UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM PLUG-IN QUE INTEGRA UM SIMULADOR DE PROJETOS A UMA FERRAMENTA DE GERENCIAMENTO DE PROJETOS.

TRABALHO DE GRADUAÇÃO

Aluno: Thierry da Silva Araujo (tsa@cin.ufpe.br)
Orientador: Hermano Perrelli de Moura (hermano@cin.ufpe.br)

ABRIL DE 2007

“O único local aonde o sucesso vem antes do trabalho é no dicionário.”

Albert Einstein.

Dedico aos meus pais, Luiz e Rozinha, pelo amor e dedicação.
A minha Irmã, Thienny, pela amizade.
A minha namorada, Milena, pelo amor e carinho.
E a todos os meus amigos, pela consideração e o apoio sempre.

Agradecimentos

Gostaria de agradecer primeiro a Deus, por deixar eu existir no mundo e proporcionar coisas boas e até mesmo experiências ruins para que eu possa aprender mais e mais.

Agradecer a minha família, em especial aos meus pais, Luiz e Rozinha, e irmã, Thienny, que com o apoio, compreensão, carinho, amor deles foi muito importante para que chegasse nessa etapa da vida.

Agradecer a minha namorada, pelo amor, paciência, estímulo e muito mais que também me impulsionou nesta jornada durante o tempo que estamos juntos.

Aos grandes amigos da juventude que até hoje me comunico e me dão votos de estímulo. E também aos amigos conquistados durante esta fase muito importante.

Agradeço ao meu orientador Hermano Perrelli, pela oportunidade e ajuda para que o mesmo tenha chegado a sua etapa final.

Ao amigo André Felipe por ter me estimulado na criação deste trabalho. Ao Marcelo Guedes por ter me ajudado no desenvolvimento dele e também a toda equipe do projeto SmartSim.

E por fim a todos que por ventura eu tenha deixado de lembrar no momento mas que são pessoas que foram muito importantes na minha vida.

A vocês,

Muito obrigado!

Resumo

Na atualidade a disciplina de gerenciamento de projetos se tornou muito importante em diversos ramos de atividade. Na engenharia de software esta atividade é primordial para se obter um produto de qualidade e que atenda aos requisitos impostos pela necessidade do cliente.

Com o forte crescimento da adoção da gerência de projetos nas organizações de desenvolvimento de software, ficou importante se ter um estudo maior nessa área e um melhor treinamento para os gerentes de projeto. Então surgiram os simuladores empresarias para ajudar no entendimento de diversas áreas, inclusive no gerenciamento de projetos. Neste contexto surgiu o Virtual Team, um jogo sério para o treinamento de gerentes de projeto.

A proposta deste trabalho é criar uma ferramenta que possa captar dados do mundo real, em ferramentas de gerenciamento de projeto, e criar um arquivo que possa ser simulado nesse jogo e então criar uma ambiente mais interessante no jogo.

Palavras chave: Gerência de projetos, jogos de negócio, simuladores empresariais, plug-in, add-in, Virtual Team, Microsoft® Project.

Sumário

1 Introdução 9

2 Gerenciamento de Projetos de Software 11
2.1 Definições e o PMBOK™ *Guide* 11
2.1.1 Grupos de Processo 16
2.1.2 Áreas de Conhecimento 20
2.2 O PMI e o Panorama do Gerenciamento de Projetos 22
2.3 Gerenciamento de Projetos de Software 24

3 Simuladores e Jogos de Negócio 27
3.1 Fundamentos teóricos 27
3.2 Dinâmica de sistemas 31
3.2.1 Modelos da Dinâmica de Sistemas 33
3.3 Áreas de aplicação e uso de Jogos de simulação 36
3.3.1 Formação acadêmica e treinamento empresarial 36
3.3.2 Uso de jogos de simulação 38

4 O Jogo Virtual Team 42
4.1 Arquitetura do Virtual Team 42

5 O *Plug-in* 48
5.1 Motivação 48
5.2 Desenvolvimento de *Add-in* para o Microsoft® Project 49
5.3 Esquema do arquivo de *Level* do Virtual Team 52
5.4 Requisitos e Arquitetura do *Add-in* 55
5.4.1 Requisitos 55
5.4.2 Wizard 57
5.4.3 Arquitetura 58
5.5 Resultados 67

6 Conclusões 72
6.1 Dificuldades encontradas 73
6.2 Trabalhos futuros 73

7 Referências Bibliográficas 75

Lista de Figuras

Figura 2.1-1 Variáveis conflitantes [GUEDES 2006]. 14

Figura 2.1-2 Áreas de conhecimento necessárias à equipe de projeto [PMBOK 2004]... 15

Figura 2.1-3 Mapeamento entre os processos de gerenciamento de projeto e o ciclo PDCA [PMBOK 2004]. 17

Figura 2.1-4 Interação entre os grupos de processo [PMBOK 2004]..... 20

Figura 2.2-1 Crescimento das Certificações PMP feitas em 2005 [PMIREL 2005]. 24

Figura 3.2-1 - Diagrama de causa e efeito [GUEDES 2006]..... 33

Figura 3.2-2 - Simbologia do diagrama repositório e fluxo [GUEDES 2006]. 35

Figura 3.3-1 - Arquitetura do SimSE [GUEDES 2006]. 40

Figura 3.3-2 Estrutura do jogo The Incredible Manager [GUEDES 2006]..... 41

Figura 4.1-1 Visão geral dos pacotes da arquitetura do jogo [GUEDES 2006]. 44

Figura 4.1-2 - Classes principais do Virtual Team [GUEDES 2006]..... 45

Figura 4.1-3 - Laço principal responsável pela simulação [GUEDES 2006]. 47

Figura 5.4-1 - Diagrama simples mostrando o relacionamento entre as camadas do MVC [MVCEN 2007]..... 58

Figura 5.4-2 - Padrão de projeto Composite [Composite 2007]..... 60

Figura 5.4-3 - Classes da camada Model do add-in..... 60

Figura 5.4-4 - Classes da camada de visão 61

Figura 5.4-5 - Tela da primeira etapa do *wizard*. Informações sobre o Projeto. 62

Figura 5.4-6 - Segunda etapa do *wizard*. Informações sobre a equipe, neste caso as remunerações. 63

Figura 5.4-7 - Terceira etapa do *wizard.* O plano de comunicação do projeto a ser simulado. 64

Figura 5.4-8 - Tela de boas vindas do Virtual Team Add-in. 65

Figura 5.4-9 - Tela de finalização do wizard para a geração do arquivo de simulação. ... 65

Figura 5.4-10 - Classes da camada Controller 66

Figura 5.5-1 - Botão de inicialização do *Add-in* no MS Project. 67

Figura 5.5-2 - Jogo Virtual Team *Add-in* para Microsoft® Project. 68

Figura 5.5-3 - Descrição do projeto no jogo Virtual Team. 69

Figura 5.5-4 - As atividades do projeto simulado no jogo. 70

Figura 5.5-5 - Plano de comunicação no jogo. Somente a descrição do plano. 71

1 Introdução

O gerenciamento de projetos é uma atividade muito importante para diversas áreas. Ela surgiu há bastante tempo e persiste até hoje com um crescimento muito grande. Na área de desenvolvimento de software, essa atividade se tornou muito importante por auxiliar no sucesso do projeto controlando variáveis como: escopo, tempo, custo e qualidade.

Já há uma especificação de um conjunto de procedimentos que visam padronizar a teoria sobre a gerência de projetos [MARTINS]. Está especificação foi criada pelo PMI (Project Management Institute), instituição internacional sem fins lucrativos que agrupa profissionais da área de Gerência de Projetos. O PMBOK™ *Guide* (*Project Management Body Of Knowledge*) é o documento criado pelo PMI que registra a teoria padronizada por essa instituição [PMBOK].

Contudo, a evolução das atividades, bem como o surgimento de outras novas necessidades, se fez necessário a geração de aprimoramentos, visando alcançar uma melhor qualidade nessa atividade. Com isso grupos de pesquisa buscaram essas novas evoluções, principalmente no que diz respeito à capacitação de profissionais [GUEDES] e o melhoramento da atividade de planejamento de projeto.

Nessa busca, um simulador de gerência de projetos foi proposto e criado para suprir essas necessidades. O SmartSim [SMARTSIM, 2006] é um projeto *open source* para o desenvolvimento de um *framework* para a construção de jogos de negócio que utilizem atores sintéticos que representam personalidades humanas. Como primeira aplicação desse *framework* foi criado o *Virtual Team*, jogo sério de simulação de gerência de projetos para capacitação de gerentes de projetos de software.

Esse jogo simula uma equipe de projeto de software trabalhando em um escopo definido e seguindo as práticas de algumas áreas definidas no PMBOK™ *Guide* [PMBOK]. Esse escopo é pré-definido e pode ser alterado por um editor do jogo, porém não há nenhuma integração desse simulador com ferramentas de gerenciamento de projetos presente no mercado, como o *Microsoft Project* [PROJECT], o Rational Portfolio Management da IBM ou o dotProject (*open source*) [DOTPROJECT], e que possa interagir com o simulador passando dados de projetos reais para serem simulados,

bem como devolver ao gerenciador de projetos o resultado da simulação como uma "previsão" do que pode ocorrer no andamento do projeto.

A proposta deste trabalho é construir um *plug-in* para um sistema de gerenciamento de projetos que irá integrá-lo ao simulador de ambiente de gerenciamento de projetos Virtual Team. O *plug-in* funcionará captando e formatando as informações importantes para o processo de simulação para que o simulador leia e possa executar essas informações. Com isso, se ter a possibilidade de executar no simulador dados do mundo real e criar um ambiente mais interessante no jogo é muito interessante para capacitar melhor os gerentes de projetos possibilitando o estudo desses casos reais no simulador.

2 Gerenciamento de Projetos de Software

A atividade de gerenciamento de projetos vem sendo desenvolvida desde a década de 60. E foi na indústria bélica e aeroespacial que surgiu a disciplina de Gerência de projetos, que logo depois foi adotada por outras áreas de atuação como, por exemplo, na engenharia civil e posteriormente em outras engenharias. Atualmente esse conceito é aplicado em diversas outras áreas como: economia e política [MARTINS 2005].

Com a Revolução Industrial o mercado passou a produzir e comercializar em escala mundial, então houve uma profunda alteração da estrutura econômica do ocidente e consequentemente o desenvolvimento do capitalismo industrial [GUEDES 2006]. Com isso as relações de produção foram alteradas e passaram a ficar mais exigentes, a exemplo disso foi o surgimento de teorias de qualidade japonesas como a TQM (*Total Quality Management*), onde o sucesso do negócio tinha como elemento o controle do processo produtivo [GUEDES 2006].

Então o desenvolvimento de produtos e serviços com requisitos levantados junto ao cliente teve que ser realizado, obrigatoriamente, em tempo hábil e dentro de um custo previsto. Com isso, várias técnicas para o aprimoramento no andamento do projeto foram surgindo, como é o caso do Gráfico de *Gantt* que surgiu na época da I Guerra Mundial, criada por *Henry Gantt*, e é utilizado até hoje com pequenas modificações.

A seguir serão feitas as seguintes descrições a respeito da atividade de gerenciamento de projetos, na Seção 2.1, *PMBOK™ Guide* e algumas definições sobre gerência de projetos, na Seção 2.2, o que é o PMI e sua importância e na Seção 2.3, será abordado algumas particularidades gerenciamento de projetos de software.

2.1 Definições e o PMBOK™ *Guide*

Projetos são ações temporárias que têm a finalidade de se produzir um produto, serviço ou resultado único [PMBOK 2004]. Unitário pelo fato de serem exclusivos, sempre um diferente do outro e a presença de algum elemento repetido não elimina a unicidade do trabalho do projeto. São ditos temporais no que diz respeito ao fato de terem

início e fim bem definidos, ou seja, quando os seus objetivos tiverem sido alcançados, que fique claro que não se poderá mais atingir seus objetivos ou que o mesmo seja encerrado [PMBOK 2004].

Em um projeto, as características de um produto ou serviço são determinadas continuamente passando aos poucos por um refinamento durante o decorrer do projeto, a essa definição se dá o nome de *elaboração progressiva*. No início do projeto essas características são bem amplas e depois vai sendo mais bem detalhadas como o decorrer do tempo, apesar disso o trabalho do projeto se mantém constante [HELDMAN 2004].

Nas organizações os trabalhos realizados podem ser definidos como sendo dois distintos, projetos ou operações, que apesar de serem diferentes eles se sobrepõem algumas vezes. Esses dois tipos de trabalho têm algumas características em comum, no entanto eles se diferem principalmente pelo fato de que os projetos são finitos (temporários) e únicos e as operações se repetem continuamente [PMBOK 2004].

Existem diversas pessoas com diferentes interesses na execução ou nos resultados de um projeto, desde o executor de uma tarefa determinada no escopo do projeto, passando pelos beneficiados (cliente, executivo, diretor de empresa, etc.), pelos que o projeto vai gerar de retorno (funcional, financeiro ou outros) e até mesmo pessoas ligadas aos integrantes do projeto. Essas pessoas são identificadas pelo termo em inglês *Stakeholder* e o mais correto é que sejam identificadas as mais importantes delas no início do projeto, pois caso uma delas, ou a sua função, sejam esquecidos, o projeto pode ser posto e perder. Algumas vezes os interesses desses stakeholders podem estar conflitando e é responsabilidade do gerente de projeto identificar esses conflitos e soluciona-los [HELDMAN 2004].

O gerenciamento de projetos engloba diversas ferramentas e técnicas, utilizadas para definir, organizar e controlar o andamento do projeto. Os gerentes de projeto são as pessoas responsáveis por administrar os processos envolvidos nos projetos, bem como aplicar essas ferramentas e técnicas [HELDMAN 2004]. Esse gerente tem as seguintes responsabilidades:

- Identificar necessidades;
- Definir objetivos claros e atingíveis;

- Balancear as demandas conflitantes de qualidade, escopo, tempo e custo;
- Adaptar as expectativas, os planos e a abordagem às diferentes preocupações das diversas áreas.

Os gerentes de projeto são profissionais que têm conhecimento mais geral, e que têm em seu repertório uma séria de competências. Apesar de poderem atuar em uma variedade de campos sua especialidade é a de resolver problemas. E seus conhecimentos técnicos específicos não são pré-requisitos para o gerenciamento de projetos. Algumas competências gerais constituem os fundamentos de boas práticas no gerenciamento de projetos [HELDMAN 2004]. Abaixo serão enumeradas algumas delas:

1. Competências de comunicação;
2. Aptidões organizacionais;
3. Habilidades para a elaboração de orçamentos;
4. Solução de problemas;
5. Competências de negociação e influência;
6. Competências de liderança;
7. Habilidades de formação de equipe e recursos humanos.

Várias são as atividades que compõem o gerenciamento de projetos, incluindo planejar, executar o planejado e acompanhar o progresso e o desempenho. A parte de planejamento se torna umas das mais importantes, pois é a que conduz o restante do processo de gerenciamento. Existem algumas restrições que todos os projetos têm em comum, elas se dispõem em um conjunto de três: tempo, custo e escopo [HELDMAN 2004]. Pode acontecer uma restrição, duas ou até mesmo as três dependendo do projeto. É comum se encontrar projetos com orçamento ou prazos fixos, ou os dois ao mesmo tempo, e a exigência com a completude do escopo é muito alta. Cabe ao gerente de projetos equilibrar esse trio de restrições, além de procurar atender ou exceder as expectativas dos stakeholders. Além dessas três variáveis existe mais uma que entra em conflito com essas três que é a qualidade do projeto que é um fator muito importante para

que o projeto saia bem sucedido. A figura abaixo mostra a relação entre as variáveis conflitantes descritas anteriormente.

Figura 2.1-1 Variáveis conflitantes [GUEDES 2006].

As premissas de um projeto são tudo aquilo que se acredita ser verdade. Por exemplo, a disponibilidade de materiais em uma obra [HELDMAN 2004]. Cada projeto terá seu conjunto de premissas a serem levadas em consideração e devem ser identificadas e documentadas para posterior utilização ao longo do andamento do projeto.

Todos os projetos possuem um ciclo de vida parecido, sejam grandes ou pequenos. Em geral eles passam por uma etapa inicial, uma outra intermediária e por fim uma etapa final, esse número de etapas depende da complexidade do projeto. Esse grupo de fases pelo qual o projeto atravessa que é chamado de *ciclo de vida do projeto.* O final de cada fase do projeto pode ser identificado por uma ou mais entregas específicas. As entregas são elementos tangíveis que podem ser avaliados e comprovados com facilidade.

As atividades que serão executadas em um projeto são chamadas pelo PMBOK™ *Guide* de processos. Estes são formados por ações que são executadas para se chegar a um subconjunto de resultado, produto ou serviço [GUEDES 2006]. Os processos podem ser de gerenciamento de projeto, que são comuns à maioria dos projetos, e estão relacionados às fases de iniciar, planejar, executar, controlar e encerrar um projeto, no entanto essas fases têm interações complexas normalmente, ou processos orientados ao produto que definem como o produto do projeto é feito e variam de acordo com a área de aplicação.

O PMBOK™ *Guide – Project Management Body of Knowledge* – teve sua primeira versão desenvolvida e publicada no ano de 2000 pelo PMI num esforço de defender a atividade de Gerente de Projetos como uma profissão com seu currículo próprio [GUEDES 2006]. Em 2004 foi lançado mais uma versão do guia com principais modificações na estrutura do guia, algumas seções alteradas, bem como os nomes dos processos e suas localizações dentro das áreas de conhecimento [PMBOK 2004]. Mesmo criando um corpo de conhecimento sobre gerenciamento de projetos, o guia mostra que outro conjunto de habilidades necessárias ao gerente de projeto. Na Figura 2.1-2 pode se observar os conhecimentos contidos no PMBOK™ e os necessários a plena atividade de gerente.

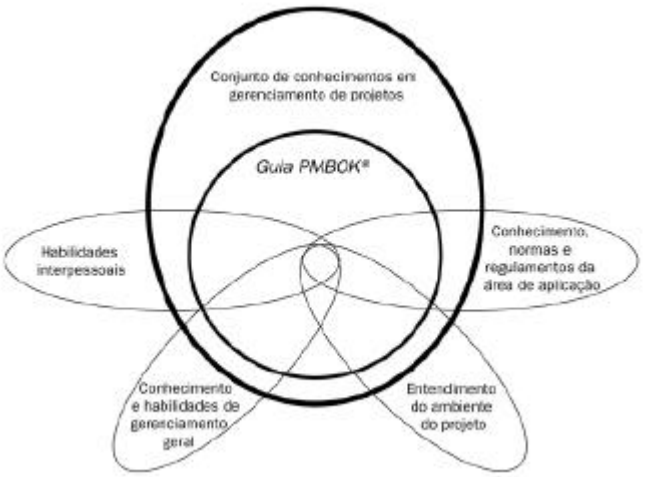

Figura 2.1-2 Áreas de conhecimento necessárias à equipe de projeto [PMBOK 2004].

O PMBOK™ *Guide* organiza seus processos em duas formas diferentes, uma delas orienta que esses processos estão em cinco grupos. Esses grupos, exceto o de iniciação, são constituídos por outros processos individuais e definem as fases de andamento do projeto, como relacionado abaixo:

- Grupo de processos de iniciação
- Grupo de processos de planejamento
- Grupo de processos de execução
- Grupo de processos de monitoramento e controle

- Grupo de processos de fechamento

A outra forma de organização dos processos de gerenciamento de projetos é a separação em áreas de conhecimento. O guia relaciona e descreve nove áreas do conhecimento em gerência de projetos. Abaixo segue listadas as áreas:

- Gerenciamento da integração do projeto
- Gerenciamento do escopo do projeto
- Gerenciamento de custos do projeto
- Gerenciamento do tempo do projeto
- Gerenciamento da qualidade do projeto
- Gerenciamento dos recursos humanos do projeto
- Gerenciamento das comunicações do projeto
- Gerenciamento dos riscos do projeto
- Gerenciamento das aquisições e sub-contratações do projeto

O gerenciamento de projetos possui um contexto maior que inclui outras áreas como o gerenciamento de programas, o gerenciamento de portfólio e o escritório de projetos. Programas são grupos de projetos com a mesma relação e que gerenciados fazendo uso das mesmas técnicas e coordenados. Portfólio é um conjunto de projetos ou programas que visa facilitar a administração de todos os trabalhos e assim chegar aos objetivos do negócio da organização. Um escritório de projetos (PMO) é uma estrutura organizacional que tem como objetivo centralizar e coordenar o gerenciamento dos projetos na companhia, o PMO também pode supervisionar programas.

2.1.1 Grupos de Processo

Os processos executados em um projeto são organizados ao longo do tempo decorrente desse projeto em um ciclo de vida. No passar do tempo do projeto ele pode atravessar diversos estágios de maturidade onde o seu desenvolvimento é divido em fases. O

modelo de processo descrito no PMBOK™ *Guide* é baseado no modelo PDCA (*Plan, Do, Check and Act*) [GUEDES 2006], no modelo criado pelo PMI essas ações podem se repetir nas várias fases do projeto, o mapeamento entre os grupos de processos de gerenciamento de projetos e o ciclo PDCA pode ser observado na Figura 2.1-3. O PDCA – também chamado de ciclo de Shewhart ou ciclo de Deming – tem por princípio tornar ágeis processos de gestão [PDCA 2006].

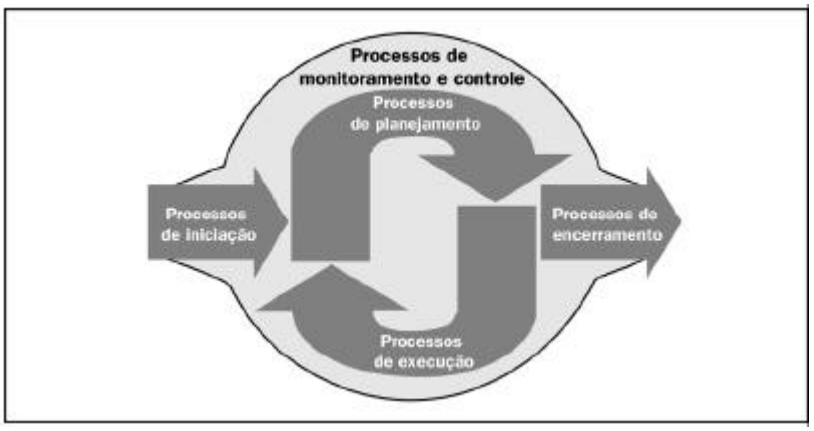

Figura 2.1-3 Mapeamento entre os processos de gerenciamento de projeto e o ciclo PDCA [PMBOK 2004].

Contudo foi observado que os processos de gerenciamento de projetos são processos que dão suporte ao desenvolvimento do produto, que têm por objetivo fazer com que o projeto finalize com sucesso, ou seja, esteja no escopo correto, no prazo, custo e qualidade previstos. Com isso, os processos de gerenciamento de projeto não se confundem com o processo específico de desenvolvimento dos produtos resultantes do projeto.

O PMBOK define cinco grupos de processos, que possuem dependência clara e são executados nos projetos em uma mesma seqüência, independentes das áreas de aplicação ou do foco do setor aplicado [PMBOK 2004].

O grupo de processo de iniciação é formado por processos que propiciam a autorização para iniciar um novo projeto ou uma fase de um projeto. Esses processos de iniciação são normalmente realizados fora do escopo de controle do projeto, baseado no processo organizacional, e fornecem subsídios para a iniciação do projeto. Com isso os

limites do projeto podem ficar menos evidentes para as entradas iniciais. Nos processos desse grupo são desenvolvidas as descrições claras dos objetivos do projeto, inserindo as razões pela qual um projeto específico possui as melhores alternativas para satisfazer os requisitos. Além disso, nos documentos gerados pelos processos desses grupos está presente a descrição básica do escopo do projeto, das entregas, da duração do projeto e uma previsão dos recursos necessários [PMBOK 2004], o principal artefato desenvolvido que possui essas informações é o *Project Charter* ou Termo de Abertura do Projeto. Nos projetos com várias fases, os processos de iniciação são desenvolvidos em fases subseqüentes para validar todas as decisões e premissas. Também faz parte do Termo de abertura qual o Gerente de Projetos, caso ele ainda não tenha sido escolhido, que irá participar do projeto e as premissas e restrições iniciais também são documentadas. Então quando ele é aprovado o projeto é oficialmente autorizado.

O grupo de processo de planejamento é utilizado pela equipe do projeto para planejar e gerenciar de modo que ele saia bem sucedido. Esse grupo de processo auxilia na coleta de diversas informações em várias fontes diferentes, muitas vezes, umas mais completas e confiáveis que outras [PMBOK 2004]. O principal artefato produzido nessa etapa é o plano de projeto. Os processos deste grupo ajudam a identificar, definir ou maturar o escopo, custo e cronograma de atividades que ocorrem no projeto. No momento em que se for identificando novas informações do projeto, dependências, requisitos, riscos, oportunidades, as premissas e restrições adicionais serão identificadas e resolvidas.

Durante a execução do projeto várias mudanças podem ser solicitadas. Essas solicitações afetam significantemente partes do plano de gerenciamento de projetos, com isso uma freqüente atualização do plano de projeto dá uma maior precisão em relação aos diversos fatores de risco do projeto, como cronograma, custo e recursos necessários. A esse detalhamento progressivo se dá o nome de “planejamento em ondas sucessivas”, indicando um processo iterativo e contínuo [PMBOK 2004]. As partes interessadas no projeto devem ser levadas em consideração em toda a etapa de criação do plano de gerenciamento do projeto, por possuírem habilidades e conhecimento que podem ser importantes para o desenvolvimento desse plano. Porém, para não se ficar planejando

indefinidamente, a equipe deve utilizar os procedimentos organizacionais para determinar o término da etapa de planejamento.

O grupo de processos de execução contém as atividades que irão desenvolver o que foi planejado anteriormente, a fim de se cumprir os requisitos do projeto. É nesta fase que há a coordenação das pessoas e dos outros recursos. Durante a execução, algumas variações podem exigir algum replanejamento, que podem afetar diretamente o plano de projeto ou exigir apenas uma análise do mesmo. Após o resultado da análise, algumas mudanças podem ser solicitadas, e se aprovadas modificariam o plano de projeto, estabelecendo uma nova linha de base [PMBOK 2004].

O grupo de processos de controle identifica os procedimentos necessários para a visualização do projeto em andamento, procurando em tempo hábil os problemas que podem ocorrer e então aplicar as operações corretivas [GUEDES 2006]. Um dos principais aproveitamentos tirados desse grupo de processos é a observação e medição do desempenho do projeto, verificando as variações relativas ao plano de projeto. Esse grupo de processo pode incluir, por exemplo:

- O controle das atividades que estão decorrendo em relação do plano de projeto;
- Avaliar os fatores que podem dificultar o controle integrado de mudanças.

O grupo de processo de encerramento tem por objetivo fechar formalmente o projeto ou uma fase, finalizando suas atividades, dependências e entregando os seus produtos. Inclui finalizar os contratos formais e devolver os recursos utilizados.

Apesar de alguns artefatos produzidos por um grupo de projetos serem insumos para outro grupo, os grupos de processo não estão separados propriamente. A Figura 2.1-4 mostra como os grupos de processo se sobrepõem durante todo o tempo do projeto.

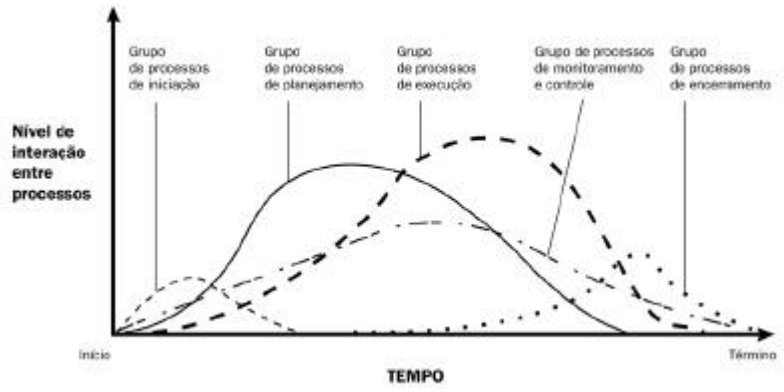

Figura 2.1-4 Interação entre os grupos de processo [PMBOK 2004].

2.1.2 Áreas de Conhecimento

O PMBOK™ também organiza os processos de gerenciamento de projetos em nove áreas do conhecimento. Enquanto que a visão dos processos em grupos se torna mais dinâmica, a divisão desses processos em áreas fornece uma estrutura estática desses processos [GUEDES 2006]. Abaixo segue uma breve descrição, segundo o PMPBOK™ *Guide* 3ª edição, de cada área de conhecimento.

- Integração: esta área de conhecimento possui o conjunto de processos e atividades necessários para identificar, definir, combinar, unificar e coordenar os diversos outros processos de gerenciamento de projeto. Para atender bem as necessidades das partes interessadas a integração tem as características de unificação, consolidação, articulação e ação integradora, que são muito importantes para a finalização do projeto. Fica mais evidente a importância da integração nos processos individuais, pois existem os que participam de áreas diferentes e são utilizados para se obter um resultado em comum.

- Escopo: área que possui os processos necessários para garantir que todo o trabalho necessário esteja inserido no projeto para que seja terminado com sucesso. Ele trata principalmente da definição e controle do que está e do que não

está incluído no projeto. Todos os processos dessa área interagem entre si e com os de outras áreas de conhecimento, além disso, podem necessitar do empenho de uma ou mais pessoas para finalizarem os artefatos a serem produzidos.

- Tempo: o gerenciamento de tempo inclui todos os processos necessários para terminar o projeto no prazo. Esta área inclui, como os mais importantes, os processos de estimativas de prazo, sequenciamento de atividades, definição e controle de cronograma e outros.

- Custo: área do conhecimento que possui os processos necessários para o gerenciamento dos custos do projeto. Os processos incluem a estimativa de custo, orçamentação e o controle dos custos.

- Qualidade: os processos que determinam as responsabilidades, os objetivos e as políticas de qualidade, a fim de atender as necessidades do projeto estão inclusos na área de gerenciamento da qualidade do projeto. Suas atividades estão relacionadas ao planejamento da qualidade, a garantia da qualidade e o controle de qualidade. O gerenciamento da qualidade do projeto aborda não só o projeto, mas também o produto desse projeto, porém o gerenciamento da qualidade do projeto se aplica a qualquer projeto, enquanto que as métricas do produto são específicas do domínio da aplicação.

- Recursos Humanos: a equipe que compõe o projeto é organizada e gerenciada segundo os processos da área de gerenciamento de recursos humanos. Apesar de existirem equipes com pessoas que possuem atribuições e responsabilidades definidas, é interessante que essas pessoas participem da etapa de planejamento do projeto para enriquecer com mais especialização e aumentar o comprometimento com o projeto. O tipo de pessoas e a quantidade em uma equipe muda durante a execução do projeto.

- Comunicação: essa área do conhecimento possui os processos que são usados para a geração, coleta, distribuição, armazenamento, recuperação e destinação final das informações sobre o projeto de uma forma adequada e admissível. Os processos fornecem a ligação entre as pessoas integrantes do projeto e a informação, para que todos tenham uma boa comunicação. Todos devem entender a importância das comunicações no projeto.

- Riscos: o gerenciamento de riscos do projeto possui os processos para a realização de identificação, análise, respostas, monitoramento e controle e planejamento do gerenciamento de riscos em um projeto. O gerenciamento de riscos tem por objetivos aumentar a probabilidade e o impacto de eventos positivos e diminuir a probabilidade e o impacto de eventos contrários ao projeto.

- Aquisições: contempla os processos de compra ou aquisição dos produtos, serviços ou resultados externos a equipe do projeto para auxiliar na realização do trabalho. Inclui processos de gerenciamento de contratos de aquisição, bem como os contratos de uma empresa externa que irá adquirir o projeto.

2.2 O PMI e o Panorama do Gerenciamento de Projetos

Fundado em 1969 por cinco voluntários nos EUA, o Instituto em Gerência de Projetos (PMI® – *Project Management Institute*) é uma fundação sem fins lucrativos que tem como objetivo promover e aumentar o conhecimento sobre gerenciamento de projetos [PMIPE 2006], bem como melhorar o desempenho dos profissionais de gerência de projetos, criando treinamentos, certificando e fazendo publicações na área.

Dentre as criações do PMI, está o *código de ética* que deve ser seguido pelos profissionais certificados pelo instituto (PMP™ - *Project Management Professional*). A certificação é obtida mediante aprovação em uma prova promovida pelos provedores de treinamento (*REPs - Registered Education Providers*) registrados junto ao PMI, porém a avaliação não é uma condição única de qualificação, pois também existe a experiência e crescente evolução, que devem ser comprovados.

Os *REPs* são empresas de mercado que têm o objetivo de prover a educação para os profissionais de gerenciamento de projetos. Além desses centros de treinamento e avaliação, o PMI™ estimula e mantém convênios com instituições que têm programas de graduação e pós-graduação [GUEDES 2006].

O PMI™ conta atualmente com mais de 170.000 membros em 150 [PMISP 2006] países que podem dispor de diversas publicações feitas ao longo da história do PMI. Nos anos setenta o instituto publica a primeira edição do *Project Management Quarterly (PMQ)* que depois foi renomeado para *Project Management Journal (PMJ)* [PMIPE 2006]. Atualmente o PMI produz publicações periódicas como: *PM Network*, *Project Management Journal, PMI Today*. Com isso o PMI é classificado hoje como o maior criador de publicações na área de gerenciamento de projetos e seu título publicado mais conhecido e utilizado é o PMBOK™ *Guide*, que reúne o corpo de conhecimento necessário aos gerentes de projeto para desenvolver a atividade [GUEDES 2006]. O PMI também desenvolveu outras extensões do PMBOK para áreas específicas como: Governo, Construção e U.S. DoD (governo dos Estados Unidos). Além disso, outras padronizações foram criadas pelo PMI e disponibilizadas aos seus associados [PMISP 2006], são elas:

- PMCDF (Project Manager Competency Development Framework)
- OPM3 (Organizational Project Management Maturity Model)
- EVM (Practice Standard for Earned Value Management)
- WBS(Practice Standard for WBS - Wok breakdown Structure)
- PPMS (Program and Portfolio Management Standards)

Os membros do PMI podem se filiar em três componentes distintos: nos *Chapters*, nos Grupos de Interesses Específicos e nos *Colleges*. Os *Chapters* são divisões locais organizadas para difundir o conhecimento sobre gerência de projetos na região onde atua, realizando seminários e reuniões periódicas para os associados poderem trocar informações técnicas e metodologias, atualmente o PMI conta com aproximadamente 235 *Chapters* no mundo [GUEDES 2006]. *Colleges* são grupos de membros do PMI que estão em um mesmo local geográfico e têm afinidade com uma ou mais áreas de

conhecimento e trabalham evoluindo esse conhecimento e disponibilizando para os profissionais da área [PMI 2006].

Após décadas de evolução do gerenciamento de projetos, hoje se faz uso de técnicas de diversas áreas sociais como: marketing, psicologia e relações humanas. Com isso vários modelos de negócio surgiram e se desenvolveram seguindo uma mesma base: os gerentes de projeto comandando uma equipe, mantendo a comunicação e o trabalho integrado.

As empresas utilizam muito o gerenciamento de projetos e com isso os gastos com esse processo vêem aumentando gradativamente. Segundo o PMI aproximadamente 25% do PIB mundial é gasto com projetos e 16,5 milhões de profissionais trabalham com gerenciamento de projetos no mundo [GUEDES 2006]. Todo esse investimento é feito para melhorar o processo de desenvolvimento de projetos nas organizações, inclusive na capacitação dos profissionais que gerenciam esses projetos, isso pode ser constatado no gráfico da Figura 2.2-1, que mostra o crescimento da quantidade de profissionais certificados (PMP) pelo PMI em 2005.

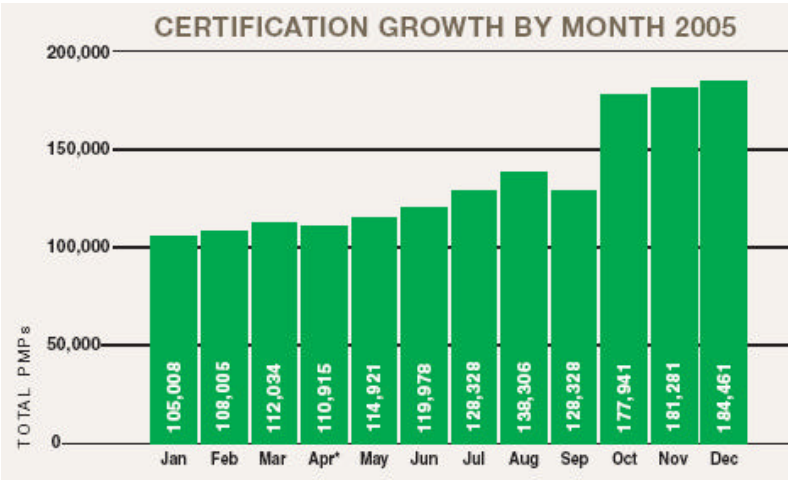

Figura 2.2-1 Crescimento das Certificações PMP feitas em 2005 [PMIREL 2005].

2.3 Gerenciamento de Projetos de Software

O gerenciamento de projetos segundo o PMBOK é composto por técnicas e ferramentas mais gerais, que podem ser utilizadas em qualquer projeto. Contudo, cada

área de domínio tem suas peculiaridades [GUEDES 2006]. A área de informática, em especial, por possuir características únicas como: muitas solicitações de mudança, desenvolvimento iterativo.

E está área não se limita à criação somente de programas de computador, mas sim a utilização e criação de novas tecnologias. Tecnologia está relacionada à inovação que indica mudanças com promoção. E inovação não está relacionada somente ao produto em si, mas a técnicas, processos e metodologias. Com isso as empresas que trabalham nesta área estão acostumadas a freqüentes mudanças e sempre as promovem, seja para criar algo para suprir uma necessidade ou melhorar uma solução já existente no mercado.

O gerenciamento de projetos de software herdou o estilo de planejamento da indústria manufatureira e da construção civil, ou seja, estruturas *top-down* e em PBS (Product Breakdown Structure), onde o produto é quebrado em componentes e artefatos e em um processo em cascata com fases seqüenciais. Porém para desenvolvimento de software essa forma de construir projetos é falha, pois geralmente (quase que na totalidade) no início do projeto não se sabe muito a respeito da especificação exata do produto a ser implementado [MARTINS 2005]. Então, o projeto sofre várias mudanças para contemplar melhor as funcionalidades para criar uma solução que resolva o problema, indicando assim um desenvolvimento iterativo. Os requisitos de um projeto de software podem mudar por diversos motivos, entre eles: o usuário muda de idéia, o problema é modificado, mudanças técnicas e mudanças de mercado.

Para entender melhor as necessidades da engenharia de software foi criado o RUP (*Rational Unified Process*). O RUP é uma metodologia criada com o objetivo de garantir que os softwares sejam produzidos com qualidade, dentro dos requisitos estabelecidos pelo cliente, respeitando o orçamento e o cronograma previstos. Ele é composto por disciplinas que fornecem diretrizes para definição das tarefas e para atribuição das responsabilidades. O RUP também segue as melhores práticas de desenvolvimento de software: desenvolvimento iterativo, gerenciamento de requisitos, arquitetura baseada em componentes, modelagem visual do software, verificação constante da qualidade e controle de mudanças.

No gerenciamento de projetos o RUP usa tanto uma abordagem *top-down* como *bottom-up*. As duas fases inicias, *Abertura e Elaboração,* são planejadas usando a *top-*

down e as duas últimas fases, *Construção e Transição*, usando a *bottom-up*, quando os artefatos a serem produzidos são conhecidos e bem definidos.

A proposta do RUP é conduzir o projeto em iterações, onde cada interação é abordada de forma a se fazer o trabalho de alguns requisitos e riscos, depois uma análise, construção e teste. E na próxima iteração novos requisitos e riscos são trabalhados, bem como a análise, implementação e testes e a assim por diante durante as outras iterações.

Usando a abordagem iterativa do RUP, para o projeto de desenvolvimento de software, recomenda-se que o projeto seja dividido em dois níveis de planejamento, ambos baseados no PMI: o *planejamento das fases*, onde há somente um plano para todo o projeto e que registra as fases dele (abertura, elaboração, construção e transição) e o *planejamento das iterações*, onde o plano de projeto detalha os trabalhos que serão executados em cada iteração.

Outro corpo de conhecimento que defini áreas e processos de gerenciamento de projetos para o domínio de software é o SWEBOK® [Abran 2004]. Ele decompõe a engenharia de software em várias áreas, dentre elas a de gerenciamento de projetos, e para esta área ele insere as cinco (cinco) primeiras áreas de conhecimento do PMBOK™ (integração, escopo, tempo, custos e qualidade) mais a área de métricas, muito importante no desenvolvimento de software. Além disso, as áreas de software devem estar relacionadas a essas de gerenciamento. O SWEBOK® define como parte integrante do gerenciamento de projetos as seguintes subáreas:

- Iniciação e definição de escopo;
- Planejamento;
- Execução;
- Revisão e avaliação;
- Fechamento;
- Métricas;

Ele também indica que as atividades de gerenciamento de projetos estejam intimamente ligadas ao modelo de processo de software adotado para o projeto. Um projeto que tenha como modelo de processo o unificado tem atividades diferentes do que adota métodos ágeis de desenvolvimento [GUEDES 2006].

3 Simuladores e Jogos de Negócio

O início do uso de simuladores ou jogos de negócio que se tem conhecimento, data da década de cinqüenta, porém a sua origem vem da antiguidade. Estes simuladores foram inspirados nos chamados jogos de guerra, onde o mais conhecido é o xadrez, que possui aproximadamente 3.000 anos. Os jogos de guerra passaram a ser utilizados, com um maior nível de detalhamento, na formação de oficiais das forças armadas.

Com o grande desenvolvimento da computação, fazendo com que fosse possível processar uma grande quantidade de dados e em um menor tempo, foi possível criar simuladores militares bem mais sofisticados e no campo da administração empresarial houve o surgimento dos chamados jogos empresariais, que são jogos utilizados para simularem os processos organizacionais. Nos últimos anos surgiram vários jogos de negócio bem mais avançados, pois utilizam tecnologias mais atuais e que agregam grande valor à simulação, por exemplo, com a vinda da internet foi possível fazer jogos à distância e até com multi-jogadores.

Neste capítulo serão mostrados os fundamentos teóricos sobre simuladores, o uso da dinâmica de sistemas para se fazer simulação e as aplicações dos simuladores, bem como os jogos de negócio na área de engenharia de software.

3.1 Fundamentos teóricos

A princípio os simuladores empresariais são classificados como modelos, que seriam uma forma de representar o mundo real, e com algo a mais, a possibilidade de jogar (simular) essa realidade. Um sistema é "um conjunto de elementos interconectados em que transformações ocorridas em uma das partes influenciarão todas as outras" [1], vários sistemas atuando em conjunto têm a função de alcançar um objetivo. Eles são compostos de subsistemas, elementos e inter-relações entre os sistemas.

[1] Definição segundo o Wikipedia. Disponível em http://pt.wikipedia.org/wiki/Sistema. [Acessado em 05 de março de 2007].

Com isso, podemos entender que uma empresa é um sistema, dado que atua para alcançar um ou vários objetivos, em geral para aumentar o lucro, que pode ser a curto ou longo prazo. Seus vários recursos, como os funcionários, por exemplo, são os elementos do sistema e estes mantêm inter-relações com os subsistemas. Já um modelo representa de forma mais simples uma parte do sistema. Esta simplificação se dá pela separação de um determinado conjunto de objetos que se quer analisar.

Os modelos são feitos baseados em um sistema real, com isso tem-se a possibilidade de obter informações sobre esse sistema, sem que para isso, tenha que fazer uma experiência na prática. Pois, em várias ocasiões não é interessante fazer essas atividades diretamente no sistema real, seja pelo fato de ser custoso, demorado ou perigoso. Um exemplo são os simuladores de vôo, com os quais os pilotos podem realizar, em forma de teste, as manobras necessárias para poder voar, sem que eles sejam colocados em uma situação de perigo, que pode ser irreversível, outro exemplo pode ser citado na medicina, onde um médico estudante pode realizar uma cirurgia em uma modelo que simule um ser humano e com isso não colocar em risco mais uma vida. Outra utilização dos modelos é da observação de situações que nunca puderam ser observadas.

Os modelos empresariais são geralmente simbólicos, onde os quais podem ser representados utilizando linguagens verbais ou matemáticas. Os modelos matemáticos são mais exatos, pois têm como características utilizar equações matemáticas para estabelecer as relações entre os elementos, em oposto a isso, os modelos verbais são menos precisos e específicos, pois são baseados em linguagens comuns.

Na simulação são desenvolvidos modelos para se estudar o comportamento do sistema real. Neste caso o sistema é descrito de tal maneira, que suas características e elementos sejam representados utilizando variáveis e as relações entre seus componentes são representadas por conectores lógicos. Em geral a simulação é utilizada para analisar as relações entre os elementos e no caso dos simuladores empresariais para explicar e justificar as relações. Os simuladores empresariais se baseiam em modelos matemáticos onde estão representadas áreas administrativas dentro de um contexto econômico. A partir destes modelos é dado valor as ações tomadas segundo as decisões do jogador. Então, simuladores empresariais utilizam modelos de simulação para representarem fatores de uma determinada realidade econômica ou processos organizacionais.

Existem vários modelos de simulação que têm um sistema de controle pré-definido, onde é possível um controle externo, porém não é necessário. Já nos simuladores empresariais a intervenção dos jogadores é indispensável para se continuar a simulação. Porém, o ambiente de jogo, no qual os jogadores atuam é formal em parte, com isso os jogadores precisam seguir algumas regras definidas pelo sistema de controle e pela estrutura do jogo.

Os modelos são classificados como mentais ou explícitos. Os modelos mentais representam o que o indivíduo percebe sobre as interações entre os componentes de um sistema e sobre os resultados dessas interações. Eles são bem flexíveis, podendo facilmente ocorrer adaptações para novas situações, por simplificação ou concatenação. Novas informações podem ir contra ao modelo atual, estimulando um rearranjo da estrutura do modelo ou podem reforçar as relações de conceitos, acrescentando novas informações, e assim, realizando o aumento pela concatenação. Contudo, a pouca capacidade de utilizar poucas variáveis distintas restringe os modelos mentais, tornando-os simples. Com isso, importantes características de sistema são esquecidas, fazendo assim uma análise equivocada sobre o sistema ou o ambiente observado e induzindo decisões erradas. Ainda tem a questão da dificuldade de compartilhar e validar os modelos mentais pelo fato de estarem apoiadas no conhecimento subentendido do indivíduo.

Já os modelos explícitos são formas de se representar os sistemas reais em uma linguagem que pode ser visualizadas e entendidas por outros indivíduos. Em geral os modelos explícitos têm a finalidade de promover alguma influência sobre os modelos mentais, pois não se toma decisão a partir deles, mas sim a partir dos modelos mentais. Estes modelos possuem um conjunto de símbolos e semânticas, e podem ser utilizados para representar, confirmar e propagar o conhecimento. Representar em um modelo com linguagem comum permite a explicitação das premissas e restrições, passando por cima das complexidades dos modelos mentais, o que facilita o seu entendimento. Validar o modelo importa, pois, enquanto que os resultados reais são obtidos, eles podem estar em divergência com os resultados esperados, com a validação isso tende a diminuir. E é possível difundir o conhecimento à medida que o modelo é descrito mediante uma linguagem que pode ser entendida por qualquer indivíduo.

Os modelos explícitos podem ser classificados em informais e formais. Os modelos formais, diferente dos informais, têm uma semântica não ambígua, com isso podem ser transformados em definições matemáticas e então serem simulados.

Os modelos informais são bastante importantes por dinfudirem um modelo ou processo em um determinado grupo, mas como sua semântica não é muito esquematizada, eles não são muito utilizados na simulação por computador. Porém eles podem ser executados por uma pessoa, que aprenda a sua semântica, em uma organização para desempenhar um papei ou executar um procedimento. Os engenheiros de software conhecem bem esse tipo de modelo, pois existem técnicas de modelagem de processos de software que possuem uma linguagem gráfica com conceitos desta atividade.

Existe uma linguagem que em muito ajudou para concatenar os conceitos e formas gráficas comuns que existem na modelagem de softwares, usando vários diagramas, mostrando formas diferentes de como o sistema pode ser visualizado. A UML (*Unified Modeling Language)* é uma linguagem de modelagem não proprietária de terceira geração [WIKI 2006]. Com o uso das extensões que a linguagem possui, foi possível também a sua utilização para a modelagem de processos, que passou a ser utilizada como uma disciplina, o *Unified Process*. Com isso a UML passou a representar não somente os processos de software, mas também os processos organizacionais de desenvolvimento de software.

Os modelos formais são definidos como os que podem ser executados computacionalmente. Com relação ao estado dos modelos formais, eles podem ser classificados como sendo estáticos ou dinâmicos. Os modelos formais estáticos são aqueles em que suas variáveis não sofrem alterações no decorrer do tempo, já os modelos formais dinâmicos sofrem alterações nas suas variáveis com passar do tempo. Ainda pode haver a classificação dos modelos formais dinâmicos como sendo discretos ou contínuos, de acordo com a forma de mudança de estado. Nos modelos dinâmicos discretos, não se tem pré-definidos os intervalos de tempo em que cada evento ocorre; já nos modelos contínuos, os eventos de mudança de variáveis ocorrem de um intervalo de tempo constante. Os modelos formais podem ser simulados, essa simulação pode ser utilizada para a verificação do modelo, principalmente quando o efeito do sistema e as interações

de seus componentes estão separados espacialmente e temporalmente. Existem duas técnicas de simulação distintas: a Simulação Discreta e a Simulação Contínua.

Na simulação discreta o simulador possui uma fila de eventos que são colocados um após o outro ordenado temporalmente. A cada rodada de simulação, o simulador executa o primeiro evento da fila e ajusta o seu relógio para o próximo instante de execução do evento seguinte atualizando assim as variáveis do modelo. Como a execução de cada evento, novos eventos podem surgir, os quais irão para o fim da fila. O intervalo de tempo entre os dois eventos não tem como ser previsto, então a simulação ocorre sem um período de tempo constante. A finalização da simulação se dá em um tempo pré-determinado ou quando não se tem mais eventos. O sistema baseado em regras é um exemplo de simulação discreta, onde os estados do sistema são definidos como fatos, e as ações que serão executadas são ditas como predicados.

Os modelos contínuos são simulados com intervalos de tempo constantes e já determinados, com suas variáveis alteradas em cada iteração de simulação. O final da simulação se dá quando o número de iterações previstas chega ao término. A simulação contínua se dá pela execução de equações que têm em sua estrutura as informações do modelo a ser simulado. Com isso, ela é utilizada para simular modelos estocásticos e então validá-los. O modelo possui variáveis escolhidas randomicamente dentro de um universo de domínio.

3.2 Dinâmica de sistemas

A Dinâmica de Sistemas (DS) foi criada em 1961 por Forrester [Forrester 1961]. Ele criou uma teoria que simula sistemas complexos, não-lineares e que contenham vários *feedback loops*. Primeiramente usou a DS para lidar com problemas industriais, como flutuação de estoques, instabilidade da força de trabalho e queda na participação de mercado. Desde então o seu uso tem se expandido para vários sistemas sociais.

A DS não se interessa em valores precisos, em um determinado momento do tempo ou em variáveis de um sistema. Seu foco principal está nas tendências dinâmicas do sistema. O seu objetivo é saber se o sistema é estável ou não, se tende a oscilar, crescer,

declinar ou se tende ao equilíbrio. O que se entende é que o comportamento dinâmico em sistemas complexos é gerado pela sua estrutura causal.

O principal conceito da metodologia é o *feedback*. O que se supõe primeiramente é que as informações sobre o sistema derivam as decisões. Por conseqüência, essas decisões resultam em ações que têm como objetivo mudar o sistema. Quando chega alguma informação nova sobre as condições do sistema, indicando se o mesmo mudou ou não, então a ação foi ou não eficaz. Essa nova informação gera outras decisões/ações que podem produzir mais mudanças no sistema. Isso é uma seqüência circular de causas e efeitos. Os modelos da DS são formados por várias seqüências de causa e efeito ou *feedback loops* inter-relacionadas.

A DS exige que cada elemento e cada relação do modelo tenham uma contrapartida na realidade. Outras metodologias têm uma abordagem diferente, usando variáveis nos seus modelos que não têm relação no mundo real. O uso dessas variáveis é comum, pois elas podem tornar mais fácil a solução analítica do modelo, ou para que as variáveis geradas pelo modelo se ajustem melhor aos dados históricos.

Por ter o objetivo de criar modelos realísticos, os modelos da DS possuem muitas relações não-lineares. Pois é acreditado que não-linearidades são importantes para explicar o comportamento de sistemas complexos. As relações não-lineares fazem com que os *feedback loops* variem mais. Com isso, quando se tem várias seqüências de causa e efeito não-lineares inter-relacionadas, em algumas condições, prevalecerá uma parte do sistema, e em outras condições outra parte do sistema que vai dominar. Nessa passagem de dominância de uma parte do sistema para a outra é que o comportamento do sistema mudará. Então, em um sistema composto por vários *feedback loops* não-lineares pode ser gerado uma quantidade muito grande de comportamentos complexos.

Quando o sistema é linear nunca se deve mudar a importância relativa dos vários *feedback loops*. Isso que dizer que esses sistemas podem ser analisados pela redução aos seus componentes individuais (solução analítica) e que, com isso, não se pode gerar comportamento complexos. Como o comportamento da realidade observado é complexo, então se conclui que deve existir uma não-linearidade na realidade.

O DS usa conceitos de outras áreas do conhecimento para ter modelos realísticos, ou seja, que sejam capazes de gerar o comportamento dinâmico do mundo real. Ele

procura os conceitos necessários na área em questão dependendo do que está sendo modelado. Esses modelos usam conceitos de diversas ciências: as exatas (física, matemática, etc.) e sociais (psicologia, sociologia, etc.). Qualquer modelo possui em sua estrutura duas partes: implicações sobre o ambiente físico e outras sobre o processo de tomada de decisão dos agentes integrantes das estruturas físicas.

3.2.1 Modelos da Dinâmica de Sistemas

Além dos diagramas de Causa e Efeito, a Dinâmica de Sistemas disponibiliza para as análises os Diagramas de Repositório e Fluxo. Esses dois diagramas serão descritos a seguir.

3.2.1.1 Diagramas de Causa e Efeito

Este diagrama é uma técnica simples e de fácil adaptação que permite aos seus analisadores, que tomam as decisões, identificar quais são as variáveis de interesse e explorar as relações entre essas variáveis.

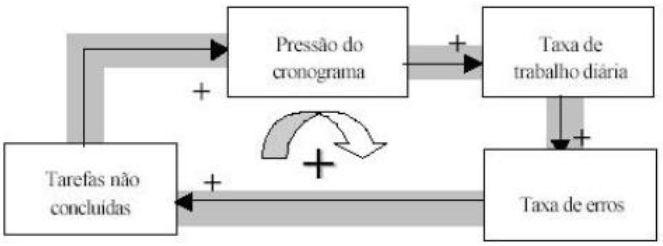

Figura 3.2-1 - Diagrama de causa e efeito [GUEDES 2006].

A Figura 3.2-1 mostra um exemplo de diagrama de causa e efeito num ambiente de gerenciamento de projeto. Nele pode-se observar uma relação positiva entre a Taxa de trabalho diária e a Taxa de Erros, onde indica proporção direta entras as duas, pois quando a taxa de trabalho diária aumenta tende a aumentar o número de erros existentes no projeto, seguindo o que ocorre naturalmente no modelo real, pois o cansaço de um aumento do trabalho diário ocasiona isso. Neste modelo, o mesmo acontece para as

demais variáveis, pela forma como a modelagem foi feita. No centro do diagrama, existe um sinal positivo com uma seta, isso indica a polaridade do ciclo [GUEDES 2006].

Este tipo de ciclo é chamado de retro-alimentação, pelo fato de possuir uma polaridade que indica seu comportamento. Neste caso, como é positivo, a tendência é o aumento ou diminuição dos valores das variáveis à medida que se percorre o diagrama no seu sentido. Em um caso onde seja negativo há uma tendência de equilíbrio no modelo.

O diagrama é montado utilizando setas que ligam as causas aos efeitos, esse é o motivo de sua nomenclatura. A ligação entre duas variáveis pode ser negativa ou positiva, indicando a proporcionalidade do efeito com relação ao causa, se é inverso ou direto, respectivamente. Segunda a definição, dada uma ligação X → Y ela será [Sterman 2000]:

- Positiva, caso a variação positiva (ou negativa) no valor X implique em uma variação positiva (ou negativa) no valor Y, seguindo a proporção direta. Assumindo que todas as outras variáveis estejam com o mesmo valor.
- Negativa, no caso quando há a variação negativa (ou positiva) de X que implica em uma variação positiva (ou negativa) na variável Y, verificando a inversão de proporcionalidade. Também se assume que as outras variáveis permaneceram com o mesmo valor.

A possibilidade de se ter um caminho, entre as relações de causa e efeito, que parta de uma variável e retorne a essa mesmo variável é definido como um ciclo de retro-alimentação. E juntamente com essa definição, se tiver o fato de que todas as ligações em um modelo sejam caracterizadas como positivas ou negativas, então é possível determinar a polaridade do ciclo, e com isso perceber qual o comportamento do mesmo. A polaridade do ciclo é definida como:

1. Positiva, se o número de ligações negativas em um ciclo for par.
2. Negativa, se o número de ligações negativas em um ciclo for ímpar.

Os diagramas de causa e efeito são mais simples e devem ser refinados para diagramas de repositório e fluxo. O diagrama de causa e efeito apresenta diversos componentes do sistema com o efeito de acumulação ou redução de volume em um componente, e o resultado provocado sobre os demais. Devido o fato de ele ser um diagrama mais simples, o diagrama de causa e efeito é utilizado para explicar conhecimentos retirados do modelo, porém não é adequado para análise de regras e simulações [GUEDES 2006].

3.2.1.2 Diagramas de Repositório e Fluxo

Os diagramas de repositório e fluxo apresentam um maior nível de detalhes e simbologia diferente. Os elementos gráficos da Dinâmica de Sistemas são mostrados na Figura 3.2-2.

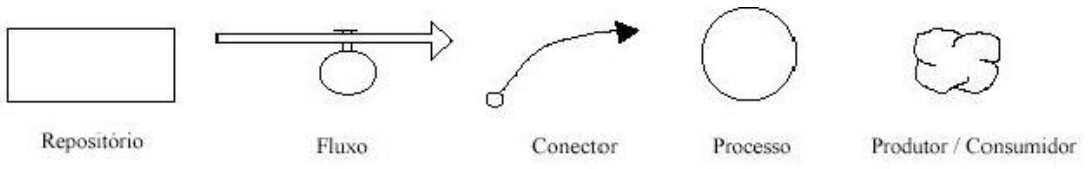

Figura 3.2-2 - Simbologia do diagrama repositório e fluxo [GUEDES 2006].

Cada um desses elementos tem um papel importante e representam um determinado fenômeno. A definição deles está relacionada abaixo:

- Repositório: é a representação do elemento que pode ser estocado e consumido com o passar do tempo. Por exemplo, um conjunto de raposas e outro de coelhos podem representar repositórios.
- Fluxo: é uma taxa de variação do repositório com relação a um determinado instante de tempo e, geralmente, está ligado a um ou dois repositórios. O número de coelhos que alimentam as raposas, ou o nascimento de coelhos.
- Processo: é utilizado para calcular informações a partir de um conjunto de parâmetros como: a taxa de variação indicada por um fluxo, nível de um repositório ou produto de um outro processo.

- Conector: são as vias de transmissão de informação no modelo, assim como os fluxos transferem elementos entre os repositórios os conectores permitem a troca de informações entre os processos.
- Produtor/Consumidor: representam os produtores e consumidores infinitos, ou seja, as fontes infinitas de informação a um fluxo ou um destino de consumo das informações geradas por um fluxo.

A modelagem na dinâmica de sistemas é baseada no principio da acumulação. O nível de acumulo de um estoque é controlado pela entrada e pela saída. O nível do estoque aumenta quando a taxa do fluxo de entrada for maior que a taxa de saída. Ao contrário, o nível diminuirá caso a taxa de entrada for menor que a taxa de saída [GUEDES 2006].

Os diagramas de repositório e fluxo são interpretados matematicamente, onde seus elementos são transformados em equações que são passíveis de execução em um simulador.

3.3 Áreas de aplicação e uso de Jogos de simulação

Os simuladores empresariais podem ser utilizados em diversas áreas de aplicação, tais como: formação acadêmica e treinamento empresarial, desenvolvimento de recursos humanos, pesquisa psicológica, desenvolvimento organizacional, suporte ao processo decisório e instrumento de pesquisa econômica.

3.3.1 Formação acadêmica e treinamento empresarial

Os simuladores ou jogos de empresas podem ser utilizados na formação acadêmica ou em treinamentos em grupos empresariais, e podem ser usadas nas seguintes instituições:

- No ensino superior (aplicação prática dos conhecimentos técnicos);
- No ensino técnico (demonstrar a importância do conhecimento teórico);

- Em empresas (treinamento gerencial e desenvolvimento organizacional).

Um exemplo do uso de jogos de negócio nas organizações é jogo Virtual-U[2], que é um jogo empresarial para o treinamento sobre gerenciamento de universidades. Os jogos de empresa podem possuir várias aplicações diferentes pedagógicas em função do público alvo que será atingido. Porém, a sua maior utilização se encontra na aprendizagem na graduação e pós-graduação. Essa aprendizagem utilizando jogos de empresa pode ser subdividida em quatro áreas:

- **Difusão do conhecimento técnico (aprendizagem cognitiva):** faz parte da aprendizagem cognitiva a aquisição de conhecimento sobre gerenciamento, por exemplo, a leitura e interpretação de um plano de projeto. Para os jogadores é possível utilizar, de forma ativa, conhecimentos teóricos adquiridos em sala de aula.
- **Capacitação para o processo decisório em situações complexas:** os simuladores empresariais permitem que os estudantes, gerentes e administradores desenvolvam aptidões para tomada de decisão, já que eles precisam estar sempre capacitados para decidir e negociar em ocasiões complexas. Uma das características de um sistema complexo é que as variáveis que compõem o sistema estejam inter-relacionadas. Uma empresa é um bom exemplo de um sistema complexo, visto que algumas decisões em um determinado departamento afetam outras áreas da organização. Sistemas complexos também são caracterizados por apresentarem uma quantidade muito grande de informações e uma pequena transparência de seus processos. Também é possível se observar que situações complexas possuem uma dinâmica própria, ou seja, mesmo que nenhuma decisão seja tomada o sistema continuará em andamento. Nos simuladores empresariais é necessária a tomada de decisões em situações complexas, o que permite alertar os jogadores quanto à importância do contexto do simulador. Estes simuladores também permitem ao participante avaliar os

[2] Consiste em uma simulação por computador sobre o gerenciamento de universidades. Disponível para download em http://www.virtual-u.org/ [acessado em 11/03/07]

efeitos de suas decisões tanto no curto quanto no longo prazo. Portanto os simuladores empresariais devem desenvolver aptidões que permitam tratar situações complexas de decisão. Mesmos que eles sejam apenas uma simplificação da realidade, pode-se entender que a experiência adquirida tenha um efeito positivo sobre o processo decisório real.

- **Desenvolvimento de aptidões para o trabalho em equipe (aprendizagem efetiva/emocional):** Este campo da aprendizagem trata de melhorar o comportamento cooperativo do jogador. O objetivo é permitir que problemas tratados individualmente ou por algumas poucas pessoas possam ser discutidos em grupo, de forma a se encontrar uma solução satisfatória para todos.
- **Treinamento e aplicação de técnicas de trabalho (aprendizagem instrumental):** A aprendizagem instrumental e metodológica está vinculada com as outras áreas de aprendizagem comentadas anteriormente. E este tipo de aprendizagem indica a utilização das técnicas utilizadas no trabalho para firmar o conhecimento sobre elas, ou seja, os simuladores ajudam na utilização destas técnicas para o amadurecimento do entendimento sobre elas.

3.3.2 Uso de jogos de simulação

Nos jogos de simulação, o ambiente e as atividades dos participantes têm características de jogo: existem papeis a se desempenhar, metas, atividades, restrições e recompensas como resultado de suas ações e as dos outros elementos do simulador [GUEDES 2006].

Algumas das características que os jogos possuem são: interação, competição, *feedback* visual, flexibilidade, efeitos dramáticos, usabilidade, graus de fidelidade e realidade, níveis de dificuldade, etc. Essas características fazem com que a utilização de jogos atue nos problemas dos ambientes de simulação como a motivação e o engajamento.

Apesar de os estudos indicarem que o uso de jogos e ferramentas de simulação seja muito bom no auxílio à capacitação de pessoas, pouco se tem de exemplos no mercado

que evidencia isso. Pois, nesse mercado, a grande maioria dos jogos disponíveis, comerciais ou não, são voltados para a diversão.

Existem algumas iniciativas de criação de jogos de treinamento nas áreas de engenharia de software e de gerenciamento de projetos [GUEDES 2006], a seguir serão apresentados alguns exemplos:

- SESAM

O *Software Engineering for Software by Animated Models* foi inicialmente descrito em 1989 e posteriormente evoluído. O SESAM possui modelos estruturados em duas partes: Uma descrição estática que apresenta a definição e os tipos dos objetos envolvidos no processo de software e seus possíveis relacionamentos. E uma perspectiva dinâmica onde o comportamento é representado através de regras que especificam ações e efeitos, os quais provocam mudanças no estado do projeto simulado. Para o início da simulação do SESAM é necessário alimentar o simulador com um conjunto inicial de estados, que são responsáveis por disparar as primeiras regras de simulação. Estes estados iniciais são uma instância dos projetos a serem simulados. A interação junto ao usuário se dá com o uso de uma interface textual mostrando o andamento da simulação. Com o fim da simulação o aluno vai poder refletir sobre os resultados de suas decisões a partir da análise de comportamento das variáveis que o SESAM realiza [GUEDES 2006].

- SIMSE

O SimSE é um jogo criado para prover o ensino de processos de engenharia de software. Ele é um jogo monousuário no qual o jogador assume o papel de um gerente de projetos e pode exercer, entre várias, as seguintes tarefas: demitir e contratar desenvolvedores, delegar tarefas, monitorar o andamento do projeto e adquirir ferramental necessário para o trabalho da equipe. Este jogo possui uma interface gráfica para a interação com o jogador, onde este pode visualizar a sua equipe em um escritório de desenvolvimento com vários objetos comuns desse tipo de escritório (computadores, mesas, cadeiras, etc.). Além da interface do jogo (simulação), o SimSE possui uma outra

interface para a geração do modelo a ser simulado, o *Model Builder.* É neste modelo que se podem ser encontradas as definições sobre os tipos, atividades, regras, representação gráfica de cada elemento do jogo (desde os personagens a mesas, cadeiras, etc.) [GUEDES 2006].

Outro componente importante da arquitetura do SimSE é o *Generator*, que é responsável por interpretar o modelo criado anteriormente e gerar um código do gerenciamento de estados e de execução de regras que estão na máquina de simulação (*Simulation Environment*). A simulação se segue com a geração de *ticks* pelo *Clock* que irão ativar a execução de regras pelo *Rule Execution*, este por sua vez verifica as ações que estão em execução no momento no gerenciador de estado (*State Mgmt*). O *Clock* também é responsável por indicar a interface gráfica para que ele se atualize e reflita o novo estado. O esquema da arquitetura do SimSE é mostrado na Figura 3.3-1.

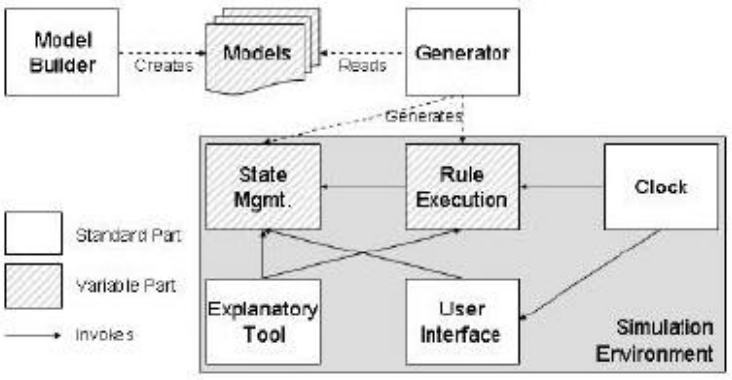

Figura 3.3-1 - Arquitetura do SimSE [GUEDES 2006].

Existe ainda o componente *Explanatory Tool,* que permite ao jogador verificar a qualquer momento um registro de todas as ações tomadas pelo por ele. O modelo criado pelo *Model Builder* é baseado em alguns meta-objetos e em ações e regras. Os meta-objetos são definidos em campos pré-definidos, já as ações e regras terão a implementação dos comportamentos do modelo. Para a simulação devem-se criar entidades que sejam subtipos dos meta-objetos na construção dos modelos específicos [GUEDES 2006].

- The Incredible Manager

Em alguns trabalhos de pesquisa realizados na COPPE-UFRJ surge o jogo The Incredible Manager. O modelo de simulação segue um meta-modelo criado para ser simulado usando dinâmica de sistemas, onde há a definição dos tipos (artefatos, papeis, atividades) e relações.

Ainda existe o modelo de projeto onde são criadas instâncias dos tipos definidos no modelo de domínio, o modelo de cenário, onde se pode criar situações específicas para a execução do projeto. Além desses elementos o jogo ainda possui o simulador de modelos, que é responsável por controlar os passos de simulação, e a máquina de jogo que é o por onde o jogador interage e recebe respostas visuais dos resultados da simulação. Abaixo segue uma figura que mostra a estrutura de atuação do jogo.

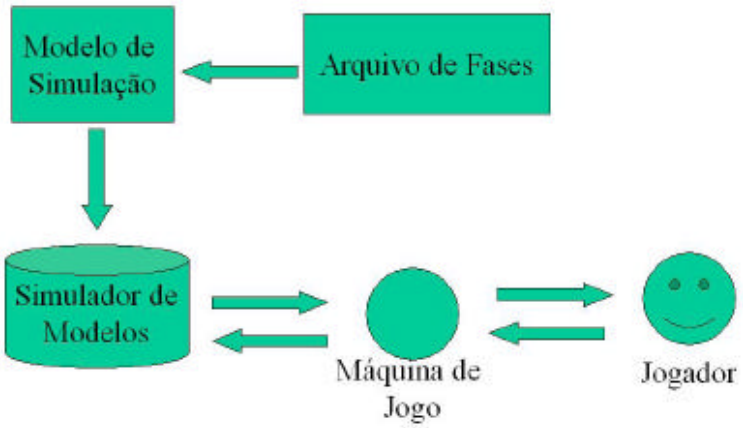

Figura 3.3-2 Estrutura do jogo The Incredible Manager [GUEDES 2006].

4 O Jogo Virtual Team

O Virtual Team é um jogo sério que é fruto do projeto SmartSim. O SmartSim é um projeto multidisciplinar, envolvendo as áreas de inteligência artificial, psicologia organizacional, jogos sérios baseados em simulação e gerenciamento de projetos, que tem por objetivo criar um *framework* de código aberto, usando a licença LGPL [LGPL 2006], para o desenvolvimento de jogos sérios baseados em máquinas de simulação e que utilizam atores sintéticos [ROUSY 2000] para simular as personagens envolvidas no processo [GUEDES 2006].

O Virtual Team foi idealizado como um protótipo de um jogo para ajudar no treinamento de gerentes de projeto. O protótipo teve ênfase em uma das áreas do gerenciamento de software, a de gestão de pessoas. Com ele é possível simular cenários de ambientes de desenvolvimento de software, onde os atores sintéticos são utilizados para desenvolver os papeis de participantes de equipe de desenvolvimento, tais como: engenheiros de software, arquitetos de software, programadores, testadores, analistas de sistema e outros. No jogo o papel de gerente de projetos é desempenhado pelo jogador.

A utilização de atores sintéticos para serem os membros da equipe tem por objetivo dar mais realismo aos cenários simulados, permitindo uma experiência mais rica para o jogador sobre os processos organizacionais, metodológicos, pessoais e culturais intrínsecos num ambiente de desenvolvimento de software.

Neste capítulo será mostrado um pouco mais sobre o jogo Virtual Team. Descrevendo em detalhes sua arquitetura.

4.1 Arquitetura do Virtual Team

Foi proposta, para o Virtual Team, uma arquitetura que permitiria a interação entre dois diferentes ambientes: um de simulação baseada nos atores sintéticos e um outro baseado na dinâmica de sistemas.

A simulação baseada nos atores sintéticos foi destinada para simular os perfis de cada um dos indivíduos que compõem a equipe de projetos, como: personalidade,

motivação, relacionamento com outros membros da equipe, entre outras coisas. A modelagem desses fatores contribui para aumentar o grau de realidade do jogo e ainda permitir a exploração de aspectos sobre o relacionamento humano. Este tipo de modelagem foi também utilizado pelo fato de haver uma maior dificuldade de criar um modelo sistêmico que comporte as teorias de personalidade [GUEDES 2006].

Já a simulação baseada na dinâmica de sistemas foi responsável por simular o processo de gerenciamento de projetos e suas variáveis diretas ou que têm alguma relação, como por exemplo: tempo, custo, erros, completude, EVA e outras.

Existe uma interação entre os dois tipos de simulação, pois as informações que advinham da simulação dos membros da equipe de projeto são utilizadas para simular como o projeto vai decorrer e os resultados da simulação do projeto devem indicar os comportamentos dos membros da equipe em sua simulação.

A arquitetura do Virtual Team possui diversas classes e pacotes diferentes relativos aos dois ambientes de simulação, de atores sintéticos e de projeto, bem como as classes sobre controle de eventos, interface gráfica, componentes gráficos e outros. Visto que o objetivo desse trabalho é criar um plug-in para integrar a parte de simulação de projetos a uma ferramenta de gerenciamento de projetos, será abordada aqui a parte de arquitetura relativa à simulação de projetos.

Para representar a arquitetura do Virtual Team será utilizado o diagrama de classes e de seqüência de UML [UML 2007]. A UML é a linguagem de modelagem utilizada hoje em dia em sistemas orientados a objetos. Foram utilizados diagramas de classe para representar a organização das principais classes que implementam a arquitetura do jogo. Também foram utilizados pacotes para organizar as classes, os pacotes na UML são estruturas que funcionam como pastas, onda as estruturas da UML, como as classes, podem ser agrupadas. Para poder se ter um melhor entendimento sobre a arquitetura de Virtual Team será apresentada a seguir, na Figura 4.1-1, uma visão geral sobre ela. As trocas de mensagens dentro do jogo serão apresentadas nos diagramas de seqüência, mostrando uma visão mais dinâmica das classes.

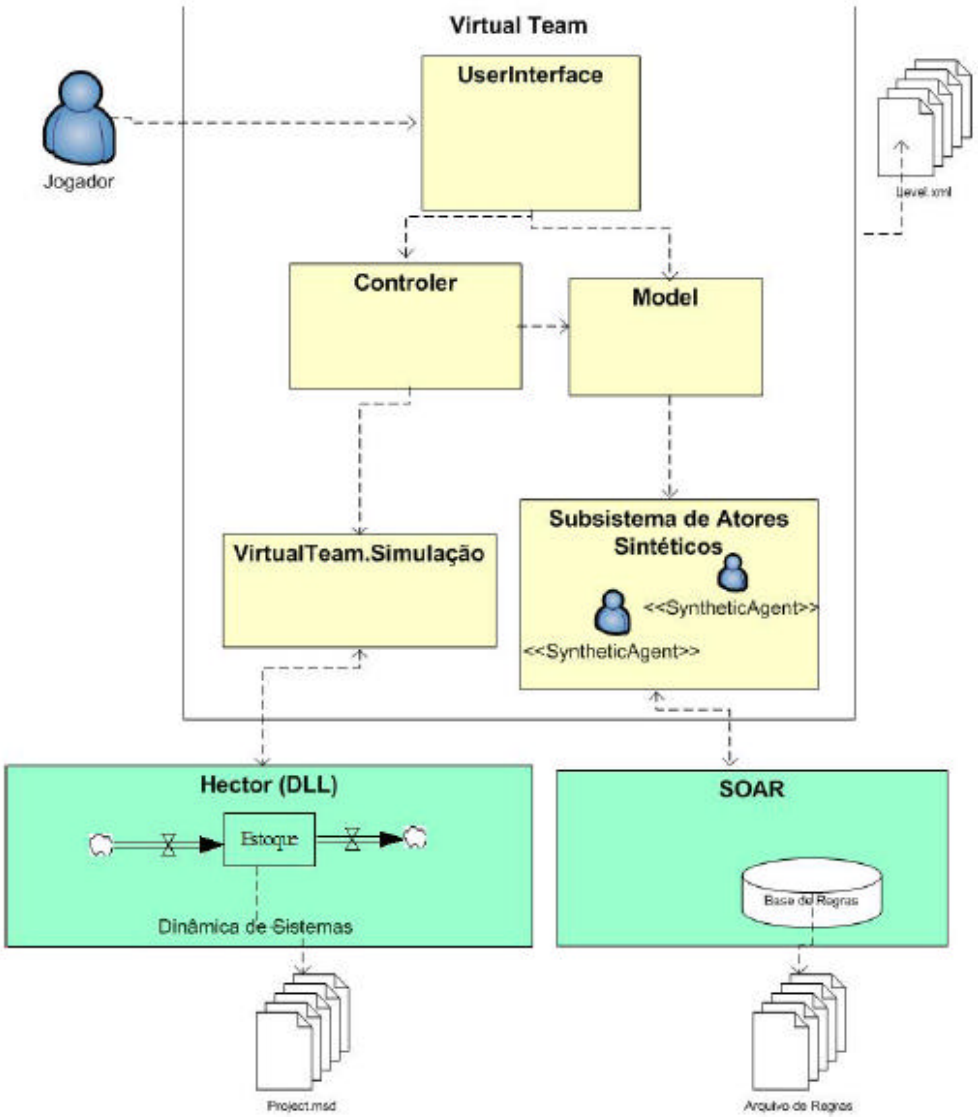

Figura 4.1-1 Visão geral dos pacotes da arquitetura do jogo [GUEDES 2006].

O jogo foi implementado utilizando a linguagem C++, pois foram levados em consideração os requisitos de alto desempenho e baixo consumo de memória. Outro fator importante para se escolher da linguagem C++ foi a utilização de um framework de construção de jogos chamado FORGE, ele compõe um conjunto de ferramentas para controlar as interações do jogador com os objetos do jogo, a interface gráfica.

A arquitetura básica, como mostrada, obedece a uma organização em camadas. A camada de interface com o usuário é representada pelo FORGE, usando algumas classes com o objetivo de controlar o comportamento dos objetos na interface, e os próprios

componentes gráficos que estendem os componentes do FORGE. A camada de controle (*Controller*) é responsável pelo *loop* de simulação do jogo, sempre interagindo com a camada do modelo do jogo (*Model*) para obter as informações sobre o estado atual dos dos atores e repassar para o subsistema de simulação. A outra interação que a camada de controle faz com a camada de modelo é a atualização das informações sobre o projeto e das atividades. O subsistema de simulação interage com a DLL que é quem realmente realiza a simulação usando as técnicas de dinâmica de sistemas. Enquanto que a camada de modelo interage com o subsistema dos atores sintéticos, responsável por simular as regras de comportamento e os estados observados por cada ator sintético.

O Virtual Team possui três pacotes mais importantes, que seguem o padrão MVC [MVC 2007] para organizar as classes. O pacote *virtualTeam.game* contém toda a máquina do jogo, as classes de interface e as classes que modelam os comandos do jogador. Já o pacote *virtualTeam.model* contém as classes que modelam o projeto e as que representam os atores sintéticos, incluindo a sua base de conhecimento e a máquina de inferência. Por fim o pacote *virtualTeam.worlddata* é responsável por iniciar todas as variáveis do simulador, sejam as de simulação, quanto às de internacionalização e outras.

As classes de mais alto nível do jogo podem ser observadas na Figura 4.1-2. Podem ser observadas algumas classes importantes. A classe `VTGameManager` é quem controlo o laço principal do jogo, trocando os estados do jogo segundo o a arquitetura do FORGE, cada estado é implementado como uma subclasse de `VTGameState`. O gerenciamento das entidades do modelo do jogo fica a cargo da classe `VTEntityManager`.

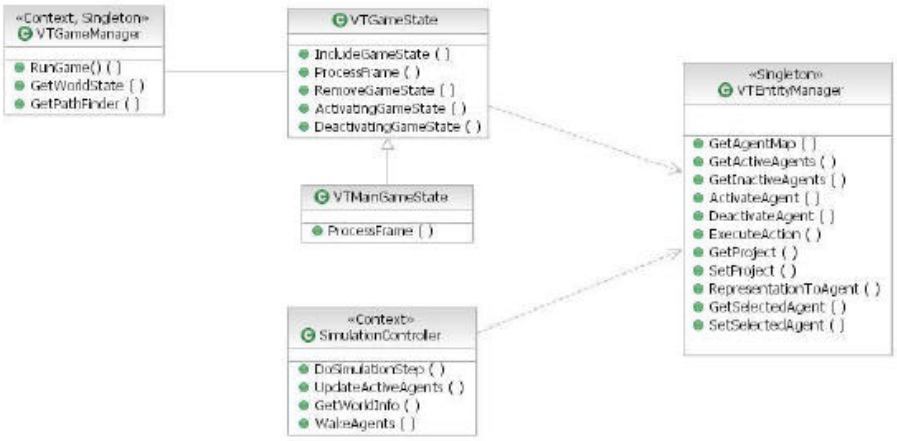

Figura 4.1-2 - Classes principais do Virtual Team [GUEDES 2006].

Segundo a arquitetura seguida pelo framework FORGE, a seqüência de telas do jogo é definida como os estados do jogo. Esses estados podem ser a tela de apresentação, o menu principal, inúmeros estados do jogo em si, finalização do jogo, etc. No Virtual Team é composto pelos seguintes estados: a tela de menu principal (implementada pela classe `VTMainMenuState`), a tela de seleção dos membros da equipe (classe `VTSelectTeamState`), a tela do jogo em si, onde o jogador interage (classe `VTMainGameState`) e a tela de final de jogo (classe `VTGameOverState`). Cada estado desses possui uma máquina de estado interna diferenciada.

É na classe `VTMainGameState` que ocorre o laço principal do jogo, responsável pela simulação do projeto e interação com o usuário para mudar os valores das variáveis e o estado da simulação. Para que o jogo rode a classe `VTMainTimer` aciona os *ticks* do relógio do jogo. Isto é importante para que o jogo seja acelerado, desacelerado ou mesmo pausado pelo jogador.

A Figura 4.1-3 apresenta o diagrama de seqüência com as trocas de mensagens entre as classes apresentadas. Com ela é possível entender a seqüência de ações do laço principal de simulação. A classe `VTMainGameState` é invocada, a cada frame, várias vezes por segundo. É definido um parâmetro para a razão entre o tempo do jogo e o tempo real. A cada passo de simulação do Virtual Team corresponde à uma hora na cronologia do jogo, porém mapeando para o tempo real corresponderia a um minuto em média. Com base na quantidade de vezes que ela é invocada pelo FORGE e na taxa de fps (frames por segundo), esta classe consegue determinar os momentos de invocar o método `TimeTick()` da classe `VTMainGameStateTimer`. Esta classe por sua vez invoca o método `DoSimulationStep()` da classe `SimulationController`.

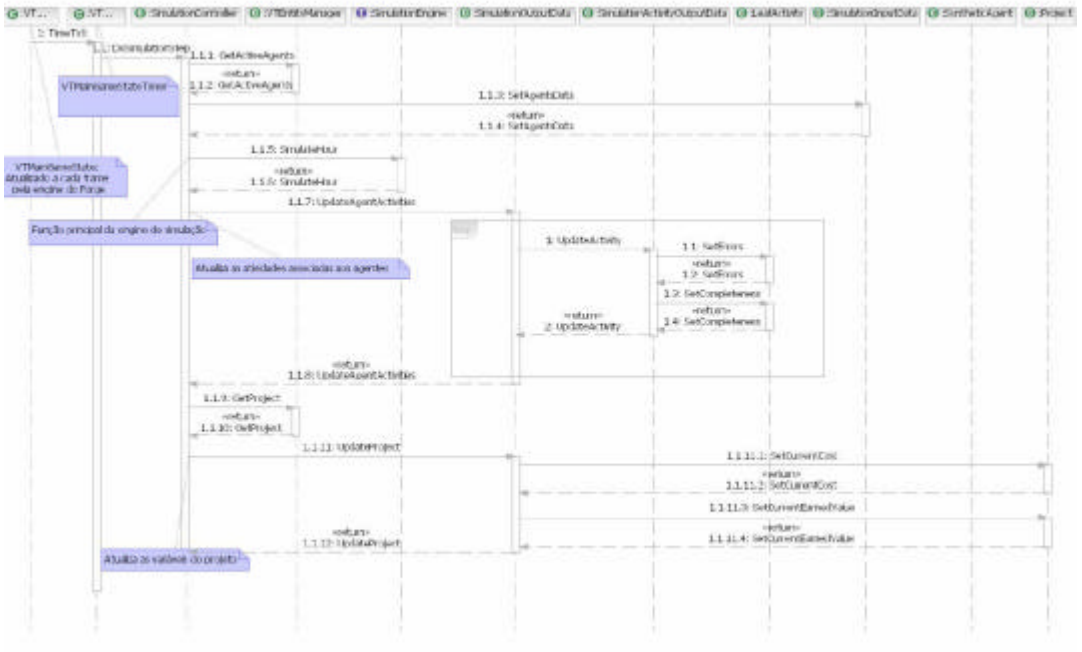

Figura 4.1-3 - Laço principal responsável pela simulação [GUEDES 2006].

5 O *Plug-in*

Neste capítulo será apresentado o desenvolvimento do *plug-in* para a ferramenta de gerenciamento de projetos escolhida. Primeiramente será feita uma breve motivação para objetivar o desenvolvimento do *plug-in* para integração do jogo a uma ferramenta de gerenciamento de projeto. Em seguida será descrito qual ferramenta foi escolhida e o motivo da escolha. Ainda será mostrado como se desenvolver *plug-ins* para essa ferramenta. E também o esquema do arquivo de entrada para a simulação no jogo Virtual Team. A arquitetura do *plug-in* com suas classes básicas e de controle. E por fim a forma de uso e as telas do *plug-in* em funcionamento.

5.1 Motivação

A idéia de se ter um simulador para gestão de projetos é muito importante por diversos motivos, como foi descrito anteriormente. Sintetizando, há várias áreas de atuação para o mesmo, desde o treinamento de pessoas a utilização dos simuladores para antecipar resultados e prover alterações no mundo real que superem os obstáculos encontrados na simulação.

Na área empresarial os jogos de negócios surgiram para um melhor entendimento, dos que estão ativamente inseridos nesse meio, sobre seus processos e ferramentas. E então aplicar melhor esse ferramental no mundo real, que é onde se tem muito prejuízo quando os problemas aparecem e principalmente os que não são percebidos imediatamente.

Gestão de projetos é uma atividade muito importante nas empresas, principalmente nas de desenvolvimento de software, onde prazo e escopo são variáveis muito mutáveis. As iniciativas de criação de simuladores para a área de engenharia de software e gerenciamento de projeto de software foram muito importantes para tentar prever os resultados dos serviços e produtos desenvolvidos, controlando melhor as adversidades, e também no treinamento do capital humano empregado nessa área, melhorando ainda mais os processos.

O Projeto SmartSim [SMARTSIM, 2006] teve com um dos objetivos criar um jogo de negócio para a área de gerenciamento de projetos de software, mais especificamente

na área de gestão de recursos humanos, uma área muito importante no projeto de software, pois trata do que traz o maior custo para o projeto e onde se tem os maiores conflitos e problemas. O Virtual Team segue o modelo de gerenciamento de projeto baseado no PMBOK™, com os conceitos de atividades, WBS (Work Breakdown Structure) e outras. Esse jogo também faz uma leitura de um arquivo, como esquema próprio, com os dados a serem simulados.

A possibilidade de poder captar dados do mundo real e gerar uma entrada para o simulador é muito interessante, pois pode ser usado para validar o projeto real, treinar gerentes de projeto usando dados reais e com isso entender melhor os efeitos das possíveis situações. Esse trabalho se propõe a criar um *plug-in* para uma ferramenta de gerenciamento de projeto que irá integrá-la ao jogo Virtual Team *Plug-in* é um termo que significa "de encaixe", em computação significa um pequeno programa que pode ser acoplado a outro aumentando as suas funcionalidades [PLUGIN 2007]. A integração se dá com a captação das informações reais de um projeto gerenciado usando a ferramenta de gerenciamento e que são necessárias para a simulação realizada utilizando o jogo. As informações são resgatadas de duas formas, dependendo do tipo e disponibilidade na ferramenta de gerenciamento: uma é a captura das tarefas e do nome do projeto diretamente do arquivo do gerenciador e a outra é a utilização de uma interface com o usuário para ser inserido outras informações necessárias e que não se encontram no arquivo.

A ferramenta escolhida para se produzir o *plug-in* foi o Microsoft® Project 2003 [MSPROJECT 2007]. Por ser uma ferramenta muito utilizada no mercado e que suporta bem os requisitos necessários para a geração do arquivo de simulação para o jogo. Os *plug-ins* para os softwares criados pela Microsoft® são chamados de *Add-in* ou suplementos.

5.2 Desenvolvimento de *Add-in* para o Microsoft® Project

Nas ferramentas do pacote Microsoft® Office há a possibilidade de se acoplar vários subprogramas (os *add-ins*) para que uma nova função mais específica seja

oferecida ao usuário, por exemplo, um *add-in* que gera a WBS no Microsoft® Visio a partir da estrutura de atividades de um projeto contida no Microsoft® Project.

O desenvolvimento de um *add-in* para as ferramentas Office se dá utilizando o Ambiente Integrado de Desenvolvimento (IDE - Integrated Development Environment) Visual Studio, uma ferramenta também da Microsoft®. As linguagens de programação que podem ser usadas para o desenvolvimento deste tipo de aplicativo são a Visual Basic, Visual C#, Visual J# e a Visual C++.

Um *add-in* é compilado como uma DLL (Dynamically Linked Library) [ADDIN 2007]. Uma DLL é um módulo de programa que não é executável diretamente, com isso pode-se guarda partes de código que serão carregadas em momentos oportunos da execução do sistema que hospeda a DLL [WIKI 2006]. Sendo assim, um *add-in* pode ser carregado de duas maneiras em umas das ferramentas do Office: assim que o programa for executado, e então funcionar equanto o programa estiver sendo executado, ou quando acionado mediante chamada de um comando. O que se diz na prática é que o *add-in* é conectado, e existe uma classe nele responsável por essa conexão, essa classe será descrita a seguir.

A criação de um *add-in* no Visual Studio é feita utilizando um *Add-in Wizard*. Um *Wizard* é um programa de computador interativo que faz uso de uma interface amigável para ajudar o usuário a resolver tarefas complexas, usando janelas de diálogo como um passo-a-passo[3]. O primeiro passo do *Wizard* de criação de um *add-in* é a escolha da linguagem de programação a ser usada para o desenvolvimento. Como há diversas possibilidades, essa escolha é importante para a configuração da IDE para o desenvolvimento do *add-in*. Outra etapa importante é a escolha dos aplicativos que irão suportar o *add-in* que será implementado, pois o pacto do Office possui diversos aplicativos, como: Word, Excel, Power Point, Access, o próprio Project e outros. O *add-in* pode ser desenvolvido para qualquer um desses aplicativos e até mesmo para funcionar em mais de um, inclusive o Visual Studio.

Para o desenvolvimento de um *add-in* específico para uma aplicação do pacote Office é necessário acrescentar ao projeto do *add-in* o SDK (Software Development Kit),

[3] Definição retirada do Wikipedia. Disponível em http://en.wikipedia.org/wiki/Wizard_%28software%29 [Acessado em 22 de março de 2007].

que é o conjunto de ferramentas utilizadas para interagir com a plataforma do aplicativo. Para o caso do MS Project, foi adicionado ao projeto o SDK específico do mesmo.

O projeto de um *add-in* é iniciado no Visual Studio com uma estrutura básica para o funcionamento do mesmo nos diversos aplicativos que irão suportá-los. Existe o pacote com as referências (`References`) que são utilizadas no projeto para o seu desenvolvimento, é aqui onde se acrescenta a API (*Application Programming Interface*) do Microsoft Project, a interface de programação para o Project. Uma outra classe existente em um projeto de *add-in* é a `AssemblyInfo`, que agrega um conjunto de atributos do assembly do projeto para o momento da compilação, como por exemplo: um título, descrição, informações de copyright, etc. Do conjunto de objetos de um projeto de *add-in* o mais importante é a classe `Connect`, que é responsável inicia-lo junto ao aplicativo que o suporta, ou seja, fazer a sua conexão e execução inicial. Essa classe implementa a interface IDTExensibility2 e com isso existem alguns métodos e eventos importantes para a suas inicialização e o fechamento do mesmo.

O evento `OnConnection` é disparado sempre que o suplemento é conectado [SuplementoCOM 2007]. O método que implementa as ações que serão executadas neste evento, quando ocorrer o disparo, recebe como parâmetros: a referência para o aplicativo hospedeiro, o modo como o suplemento será conectado, um objeto referenciando o próprio suplemento e uma matriz de valores que podem ser dados do usuário.

O `OnDisconnection` é o evento responsável por acionar as ações quando o *add-in* for desconectado e descarregado da memória principal, é neste momento que o suplemento deve fazer a limpeza de qualquer recurso que tenha sido usado e precise ser descarregado. Os parâmetros utilizados neste evento são: uma constante que define qual o modo que o *add-in* foi desconectado, que pode ser quando o aplicativo hospedeiro fechar, quando o usuário o desconectar manualmente ou por um controlador de automação; e mais um conjunto de dados que podem ter sido definidos pelo usuário.

Outro evento que pode ser acionado é o `OnAddInsUpdate`. Executado quando o conjunto de *add-ins* COM do aplicativo hospedeiro é alterado de alguma forma, seja com a instalação ou remoção de um novo *add-in* ou com a atualização de qualquer um já existente.

Existem ainda mais dois métodos que podem ser chamados quando o *add-in* é conectado ou desconectado do aplicativo hospedeiro. Um deles é o `OnStartupComplete` que é invocado somente se o suplemento for conectado durante a iniciação do aplicativo hospedeiro. O outro é o `OnBeginShutdown` que só é chamado quando o *add-in* for desconectado durante o encerramento do programa hospedeiro. A utilização desses métodos pode ser a única maneira de executar determinadas ações junto ao usuário, que não possam ser executadas nos eventos `OnConnection` e `OnDisconnection`.

Outro fator importante para a construção de um suplemento para o MS Office é que ele necessita de uma chave própria no registro do Microsoft® Windows para cada aplicação que irá suportar esse suplemento [SuplementoCOM 2007].

O SDK do MS Project foi utilizado no projeto do add-in por incluir a documentação necessária desenvolver um suplemento para o MS Project. Alguns exemplos de ações comuns também fazem parte desse pacote. Faz parte desse conjunto de informação [ProjectSDK 2007]:

- Project Guide 101: explica a arquitetura do Project Guide, bem como mostra como criar e implementar guias e visões para Project 2003;
- Microsoft Office Project Visual Basic Reference: mostra o modelo de objetos em forma de tópicos para pode se programar usando Visual Basic for Application (VBA);
- XML Schema Reference: referência para o esquema XML que pode ser usado com o VBA para validar dados exportados em XML.

5.3 Esquema do arquivo de *Level* do Virtual Team

A integração do MS Project com o jogo Virtual Team se dá com a geração de um arquivo no formato XML que contém todas as informações necessárias para que a simulação possa ser executada e então gerar resultados. São os chamados arquivos de *level* do jogo que seguem um esquema próprio definido pelo Game Design. A seguir serão enumerados os componentes que estão presentes no esquema XML (*XML Schema*) do jogo Virtual Team e mais bem descritos os que representam os dados referentes ao projeto que será simulado, como: o projeto e as atividades.

A primeira entidade que compõe o esquema do jogo é a *level* que é a raiz da árvore do documento e é composto por conjunto de entidades. Essas entidades podem ser: o projeto a ser simulado, com seus diversos atributos; a equipe do projeto, descrevendo cada um dos integrantes; e os objetivos do jogo, que no caso é o da simulação descrita neste arquivo. *Entities* é o elemento da árvore que pode possuir os elementos que são entidades de um projeto, identificado pelo nome *project*, e o conjunto de atores sintéticos, nomeado de *syntheticAgents*.

Descrevendo melhor o elemento projeto, ele possui os atributos: *name*, que identifica o projeto na interface com o usuário, *beginDate, currentDate, endDate*, que indicam, respectivamente, qual a data de início, a data atual e a data de finalização do projeto para o simulador, *budget*, que representa o orçamento previsto para o projeto simulado, *currentCost, currentEarnedValue, currentPlannedValue*, que respectivamente representam o custo atual, o valor agregado corrente e o atual valor planejado para o projeto que o jogo está simulando e *managerWage, analystWage, architectWage, engineerWage*, representam as remunerações das funções de gerente, analista de sistema, arquiteto de software e engenheiro de software do jogo e *currentPhase*, representa a fase atual do projeto, entendendo fase como uma das fases do processo unificado [RUP 2006]. Os elementos que o projeto pode ou deve possuir são: *description*, uma possível descrição para o projeto, *communicationPlan*, o plano de comunicação que define como será feito a comunicação no projeto, *phases*, as possíveis fases do projeto e *acceptanceRequirements* que são os requisitos para a aceitação do projeto.

O plano de comunicação do projeto contém como elementos a sua descrição (*description*), que explica como será o plano, a forma e a periodicidade com que as informações chegam ao cliente e à equipe de projeto. Pelo esquema definido, a única forma é a reunião.

Uma fase (*phase*) do projeto é composta, como elemento do arquivo, por uma descrição e um conjunto de atividades. Este elemento ainda possui dois atributos: o nome (*name*) e um identificador (*id*).

O elemento *activities* representa o conjunto de atividades do projeto e pode ser composto por outros dois tipos de elementos: o *leafActivity* e o *compositeActivity*. A atividade folha é a unidade da árvore de atividades, ou seja, não existe nenhuma atividade

filha dela na hierarquia. Já a atividade composta, como o próprio nome indica, é uma atividade que agrega um conjunto de sub-atividades.

A atividade folha define o pacote de trabalho a ser executado por um ator sintético do jogo. Ele possui os seguintes atributos: *id*, que é o identificador da atividade, *role*, que representa o tipo de habilidade necessário para desempenhar aquela atividade e *estimatedBeginningDate*, que é a data estimada para o início da execução da atividade. Além desses atributos as atividades podem ou devem possuir os seguintes elementos: *name*, pois é necessário ter o nome da atividade, *description*, que é a descrição da atividade e *dependences* que são as dependências das atividades em relação às outras. A atividade composta também possui os elementos *name, description* e *dependences*, porém possui um elemento a mais que seria a *leafActivity*, que no arquivo XML vai representar o conjunto de sub-atividades. Estes dois tipos de atividade, folha e composta, ainda possuem mais um elemento, o *artifact*, que representa o produto daquela atividade.

O artefato (*artifact*) produto de uma atividade possui como atributos a sua completude (*completeness*), a quantidade de erros (*errorAmount*), o tamanho atual (*actualSize*), a completude (*completeness*) e o tamanho estimado (*estimatedSize*), esses dois últimos auxiliam nos cálculos sobre o quanto um ator sintético irá trabalhar nele, se será mais ou menos eficaz ou eficiente.

Os requisitos de aceitação do projeto são utilizados para definir se o andamento do projeto no simulador será aceito ou não, determinando se o jogador irá ganhar ou não, além dos objetivos do jogo. O *acceptanceRequiriments* possui dois elementos, o percentual da quantidade de erros máxima (*errorAmount*) e o percentual de atraso máximo do e tempo do projeto (*delay*).

Outros elementos do arquivo de esquema para geração do XML utilizado para simulação são referentes à equipe de projeto e definem os atores sintéticos em termos de descrição, habilidades e comportamento. Um dos seus principais elementos é o *SyntheticAgent* que possui entre outros esses atributos: *id, name, wage, age, sex, energy, motivation*. Além dos atributos que caracterizam o ator sintético, existem os elementos que compõem o mesmo, os mais importantes são: *profile, observations, personality, skills, moods, attitudes* e *goalValues*. Todos esses elementos contribuem para que a

simulação do ator sintético seja a mais interessante possível para os resultados da simulação.

5.4 Requisitos e Arquitetura do *Add-in*

Para o desenvolvimento do *Add-in* que integra o Virtual Team ao MS Project foi utilizando uma arquitetura do padrão MVC (Model-View-Controller) [MVC 2007]. Esse modelo de arquitetura possibilita a separação das classes em três camadas distintas. Alguns requisitos foram levantados para especificar o funcionamento do suplemento para o MS Project. Além disso, a linguagem de programação escolhida satisfaz uma estruturação de código baseado no paradigma orientado a objetos, e foi também por esse motivo que houve a possibilidade da utilização do padrão MVC.

Outro componente importante para o *Add-in* foi o de *Wizard*, que possibilitou a geração de um passo-a-passo para as classes de interface com o usuário, e com isso receber as informações que não são retiradas do arquivo de projeto do MS Project. O componente utilizado para a criação do *wizard* foi o TSWizard (a wizard framework for .Net) [TSWizard 2007].

A seguir serão descritos os requisitos, como se utilizar o componente wizard para a implementação e por fim será mostrada a arquitetura do *add-in*.

5.4.1 Requisitos

Para o desenvolvimento do *add-in* alguns requisitos funcionais foram levantados para um melhor entendimento do produto final a ser produzido. Esses requisitos levaram em consideração a retirada de informação de um arquivo do MS Project, a entrada de informação pelo usuário para completar o arquivo de simulação, e a geração desse arquivo de simulação. Abaixo seguem os requisitos:

[RQ1] **Captar informações sobre o projeto** – para poder preencher as informações sobre o nome do projeto e outras. Será retirado direto do arquivo de origem no MS Project;

[RQ2] **Captar as atividades que compõem o projeto** – as tarefas de um projeto são alguns dos elementos mais importantes do projeto, pois definem o que tem que ser

feito para se completar o escopo do projeto. Este requisito irá captar do MS Project as atividades do projeto ativo, levando em consideração a sua hierarquia.

[RQ3] **Solicitar ao usuário informações complementares sobre o projeto** – Algumas informações sobre o projeto podem ser modificadas, como o nome e a descrição, e outros deverão ser solicitadas para completar o projeto. Informações sobre o orçamento do projeto e os requisitos de aceitação são algumas dessas informações.

[RQ4] **Solicitar ao usuário o Plano de Comunicação** – O plano de comunicação do projeto define quais as informações que serão importantes aos interessados no projeto para que os mesmos estejam cientes e como essas informações irão circular nesse projeto, como em reuniões, cartas, e-mails e outros. Essas informações serão captadas pelo *Add-in* em uma interface com o usuário para preenchimento dos campos solicitados.

[RQ5] **Solicitar informações sobre a remuneração dos recursos** – Os recursos humanos que compõem a equipe de projeto no jogo Virtual Team são compostos por um analista de sistemas, um arquiteto de software, engenheiros de software e o gerente que é representado pelo jogador. Os custos desses recursos devem ser levados em consideração no projeto simulado. Então serão solicitadas ao usuário as informações necessárias para este fim.

[RQ6] **Gerar arquivo XML para simulação no jogo Virtual Team** – O jogo Virtual Team faz leitura de um arquivo XML contendo as informações sobre o projeto, a equipe e os objetivos do jogador. Para isso, o *add-in* terá que ser capaz de captar as informações básicas, necessárias para a simulação, do projeto ativo no Microsoft® Project e gerar o arquivo no formato XML, passível de simulação.

[RQ7] **Utilizar um wizard** – para que as informações necessárias à complementação do arquivo a ser simulado pelo jogo. Uma seqüência de telas de diálogo com o usuário (o *wizard*) será usada para receber essas informações.

[RQ8] **Ser implementado em C#** - o desenvolvimento de um suplemento para os aplicativos do Microsoft® Office requer a utilização de uma das seguintes linguagens: Visual Basic, Visual J#, Visual C++ ou Visual C#. O desenvolvimento do *add-in* se dará utilizando a linguagem Visual C# por ser uma

linguagem orientada a objetos com boa sintaxe, poder usar diretamente DLLs e ter semelhança com a linguagem Java.

[RQ9] **Estar sob licença *open source* –** seguindo o tipo de licença utilizado pelo jogo Virtual Team e para estimular a criação de uma comunidade que contribua com a continuação do projeto e a sua licença de distribuição o *add-in* também terá este tipo de licença.

5.4.2 Wizard

Para a construção do *wizard* do *add-in* do Virtual Team foi utilizado o componente TSWizard [TSWizard 2007], que é um componente desenvolvido em C# e que implementa as telas e atributos necessários para a construção de um *wizard.* Esse componente teve sua criação iniciada em 2002 e está sob a licença BSD [BSD 2007].

Ele é um *wizard* bem comum, pois consiste de uma janela principal (*BaseWizard*) que contém etapas individuais (*BaseStep*) para formar o passo-a-passo. Essas etapas podem ser navegadas de acordo com a ordem definida.

As páginas do *wizard* ainda podem possuir dois tipos diferentes de formas (*layouts*), uma para as páginas internas e outro para as páginas externas. O TSWizard foi modelado para que o usuário possa escolher da forma que lhe for mais conveniente os *layouts* de página. A construção desse *wizard* foi baseado na especificação da Microsoft para *wizards* [Wizard 97].

- Páginas Externas: Geralmente são utilizadas nas páginas inicial e final do *wizard*, como uma introdução e uma finalização (agradecimentos, etc.). É composta por uma imagem lateral (lado esquerdo) com um logo lateral também. Pode conter um texto com fonte maior para o título. No caso da página inicial, pode haver um ou dois parágrafos descrevendo qual o intuito daquele *wizard*. Se for a página final, pode haver uma lista dos resultados de cada etapa executada, avisando que foi finalizado ou mesmo dando instruções ao usuário do que efetuar após o *wizard*.
- Páginas Internas: são utilizadas para as etapas do *wizard*, captando informações ou executando tarefas. No topo há o título da etapa, bem como pode haver uma pequena descrição dessa etapa em uma ou duas linhas. Um pequeno logo pode aparecer no canto superior direto para especificar ainda mais a etapa. Por fim

existe uma barra inferior com os botões de controle do *wizard*, que são o cancelar, voltar e avançar.

A utilização do *wizard* se dá com essa captação de informações ou execução de tarefas em cada etapa e no final pode mostrar ao usuário o que foi executado, ou pedir ao mesmo que finalize e então o *wizard* executa o que lhe foi solicitado ou informado. Neste último caso as informações de cada etapa são resgatadas na etapa de finalização e então executadas.

5.4.3 Arquitetura

O *Add-in* foi implementado sobre o padrão de projeto MVC, que como já mencionado possui três camadas distintas. O intuito desse padrão é separar totalmente o modelo da aplicação, a lógica de programação e as classes de interface com o usuário. Toda a lógica da aplicação, as classes básicas, que representa o modelo do seu estado atual ficam na camada de Modelo (Model). Já a camada de Visão (View), incorpora as classes referentes à interface com o usuário. E por fim a camada de Controle (Controller) fica responsável por abrigar as classes de fluxo da aplicação, pois capta as informações das classes da camada de visão e com isso modifica o modelo atual da aplicação na camada de Modelo. A Figura 5.4-1 exibe uma macro-visão das camadas do padrão de projeto Model-View-Controller e suas interações.

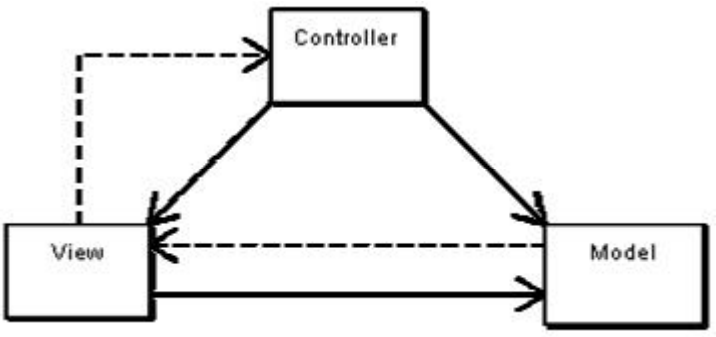

Figura 5.4-1 - Diagrama simples mostrando o relacionamento entre as camadas do MVC [MVCEN 2007].

Camada Model

A camada de modelo (*Model*) do *add-in* possui as classes que modelam as estruturas básicas dele. Cada uma dessas classes encapsula o modelo que receberá uma determinada informação do MS Project e após isso será utilizada para gerar o arquivo XML a ser simulado pelo jogo Virtual Team.

A classe `VTProject` é a principal classe do modelo do suplemento. Ele encapsula as informações referentes ao projeto ativo no MS Project, como: o nome, a data de início do projeto, data prevista de finalização, data corrente do projeto e o orçamento estimado para o projeto. Além desses dados ainda existem algumas variáveis que são fornecidas pelo o usuário, como: a descrição, a quantidade de erros possíveis para o projeto, e o atraso máximo possível, estas variáveis são importantes para a simulação do projeto e são captas de acordo com o interesse do usuário. Estruturas compostas também fazem parte da classe `VTProject` são elas as classes: `VTCommunicationPlan`, `VTWages` e um conjunto de objetos atividades referentes à classe `VTActivity`.

O plano de comunicação do projeto que for cadastrado pelo usuário será encapsulado na classe `VTCommunicationPlan`. São três as informações necessárias para se criar um objeto dessa classe, são elas: um texto que define a descrição do plano, ou seja, uma descrição de como vai ser feita a comunicação no projeto, a periodicidade das reuniões para a comunicação entre a equipe de desenvolvimento e das reuniões para que sejam efetuados os comunicados ao cliente sobre o projeto.

A classe `VTWages` guarda as informações sobre as remunerações dos cargos dos recursos humanos do projeto. Existe a remuneração do gerente do projeto, que é uma figura representada pelo jogador, mas o valor de sua remuneração deve ser levado em consideração para os custos do projeto. Além dessa remuneração ainda existem as dos cargos: de analista de sistema, de arquiteto de software e a de engenheiro de software, este último desempenha o papel de programador do projeto.

Para o projeto do suplemento do MS Project para o jogo Virtual Team foi utilizado também outro padrão de projeto muito utilizado para se montar uma estrutura em árvore, o padrão *Composite* [Composite 2007]. Nesse padrão existe uma classe que representa a classe principal (*Component*) e mais outras duas classes que herdam dessa classe

principal, a que representa a folha (*Leaf*) e a que vai representar um nó composto (*Composite*). A Figura 5.4-2 mostra como se deve ser utilizado esse padrão de projeto.

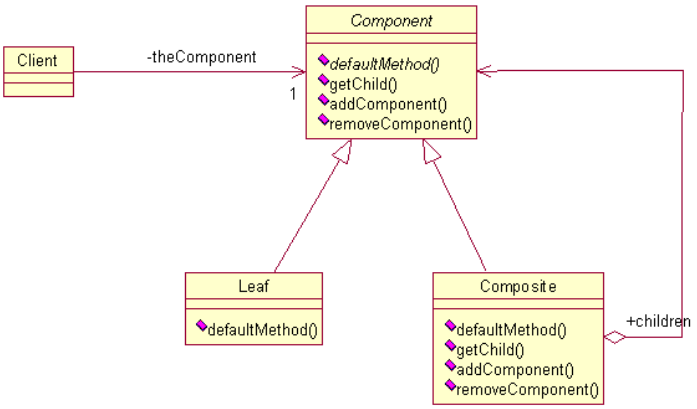

Figura 5.4-2 - Padrão de projeto Composite [Composite 2007].

Esse padrão foi utilizado no projeto para modelar a árvore de atividades de um projeto. A classe que representa o *Component* no projeto do suplemento é a classe `VTActivity` que é a raiz das atividades do projeto. Essa classe possui como atributos o nome da atividade (`name`), um identificador da atividade (`id`) e uma descrição da atividade (`description`). A atividade folha é encapsulada na classe `VTLeafActivity` e possui os seguintes atributos: a data de início da atividade (`beginningDate`) e a data de finalização da atividade (`endingDate`). As atividades compostas por outras são definidas pela classe `VTCompositeAcitivy` que tem como atributo o conjunto de objetos das classes filhas (`children`). A arquitetura da camada Model do add-in pode ser visualizado na Figura 5.4-3.

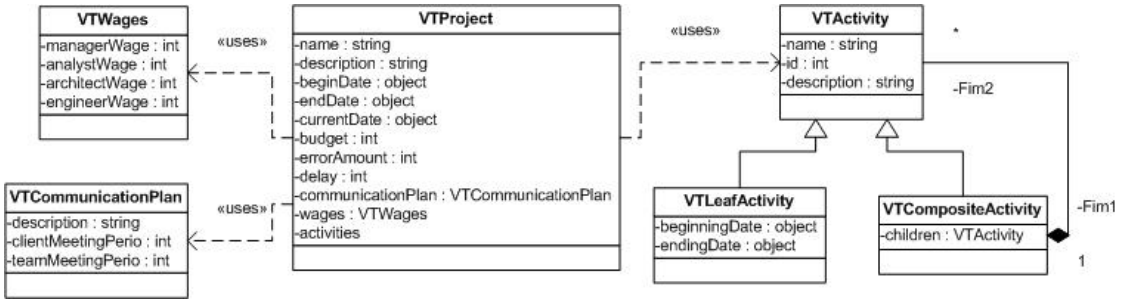

Figura 5.4-3 - Classes da camada Model do add-in

Camada View

A camada de visão do projeto é composta pelos formulários inseridos nas janelas das páginas interiores do *wizard* criado. Estas classes herdam dos tipos definidos pelo o TSWizard de acordo com a página que será exibida, se será uma página externa ou página interna. As classes da camada View podem ser visualizadas na Figura 5.4-4.

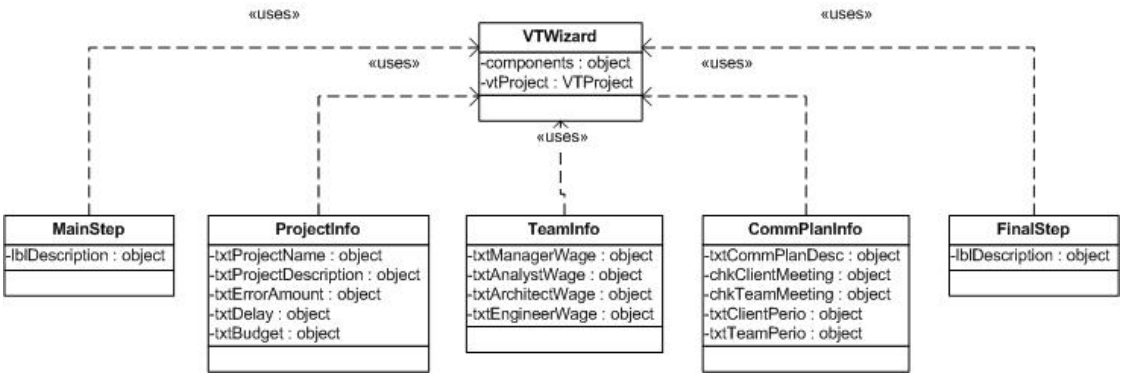

Figura 5.4-4 - Classes da camada de visão

A classe principal do *wizard* é a `VTWizard` que herda de `BaseWizard` do pacote `TSWizards`. É nessa classe que são adicionadas as outras classes referentes às etapas do *wizard*. Para se adicionar as etapas se utiliza o método `AddStep`, que recebe como parâmetros um nome para a etapa e o objeto da classe que implementa a etapa mencionada.

Para as páginas internas foram construídas classes para cada uma delas. A primeira delas é `ProjectInfo` que recebe as informações complementares do projeto a ser simulado. Os campos dessa classe são: o nome do projeto (que é uma opção para o usuário trocar o nome), a descrição do projeto, o orçamento que o usuário deseja para o projeto e os requisitos de aceitação (quantidade de erros máxima e atraso máximo). A Figura 5.4-5 mostra a tela com os campos dessa classe.

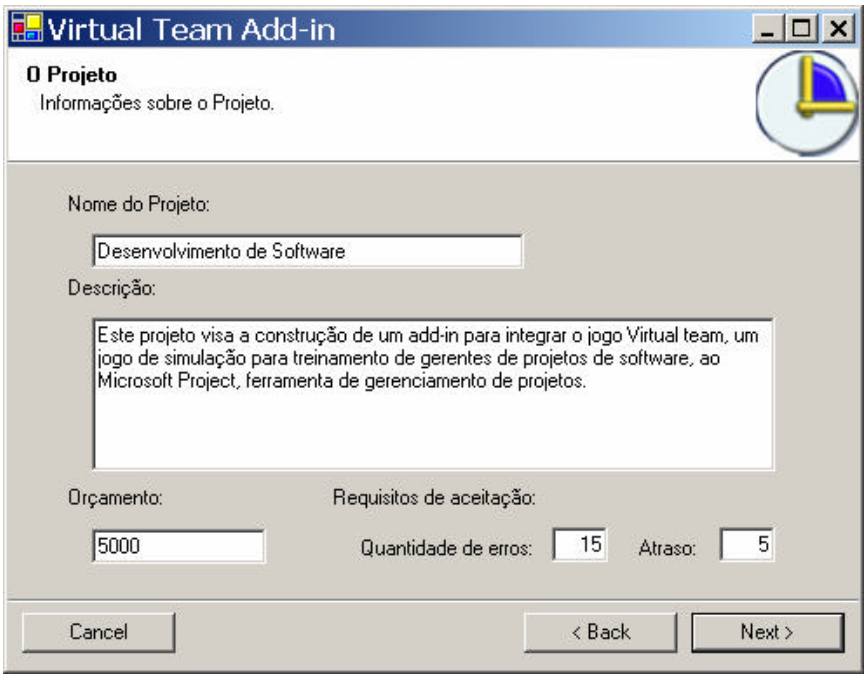

Figura 5.4-5 - Tela da primeira etapa do *wizard*. Informações sobre o Projeto.

Outra classe da camada de visão do *add-in* é a `TeamInfo` que contém os campos necessários para a captação de informações sobre as remunerações dos recursos humanos do jogo. As remunerações solicitadas para os recursos humanos do jogo são: a do gerente de projeto, analista de sistema, arquiteto de software e desenvolvedores. A tela é mostra a seguir na Figura 5.4-6.

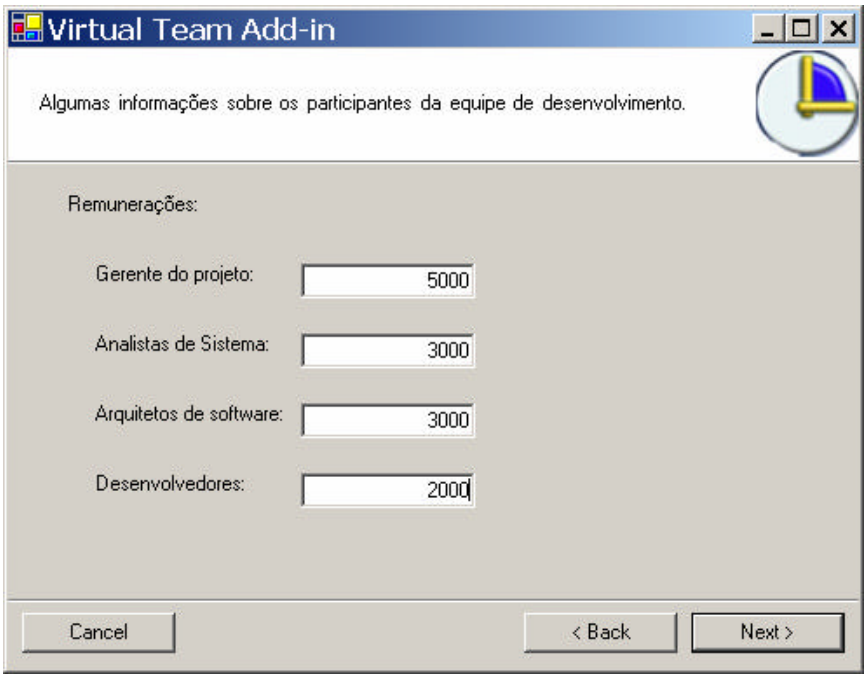

Figura 5.4-6 - Segunda etapa do *wizard*. Informações sobre a equipe, neste caso as remunerações.

A terceira etapa de recebimento de informação do *wizard* é implementada na classe `CommPlanInfo`. Esta classe é responsável por guarda as informações dos campos com dados sobre o plano de comunicação que será utilizado no jogo Virtual Team. Além da descrição que representa o texto do plano de comunicação, a tela dessa classe recebe as periodicidades com que as reuniões com clientes e equipe de desenvolvimento serão feitas. A figura que mostra a tela implementada por essa classe é a Figura 5.4-7.

Com relação ao período com que haverá as reuniões no simulador do jogo, isso poderá ocorrer ou não. Então, para que seja possível o usuário escolher como se comunicar com os interessados e outros parâmetros que serão necessários para cada tipo de comunicação, na tela da etapa de plano de comunicação será exibido *checkboxes* para que o usuário possa marcá-los. A princípio só há uma opção definida no arquivo de esquema para o XML gerado no jogo Virtual Team para a comunicação com o cliente e a equipe de projetos, que é a reunião e seu parâmetro é a periodicidade.

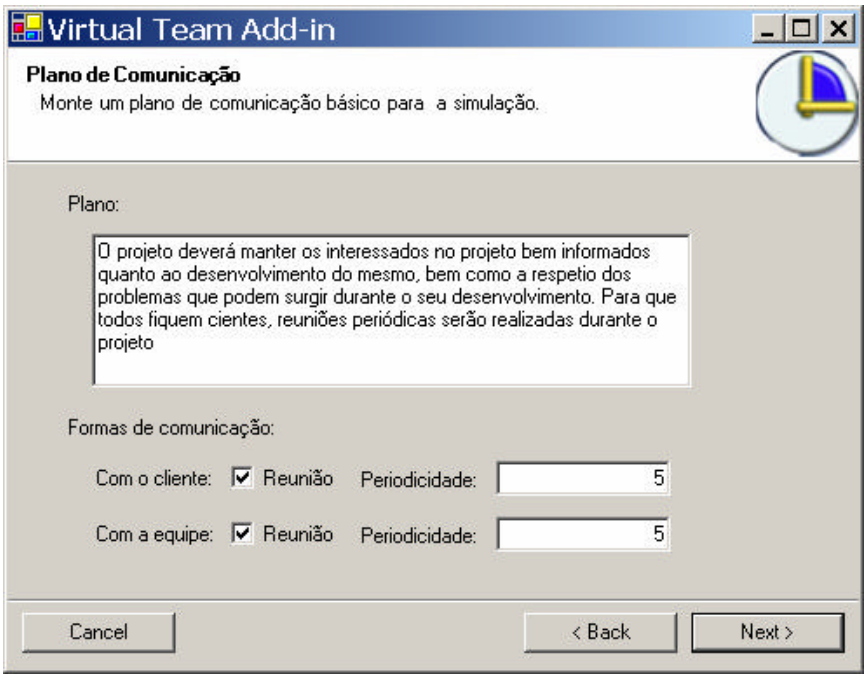

Figura 5.4-7 - Terceira etapa do *wizard*. O plano de comunicação do projeto a ser simulado.

Além das classes interiores do *add-in*, as que foram utilizadas para os formulários, existem ainda as classes externas. A classe `MainStep` foi criada para representar a página de apresentação do *wizard* e herda da classe `BaseExteriorStep`. Esta classe possui dois componentes *Label*, um para o título do *wizard* e o outro para a sua descrição. Da mesma forma ocorre com a classe `FinalStep`, que herda também da classe `BaseExteriorStep`, e neste caso essa classe foi utilizada para mostra a finalização do *wizard* e solicitar ao usuário que finalize para que seja processada a criação do arquivo XML para o simulador. A Figura 5.4-8 representa a tela da classe MainStep, que são as boas vindas ao add-in, e a Figura 5.4-9 é a tela da classe FinalStep, que finaliza o wizard e gera o arquivo XML.

Figura 5.4-8 - Tela de boas vindas do Virtual Team Add-in.

Figura 5.4-9 - Tela de finalização do wizard para a geração do arquivo de simulação.

Camada Controller

Ainda existem mais duas classes no projeto que fazem o controle do que vem da camada View para a modificação do modelo do projeto e geração do arquivo para a simulação. A classe `ProjectControl` é responsável por receber um projeto do MS Project e encapsula-lo nas classes de projeto do *add-in*. Os métodos mais importantes desta classe são: `createProject()` e `addActivities()`, o primeiro capta as informações sobre o projeto enquanto a segunda pega as informações das atividades e insere na classe de `VTProject`.

A outra classe da camada Controller é a `XMLControl`, esta classe também possui dois métodos, um principal e um auxiliar. O método principal é o que gera todo o arquivo XML para o simulador do jogo e o outro método auxilia o primeiro na inserção das atividades do projeto, utilizando a técnica de recursão para percorrer a árvore das atividades e escrever corretamente no arquivo XML. Caso a atividade seja folha, então o arquivo possui uma sintaxe, caso seja uma atividade composta terá outra sintaxe. A Figura 5.4-10 mostra as duas classes que compõem a camada Controller.

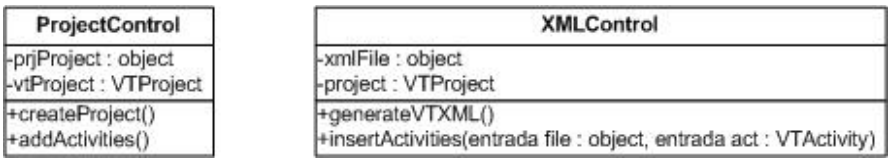

Figura 5.4-10 - Classes da camada Controller

Além das classes que foram modeladas dentro do padrão do projeto. Existem ainda as que são referente ao projeto de um *add-in* definidos pelo Visual Studio e que foram mencionadas anteriormente. Na classe de conexão do *add-in* (`Connect`) foram instanciados os objetos das classes `VTWizard` e `ProjectControl`, a primeira para iniciar o *wizard* do *add-in* e a segunda para receber o projeto ativo do MS Project e repassar para o objeto do passo-a-passo.

5.5 Resultados

Nesta seção serão exibidas as imagens para demonstrar a utilização do *wizard* no MS Project e quais os resultados da captura de informações do gerenciador de projetos no jogo Virtual Team. A Figura 5.5-1 mostra o botão no Microsoft® Project que inicia o *add-in* do jogo.

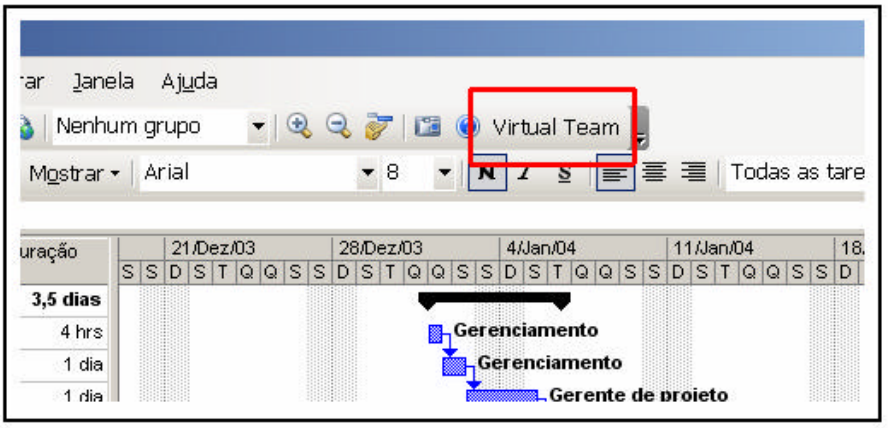

Figura 5.5-1 - Botão de inicialização do *Add-in* no MS Project.

A tela da Figura 5.5-2 mostra a tela de boas vindas quando o suplemento foi inicializado no Microsoft Project com um projeto ativo onde está planejado o desenvolvimento de um software.

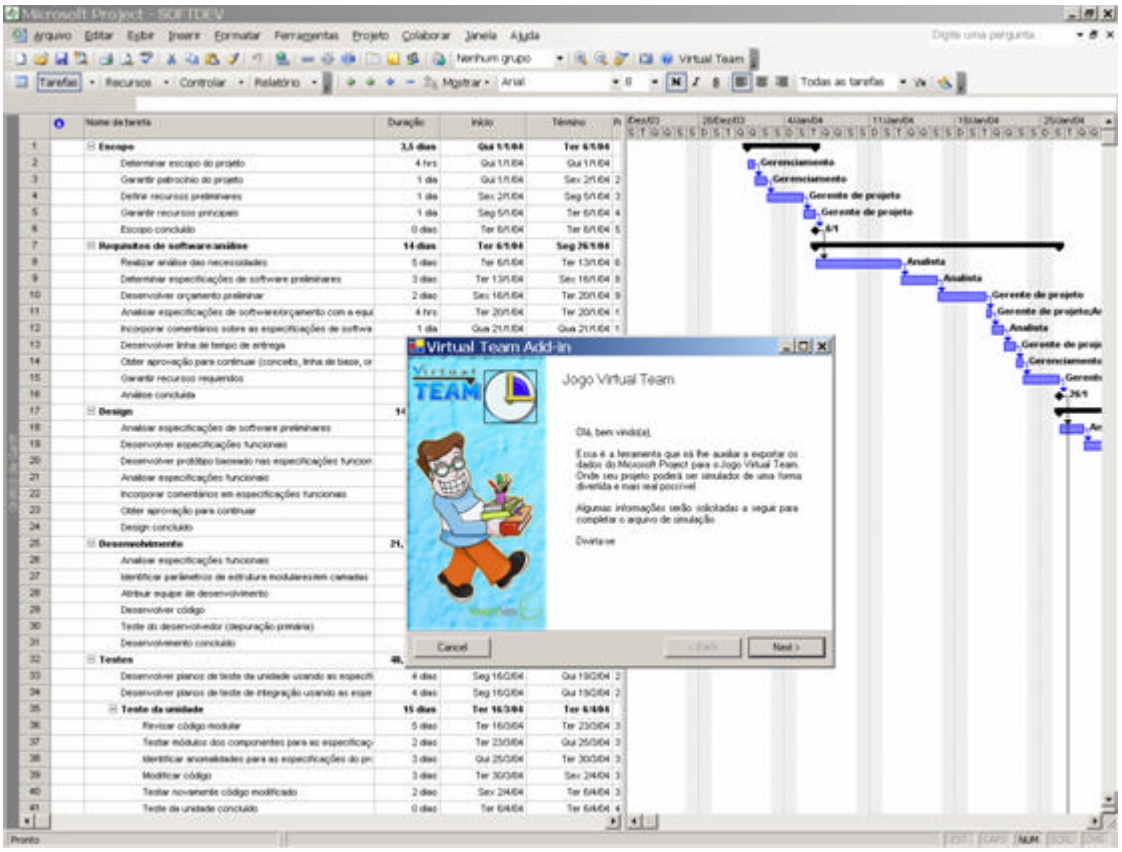

Figura 5.5-2 - Jogo Virtual Team *Add-in* para Microsoft® Project.

No jogo Virtual Team os dados captados pelo *add-in* podem ser visualizados em diversos locais. Outros dados não poderão ser exibidos, porém são utilizados para a simulação. A seguir a Figura 5.5-3, que representa a tela que mostra a descrição do projeto no jogo. Lembrando que os dados que estão sendo exibidos nas telas do jogo são os mesmo que foram mostrados, anteriormente, nas telas do *wizard* .

Figura 5.5-3 - Descrição do projeto no jogo Virtual Team.

A seguir, na Figura 5.5-4, se tem a tela com as atividades no jogo Virtual Team, essas atividades são as que foram extraídas do MS Project para a geração do arquivo a ser simulado. Na imagem é possível verificar as atividades folha e as atividades compostas.

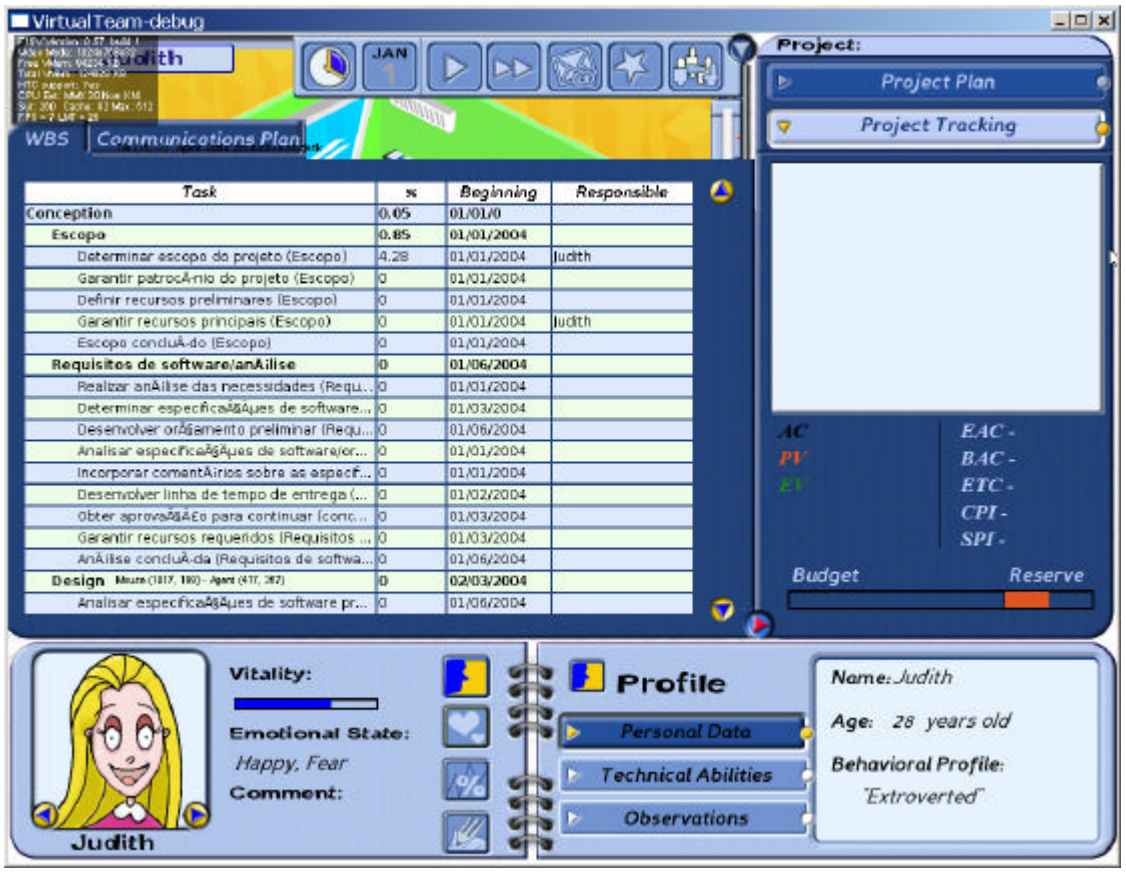

Figura 5.5-4 - As atividades do projeto simulado no jogo.

E por fim a tela da Figura 5.5-5 que mostra a descrição do Plano de Comunicação do projeto simulado no jogo. Verificando-se, mais uma vez, que somente é exibido a descrição do plano, as outras informações que foram solicitadas ao usuário no *add-in*, que foram as periodicidades das reuniões, estão sendo utilizadas na simulação.

Figura 5.5-5 - Plano de comunicação no jogo. Somente a descrição do plano.

6 Conclusões

O gerenciamento de projeto é uma disciplina muito importante para diversas áreas de atuação. Em especial a área de engenharia de software onde diversos riscos sempre são factíveis de acontecer, principalmente no que diz respeito ao tempo, escopo, custo e recursos humanos.

As iniciativas de criação de ferramentas que auxiliem no estudo da atividade de gerenciamento de projeto são muito importantes. Os simuladores são exemplos dessas ferramentas e podem ser utilizadas de diversas formas, como: na capacitação de gerentes de projeto ou na simulação de projeto para antecipar resultados e auxiliar no planejamento do projeto, minimizando ao máximo os riscos que atrapalhem o sucesso do projeto.

Com a criação do jogo Virtual Team, que simula o gerenciamento de projetos de software com ênfase em gerência de recursos humanos utilizando atores sintéticos, foi aberto um conjunto de oportunidades para uma melhoria do projeto. Principalmente no que diz respeito à captura de informações do mundo real para poderem ser simulados no jogo, pois dão uma maior originalidade e especificidade em cada partida jogada pelo usuário. Além disso, a possibilidade de se ter resultados sobre a simulação que ajudam no planejamento do projeto tornou interessante se ter estudos nesta área.

Com o objetivo de aproximar mais o mundo real das ferramentas de simulação, em especial o jogo Virtual Team, é que foi dada partida na construção de um *plug-in* para integrar uma ferramenta de gerenciamento de projetos a esse jogo. Principalmente um suplemento para uma das ferramentas mais utilizadas atualmente na atividade de gerência de projeto, inclusive no âmbito da engenharia de software, que é o Microsoft® Project 2003.

Com isso o treinamento de gerentes de projeto se tornará mais interessante utilizando na simulação um projeto real que o gerente possa ter contato no seu dia-a-dia. E então permitir que ele possa visualizar situações e resultados antecipadamente, podendo prever melhor o que executar para que situações ruins não cheguem a vir à tona ou atenuar os impactos das que venha a acontecer.

6.1 Dificuldades encontradas

Durante o desenvolvimento do *add-in* do jogo Virtual Team para o MS Project foram encontradas algumas dificuldades. Dentre elas pode-se mencionar e descrever as que foram importantes.

- A aprendizagem de uma linguagem de programação não utilizada anteriormente foi um fator que influenciou muito na conclusão do projeto;
- A dificuldade em se encontrar material didático ensinando como construir add-ins na IDE Visual Studio para as ferramentas do pacote do Office da Microsoft. Além disso, iniciar o projeto utilizando o material encontrado foi outra questão dificultosa;
- Associado à linguagem e ao processo de construção do *add-in* ainda surgiu a necessidade de se estudar a API do Microsoft® Project para poder recuperar os dados necessários para gerar o arquivo de simulação.
- O pequeno espaço de tempo para realizar os estudos necessários, desenvolver a análise, a arquitetura e implementação do projeto foram fatores que contribuíram para a redução do escopo do *add-in*, que poderia ter mais alguns requisitos. Porém, os requisitos mais importantes e interessantes foram desenvolvidos para gerar uma ferramenta executável e utilizável, contemplando informações que já podem ser simuladas no jogo.
- O jogo Virtual Team gera o log da partida jogada, porém não há uma documentação que explique como é feito a construção do mesmo e não é facilmente perceptível que é gerado esse log. Com isso, impossibilitou, para essa etapa do projeto, a construção do módulo do *add-in* que retornar os resultados do jogo para o MS Project, ficando assim para um trabalho futuro.

6.2 Trabalhos futuros

Apesar de o *Add-in* do jogo Virtual Team para o Project já ser uma ferramenta aplicável em um ambiente empresarial real, algumas extensões estão previstas para serem incorporadas ao mesmo e melhora-lo ainda mais. São elas:

1. **Recuperação das informações dos atores sintéticos:** além do projeto, o jogo simula o comportamento dos atores sintéticos. Com isso, a implementação de etapas no *wizard* do *add-in* para resgatar informações para modificar o comportamento dos atores existentes será muito interessante;
2. **Edição de informações de cada atividade:** as atividades possuem algumas informações, como sua descrição, que podem ser alteradas. Então o *wizard* poderá possui etapas que dêem a possibilidade de se fazer essa edição;
3. **Retorno do resultado da simulação:** a geração de um log da partida jogada, pelo Virtual Team, poderá ser lido pelo *add-in* para que os resultados retornem ao MS Project e com isso o usuário poderá visualizar o andamento da simulação do projeto na própria ferramenta de gerenciamento de projetos.

7 Referências Bibliográficas

[GUEDES 2006] GUEDES, MARCELO SANTIAGO. *Um Modelo Integrado para Construção de Jogos de Computador Aplicada à Capacitação em Gestão de Projetos*. Dissertação de mestrado, CIn – UFPE, Recife, 2006.

[ROUSY 2000] ROUSY, D. D. DA S. *Atores Sintéticos em Jogos de Aventura Interativos: O Projeto Enigmas no Campus*. Dissertação de mestrado, CIn – UFPE, Recife, 2000.

[MARTINS 2005] MARTINS, JOSÉ CARLOS CORDEIRO. *Gerenciando projetos de desenvolvimento de Software com PMI, RUP e UML*, 2ª ed.rev., Rio de Janeiro, Brasport, 2005.

[HELDMAN 2004] HELDMAN, KIM. *Gerência de projetos: guia para o exame oficial do PMI*, 2ª ed.erv., Elsevier.

[Abran 2004] A. ABRAN ET AL., EDITORS. *Guide to the software engineering body of knowledge: SWEBOK*. IEEE Computer Society, Los Alamitos, California, 2004.

[Corbett 2003] CORBETT, THOMAS. *Introdução à Dinâmica de Sistemas*. Disponível em: http://www.corbett.pro.br/artigos.asp [Acessado em 08 de março de 2007].

[Forrester 1961] FORRESTER, J.W. *Industrial Dynamics*. MIT Press, Cambridge, MA, 1961.

[SMARTSIM, 2006] *Projeto SmartSim – Simulação em gerência de projetos com atores sintéticos*. Disponível em http://www.cin.ufpe.br/~smartsim. Acessado em 27.11.2006.

[PMBOK 2004] DIVERSOS AUTORES. *Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projeto - PMBOK*, terceira edição. Project Management Institute, Inc. 2004.

[PMK 2006] TORREÃO, PAULA G. B. C. *PMK Learning Environment*, 2006, [online] Disponível em: http://php.cin.ufpe.br/~pmk/hp/portugues/home.php?pagina=victor [Acessado 2 de Junho de 2006].

[PMI 2006] PROJECT MANAGEMENT INSTITUTE. *PMI® Colleges*. [online] Disponível em: http://www.pmi.org/info/GMC_CollegesOverview.asp [Acessado em 21 de janeiro de 2007].

[PMIPE 2006] PMI PERNAMBUCO CHAPTER. *O PMI Institucional.* [online] Disponível em: http://www.pmipe.org.br/web/br/pmi.php [Acessado em 21 de janeiro de 2007].

[PMISP 2006] PMI SÃO PAULO CHAPTER. *O Instituto.* [online] Disponível em: http://www.pmisp.org.br/exe/pmi/instituto.asp [Acessado em 21 de janeiro de 2007].

[PMIREL 2005] PROJECT MANAGEMENT INSTITUTE. *PMI 2005 Annual Report*. [online] Disponível em: http://www.pmi.org/info/AP_AnnualReport.pdf [Acessado em 30 de Janeiro de 2007].

[PDCA 2006] WIKIPEDIA. *Ciclo PDCA*. [online] Disponível em: http://pt.wikipedia.org/wiki/Pdca [Acessado em 04 de fevereiro de 2007].

[RUP 2006] WIKIPEDIA. *RUP*. [online] Disponível em: http://pt.wikipedia.org/wiki/RUP [Acessado em 26 de fevereiro de 2007].

[PLUGIN 2007] WIKIPEDIA. *Plug-in*. [online] Disponível em http://pt.wikipedia.org/wiki/Plug-in [Acessado em 21 de março de 2007].

[ADDIN 2007] MSDN LIBRARY. *How to: Create an Add-in.* [online] Disponível em http://msdn2.microsoft.com/en-us/library/80493a3w(VS.80).aspx [Acessado em 21 de março de 2007].

[SuplementoCOM 2007] MICROSOFT SUPPORT. *Como construir um suplemento Office COM usando Visual Basic .NET.* Disponível em http://support.microsoft.com/kb/302896 [Acessado em 23 de março de 2007].

[ProjectSDK 2007] MSDN. *Microsoft Office Project 2003 Software Development Kit*. Disponível em http://www.microsoft.com/downloads/details.aspx?familyid=4d2abc8c-8bca-4db9-8753-178c0d3099c5&displaylang=en [Acessado em 10 de março de 2007].

[TSWizard 2007] JOHNSON, JAMES T. *TSWizard - a wizard framework for .NET*. Disponível em http://www.codeproject.com/cs/miscctrl/tswizard.asp. [Acessado em 12 de março de 2007].

[Wizard 97] MSDN. *Wizard 97*. Disponível em http://msdn2.microsoft.com/en-us/library/ms738248.aspx [Acessado em 22 de março de 2007].

[MSPROJECT 2007] Microsoft Project. http://www.microsoft.com/brasil/office/project/default.asp. Acessado em 27.11.2006.

[DOTPROJECT] dotProject. http://www.dotproject.net. Acessado em 26.11.2006.

[Faria 1996] FARIA, A. J. E NULSEN, R., 1996. *Developments In Business Simulation & Experiential Exercises*, volume 23. University of Windsor e Xavier University.

[Sterman 2000] STERMAN, JOHN D. *Business Dynamics: System Thinking and Modeling for a Complex World*. McGraw-Hill Higher Education, 2000.

[Sterman 1992] STERMAN, JOHN D. *System Dynamics Modeling for Project Management*. Massachusetts Institute of Technology (MIT), 1992

[Gama et al. 1995] GAMA, E., HELM, R., JOHNSON, R. E VILISSIDES J., 1995. *Design patterns: elements of reusable object-oriented software,* first edition. Addison-Wesley Professional.

[Google 2006] GOOGLE, 2006. *Google Acadêmico Beta*. [online] Disponível em: http://scholar.google.com.br [Acessado 10 de Janeiro de 2007].

[GPL 2006] GPL, 2006. *General Public Licence* [online] Disponível em: http://www.gnu.org/copyleft/gpl.html [Acessado 11 de Agosto de 2006].

[LGPL 2006] LGPL, 2006. *Lesser General Public Licence* [online] Disponível em: http://www.gnu.org/licenses/lgpl.html [Acessado 11 de Agosto de 2006].

[BSD 2007] OPEN SOURCE INITIATIVE OSI. *The BSD License: Licensing*, 2007. Disponível em: http://www.opensource.org/licenses/bsd-license.php [Acessado em 20 de março de 2007].

[Luo et al. 1995] LUO, G., PROBERT, R. L. E URAL, H., 1995. *Approach to constructing software unit testing tools*. Department of Computer Science, University of Ottawa, Ottawa.

[Silva et al. 2006a] SILVA, D. R. D. S., RAMALHO, G. L., E TEDESCO, P. 2006. *Atores Sintéticos em Jogos Sérios: Uma Abordagem Baseada em Psicologia Organizacional.* Proposta de doutoramento apresentada à Universidade Federal de Pernambuco, como parte do programa de pós-graduação em Ciências da Computação, área de concentração Computação Inteligente.

[MVC 2007] WIKIPEDIA, 2007. *MVC – Model-View-Controller*. [online] Disponível em http://pt.wikipedia.org/wiki/MVC [Acessado em 12 de março de 2007].

[MVCEN 2007] ENGLISH WIKIPEDIA, 2007. *Model-View-Controller*. Disponível em http://en.wikipedia.org/wiki/Model_view_controller [acessado em 23 de março de 2007].

[Composite 2007] ENGLISH WIKIPEDIA, 2007. *Composite Pattern.* Disponível em http://en.wikipedia.org/wiki/Composite_pattern [acessado em 23 de março de 2007].

[UML 2007] WIKIPEDIA, 2006. *UML*. [online] Disponível em http://pt.wikipedia.org/wiki/UML [Acessado em 12 de março de 2007]

[WIKI 2006] WIKI, 2006. *Wiki*. [online] Disponível em http://en.wikipedia.org/wiki/Wiki [Acessado 13 de Agosto de 2006].

Data e assinaturas

31 de março de 2007

Este trabalho de graduação é o resultado do esforço do aluno Thierry da Silva Araujo, sob orientação do professor Hermano Perrelli de Moura, com o título de "*Um plug-in que integra um simulador de projetos a uma ferramenta de gerenciamento de projetos*". Todos Abaixo estão de acordo com o conteúdo deste documento.

_____

Thierry da Silva Araujo
(Aluno)

_____

Hermano Perrelli de Moura
(Orientador)