UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Alexsander Domingues Rodrigues

SIMULAÇÃO DE PROCESSOS BASEADOS NO SPEM ATRAVÉS DO JOGO EDUCATIVO SPARSE

Alfenas, 16 de junho de 2011.

UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SIMULAÇÃO DE PROCESSOS BASEADOS NO SPEM ATRAVÉS DO JOGO EDUCATIVO SPARSE

Alexsander Domingues Rodrigues

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Alfenas como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Mariane Moreira de Souza

Alfenas, 16 de junho de 2011.

Alexsander Domingues Rodrigues

SIMULAÇÃO DE PROCESSOS BASEADOS NO SPEM ATRAVÉS DO JOGO EDUCATIVO SPARSE

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Prof. Rodrigo Martins Pagliares

Universidade Federal de Alfenas

Prof. Flávio Barbieri Gonzaga

Universidade Federal de Alfenas

Profa. Mariane Moreira de Souza (Orientador)

Universidade Federal de Alfenas

Alfenas, 16 de junho de 2011.

AGRADECIMENTO

Agradeço a minha família, pelo apoio e suporte durante a graduação, por estarem sempre presentes nos momentos em que mais precisei.

Agradeço a professora Mariane Moreira de Souza, por possibilitar o aprendizado de novos conceitos e pela orientação durante o trabalho.

Agradeço aos amigos, Flávio, Lucas, Luiz Henrique, Neubio pela ajuda durante a graduação, com trabalhos e com os estudos.

Agradeço ao amigo Jan Victor, pelo auxilio com a adaptação da interface 3D para realização da avaliação do SPARSE.

RESUMO

Um dos problemas existentes na área de Engenharia de Software é o contraste entre os métodos de ensino, predominantemente teóricos, e a natureza da área, essencialmente prática, limitando assim a exposição dos alunos às diversas situações típicas de ambientes de desenvolvimento de software que ele pode vir a enfrentar no mercado de trabalho. Na tentativa de resolver tal problema, tem-se utilizado recursos como jogos e simulação que, aliados às aulas teóricas, visam proporcionar ao aluno uma experiência prática através da simulação de diversos cenários típicos de ambientes reais, aumentando a motivação do estudante no aprendizado da área. O SPARSE é um jogo educativo baseado em simulação com foco no ensino de modelos de processo e boas práticas da Engenharia de Software. Em sua primeira versão o SPARSE apresentava uma limitação de suporte à simulação exclusiva do modelo cascata, que não reflete a realidade de mercado, havendo a necessidade de criação de novas versões para novos modelos. Visando resolver este problema, foi proposta uma nova estrutura para o SPARSE para possibilitar a simulação de diversos modelos de processo, criados utilizando o meta-modelo SPEM. Este trabalho desenvolve um novo simulador baseado na nova estrutura do SPARSE. Após a criação do simulador, o mesmo foi verificado através do uso do jogo SPARSE por alunos de graduação, cursando a disciplina de Engenharia de Software.

Palavras-Chave: Engenharia de Software, Simulação, Jogos Educativos, SPARSE, SPEM.

ABSTRACT

One of the problems in the area of Software Engineering is the contrast between the teaching methods, mainly theoretical, and the nature of the area, essentially practical, thus limiting students' exposure to various situations typical of software development environments that it can come to face in the labor market. In an attempt to solve this problem, it have been used approaches like games and simulation that, combined with lectures, designed to provide students with practical experience through simulation of several typical scenarios in real environments, increasing student motivation in learning area . SPARSE is an educational game-based simulation with a focus on teaching process models and best practices of Software Engineering. In its first version SPARSE had a limitation related to the simulation exclusively of the waterfall model, which does not reflect market reality, there is a need to create new versions for new models. Aiming to solve this problem, it is proposed a new structure for SPARSE to enable simulation of different process models, created using the SPEM meta-model. This work develops a new simulator based on the new structure of SPARSE. After the creation of the simulator, the same was observed by using the game SPARSE by undergraduate students studying the discipline of Software Engineering.

Keywords: Software Engineering, Simulation, Educational Games, SPARSE, SPEM.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

EPF Composer	*Eclipse Process Framework Composer*
OMG	*Object Management Group*
OpenUP	*Open Unified Process*
SGML	*Standard Generalized Markup Language*
SPEM	*Software Process Engeneering Metamodel*
SPARSE	*Software Process semi-automated tool for Software Engineering*
XML	*eXtensible Markup Language*
W3C	*World Wide Web Consortium*
LP&D	*Laboratório de Pesquisa e Desenvolvimento*

SUMÁRIO

1
Introdução

Este capítulo apresenta a razão pela qual este trabalho foi desenvolvido, o problema que o mesmo procurou resolver e os objetivos definidos para que o mesmo pudesse ser concluído. Na seção 1.1 são apresentadas as justificativas e motivações para o desenvolvimento do trabalho. A seção 1.2 detalha o problema envolvido no desenvolvimento deste trabalho. Finalmente, na seção 1.3 são detalhados os objetivos do mesmo.

1.1 Justificativa e Motivação

Segundo Souza (2010) um problema crítico constantemente abordado é o contraste entre as práticas de ensino e a natureza, essencialmente prática da área de Engenharia de Software. Atualmente, na maioria dos cursos, a disciplina de Engenharia de Software consiste de aulas teóricas e, como tentativa de colocar em prática conceitos vistos em sala de aula, os alunos desenvolvem pequenos projetos fictícios, limitando assim a exposição a situações que encontrarão no mercado (Pfahl et al. 2004).

Segundo Pfahl et al (2004) isso pode ser resolvido através do uso de simulação, que pode reproduzir a realidade do desenvolvimento de software. Neste contexto foi criado o SPARSE (*Software Project semi-Automated Reasoning tool for Software Engineering*), um ambiente baseado em jogos e simulação com o objetivo de proporcionar aos alunos uma visão prática em relação aos conceitos vistos em aulas teóricas. De acordo com Souza (2010) o jogo SPARSE foi desenvolvido com a proposta de auxiliar na melhoria do ensino de Engenharia de Software, aproximando os alunos de condições reais e melhorando a qualidade de formação dos mesmos nesta área.

A fim de melhorar a qualidade do aprendizado proporcionado pelo jogo, Rodrigues (2010) propõe adaptações na então existente estrutura do SPARSE, permitindo ao mesmo simular qualquer processo que seja modelado de acordo

com o meta-modelo SPEM. Este trabalho tem como finalidade possibilitar a simulação de processos no jogo SPARSE, modelados seguindo esta nova abordagem. Para verificação do simulador construído foi realizado um estudo de caso com a simulação do modelo de processo utilizado na fábrica de software LP&D. Em seguida, foi analisado ainda o uso do jogo SPARSE por alunos de graduação, cursando a disciplina de Engenharia de Software.

1.2 Problematização

Como já dito, a disciplina de Engenharia de Software apresenta um problema crítico que consiste na divergência entre métodos teóricos de ensino e a natureza prática da área. Na tentativa de amenizar tal problema, o uso de jogos e simulação tem sido empregado, sendo o SPARSE um dos trabalhos desenvolvidos neste sentido. Porém, em sua primeira versão, o SPARSE possui a limitação de suportar apenas o modelo de ciclo de vida cascata, de modo que a estrutura do modelo se encontrava acoplada ao modelo simulação. Visando resolver este problema, o trabalho de Rodrigues (2010) propõe uma nova estrutura, para incorporar o meta-modelo SPEM ao SPARSE, possibilitando a simulação de qualquer processo baseado no meta-modelo.

Neste contexto, torna-se necessária a criação de um simulador que possibilite a simulação de qualquer processo construído com base no meta-modelo SPEM no jogo educativo SPARSE.

Sabendo disso, deseja-se averiguar:

Como simular, no SPARSE, um dado modelo de processo, construído com base no meta-modelo SPEM?

1.3 Objetivos

1.3.1 Gerais

A partir da integração do meta-modelo SPEM no SPARSE definido no trabalho de Rodrigues (2010) faz-se necessária a construção de uma nova versão do simulador do jogo SPARSE visando a adaptação do mesmo para contemplar esta nova abordagem. Logo, o objetivo geral deste trabalho é possibilitar que o SPARSE seja capaz de simular um dado processo, desde que o mesmo seja construído com base no meta-modelo SPEM.

1.3.2 Específicos

Para que o objetivo geral fosse atingido, alguns objetivos específicos foram determinados:

- Adaptação das classes de implementação do meta-modelo SPEM para possibilitar a execução de diferentes processos;
- Construção do arquivo XML para representação do processo a ser simulado;
- Criação da classe de leitura do arquivo XML que representa o processo a ser simulado;
- Criação de uma nova interface tabular, para contemplação de novos requisitos de simulação;
- Revisão/adaptação das regras relacionadas às boas práticas da Engenharia de Software, já implementadas no SPARSE;
- Simulação do processo da fábrica de software LP&D;
- Realização de testes para calibragem de variáveis e constantes de simulação;
- Realização de verificação da execução do SPARSE, através do seu uso por alunos de graduação.

1.4 Organização da Monografia

Este trabalho encontra-se organizado da seguinte forma: No Capítulo 2 apresenta-se todo o referencial teórico necessário para o entendimento deste trabalho; No Capítulo 3 é apresentada o simulador, bem como todos os passos utilizados para o desenvolvimento do mesmo; No Capítulo 4 é apresentado o estudo de caso, realizado para a avaliação do trabalho. No Capítulo **Erro! Fonte de referência não encontrada.** são apresentado os resultados da avaliação do mesmo; Finalmente no Capítulo 5 são apresentadas as considerações finais da monografia.

2
Revisão Bibliográfica

Este capítulo apresenta os conceitos necessários para um bom entendimento deste trabalho. A seção 2.1 apresenta os conceitos básicos da área de Engenharia de Software. As seções seguintes apresentam o conceito de simulação de processos, bem como os diferentes tipos de simulação, o meta-modelo SPEM, o modelo de processo OpenUP, no qual o processo utilizado no estudo de caso é baseado, a linguagem XML e uma explicação sobre o funcionamento do jogo educativo SPARSE.

2.1 Engenharia de Software

Para a construção de um produto de software, uma sequência de etapas e tarefas deve ser realizada. A ordenação de tais etapas e tarefas é conhecida como um processo. De acordo com Sommerville (2007, pag.5) um processo é um conjunto de atividades cujo objetivo é o desenvolvimento ou a evolução de software. Segundo Paula (2009, pag. 89) um processo é uma receita que é seguida por um projeto e o projeto caracteriza a abstração que é o processo.

Na literatura existem vários modelos de processo, e estes possuem algumas etapas comuns entre si. Segundo Sommerville (2007, pag. 43) elas são: especificação de software, projeto e implementação de software, validação de software e evolução de software.

Embora existam vários processos, não se pode afirmar que um dado processo é o ideal para qualquer tipo de projeto, sendo estes modificados a fim de atender as necessidades de cada tipo. Neste sentido, existem os modelos de processo ou modelos de ciclo de vida.

Segundo Sommerville (2007, pag.43) um modelo de processo de software (ou modelo de ciclo de vida) é uma representação abstrata de um processo de software. Cada modelo representa um processo sobre determinada perspectiva e, dessa forma, fornece somente informações parciais sobre esse processo.

Paula (2009) descreve os seguintes modelos de ciclo de vida:

- Ciclo de vida em Cascata: os principais subprocessos são executados em sequência. O ciclo de vida cascata é recomendado para projetos com pequena duração.
- Ciclo de vida em Espiral: o produto é desenvolvido em iterações e cada nova iteração corresponde a uma volta na espiral. Ao final de cada iteração é possível a liberação parcial do produto, que é submetido à avaliação dos usuários, chamadas de liberações ou *releases*.
- Outros modelos de ciclo de vida: modelos dirigidos por prazo, onde o produto é aquilo que se consegue fazer dentro de determinado prazo, indicado quando se consegue definir um conjunto de requisitos indispensáveis para os quais o prazo estabelecido é suficiente. Em modelos dirigidos por ferramenta, a ferramenta impõe processos rígidos que podem ser adequados a projetos específicos. A qualidade do processo depende da qualidade da ferramenta.

Embora existam vários modelos de processo propostos, é um consenso na comunidade que não existe um que se aplica a todas as situações de desenvolvimento (Rodrigues apud Bezerra 2007).

No caso deste trabalho, o processo a ser simulado é definido com base no modelo de ciclo de vida iterativo e incremental, que é uma classificação geral para modelos que desenvolvem o produto de software de maneira cíclica e evolutiva como o modelo espiral, prototipagem, etc.

2.2 Simulação de processos

Segundo Souza (2007) a importância da simulação de modelos está no fato de tais modelos poderem ser executados e seu comportamento simulado. De acordo com Souza (Souza apud Hoover e Perry 1989)

> Simulação é o processo de desenvolver um modelo matemático ou lógico de um sistema real e então conduzir experimentos baseados em computador, usando o modelo para descrever, explicar e predizer o comportamento de um sistema real.

Segundo Souza (Souza apud Madachy e Boehm 1999)

> A simulação torna-se uma ferramenta importante para avaliação de um modelo, quando este não possui uma solução analítica, ou seja, uma solução através de uma fórmula fechada. Além disso, o papel da simulação é fundamental em modelos, onde o efeito da interação entre os componentes do mesmo, e o modelo como um todo, estão separados no tempo e no espaço.

De acordo com a literatura (Barros 2006), a simulação baseada em computador se divide em dois tipos principais: simulação discreta e simulação contínua. As próximas subseções detalham cada um destes tipos de simulação.

2.2.1 Simulação Discreta

Segundo Barros (2001) em um modelo de eventos discretos as variáveis descrevem o estado do sistema. Seus valores mudam em intervalos de tempo discretos não determinados, de acordo com a ocorrência de eventos. A ocorrência de novos eventos renova o processo, fazendo uma atualização de variáveis e gerando novos eventos. A simulação prossegue até que não haja mais eventos ou um tempo limite seja atingido.

De acordo com Souza (2007), a simulação de eventos discretos possui aplicação em diversas áreas, como planejamento de processos e layouts de fábricas, sistemas de transporte, redes de telecomunicações, dentre outros.

2.2.2 Simulação Contínua

Segundo Barros (2001) um modelo de eventos contínuos ocorre em intervalos constantes e previamente determinados, de modo que em cada um destes intervalos as variáveis são alteradas. A simulação termina após um número de intervalos definidos.

De acordo com Souza (2007) a simulação contínua possui aplicação em diversas áreas, como resolução de problemas relacionados ao balanço térmico de reatores nucleares, e ao escoamento de líquidos; validação de sistemas estocásticos na área de pesquisa operacional, dentre outros.

2.2.3 Simulação Aplicada a Jogos

De forma geral, o uso de simulação visa dar suporte ao desenvolvimento da visão sistêmica, da prática de pensar estrategicamente, de compartilhar conhecimento em grupo. Nesse contexto a simulação pode ser explorada nos seguintes pontos: (1) como um processo de mapeamento cognitivo que captura o conhecimento e estimula a aprendizagem; (2) como um meio propício à experimentação e; (3) como uma forma de aprender a lidar com conflito de interesses (Belhot, 2001).

Segundo Souza (2007) o ensino baseado em simulação está diretamente relacionado à construção de modelos, sendo que um dos objetivos da simulação de modelos é capacitar e treinar pessoas, de forma a se tornarem mais aptas a realizar dadas funções.

Segundo Souza (Souza apud Catapan et al 1999) o uso de jogos baseados em simulação tem surtido efeito em diferentes áreas do conhecimento, principalmente com relação à motivação do aprendiz. No contexto deste trabalho isto é importante, uma vez que o excesso de teoria nos cursos da área de Engenharia de Software causa o desinteresse de muitos alunos.

O SPARSE foi criado com o objetivo de estimular a aprendizagem através da abordagem de jogos e simulação. Existem alguns trabalhos na literatura que motivaram o desenvolvimento deste jogo, dentre eles os ambientes SESAM e SimSE.

O SESAM (*Shell for Simulated Agent Systems*) (Drappa e Ludewig, 2000) é um ambiente pioneiro na área de simulação para verificação de processos através de instâncias dos mesmos em forma de projetos. O ambiente SESAM não é baseado em jogos como o SPARSE, sendo mais uma ambiente de simulação para validação de processos.

O SimSE (Navarro, 2006) é um jogo baseado em simulação para o ensino de Engenharia de Software, em que o jogador assume o papel de gerente de projetos e é responsável por desempenhar atividades como contratar e demitir desenvolvedores, alocar tarefas, dentre outras. Utiliza conceitos de gerência de projetos para o ensino de conceitos de Engenharia de Software. O SimSE é o que mais se aproxima da proposta do SPARSE, porém apresenta uma interface 2D bastante limitada, além de simular processos baseado em modelos próprios, criados em diferentes protótipos do jogo, e não no meta-modelo SPEM. Este meta-modelo é detalhado na próxima seção.

2.3 SPEM

O SPEM foi criado pelo *Object Management Group* (OMG), como um padrão de alto nível para processos utilizados no desenvolvimento de software orientado a objeto. Inicialmente foi criado como um meta-modelo independente, mas depois foi reformulado para ser um perfil UML. Isso significa que autores de processos podem transformar concepções orientadas a processo em concepções orientadas a modelo (Henderson-Sellers, 2005).

Apesar de o SPEM conter um número significativo de classes, no mais alto nível, há três meta-classes significantes: atividades, papéis e produtos de trabalho. Este é descrito como o modelo conceitual e é usado para dar uma compreensão geral, ao invés de criar bases para a herança de classes e a hierarquia das subclasses (Henderson-Sellers, 2005).

SPEM se refere ao processo de desenvolvimento de software como uma colaboração entre entidades ativas (chamadas papéis), destinadas a realizar operações específicas, chamadas atividades, sobre um conjunto de artefatos, chamados produtos de trabalho, até que os artefatos sejam levados a um estado

bem definido e declarado completo. O meta-modelo então, se refere à estrutura central de um processo de desenvolvimento de software como um conjunto de papéis, os produtos de trabalho que cada papel é responsável, e as atividades que realizam sobre os produtos de trabalho (Ransim, 2008). A Figura 1 ilustra a estrutura central de um processo, segundo o meta-modelo SPEM.

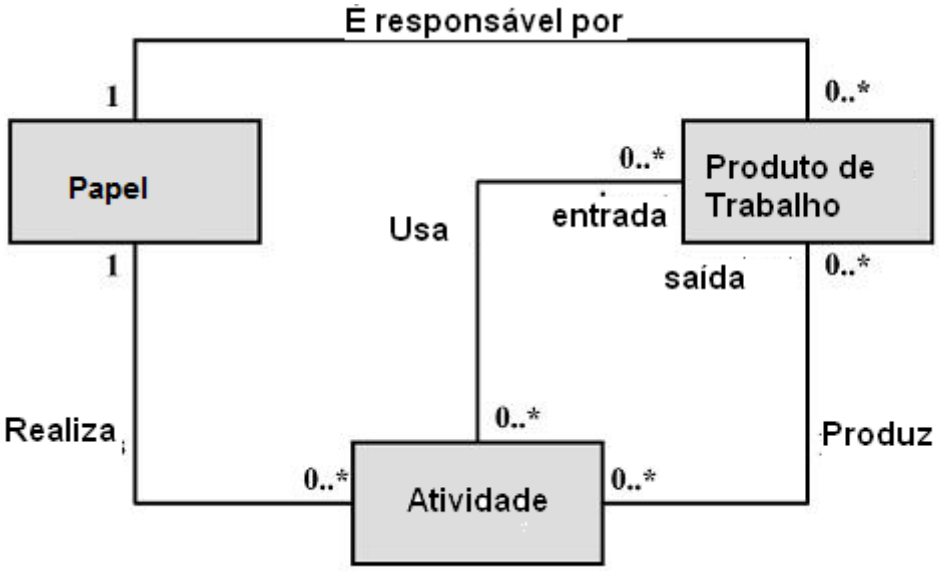

Figura 1- Estrutura Central de um Processo como definida pelo SPEM (adaptado de Ramsin 2008).

A estrutura completa de um processo no SPEM é na verdade muito mais complexa do que a citada acima. Um produto de trabalho pode ser composto de outros produtos de trabalho. As atividades podem ser divididas em disciplinas, com base nos temas estruturais e funcionais em comum, e cada atividade pode consistir de subatividades chamadas etapas, uma atividade pode ter uma condição e um objetivo como restrição (Ransim, 2008).

A fim de limitar a ordem na qual as atividades são realizadas, e para definir a estrutura do ciclo de vida do processo, o SPEM incorpora definições de fase, iteração e ciclo de vida. O processo de estrutura proposto pelo SPEM também inclui várias classes abstratas encapsulando as semelhanças estruturais e comportamentais dos diversos tipos de elementos do processo (Ransim, 2008).

No caso deste trabalho, o meta-modelo SPEM é utilizado como base para representação dos elementos do processo a ser simulado. A estrutura do processo a ser simulado é definida por meio de um arquivo XML. A linguagem XML é detalhada na próxima seção.

2.4 eXtensible Markup Language

XML é uma linguagem para especificação de dados, semi ou completamente estruturados. Tem sido explorada continuamente, tanto na área acadêmica quanto na industrial (Moro et al. 2009).

A XML descreve uma classe de objetos de dados chamados documentos XML e descreve parcialmente o comportamento de programas que os processam. É um perfil de aplicação ou uma forma restrita da SGML (*Standard Generalized Markup Language*) (ISO 8879). Desta forma documentos XML são por conformidade documentos SGML (W3C, 2008).

Documentos XML são compostos de unidades de armazenamento chamadas de entidades, que contém dados analisados ou não analisados. Dados analisados são compostos de caracteres. Alguns compõem dados e outros marcações. Marcações codificam uma descrição de *layout* do documento e sua estrutura lógica (W3C, 2008).

A linguagem foi desenvolvida pelo XML *Working Group* (originalmente conhecido como SGML *Editorial Review Board*), formado sobre a supervisão do W3C (*World Wide Web Consortium*) em 1996. Foi presidido por Jon Bosak da Sun Microsystems, com a participação ativa de um *Special Interest Group* XML (anteriormente conhecido como o Grupo de Trabalho SGML) também organizado pela W3C (W3C, 2008).

Desenvolveu-se a XML com a finalidade de atingir os seguintes objetivos (W3C 2008):

Ser usada diretamente na Internet;

Suportar uma grande variedade de aplicações;

Ser compatível com SGML;

Possibilitar a escrita de programas para processá-la facilmente;

Manter no mínimo possível o número de capacidades adicionais do XML, idealmente zero;

Criar documentos utilizando XML, de maneira inteligível para humanos.

Criar *design* em XML rapidamente;

O *design* da XML ser formal e conciso;

Documentos XML serem fáceis de ser criados;

Concisão em marcação XML ser de mínima importância.

Mais de dez anos após a sua proposta, a linguagem XML não conseguiu cumprir suas expectativas iniciais, mas conseguiu resolver eficientemente alguns problemas, que incluem: integração de dados, interoperabilidade de dados, e publicação de dados na WEB (Moro et al. 2009).

Neste trabalho, a linguagem XML é utilizada para a importação de dados do processo, para que o mesmo possa então ser simulado. A instância do processo a ser definido no arquivo XML e simulado no jogo é baseado no processo OpenUP. Este processo é detalhado na próxima seção.

2.5 OpenUP

O OpenUP é um processo unificado que aplica métodos iterativos e incrementais dentro de um ciclo de vida estruturado. O OpenUP adota uma filosofia pragmática e ágil que foca na natureza colaborativa do desenvolvimento de software, que pode ser estendido para lidar com uma ampla variedade de projetos (OpenUP, 2009).

O OpenUP é dividido em três camadas, como mostra a Figura 2. São elas: Foco pessoal (*Micro-Increment*), Foco da equipe (*Iteration Lifecycle*) e Foco de *Stakeholder* (*Project Lifecycle*).

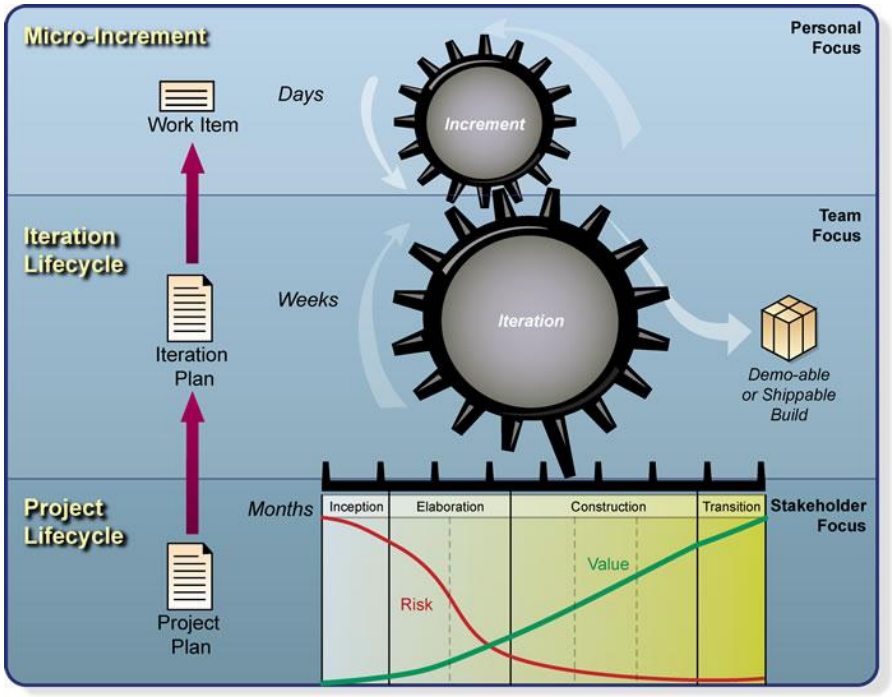

Figura 2 - Camadas do OpenUP (OpenUP 2009)

O esforço pessoal em um projeto do OpenUP é organizado em microincrementos, que representam pequenas unidades de trabalho que produzem um passo do progresso do projeto instável e mensurável (geralmente medido em horas ou alguns dias). O processo aplica intensiva colaboração à medida que o sistema é incrementalmente desenvolvido, por uma equipe comprometida e auto-organizada. Estes microincrementos fornecem um *feedback* extremamente curto, que direciona as decisões de adaptação a cada iteração (OpenUP, 2009).

O OpenUP divide o projeto em iterações, que são planejadas, com intervalos de tempo fixos normalmente medidos em semanas. A Iteração foca na equipe entregando incrementos aos *stakeholders*, definindo o que deve ser entregue, tendo como resultado uma construção demonstrável. A equipe se auto-organiza em como alcançar os objetivos da iteração, comprometendo-se a entregar os resultados, através da definição de tarefas a partir de uma lista de trabalho. O ciclo de vida de iterações estrutura como os microincrementos são aplicados para entregar versões do sistema que sejam estáveis e coesas, que evoluem progressivamente através dos objetivos da iteração (OpenUP, 2009).

O OpenUP estrutura o ciclo de vida do projeto em quatro fazes: Concepção (*Inception*) fase para se determinar o escopo do projeto e seus objetivos; Elaboração

(*Elaboration*) fase que trata dos riscos arquiteturais; Construção (*Construction*) fase que foca no design, implementação e testes para desenvolvimentos do sistema como um todo; Transição (*Transition*) fase que foca na entrega do software para usuários (OpenUP, 2009).

Cada fase é composta de uma ou mais iterações, nas quais versões do software em funcionamento são desenvolvidas e entregues. Cada iteração representa um *milestone* menor para o projeto que contribui para alcançar o *milestone* da fase. Segundo Paula (2009, pag. 116) um *milestone* é um evento significativo em um projeto, como uma decisão importante, ou a realização completa de uma entrega ou uma etapa.

O ciclo de vida do projeto fornece aos *stakeholders* e membros da equipe visibilidade e pontos de decisões durante o projeto. Isso possibilita uma efetiva supervisão, permitindo uma tomada apropriada de decisões em certos momentos. A estrutura de ciclo de vida do projeto pode ser vista na Figura 3.

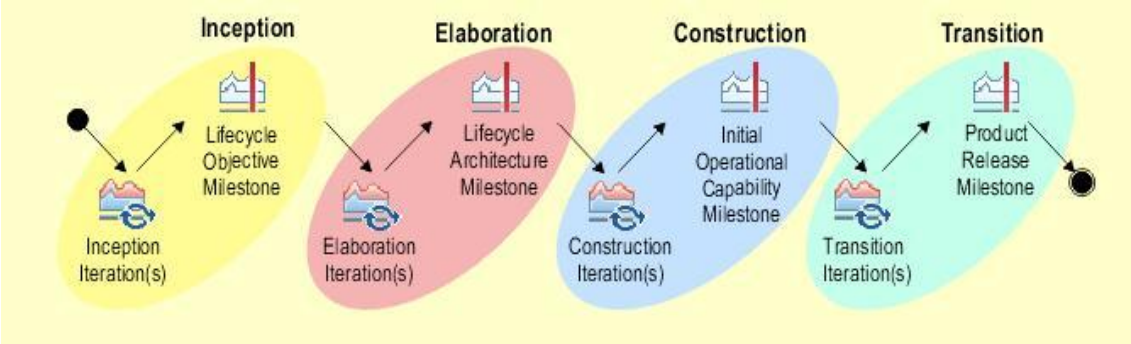

Figura 3 - Ciclo de vida de um projeto OpenUP (OpenUP 2009)

O OpenUP é um processo iterativo de desenvolvimento de software que é mínimo, completo e extensível. É regido por quatro princípios fundamentais:

- Colaborar para alinhar interesses e compartilhar conhecimento.
- Equilibrar prioridades concorrentes para maximizar o valor dos *stakeholders*.
- Concentrar-se na arquitetura no início para minimizar riscos e organizar o desenvolvimento.
- Evoluir para continuamente obter *feedback* e melhorar.

Pessoas em papéis executam tarefas que utilizam e produzem artefatos (produtos de trabalho). O OpenUP descreve o conjunto mínimo de papéis, tarefas (organizados por disciplinas) e artefatos (organizados por domínios de produto de trabalho) envolvidos no desenvolvimento de software.

O OpenUP representa a base de um processo customizado para uma fábrica de software real, sendo este o processo utilizado no estudo de caso no contexto deste trabalho.

2.6 SPARSE

O SPARSE (*Software Project semi-Automated Tool for Software Engineering*) é um jogo educativo que foi desenvolvido com o objetivo de auxiliar no ensino de Engenharia de Software, possibilitando aos alunos colocar em prática conceitos visto em sala de aula, tornando-os mais preparados para tomadas de decisões em cenários reais (Souza et al 2010).

O SPARSE utiliza um modelo de simulação discreta, baseado em regras, em que o sistema armazena um histórico dos fatos acontecidos, permitindo a observação do comportamento de cada variável (Souza apud Barros 2001).

Alguns dos elementos essenciais no desenvolvimento de qualquer projeto de software estão presentes no simulador, sendo eles:

- Modelo de processo - Um conjunto de atividades pré-definidas para atingir um determinado objetivo, o desenvolvimento ou evolução de um software (Sommerville, 2007). O modelo utilizado, baseado no SPEM, contém um conjunto de fases, onde cada fase possui de 1 a n iterações e cada iteração possui uma ou mais atividades, como mostra a Figura 4.

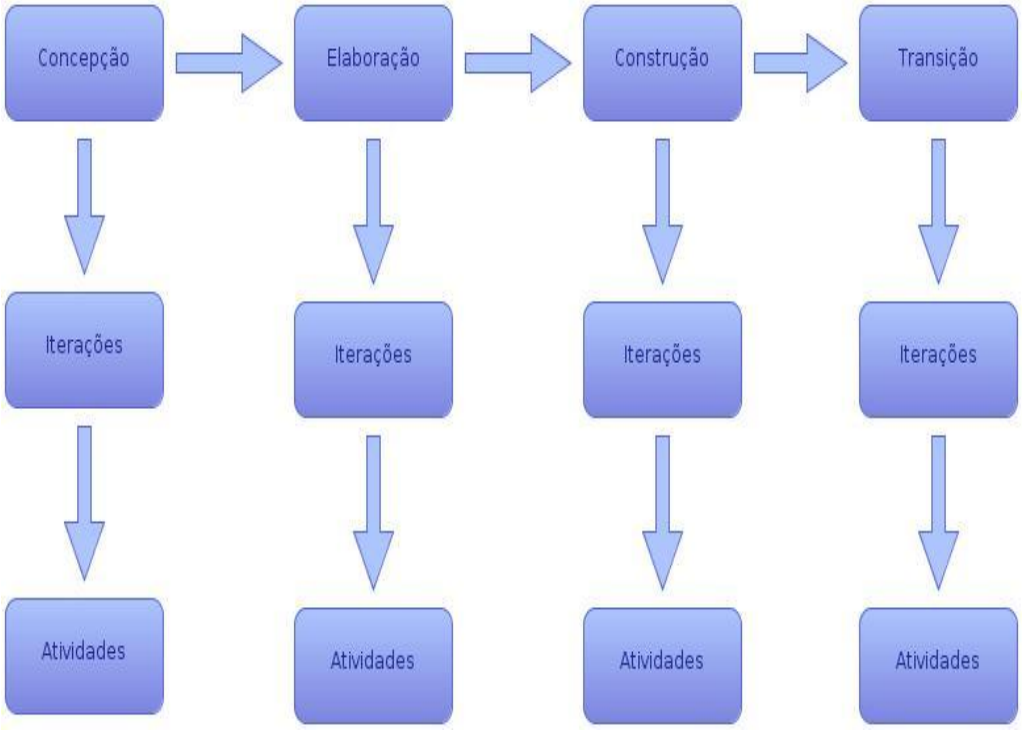

Figura 4 - Representação do modelo de processo baseado no SPEM

- Projeto de software - O elemento de projeto de software é o principal, pois é através dele que passam todas as interações e comandos do jogo, e também é o alvo de todo o gerenciamento feito pelo jogador (Gerente de Projetos) e outras tarefas automáticas do jogo (Souza et al 2010). Com o objetivo de tornar o projeto mais realista, foram considerados os seguintes elementos: prazo, que é o tempo definido inicialmente para entrega do projeto; recursos monetários que representa o caixa disponível; e qualidade do produto, mensurada com base no numero de erros existentes e na gerência de outros elementos (Souza et al 2010).
- Desenvolvedores - Desenvolvedores são membros da equipe, responsáveis por executar tarefas, contribuindo assim para o progresso do projeto. No SPARSE desenvolvedores possuem características como energia, habilidade para execução de um determinado tipo de tarefa, experiência em relação ao projeto, salário e atividades em que o mesmo se encontra alocado (Souza et al 2010). Na nova versão do simulador, outras características foram adicionadas e serão detalhadas posteriormente.
- Ferramentas - Além dos desenvolvedores, outro fator que influencia no progresso de projeto são as ferramentas. As ferramentas são softwares ou

técnicas de trabalho que agregam valor no desenvolvimento de um projeto, podendo influenciar no tempo de produção, quantidade de erros produzidos, dentre outros (Souza et al 2010). No SPARSE os atributos de ferramentas que foram considerados são: custo da ferramenta; habilitação, que determina se a ferramenta está em uso; influência que a ferramenta possui no desenvolvimento de determinada tarefa (Souza et al 2010).

Além destes elementos, existe ainda um conjunto de regras que são utilizadas durante a simulação, assim como as diferentes interfaces do jogo. As próximas seções tratam destes assuntos.

2.6.1 Regras

As regras do simulador foram definidas com base em um conjunto de boas práticas de Engenharia de Software (Navarro, 2006). Tais regras definem a maneira mais correta em que um projeto deve ser desenvolvido. Existem ainda eventos, que são aleatórios ou que podem ser disparados por regras. A contemplação de eventos no simulador tem como objetivo aproximar o simulador da realidade do desenvolvimento (Souza et al 2010). Mais detalhes sobre regras e eventos são descritos no capítulo 3.

2.6.2 Interface do Jogo

A interação do jogador com o SPARSE é feita através de uma interface tabular, por meio da qual o jogador visualiza as informações gerais de projeto que são necessárias para operar o jogo. Além disso, a interface também mantém um resumo textual com todas as ações previamente realizadas pelo jogador (Souza et al 2010). Para a nova versão do SPARSE foi criada uma nova interface tabular, contemplando novos requisitos de simulação. Esta interface pode ser visualizada na Figura 5.

Em paralelo ao desenvolvimento do simulador foi desenvolvido um novo padrão de interface com vários recursos gráficos em 3D conforme mostra a Figura 6.

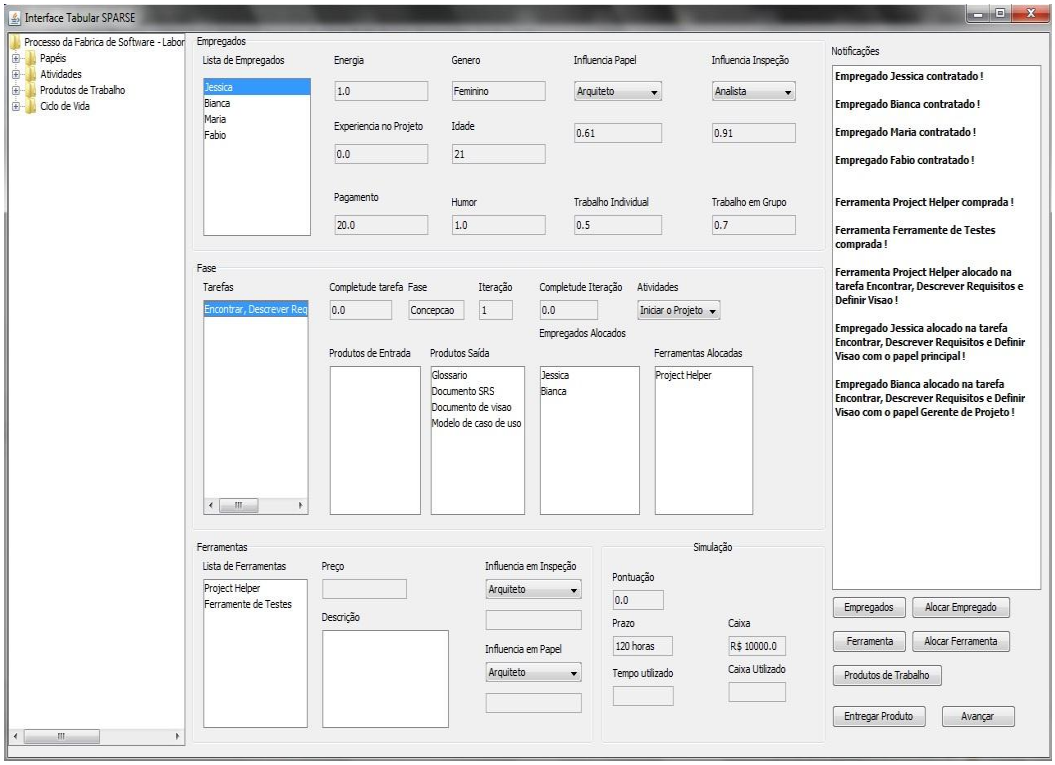

Figura 5 - Interface tabular do SPARSE

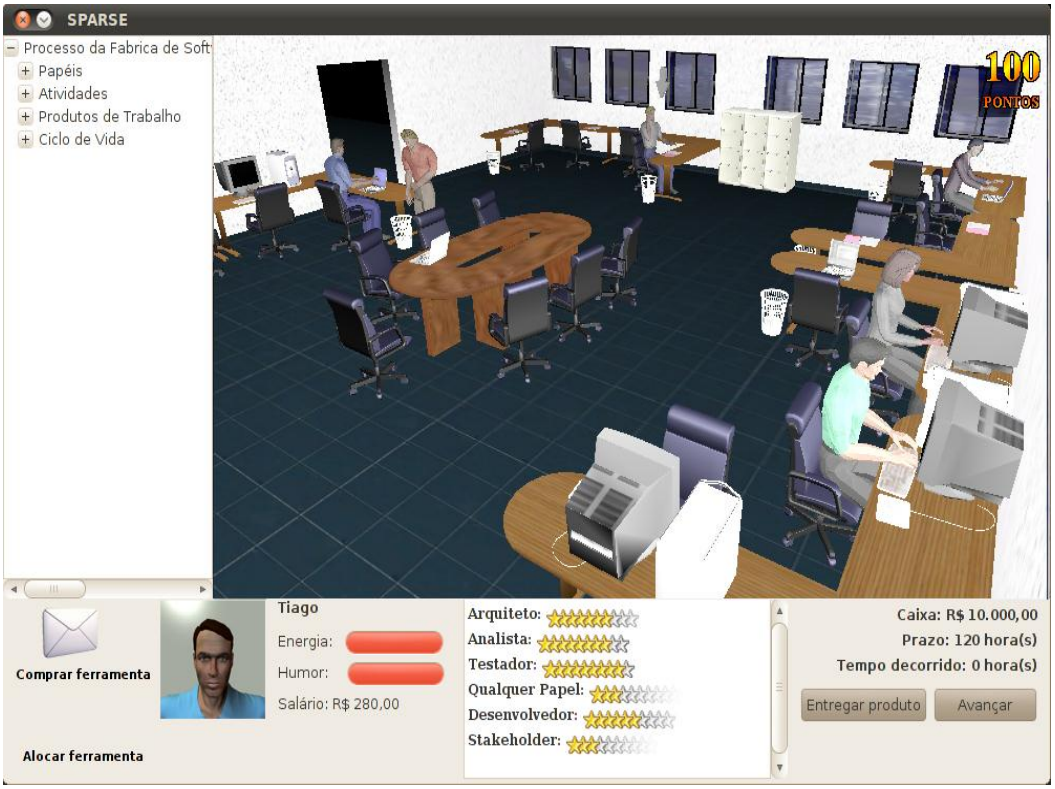

Figura 6 - Interface 3D do SPARSE

2.6.3 Proposta da nova estrutura do SPARSE

A nova estrutura do SPARSE proposta por Rodrigues (2010) foi desenvolvida para fornecer suporte à simulação de processos SPEM. A Figura 7 fornece uma visão geral desta nova estrutura, dividida em pacotes. Tal estrutura foi definida no trabalho de Rodrigues (2010), em conjunto com um trabalho de iniciação cientifica.

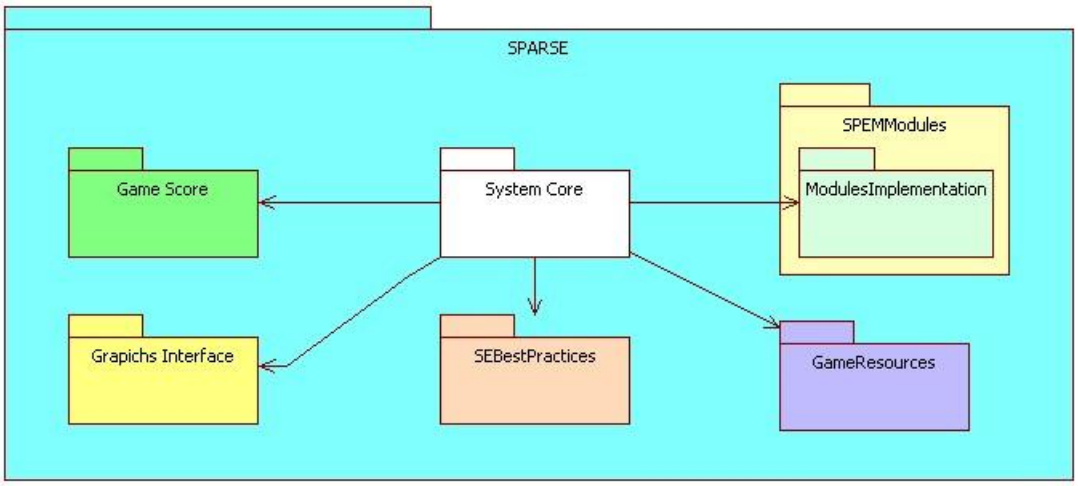

Figura 7 - Arquitetura proposta para a nova versão do SPARSE (Rodrigues 2010)

Segundo Rodrigues (2010) a proposta da nova versão apresenta uma melhor separação entre os conceitos relacionados ao modelo de processo e as regras de simulação relacionadas às boas práticas de Engenharia de software. Além disso, a nova estrutura apresenta uma maior capacidade para simular processos, possibilitando o aprendizado de uma maior variedade dos mesmos ao se utilizar a ferramenta.

O pacote SPEM Modules construído no trabalho de Rodrigues (2010) contém as classes abstratas e as classes de implementação de cada classe abstrata. É responsável por armazenar toda a lógica de processo.

O pacote Game Resources contém as classes que representam ferramentas, empregados e a figura do cliente. Mais detalhes podem ser vistos na seção 3.1.2.

O pacote Game Score é responsável por armazenar a pontuação do jogador e realizar os devidos cálculos. Recebendo códigos de erros, classes do pacote são responsáveis por recalcular a pontuação e retornar a mensagem correspondente. Para mais detalhes vide seção 3.1.3.5.

O pacote Graphics Interface contém as classes utilizadas para construção da interface com o usuário, tanto a tabular quanto a 3D. Para mais detalhes vide seção 3.1.3.4

O pacote principal é o System Core, sendo responsável por coordenar a interação entre os demais pacotes e atualizar as interfaces a partir da ocorrência de eventos. Para mais detalhes vide seção 3.1.3.1.

3
Desenvolvimento do jogo SPARSE baseado no meta-modelo SPEM

Este capítulo apresenta os passos necessários para a construção do simulador para o jogo educativo SPARSE. Na seção 3.1 são definidos quais foram os passos, e as seções posteriores detalham cada um destes passos respectivamente.

3.1 Considerações Iniciais

A construção de um novo simulador do jogo SPARSE foi necessária para permitir que todo processo construído com base no meta-modelo SPEM pudesse ser simulado no jogo.

O desenvolvimento do simulador foi realizado nos seguintes passos:

- Modelagem do simulador com o objetivo de adequação ao meta-modelo SPEM;
- Construção do modelo base de representação do processo a ser utilizado na simulação;
- Implementação do simulador segundo a modelagem previamente definida;
- Realização de testes para calibragem de variáveis e constantes do simulador.

Cada um destes passos será detalhado nas subseções seguintes.

3.1.1 Modelagem do simulador

Para tornar possível a simulação de processos baseados no SPEM, foi necessária a criação de uma nova estrutura para o simulador, visto que a versão anterior do SPARSE não suportava tal mudança.

Para a criação da nova estrutura, foi feita uma análise da especificação do SPEM (OMG 2008) e do Capítulo 5 do livro de Paula (2009), e foi então definido o conjunto de classes abstratas para a criação da estrutura de processos SPEM no contexto do SPARSE, como mostra a Figura 8 (Rodrigues, 2010).

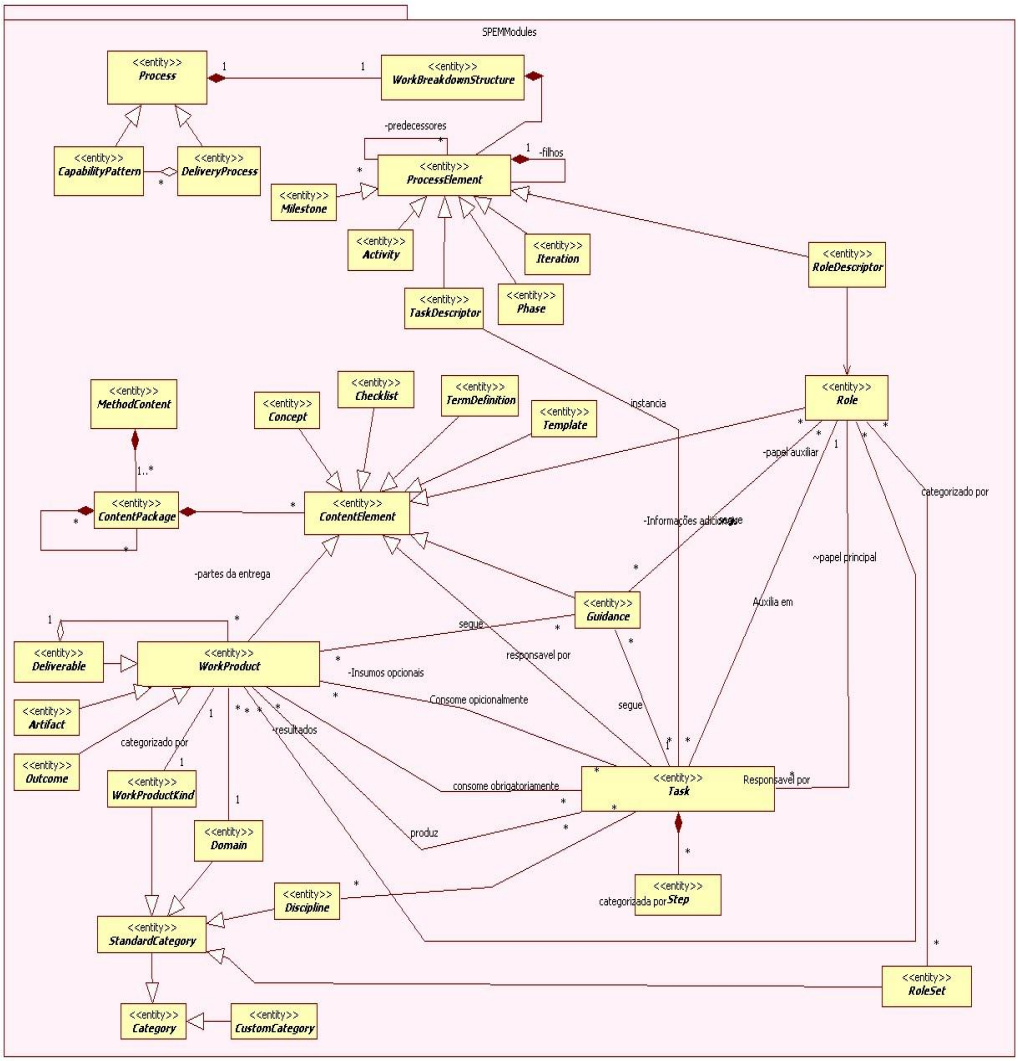

Figura 8 - Modelo de classes abstratas do SPEM (Rodrigues 2010)

Concluído o modelo de classes abstratas foi definido o conjunto de classes concretas para implementação das mesmas (Figura 9).

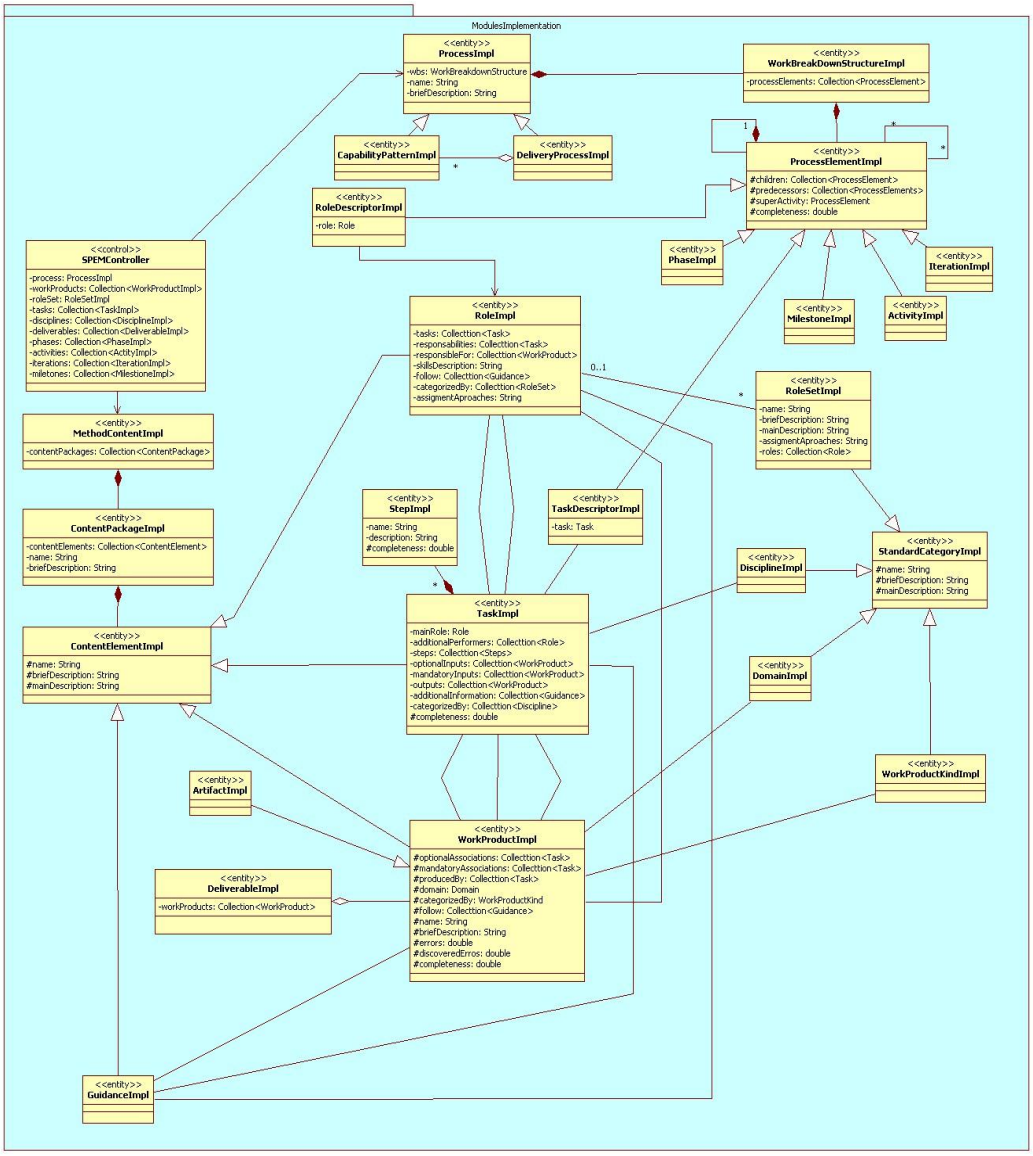

Figura 9 - Modelo de classes de implementação do SPEM no SPARSE (Rodrigues 2010)

A partir desta modificação foi possível iniciar a construção do simulador para permitir a execução de processos baseados no meta-modelo SPEM. Antes de iniciar a construção foi necessária a criação de um modelo para representação de tais processos. A seção seguinte detalha o modelo utilizado.

3.1.2 Construção do modelo de representação do processo

O modelo de processo foi criado utilizando um arquivo XML, com todo o conteúdo necessário para se iniciar a simulação: informações sobre o projeto, informações sobre empregados disponíveis, ferramentas que poderão auxiliar durante o jogo e o processo que será simulado (Figura 10).

Figura 10 – Estrutura geral do arquivo XML utilizado

Nas informações sobre o projeto estão inclusos: nome, descrição, orçamento que o jogador terá, e o prazo do projeto (Figura 11).

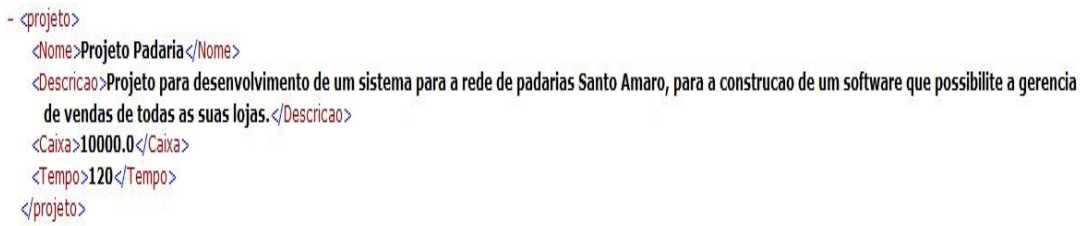

Figura 11 - Informações do projeto

Para aproximar a simulação ainda mais da realidade, foram adicionadas mais informações sobre empregados, além das já contempladas na primeira versão do SPARSE. As informações adicionadas foram: humor, que influencia diretamente na produtividade do empregado; habilidades de desenvolvimento e inspeção associadas aos papéis definidos pelo processo. Por exemplo, o empregado Jessica apresenta nível de habilidade 0.58 de desenvolvimento, quando o mesmo é alocado com o papel de desenvolvedor (Figura 12).

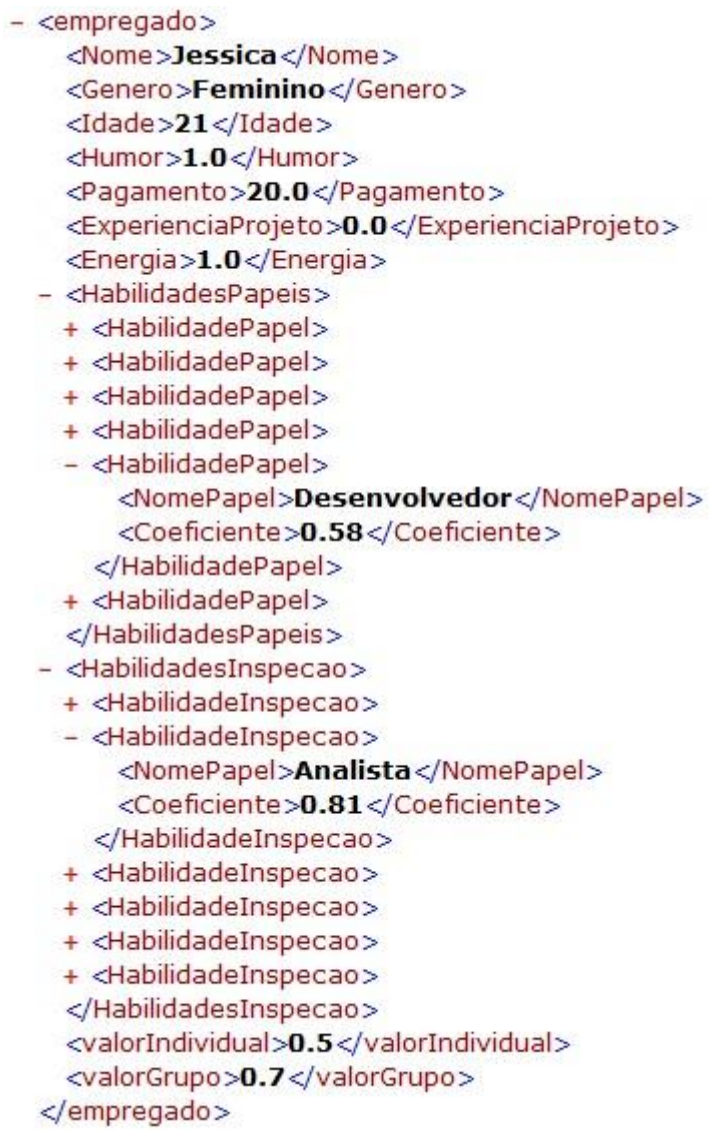

Figura 12 - Exemplo de Empregado

Com relação ao atributo humor, este pode aumentar caso o empregado receba um bônus salarial ou diminuir caso o empregado seja alocado em um número excessivo de atividades ou sua alocação seja feita sem respeitar sua preferência (atividades em grupo ou não). Assim como o humor, essa preferência influencia diretamente na sua produtividade, além de fatores como número de tarefas, habilidade e experiência.

No contexto da simulação, o nível de habilidade é representado por um coeficiente que define um valor associado a cada papel definido pelo processo. Tais coeficientes são utilizados para cálculos envolvendo tarefas e produtos de trabalho. Na versão anterior do simulador, coeficientes eram associados a disciplinas do

processo, como Requisitos, Testes, dentre outras. Tal modificação visa fornecer uma informação mais especifica com relação à habilidade do empregado em cada papel existente no processo. Os valores dos coeficientes variam entre 0 e 1, sendo 1 o valor máximo assumido.

Assim como para empregados existe um coeficiente de habilidade associada a cada papel, para ferramenta existem tais coeficientes representando a influência da ferramenta no desenvolvimento de tarefas e na inspeção de produtos de trabalho. Por exemplo, a ferramenta de testes, quando utilizada pelo papel testador, executando uma tarefa de teste, possui influência de 1.0 (Figura 13). Estes coeficientes foram adicionados para complementar as informações já contempladas na primeira versão do SPARSE.

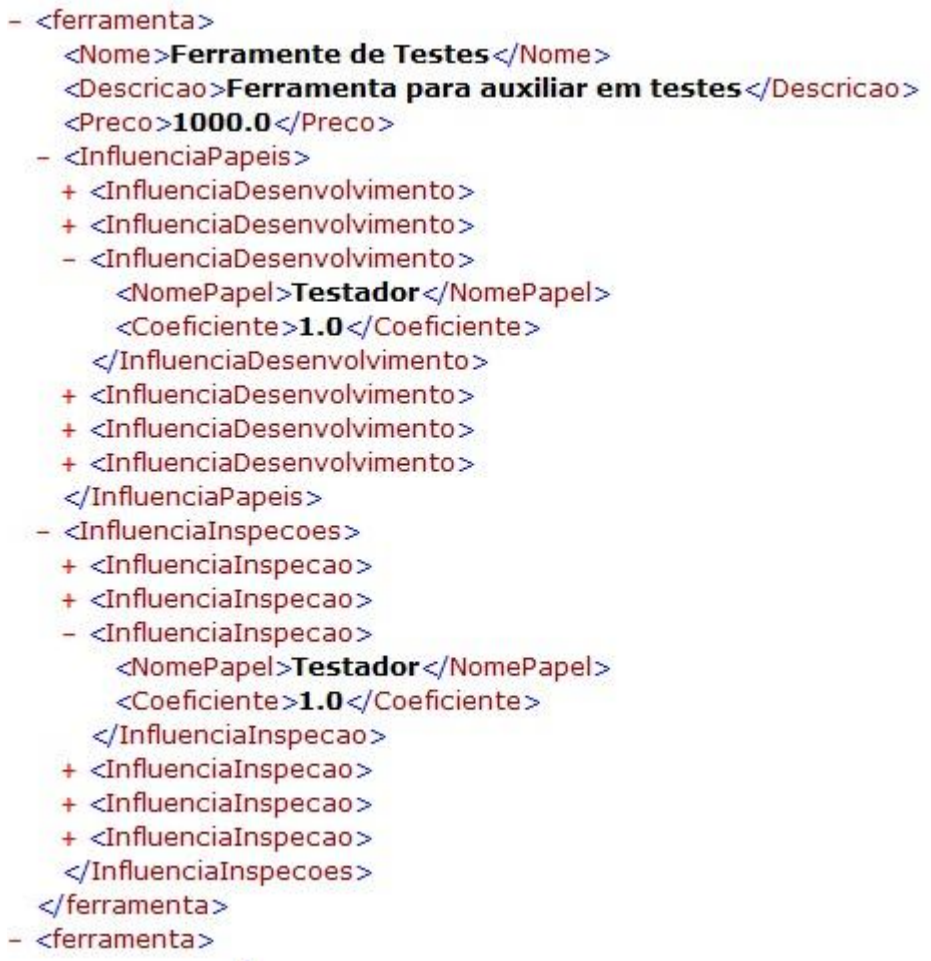

Figura 13 - Exemplo de ferramenta

As informações de processo incluem: nome, papéis, atividades, produtos de trabalho e informações do ciclo de vida do projeto (Figura 14).

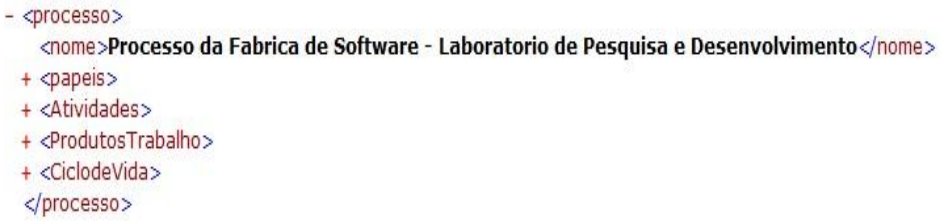

Figura 14 - Informações de projeto

Para cada papel existe a habilidade de um empregado para desenvolver ou para inspecionar, da mesma forma existe o nível de influência de uma ferramenta sobre uma atividade, seja ela uma tarefa ou relacionado a um produto de trabalho (Figura 15).

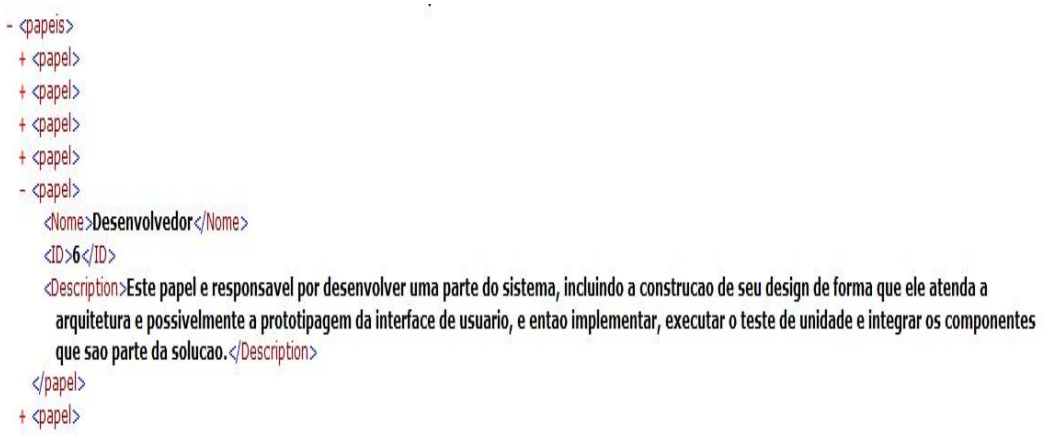

Figura 15 - Papéis do Processo

Uma atividade possui além de um nome, um conjunto de tarefas que a compõe (Figura 16).

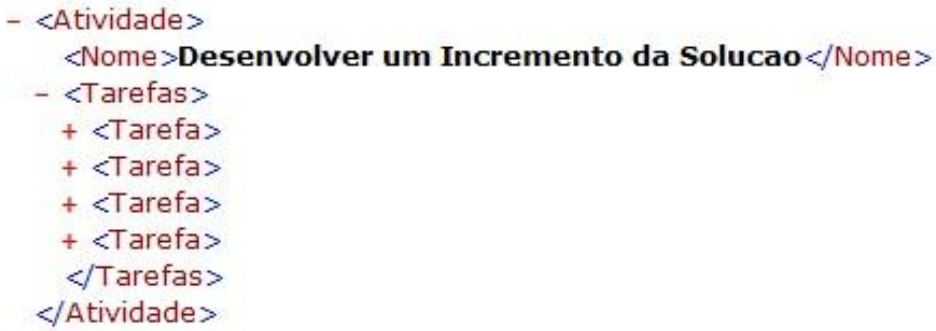

Figura 16 - Exemplo de uma atividade do processo

Uma tarefa é composta de nome, descrição, o papel principal associado à mesma, e os papéis adicionais que podem auxiliar durante a execução da tarefa.

Além disso, uma tarefa pertence a uma dada disciplina e possui uma lista com produtos de trabalho que são obrigatórios para que a mesma possa ser iniciada, e uma lista de produtos de trabalho que são gerados após a conclusão da tarefa (Figura 17).

```
- <Tarefa>
    <ID>4</ID>
    <Nome>Descrever a Arquitetura</Nome>
    <Descricao>Descreve a arquitetura atraves da analise dos requisitos arquiteturalmente signifucantes e pela identificacao de restricoes,
      decisoes e objetivos arquiteturais.</Descricao>
    <PapelPrincipal>3</PapelPrincipal>
  - <PapeisAdicionais>
    - <PapelAdicional>
        <IDPapel>7</IDPapel>
      </PapelAdicional>
    </PapeisAdicionais>
  - <ProdutosEntrada>
    - <Produto>
        <ID>6</ID>
      </Produto>
    - <Produto>
        <ID>7</ID>
      </Produto>
    - <Produto>
        <ID>8</ID>
      </Produto>
    - <Produto>
        <ID>9</ID>
      </Produto>
    </ProdutosEntrada>
  - <ProdutosSaida>
    - <Produto>
        <ID>1</ID>
      </Produto>
    </ProdutosSaida>
  </Tarefa>
```

Figura 17 - Exemplo de tarefa de uma das atividades do processo

Produtos de trabalho são organizados de acordo com a disciplina que o categoriza. Como citado acima, podem ser consumidos como entrada para iniciar uma tarefa e produzidos com a conclusão de uma tarefa.

Associado a um produto de trabalho, existe um identificador (ID) da tarefa que o produz. Esta informação é importante para identificar qual o coeficiente de desenvolvimento ou inspeção de um empregado, a ser utilizado para cálculos envolvendo o produto de trabalho (Figura 18).

```
<NomeDisciplina>Requisitos</NomeDisciplina>
- <Produtos>
  - <Produto>
      <ID>6</ID>
      <Nome>Glossario</Nome>
      <Description>Este artefato define termos importantes usados no projeto. Estes termos sao a base para a colaboracao eficaz com os
        Stakeholders e outros membros da equipe.</Description>
      <IDTarefa>1</IDTarefa>
    </Produto>
  - <Produto>
      <ID>7</ID>
      <Nome>Documento SRS</Nome>
      <Description>Detalha requisistos arquiteturalmente significantes.</Description>
      <IDTarefa>1</IDTarefa>
    </Produto>
  - <Produto>
      <ID>8</ID>
      <Nome>Documento de visao</Nome>
      <Description>Este artefato contem a definicao da visao dos Stakeholders a respeito do produto a ser desenvolvido, especificada em termos das
        principais caracteristicas e necessidades dos Stakeholders. Contem um esboco dos principais requisitos vislumbrados para o
        sistema.</Description>
      <IDTarefa>1</IDTarefa>
    </Produto>
  - <Produto>
      <ID>9</ID>
      <Nome>Modelo de caso de uso</Nome>
      <Description>Este artefato captura um modelo das funcoes desejadas do sistema e de seu ambiente, e serve como um contrato entre o cliente
        e os desenvolvedores.</Description>
      <IDTarefa>1</IDTarefa>
    </Produto>
</Produtos>
```

Figura 18 - Produtos de trabalho da disciplina de Requisitos

O ciclo de vida do processo é representado através de um conjunto de fases. Cada fase possui um marco que representa um ponto importante do projeto, além de iterações. Em cada iteração é definido o objetivo que deve ser alcançado ao fim da mesma e quais as atividades que devem ser desenvolvidas (Figura 19).

Todas as informações apresentadas nesta seção representam uma versão parcial do modelo utilizado para representação do processo a ser simulado.

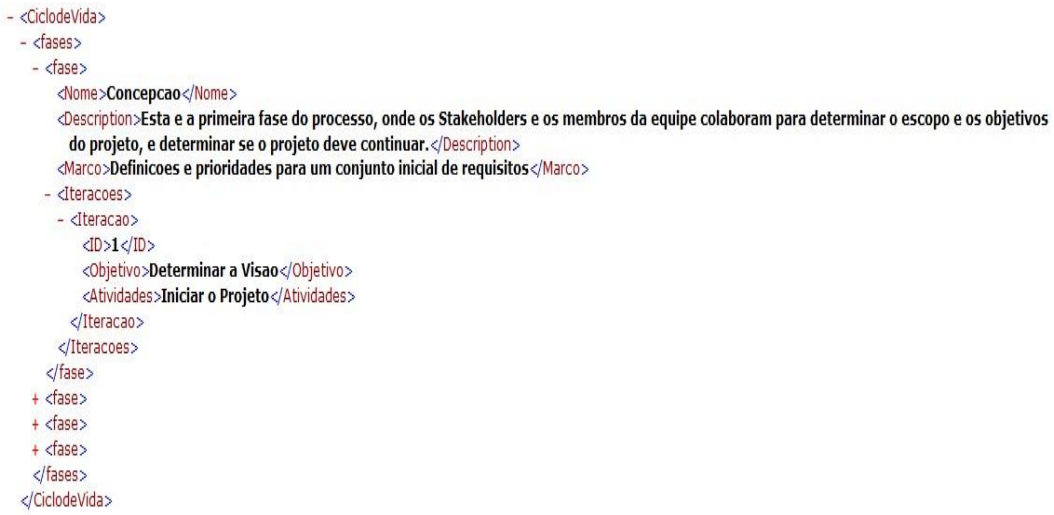

Figura 19 - Ciclo de vida do processo

3.1.3 Implementação do simulador

Após a construção do modelo de representação do processo foi iniciada a implementação do simulador, realizada nos seguintes passos:

- Implementação de classes do núcleo do simulador;
- Implementação de eventos;
- Adequação de regras já existentes na versão anterior do SPARSE;
- Implementação da interface gráfica do jogo em sua versão tabular;
- Implementação do módulo de pontuação e tutoria do jogo.

Cada uma das etapas definidas acima será detalhada nas próximas subseções.

3.1.3.1 Classes do núcleo do simulador

O primeiro passo para a implementação do simulador foi a implementação das classes do seu núcleo, que controla todos os elementos da simulação e é responsável por fornecer as informações necessárias às interfaces gráficas.

No núcleo se encontra a principal classe, a classe simulador, que possui acesso aos controladores, sendo responsável por redirecionar requisições ao devido controlador. Uma vez que o jogador executa uma ação, a mesma é enviada ao simulador, e repassada ao controlador responsável por processar as informações.

Dentre os controladores existentes na classe simulador, existe o controlador de interação, responsável pela interação entre empregados e ferramentas com tarefas e produtos de trabalho. Este controlador apresenta as funcionalidades de alocar e desalocar empregados e ferramentas de tarefas, bem como outras atividades relacionadas aos produtos de trabalho (inspeção e correção). Esta classe possui ainda os controladores de prazo, orçamento e qualidade.

O controlador de interação possui controlador para empregados com as funcionalidades de contratar, demitir e dar bônus a um determinado empregado, e o controlador de ferramentas com a funcionalidade de compra de ferramentas. Possui ainda um controlador de processo utilizado para verificações relacionadas ao processo e o controlador da pontuação com as funcionalidades de obter a pontuação atual, penalizar o jogador e exibir mensagens do tutor. Este controlador é responsável também pela execução das regras do jogo. Para mais detalhes sobre pontuação e tutoria, vide seção 3.1.3.5.

Para realizar alocações de empregados e ferramentas, o controlador de interação possui os métodos de alocação e desalocação. Estes métodos são responsáveis por acionar a alocação e desalocação, através das classes responsáveis: uma para alocação em tarefa, uma para alocação para inspeção de produtos de trabalho e outra para alocação em correção de produtos de trabalho.

Sempre que o jogador avançar uma hora no jogo, as seguintes ações são realizadas: adição de uma hora ao tempo utilizado pelo jogador; adição do custo por hora associado aos empregados contratados no inicio do jogo, e disparo das regras do jogo.

A ocorrência de qualquer evento durante a simulação faz com que o simulador notifique à interface as modificações ocorridas. A interface é então atualizada de acordo com a notificação recebida do simulador. A comunicação entre o simulador e a interface foi feita utilizando o padrão de projeto *Observer*. Para mais detalhes sobre este padrão vide Gamma et al 1995.

A próxima subseção apresenta a implementação de eventos no jogo.

3.1.3.2 Implementação de eventos

Durante a simulação existe a ocorrência de eventos. Um dos eventos é o surgimento de novos requisitos (Figura 20). Trata-se de um evento parcialmente aleatório que segue uma distribuição log-normal, como mostra a Figura 21.

Figura 20 - Evento novos requisitos

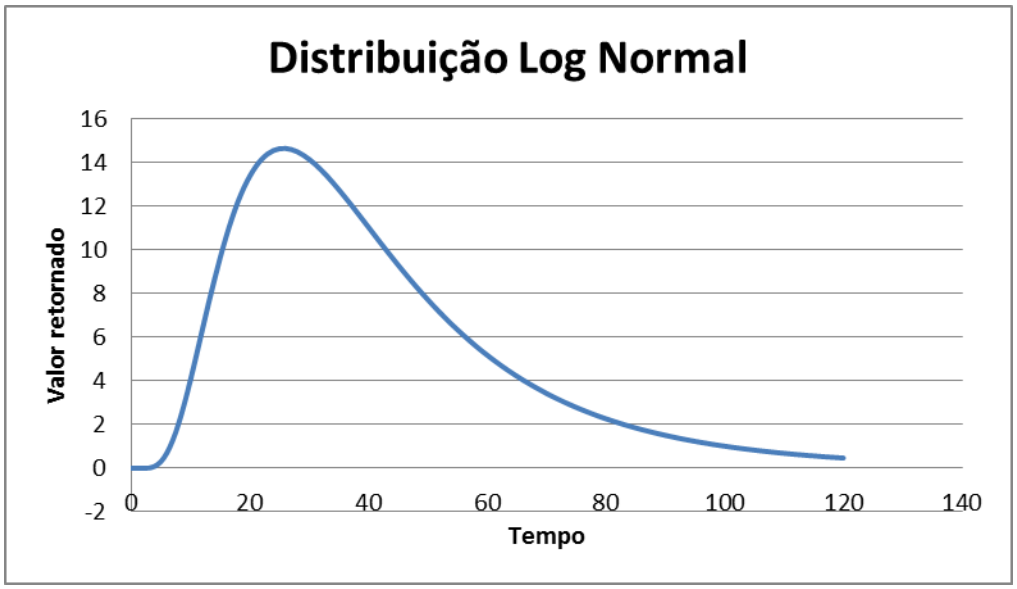

Figura 21 - Distribuição Log-Normal utilizada para o evento de novos requisitos

O comportamento desta distribuição, definido pela Equação 1, faz com que o evento aconteça durante o início da simulação (entre 18 e 38 horas) e à medida que o tempo avança o evento pára de acontecer. Esta distribuição foi escolhida por se assemelhar ao que acontece na realidade do desenvolvimento de software, em que o surgimento de novos requisitos é mais frequente no início dos projetos. A ocorrência real do evento é determinada através de um número aleatório e através do valor retornado pela distribuição. Caso este valor esteja em um intervalo definido, o evento poderá ocorrer.

$$f(x;\mu;\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}, x \succ 0$$

Equação 1 - Distribuição Log-Normal utilizada

Na Figura 21, a variável x representa o tempo atual da simulação e as variáveis μ e σ são utilizadas para determinar a altura e a amplitude da curva da distribuição log-normal.

Outro evento existente na simulação é a desalocação de um empregado por falta de energia, conforme exibido na Figura 22. Diferente do evento definido anteriormente, sua ocorrência não é aleatória, sendo determinada por ações do usuário. O evento ocorre quando um empregado está alocado e sua energia é menor do que 30% do total.

Figura 22- Evento desalocação por falta de energia

3.1.3.3 Adequação das regras

Para se adequar ao novo modelo de processo a ser simulado foi necessária a realização de modificações nas regras já existentes. Em um primeiro momento foi adicionado o atributo humor no cálculo de produtividade de empregados.

Após isso, diferentemente da versão anterior do SPARSE, onde um desenvolvedor possuía habilidade em relação às disciplinas do processo, na versão atual o empregado possui habilidade com relação aos papéis do processo, sendo alterada a maneira como tal habilidade é obtida através das regras.

Por fim foi adicionada a regra para cálculo de humor de empregado, que não existia anteriormente.

As regras utilizadas durante a simulação são:

- Completude;
- Erros existentes;
- Erros descobertos;
- Erros corrigidos;
- Energia;
- Humor;
- Experiência.

A regra de completude está associada às tarefas que possuem empregados e ou ferramentas alocadas. A completude da tarefa é calculada no tempo atual, sendo influenciada pela capacidade do empregado e a influencia de ferramentas utilizadas. O cálculo da capacidade é influenciado pela preferência de trabalho e nível de habilidade do empregado (valores fixos), bem como pelo nível de energia, humor e experiência atuais do mesmo. Durante a execução desta regra um empregado com energia consideravelmente baixa é desalocado automaticamente. Para mais detalhes vide seção 3.1.3.2.

A regra de erros existentes está associada a um produto de trabalho, e assim como a regra de completude, também é influenciada pela capacidade do empregado e pela influência de ferramentas utilizadas. Tal regra é acionada sempre que uma tarefa é concluída, e produtos de trabalho são de fato produzidos, definindo a quantidade de erros que o produto apresentará.

A regra erros descobertos também está associada a um produto de trabalho e é utilizada quando o jogador utiliza a atividade de inspeção, lembrando que não se pode encontrar mais erros do que a quantidade existente. A regra de erros descobertos é influenciada pela capacidade do empregado e influência da ferramenta.

A regra de erros corrigidos é utilizada quando o jogador utiliza a atividade de correção, sendo que não se pode corrigir mais erros do que foi descoberto. Esta regra é influenciada pela capacidade do empregado e influência da ferramenta.

As regras de energia, humor e experiência estão associadas a empregados, onde os novos valores dependem da quantidade de tarefas em que um empregado está alocado.

3.1.3.4 Interface gráfica

Ao iniciar o jogo uma tela inicial é exibida com as informações gerais do projeto a ser desenvolvido, bem como algumas instruções ao jogador, como pode ser visto na Figura 23.

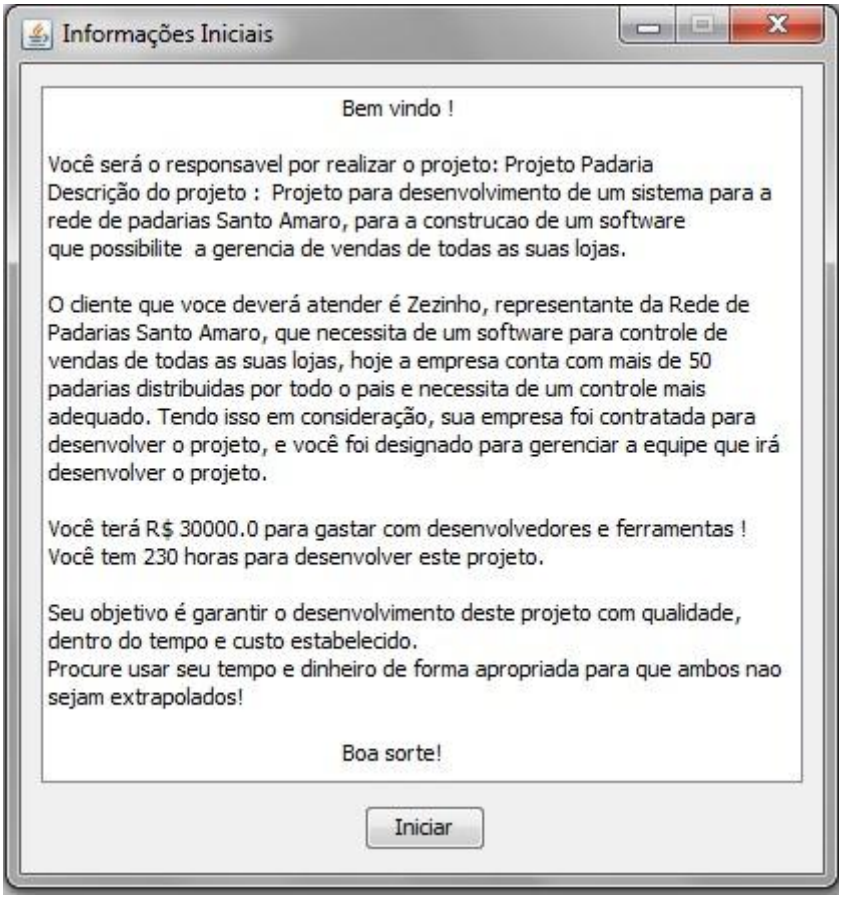

Figura 23 - Interface Inicial do jogo

A primeira decisão do jogador é contratar a sua equipe a partir da lista de empregados, definida previamente através do arquivo XML de entrada de dados (mais detalhes vide 3.1.2). A Figura 24 ilustra a interface de contratação de empregados.

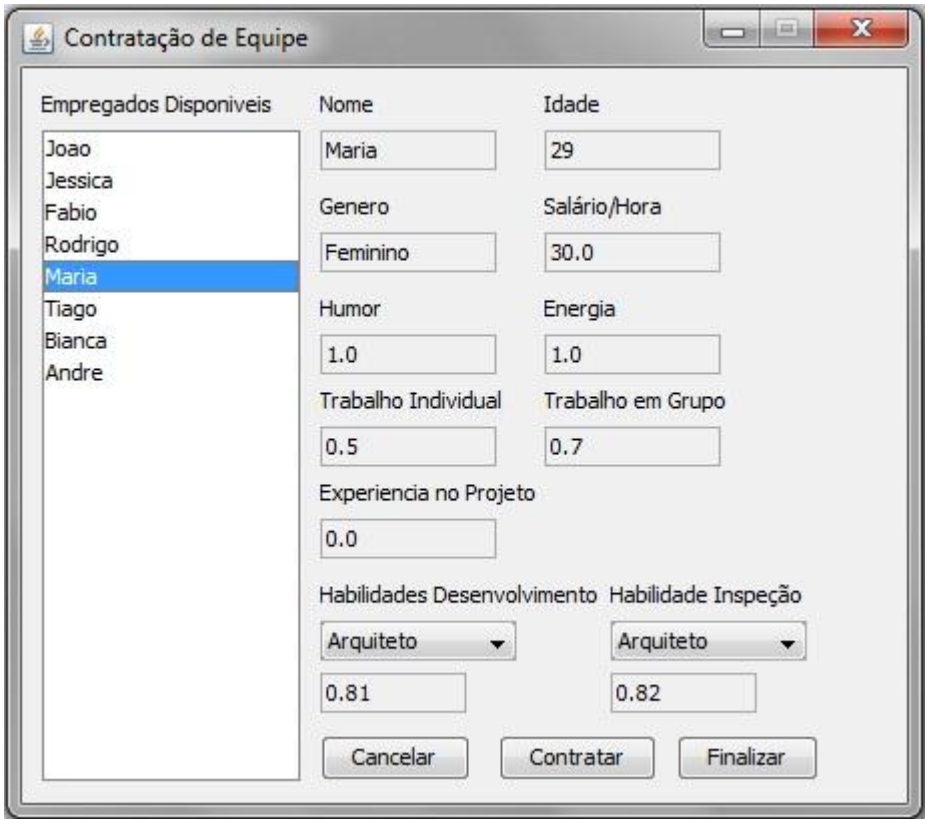

Figura 24 - Interface Contratação Equipe

O jogador, ao selecionar um empregado da lista, pode visualizar todas as informações do mesmo. Dessa forma é possível montar uma equipe da maneira que o jogador achar conveniente. Após a contratação da equipe, ao pressionar o botão finalizar, o jogador é redirecionado para a tela principal do jogo.

Na tela principal do jogo, à esquerda, é apresentada uma estrutura em árvore representando o processo simulado, com as informações de papéis, atividades e suas respectivas tarefas, os produtos de trabalho, organizados por disciplinas e a representação do ciclo de vida do projeto (Figura 25).

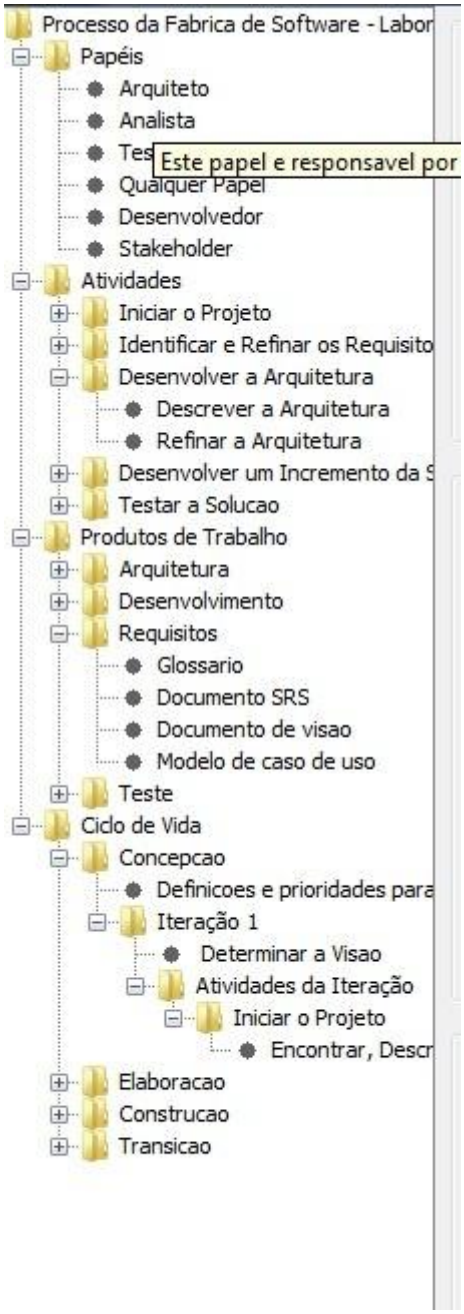

Figura 25 - Estrutura em árvore para representação do processo

Existe também na parte central, a lista dos empregados que foram contratados, e ao selecionar algum empregado, os campos são preenchidos com as suas informações (Figura 26).

Figura 26- Interface Tabular - Empregados Contratados

Em seguida são exibidas as informações da fase e iteração atuais, a completude da iteração, bem como suas atividades e tarefas. Ao selecionar uma tarefa, são mostradas as informações dos produtos de trabalho de entrada, necessários para iniciar a tarefa, os produtos de trabalho que serão produzidos após a conclusão da tarefa, os empregados que estão alocados na tarefa e as ferramentas que estão sendo utilizadas (Figura 27).

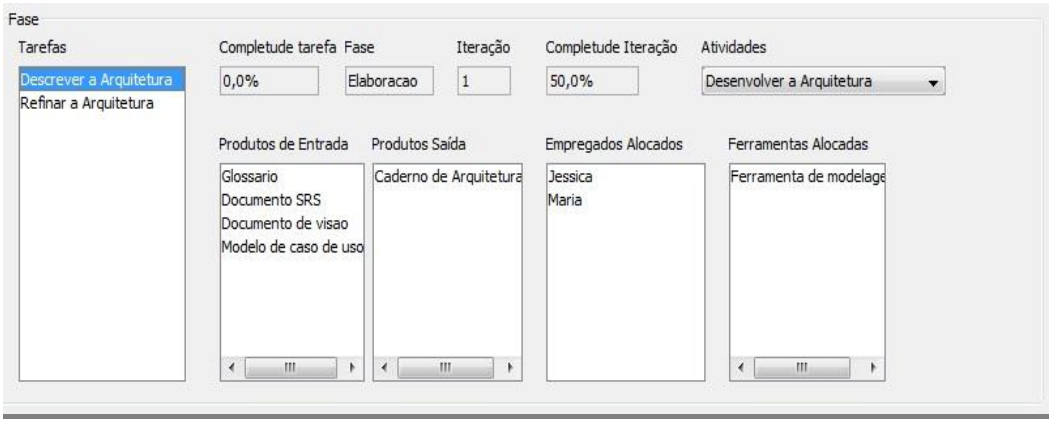

Figura 27 - Interface Tabular - Fase atual do processo

Além disso, são exibidas informações sobre as ferramentas que foram compradas, e estão disponíveis para serem utilizadas no auxílio do desenvolvimento de tarefas (Figura 28).

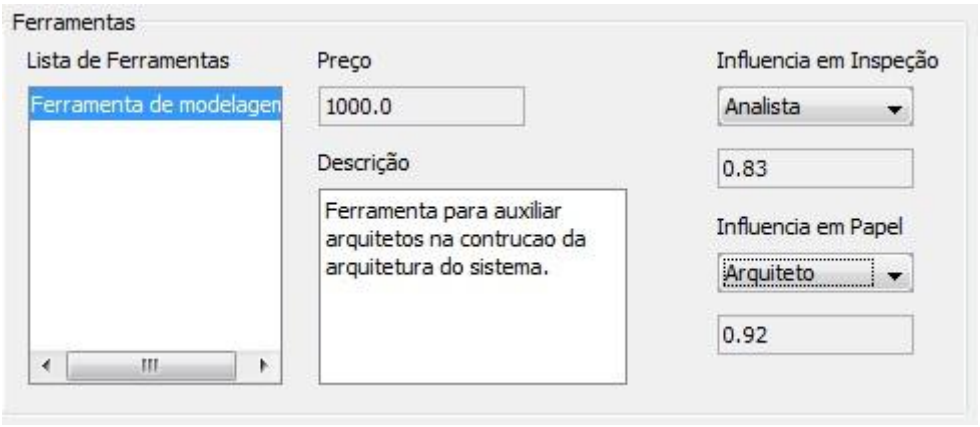

Figura 28 - Interface Tabular - Ferramentas Compradas

Também são exibidas as informações gerais do projeto, tais como o prazo que o jogador tem para desenvolvê-lo e o tempo já consumido, o caixa disponível e o caixa utilizado, e a pontuação atual do jogador (Figura 29).

Figura 29 - Interface Tabular - Informações da Simulação

Além disso, a interface tabular conta com um painel de texto através do qual o jogador tem acesso a decisões tomadas previamente, e também a mensagens de erros, relacionadas a ações do jogador não condizentes com boas práticas da Engenharia de Software, além de dicas de como prosseguir durante o jogo (Figura 30).

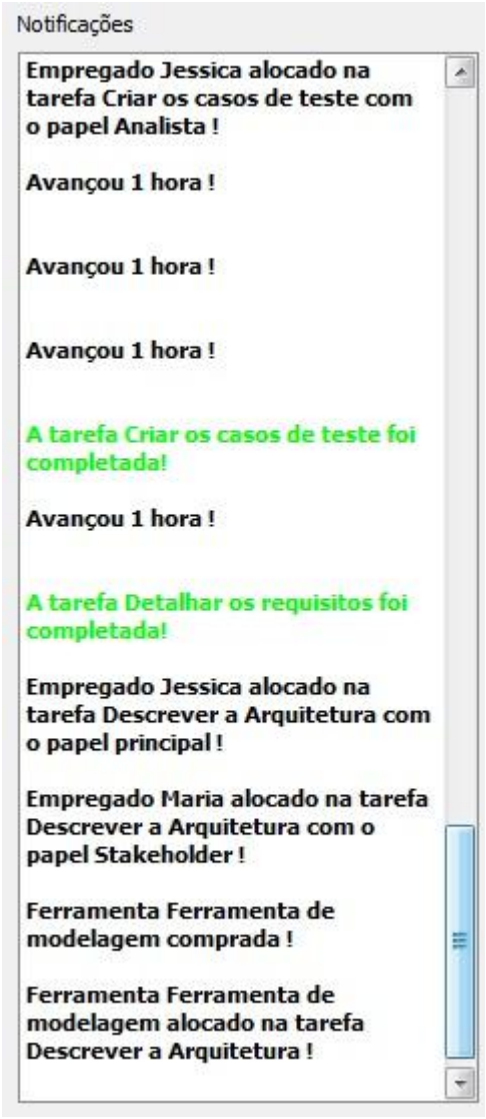

Figura 30 - Interface Tabular - Notificações

Através da tela principal do jogo, tem-se acesso a algumas funcionalidades, tais como: gerência da equipe contratada, compra de ferramentas disponíveis, alocação de funcionários e ferramentas em atividades, visualização de produtos de trabalho já concluídos, opção de finalização do projeto com entrega do produto final ao cliente e a opção de avançar uma hora no jogo (Figura 31).

Figura 31 - Interface Tabular - Funcionalidades

Ao selecionar a funcionalidade de entrega do projeto são feitas verificações finais para saber o desempenho do jogador. São verificados se prazo e orçamento iniciais foram respeitados, a quantidade de erros em produtos de trabalho que não foram detectados e um cálculo para a pontuação final do jogador.

A interface de gerência de empregados permite ao jogador tanto demitir um empregado previamente contratado, como fornecer um bônus salarial, fazendo com que o humor do empregado aumente, influenciando sua produtividade (Figura 32).

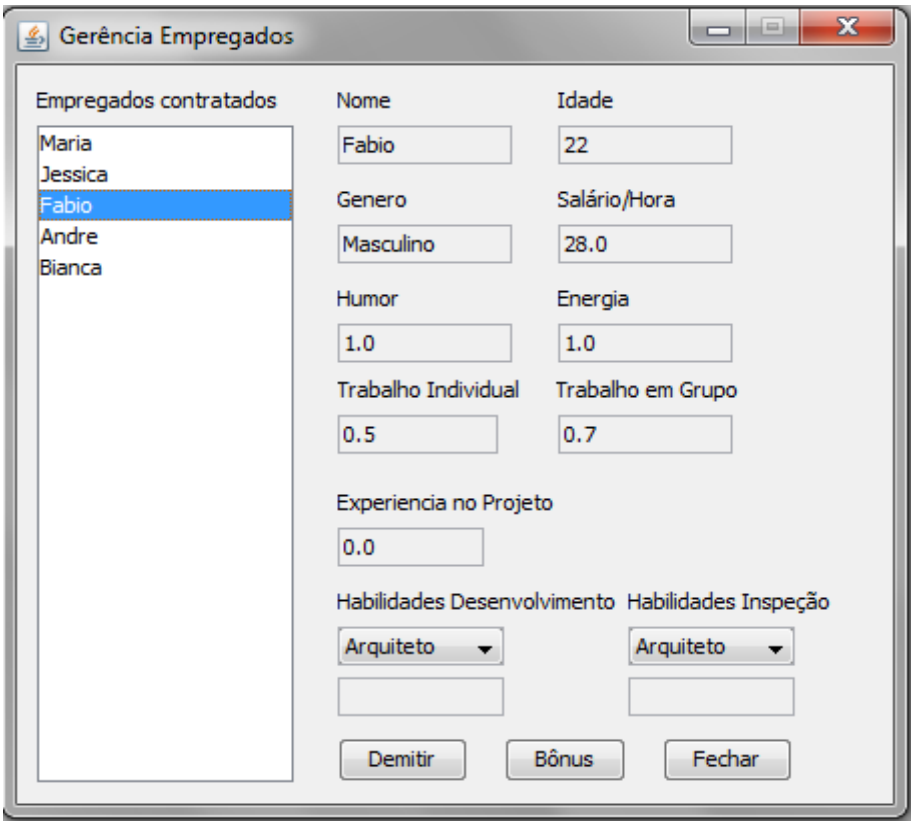

Figura 32 - Interface Gerência de Empregados

A interface de gerência de ferramentas permite ao jogador comprar ferramentas de uma lista disponibilizada. Ferramentas influenciam diretamente na produção de uma tarefa (Figura 33).

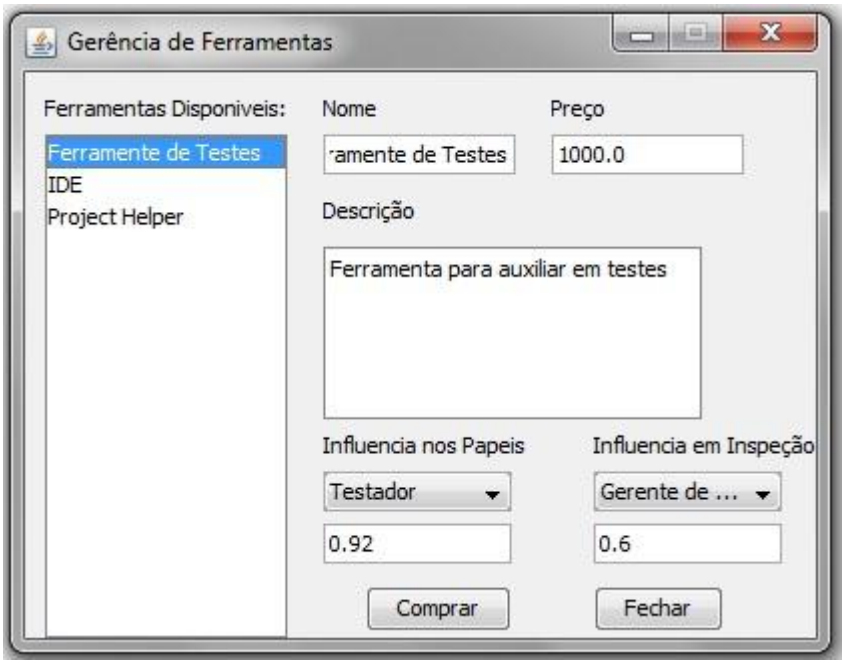

Figura 33 - Interface Gerência de Ferramentas

A interface de alocação de empregado conta com uma lista de empregados contratados, atividades da iteração atual, uma lista das tarefas da atividade selecionada, um campo com o papel principal da tarefa, e uma lista com os papéis adicionais à tarefa. São disponibilizadas ainda as funcionalidades de alocar papel principal, alocar papel adicional e desalocar um empregado de uma tarefa (Figura 34).

Similar à Interface de Alocação de Empregados, a Interface Alocação de Ferramentas possui uma lista de ferramentas compradas, atividades da iteração atual, uma lista de tarefas da atividade selecionada e as funcionalidades de alocar ferramenta e desalocar ferramenta (Figura 35).

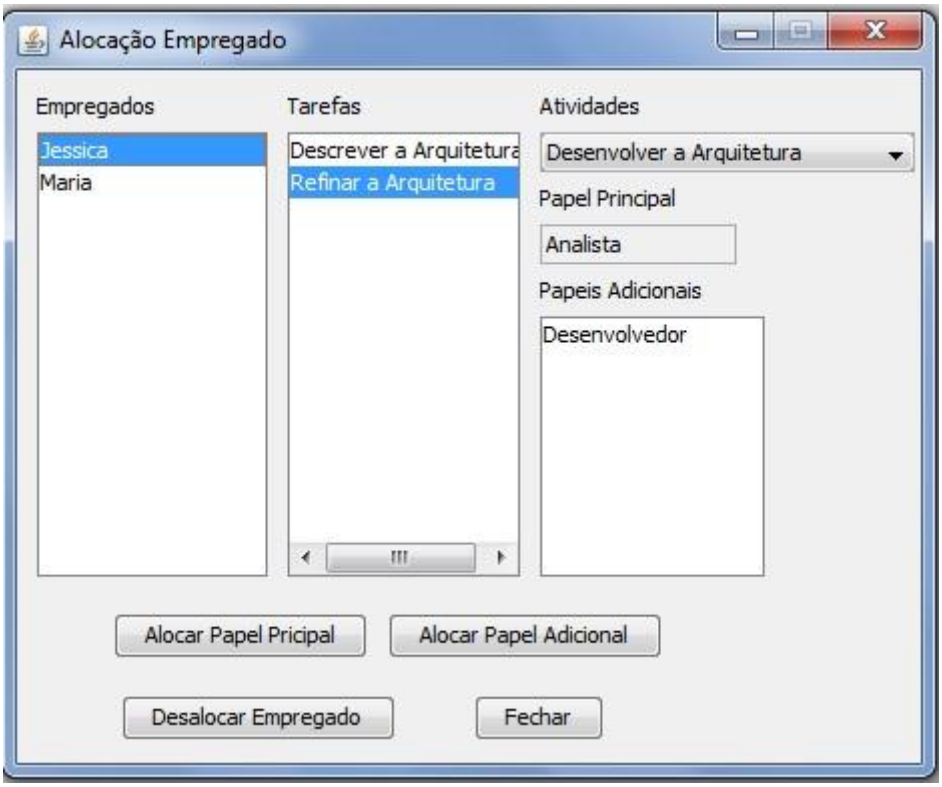

Figura 34 - Interface Alocação Empregados

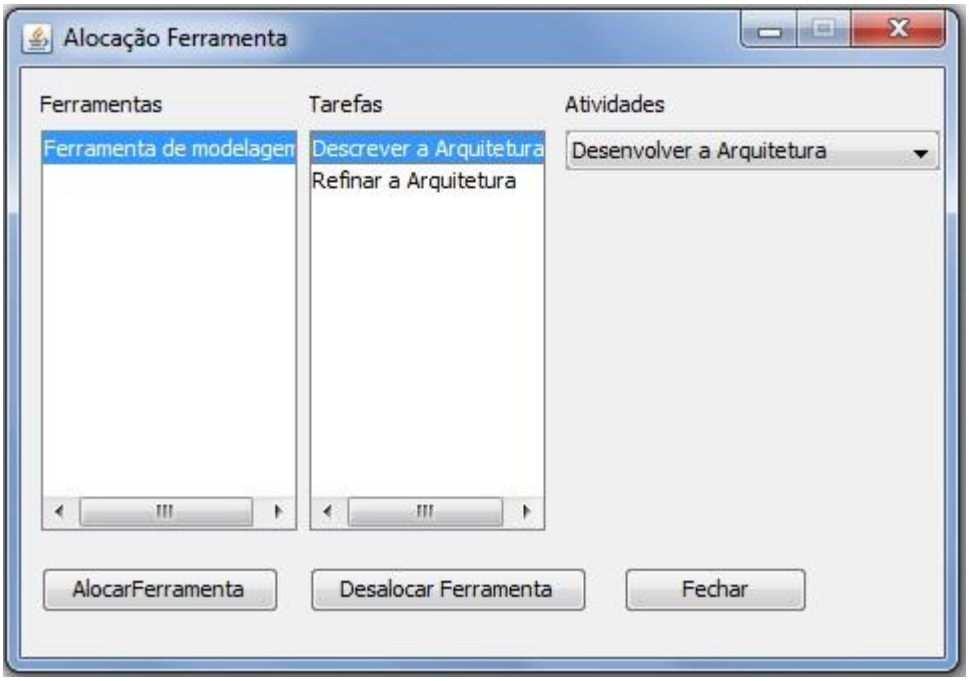

Figura 35 - Interface Alocaçao Ferramentas

A Interface Produtos de trabalho contém uma lista dos produtos de trabalho já produzidos, a lista de empregados contratados e a lista de ferramentas compradas. Além disso, possui as funcionalidades de alocar e desalocar um empregado para inspeção (através da inspeção o jogador encontra erros existentes no produto de trabalho) e correção (através da correção o jogador corrige os erros que a inspeção detectou), e as funcionalidades de alocar e desalocar uma ferramenta para auxilio na inspeção e correção de produtos de trabalho (Figura 36).

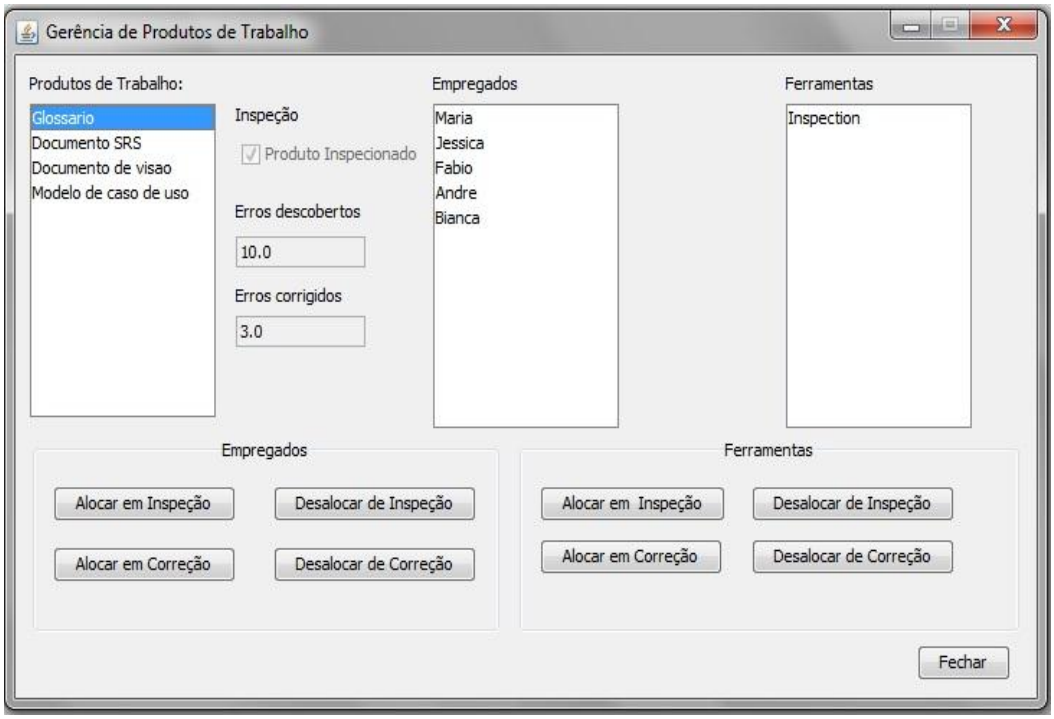

Figura 36- Interface Gerência de Produtos de Trabalho

3.1.3.5 Módulo de pontuação e tutor

Sendo o SPARSE um jogo educativo, surge a necessidade de definir uma forma de avaliação para as ações tomadas (pontuação), e uma forma de, caso o aluno não as tome corretamente, direcioná-lo ao que seria o mais correto a se fazer (tutor).

O módulo de pontuação possui a classe Score, responsável por armazenar a pontuação atual do jogador. O jogo inicia com uma pontuação máxima de 100 pontos e o jogador perde pontos à medida que ações errôneas são tomadas. Além disso, mensagens associadas a erros cometidos são exibidas, bem como mensagens

do tutor, que representam o direcionamento para a melhor tomada de decisão (Figura 37).

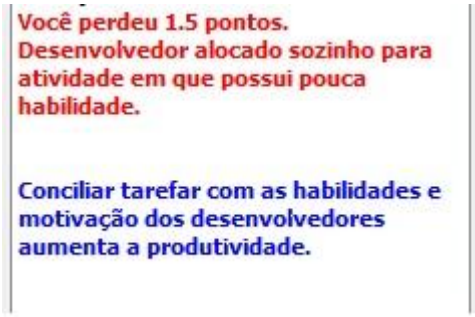

Figura 37 - Exemplo de mensagem associada a erro e mensagem do tutor

Para o acesso à pontuação e às mensagens de erro e tutoria, existe o controlador da pontuação, Score Controller. Para se alterar a pontuação, é necessário informar o código do erro a este controlador, de forma que o mesmo retorne a mensagem associada ao erro, o mesmo acontecendo para as mensagens do tutor. As mensagems do tutor também estão associadas a erros, indicando o que o jogador deveria ter feito para que o erro não acontecesse.

A classe Score Controller recebe mensagens de erros de outras classes, altera a pontuação e é responsável por mantê-la sempre atualizada para as classes que necessitam de tal valor.

3.1.4 Testes

Após a conclusão da implementação foi necessária a realização de testes, afim de tornar a simulação o mais próximo possível da realidade. Em um momento inicial foram feitos testes para definição de valores de constantes de simulação. Após isso testes foram realizados para definição dos valores para cálculo da pontuação.

Durante a realização de testes para definição de constantes, foi definido um valor inicial para cada constante, e com o andamento dos testes os mesmos foram alterados até que se conseguisse uma configuração satisfatória. Tais constantes são utilizadas para os cálculos de regras.

Em relação aos testes da pontuação, inicialmente foi definido qual seria a gravidade de cada um dos erros existentes, onde erros mais graves teriam

uma penalidade maior comparado a um erro mais simples. Após isso testes foram feitos para definição do valor de penalidade para cada erro.

Após a conclusão dos testes o simulador pôde ser utilizado por alunos de graduação para uma avaliação da ferramenta. Detalhes sobre a avaliação são exibidos no próximo capítulo.

4
Estudo de caso: o uso do jogo SPARSE por alunos de graduação

Este capítulo apresenta uma avaliação inicial do jogo SPARSE, a fim de validar a simulação no jogo. Essa avaliação se deu através da utilização do SPARSE por alunos da disciplina de Engenharia de Software do curso de Ciência da Computação da Universidade Federal de Alfenas. Os questionários utilizados para avaliação são encontrados nos Apêndices A e B.

4.1 Descrição do estudo de caso

A fim de validar a simulação do SPARSE, foi feita uma avaliação do uso do jogo por alunos da disciplina de Engenharia de Software. Após o uso, os alunos preencheram dois questionários, sendo que um deles avaliava o perfil de cada aluno e o outro apresentava questões relacionadas à opinião pessoal sobre a ferramenta. A versão de interface utilizada na avaliação foi a interface gráfica 3D.

A avaliação foi realizada em um dos laboratórios do curso, com alunos do 3° período do curso de Ciência da Computação da Universidade Federal de Alfenas (UNIFAL), cursando a disciplina de Engenharia de Software.

Inicialmente, o jogo foi apresentado aos alunos, junto com uma explicação inicial de seu funcionamento e regras, bem como sobre as funcionalidades da interface 3D, significado de animações, dentre outros. Após isso, os alunos jogaram por um tempo para se familiarizarem com a ferramenta.

Durante o processo de familiarização, à medida que surgiam dúvidas, as mesmas eram esclarecidas. A principal dúvida foi em relação ao evento de novos requisitos, de modo que os alunos a princípio não entendiam o impacto do

acontecimento do evento, necessitando uma nova explicação. Outra dúvida foi em relação ao motivo de certas mensagens do tutor no painel de notificações.

Em outro momento, com os alunos já familiarizados com a ferramenta e as dúvidas esclarecidas, os mesmos jogaram para que pudesse então ser feita a avaliação de cada aluno sobre a ferramenta. Em seguida os alunos responderam os questionários de avaliação de perfil e avaliação da ferramenta. Os questionários são encontrados nos Apêndices A e B. Os resultados da avaliação realizada são apresentados no próximo capítulo.

4.2 Resultados

Concluída a avaliação e finalizado o preenchimento dos questionários, tornou-se possível à análise dos resultados para verificação do perfil dos alunos que utilizaram a ferramenta e da opinião pessoal dos mesmos sobre a ferramenta. A partir dos resultados obtidos foram produzidos gráficos para facilitar a visualização dos mesmos. A Figura 38, Figura 39, Figura 40, Figura 41, Figura 42 e Figura 43 ilustram os resultados relacionados ao primeiro questionário de perfil.

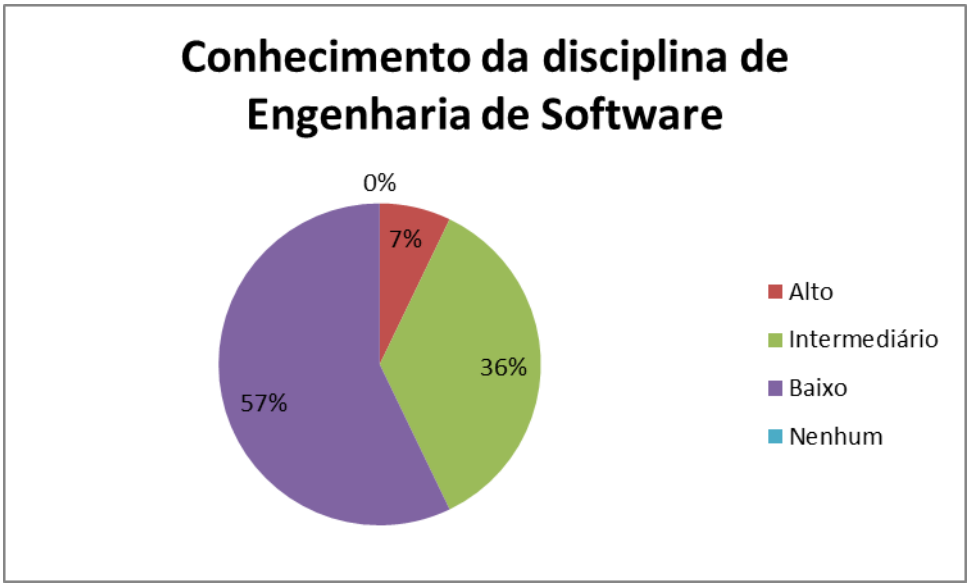

Figura 38 - Resultado: Conhecimento sobre Engenharia de Software

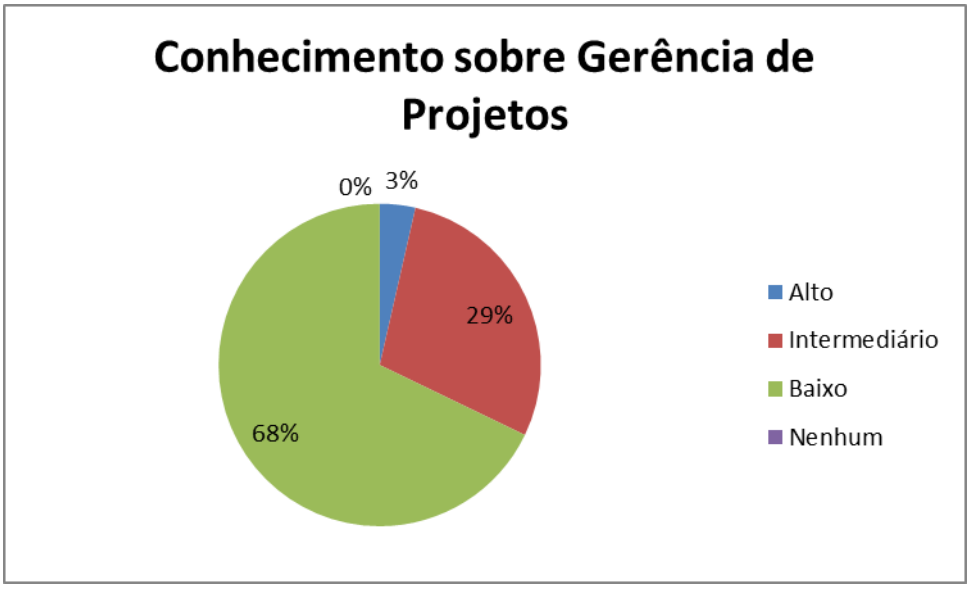

Figura 39 - Resultado: Conhecimento sobre Gerência de Projetos

Como pode ser observado na Figura 38 e na Figura 39 a maioria dos alunos possui conhecimento entre baixo e intermediário sobre conceitos de Engenharia de Software e Gerência de Projetos. Isso pode estar relacionado ao caráter teórico da disciplina de Engenharia de Software e o fato do conteúdo de Gerência de Projetos ser abordado somente ao final do curso, sem aprofundamento no tema.

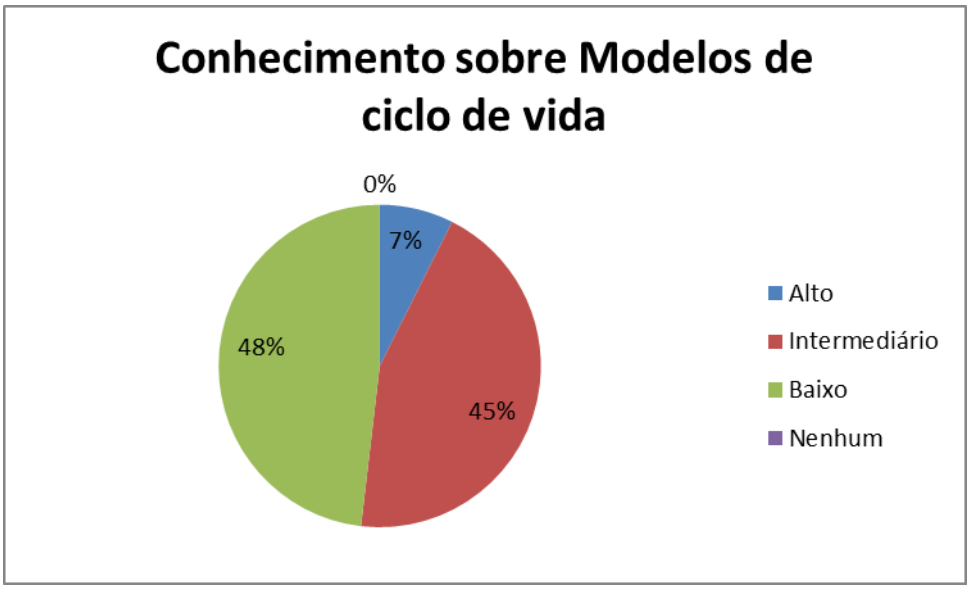

Figura 40 - Resultado: Conhecimento sobre modelos de ciclo de vida

Em relação aos modelos de ciclo de visa básicos, tais como modelo cascata, modelo espiral, modelo de prototipagem em sua maioria os alunos possuem conhecimento entre intermediário e baixo (Figura 40).

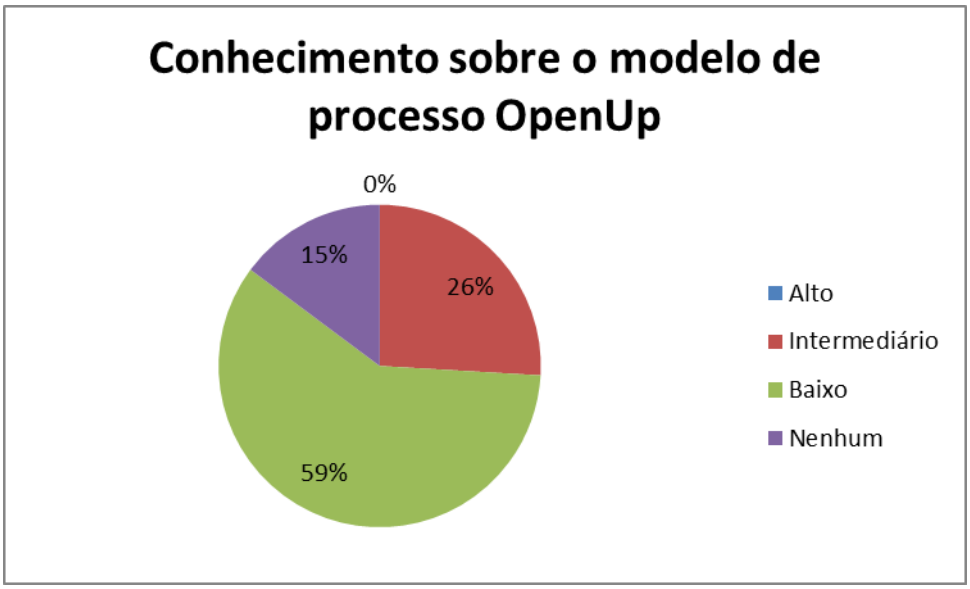

Figura 41 - Resultado: Conhecimento sobre o modelo de processo OpenUp

Em relação ao modelo de processo OpenUp (Figura 41), a maioria dos alunos possui conhecimento baixo, sendo que o OpenUp não é um modelo apresentado com detalhes pela disciplina de Engenharia de Software. É importante frisar que este é o modelo utilizado como base para simulação no jogo SPARSE.

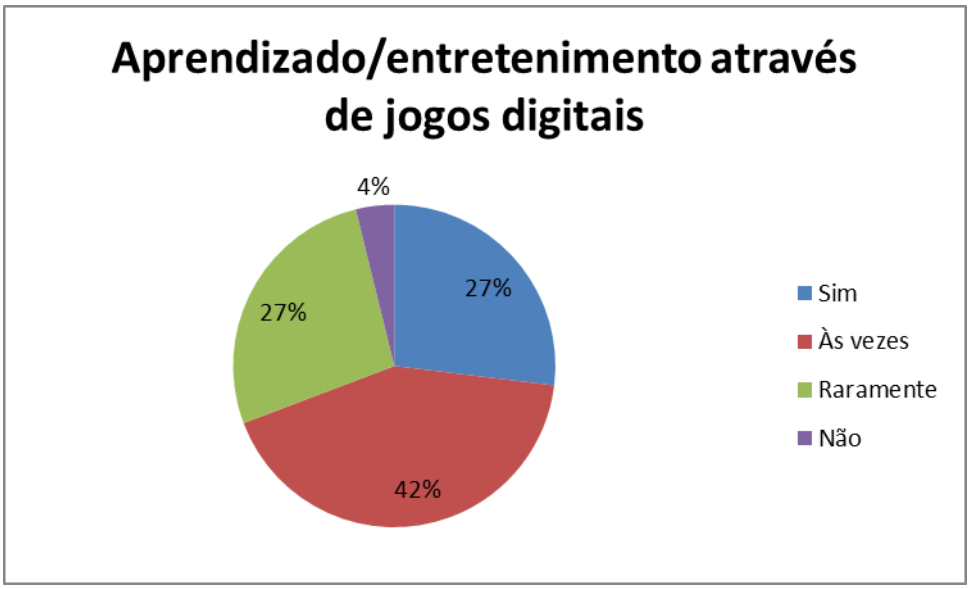

Figura 42 - Resultado: Aprendizado/entretenimento através de jogos digitais

Através dos resultados mostrados na Figura 42 observa-se que a maioria dos alunos utilizam jogos digitais como forma de aprendizado/entretenimento, o que mostra o interesse dos mesmos para este tipo de ferramenta, sendo uma das razões para sua utilização como forma de estímulo ao aprendizado.

Perguntados sobre o conhecimento sobre o jogo educativo SimSE (mais detalhes vide Navarro, 2006) a maioria dos alunos responderam que não conhecem, como pode ser visto na Figura 43.

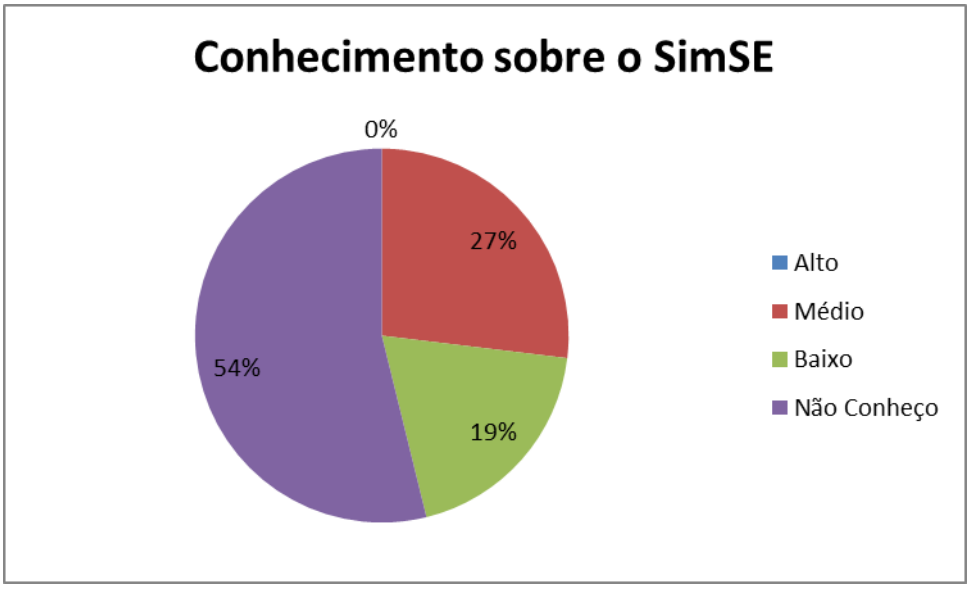

Figura 43 - Resultado: Conhecimento sobre o jogo SimSE

A Tabela 1 exibe o conhecimento dos alunos em relação a alguns softwares específicos. Observa-se que os alunos apresentam pouco conhecimento com relação às ferramentas de auxílio à Engenharia de Software e Gerência de Projetos, o que pode estar ligado à falta de vivência prática dos alunos com relação à área. Neste contexto o SPARSE pode introduzir tais conceitos, incentivando o uso de tais ferramentas.

Com relação à avaliação da ferramenta, para cada pergunta do questionário os alunos tiveram que atribuir uma nota entre 0 e 5. Os resultados são exibidos na Figura 44. Observando a figura, percebe-se a satisfação dos alunos com o uso da ferramenta. Além disso, os alunos sugerem algumas melhorias que podem ser feitas para tornar a ferramenta ainda mais completa.

Tabela 1 – Conhecimento de Softwares

Software	Conheço e utilizo todo dia	Conheço e utilizo frequentemente	Conheço e utilizo raramente	Conheço, mas nunca utilizei/ Não conheço
Ferramenta CASE (Rational Rose/StarUML/Outras)	*0*	*2*	*3*	*22*
Netbeans/ Eclipse	*14*	*10*	*2*	*1*
Microsoft Project	*1*	*2*	*3*	*21*
Requisite-Pro/ Outra ferramenta de gestão de requisitos	*0*	*0*	*1*	*26*
JUnit	*0*	*0*	*1*	*26*

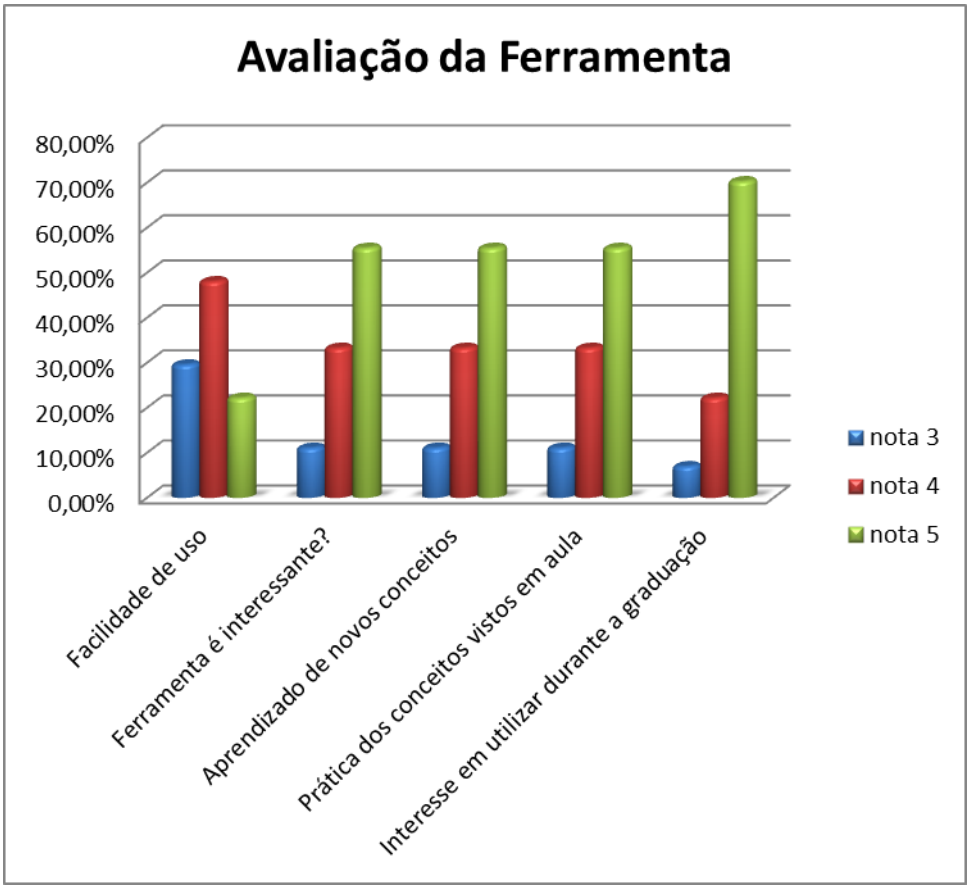

Figura 44 - Avaliação da Ferramenta

5
Conclusões

Este capítulo apresenta as conclusões relacionadas com este trabalho. O capítulo é organizado da seguinte forma: Na seção 5.1, são apresentadas as considerações finais referentes ao trabalho. Na seção 5.2 são apresentadas sugestões de trabalhos futuros, visando melhorias no jogo educativo SPARSE.

5.1 Considerações finais

Visando amenizar o problema do contraste entre métodos teóricos e natureza prática da área de Engenharia de Software, o SPARSE foi construído como proposta de um jogo educativo baseado em simulação para complementar o ensino teórico da área, introduzindo uma visão prática de ambientes de desenvolvimento de software.

Este trabalho apresenta a construção de uma nova versão do simulador do jogo SPARSE, com base na nova estrutura proposta no trabalho de Rodrigues (2010), com o objetivo de aproximar o jogador de modelos de ciclo de vida utilizados na prática, bem como possibilitar a simulação de qualquer processo modelado com base no meta-modelo SPEM. Após a avaliação do software através do uso do mesmo por alunos de graduação, pôde-se então validar a nova versão do simulador construída neste trabalho.

A partir dos resultados obtidos, conclui-se que a ferramenta é interessante e consegue fazer com que o aluno coloque em prática conceitos visto em aula, permitindo ainda o aprendizado de novos conceitos. Observa-se ainda que houve uma grande aceitação da ferramenta, de modo que a maioria dos alunos gostaria de utilizá-la durante o curso de Engenharia de Software para complementar os conhecimentos obtidos através das aulas teóricas (Figura 44).

A próxima seção apresenta os principais trabalhos futuros relacionados a esta pesquisa.

5.2 Trabalhos Futuros

Com base nos resultados obtidos com a avaliação do uso do jogo SPARSE, pôde-se observar uma boa aceitação dos alunos com relação à ferramenta. Além disso, foram sugeridas melhorias na ferramenta, além de novas idéias com relação às abordagens do jogo, que podem ser sugeridos como trabalhos futuros.

Para facilitar a importação de processos pelo SPARSE, sugere-se como trabalho futuro a criação de um *plugin* acoplado diretamente à ferramenta EPF Composer, facilitando a importação dos processos, uma vez que o arquivo gerado pela ferramenta é extenso e de difícil personalização.

Outra sugestão de trabalho futuro é a incorporação da metodologia ágil de gerência de projetos SCRUM (Schwaber, 2001) no SPASE. Isso permite ao aluno a vivência prática de uma metodologia pouco explorada em aulas teóricas e muito utilizada na prática. Para cada nova metodologia incorporada, trabalhos futuros relacionados à interface gráfica 3D são necessários, para que sejam criados novos cenários, típicos de cada metodologia.

Outra sugestão é adaptar o jogo ou criar uma nova versão, permitindo ao aluno um maior controle sobre o projeto sendo simulado, de modo que o mesmo possa, a partir de um dado modelo de ciclo de vida e um conjunto de recursos, fazer o planejamento das atividades em iterações e então promover e acompanhar a execução deste planejamento.

Uma sugestão dada por um dos alunos durante a avaliação da ferramenta é a possibilidade de armazenamento de estados do jogo. Nesta abordagem o aluno poderia salvar o estado atual do jogo para posteriormente poder retornar sem ter que reiniciar todas as atividades propostas.

6 Referências Bibliográficas

Barros, M. (2001) "Gerenciamento de Projetos Baseados em Cenários". Tese de PhD, UFRJ, Rio de Janeiro, Brasil.

Beholt, R. V; Figueiredo, R.S; Malavé C. O. *O uso de simulação no ensino de Engenharia,* 2001.

Bezerra, E. *Princípios de Análise e Projeto de Sistemas com UML*. Elsevier, 2007.

Catapan, A. H; Plínio, C. F; Souza, A. C; Thomé, Z. R. C; Cybis, W.D.A. *Ergonomia em software educacional: A possível integração entre usabilidade e aprendizagem.* In *II Workshop* sobre Fatores Humanos em Sistema Computacionais, Florianopolis – SC, 1999.

Drappa, A; Ludewig, J. *Simulation in Software Engineering Training*, ICSE 2000, ACM, 2000.

Gamma, E; Helm, R; Johnson, R; Vlissides, J. *Design Patterns: Elements of Reusable Object- Oriented Software*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1995.

Henderson-Sellers, B., Gonzalez-Perez, C., *A comparsion of four process metamodels and the creation of a new generic standard*, Univesity of Technology, Sidney, Australia. Publicado em: Information and Software Technology nº 47, , páginas 49-65, 2005.

Hoover, S. V; Perry, R.F. Simulation: a problem-solving approach. Reading, Massachusetts: Addison=Wesley Longman Publishing Co., Inc. Boston, EUA, 1989.

ISO 8879, *International Organization for Standardization* (ISO) *Information processing – Text and Office Systems – Standard Generalized Markup Language* (SGML). *First edition* - 1986-10-15 [Geneva]: *International Organization for Standardization*, 1986.

Madachy, R. J; Boehm, B. W. *Software Process Dynamics*. Early Drafft v.4, 1999. IEEE Computer Society Press, Los Alamitos, CA, EUA.

Moro, M; Braganholo, V; Dorneles, C; Duarte, D; Galante,R; Mello, R., *XML: Some Papers in a Haystack*, SIGMOD Record, 2009.

Navarro, E. *SimSE: A Software Engineering Simulation Environment for Software Process Education* , Universidade da Califórnia, Irvine, 2006.

OMG, *Software & System Process Engineering Metamodel Specification*, Versão 2.0, 2008.

OpenUP, *"Open Unified Process". Versão 1.5.0.4. 10-08* 2009, disponível em: http://epf.eclipse.org/wikis/openup/ data de acesso: 02/06/2011.

Paula Filho, W. *Engenharia de Software, métodos e padrões.* LTC, 2009.

Pfahl, D; Lainterbenger, O; Ruhe, G; Dorsch, J; Krivobokova, T. "Evaluating the learning effectiveness of using simulation in software project management education: results from a twice replicated experiment", 2004.

Ramsin, R., Paige, F. *Process-Centered Review of Object Oriented Software Development Methods*, ACM Computing Survey, Vol 40, N 1, artigo 3, 2008.

Rodrigues, L. "Implementação da Execução de Processos utilizando o meta-modelo SPEM no jogo educativo SPARSE", 2010.

Schwaber, K; Beedle, M. Agile Software Development with Scrum (Series in Agile Software Development), 2001.

Sommerville, I, *Engenharia de Software*, Oitava Edição, Pearson Addison-Wesley, 2007.

Souza, M. "Uma metodologia de predição estatística de projetos baseada em simulação", Dissertação de Mestrado, 2007.

Souza, M; Resende, R; Rodrigues, L; Rodrigues. A; Carvalho, F; Franco Junior, E; *SPARSE: Um Ambiente de Ensino e Aprendizado de Engenharia de Software Baseado em Jogos e Simulação*, SBIE, 2010.

W3C *Recommentadion, Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C*, 2008, disponível em: http://www.w3.org/TR/2008/REC-xml-20081126/ acessado em 02/06/2011.

7 Apêndices e Anexos

7.1 Apêndice A

QUESTIONÁRIO DE PERFIL

DADOS PESSOAIS

Idade

Período e curso:

Trabalha? ( ) Se. Atividades?_____

( )Não

EXPERIÊNCIA PROFISSIONAL

Com qual frequência utiliza o computador?

( ) Diariamente

( ) 3 vezes por semana

( ) 1 vez por semana

( ) Raramente

Para que atividade(s) utiliza o computador?

( ) Atividades pessoais (Internet, e-mail, etc.)

( ) Atividades escolares

( ) Atividades profissionais

( ) Outra(s). Quail(is)?_____

Qual o seu conhecimento sobre a disciplina de Engenharia de Software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre a área de Gerência de Projetos, no contexto da Engenharia de Software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre a área de Garantia da Qualidade, no contexto da Engenharia de Software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre modelos de ciclo de vida de software (cascata, espiral, iterativo-incremental, etc)?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre a disciplina de Requisitos do desenvolvimento de software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre as disciplinas de Análise e Desenho (Projeto/ Design) de Software do desenvolvimento de software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre a disciplina de Implementação de Software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre a disciplina de Testes do desenvolvimento de Software?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Qual o seu conhecimento sobre o modelo de processo OpenUp?

( ) Alto, utilizo os conceitos diariamente no meu trabalho ou na universidade

( ) Intermediário, utilizo os conceitos eventualmente no meu trabalho ou na universidade

( ) Baixo, já estudei o assunto, mas não apliquei/apliquei pouco na prática

( ) Nenhum, nunca ouvi falar

Dos softwares abaixo, marque com um "X" o seu conhecimento/utilização em cada um deles:

1. Tabela 2 – Conhecimento de Softwares

Software	Conheço e utilizo todo dia	Conheço e utilizo frequentemente	Conheço e utilizo raramente	Conheço, mas nunca utilizei/ Não conheço
Ferramenta CASE (Rational Rose/StarUML/Outras)				
Netbeans/ Eclipse				
Microsoft Project				
Requisite-Pro/ Outra ferramenta de gestão de requisitos				
JUnit				

OUTRAS PERGUNTAS

1. Você costuma utilizar o computador para entretenimento/aprendizado, através de jogos digitais?

( ) Sim, diariamente.

( ) Às vezes, jogo esporadicamente.

( ) Raramente, já joguei algumas vezes.
( ) Não

2. Qual o seu conhecimento sobre o SimSE (jogo para ensino de Engenharia de Software)?
( ) Alto
( ) Médio
( ) Baixo
( ) Não conheço

Agradeçemos pela contribuição!

7.2 Apêndice B

SPARSE – (**S**oftware **P**roject semi-**A**utomated **R**easoning tool for Software **E**ngineering)

Introdução

A área de Engenharia de Software é uma área essencialmente prática. No entanto, a maioria dos conceitos relacionados ao desenvolvimento de software é aprendida de maneira teórica em sala de aula. Hoje será possível praticar alguns desses conceitos teóricos da área de Engenharia de Software, utilizando para isso um jogo chamado SPARSE.

Neste jogo, você, jogador, desempenhará o papel de um gerente de projetos, responsável por conduzir o desenvolvimento do projeto em questão, através da alocação de desenvolvedores para a execução de tarefas.

O SPARSE proporciona uma visão prática da realidade que ocorre em organizações desenvolvedoras de software, envolvendo a ocorrência de eventos de eventos típicos tais como surgimento de novos requisitos durante o projeto, ausência de desenvolvedores em momentos críticos, dentre outros.

Orientações

No SPARSE, o jogador precisa finalizar o desenvolvimento de um dado projeto, realizando um conjunto de atividades pré-definidas do desenvolvimento de software, segundo um modelo de ciclo de vida (no caso específico desta versão do jogo, o modelo de ciclo de vida iterativo incremental seguido pelo modelo de processo do jogo que é uma extensão do Open-Up). O objetivo final é conseguir terminar o projeto com o menor número de defeitos, de acordo com o prazo e custo inicialmente definidos para o mesmo.

Após o jogo, as questões apresentadas abaixo deverão ser respondidas e entregues aos responsáveis pelo experimento, juntamente com o questionário de perfil.

Para um melhor resultado você deve jogar uma vez, inicialmente, para ter ideia geral do jogo e das regras, e depois jogar mais três vezes para responder as questões apresentadas abaixo.

Questionário

Parte A: Avaliação da ferramenta

Instruções: utilizar uma escala de 1 a 5, sendo 1 pior índice e 5 melhor índice

A ferramenta é fácil de usar?

Em sua opinião, a ferramenta é interessante?

A ferramenta consegue apresentar novos conceitos relacionados à teoria/aplicação da Engenharia de Software?

É possível praticar os conceitos aprendidos em sala de aula com esta ferramenta?

Você acharia interessante incorporar esta ferramenta na disciplina de Engenharia de Software?

Questões abertas:

O que você modificaria na ferramenta?

O que você mais gostou na ferramenta?

O que você menos gostou na ferramenta?

Em sua opinião, a ferramenta simula, de certa forma, a realidade do desenvolvimento de software? Justifique.

Parte B: Execução do Jogo

Instruções: as respostas devem ser completas e devem possuir justificativas (Responder no verso).

Descreva o modelo de processo apresentado no jogo em termos de fases, iterações, atividades, disciplinas, e tarefas.

O que representam os papéis no modelo de processo simulado? Existe impacto da alocação de empregados em papéis específicos na execução das atividades? Justifique.

O que são produtos de trabalho? De que forma eles interferem na execução das atividades?

O uso de ferramentas na execução das atividades do projeto possui algum impacto significativo? Justifique.

Qual o propósito das inspeções apresentadas no jogo? Qual o impacto da realização e não realização de tais atividades no projeto?

Considerando que o jogo já esteja em estágio avançado, qual o impacto da alocação de um empregado ocioso (ainda não alocado em atividades) desde o início do projeto?

Como você poderia melhorar a produtividade de um empregado? Descreve pelo menos três maneiras.

Identifique no jogo um fator positivo e um fator negativo que influenciou diretamente na produtividade da equipe de desenvolvimento.

Identifique no jogo um fator positivo e um fator negativo que influenciou diretamente na qualidade do produto final obtido.

Liste algumas atitudes tomadas durante o jogo que podem ter levado ao fracasso/sucesso do projeto.

Parte C: Pontuação do jogo

Instruções: preencha a tabela abaixo com a pontuação obtida em cada jogada

Tabela 3 – Pontuação

Jogo	Pontuação	Dinheiro utilizado	Número de defeitos na versão entregue ao cliente	Tempo utilizado
Jogo 1				
Jogo 2				
Jogo 3				
Jogo 4				