base para o elemento de extensão. O resultado é que o elemento de extensão tem as mesmas propriedades que o elemento base, mas pode definir suas próprias alterações.
Substituir	Um elemento de substituição substitui as partes do elemento base. A substituição fornece um mecanismo para um elemento substituir o elemento base sem alterar diretamente qualquer uma das suas propriedades. A substituição aparece no site publicado, mas o elemento base não.
Contribuir	Fornece uma maneira para elementos contribuírem com as suas propriedades ao seu elemento base sem diretamente alterar quaisquer de suas propriedades já existentes, de uma forma aditiva. A base aparece no site publicado, mas o elemento que contribui não.
Estender e Substituir	Esta relação combina os efeitos de variabilidade de extensão e substituição. Tal tipo de variabilidade substitui todos os atributos e instâncias do elemento base com novos valores e instâncias ou remove todos os valores ou associações se o elemento de substituição não tiver definido nenhum.

3.Proposta do Modelo de Processo Padrão

*3.1.*Arquitetura de Processos

Os trabalhos de Kammer (2000) e Oquendo (2006) são os que mais aproximam o conceito de Arquitetura de Software da idéia de Arquitetura de Processos, embora eles se refiram a processos distribuídos e cooperativos. Kammer (2000) utiliza o termo *Arquitetura de Processos* para descrever uma infraestrutura para execução de processos que relaciona os vários elementos do processo, como as pessoas, os artefatos, os processos automatizados e os recursos. Oquendo (2006) refere-se à *arquitetura de modelo de processo* como um conjunto de papéis e de componentes de processo. Os componentes definem as atividades que os membros de um projeto devem realizar e os papéis representam o que esses membros podem executar. Os papéis estão sujeitos a obrigações e habilidades necessárias [Borsoi, 2008].

Desse modo, uma Arquitetura de Processos pode ser conceituada como um conjunto de visões representadas por modelos de processos. Esses modelos representam

uma estrutura de processos e dos seus componentes em termos da sua composição, comportamento e relacionamentos, de acordo com um domínio e um contexto [Borsoi 2008]. O conjunto de visões determina os níveis de processos de software definidos, sendo que, desde o processo padrão até o nível de instanciação, os componentes são descritos em diferentes níveis de abstração. Sendo assim, em uma abordagem *top down*, quanto mais baixo o nível, mais específico será o seu conteúdo. Percebe-se então que a organização em níveis da Arquitetura de Processos permite o reuso de todo o conteúdo do processo padrão.

A *Arquitetura de Processos* proposta neste trabalho foi organizada conforme a visão de Bass *et al.* (2003) (Figura 1). As camadas da arquitetura de processo de software são apresentadas na Figura 2.

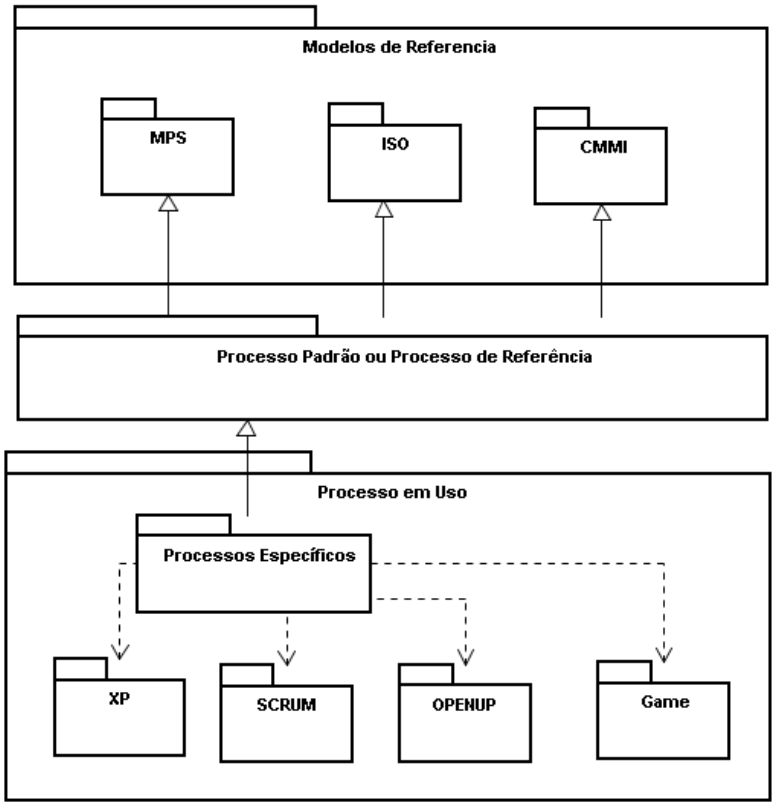

Figura 2. Arquitetura de Processos de Software

Os **modelos de referência** agregam solução aos problemas do ponto de vista do domínio e carregam um conjunto de conhecimento bastante amadurecido. Nesse sentido, para o domínio Desenvolvimento de Software – por exemplo, podem-se citar como modelos de referência o CMMI-DEV [SEI 2006] e o MR-MPS [SOFTEX 2009].

Normalmente, os modelos de referência são bastante genéricos e necessitam de interpretação para serem colocados em prática. Assim, o **Processo Padrão** ou **Processo de Referência** deve carregar os elementos mínimos de um processo, tais como: tarefas; entradas; saídas; papéis; responsabilidades; habilidades necessárias e medidas de qualidade de processo e de produto, em conformidade com os modelos de referência adotados.

As instâncias de processos definem a camada **Processo em Uso**. Essa é composta por processos específicos, que podem possuir dependência com outros processos ou práticas, tais como: SCRUM [Schwaber e Sutherland 2010] para gestão de projetos; OPENUP [Eclipse Foundation 2010]; e XP [Wells 2011]. Com isso, o estilo arquitetural em camadas (Figura 2) fornece um esquema de organização estrutural bastante favorável à reutilização, deixando claras as responsabilidades e a forma de interação entre cada camada, além de definir um padrão de utilização.

3.2. Processo Padrão

Para a elaboração do modelo de Processo Padrão foi seguido um ciclo de vida para definição de processos reutilizáveis [Hollenbach e Frakes *apud* Fiorini 2001], em que na fase "*Definir o processo reutilizável*" o Modelo de Processo Padrão (Figura 3) é baseado no meta-modelo SPEM [OMG 2008]. Os elementos com o estereótipo <<*spem*>> possuem relação direta com o metamodelo. Adicionalmente, os demais elementos de processo foram inseridos para estender a semântica de representação do Modelo do Processo padrão. É o caso, por exemplo, dos elementos de processo Medição e Ferramenta.

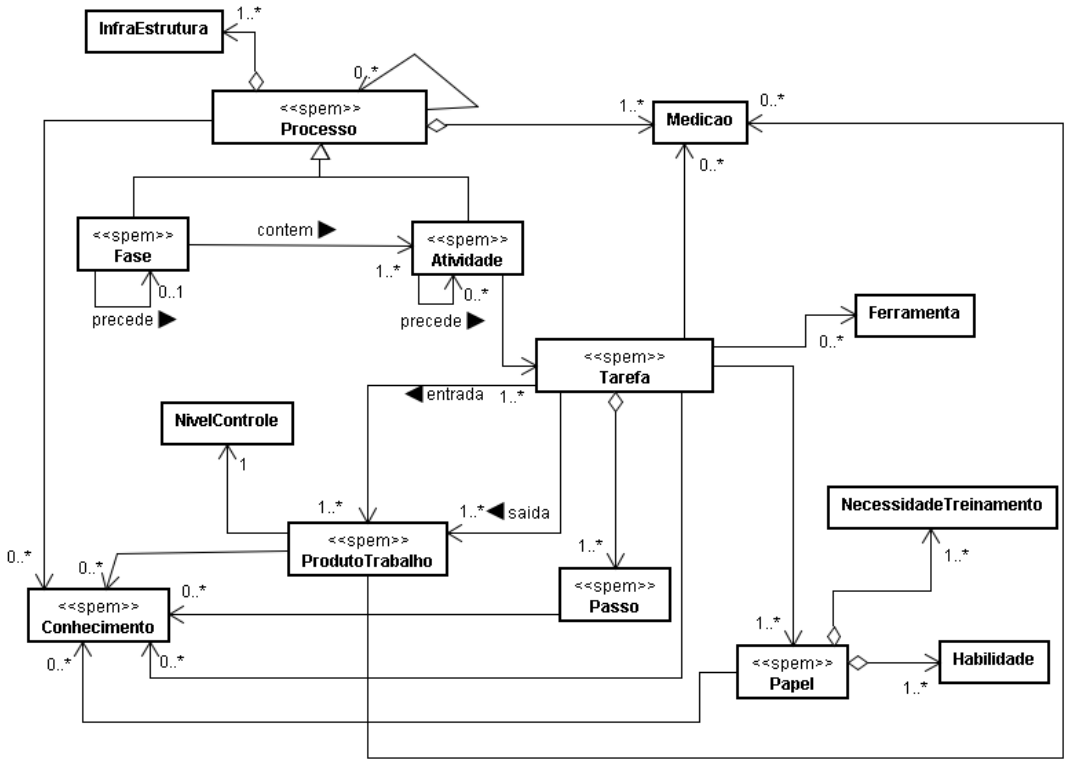

Figura 3. Modelo do Processo Padrão

No Quadro 2, apresenta-se uma breve descrição de alguns elementos do processo padrão.

Quadro 2 – Descrição dos elementos do Modelo do Processo Padrão

Elemento do Processo	Descrição
Infraestrutura	Os processos necessitam de infraestrutura e ambiente de trabalho adequados. Podem incluir questões ergonômicas, estações de trabalho, servidores e ambiente físico.
Medição	Todo processo, tarefa ou produto de trabalho, pode ter medições associadas. Uma medição pode ser útil para analisar o desempenho do processo em uma organização e subsidiar um plano de melhoria que contemple o processo em questão. Também será útil para medir a qualidade de um produto de trabalho de entrada ou resultante de um processo. Uma medição contém informações como necessidade de informação, descrição e fórmula.
Ferramenta	As tarefas dos processos de software podem ser apoiadas por ferramentas de software, que automatizam todo ou parte do processo.
Necessidade de Treinamento	Papéis são os responsáveis pelas tarefas dos processos. Tais tarefas podem exigir que os ocupantes de um papel realizem treinamentos para desempenhar adequadamente a função, desde que não se encaixem nos critérios de dispensa de treinamento.
Habilidade	Aos papéis são associadas habilidades necessárias para desempenhar a função. As habilidades podem ser de ordem técnica, gerencial, intrapessoal e interpessoal.
Nível de Controle	Os produtos de trabalho estão sujeitos a um determinado nível de controle ao longo de sua vida útil, com o objetivo de manter sua integridade. Níveis de controle incluem: padrões de versionamento de produtos, controle de mudanças, aprovações necessárias, controle de linha de base, entre outros.

A Figura 4 apresenta um exemplo de um processo padrão e seus elementos para Desenho de Software, em conformidade com a área de processo *Technical Solution* (TS) do modelo de referência CMMI-DEV [SEI 2006].

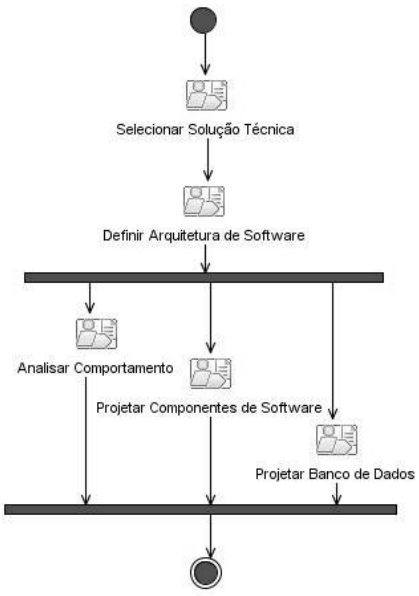

Figura 4. Processo Padrão Desenho de Software

Na Figura 5 apresenta-se o detalhamento da atividade "*Projetar Componentes de Software*".

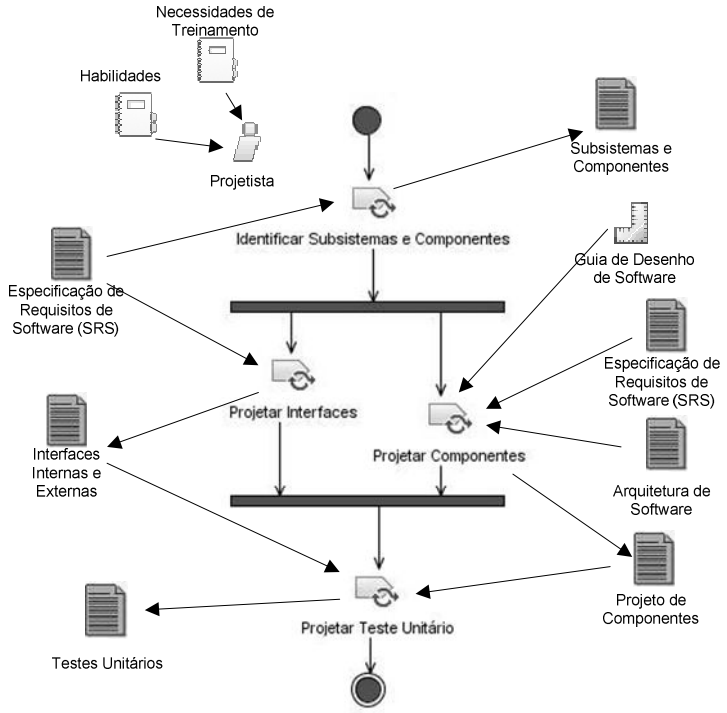

Figura 5. Atividade "Projetar Componentes de Software"

O diagrama da atividade "*Projetar Componentes de Software*" apresenta o fluxo de tarefas, os produtos de trabalho de entrada e saída de cada tarefa, um guia de projeto de software e o papel responsável pelas tarefas. A especificação de cada elemento deste modelo deve estar em conformidade com as práticas da Área de Processo TS do CMMI-DEV [SEI 2006], mas não indica modelos de documentos ou artefatos, uma vez que o processo mais tarde pode ser instanciado com uso de técnicas variadas, como por exemplo, Projeto Orientado a Objetos ou Diagramas de Fluxo de Dados. No Quadro 3 apresenta-se a especificação da *Tarefa Projetar Componentes*.

Quadro 3 – Detalhamento da tarefa "Projetar Componentes"

Atributos	Descrição
Propósito	Especificar os elementos de projeto (design), fazendo uso de métodos que englobam técnicas e ferramentas, a partir dos requisitos detalhados do software.
Descrição	A tarefa consiste em: analisar o comportamento dos subsistemas / componentes identificados por meio dos requisitos de software, identificar responsabilidades, atributos e associações entre as entidades dos componentes / subsistemas e detalhar a interação entre as entidades do componente / subsistema. Gerar o modelo estático e modelo dinâmico do software. O principal produto de trabalho gerado na atividade é o Projeto de Componentes, um modelo que descreve a realização dos requisitos funcionais e serve como uma abstração do modelo de implementação e seu código-fonte. O projeto de componentes é usado como base para atividades de implementação e teste. O projeto de componentes deve conter, no mínimo: a) Modelagem dinâmica (sequência de operações) b) -Modelagem estática (dados e estados) c) -Relacionamentos entre os elementos de projeto

Quadro 3 – Detalhamento da tarefa "Projetar Componentes" (continuação)

Atributos	Descrição
Passos	1. Analisar funcionalidades do software 2. Identificar entidades e responsabilidades 3. Projetar modelo estático 4. Projetar modelo dinâmico 5. Validar modelos
Papéis	Projetista
Entradas	Especificação de Requisitos de Software, Arquitetura de Software
Saídas	Subsistemas e componentes, Interfaces internas e externas, projeto (design) de componentes, testes unitários.
Conhecimento	a) Guia de Projeto (Design). b) Material de estudo sobre Análise e Projeto Orientado a Objetos com uso de UML.
Habilidades	a) Técnicas de modelagem estática e dinâmica. Ex: UML, DFD, SADT, IDEF0. b) Conhecimentos em requisitos de sistema e software c) Tecnologias e linguagens de programação. Ex: (JEE, dotNET, C, C++, COBOL, NATURAL, SGBD, Webservices, XML, Web) d) Técnicas de componentização de sistemas.
Treinamentos necessários	a) Análise e Projeto de Software com UML. b) Componentização de software.

Conforme classifica Fiorini (2001), uma instância de processo é um processo usual, não sendo um padrão normativo e tampouco um *pattern*, pois não precisa necessariamente ter sido testado um considerável número de vezes para garantir a solução de um problema recorrente.

Na arquitetura proposta, uma instância do processo é resultado da especialização do processo padrão e incluem detalhes suficientes para execução em um ambiente real. O processo padrão pode ser instanciado para cada projeto, sendo que novos elementos específicos de processo podem ser incluídos quando necessário. Um processo específico pode possuir dependência com outras instâncias de processos, tais como SCRUM [Schwaber e Sutherland, 2010], XP Wells 2011], OPENUP [Eclipse Foundation 2010] ou algum outro específico da organização. A próxima seção apresenta um exemplo de reutilização do modelo de processo padrão.

4.Exemplo de Processos em Uso – *Game Process Development*

Para a definição do processo em uso, além dos treinamentos no processo padrão e sua arquitetura, também foram realizados treinamentos na ferramenta EPF e os mecanismos disponíveis para facilitar a reutilização de processo (tipos de variabilidade). Os treinamentos nesta etapa referem-se à fase "Desenvolver treinamento para o processo reutilizável" do ciclo de vida de Hollenbach e Frakes.

A fase seguinte "Adaptar o processo reutilizável ao projeto" ocorreu com a implementação do Processo em Uso. Para implementar esse componente da Arquitetura de Processos, foi utilizada a ferramenta *Eclipse Process Framework*[1] (EPF). O EPF é uma ferramenta livre, desenvolvida com a mesma aparência do Eclipse, mas com a função de criação, gerenciamento da biblioteca de componentes, configuração,

[1] Ferramenta disponível em http://www eclipse.org/epf/downloads/tool/tool_downloads.php

manutenção e publicação de processos e métodos e aderente ao meta-modelo SPEM. Trabalha com uma terminologia comum, o *Unified Method Architecture* (UMA), que permite que métodos ou práticas sejam reaproveitados em processos diferentes e possui ainda mecanismos de extensão para adaptar práticas em contextos variados, o que possibilita uma maior flexibilização dos modelos disponíveis. O *framework* conta com vários exemplos de processos prontos que podem ser adaptados, tais como o *OPENUP*, que é uma versão básica do *Rational Unifield Process* (RUP), o *eXtreme Programming* (XP) e o *Scrum*, que descrevem práticas de desenvolvimento ágil. Ainda possui ferramentas que dão suporte a vários tipos de formas de desenvolvimento.

O desenvolvimento de um jogo eletrônico é uma tarefa de grande desafio e dificuldade, assim como é a de desenvolvimento de um software. Desenvolver jogos pode ser encarado, a principio, como desenvolver softwares [Bethke, 2003; Gershenfeld *et al.*, 2003]. A cada avanço no mercado de jogos, a complexidade em criá-los aumenta de maneira significativa devido a novas tecnologias e à necessidade de equipes maiores.

Devido ao fato da indústria de jogos envolver atividades multidisciplinares, profissionais com perfis totalmente diferentes (artistas, programadores e produtores) somam criatividade ao ambiente. Porém, uma verdadeira cisão pode ocorrer na equipe, sendo ela dividida entre "programadores" e "artistas" [Callele *et al,* 2005), sem que os métodos clássicos de desenvolvimento de software prevejam como lidar com essa situação de desencontro entre os grupos.

Com o aumento de complexidade, alguns fatores trazem à tona a necessidade da criação de um processo específico para a área de desenvolvimento de jogos [Callele *et al.,* 2005], como a multidisciplinaridade dos profissionais presentes no desenvolvimento interagindo com um processo tradicional de software; problemas como a definição de escopos irreais com acréscimo ou retirada de funcionalidades ao longo dos projetos; os problemas recorrentes na indústria de jogos [Petrillo 2007] e as dificuldades no levantamento de requisitos.No entanto, o mais comum é a falta de um processo, também chamado de "*code-and-fix*". Essa técnica envolve pouco ou nenhum planejamento e os problemas são resolvidos à medida que eles aparecem. A ausência de um processo contribui para a baixa qualidade e a não confiança no jogo final.

Percebe-se que os problemas recorrentes de qualidade pela falta de um processo definido e institucionalizado são potencializados no caso de desenvolvimento de jogos eletrônicos. A multidisciplinaridade envolvida, embora proporcione grande criatividade para elaboração do jogo, incorre no esquecimento de boas práticas básicas em desenvolvimento de software.

Nesse sentido, o modelo de processo padrão pode fornecer uma base de conhecimento importante para os desenvolvedores de jogos, permitindo a criação do próprio processo de desenvolvimento de jogos a partir de um processo convencional. Obviamente, o conhecimento embutido no processo padrão será reutilizado, pois o jogo eletrônico é um software, porém com algumas particularidades.

A Figura 6 apresenta um exemplo de reutilização da atividade "Projetar Componente de Software" (Figura 5) e também exemplifica a implementação da 3ª. etapa ("Adaptar o processo reutilizável ao projeto") do ciclo de vida de Hollenbach e Frakes.

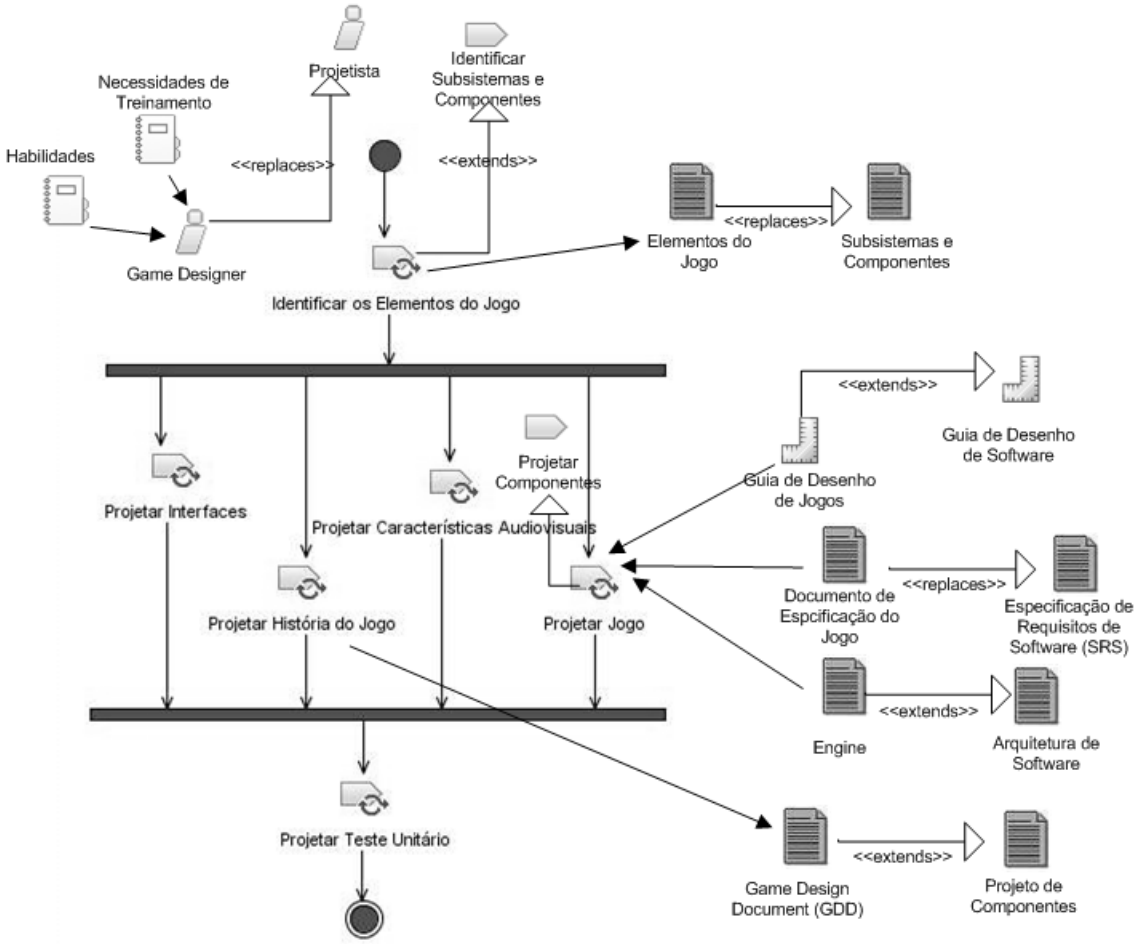

Figura 6. Atividade "Projetar Componentes de Software"

A Figura 6 apresenta algumas tarefas, produtos e papéis que foram estendidos ou substituídos a partir do Processo Padrão. A tarefa *Identificar Elementos do Jogo*, por exemplo, estende a tarefa Identificar Subsistemas e Componentes. Por sua vez, o produto *Elementos do Jogo* substitui o produto padrão *Subsistemas e Componentes*.

A tarefa *Projetar Componentes do Processo Padrão* é estendida pela tarefa *Projetar Jogo*, no processo de desenvolvimento de jogos. Isso ocorre, pois além do projeto de software comum definido no Processo Padrão, o projeto de jogos deve incluir as especificações das características do jogo, os elementos do jogo, o fluxo do jogo e a história do jogo. O principal produto de trabalho gerado pela tarefa passou a ser o *Game Designer Document* (GDD), que inclui o conteúdo do projeto de componentes de software do processo padrão, além das particularidades do projeto de um jogo eletrônico.

Encontram-se em planejamento as fases "*Adaptar o treinamento do processo reutilizável para o processo adaptado ao projeto*" e "*Executar o processo no projeto*". O processo será executado por estudantes da Universidade de Brasília que trabalham em projetos de desenvolvimento de jogos eletrônicos, por meio de um estudo de caso. Após essa fase, será realizado o refinamento do processo de acordo com as necessidades identificadas.

5. Conclusões

Neste artigo foi apresentada uma proposta de modelo de processo padrão baseada em uma arquitetura de componentes de processo, adotando o ciclo de vida para definição de processos reutilizáveis.

A partir do *processo padrão* foi gerado um *processo em uso* para o contexto de desenvolvimento de jogos eletrônicos. Foram utilizados os conceitos de reutilização e os tipos de variabilidade de processo por meio das funcionalidades da ferramenta *Eclipse Process Framework,* que implementa os conceitos do meta-modelo SPEM (*Software and Systems Process Engineering Meta-Model Specification*).

A reutilização do processo de software padrão para a definição do processo de jogos forneceu uma base de conhecimento relevante para os desenvolvedores de jogos, uma vez que, apesar das suas particularidades no processo de desenvolvimento, desenvolver jogos eletrônicos é desenvolver software.

A proposta do *Processo em Uso* para desenvolvimento de jogos encontra-se em fase de validação e planejamento para uso em projetos reais de desenvolvimento de jogos. Encontra-se também em andamento a definição de um novo *Processo em Uso* abordando metodologias ágeis de desenvolvimento de software e o modelo de referência MR-MPS [SOFTEX 2009].

Referências

Aleixo, F. A.; Freire, M. A.; Santos, W. C.; Kulesza, U. (2010) "An Approach to Manage and Customize Variability in Software Processes". In: Brazilian Symposium on Software Engineering, p. 118-127.

Barreto, A., Murta, L., Rocha, A.R. (2008) "Software Process Definition: a Reuse-based Approach". In: XXXIV Conferencia Latinoamericana de Informática (CLEI'08), Santa Fe, Argentina.

Barreto, A.; Murta, L.; Rocha, A. R. (2007) “Uma abordagem de definição de processo de software baseada em reutilização”. In: Workshop Anual de Melhoria de Processo de Software.

Barreto, A.; Murta, L.; Rocha, A. R. (2009) “Componentizando processos legados de software visando a reutilização de processos”. In: Simpósio Brasileiro de Qualidade de Software. [S.l.: s.n.].

Bass, L., Clements, P., Kazman, R. (2003) “Software Architecture in Practice”. Second Edition. Addison-Wesley.

Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J. (1999) “PuLSE: “A methodology to develop software product lines”. In: Symposium on Software Reusability (SSR). Los Angeles, USA: ACM Press, 199. p. 122-131.

Bethke, E. (2003) “Game Development and Production”. Plano: Wordware Publishing.

Borsoi, B. T. (2008) “Arquitetura de processo aplicada na integração de fábricas de software”, p. 37-61. Tese de Doutorado em Engenharia Elétrica. Universidade de São Paulo, USP, Brasil.

Brown, A. W.; Short, K. (1997) "On components and objects: The foundations of component-based development". In: International Symposium on Assessment of Software Tools, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 0112.

Callele, D., Neufeld, E., Sschneider, K. (2005) "Requirements engineering and the creative process in the video game industry". In: 13th IEEE International Conference on Requirements Engineering. Proceedings. Washington, DC, USA: IEEE Computer Society, p. 240–252.

Eclipse Foundation. (2010) "OPENUP". http://epf.eclipse.org/wikis/openup/, Março.

Fiorini, S. T. (2001) "Arquitetura para reutilização de processos de software". Tese (Doutorado) - PUC-RJ, Rio de Janeiro.

Fuggetta, A. (2000) "Software process: a roadmap". In: ICSE '00: Proceedings of the Conference on The Future of Software Engineering. New York, NY, USA: ACM, p. 25-34.

Gershenfeld, A., Loparco, M., Barajas, C. (2003) "Game Plan: the insider's guide to breaking in and succeeding in the computer and vieo game business". New York: St. Martin' s Griffin Press.

Hollenbach, C., Frakes, W. (1996) "Software Process Reuse in an Industrial Setting". In: Fourth International Conference on Software Reuse, Orlando, Florida, IEEE Computer Society Press, Los Alamitos, CA, p. 22-30, 1996.

Kammer, P. J. (2000) "Supporting dynamic distributed work processes with a component and event based approach". In: 22nd International Conference on Software Engineering (IEEE-CS), ACM Press, p. 710-712.

Kruchten, P., Obbink, H., Stafford, J. (2006) "The past, present, and future of software architecture". IEEE Software, v. 23, n. 2, p. 22-30.

Lanna, A. L. P. M.; Pietrobon, C. A. M. (2010) "Reuso de Processos de Software Baseado na Componentização de Processos e Conhecimento". In: Concurso de Teses e Dissertação em Qualidade de Software, 2010, Belém. Simpósio Brasileiro de Qualidade de Software.

Larman, C. (2002) "Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design and the Unified Process". USA.

OMG. (2008) "Software & Systems Process Engineering Meta-Model Specification". [S.l.], Abril.

Oquendo, F. (2006) "Formally modelling software architectures with the UML 2.0 profile for π-ADL". ACM SIGSOFT Software Engineering Notes, v. 31, n. 1, p. 1-13, jan.

Petrillo, F. S. (2009) "Práticas Agéis no Processo de Desenvolvimento de Jogos Eletrônicos". Dissertação (mestrado). UFRGS, Rio Grande do Sul.

Schwaber, K., Sutherland, J. (2010) "Scrum Guide". http://www.scrum.org/scrumguides/, Março.

SEI, Software Engineering Institute, (2006) "CMMI for Development". http://www.sei.cmu.edu/library/abstracts/reports/06tr008.cfm, Março.

SOFTEX, Associação para Promoção da Excelência do Software Brasileiro. (2009) "MPS.BR - Guia Geral". http://www.softex.br, Março.

Sommerville, I. (2007) "Engenharia de Software". 8. ed. São Paulo: Pearson Addison-Wesley.

Succi, G., Benedicenti, L., Predonzani, P., Vernazza, T. (1997) "Standardizing the Reuse of Software Process". In StandardView - Special issue on reuse standards and software. ACM. Volume 5, Issue 2, p. 74-83.

Wells, D. (2011) "Extreme Programming: A gentle introduction". http://www.extremeprogramming.org/, Março.