UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

SE•RPG 2.0: UMA NOVA VERSÃO DO SOFTWARE ENGINEERING-ROLEPLAYING GAME

FELIPE KOCHE AMBROSIO

BLUMENAU
2008

2008/1-13

FELIPE KOCHE AMBROSIO

SE•RPG 2.0: UMA NOVA VERSÃO DO SOFTWARE ENGINEERING-ROLEPLAYING GAME

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Fabiane Barreto Vavassori, Dra. - Orientador

BLUMENAU
2008

2008/1-13

SE•RPG 2.0: UMA NOVA VERSÃO DO SOFTWARE ENGINEERING-ROLEPLAYING GAME

Por

FELIPE KOCHE AMBROSIO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

	_____
Presidente:	Prof. Fabiane Barreto Vavassori Benitti, Dra. – Orientadora, FURB
	_____
Membro:	Prof. Marcel Hugo, M.Eng. – FURB
	_____
Membro:	Prof. Everaldo Artur Grahl, M.Eng. – FURB

Blumenau, 30 de junho de 2008

Dedico este trabalho à minha família, que sempre me apoiou de alguma forma, aos meus amigos que me incentivaram a continuar a caminhada, e à minha orientadora.

AGRADECIMENTOS

A Deus, por me dar forças e sabedoria para atingir os meus objetivos.

À minha noiva, pela sua paciência, compreensão e inteligência ao me ajudar nos momentos que precisei de sua ajuda.

À minha família, que sempre acreditou em mim e me deu forças para seguir em frente.

À minha orientadora, Fabiane Barreto Vavassori Benitti, por ter me confiado o desenvolvimento desse trabalho, pelo seu apoio, dedicação, e pela sua sabedoria em me instigar, cobrar e incentivar nos momentos que foi preciso.

Aos meus amigos que a todo o momento se preocupavam e se interessavam pelo meu trabalho.

É melhor lapidar até a exaustão o talento médio e determinado, do que tentar polir o diamante preguiçoso que não deseja polimento.

João Pedro Paes Leme

RESUMO

No presente trabalho é apresentada uma forma de aplicar na prática, os conhecimentos teóricos obtidos nas disciplinas que envolvem engenharia de software. O RPG (*Roleplaying Game*) fornece uma estrutura desde o ambiente de simulação até regras e modelos, que estimulam de modo descontraído e eficaz o desenvolvimento pedagógico de quem participa do jogo. A ferramenta propicia o desenvolvimento e refinamento dos conhecimentos, através da simulação de uma empresa de desenvolvimento de software, envolvendo o jogador em uma trama onde há personagens a serem interpretados, necessidades de gerenciamento de tempo, custo e eventuais situações não esperadas, bem como a avaliação do cliente quanto o resultado do produto.

Palavras-chave: Engenharia de software. Processo de desenvolvimento de software. *Role-playing games*.

ABSTRACT

In the present work is presented a form of applying in practice, the theoretical knowledges obtained in the disciplines which envolve software of engineering. The RPG (*Roleplaying game*) provides a structure from the environment of simulation up to rules and models, which stimulate in a casual and efficient way the pedagogic development of the participants. The tool favors the development and refinement of the knowledges, through the simulation of a software development enterprise, involving the player in a plot where are characters to be interpreted, necessities of time management, cost and eventual non-expected situations, as well as the client evaluation in compliance with the result of the product.

Key-words: Software engineering. Process of software development. *Roleplaying games*.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

CASE – *Computer aided software engineering*

RPG – *Role Playing Game*

XML – *eXtensible Markup Language*

W3C – *World Wide Web Consortium*

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

A globalização tem provocado uma mudança do paradigma pedagógico de ensinar, onde a utilização do computador e as novas técnicas de ensino auxiliam a troca de conhecimento entre aluno e professor. O uso de jogos que estimulam a criatividade e aumentam a motivação de aprender já são temas amplamente pesquisados e discutidos, conforme Oh e Hoek (2001, p. 1). As perspectivas das áreas do saber exigem que o modelo educacional esteja atento a uma realidade que se transforma a cada momento, onde a tecnologia está cada vez mais presente nas salas de aula.

Prince (2004, p. 2) comenta que o objetivo do paradigma educacional é fazer com que o aluno não apenas saiba fazer, mas principalmente compreender, e que formar um indivíduo para um mundo globalizado envolve a capacidade de utilização dos recursos diversificados da computação. Andrade (1997) enfatiza que o computador deve ser utilizado não apenas como ferramenta pedagógica, ou como substituto do professor, a proposta é criar ambientes enriquecidos com a tecnologia, em que o professor seja o dinamizador das experiências e atividades pedagógicas.

A utilização de softwares como ferramenta cognitiva na área da Educação, além de expor o conteúdo de uma maneira mais motivadora, auxilia tanto os alunos quanto os professores a atingirem seus objetivos educacionais, pois permite a resolução de problemas de forma dinâmica, o que estimula o desenvolvimento do senso crítico. Valente (1999, p. 71) afirma que todo software, desde processadores de texto a softwares multimídia e jogos, possui características que podem favorecer o crescimento do processo de construção do conhecimento.

Dentre os diversos tipos de jogos existentes, o *Roleplaying game* (RPG) é um que se destaca na área da educação. A dinâmica de um jogo de RPG, onde várias "personagens" interagem entre si, o torna uma excelente ferramenta educacional promovendo a socialização, cooperação, criatividade, interatividade e interdisciplinidade (KLIMICK, 2003).

Segundo Zuchi (2000, p. 84), o RPG como instrumento de ensino é uma ferramenta para simulações práticas em sala de aula, incentivando a criatividade, a motivação e um melhor entendimento sobre determinado assunto. Ele é adaptável a qualquer matéria ou conteúdos didáticos, para crianças, adolescentes ou adultos.

A disciplina de Engenharia de Software é tipicamente introduzida aos estudantes da seguinte maneira: a teoria geral é apresentada numa série de aulas e posta numa prática

(limitada) em um projeto em sala de aula. Apesar de parecer uma aproximação razoável, é insuficiente, por si só, para comunicar efetivamente a dinâmica fundamental do processo de desenvolvimento de software do mundo real (NAVARRO; HOEK, 2002).

Figueiredo et al. (2006, p. 1) afirma que apesar de o professor poder explicar assuntos relacionados à gerência de projetos, planejamento, documentação e teste de um software, problemas encontrados em grandes projetos, não são satisfatoriamente cobertos na prática em uma disciplina.

Neste contexto, Molléri (2006, p. 3) propôs utilizar o RPG como apoio ao ensino de Engenharia de Software, através de um jogo denominado Software Engineering-Roleplaying Game (SE•RPG), mais especificamente, abordando gerenciamento de projetos e ciclo de vida de desenvolvimento. O cenário do jogo é uma empresa de desenvolvimento de software fictícia, onde o jogador deverá optar por um objetivo apresentado pelos clientes.

O gerente de projeto (papel do aluno) deverá definir a maneira como o projeto será desenvolvido, o modelo de desenvolvimento, a linguagem de implementação e contratar as personagens que irão compor a sua equipe para a realização da tarefa. Durante o processo de desenvolvimento, é possível verificar as estatísticas de progresso de cada tarefa, distribuir atividades e analisar a produtividade de cada personagem, bem como contratar ou demitir funcionários. Com a conclusão do projeto é possível realizar a entrega do software ao cliente. Após isso, o jogo apresenta um resumo sobre o prazo, tempo de entrega e ações do jogador.

No entanto, nem todas situações vivenciadas em uma empresa de desenvolvimento de software são contempladas pelo SE•RPG. Sendo assim, propõe-se ampliar o escopo do SE-RPG, objetivando novas opções de escolha para as ações e desenvolvimento do projeto, uma nova característica das personagens e o acontecimento de efeitos aleatórios durante a realização do projeto, aproximando ainda mais as situações ocorridas durante o jogo com as que ocorrem no cotidiano de uma empresa de software.

1.1 OBJETIVOS DO TRABALHO

O objetivo desse trabalho é desenvolver melhorias no SE•RPG, visando ampliar suas funcionalidades e, conseqüentemente, seu potencial de ensino.

Os objetivos específicos do trabalho são:

a) re-escrever o jogo utilizando orientação a objetos;

b) incluir no SE•RPG a possibilidade de aquisição de ferramentas *Computer Aided Software Engineering* (CASE);
c) criar um novo atributo nas características das personagens, visando proporcionar "paradas" no trabalho;
d) disponibilizar o modelo de prototipação, que poderá ser escolhido pelo jogador para o desenvolvimento do projeto.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está divido em quatro capítulos. O primeiro capítulo apresenta a introdução sobre o tema abordado e os objetivos. O segundo capítulo apresenta a fundamentação teórica necessária para a realização deste trabalho, bem como é apresentado o conceito do RPG e de seu uso na educação, o uso do modelo de prototipação e ferramenta CASE. Ainda são apresentados alguns trabalhos correlatos. No terceiro capítulo é abordada a ambientação do cenário, seguindo pelos requisitos e regras do jogo, concluindo com aspectos técnicos e sobre a implementação do mesmo.

Por fim, no quarto capítulo são apresentadas as conclusões e sugestões para futuras extensões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos sobre RPG, desde a sua origem, os jogos de mesa tradicionais, o sistema de regras e a sua relação com o SE•RPG, que é detalhado na seção 2.6 SE•RPG. São relacionadas características do RPG que podem ser utilizadas em um ambiente de ensino, focando no uso de jogos de computador para auxiliar a aprendizagem. Também são mostrados conceitos de Engenharia de Software, seus métodos, teorias, ferramentas e o modelo de prototipação. Na última seção são examinadas características semelhantes dos trabalhos correlatos.

2.1 RPG

A sigla RPG vem da expressão inglesa *roleplaying game*, podendo ser traduzido como "jogo de interpretação", surgido no início da década de 1970, com o lançamento da edição Dungeons & Dragons, criado por Gary Gygax e Dave Arneson. Segundo Rodrigues (2004, p. 10), o RPG no Brasil, teve grande influência da literatura de Monteiro Lobato. Nele, os jogadores interpretam personagens fictícios em cenários variados. Esses cenários podem ter como ambientação a Idade Média, os anos 40, a época atual, a pré-história, ou até mesmo o futuro.

Veras e Santos (2004) caracterizam o RPG como um gênero de jogo, um amplo universo lúdico que abriga dezenas de jogos diferentes - todos unidos por um elemento em comum, a interpretação de uma personagem. O RPG pode ser comparado a um teatro improvisado, onde uma história vai sendo contada pelo narrador de acordo com as interações e ações dos jogadores.

Uma partida de RPG tradicional é composta por um narrador principal denominado mestre, narrador ou juiz e, interpretada por um grupo de jogadores. O mestre precisa conhecer cada detalhe sobre a ambientação e o sistema de regras. É ele quem vai apresentar ao grupo de jogadores uma história, uma aventura que contenha enigmas, simulações de combates, ou situações que exijam a interpretação da personagem. Segundo Bolzan (2003, p. 145), as ações dos jogadores são guiadas por um sistema de regras que indicam o resultado da ação para a personagem em questão dentro da situação em que ela se encontra. Um jogo de RPG costuma

ocorrer com um grupo de 4 a 7 jogadores, geralmente ao redor de uma mesa, o material necessário são apenas dados, lápis, papel, borracha e algum livro contendo as regras do jogo.

Rodrigues (2004) cita que para participar do jogo é necessário o jogador criar uma personagem, sendo que toda personagem possui habilidades e características específicas, como por exemplo, ter um conhecimento avançado em uma determinada ferramenta CASE, mas ser pouco hábil com a linguagem de programação Delphi.

Em um jogo tradicional de RPG o valor desses atributos é escolhido de forma aleatória através de dados, mas a qualquer momento o mestre do jogo pode intervir para evitar personagens excessivamente poderosos que possam desequilibrar o jogo (RODRIGUES, 2004). No SE•RPG, todas as personagens possuem as mesmas habilidades e atributos, bem como, com os seus valores já pré-definidos, mas sendo diferenciados para cada personagem.

É necessário também escolher uma classe que irá simbolizar sua personagem, como se fosse a sua profissão, exemplificando em um RPG ambientado em uma empresa de software, pode-se considerar personagens: estagiário, programador ou analista de sistemas. Sendo que os níveis das habilidades das personagens variam de acordo com a escolha da classe.

Com essa personagem o jogador irá tomar decisões, interagir com outros jogadores e essencialmente interpretar a sua personagem. Durante uma partida de RPG, a personagem do jogador é colocada no meio de situações desconhecidas e de diferentes dificuldades, criadas pelo narrador, e utilizando as suas habilidades, deve dar um jeito de passar por tais situações. O objetivo de uma aventura de RPG pode ser desde a conclusão de uma tarefa simples até o desenvolvimento de todo um projeto.

Andrade (2000), afirma que a caracterização do "mundo" e ambientação em que se passa uma partida de RPG é muito importante pelo fato de que, por exemplo, se a aventura for simulada na época medieval[1], com dragões, orcs, humanos e anões, fica incoerente um jogador querer construir um motor a gasolina ou desejar criar pólvora. No caso do SE•RPG, o jogo se passa na época atual, simulando uma empresa de desenvolvimento de software, sendo assim, o ambiente contém, mesas, telefone, impressora e computadores.

Para criar uma atmosfera de tensão, dificuldades e divertimento, é necessário que as regras sejam abrangentes, mas ao mesmo tempo flexíveis, permitindo que os jogadores visualizem os resultados de suas ações tomadas e percebam que devido a uma idéia brilhante ou a um conhecimento avançado em certa habilidade, o resultado foi melhor do que o esperado (RODRIGUES, 2004). As regras servem para organizar as ações das personagens

[1] A época medieval é inspirada na obra de Tolkien (escritor inglês do gênero fantasia, sendo a trilogia de O Senhor dos Anéis a sua obra mais famosa).

durante o jogo, determinando os limites do que este pode ou não realizar. Conforme o RPG foi se tornando popular, a empresa Wizards of the Coast (2000) sentiu a necessidade de uma padronização no sistema de regras. Dessa forma, foi criado o Sistema D20, que recebeu esse nome por ter como base um dado de 20 faces. No tópico 2.1.1 Sistema de regras D20 será tratado com maiores detalhes sobre esse sistema de regras.

2.1.1 Sistema de regras D20

O Sistema de regras D20 é um mecanismo de jogo para RPG, que possui esse nome devido à utilização de um dado de 20 faces (20 possibilidades), como base. Foi publicado em 2000 pela empresa Wizards of the Coast, criado por Jonathan Tweet, Monte Cook e Skip Williams, baseado no material original de E. Gary Gygax and Dave Arneson, Wizards of the Coast (2008).

Alguns livros de jogos da Wizards of the Coast como por exemplo, Dungeons & Dragons, Wheel of Time e Star Wars utilizam esse sistema de regras em sua ambientação. Por se tratar de um sistema de regras aberto (Anexo A), o sistema D20 é utilizado por outras editoras, mas essas devem seguir determinadas normas, que asseguram uma uniformidade ao sistema (SVALDI, 2003).

Svaldi (2003) ainda afirma que a idéia de tornar o sistema D20 aberto, foi da própria Wizards, que viu a chance de popularizar o seu sistema de regras e dominar o mercado mundial. Com isso a Wizards teve um aumento no número de jogadores utilizando o seu sistema, as editoras passaram a utilizar um sistema de regras conhecido pelo público e os jogadores passaram a ter uma variedade maior de jogos à sua disposição.

Segundo a Wizards (2008), o sistema consiste que ao executar uma ação qualquer, existe uma possibilidade de falha, então é gerado um valor aleatório de 1 a 20 ao qual é adicionado os modificadores relevantes.

Esses modificadores podem ser bônus (acrescentam ao valor do resultado), ou penalidades (subtraem do valor do resultado). Tais modificadores são calculados levando em consideração causas externas da ação da personagem ou a uma habilidade que ela possui, por exemplo, com um conhecimento avançado na ferramenta CASE Enterprise Architect, uma personagem irá receber bônus quando for realizar uma ação de analisar requisitos com a utilização da ferramenta citada, onde o oposto também é verdade, se uma personagem que não possui conhecimento na ferramenta e mesmo assim for utilizá-la, passará a receber uma

penalidade nos seus testes de tentativa de sucesso.

2.2 RPG NA EDUCAÇÃO

De acordo com Veras e Santos (2004), um grande problema para o educador é demonstrar a importância de conteúdos que não tenham aplicação prática imediata, mas que contribuam para formação geral do aluno, tornando-o mais capacitado para conteúdos mais complexos e para a própria vida. Tanaka (2001) cita que um dos assuntos mais discutidos em educação, atualmente, é como aumentar a motivação dos alunos, seja para evitar a evasão escolar, seja para melhorar os índices de aprendizagem de conteúdo, mensuráveis em avaliações. A complexidade e o acesso cada vez mais fácil e imediato às informações talvez sejam os responsáveis por um certo colapso da educação, onde os jovens, e cada vez mais as crianças, têm acesso ao mundo de forma mais direta e objetiva através dos meios de comunicação do que pelos métodos tradicionais de educação, afirma Andrade (1997).

Por se tratar de um jogo que estimula a socialização e a criatividade do aluno, o RPG é utilizado em escolas e universidades servindo como instrumento para auxiliar a aprendizagem e servindo como motivação ao aluno, onde o jogador vai interferindo e mudando a história que vai sendo contada podendo aprender ao mesmo tempo em que vai utilizando o que está sendo aprendido. Mas, segundo Marcatto (2005), o RPG só trará resultado se for preservada a sua forma original, como jogo, e não através da obrigatoriedade, trazido para dentro da sala, como uma outra matéria valendo nota, pois assim ele perde seu maior trunfo, que é a espontaneidade e a sensação, ainda que parcial, do jogador dominar sua própria história.

Andrade (1997) afirma que o RPG estimula o raciocínio globalizante, muito importante para os dias de hoje. Ele deixa para trás o raciocínio linear da maioria dos jogos para assimilar um raciocínio totalitarista, que tenta agrupar ao mesmo tempo pessoas, acontecimentos, ações e intenções. Além da diversão o RPG funciona como ferramenta para preparar o jovem a interagir na sociedade, tanto profissional quanto socialmente. Andrade ainda cita que algumas empresas utilizam o RPG para treinamento do pessoal, onde é possível resgatar valores morais e éticos.

Para realizar a atividade extra classe é formada uma equipe responsável em elaborar as sessões do jogo, as aventuras e determinar quem serão os narradores e os jogadores. Essas sessões são elaboradas com a orientação dos professores que especificam os elementos e

temas que devem ser abordados, sendo uma aventura diferente por matéria e série.

Por exemplo, um professor de História decide criar uma aventura envolvendo a Guerra do Paraguai, os interesses ingleses e argentinos e a situação política da época. Com o tema e a aventura definida são formados os grupos de jogadores, esses podendo ser totalmente independentes, agindo na mesma história, mas, em situações diferentes ou todos ao mesmo tempo.

Outra forma da utilização do RPG na educação é através do computador, podendo ser conduzido via e-mail, onde os jogadores recebem a mensagem inicial e através do correio eletrônico fazem a troca de mensagens descrevendo as suas ações e decisões tomadas em relação a sua personagem. A vantagem desse método é que um único mestre pode administrar um grupo maior de jogadores (ANDRADE, 1997). Segundo Bittencourt (2003) os mundos virtuais surgem dentro do contexto da cibercultura como uma nova alternativa para tratar à questão da aprendizagem.

De acordo com Junior e Nabais (2002 apud BITTENCOURT, 2003, p. 6), foi realizada uma pesquisa com jovens de 10 a 17 anos, de classe média de moradores da cidade do Rio de Janeiro com o objetivo de verificar quais as razões que tornam os jogos computadorizados tão atrativos e o que pensam sobre jogos educacionais. Para 85% dos jovens o que torna os jogos computadorizados atrativos é o desafio. Além disso, os jovens preferem ambientes imersivos com histórias ricas, jogos com qualidade gráfica e com algoritmo de inteligência artificial sofisticados. Destes jovens 68% consideram os jogos educativos ruins.

2.3 CONCEITOS DE ENGENHARIA DE SOFTWARE

O objetivo inicial da Engenharia de Software, para Soares (2005), é a utilização de teorias, métodos e ferramentas para auxiliar o desenvolvimento de software mais confiável e entregue de acordo com as restrições de custo e prazo previamente estabelecidos.

Para Pressman (2006), a Engenharia de Software é uma tecnologia em camadas, sendo que o alicerce é a camada de processo. O processo de engenharia de software é o que mantém unidas as camadas de tecnologia e permite o desenvolvimento racional e oportuno de softwares de computador.

Associado com o desenvolvimento de um software, é preciso também aplicar métodos, técnicas e ferramentas para o gerenciamento do processo de produção. Isto envolve

planejamento de custos e prazos, montagem da equipe e garantia da qualidade do produto e do processo. A Engenharia de Software visa à produção da documentação formal do produto, do processo, dos critérios da qualidade e dos manuais de usuários finais (LEITE 2007).

Pressman (2006) afirma que a Engenharia de Software possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade de produção.

A aplicação do paradigma de desenvolvimento da Engenharia de Software para Pressman (2006), é escolhido tendo como base o projeto, métodos, ferramentas a serem utilizadas e o produto a ser entregue. Em muitos casos os paradigmas podem e devem ser combinados de forma que as potencialidades de cada um possam ser obtidas em um único projeto, sendo que nesse caso, o conhecimento e a experiência são fatores decisivos para a escolha do paradigma ou junção de paradigmas mais apropriados ao projeto.

Segundo Sommervile (2003, p. 10), um método de Engenharia de Software é uma abordagem estruturada para o desenvolvimento de software, cujo objetivo é facilitar a produção de alta qualidade. Os métodos sofreram evoluções durante os anos, passando de análise estruturada, para orientados a função e orientados a objetos. Entre os principais paradigmas do Processo de Desenvolvimento de Software, citados por Pressman (2006, p. 38), tem-se os modelos Cascata (ou Clássico), Prototipação e Iterativo. A prototipação consiste no modelo a ser incorporado ao SE•RPG e portanto encontra-se detalhado na seção 2.4.

Para apoiar as atividades de processo de software como a engenharia de requisitos, o projeto, o desenvolvimento de programa e os testes, utiliza-se as ferramentas CASE (Computer Aided Software Engineering). Desenvolvimento de modelos gráficos de sistema, compreensão de um projeto utilizando dicionário de dados, geração de interfaces com usuários, depuração e tradução automatizada de programas, são exemplos de atividades que podem ser automatizadas utilizando-se CASE. A tecnologia CASE proporciona apoio ao processo de software e está disponível para a maioria das atividades de rotina no processo de software (SOMMERVILLLE, 2003, p. 53).

No SE•RPG o jogador tem a possibilidade de efetuar a aquisição de uma ou mais ferramentas CASE e atribuir essa ferramenta a uma personagem, melhorando assim a produtividade da atividade que aquela personagem está efetuando. Na seção 2.5 ferramenta CASE, é realizada uma análise mais específica sobre o objetivo de utilização de ferramentas CASE.

2.4 MODELO DE PROTOTIPAÇÃO

De acordo com Pressman (2006, p. 42), a prototipação é um modelo de processo que capacita o desenvolvedor a criar um modelo do software a ser implementado, podendo ser utilizada como uma técnica de análise e redução de riscos.

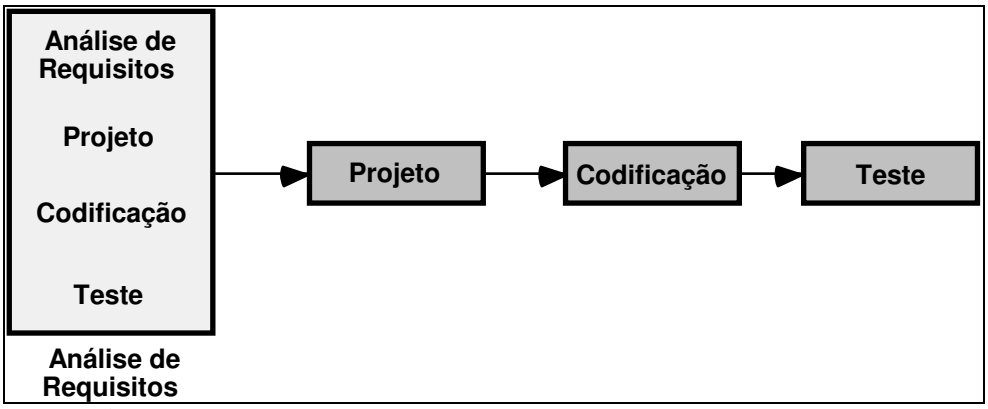
Indefinição dos objetivos gerais do software, falta da identificação detalhada dos requisitos de entrada, processamento ou saída, insegurança do desenvolvedor em relação a um algoritmo, compatibilidade com um determinado sistema operacional ou da interação homem/máquina, são alguns exemplos de situações onde o modelo de prototipação pode oferecer uma melhor abordagem ao desenvolvedor.

Sommerville (2003, p. 145), afirma que é quase impossível prever como um sistema afetará práticas de trabalho, como interagirá com outros sistemas e que operações dos usuários devem ser automatizadas.

Ainda segundo Sommerville (2003, p.145), um protótipo é uma versão inicial de um software, sendo utilizada para mostrar conceitos, experimentar opções de projetos, conhecer mais sobre problemas e suas possíveis soluções. O paradigma da prototipagem começa com a definição dos objetivos gerais entre o engenheiro de software e o cliente, realizando assim a análise dos requisitos.

Com isso são identificadas as necessidades conhecidas e levantadas as áreas que necessitam de mais definições. Em seguida é feita a modelagem de um projeto rápido, que é uma iteração de prototipagem planejada rapidamente.

O projeto rápido leva ao desenvolvimento de um protótipo, que é implantado e depois avaliado pelo cliente ou usuário. Caso haja necessidade, a etapa do *Feedback* é executada para refinar os requisitos do software (PRESSMAN, 2006, p. 42). A Figura 1 apresentada abaixo ilustra as etapas do modelo de prototipação.



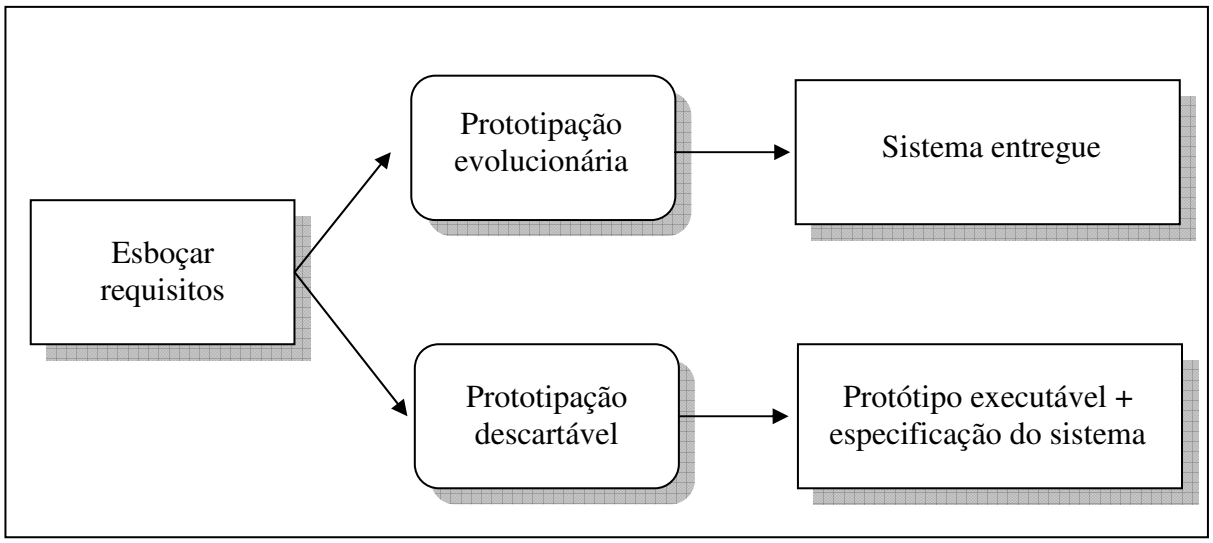
Fonte: Jalote (1997).

Figura 1 – Modelo da prototipação

Gordon e Bieman (1995 apud SOMMERVILLE, 2003, p. 146) relatam, a partir de estudo de 39 diferentes projetos de prototipação, que o uso da prototipação no processo de software proporciona a melhoria na facilidade de uso do sistema, maior aproximação do sistema com as necessidades dos usuários, melhoria na qualidade do projeto, facilidade de manutenção e esforço do desenvolvimento reduzido.

O protótipo pode ser desenvolvido considerando as seguintes formas: (i) em papel, fornecendo uma visão abstrata do sistema, (ii) incompleto, onde implementa algum subconjunto de funções exigidas ou (iii) protótipo final, sendo um software que executa uma função desejada, mas possui outras características que serão melhoradas ou implementadas (SOMMERVILLE, 2003, p. 146).

Existe dois tipos de modelos de prototipação, Descartável e Evolucionária, as quais são respectivamente detalhadas nas seções 2.4.1 Modelo Descartável e 2.4.2 Modelo Evolucionária, sendo representado na Figura 2.



Fonte: Sommerville (2003, p. 147).

Figura 2 – A prototipação evolucionária e a prototipação descartável

2.4.1 Modelo Descartável

O objetivo da prototipação descartável é validar ou derivar os requisitos do sistema. É necessário iniciar com os requisitos mais complexos e menos compreendidos, porque é necessário saber mais sobre eles. Os requisitos mais simples ou já compreendidos podem nunca ser prototipados (SOMMERVILLE, 2003).

O modelo se destina a ajudar no aprimoramento e na classificação da especificação do sistema, o protótipo é escrito, avaliado e modificado caso necessário, após a conclusão da especificação, o protótipo não tem mais utilidade e é descartado.

Para Sommerville (2003), os protótipos descartáveis têm duração muito curta, devendo ser possível modificá-los muito rapidamente durante o desenvolvimento, mas não é exigida uma fácil manutenção em longo prazo.

Assim, o baixo nível de desempenho e confiabilidade pode até ser aceitável em um protótipo descartável, desde que seja atendida a sua função principal de ajudar no entendimento dos requisitos.

2.4.2 Modelo Evolucionária

O objetivo da prototipação evolucionária é fornecer um sistema funcional aos usuários finais. Iniciando o sistema com os requisitos mais compreendidos e prioritários, os demais requisitos são implementados quando forem solicitados pelos usuários, (SOMMERVILLE, 2003).

O desenvolvimento de um protótipo do modelo Evolucionária é relativamente simples, sendo implementado os requisitos mais importantes do usuário e realizando as alterações e modificações à medida que novos requisitos vão sendo descobertos. No fim, ele se torna o sistema que foi requerido, não existindo nenhuma especificação detalhada do sistema e, em muitos casos pode não haver um documento formal de requisitos.

Mas segundo Sommerville (2003), os protótipos que evoluem para o sistema final, devem ser desenvolvidos com os mesmos padrões de qualidade de qualquer outro software. Devem possuir uma estrutura sólida, de modo que possam receber manutenção por muitos anos, ser confiáveis, eficientes e se adequar a padrões organizacionais relevantes.

2.5 FERRAMENTA CASE

A sigla CASE significa *Computer-Aided Software Engineering* (Engenharia de Software Auxiliada por Computador). O objetivo de uma ferramenta CASE é auxiliar os profissionais envolvidos na tarefa de produção de um sistema..

Sommerville (2003) classifica a Ferramenta CASE como diferentes tipos de programas utilizados para apoiar as atividades de processo de software, como análise de requisitos, a modelagem de sistema, depuração e testes. As ferramentas CASE também podem incluir um gerador de códigos, onde a partir de um modelo de sistema e uma orientação de processo, origina um código-fonte que fornece informações ao engenheiro de software sobre o que fazer no projeto.

As ferramentas destinadas a dar apoio a análise, as fases iniciais do processo e ao projeto é chamada de *Upper-CASE.* As ferramentas *Lower-CASE* são projetadas para dar apoio à implementação, testes, depuradores, sistemas de análise de programa, geradores de casos de testes e editores de programas.

A tecnologia CASE proporciona apoio ao processo de software pela automação de algumas atividades de processo e pelo fornecimento de informações sobre o software que está sendo desenvolvido. Segundo Sommerville (2003), os exemplos de atividades que podem ser automatizadas utilizando CASE são:

a) o desenvolvimento de modelos gráficos de sistemas de parte dos requisitos ou do projeto de software;
b) a compreensão de um projeto utilizando-se de um dicionário de dados que contém informações sobre entidades e tabelas;
c) geração de interface com usuários, através de uma descrição gráfica da interface;
d) depuração de programas, pelo fornecimento de informações sobre um programa em execução;
e) tradução automatizadas de programas, a partir de uma versão antiga de uma linguagem de programação, para uma versão mais recente.

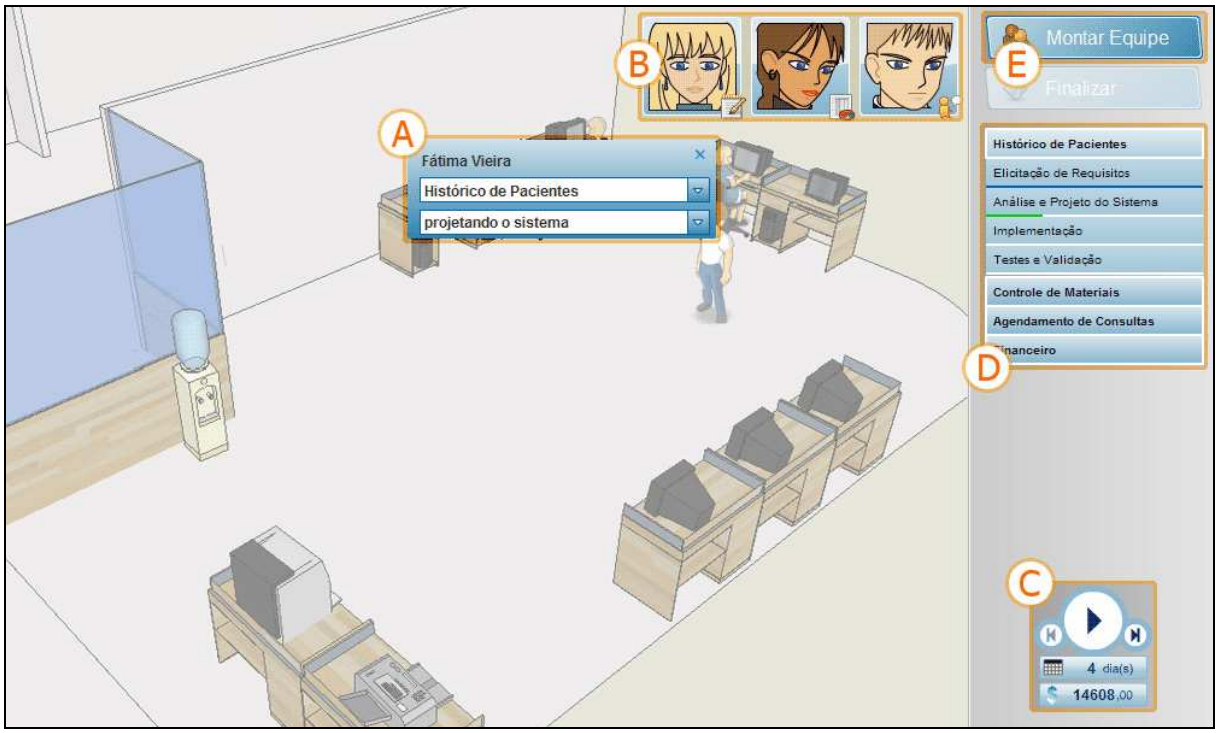
Algumas das ferramentas existentes no mercado são Rational Rose, System Architect, Enterprise Architect e TestComplet.

2.6 SE•RPG

De acordo com Molléri (2006, p. 74), o SE•RPG é uma ferramenta que simula o ambiente de desenvolvimento de software através de um jogo que tem por cenário uma empresa de desenvolvimento fictícia. O cenário apresenta 3 ambientes na interface gráfica: recepção e sala de reuniões, sala de direção e sala de produção (interface principal do jogo detalhada na Figura 3).

O jogo possui uma seqüência de passos que condizem com a realidade de uma empresa de software, onde primeiro o jogador deverá decidir qual projeto irá desenvolver. Em seguida é necessário definir o modelo de processo de desenvolvimento (escolhendo entre os modelos cascata e iterativo), linguagem de implementação e a contratação das personagens que irão participar da equipe para o desenvolvimento. Cada personagem possui habilidades próprias, e caberá ao jogador escolher aquelas que possuem as características mais adequadas para a realização do projeto.

Durante o jogo o jogador deverá atribuir tarefas para a sua equipe (Figura 3A), observando o modelo de processo de desenvolvimento escolhido. É indicado na figura de cada personagem a atividade que está sendo desempenhada (Figura 3B). É possível o jogador controlar a velocidade que o jogo irá se passar, alterando assim, o tempo necessário para a troca dos dias no jogo (Figura 3C). O progresso de cada atividade bem como o processo de desenvolvimento do projeto podem ser acompanhados pelo jogador (Figura 3D). Durante o desenvolvimento de um projeto há possibilidade do jogador alterar a sua equipe de personagens, podendo demitir ou contratar novas personagens (Figura 3E).



Fonte: Molléri (2006, p. 62).

Figura 3 - Ambiente da sala de desenvolvimento: (a) atribuição de tarefas; (b) representação das personagens; (c) controle de tempo; (d) progresso das atividades do projeto; (e) edição da equipe

Concluindo a etapa de codificação o projeto poderá ser entregue. Em seguida é apresentado um relatório com informações a respeito do cumprimento das metas (escopo, prazo e custo), a satisfação do cliente e se o modelo de processo escolhido para o desenvolvimento do projeto foi o correto.

O SE•RPG foi desenvolvido utilizando o sistema de regras D20 System como base para o mecanismo do jogo. Sendo que esse sistema consiste na geração de um número aleatório entre 1 e 20 que, irá definir o grau de dificuldade de determinada tarefa, o sucesso ou a falha na execução. Na seção 2.6.1 Regras SE•RPG, é explicado em maiores detalhes o sistema de regras do SE•RPG.

2.6.1 Regras SE•RPG

Conforme Molléri (2006), o RPG desenvolvido utiliza o sistema de regras D20 System como base para o mecanismo do jogo, onde com esse sistema, é definido o sucesso das ações das personagens quando executam suas atividades ou perícias.

As personagens do SE•RPG possuem perícias específicas e adequadas às tarefas que poderão desempenhar:

a) especificação de requisitos;
b) análise e projeto de sistemas;
c) codificação;
d) teste.

O valor do nível da perícia da personagem é somada diretamente no momento da execução da mesma, ou seja, quanto maior o nível que a personagem possui em determinada perícia, mais chances ela terá de conseguir um melhor resultado na execução de uma tarefa que esteja ligada diretamente com aquela perícia.

A perícia Codificação é subdivida em 4 outras perícias, sendo considerado como especializações na perícia Codificação, se referindo às linguagens de programação C++, Java, Pascal e PHP.

Além das perícias, cada personagem possui um salário, trata-se de um custo para mantê-lo trabalhando no projeto. A cada período de trabalho, o valor referente a 4 (quatro) horas é descontado do orçamento do projeto, sendo que, para efeitos de jogo tem a proporção equivalente a 10 segundos.

A cada período percorrido no desenvolvimento de uma etapa, a personagem que executa a tarefa realiza um teste da sua perícia para a atividade que está realizando contra a dificuldade de desenvolvimento do software. Esse teste é realizado utilizando as regras do D20 System. Caso o resultado do teste seja um valor igual ou maior do que a dificuldade da tarefa, então dividi-se o resultado por 100 para obter o esforço realizado neste período de tempo para o progresso da etapa. As falhas do teste significam que o processo não obteve progresso naquele período e nenhum valor é somado a atividade no momento.

Ao final de cada período o artefato pertinente à etapa é atualizado em um valor porcentual de acordo com o progresso, ou seja, em caso de sucesso, multiplica-se o valor obtido com o teste da perícia ao tamanho da etapa. Assim que o artefato atingir a marca de 100%, a etapa é concluída. O valor porcentual de progresso na etapa pode ser consultado pelo jogador a qualquer momento.

Na seqüência de tarefas do jogo, a importância do desenvolvimento de cada etapa é distribuído proporcionalmente, 10% para a Especificação de Requisitos, 30% para Análise e Projeto do Sistema, 40% para Codificação, 10% para Integração de Código; 5% para Elaboração do Plano de Testes e 5% para Execução do Plano de Testes.

Mais detalhes sobre o sistema de regras da primeira versão do SE•RPG podem ser encontrados em “Utilizando o RPG como ferramenta de aprendizado para o processo de desenvolvimento de software” (MOLLÉRI, 2006).

2.6.2 Objetivos SE•RPG

O SE•RPG tem por objetivo a conclusão do processo de desenvolvimento de um software de acordo com um projeto escolhido pelo jogador entre as opções existentes. O projeto possui algumas características que guiam o desenvolver do jogo, entre as quais pode-se citar:

a) dificuldade: mede o grau de complexidade do software a ser desenvolvido, podendo variar de 10 (mediano) a 20 (desafiador);
b) tamanho: é medido pela quantidade de *Use Case Points* (UCP) do projeto, podendo ser fracionado caso o projeto aceite o Modelo Iterativo de desenvolvimento;
c) orçamento: valor que a empresa possui para desenvolvimento do software e que pode ser gasto na manutenção de funcionários;
d) prazo: tempo para a conclusão do projeto, medido em dias úteis;
e) *deadline*: tolerância do cliente a um atraso do projeto.

Durante o desenvolvimento do jogo é calculado o tempo percorrido de cada etapa e, caso seja ultrapassado o prazo estipulado para o projeto, então é emitida uma mensagem avisando o jogador sobre esse fato, mas o jogo continua normalmente até o prazo estipulado pela *deadline*. Caso o tempo percorrido após o prazo estipulado iguala-se ao *deadline*, o jogador é avisado que o projeto foi cancelado e o jogo é encerrado automaticamente, apresentado os resultados obtidos até o momento.

O jogo também pode ser finalizado pelo usuário mediante a entrega do software, que pode ser feita desde que a codificação seja iniciada.

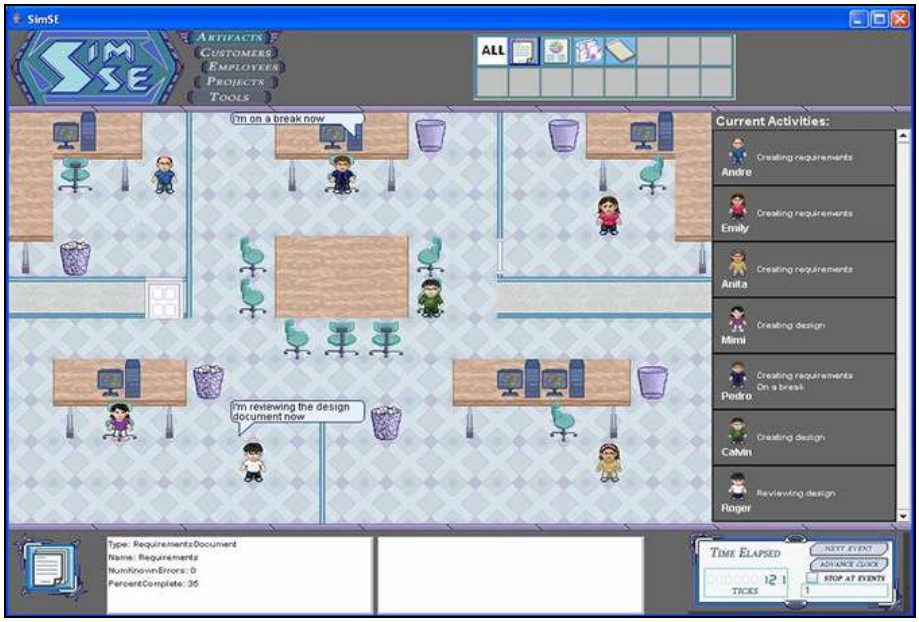
2.7 TRABALHOS CORRELATOS

Existem outros jogos voltados para o auxílio do ensino da disciplina de Engenharia de Software, sendo que a seguir são apresentados o SimSE, o SimulES e a primeira versão do SE•RPG 2.0.

2.7.1 SimSE

Segundo Navarro e Hoek (2002, p. 1), uma aproximação bem sucedida no processo de desenvolvimento é o ambiente simulado educacional para engenharia de software (SimSE). O SimSE conta com 3 componentes principais que são eles: o modelo de simulação contendo o cenário e elementos presentes no processo, a interface gráfica que interage com o usuário e um mecanismo de regras que rege o modelo, sendo que dessa forma o jogo retrata fielmente as situações envolvidas em um projeto real de uma empresa de software.

O jogo permite que o jogador assuma o papel de um gerente de projeto e desempenhe atividades como contratar e demitir programadores, delegar tarefas, monitorar o progresso das atividades, entre outras. A característica principal do SimSE fica por conta de sua interface gráfica, ilustrada na Figura 4, onde os artefatos, papéis, ferramentas, as escolhas do jogador e as falas das personagens são representadas de forma visual.



Fonte: Navarro e Hoek (2002, p. 32).

Figura 4 - Interface do jogo SimSE

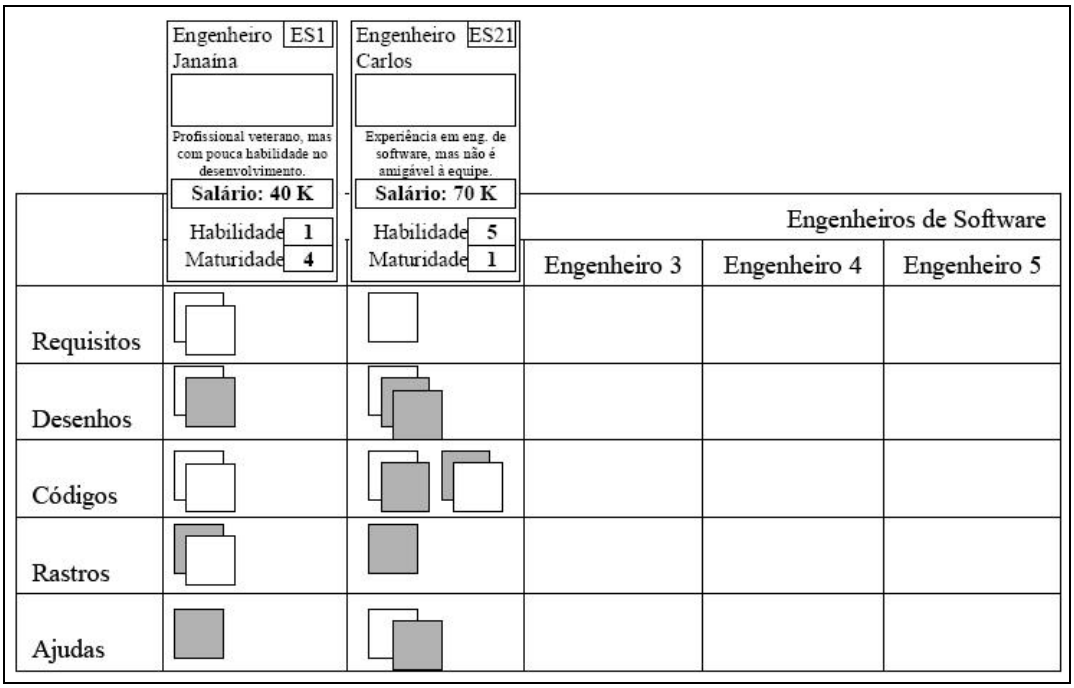
2.7.2 SimulES

Simulador de Uso da Engenharia de Software (SimulES) é um jogo educacional de cartas que simula o processo de desenvolvimento de softwares. De acordo com Figueiredo et al. (2006), este jogo permite que um estudante assuma o papel de gerente de projeto e depare

com problemas que não são bem cobertos em aulas tradicionais. O projeto é baseado em uma versão preliminar chamada *Problems and Programmers (PnP).*

Figueiredo et al. (2006) afirma que a motivação da pesquisa partiu após terem jogado o PnP e identificado problemas que podem ser classificado em duas categorias: as técnicas de ensino são limitadas e não permite aos jogadores adquirirem conhecimento externo ao jogo e muitos conceitos de engenharia de software utilizados são vagos ou obsoletos.

O SimulES pode ser jogado entre 4 e 8 jogadores, o objetivo é que os jogadores disputem para terminar um projeto de software e o vencedor será quem implantar o projeto primeiro. O jogo dispõe de alguns recursos que são: cartões de projetos, um tabuleiro, cartas e um dado. As cartas são o principal recurso do SimulES e, estão dividas em 4 tipos: problemas, conceitos, engenheiros de software e artefatos. A figura 5 mostra o tabuleiro com as cartas dispostas.



Fonte: Figueiredo et al. (2006).

Figura 5 - Tabuleiro de jogo com uma configuração de dois engenheiros

O objetivo do jogo é escolhido aleatoriamente através de um cartão de projetos que deve ficar visível a todos os jogadores. Durante a partida deve-se lançar um dado de 6 faces, e de acordo com o resultado, o jogador deverá pegar no monte de cartas, um número equivalente ao valor do dado. Essas cartas irão simular os problemas e desafios encontrados em um projeto de software de uma empresa.

O tipo de carta "Engenheiro de Software", é o principal recurso do jogador para progredir no jogo. Os engenheiros tem quatro opções de tarefas que são, construir artefatos, inspeção de artefatos, corrigir defeito e integrar artefatos em um módulo.

O jogo termina quando o primeiro jogador conseguir completar os componentes necessários de acordo com o seu cartão de projetos escolhido no início da partida. Nesse momento é feita uma validação por parte do cliente, onde alguns módulos são verificados se estão livres de problemas. O jogador somente é declarado vencedor se em nenhum dos módulos verificados for encontrado defeito.

3 DESENVOLVIMENTO

Este capítulo tem por objetivo detalhar as etapas para o desenvolvimento do jogo, que inicia com a descrição da ambientação do RPG, contendo o cenário, as personagens, a seqüência do jogo, também são apresentados os requisitos principais, tanto funcionais quanto não funcionais, seguindo com as especificações do jogo.

Na seção posterior é apresentada a implementação realizada no projeto, detalhando a operacionalidade e a jogabilidade do RPG, bem como as técnicas e regras envolvidas. Por fim, é apresentado uma discussão sobre os resultados do jogo.

3.1 AMBIENTAÇÃO

Conforme já citado na seção 2.1 RPG, todo jogo possui a sua caracterização do "mundo" em que se desenvolve a trama, havendo diferenças entre os jogos em relação ao cenário, características das personagens, cultura e costumes dos habitantes do jogo, bem como os recursos físicos envolvidos.Todos esse itens citados anteriormente possuem forte influência sobre uma partida de RPG, sendo assim as seções seguintes descrevem alguns itens característicos do SE•RPG.

3.1.1 Cenário

O jogo simula uma empresa de desenvolvimento de software, onde essa empresa é ilustrada graficamente através de 3 ambientes:

a) recepção e sala de reuniões: sala que possui a representação da secretária, a qual possui informações sobre a equipe de desenvolvimento, documentação e entrega do projeto final. Tem por finalidade tornar o jogo mais atrativo e realista;
b) sala de produção: lugar onde os membros contratados para a equipe de desenvolvimento estão distribuídos. É nessa sala que o jogador irá efetivamente controlar as atividades e o progresso de cada membro da equipe;
c) sala do diretor presidente: ambiente onde o gerente de projetos tem acesso a

relatórios, literatura sobre modelos de desenvolvimento e informações conceituais de gestão de projetos.

3.1.2 Personagens

O gerente de projetos, no caso o jogador, é a figura central do projeto e do jogo, pois é ele quem vai escolher o projeto a ser desenvolvido, montar a equipe, delegar atividades e realizar a entrega do projeto. Mas para que isso ocorra existe no cenário do jogo diversas personagens que o jogador pode ou não interagir, que são:

a) cliente: trata-se de uma personagem que não é controlada pelo jogador, sendo exibida apenas em determinados momentos do jogo, por exemplo, para as definições sobre escopo, requisitos do projeto e objetivos. É essa personagem que irá avaliar, no caso do desenvolvimento pelo modelo de prototipação, se o protótipo desenvolvido pelo jogador é aceito ou não. Por se tratar de uma personagem não-jogador, o Cliente não possui perícias ou atributos[2] dentro do sistema de regras do jogo;
b) equipe de desenvolvimento: São 8 personagens alocados na sala de produção onde pode ser atribuída as tarefas que cada um deve realizar. As personagens possuem diferentes papéis no RPG, sendo analista de requisitos, projetista, programador ou testador. Cada personagem possui perícias que caracteriza e define seu papel no jogo (detalhadas na seqüência a partir da figura 6);
c) gerente de projeto: personagem principal do jogo, a qual é comandada livremente pelo jogador, podendo interagir com a secretária ou com as outras personagens do desenvolvimento, circular pelas 3 salas do cenário, delegar as tarefas a serem executadas e verificar o status do projeto. Por não desempenhar tarefas no desenvolvimento do software, essa personagem não possui atributos e perícias;
d) secretária: personagem localizada na sala de recepção e reuniões, que tem por objetivo passar informações para o gerente de projeto ou possibilitar a troca da equipe de desenvolvimento. Esta personagem também não desempenha tarefas e não possui atributos e perícias.

[2] De acordo com o d20 System, perícias são habilidades ou conhecimentos que a personagem possui, sejam atividades físicas ou intelectuais, sendo que foi necessário um treinamento ou uma formação para aprendê-la. Ações costumeiras, como andar, falar ou correr por exemplo, que não exigem nenhum treinamento especial, não são consideras perícias (WIZARDS OF THE COAST 2000).

No SE•RPG mesmo tendo cargos diferentes as personagens possuem as mesmas perícias, o que varia de uma personagem para outra é o valor do nível que a personagem possui na perícia. Por exemplo, o nível da perícia na ferramenta Delphi da programadora júnior Juliana Dias é 6, sendo maior do que o nível 2 do estagiário Marco Coelho, portanto é correto dizer que a personagem Juliana Dias sabe utilizar a ferramenta Delphi melhor do que o Marco Coelho.

Todas as perícias existentes no SE•RPG utilizam como atributo chave a Inteligência, dessa forma foi omitido os outros atributos do D20 System (Força, Destreza, Constituição, Sabedoria e Carisma), de modo a simplificar a interface. Desta forma os itens de "a" a "g" referem-se a atributos da versão 1.0, sendo os demais acrescentados nessa nova versão:

a) habilidade na elicitação de requisitos;
b) facilidade de comunicação;
c) habilidade na elaboração de projetos;
d) conhecimento de padrões de software;
e) conhecimento de programação (diferentes linguagens de programação);
f) conhecimento de normas de qualidade;
g) atenção a detalhes;
h) uso da ferramenta RequisitePro;
i) uso da ferramenta EnterpriseArchitect;
j) uso da ferramenta RationalRose;
k) uso da ferramenta Delphi;
l) uso da ferramenta Eclipse;
m) uso da ferramenta NetBeans;
n) uso da ferramenta JUnit;
o) uso da ferramenta TestComplet.

Nesse projeto foi incluso também para cada personagem o atributo "*Stamina*", que representa a dedicação e o empenho que a personagem dedica ao projeto, assim quando a *Stamina* de uma personagem acaba, a mesma fica ausente do projeto por um certo período e uma mensagem é exibida ao jogador informando o motivo da ausência temporária da personagem.

As figuras 6, 7 e 8 apresentam respectivamente as personagens da classe Estagiário, Analista de Software Sênior e Programador Júnior e as suas respectivas perícias conforme são apresentadas ao jogador/ gerente de projetos. Para que sejam executados os testes de sucesso das personagens utilizando o d20System, todas as perícias são mapeadas quantitativamente

em um arquivo XML (cuja estrutura é apresentada na seção 3.3.4 XML Schema, e a listagem consta no Apêndice A), por exemplo, se na descrição da personagem está informado que a mesma possui um bom conhecimento sobre elicitação de requisitos, isso significa que no arquivo XML, na perícia Elicitação de requisitos, para essa personagem está gravado o valor numérico entre 2 ou 3.

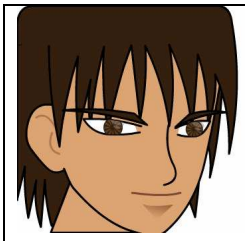
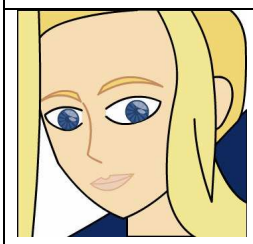
	**Marco Coelho** Cargo: Estagiário Possui dificuldades de comunicação e um conhecimento bom de padrões de software, mas tem experiência de programação em Java por 2 anos e Pascal por 1 ano, pouca atenção aos detalhes. Possui um bom conhecimento nas ferramentas RationalRose e NetBeans e já fez algumas vídeo aulas em EnterpriseArchitect e Eclipse. Possui muita disposição e vontade para trabalhar.
	**Cristiane Franca** Cargo: Estagiária É boa com a elicitação de requisitos, mas possui pouco conhecimento de padrões de software, tem um conhecimento bom das linguagens de programação Java e Pascal, uma ótima atenção aos detalhes. Possui um ótimo conhecimento na ferramenta RequisitePro, já fez vários cursos em Delphi e JUnit, adquirindo uma boa experiência, possui uma noção de Eclipse e TestComplet. Tem muita dedicação e disposição para o trabalho.

Figura 6 – Descrição das personagens da classe Estagiária

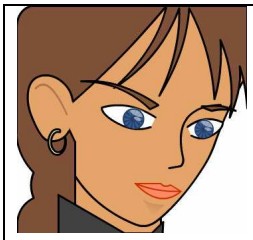
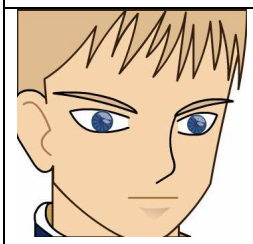
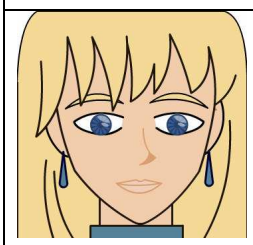
	**Fátima Vieira** Cargo: Analista de Software Sênior Possui uma facilidade de comunicação extraordinária e um ótimo conhecimento de padrões de desenvolvimento de software, com 3 anos de experiência em programação C++ e mais um semestre em PHP, possui um bom conhecimento de normas de qualidade. Possui uma experiência extraordinária na ferramenta RequisitePro, um ótimo conhecimento nas ferramentas EnterpriseArchitect e RationalRose, um bom conhecimento em NetBeans, JUnit e TestComplet e, um razoável conhecimento na ferramenta de desenvolvimento Eclipse. Possui uma razoável disposição para o trabalho.
	**Paulo de Mello** Cargo: Analista de Software Sênior Possui uma boa capacidade para elicitação de requisitos e ótima elaboração de projeto, além de um ótimo conhecimento da linguagem de programação C++, e experiência de 4 anos com PHP, extraordinária atenção aos detalhes. Possui um bom conhecimento nas ferramentas RequisitePro e RationalRose, uma ótima habilidade em EnterpriseArchitect, um bom conhecimento em Eclipse, apenas uma noção de JUnit, mas é extremamente competente com as ferramentas NetBeans e TestComplet. Possui muita disposição para o trabalho.
	**Alessandra Gonçalves** Cargo: Analista de Software Sênior Tem uma boa facilidade de comunicação e experiência de 4 anos na elaboração de projetos de software, conhecimento bom com as linguagens de programação Java e Pascal, uma boa atenção aos detalhes. Possui ótimo conhecimento nas ferramentas RequisitePro e RationalRose, é extraordinariamente experiente na ferramenta EnterpriseArchitect, possui um bom conhecimento em Delphi e TestComplet, e já fez alguns mini-cursos de NetBeans e Junit. Não possui muita disposição para o trabalho.

Figura 7 – Descrição das personagens da classe Analista de Software Sênior

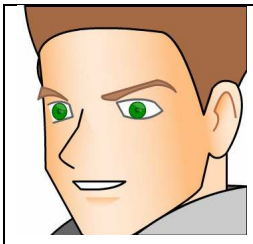
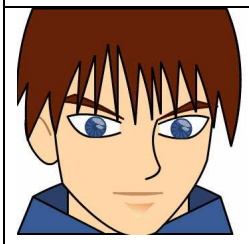
	**Fabrício de Souza** Cargo: Programador Júnior É bom com a elicitação de requisitos, possui 2 anos de experiência na elaboração de projetos, um conhecimento bom das linguagens de programação C++ e Delphi, um conhecimento razoável de normas de qualidade. Tem um bom conhecimento das ferramentas RequisitePro, EnterpriseArchitect, Delphi e NetBeans, um conhecimento razoável do JUnit e Eclipse e uma ótima experiência na ferramenta RationalRose. Possui boa disposição para o trabalho.
	**Afonso Trevi** Cargo: Programador Júnior Possui uma habilidade ruim com a elicitação dos requisitos e um conhecimento razoável de padrões de software, mas trabalhou com programação em Java por 3 anos e PHP por 2 anos, um ótimo conhecimento de normas de qualidade. Possui um bom conhecimento nas ferramentas Delphi, Eclipse e TestComplet além de uma ótima habilidade em NetBeans e Junit. Possui boa disposição para o trabalho.
	**Juliane Dias** Cargo: Programador Júnior Possui uma boa habilidade com a elicitação de requisitos, mas um conhecimento ruim dos padrões de software, experiência de 2 anos com programação PHP e outros 4 anos com Pascal, uma ótima atenção aos detalhes. Possui um bom conhecimento nas ferramentas RequisitePro, NetBeans e Junit além de um domínio extraordinário em Delphi e um ótimo conhecimento em TestComplet. Possui uma boa disposição para o trabalho.

Figura 8 – Descrição das personagens da classe Programador Júnior

3.1.3 Sequência do jogo

Por simular uma empresa de desenvolvimento de software, o processo do jogo possui uma seqüência de passos definidos através de um diagrama de atividades em Molléri (2006. p. 43). Onde tal sequência se assemelha à realidade existente nas empresas reais.

No início do jogo o jogador decide em qual projeto irá trabalhar, sendo que cada projeto possui um prazo, orçamento e cliente diferente que está solicitando o mesmo. Os projetos disponíveis para a escolha do jogador são apresentados conforme a figura 9.

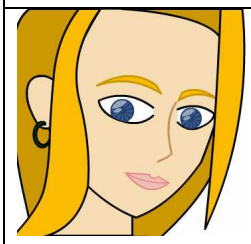
	**Controle de Estoque para Loja de Calçados:** Desejo um pequeno sistema para automatizar o controle do estoque de minha loja de calçados. Trata-se de uma loja familiar e pequena, com poucos funcionários. Observação: Os usuários sabem exatamente o que desejam, sendo que os requisitos estão bem definidos.
	**Sistema Gerencial para Clínica Odontológica:** Gostaria de automatizar as atividades de uma clínica odontológica onde trabalham diversos profissionais (tanto dentistas quanto secretárias). A aplicação pode ser considerada de médio porte. Observação: Os usuários conhecem bem os procedimentos da clínica, mas não há consenso sobre os requisitos do software.
	**Sistema de comércio eletrônico para Supermercado:** Um supermercado da região deseja expandir sua atuação utilizando como parte de sua estratégia a Internet. Para isto, deseja que seja desenvolvido um e-mercado, ou seja, o conceito de comércio eletrônico aplicado para o ramo de supermercados. Todas as atividades cotidianas de um supermercado já estão automatizadas através de um sistema desenvolvido pelo núcleo de informática do supermercado, tal como: controle do estoque, frente de caixa, logística, financeiro. No entanto, a equipe interna está sem disponibilidade de assumir novos projetos e não deseja contratar novos profissionais, sendo que este projeto Web será terceirizado tendo o acompanhamento de um profissional da TI do supermercado. O principal aspecto que foi considerado fundamental pelos gestores trata-se da interface do sistema. Ou seja, os gestores do supermercado desejam uma interface de fácil navegação, leve, que reforce a imagem do supermercado junto aos seus clientes, tendo um projeto visual que amplie as vendas por cliente, através de ações de marketing. Este projeto web deverá estar completamente integrado aos demais sistemas já existentes, como, a logística, financeiro e controle de estoque.

Figura 9 – Projetos disponíveis no jogo

Após a escolha do projeto a ser desenvolvido, o gerente de projeto deve definir o modelo de desenvolvimento que será utilizado, onde os existentes são o Modelo Cascata[3], Modelo Iterativo[4] e Modelo de Prototipação. Em seguida deve escolher a linguagem de programação a ser utilizada no projeto, após essas escolhas deve ser montada a equipe que irá desenvolver o software. Com a equipe formada o gerente de projetos pode iniciar as atividades de desenvolvimento do projeto.

O gerente de projeto pode a qualquer momento verificar os relatórios de produção de cada personagem ou se deslocar à sala da direção para ter acesso à documentação de ajuda do jogo, além de ser possível à verificação de datas e relatórios com a personagem da secretária.

No decorrer do jogo algumas ações ocorridas requerem uma atenção maior do gerente de projetos, como a finalização de uma atividade ou conclusão do desenvolvimento de um módulo e a saída momentânea e inesperada de alguma personagem em virtude do atributo *stamina*. O jogador poderá também a qualquer momento, efetuar a aquisição de uma ou mais ferramentas CASE dentre as oito[5] existentes disponíveis para a compra.

É possível comprar apenas uma licença de cada ferramenta e, cada personagem poderá estar utilizando apenas uma ferramenta por vez. A partir do momento que uma personagem está utilizando uma determinada ferramenta CASE, essa ferramenta não estará mais disponível para outras personagens até que seja liberado o seu uso. Se uma personagem estava utilizando uma ferramenta no momento em que teve uma ausência no jogo em virtude da *stamina*, automaticamente essa ferramenta estará livre para uso.

Uma ferramenta pode ser equipada em qualquer personagem, mas para que o bônus[6] que a ferramenta possui seja aplicado integralmente no jogo, é necessário que a personagem possua pelo menos nível 1 na perícia da ferramenta em questão e que a ferramenta seja utilizada na etapa correta de acordo com a sua especificação.

Assim que a etapa de codificação for concluída já é possível realizar a entrega do software para o cliente, após esse processo, o jogo apresenta um relatório informando ao jogador sobre as metas atingidas, satisfação do cliente, tempo e recursos utilizado. Além da informação sobre a correta utilização do modelo de processo de desenvolvimento escolhido.

---

[3] O diagrama de atividade do modelo Cascata pode ser encontrado em Molléri (2006, p. 45).
[4] O diagrama de atividade do modelo Iterativo pode ser encontrado em Molléri (2006, p. 45)
[5] As ferramentas abordadas no jogo foram selecionadas tendo em vista as disciplinas atualmente ministradas nos curso de Ciências da Computação e Sistemas de Informação. No entanto, podem ser alteradas através da edição do arquivo XML.
[6] Para o d20 System denomina-se modificador o número que se aplica na rolagem do dado quando a personagem tenta fazer alguma ação relacionada com a sua habilidade. Também pode ser aplicado modificadores independente da rolagem do dado. Um modificador positivo é chamado de bônus e um modificador negativo é chamado de penalização (WIZARDS OF THE COAST 2000).

3.2 REQUISITOS DO SE•RPG

Os requisitos, segundo Sommervile (2003, p. 83), são declarações de funções que o sistema deve fornecer, de que maneira deve reagir a entradas específicas e como deve se comportar em determinadas situações.

A seguir são apresentados os requisito funcionais (Quadro 1) que foram re-implementados na primeira versão do SE•RPG e os novo requisitos funcionais (Quadro 2) implementados neste trabalho

REQUISITOS FUNCIONAIS RE-IMPLEMENTADOS	
RF01	Permitir ao jogador selecionar o projeto a ser desenvolvido, o modelo de processo e a linguagem de desenvolvimento, considerando os processos cascata e iterativo.
RF02	Permitir ao jogador definir a sua equipe de desenvolvimento.
RF03	Permitir a atribuição de tarefas aos membros da equipe.
RF04	Permitir ao jogador controlar o "fator tempo" utilizado no jogo.
RF05	Permitir ao jogador acompanhar a produtividade da equipe de desenvolvimento.
RF06	Permitir ao jogador acompanhar o progresso das atividades do projeto.
RF07	Apresentar o orçamento do projeto atualizando-o diariamente durante a execução do jogo.
RF08	Permitir ao jogador finalizar o projeto e neste momento o sistema deve apresentar o resultado de sua atuação.

Quadro 1 – Requisitos funcionais re-implementados

NOVOS REQUISITOS FUNCIONAIS	
RF01	Disponibilizar o modelo de processo de desenvolvimento de prototipação.
RF02	Possibilitar no jogo adotar ferramentas CASE para auxiliar no desenvolvimento do projeto.

Quadro 2 – Novos requisitos funcionais

Requisitos não funcionais são restrições sobre os serviços ou as funções oferecidos pelo sistema (SOMMERVILLE, 2003).

REQUISITOS NÃO FUNCIONAIS	
RNF01	Implementar o jogo no ambiente Flash 8 utilizando orientação a objeto com a linguagem Action Script.
RNF02	Simular um ambiente de desenvolvimento (com interface gráfica), semelhante a uma empresa real.
RNF03	O sistema de regras do RPG deve ser baseado no conjunto de regras do Sistema D20 System.
RNF04	As informações dos projetos e personagens devem ser armazenadas em arquivos XML.

Quadro 3 – Requisitos não funcionais

As regras do jogo definem a mecânica de como deve progredir as ações tomadas pelo jogador, ela tem influência direta nos resultados e no decorrer do jogo. Nos quadro 4, 5 e 6 são apresentadas todas as regras do jogo, estando divididas respectivamente da seguinte forma: as regras mantidas e inalteradas da primeira versão do jogo, as regras mantidas mas que foram atualizadas para a segunda versão do jogo e, as novas regras que foram inclusas nesse projeto.

REGRAS DO JOGO – ORIGINAL	
RJ01	O progresso do jogo terá proporção equivalente a 10 segundos para cada período de 4 horas de desenvolvimento, sendo cada período desse considerado um ciclo, portanto cada dia de trabalho possui 2 ciclos.
RJ02	A produtividade de cada membro da equipe pode ser calculada através da Equação produtividade do período = valor randômico (d20) + perícia da personagem + bônus da ferramenta utilizada (definida na regra RJ08).
RJ03	O uso de mais de uma personagem na mesma tarefa, fornece um auxílio no resultado do teste do d20System para as personagens envolvidas na mesma tarefa, tornando assim mais fácil o sucesso no teste para verificar a progressão na atividade.

Quadro 4 – Regras do jogo mantidas da primeira versão

REGRAS DO JOGO – ALTERADAS		
RJ04	Para o desenvolvimento dos projetos, o RPG deve considerar as características dos modelos em Cascata, Iterativo ou Prototipação, conforme a escolha do jogador.	Alterada: Incluso o modelo de Prototipação no projeto e na regra.
RJ05	O progresso de cada atividade durante o período é calculada conforme a equação: SE (dificuldade da tarefa <= valor randômico (d20) + perícia da personagem na tarefa + bônus da ferramenta utilizada) ENTÃO progresso = valor randômico (d20) + perícia da personagem na tarefa + bônus ferramenta.	Alterada: Incluso o bônus da ferramenta utilizada na etapa.
RJ06	O orçamento será atualizado segundo o cálculo demonstrado na equação: orçamento atual = orçamento atual – custo diário de cada membro da equipe – custo de aquisição de uma ferramenta.	Alterada: Ao realizar a compra de uma ferramenta o orçamento também será atualizado.

Quadro 5 – Regras do jogo alteradas

NOVAS REGRAS DO JOGO	
RJ07	A aprovação ou não da construção de um protótipo no modelo de prototipação irá depender da média de sucessos nos testes das personagens nas 3 fases relacionadas com o protótipo, com a seguinte equação: %Sucesso = (Quantidade de sucessos nos testes de progressão da fase * 100) / Quantidade de ciclos utilizados por cada personagem envolvido. SE %Sucesso > 60 ENTÃO Aprovado. O protótipo ficou perfeito, pode prosseguir o projeto (O projeto segue normalmente) SENÃO SE %Sucesso > 30 E %Sucesso < 61 ENTÃO Aceito. O protótipo ficou bom, falta apenas alguns ajustes que poderão ser realizados durante projeto (O projeto segue normalmente). SENÃO Reprovado. Não era esse o protótipo que eu desejava, por favor realize os ajustes necessários (O progresso das fases Elicitação de requisitos, Projeto rápido e Construção do protótipo é reduzido pela metade e o jogador deverá complementar novamente essas fases para ser possível seguir em frente no projeto).
RJ08	Cada ferramenta pode acrescentar um bônus no teste e no progresso de uma atividade de acordo com a seguinte regra: SE utilizou uma ferramenta na fase certa e a personagem possui habilidade na ferramenta ENTÃO bônus total da ferramenta. SE utilizou ferramenta na fase certa e a personagem não possui habilidade na ferramenta. ENTÃO a ferramenta irá acrescentar apenas a metade do bônus para a personagem. SE utilizou ferramenta na fase incorreta ENTÃO a personagem irá receber uma penalização de -10 no resultado do seu teste final.
RJ09	Ao zerar a *stamina* a personagem irá parar o seu desenvolvimento e irá se ausentar do jogo durante 8 ciclos, sendo essa a quantidade de ciclos para que haja o retorno da personagem para o projeto. O jogo irá apresentar uma mensagem informando o motivo da ausência da personagem para que o gerente de projeto tenha conhecimento.
RJ10	Incluir atributo de energia às personagens, visando proporcionar "paradas" no trabalho.

Quadro 6 – Regras do jogo criadas nesse projeto

3.3 ESPECIFICAÇÃO

A especificação desta ferramenta foi realizada utilizando diagramas da *Unified Model Language* (UML). Os diagramas foram elaborados na ferramenta Enterprise Architect, sendo um diagrama de atividades, que demonstra a seqüência do processo das etapas do jogo, quando utilizado o modelo de prototipação para o desenvolvimento do software. Também é apresentado um diagrama de classes, que explica como as classes estão estruturadas no sistema, um diagrama de componentes e o XML Schema contendo a estrutura das personagens e das ferramentas.

3.3.1 Diagrama de atividades

Os diagramas de atividade são utilizados para documentar o fluxo de uma atividade ou comportamento. Assim, são conceitualmente muito similares a um fluxograma (PILONE; PITMAN, 2006, p. 6).

As atividades que norteiam o modelo de prototipação, bem como os passos realizados pelo jogador que definem as próximas etapas após a avaliação do protótipo pelo cliente, são apresentadas na Figura 10. Os diagramas de atividades que ilustram os modelos cascata e iterativo podem ser encontrados em Molléri (2006, p. 45).

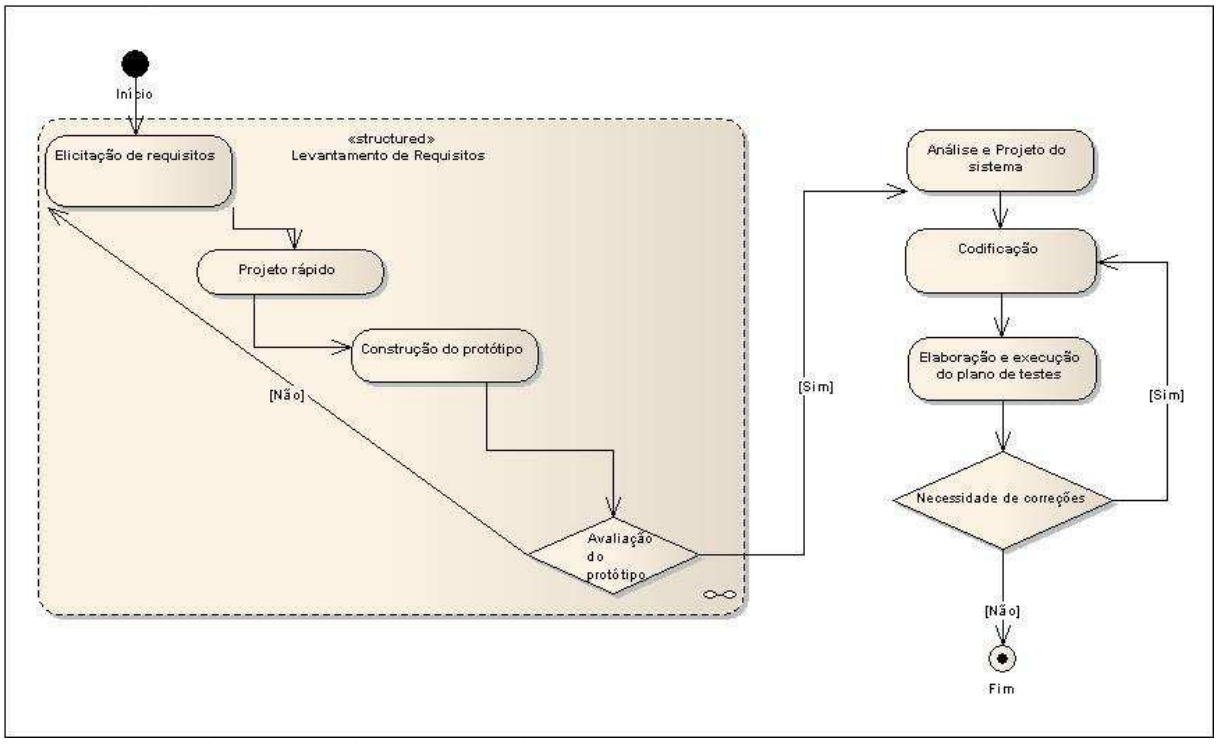


Figura 10 – Digrama de atividades do modelo de processo Prototipação

3.3.2 Diagrama de classes

O diagrama de classes apresenta uma visão de como as classes estão estruturadas e relacionadas. Pilone e Pitman (2006, p.5) destacam que os diagramas de classes utilizam classes e interfaces para documentar detalhes sobre as entidades que formam o sistema e as relações estáticas entre elas. O diagrama de classes é apresentado na figura 11.

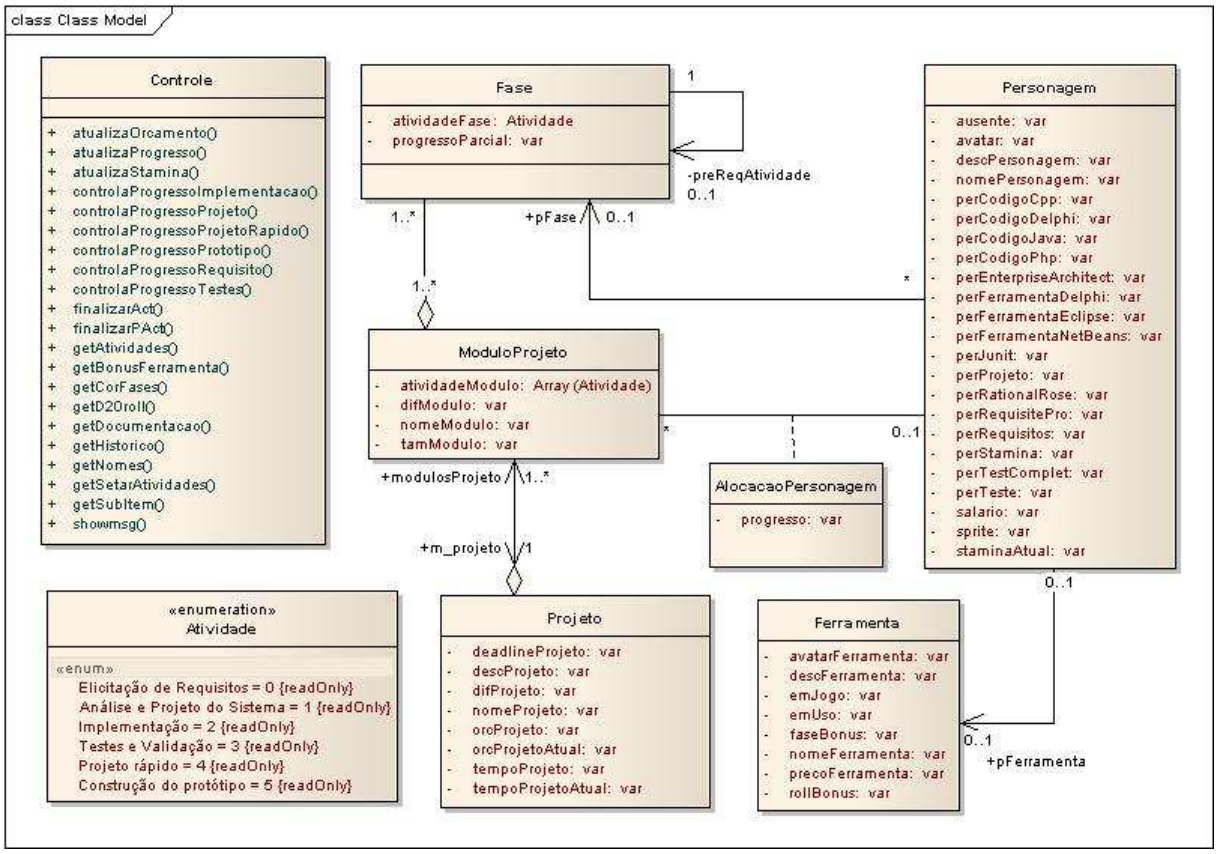


Figura 11 – Digrama de classes do RPG

Na seqüência uma breve explicação contendo a responsabilidade de cada classe. A descrição dos atributos do diagrama de classe podem ser consultados no Apêndice C:

a) `controle`: podendo ser considerado o "cérebro" do jogo, é nessa classe que são realizados todos os cálculos das ações das personagens, o controle da progressão dos módulos, orçamento e tempo do jogo, ou seja, todas as regras do RPG estão implementadas nessa classe;

b) `fase`: classe de domínio que representa as fases executadas no jogo;

c) `ferramenta`: classe que após a leitura do XML representa as informações referentes às ferramentas CASE;

d) `moduloProjeto`: classe do domínio dos módulos dos projetos. Essa classe faz a definição de quais fases estão presentes no jogo de acordo com o modelo de desenvolvimento escolhido pelo jogador;

e) `personagem`: classe de domínio dos dados das personagens, que são armazenados após a leitura do XML;

f) `projeto`: classe de domínio dos dados dos projetos, como nome, descrição, tempo e custo. Os dados são obtidos a partir do XML;

g) `atividades`: classe que contém os tipos de fases existentes no jogo;

h) `alocacaoPersonagem`: classe que contém os módulos que estão sendo

desenvolvidos com as respectivas personagens que estão realizando a implementação.

3.3.3 Diagrama de componentes

O diagrama de componentes, segundo Melo (2004), mostra as dependências entre componentes de software, apresentando suas interfaces.

Os diagramas de componentes mostram a estrutura estática do modelo de implementação, sendo apresentados como um conjunto de elementos do modelo estático, como componentes, subsistemas, e seus relacionamentos, que são conectados entre si como um gráfico (RATIONAL SOFTWARE CORPORATION, 2008). O jogo de RPG desenvolvido apresenta a seguinte estrutura de componentes, conforme ilustra a Figura 12, destacando em azul os componentes alterado, em verde os novos componentes e em amarelo os componentes mantidos conforme versão 1.0.

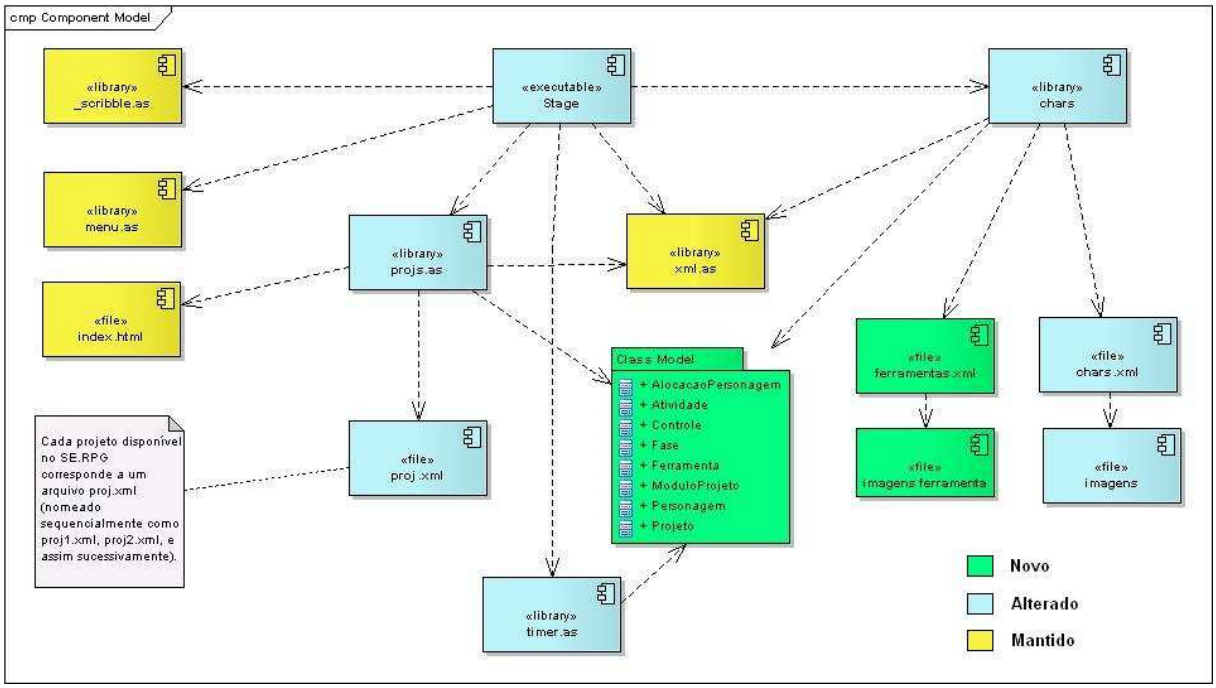


Figura 12 – Digrama de componente do RPG

a) stage (stage.fla): arquivo executável do jogo, contém a interface e centraliza as chamadas de funções dos demais módulos;
b) componente de interface de controle de movimentação (_scribble.as): módulo que contém as funções para o movimento da personagem jogadora “gerente de projeto” e interação deste com o ambiente externo;
c) componente de interface de menu (menu.as): compreende as funções de leitura dos

dados do projeto e desenho da interface em forma de menu;

d) componente de controle das iterações (timer.as): esse componente juntamente com a classe `controle,` contém o mecanismo do jogo, realizando o tratamento das variáveis, atualizando os dados referente à produtividade e atributos das personagens, progresso dos módulos e/ou atividades dos processos;
e) componente de *Parsing* (xml.as): biblioteca que contém as funções de leitura e *parsing* (análise das seqüências de dados de um arquivo de forma a determinar suas estruturas gramaticais, também chamada de análise de sintaxe) dos arquivos XML necessários para o jogo;
f) componente de controle das personagens (chars.as): compreende funções de manipulação dos dados das personagens e permite atribuição de tarefas para a personagem.As características das personagens, bem como seus atributos e sua imagem (avatar), constam nos dados do XML utilizado por esse componente que é analisado e estruturado pelo componente de *parsing*;
g) componente de controle de projetos (projs.as): similar ao controle de personagens, possui funções referentes à manipulação dos dados dos projetos. Os dados dos projetos, com a sua descrição, orçamento, tempo e custo, constam nos dados do XML utilizado por esse componente que é analisado e estruturado pelo componente de *parsing*;
h) Arquivos XML (index.xml, proj.xml, chars.xml e ferramenta.xml): contém as informações utilizadas pelo jogo para as personagens, projetos e ferramentas;
i) Arquivos de imagens: compreendem os elementos gráficos e animações utilizadas pelo jogo para ilustrações das personagens na interface do jogo.

3.3.4 XML Schema

Conforme previsto no requisito não-funcional (RNF 04), foi utilizado o XML para a elaboração dos documentos com as informações das personagens, ferramentas e projetos do jogo.

XML é uma linguagem desenvolvida pelo W3C (World Wide Web Consortium), principalmente para solucionar limitações HTML. De acordo com a W3C (2008) a XML é um formato de texto com marcações que tem adquirido importância crescente na transmissão de dados via Internet. Algumas características do XML são: separação do conteúdo da

formatação, independência de plataforma, possibilidade de marcações e possibilidade de interagir com banco de dados distintos.

Segundo Tesch (2002, p. 19), o XML Schema tem como objetivo definir as partes de um documento e descrever como elas podem ou não serem usadas, o que pode ser colocado em seus interiores e se são ou não elementos obrigatórios do documento.

Na construção desse projeto foram utilizados 4 arquivos XML para armazenar e viabilizar as regras do jogo. A Figura 13, em XML Schema, detalha a estrutura e forma de armazenamento dos dados das ferramentas CASE disponíveis no jogo, detalhando o componente ferramentas.xml (demonstrado na figura 14). A Figura 14, também em XML Schema, apresenta o XML Schema que contém a estrutura que armazena os dados das personagens, ou seja, detalha o componente chars.xml (apresentado na figura 14). O XML Schema referente aos projetos do jogo foi mantido conforme a primeira versão do jogo, podendo ser encontrados em Molléri (2006, p. 56).

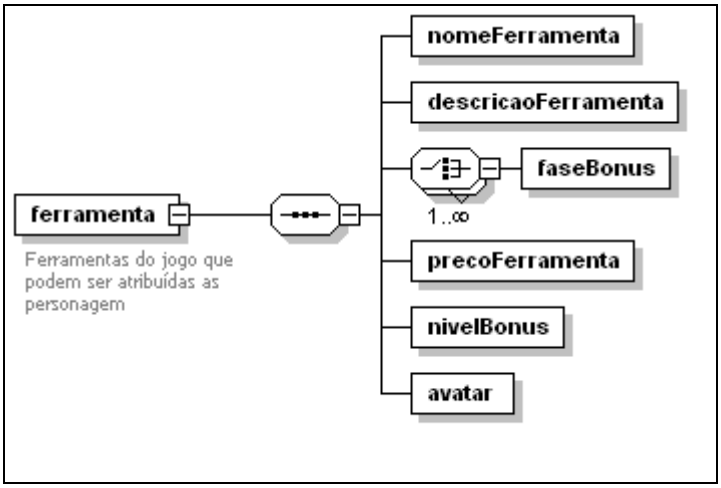


Figura 13 – Detalhes do elemento ferramenta

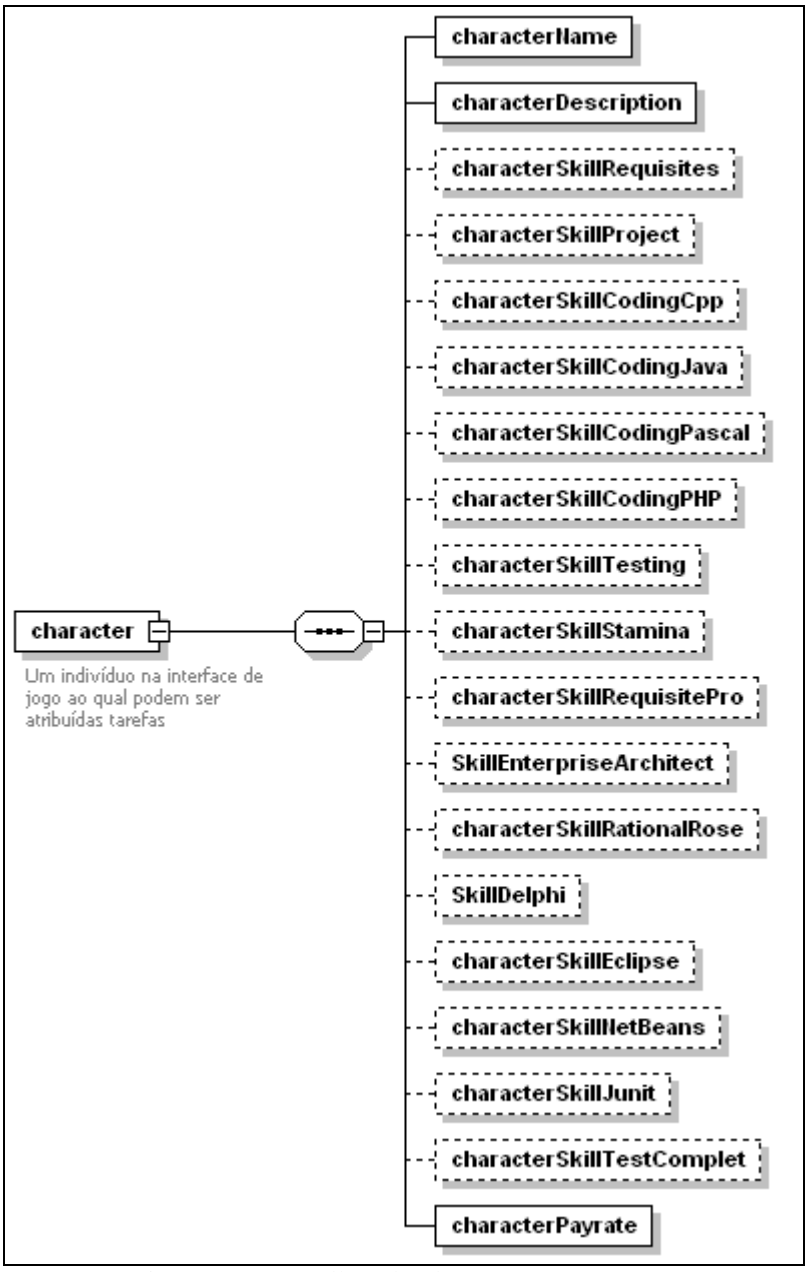


Figura 14 – Detalhes do elemento personagem

3.4 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.4.1 Técnicas e ferramentas utilizadas

A linguagem de programação utilizada no desenvolvimento do jogo é o ActionScript, no ambiente de programação Macromedia Flash Player. Os elementos gráficos criados, como os novos botões e novas caixas de conteúdo, também foram feitos utilizando o programa Macromedia Flash. Animações em Flash que compreendem a movimentação e alteração de estilos visuais nos elementos gráficos foram mantidas na interface.

O ActionScript permitiu desde o desenvolvimento do mecanismo de regras, algoritmos de leitura de dados e da construção lógica das ações das personagens, até a programação orientada a objetos, com classes e métodos.

Dentre os *scripts* existentes da primeira versão do projeto, dois deles são códigos de repositório público:

a) `menu.as`: *script* que efetua a criação de um menu de interface em Flash através da leitura de dados de um XML que especifica tanto os itens principais quanto secundários e permite alguma customização de cores e fonte (OJEDA, 2006);
b) `scribble.as`: este módulo é utilizado para realizar a movimentação de personagens através da repetição de ilustrações de modo que pareça desempenhar um movimento contínuo, utilizando o clique do mouse como parâmetro para o destino do movimento (BHANGAL, 2004).

O trecho de código fonte mostrado no Quadro 7 abaixo, apresenta a criação da classe `ferramenta` no ActionScript.

```
class Ferramenta{
      private var nomeFerramenta;
      private var descFerramenta;
      private var faseBonus;
      private var precoFerramenta;
      private var rollBonus;
      private var emJogo;
      private var emUso;
      private var avatarFerramenta;

      // Construtor da classe que grava os parâmetros
      function Ferramenta(nomeFerramenta, descFerramenta, faseBonus,
                                    precoFerramenta, rollBonus, emJogo,
                                    emUso, avatarFerramenta:String){
            this.nomeFerramenta           = nomeFerramenta;
            this.descFerramenta           = descFerramenta;
            this.faseBonus          = faseBonus;
            this.precoFerramenta          = precoFerramenta;
            this.rollBonus                = rollBonus;
            this.emJogo             = emJogo;
            this.emUso              = emUso;
            this.avatarFerramenta         = avatarFerramenta;
      }
      function setNomeFerramenta(ferramenta){
            this.nomeFerramenta     = ferramenta;
      }
      function getNomeFerramenta(){
            return this.nomeFerramenta;
      }
      function setDescFerramenta(ferramenta){
            this.descFerramenta = ferramenta;
      }
      function getDescFerramenta(){
            return this.descFerramenta;
      }
      function setFaseBonus(ferramenta){
            this.faseBonus    = ferramenta;
      }
      function getFaseBonus(){
            return this.faseBonus;
      }
```

Quadro 7 – Criação de uma classe no ActionScript

O desenvolvimento do trabalho foi realizado com orientação a objetos, sendo a sua arquitetura estruturada em 3 camadas, sendo a camada que contém a estrutura do jogo, a camada de controle e a camada de interface. Sendo nessa última, onde são realizadas as alterações gráficas visíveis ao jogador. A camada da interface sempre requisita informações a classe `controle`, podendo essa ser considerada o cérebro do jogo, pois é nela que estão implementadas todos os tratamentos e regras utilizadas no decorrer do jogo, como por

exemplo, os testes de sucesso das personagens, os bônus gerados quanto a ferramentas e o controle da *stamina*.

A classe `controle` tem acesso direto às classes de domínio do jogo (apresentadas no Quadro 6), que são as classes que armazenam os dados do jogo, como por exemplo, dos projetos e seu prazo, custo, estrutura, os dados das personagens juntamente com seus atributos, perícias e status, bem como os dados das fases e dos módulos.

O mecanismo do jogo está contido no script `timer.as`, é ele que a todo momento acessa a classe controle para realizar tratamentos das variáveis, atualizar dados referente a produtividade e status das personagens, e o progresso dos módulos ou atividades através das regras do RPG. No Quadro 8 é apresentado trecho do código da classe `controle` em que é feita a verificação e a atualização do orçamento do jogador, nessa verificação é descontado o salário de cada personagem que está trabalhando no projeto e, caso o jogador tenha comprado uma ferramenta CASE, o valor dela será também descontado do orçamento do jogador.

```
/* Método que atualiza o orçamento do jogador */

function atualizaOrcamento(salario){
      _global.projeto.setOrcProjetoAtual(_global.projeto.getOrcProjetoAt
ual()-parseInt(salario));
            /* Teste realizado para identificar se o jogador comprou
alguma ferramenta CASE */
      if    (_global.precoFerramenta      > 0){
            _global.projeto.setOrcProjetoAtual(
_global.projeto.getOrcProjetoAtual()-parseInt(_global.precoFerramenta));
            _global.precoFerramenta       = 0;
      }
      return _global.projeto.getOrcProjetoAtual();
}
```

Quadro 8 – Atualização do orçamento do jogador

3.4.2 Operacionalidade da implementação

O SE•RPG é um software desenvolvido em Flash, necessitando assim do Macromedia Flash Player para ser utilizado, sendo esse, um *player* bastante disseminado atualmente, permitindo a execução tanto local quanto remotamente (em ambiente web) do jogo.

A seguir são apresentadas as telas do jogo acompanhado de uma descrição de seu funcionamento.

A tela inicial apresenta as duas opção de avatar da personagem gerente de projeto que irá ser representada pelo jogador e guiar a narrativa do enredo. Como ilustra a Figura 15, o jogador deverá escolher um dos dois avatares clicando em cima do desejado.

Figura 15 – Tela de apresentação do jogo

Após deve-se selecionar o projeto com o qual se quer trabalhar. O jogo apresenta uma descrição sobre o projeto escolhido contendo orçamento e prazo para o desenvolvimento. Nessa mesma etapa, o jogador terá que definir em que modelo de processo será o desenvolvimento (modelo cascata, iterativo ou prototipação) e que linguagem de programação será utilizada na implementação, conforme a Figura 16.

Figura 16 – Tela de seleção do projeto

Com a definição do projeto, o jogador deve montar a sua equipe de desenvolvimento, como mostra a Figura 17. Tendo como opção 8 personagens, que são representados graficamente com o nome, cargo, descrição, perícias e custo diário para o projeto. Essa descrição fornece um resumo das habilidades da personagem, quais são seus pontos fortes e fracos, bem como, as linguagens de programação e ferramentas que é familiarizado.

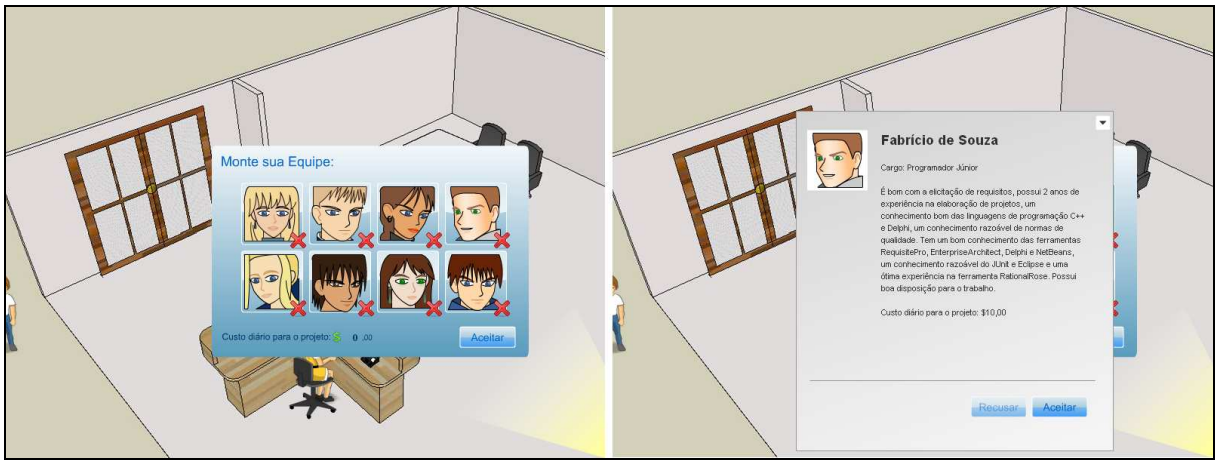


Figura 17 – Tela de seleção da equipe de desenvolvimento

A partir deste momento, o gerente de projeto (jogador), pode iniciar o desenvolvimento do software, delegando tarefas para a sua equipe. Para isto, basta clicar sobre o avatar da personagem que foi contratada e está presente na sala de desenvolvimento. Deve-se escolher o módulo (caso o modelo de desenvolvimento escolhido for Iterativo) e a atividade a ser desempenhada.

As atividades devem ser atribuídas seqüencialmente, iniciando com a primeira fase do jogo que é a elicitação de requisitos, sendo que sempre que uma fase é concluída, a fase seguinte é habilitada para o jogador. É possível alocar mais de uma personagem em uma mesma tarefa, permitindo assim, um progresso mais rápido na atividade.

No topo da tela, a ilustração da face das personagens exibem as tarefas que estão sendo realizadas através de pequenos ícones, como é ilustrado na Figura 18(b). Ao clicar sobre essa imagem da face da personagem é possível rever a descrição da mesma e um gráfico de produtividade que demonstra a contribuição para o projeto nas diferentes etapas do desenvolvimento, apresentado na Figura (19).

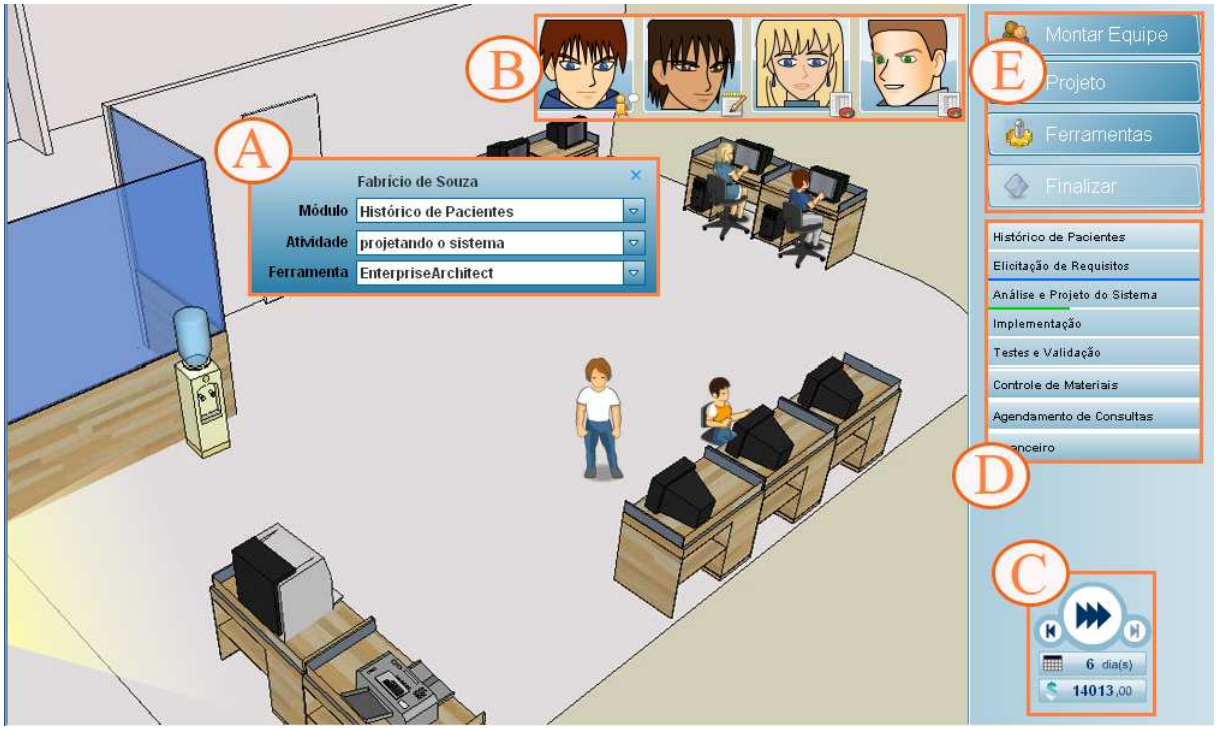


Figura 18 – Ambiente da sala de desenvolvimento: (a) atribuição de tarefas e ferramenta; (b) ilustração de face das personagens; (c) controle de tempo; (d) progresso dos módulos do projeto; (e) botões para edição da equipe, visualização do projeto, aquisição de ferramentas e finalização do projeto.

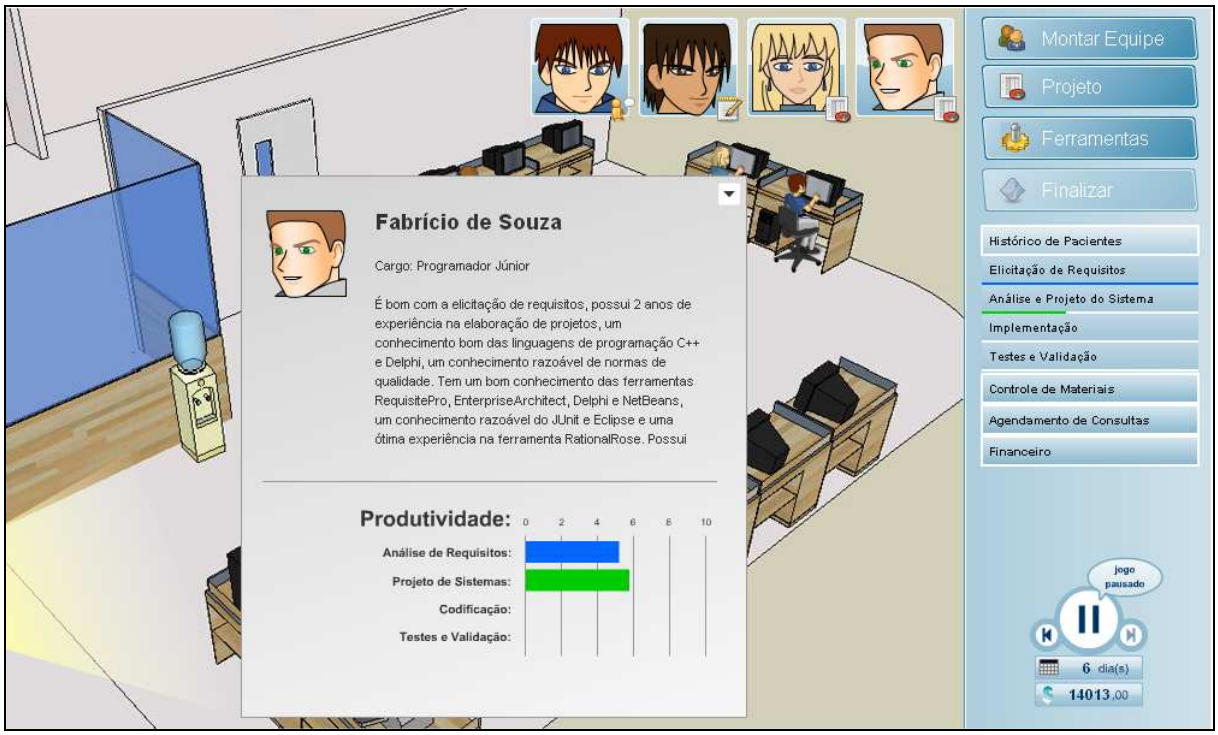


Figura 19 – Descrição da personagem e histórico da produtividade.

No botão “Ferramentas”, conforme representado na Figura 18(e), é possível que o gerente de projetos adote ferramentas CASE para equipar suas personagens. Para isso basta escolher a ferramenta que deseja e clicar no botão comprar conforme ilustrado na Figura 20. Automaticamente será descontado do orçamento o valor daquela ferramenta.

Figura 20 – Tela de aquisição de ferramentas

Para equipar a ferramenta em uma personagem, assim como para as atividades, basta clicar no avatar da personagem que deseja, e no campo Ferramenta selecionar uma em questão, conforme a Figura 21. É permitido a aquisição de apenas uma licença de cada ferramenta e cada personagem poderá utilizar apenas uma ferramenta por vez. A partir do momento que uma personagem estiver utilizando uma ferramenta, esta não estará mais disponível para que outra personagem utilize.

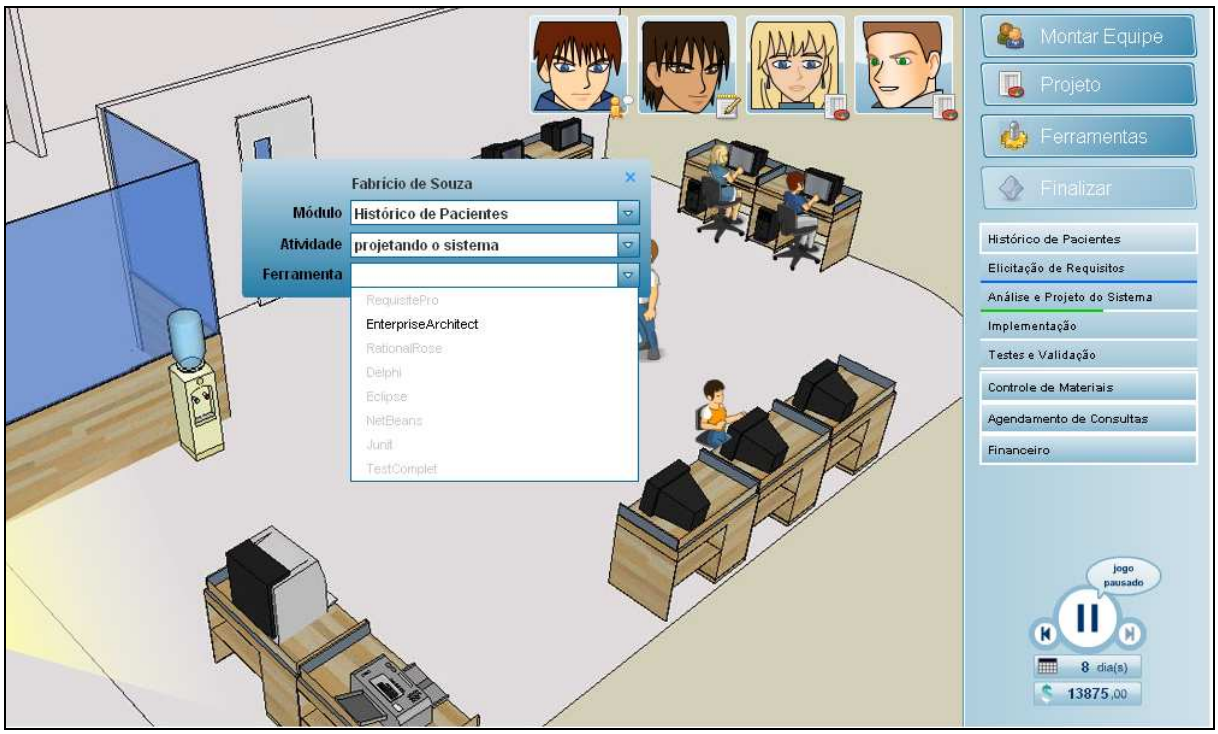


Figura 21 – Tela seleção de uma ferramenta para a personagem

O jogo apresenta um controle de custo e tempo, indicado na Figura 18(c), que permite que o jogador fique a todo momento atento quanto ao prazo de desenvolvimento e custos envolvidos com as personagens e ferramentas CASE. Através desse controle é possível alterar as velocidades do jogo inclusive pará-lo quando desejar.

Nesse projeto foi implementada a opção do modelo de prototipação aos modelos de desenvolvimento que o jogador pode selecionar para realizar a construção do software. Nesse modelo, após o término da etapa "Construção do protótipo", a personagem do cliente será exibida, informando sobre a aprovação ou não do protótipo, conforme citado na Regra do Jogo 07.

A opinião do cliente gera um evento no jogo apresentando uma mensagem para o jogador entre as 3 possíveis, que são:

a) aprovado, o protótipo atendeu muito bem as expectativas, pode prosseguir o projeto: o jogador segue o fluxo normal do jogo;
b) aceito, o protótipo ficou bom, faltam apenas alguns ajustes que poderão ser realizados durante o projeto: o jogador segue o fluxo normal do jogo;
c) reprovado, não era esse o protótipo que eu desejava, por favor realize os ajustes necessários: nesse caso a progressão das 3 fases referentes ao desenvolvimento do protótipo (Elicitação de requisitos, Projeto rápido e Construção do protótipo) são reduzidas pela metade e, será necessário que o jogador conclua novamente essas etapas para assim seguir o restante do desenvolvimento do software.

Foi implementado também nesse projeto o atributo *Stamina,* conforme descrito na Regra do Jogo 09. O objetivo da *Stamina* é criar situações aleatórias não esperadas pelo jogador. Até em determinadas situações o gerente de projetos terá que tomar decisões em relação a sua equipe de desenvolvimento, para que o desenvolvimento do software não seja prejudicado.

Cada personagem possui um valor pré-determinado em seu atributo *Stamina* (Esse valor representa a quantidade de ciclos do jogo que a personagem estará trabalhando). Quando a quantidade dos ciclos atingir o valor da *Stamina* da personagem, então o jogo irá apresentar uma mensagem aleatória informando o ocorrido com a personagem, sendo que esta não estará mais presente na sala do desenvolvimento por um período de 8 ciclos. As mensagens que podem ser apresentadas em função da *Stamina* da personagem são:

a) descansar, a personagem se retirou para descansar;
b) atestado médico, o médico recomendou descanso a personagem;
c) problemas familiares, a personagem está com problemas familiares, irá ficar um

período sem trabalhar;

d) casamento, a personagem casou, ficará um período ausente em lua de mel;

e) problemas disciplinares, a personagem apresentou problemas disciplinares e ficará um período afastado do serviço.

Na Figura 22, é apresentado um momento do jogo em que uma personagem ficou ausente em virtude de sua *Stamina.*

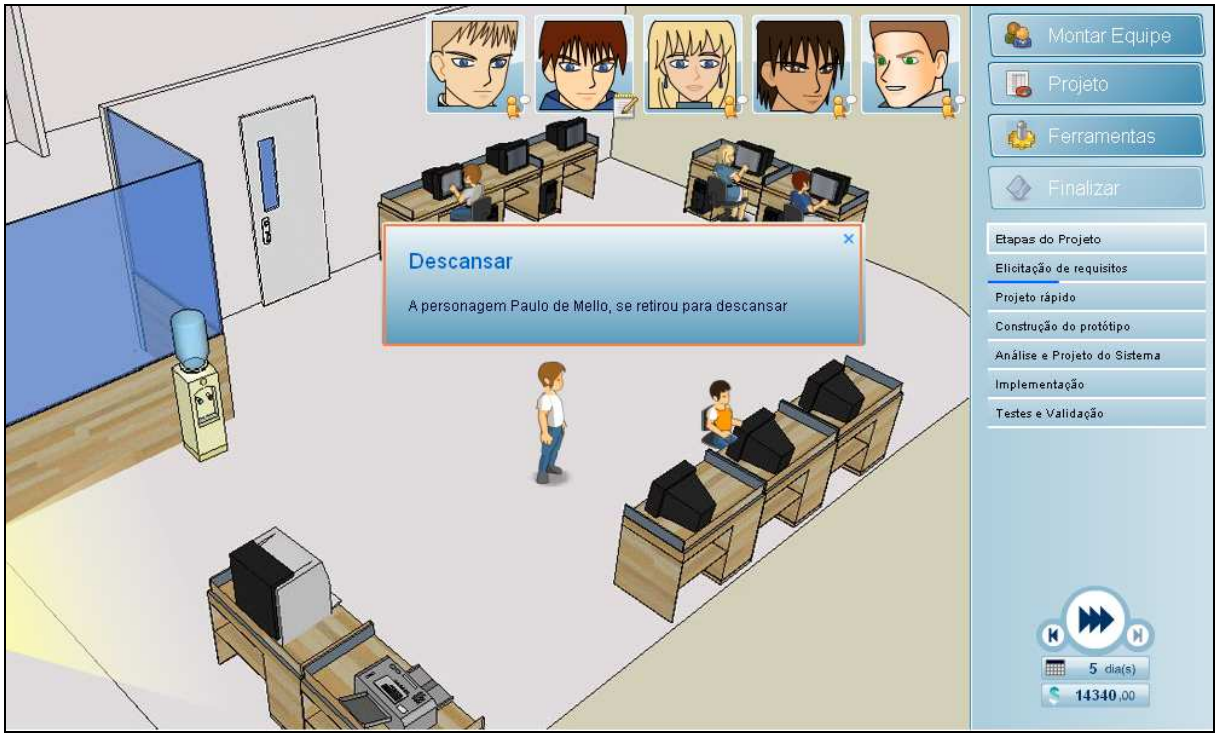


Figura 22 – Ausência da personagem em função da *Stamina*

Através dos menus localizados na lateral direita, conforme a Figura 18(d), são demonstrado os módulos em desenvolvimento e a situação de progressão de cada um. A cada período de quatro horas, considerando o tempo do jogo, é realizada a atualização deste progresso, levando em consideração as personagens alocadas em suas respectivas atividades.

A equipe de desenvolvimento pode ser editada a qualquer momento do jogo, adicionando ou removendo personagens, através do botão “Monta equipe”, conforme indicado na Figura 18(e).

A partir do momento que a etapa de implementação estiver ocorrendo por alguma personagem, é possível que o gerente finalize a entrega do software ao cliente e assim finalize o jogo. Com isso é exibido um breve comentário a respeito das metas do jogador e dos resultados obtidos. Caso o jogador tenha feito escolhas erradas, como o modelo incorreto para o desenvolvimento do projeto ou a entrega incompleta deste, a conclusão do jogo refletirá essas decisões, conforme ilustrado na Figura 23.

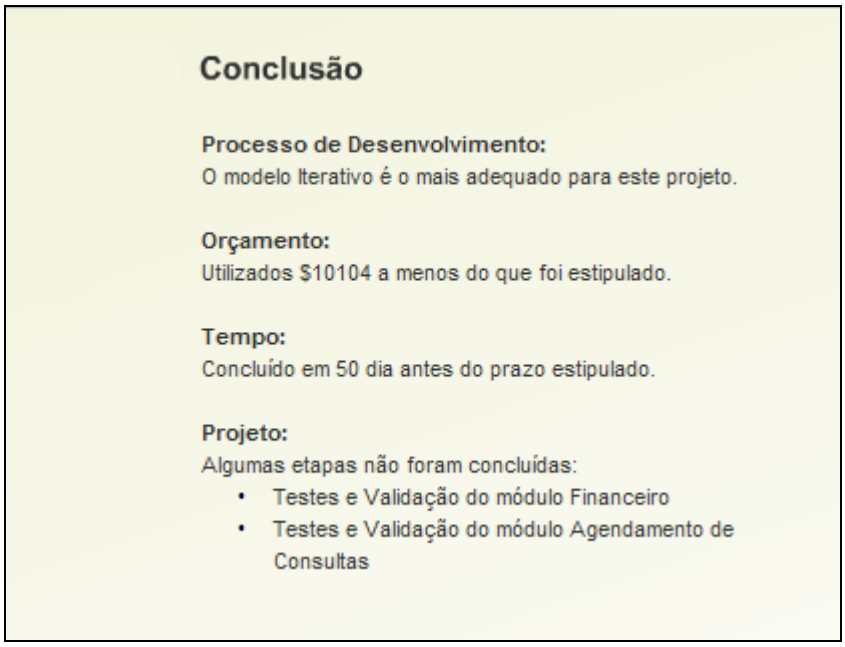


Figura 23 – Tela da conclusão do jogo

3.5 RESULTADOS E DISCUSSÃO

O uso de instrumentos de ensino que vão além do que é apresentado apenas em sala de aula, como por exemplos os jogos, propiciam um estímulo maior a socialização e criatividade do aluno, servindo como uma ferramenta motivadora onde o jogador vai montando a sua própria história através do que foi aprendido em sala de aula.

O RPG, sendo um estilo de jogo diversificado e que estimula o raciocínio dos jogadores, se encaixa muito bem na utilização pedagógica, pois fornece caminhos para que o aluno de maneira descontraída agrupe ao mesmo tempo acontecimentos, ações e intenções. O RPG também pode ser moldado e trabalhado de várias formas, não ficando preso a uma única linha de pensamento ou regras, dessa maneira os jogadores sempre estão vivendo novos desafios no jogo.

Pode-se observar, conforme avaliação efetuada por Molléri (2006), que o SE•RPG atende ao objetivo proposto pelos RPGs educativos, viabilizando aos professores uma ferramenta capaz de auxiliar a aplicação na prática dos conhecimentos passados em sala de aula. Além da avaliação formal realizada no âmbito do TCC de Molléri, a ferramenta (em sua versão 1.0) foi utilizada em outras 4 turmas de Engenharia de Software, sendo constatado pela professora o aspecto motivador e educativo do SE•RPG.

Estes aspectos são obtidos através de um ambiente simulando uma empresa de software, contendo uma interface prática, que ajuda na fixação dos conceitos da matéria. O jogador (aluno) é levado a efetuar atividades, tais como: definição adequada do processo de desenvolvimento, concluir um projeto no prazo estipulado, realizar a contratação e delegação de tarefas para uma equipe de desenvolvimento, bem como adotar ferramentas CASE para o desenvolvimento de projetos. Tudo isso envolvendo as técnicas e conceitos de Engenharia de Software, focando gerenciamento de projetos e modelos de desenvolvimento.

Alguns aspectos encontrados em um desenvolvimento de software real também são levantados no SE•RPG, como planejamento de projeto envolvendo custos e prazos, definição de equipe, dentre outros. Dessa forma, o SE•RPG fornece uma visão prática dos conceitos estudados, desenvolve a motivação e o desafio ao aprendizado, propiciando ao aluno uma oportunidade para experimentações. Além disso o uso do jogo em sala de aula pode propiciar momentos de discussão na área de Engenharia de Software.

Com a possibilidade de 3 modelos diferentes de desenvolvimento e 3 projetos distintos, o jogador não se atém sempre a uma mesma seqüência de jogo, além da necessidade do gerenciamento da equipe de desenvolvimento e a compra e uso de ferramentas CASE, é necessário a constante intervenção e ação do usuário no jogo, evitando que o progresso do software para o cliente seja prejudicado.

No Quadro 9 é apresentado um comparativo entre alguns jogos de RPG existentes que abordam o tema de engenharia de software.

CARACTERÍSTICA	**SimSE**	**SimulES**	**SE•RPG**	**SE•RPG 2.0**
Tipo de jogo	simulador	jogo de mesa	RPG	RPG
Interface	software	cartas	software	software
Ambiente	gráfico, representando sala de desenvolvi-mento	cartões de projetos, tabuleiro, cartas e dado	gráfico, 3 salas representando empresa de desenvolvi-mento	gráfico, 3 salas representando empresa de desenvolvimento
Jogadores	jogo solo (1)	de 4 a 8 jogadores	jogo solo (1)	jogo solo (1)
Paradigmas de desenvolvimento	modelos cascata, incremental,	modelo cascata	modelos cascata e iterativo	modelos cascata, iterativo e prototipação

	inspeção, prototipação, rápida e RUP			
Interpretador	java	-	Macromedia Flash Player ou arquivo executável	Macromedia Flash Player ou arquivo executável
Etapas do processo	varia de acordo com o paradigma de desenvolvi-mento	construir artefatos, inspeção de artefatos, corrigir defeito e integrar artefatos em um módulo	(i) análise de requisitos; (ii) projeto; (iii) implementação ; (iv) testes e validação.	varia de acordo com o paradigma de desenvolvimento
Equipe de desenvolvimento	pré-definida embora possa ser alterada durante o andamento do jogo	definida através de cartas	é definida pelo jogador e pode ser alterada a qualquer momento do jogo	é definida pelo jogador e pode ser alterada a qualquer momento do jogo
Conceito de ferramentas CASE	presente	definida através de cartas	Ausente	8 ferramentas que podem ser compradas pelo gerente de projetos a qualquer momento do jogo
Representação do cliente	presente em momentos específicos do	ausente	presente através da descrição do	presente através da descrição do projeto e da

	jogo		projeto somente	avaliação do protótipo quando for modelo de prototipação

Quadro 9 – Comparativo entre jogos existentes

O software SimSE apesar de possuir um componente chamado Model Builder, que permite a edição dos modelos e recursos do jogo, além de ter vários recursos em relação aos modelos e construção dos desafios, é bastante estático se comparado com as versões do SE•RPG, pois não permite a movimentação das personagens e como não possui a figura do gerente de projetos representada graficamente, não há a troca de ambientes.

O SimulES, apesar de ser um jogo com técnicas de RPG também voltado para Engenharia de Software, possui algumas características bem distintas em relação aos outros jogos apresentados no Quadro 9. Uma das principais diferença é que o SimulES é um modelo competitivo de jogo de cartas, onde é necessário no mínimo 4 jogadores e, no final do jogo um desses jogadores é declarado o vencedor, sendo que no RPG tradicional, não existe uma declaração visível de vencedor do jogo, mas sim, de um grupo de jogadores que atingiram um objetivo em comum.

Em relação a primeira versão do SE•RPG, com a versão desenvolvida nesse trabalho, pode-se notar, analisando o quadro 9, que foi mantida a interface gráfica dos ambientes do jogo, mas que houve uma melhoria significativa nos processos, regras e opções de ações que o jogador pode tomar durante a partida. Alguns conceitos que existem no SimSE e não estavam presentes no SE•RPG, foram implementados nessa versão, como por exemplo, o conceito de ferramenta CASE e a variedade das etapas de processo de acordo com o paradigma de desenvolvimento escolhido.

4 CONCLUSÕES

Os objetivos propostos neste trabalho foram atingidos plenamente. As melhorias e alterações realizadas na primeira versão do SE•RPG para a versão atualizada nesse trabalho são visíveis ao jogador já a partir do início do jogo, onde é possível a escolha do modelo de prototipação para o desenvolvimento da atividade escolhida. Através do uso do Flash com o Action Script, foi possível reescrever o jogo de forma orientada a objetos, deixando assim, o código mais legível, padronizado e de fácil entendimento.

O acréscimo de ferramentas CASE no jogo, tornou a trama e o ambiente ainda mais completo. Assim como em uma partida de RPG tradicional, agora o jogador pode comprar equipamentos, nesse caso ferramentas CASE e equipar em uma personagem.

Dessa forma propicia um aumento ainda maior a realidade do jogo, com a situação de uma empresa real e o próprio RPG, onde os programadores podem ter acesso a determinadas ferramentas CASE. Mas, independente de qual ferramenta for, o programador terá que ter o conhecimento de como utilizá-la, para que assim se tenha um melhor aproveitamento e uma maior eficiência e produtividade no desenvolvimento da tarefa.

Com as ferramentas no jogo, houve uma alteração nos cálculos dos testes de sucesso das personagens e o sistema d20, juntamente com as regras fundamentadas na Engenharia de Software e relacionadas com as perícias das personagens. Isto exige uma atenção maior do gerente de projetos ao comprar e equipar uma ferramenta em uma personagem, pois é necessário certificar que aquela personagem realmente sabe utilizar aquela ferramenta, para assim compensar o valor investido.

As ausências inesperadas das personagens durante o desenvolvimento do projeto, acrescentou ao jogo além de uma melhoria em relação a simulação com a realidade, um fator descontraído e ao mesmo tempo importante, pois estimula que o gerente de projetos esteja preparado para tomar as decisões corretas em relação a sua equipe, evitando que haja problemas no projeto.

Por fim, observa-se uma ferramenta mais completa e preparada para ser utilizada para fins didáticos, onde além de contemplar ainda mais os conceitos da Engenharia de Software, abre um leque maior de possibilidade e ações que jogador pode tomar durante o jogo.

4.1 EXTENSÕES

Durante o desenvolvimento do trabalho foram apuradas algumas melhorias desejáveis ao jogo, garantindo assim, mais realidade e aproximação com as características de um RPG. As sugestões para trabalhos futuros são:

a) implementar uma ferramenta para edição dos módulos e recursos do jogo;
b) possibilitar uma opção para salvar o jogo;
c) progressão do nível das habilidades das personagens durante o jogo (passagem de nível);
d) criar uma trama seqüencial entre projetos, permitindo ao jogador que após a conclusão do desenvolvimento de um software, ele possa começar o desenvolvimento de outro software, com os recursos adquiridos do trabalho anterior e a sua equipe mais experiente;
e) ampliar os modelos de desenvolvimento não contemplados inicialmente, como o modelo em espiral;
f) apresentar exemplos de artefatos produzidos ao final de cada atividade.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, F. **Caminhos para o uso do RPG na Educação**. [S.l.], [2000?]. Disponível em: < http://www.uff.br/aleph/textos_em_pdf/caminhos_para_o_uso_do_rpg_na_educacao.pdf>. Acesso em: 11 fev. 2008.

ANDRADE, F. **RPG e educação**: possibilidades de uso do RPG. [S.l.], 1997. Disponível em: <http://www.historias.interativas.nom.br/educ/rpgtese.htm>. Acesso em: 14 ago. 2007.

BHANGAL, S. **Flash Hacks**. [S.l.]. O'Reilly & Associates, Inc, 2004.

BITTERNCOURT, R. J. **A Utilização dos role-playing games digitais no processo de ensino-aprendizagem.** [Porto Alegre], 2003. p. 26-27. Disponível em: <http://www.inf.pucrs.br/tr/tr031.pdf>. Acesso em: 11 fev. 2008.

BOLZAN, R. F. F. A. **O aprendizado na internet utilizando estratégias de roleplaying game (RPG).** 2003. 303 f. Tese (Doutorado em Engenharia de Produção) – Curso de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http:// http://teses.eps.ufsc.br/Resumo.asp?4437>. Acesso em: 09 ago. 2007.

FIGUEIREDO, E. M. L. et al. **SimulES**: um jogo para o ensino de engenharia de software**.** 2006. 38 f. Trabalho apresentado como requisito parcial para aprovação na Disciplina Evolução de Software – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro. Disponível em: http://www.teccomm.les.inf.puc-rio.br/emagno/simules/simules.htm. Acesso em: 20 ago. 2007.

JALOTE, P. **An integral approach to software engineering.** 2. ed. New York: Springer, 1997.

KLIMICK, C. **Construção de personagem & aquisição de linguagem:** o desafio do RPG no INES. [S.l.], 2003. Disponível em: <http://www.historias.interativas.nom.br/zoo/projeto/index.html>. Acesso em: 10 ago. 2007.

LEITE, J.C. **Engenharia de software**. [S.l.], [2007]. Disponível em: <http://engenhariadesoftware.blogspot.com/2007/02/o-que-engenharia-de-software.html>. Acesso em: 10 mar. 2008.

MARCATTO, A. **Aventuras educacionais**. [S.l.], [2005?]. Disponível em: <http://www.alfmarc.psc.br/avent_edu.asp>. Acesso em: 08 set. 2007.

MELO, A.C. **Desenvolvendo aplicações com UML 2.0:** definitivo. São Paulo: Pearson Makron Books, 2004.

MOLLÉRI, J. S. **Utilizando o RPG como ferramenta de aprendizado para o processo de desenvolvimento de software.** 2006. 78 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí.

NAVARRO, E.; HOEK, A. Towards game-based simulation as a method of teaching software engineering. In: ASEE/IEEE FRONTIERS IN EDUCATION CONFERENCE, 32., 2002, Boston. **Proceedings**… Boston: IEEE, 2002. Não paginado. Disponível em: <http://www.ics.uci.edu/~emilyo/papers/FIE2002.pdf>. Acesso em: 02 set. 2007.

OH, E.; HOEK, A. Adapting game technology to support individual and organizational learning. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 13., 2001, Buenos Aires. **Proceedings...** Skokie: Knowledge Systems Institute, 2001. Não paginado Disponível em: <http://www.ics.uci.edu/~emilyo/papers/SEKE2001.pdf>. Acesso em: 19 set. 2007.

OJEDA, D. A. **XML semi dynamic menu**. [S.l.],[2006]. Disponível em: <http://www.actionscript.org/showMovie.php?id=1149>. Acesso em: 03 mar. 2008.

PILONE, Dan; PITMAN, Neil. **UML 2:** rápido e prático**.** Rio de Janeiro: Alta Books, 2006.

PRESSMAN, R. S. **Engenharia de software.** Tradução Rosângela Delloso Penteado. São Paulo: Makron Books, 2006.

PRINCE, A. E. **A importância dos softwares na educação.** Jacareí: Campus, 2004.

RATIONAL SOFTWARE CORPORATION. [S.l.],[2006]. Disponível em: <http://www.wthreex.com/rup>. Acesso em: 08 mai. 2008.

RODRIGUES, S. **Roleplaying game e a pedagogia da imaginação no Brasil.** Rio de Janeiro: Bertrand Brasil, 2004.

SOARES, M. S. **Uma experiência de ensino de engenharia de software orientada a trabalhos práticos**. [S.l.], [2005?]. Disponível em: <http://www.sbc.org.br/bibliotecadigital/download.php?paper=46>. Acesso em: 28 ago. 2007.

SOMMERVILLE, I. **Engenharia de software.** 6. ed. Tradução Maurício de Andrade. São Paulo: Addison Wesley, 2003.

SVALDI, R. D. **O Brasil e o sistema d20**. [S.l.], [2003]. Disponível em: <http://www.sobrecarga.com.br/node/view/56>. Acesso em: 05 mar. 2008.

TANAKA, M. **RPG e educação**. [S.l.], [2001?]. Disponível em: <http://www.jogodeaprender.com.br/artigos_1.html>. Acesso em: 02 set. 2007.

TESCH, J. R. **XML schema.** Florianópolis: Visual Books Ltda, 2002.

VALENTE, J. A. **Análise dos diferentes tipos de software usados na educação.** [Canoas], 1999. p. 71-85. Disponível em: <http://www.ulbra.tche.br/~magda/edumat/aswvalente.pdf >. Acesso em: 19 ago. 2007.

VERAS, A.; SANTOS, C. B. **RPG como ferramenta de ensino de literatura**. [S.l.], 2004. Disponível em: <http://www.rpgeduc.com/artigo02.pdf>. Acesso em: 27 ago. 2007.

WIZARDS OF THE COAST. **O Brasil e o sistema d20**. [S.l.], [2003]. Disponível em: < http://www.wizards.com/>. Acesso em: 06 mar. 2008.

W3C. [S.l.], [2006]. Disponível em: <http://www.w3.org/>. Acesso em: 06 abril 2008.

ZUCHI, I. **O desenvolvimento de um protótipo de sistema especialista baseado em técnicas de RPG para o ensino de matemática.** 2000. 136 f. Dissertação (Mestrado em Engenharia de Produção e Sistemas) – Curso de Pós-Graduação em Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://teses.eps.ufsc.br/Resumo.asp?1015>. Acesso em: 20 ago. 2007.

APÊNDICE A – Arquivo XML das personagens do jogo

No Quadro 10 é apresentado o arquivo XML com as informações de nome, cargo, descrição perícias, salário e o avatar das personagens do jogo.

```
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Vivian
Fagundes (Fundação Universidade Regional de Blumenau - FURB) -->
<characters>
      <character>
            <characterName>Fabrício de Souza</characterName>
            <characterDescription>Cargo: Programador Júnior
É bom com a elicitação de requisitos, possui 2 anos de experiência na
elaboração de projetos, um conhecimento bom das linguagens de programação C++
e Delphi, um conhecimento razoável de normas de qualidade. Tem um bom
conhecimento das ferramentas RequisitePro, EnterpriseArchitect, Delphi e
NetBeans, um conhecimento razoável do JUnit e Eclipse e uma ótima experiência
na ferramenta RationalRose. Possui boa disposição para o
trabalho.</characterDescription>
            <characterSkillRequisites>2</characterSkillRequisites>
            <characterSkillProject>4</characterSkillProject>
            <characterSkillCodingCpp>2</characterSkillCodingCpp>
            <characterSkillCodingJava>0</characterSkillCodingJava>
            <characterSkillCodingDelphi>3</characterSkillCodingDelphi>
            <characterSkillCodingPHP>0</characterSkillCodingPHP>
            <characterSkillTesting>-1</characterSkillTesting>
            <characterSkillStamina>40</characterSkillStamina>
            <characterSkillRequisitePro>2</characterSkillRequisitePro>

      <characterSkillEnterpriseArchitect>3</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>5</characterSkillRationalRose>
            <characterSkillDelphi>3</characterSkillDelphi>
            <characterSkillEclipse>1</characterSkillEclipse>
            <characterSkillNetBeans>2</characterSkillNetBeans>
            <characterSkillJunit>0</characterSkillJunit>
            <characterSkillTestComplet>-1</characterSkillTestComplet>
            <characterPayrate>10</characterPayrate>
            <avatar>char0.swf</avatar>
            <sprite>char0.swf</sprite>
      </character>
      <character>
            <characterName>Fátima Vieira</characterName>
            <characterDescription>Cargo: Analista de Software Sênior
Possui uma facilidade de comunicação extraordinária e um ótimo conhecimento
de padrões de desenvolvimento de software, com 3 anos de experiência em
programação C++ e mais um semestre em PHP, possui um bom conhecimento de
normas de qualidade. Possui uma experiência  extraordinária na ferramenta
RequisitePro, um ótimo conhecimento nas ferramentas EnterpriseArchitect e
RationalRose, um bom conhecimento em NetBeans, JUnit e TestComplet e, um
razoável conhecimento na ferramenta de desenvolvimento Eclipse. Possui uma
razoável disposição para o trabalho. </characterDescription>
            <characterSkillRequisites>6</characterSkillRequisites>
            <characterSkillProject>4</characterSkillProject>
            <characterSkillCodingCpp>4</characterSkillCodingCpp>
            <characterSkillCodingJava>0</characterSkillCodingJava>
            <characterSkillCodingDelphi>0</characterSkillCodingDelphi>
            <characterSkillCodingPHP>1</characterSkillCodingPHP>
            <characterSkillTesting>3</characterSkillTesting>
            <characterSkillStamina>30</characterSkillStamina>
            <characterSkillRequisitePro>6</characterSkillRequisitePro>

      <characterSkillEnterpriseArchitect>5</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>4</characterSkillRationalRose>
            <characterSkillDelphi>-1</characterSkillDelphi>
            <characterSkillEclipse>0</characterSkillEclipse>
            <characterSkillNetBeans>2</characterSkillNetBeans>
```

```
            <characterSkillJunit>3</characterSkillJunit>
            <characterSkillTestComplet>2</characterSkillTestComplet>
            <characterPayrate>18</characterPayrate>
            <avatar>char1.swf</avatar>
            <sprite>char1.swf</sprite>
      </character>
      <character>
            <characterName>Paulo de Mello</characterName>
            <characterDescription>Cargo: Analista de Software Sênior
Possui uma boa capacidade para elicitação de requisitos e ótima elaboração de
projeto, além de um ótimo conhecimento da linguagem de programação C++, e
experiência de 4 anos com PHP, extraordinária atenção aos detalhes. Possui um
bom conhecimento nas ferramentas RequisitePro e RationalRose, uma ótima
habilidade em EnterpriseArchitect, um bom conhecimento em Eclipse, apenas uma
noção de JUnit, mas é extremamente competente com as ferramentas NetBeans e
TestComplet. Possui muita disposição para o trabalho.</characterDescription>
            <characterSkillRequisites>3</characterSkillRequisites>
            <characterSkillProject>5</characterSkillProject>
            <characterSkillCodingCpp>5</characterSkillCodingCpp>
            <characterSkillCodingJava>0</characterSkillCodingJava>
            <characterSkillCodingDelphi>-1</characterSkillCodingDelphi>
            <characterSkillCodingPHP>5</characterSkillCodingPHP>
            <characterSkillTesting>6</characterSkillTesting>
            <characterSkillStamina>10</characterSkillStamina>
            <characterSkillRequisitePro>3</characterSkillRequisitePro>

      <characterSkillEnterpriseArchitect>5</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>2</characterSkillRationalRose>
            <characterSkillDelphi>-2</characterSkillDelphi>
            <characterSkillEclipse>4</characterSkillEclipse>
            <characterSkillNetBeans>6</characterSkillNetBeans>
            <characterSkillJunit>1</characterSkillJunit>
            <characterSkillTestComplet>6</characterSkillTestComplet>
            <characterPayrate>14</characterPayrate>
            <avatar>char3.swf</avatar>
            <sprite>char3.swf</sprite>
      </character>
      <character>
            <characterName>Alessandra Gonçalves</characterName>
            <characterDescription>Cargo: Analista de Software Sênior
Tem uma boa facilidade de comunicação e experiência de 4 anos na elaboração
de projetos de software, conhecimento bom com as linguagens de programação
Java e Pascal, uma boa atenção aos detalhes.  Possui ótimo conhecimento nas
ferramentas RequisitePro e RationalRose, é extraordinariamente experiente na
ferramenta EnterpriseArchitect, possui um bom conhecimento em Delphi e
TestComplet, e já fez alguns mini-cursos de NetBeans e Junit. Não possui
muita disposição para o trabalho.</characterDescription>
            <characterSkillRequisites>3</characterSkillRequisites>
            <characterSkillProject>6</characterSkillProject>
            <characterSkillCodingCpp>0</characterSkillCodingCpp>
            <characterSkillCodingJava>2</characterSkillCodingJava>
            <characterSkillCodingDelphi>3</characterSkillCodingDelphi>
            <characterSkillCodingPHP>0</characterSkillCodingPHP>
            <characterSkillTesting>3</characterSkillTesting>
            <characterSkillStamina>20</characterSkillStamina>
            <characterSkillRequisitePro>4</characterSkillRequisitePro>

      <characterSkillEnterpriseArchitect>6</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>5</characterSkillRationalRose>
            <characterSkillDelphi>3</characterSkillDelphi>
            <characterSkillEclipse>-1</characterSkillEclipse>
            <characterSkillNetBeans>1</characterSkillNetBeans>
            <characterSkillJunit>1</characterSkillJunit>
            <characterSkillTestComplet>2</characterSkillTestComplet>
            <characterPayrate>17</characterPayrate>
            <avatar>char7.swf</avatar>
            <sprite>char7.swf</sprite>
      </character>
      <character>
            <characterName>Afonso Trevi</characterName>
```

```
            <characterDescription>Cargo: Programador Júnior
Possui uma habilidade ruim com a elicitação dos requisitos e um conhecimento
razoável de padrões de software, mas trabalhou com programação em Java por 3
anos e PHP por 2 anos, um ótimo conhecimento de normas de qualidade. Possui
um bom conhecimento nas ferramentas Delphi, Eclipse e TestComplet além de uma
ótima habilidade em NetBeans e Junit. Possui boa disposição para o
trabalho.</characterDescription>
            <characterSkillRequisites>-1</characterSkillRequisites>
            <characterSkillProject>1</characterSkillProject>
            <characterSkillCodingCpp>0</characterSkillCodingCpp>
            <characterSkillCodingJava>5</characterSkillCodingJava>
            <characterSkillCodingDelphi>4</characterSkillCodingDelphi>
            <characterSkillCodingPHP>0</characterSkillCodingPHP>
            <characterSkillTesting>4</characterSkillTesting>
            <characterSkillStamina>50</characterSkillStamina>
            <characterSkillRequisitePro>-2</characterSkillRequisitePro>
            <characterSkillEnterpriseArchitect>-
1</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>0</characterSkillRationalRose>
            <characterSkillDelphi>3</characterSkillDelphi>
            <characterSkillEclipse>2</characterSkillEclipse>
            <characterSkillNetBeans>4</characterSkillNetBeans>
            <characterSkillJunit>4</characterSkillJunit>
            <characterSkillTestComplet>2</characterSkillTestComplet>
            <characterPayrate>13</characterPayrate>
            <avatar>char4.swf</avatar>
            <sprite>char4.swf</sprite>
      </character>
      <character>
            <characterName>Juliane Dias</characterName>
            <characterDescription>Cargo: Programador Júnior
Possui uma boa habilidade com a elicitação de requisitos, mas um conhecimento
ruim dos padrões de software, experiência de 2 anos com programação PHP e
outros 4 anos com Pascal, uma ótima atenção aos detalhes. Possui um bom
conhecimento nas ferramentas RequisitePro, NetBeans e Junit além de um
domínio extraordinário em Delphi e um ótimo conhecimento em TestComplet.
Possui uma boa disposição para o trabalho.</characterDescription>
            <characterSkillRequisites>2</characterSkillRequisites>
            <characterSkillProject>-1</characterSkillProject>
            <characterSkillCodingCpp>0</characterSkillCodingCpp>
            <characterSkillCodingJava>0</characterSkillCodingJava>
            <characterSkillCodingDelphi>6</characterSkillCodingDelphi>
            <characterSkillCodingPHP>4</characterSkillCodingPHP>
            <characterSkillTesting>4</characterSkillTesting>
            <characterSkillStamina>60</characterSkillStamina>
            <characterSkillRequisitePro>2</characterSkillRequisitePro>
            <characterSkillEnterpriseArchitect>-
2</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>-1</characterSkillRationalRose>
            <characterSkillDelphi>6</characterSkillDelphi>
            <characterSkillEclipse>-1</characterSkillEclipse>
            <characterSkillNetBeans>3</characterSkillNetBeans>
            <characterSkillJunit>2</characterSkillJunit>
            <characterSkillTestComplet>4</characterSkillTestComplet>
            <characterPayrate>15</characterPayrate>
            <avatar>char5.swf</avatar>
            <sprite>char5.swf</sprite>
      </character>
      <character>
            <characterName>Marco Coelho</characterName>
            <characterDescription>Cargo: Estagiário
Possui dificuldades de comunicação e um conhecimento bom de padrões de
software, mas tem experiência de programação em Java por 2 anos e Pascal por
1 ano, pouca atenção aos detalhes. Possui um bom conhecimento nas ferramentas
RationalRose e NetBeans e já fez algumas vídeo aulas em EnterpriseArchitect e
Eclipse. Possui muita disposição e vontade para
trabalhar.</characterDescription>
            <characterSkillRequisites>-1</characterSkillRequisites>
            <characterSkillProject>2</characterSkillProject>
            <characterSkillCodingCpp>0</characterSkillCodingCpp>
```

```
            <characterSkillCodingJava>4</characterSkillCodingJava>
            <characterSkillCodingDelphi>0</characterSkillCodingDelphi>
            <characterSkillCodingPHP>3</characterSkillCodingPHP>
            <characterSkillTesting>-2</characterSkillTesting>
            <characterSkillStamina>70</characterSkillStamina>
            <characterSkillRequisitePro>-1</characterSkillRequisitePro>

      <characterSkillEnterpriseArchitect>1</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>2</characterSkillRationalRose>
            <characterSkillDelphi>0</characterSkillDelphi>
            <characterSkillEclipse>1</characterSkillEclipse>
            <characterSkillNetBeans>3</characterSkillNetBeans>
            <characterSkillJunit>-2</characterSkillJunit>
            <characterSkillTestComplet>-1</characterSkillTestComplet>
            <characterPayrate>6</characterPayrate>
            <avatar>char2.swf</avatar>
            <sprite>char2.swf</sprite>
      </character>
      <character>
            <characterName>Cristiane Franca</characterName>
            <characterDescription>Cargo: Estagiária
É boa com a elicitação de requisitos, mas possui pouco conhecimento de
padrões de software, tem um conhecimento bom das linguagens de programação
Java e Pascal, uma ótima atenção aos detalhes. Possui um ótimo conhecimento
na ferramenta RequisitePro, já fez vários cursos em Delphi e JUnit,
adquirindo uma boa experiência, possui uma noção de Eclipse e TestComplet.
Tem muita dedicação e disposição para o trabalho.</characterDescription>
            <characterSkillRequisites>3</characterSkillRequisites>
            <characterSkillProject>-2</characterSkillProject>
            <characterSkillCodingCpp>0</characterSkillCodingCpp>
            <characterSkillCodingJava>2</characterSkillCodingJava>
            <characterSkillCodingDelphi>2</characterSkillCodingDelphi>
            <characterSkillCodingPHP>0</characterSkillCodingPHP>
            <characterSkillTesting>4</characterSkillTesting>
            <characterSkillStamina>80</characterSkillStamina>
            <characterSkillRequisitePro>4</characterSkillRequisitePro>
            <characterSkillEnterpriseArchitect>-
2</characterSkillEnterpriseArchitect>
            <characterSkillRationalRose>-2</characterSkillRationalRose>
            <characterSkillDelphi>2</characterSkillDelphi>
            <characterSkillEclipse>1</characterSkillEclipse>
            <characterSkillNetBeans>-1</characterSkillNetBeans>
            <characterSkillJunit>2</characterSkillJunit>
            <characterSkillTestComplet>1</characterSkillTestComplet>
            <characterPayrate>9</characterPayrate>
            <avatar>char6.swf</avatar>
            <sprite>char6.swf</sprite>
      </character>
</characters>
```

Quadro 10 – Arquivo XML das personagens do jogo

APÊNDICE B – Arquivo XML das ferramentas do jogo

No Quadro 11 é apresentado o arquivo XML com as informações de nome, descrição, fase do bônus, preço e o avatar das ferramentas do jogo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Vivian
Fagundes (Fundação Universidade Regional de Blumenau - FURB) -->
<ferramentasSeRpg>
      <ferramenta>
            <nomeFerramenta>RequisitePro</nomeFerramenta>
            <descricaoFerramenta>A solução RequisitePro é uma ferramenta
para gerenciamento de requisitos e casos de uso para equipes de projeto
que desejam melhorar o entendimento dos objetivos do projeto, melhorar o
desenvolvimento colaborativo, reduzir riscos e aumentar a qualidade de
aplicações antes da entrega.</descricaoFerramenta>
            <faseBonus>0</faseBonus>
            <precoFerramenta>4724</precoFerramenta>
            <rollBonus>4</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta>
            <nomeFerramenta>EnterpriseArchitect</nomeFerramenta>
            <descricaoFerramenta>Enterprise Architect é uma ferramenta de
análise, design e desenvolvimento de aplicações em UML (Unified Modeling
Language). Pode ser utilizada por vários usuários, é baseada em Windows e
totalmente gráfica, auxilia no desenvolvimento de softwares robustos, de
fácil manutenção e produz uma documentação flexível e com alta
qualidade.</descricaoFerramenta>
            <faseBonus>1</faseBonus>
            <precoFerramenta>480</precoFerramenta>
            <rollBonus>4</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta>
            <nomeFerramenta>RationalRose</nomeFerramenta>
            <descricaoFerramenta>Rational Rose é uma ferramenta CASE que
ajuda nos processos de construção de um software profissional. Permite a
modelagem com os nove diagramas da UML, a construção de modelos de Dados
com possibilidade de exportação para construção da base de dados ou
realização de engenharia reversa de uma base de dados
existente.</descricaoFerramenta>
            <faseBonus>1</faseBonus>
            <precoFerramenta>3355</precoFerramenta>
            <rollBonus>6</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta>
            <nomeFerramenta>Delphi</nomeFerramenta>
            <descricaoFerramenta>O Delphi é largamente utilizado no
desenvolvimento de aplicações desktop, aplicações multicamadas e
cliente/servidor, compatível com os banco de dados mais conhecidos do
mercado. Como uma ferramenta de desenvolvimento genérica, o Delphi pode
ser utilizado para diversos tipos de desenvolvimento de projeto. Pode ser
usado para desenvolver aplicações que exijam tanto uma linguagem de alto
nível como também de baixo nível. Um exemplo de software conhecido que foi
desenvolvido em Delphi é o Skype.</descricaoFerramenta>
            <faseBonus>2</faseBonus>
            <precoFerramenta>1674</precoFerramenta>
            <rollBonus>6</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta>
            <nomeFerramenta>Eclipse</nomeFerramenta>
            <descricaoFerramenta>Eclipse é uma IDE de código aberto para a
```

```
construção de programas de computador. O projeto Eclipse foi iniciado na
IBM que desenvolveu a primeira versão do produto e doou-o como software
livre para a comunidade. Possui como características marcantes o uso da
SWT e não do Swing como biblioteca gráfica, a forte orientação ao
desenvolvimento baseado em plug-ins e o amplo suporte ao desenvolvedor com
centenas de plug-ins que procuram atender as diferentes necessidades de
diferentes programadores.</descricaoFerramenta>
            <faseBonus>2</faseBonus>
            <precoFerramenta>0</precoFerramenta>
            <rollBonus>3</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta>
            <nomeFerramenta>NetBeans</nomeFerramenta>
            <descricaoFerramenta>A IDE NetBeans é um ambiente de
desenvolvimento multiplataforma, uma ferramenta que auxilia programadores
a escrever, compilar, debugar e instalar aplicações, foi arquitetada em
forma de uma estrutura reutilizável que visa simplificar o desenvolvimento
e aumentar a produtividade pois reúne em uma única aplicação todas estas
funcionalidades. Totalmente escrita em Java, mas que pode suportar
qualquer outra linguagem de programação ou linguagem que desenvolva com
Swing, algumas das linguagens que o NetBeans suporta são o C, C++, Ruby,
PHP, XML e linguagens HTML.</descricaoFerramenta>
            <faseBonus>2</faseBonus>
            <precoFerramenta>0</precoFerramenta>
            <rollBonus>4</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta>
            <nomeFerramenta>Junit</nomeFerramenta>
            <descricaoFerramenta>O JUnit é um framework open-source, criado
por Eric Gamma e Kent Beck. Com JUnit, o programador tem uma ferramenta
que o ajudará a eliminar os erros de seu código de maneira mais atraente.
Pois, se criou uma forma interessante de realizar testes onde é possível a
criação de programas que realizem os testes pelo programador. É utilizando
esse conceito que JUnit permite deixar a fase de teste de unidades bem
mais agradável ao programador.</descricaoFerramenta>
            <faseBonus>3</faseBonus>
            <precoFerramenta>0</precoFerramenta>
            <rollBonus>2</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
      <ferramenta> <nomeFerramenta>TestComplet</nomeFerramenta>
            <descricaoFerramenta>TestComplete é uma ferramenta completa
para teste automatizado. Esta ferramenta cria testes automatizados
funcionais, unitários, de regressão, manuais, orientados a dados ou
objetos, de carga distribuída e HTTP, stress e de
escalabilidade.</descricaoFerramenta>
            <faseBonus>3</faseBonus>
            <precoFerramenta>3278</precoFerramenta>
            <rollBonus>4</rollBonus>
            <avatar>d</avatar>
      </ferramenta>
</ferramentasSeRpg>
```

Quadro 11 – Arquivo XML das ferramentas do jogo

APÊNDICE C– Descrição dos atributos do digrama de classe

A seguir são apresentado os atributos da classe `personagem`.

a) ausente: flag que identifica se a personagem está fora do jogo ou não em virtude da stamina;
b) avatar: caminho para identificar a imagem da personagem;
c) descPersonagem: atributo que descreve a personagem com as suas habilidades;
d) nomePersonagem: atributo do nome da personagem;
e) perCodigoCpp: perícia da personagem na linguagem de programação C++;
f) perCodigoDelphi: perícia da personagem na linguagem de programação Delphi;
g) perCodigoJava: perícia da personagem na linguagem de programação Java;
h) perCodigoPhp: perícia da personagem na linguagem de programação PHP;
i) perEnterpriseArchitect: perícia da personagem no uso da ferramenta Enterprise Architect;
j) perFerramentaDelphi: perícia da personagem no uso da ferramenta Delphi;
k) perFerramentaEclipse: perícia da personagem no uso da ferramenta Eclipse;
l) perFerramentaNetBeans: perícia da personagem no uso da ferramenta NetBeans;
m) perJunit: perícia da personagem no uso da ferramenta JUnit;
n) perProjeto: perícia da personagem na elaboração de projetos;
o) perRationalRose: perícia da personagem no uso da ferramenta Rational Rose;
p) perRequisitePro: perícia da personagem no uso da ferramenta RequisirtPro;
q) perRequisitos: perícia da personagem com elicitação de requisitos;
r) perStamina: valor numérico que define a quantidade da *Stamina* da personagem;
s) perTestComplet: perícia da personagem no uso da ferramenta TestComplet;
t) perTeste: perícia da personagem em normal de qualidade;
u) salario: custo diário que é pago para manter a personagem trabalhando;
v) sprite: caminho para identificar a foto da personagem;
w) staminaAtual: identificador da Stamina atual da personagem.

A seguir são apresentado os atributos da classe `ferramenta`.

a) avatarFerramenta: caminho para identificar a imagem da ferramenta;
b) descFerramenta: atributo que descreve a ferramenta;
c) emJogo: flag que define se a ferramenta foi comprada pelo jogador;

d) emUso: flag que define se a ferramenta está sendo utilizada por uma personagem;
e) faseBonus: atributo que indica em qual fase a ferramenta fornece o bônus;
f) nomeFerramenta: atributo do nome da ferramenta;
g) precoFerramenta: custo da ferramenta para ser comprada;
h) rollBonus: valor do random que é feito para saber o bônus que a ferramenta agrega.

A seguir são apresentado os atributos da classe fase.

a) fases: atributo do nome da fase;
b) preReqAtividade: atributo que indica o pré-requisito da fase;
c) progressoParcial: atributo que indica a escala de progressão da fase.

A seguir são apresentado os atributos da classe projeto.

a) deadlineProjeto: atributo que indica a quantidade de dias extra após o prazo estipulado para ser realizada a entrega do software;
b) descProjeto: atributo que descreve o projeto;
c) difProjeto: valor da dificuldade do projeto;
d) nomeProjeto: atributo do nome da ferramenta;
e) orcProjeto: atributo que define o orçamento liberado para o desenvolvimento do projeto;
f) orcProjetoAtual: atributo que atualiza o orçamento do jogador durante o jogo;
g) tempoProjeto: quantidade de dias permitidos para a conclusão do software;
h) tempoProjetoAtual: atributo que atualiza os dias que se passaram no jogo.

A seguir são apresentado os atributos da classe modeloprojeto.

a) atividadeModulo: atributo que grava a ultima fase que teve progresso no módulo;
b) difModulo: atributo que contém a dificuldade do módulo, essa dificuldade é utilizada nos testes do d20 System;
c) nomeModulo: atributo do nome do módulo;
d) tamModulo: tamanho do módulo, esse atributo é utilizado para os calculos de progressão das fases do jogo.

A seguir são apresentado os atributos da classe alocacaopersonagem.

a) progresso: progressão da personagem em um determinado módulo.

ANEXO A – OPEN GAME LICENSE

The following text is the property of Wizards of the Coast, Inc. and is Copyright 2000 Wizards of the Coast, Inc ("Wizards"). All Rights Reserved.

1. Definitions: (a)"Contributors" means the copyright and/or trademark owners who have contributed Open Game Content; (b)"Derivative Material" means copyrighted material including derivative works and translations (including into other computer languages), potation, modification, correction, addition, extension, upgrade, improvement, compilation, abridgment or other form in which an existing work may be recast, transformed or adapted; (c) "Distribute" means to reproduce, license, rent, lease, sell, broadcast, publicly display, transmit or otherwise distribute; (d)"Open Game Content" means the game mechanic and includes the methods, procedures, processes and routines to the extent such content does not embody the Product Identity and is an enhancement over the prior art and any additional content clearly identified as Open Game Content by the Contributor, and means any work covered by this License, including translations and derivative works under copyright law, but specifically excludes Product Identity. (e) "Product Identity" means product and product line names, logos and identifying marks including trade dress; artifacts; creatures characters; stories, storylines, plots, thematic elements, dialogue, incidents, language, artwork, symbols, designs, depictions, likenesses, formats, poses, concepts, themes and graphic, photographic and other visual or audio representations; names and descriptions of characters, spells, enchantments, personalities, teams, personas, likenesses and special abilities; places, locations, environments, creatures, equipment, magical or supernatural abilities or effects, logos, symbols, or graphic designs; and any other trademark or registered trademark clearly identified as Product identity by the owner of the Product Identity, and which specifically excludes the Open Game Content; (f) "Trademark" means the logos, names, mark, sign, motto, designs that are used by a Contributor to identify itself or its products or the associated products contributed to the Open Game License by the Contributor (g) "Use", "Used" or "Using" means to use, Distribute, copy, edit, format, modify, translate and otherwise create Derivative Material of Open Game Content. (h) "You" or "Your" means the licensee in terms of this agreement.

2. The License: This License applies to any Open Game Content that contains a notice indicating that the Open Game Content may only be Used under and in terms of this License. You must affix such a notice to any Open Game Content that you Use. No terms may be

added to or subtracted from this License except as described by the License itself. No other terms or conditions may be applied to any Open Game Content distributed using this License.

3.Offer and Acceptance: By Using the Open Game Content You indicate Your acceptance of the terms of this License.

4. Grant and Consideration: In consideration for agreeing to use this License, the Contributors grant You a perpetual, worldwide, royalty-free, non-exclusive license with the exact terms of this License to Use, the Open Game Content.

5.Representation of Authority to Contribute: If You are contributing original material as Open Game Content, You represent that Your Contributions are Your original creation and/or You have sufficient rights to grant the rights conveyed by this License.