Pós-Graduação em Ciência da Computação

Alberto Ramos de Lima

UMA METODOLOGIA PARA DESENVOLVIMENTO DE COMPANHEIROS VIRTUAIS INTELIGENTES

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

Recife

2009

Alberto Ramos de Lima

UMA METODOLOGIA PARA DESENVOLVIMENTO DE COMPANHEIROS VIRTUAIS INTELIGENTES

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. Hermano Perrelli de Moura

Recife

2009

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

L732m Lima, Alberto Ramos de
Uma metodologia para desenvolvimento de companheiros virtuais inteligentes / Alberto Ramos de Lima. – 2009.
146 f.: il., fig., tab.

Orientador: Hermano Perrelli de Moura.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2009.
Inclui referências e apêndice.

1. Engenharia de software. 2. Ambientes virtuais. 3. Metodologias de desenvolvimento de software. I. Moura, Hermano Perrelli de (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2019-133

Dissertação de Mestrado apresentada por **Alberto Ramos de Lima,** a Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Uma Metodologia para Desenvolvimento de Companheiros Virtuais Inteligentes",** orientada pelo **Prof. Hermano Perrelli de Moura** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

Profa. Cristine Martins Gomes de Gusmão
Departamento de Sistemas Computacionais / UPE

Prof. Hermano Perrelli de Moura
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 27 de agosto de 2009.

Prof. Nelson Souto Rosa
Vice-Coordenador da Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco.

Aos meus queridos pais, Inácio e Fátima, pelo amor incondicional dispensado durante todos esses anos; pelo exemplo de bondade e humildade ensinado; por sempre colocarem minha educação como prioridade, mesmo quando a situação dificultava; por acreditarem em meu esforço e me apoiarem em todas as minhas decisões e por estarem sempre presentes no perto e no longe.

Dedico.

AGRADECIMENTOS

A Deus, pelas maravilhas e bênçãos em minha vida e por me proporcionar forças nessa jornada.

Ao meu orientador, Hermano Perrelli, pelo profissionalismo, compreensão e apoio durante todo o processo.

Aos membros da Banca Examinadora por tão gentilmente terem aceitado o convite de contribuir com este trabalho.

Ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática, seus funcionários e professores.

A Paula Coelho, pelas contribuições dadas no início do trabalho.

Aos colegas de mestrado, em especial, a Márcio Magalhães, companheiro nas aventuras acadêmicas, pela constante paciência em me ajudar e me acalmar nos momentos de ansiedade.

Ao Serviço Federal de Processamento de Dados, em especial a Helena Cristina e Mônica Falcão, pelo incentivo dado durante o curso.

Aos colegas de trabalho, em especial aos integrantes das equipes de Teste e do STI.

Aos meus irmãos, Humberto, Roberto e Estanley, pelo carinho e dedicação dispensados, por acreditarem na minha capacidade e pelo constante incentivo em persistir nos estudos.

À minha querida tia Irene, por sempre expressar o orgulho de me ter como sobrinho e pela constante alegria que me passa quando vou a Campina Grande.

Ao meu grande amigo Madson, o irmão que escolhi ter, meu porto seguro em Recife, quem sempre me acolheu quando necessitei. Agradeço pelos ensinamentos incessantes; pela constante companhia nos momentos de diversão, como também, pelas palavras de conforto, nos momentos tristes; por sempre me fazer acreditar que sou capaz, mesmo quando tudo parece impossível. Enfim, obrigado pelo exemplo de pessoa determinada que tive e tenho a possibilidade de conviver.

A Márcio Andresson, pelo exemplo de perseverança, esforço e trabalho que tem demonstrado; pelo constante incentivo em persistir e pelo ombro amigo e sincero doado sempre que necessitei. Obrigado pela presença fundamental em minha vida nos últimos meses e pelas palavras motivadoras ditas durante esse tempo.

A Adélia, minha *darlinha* querida, pelo exemplo de verdadeira amizade e pelo humor saudável com o qual me faz rir diariamente, mesmo quando estamos falando "negatividades". Agradeço pelo companheirismo e amor sem limites que sempre dedicou à minha pessoa.

A Marcela, minha grande amiga, por sempre estar disposta a me ouvir, me entender, me consolar e me aconselhar. Pela maneira doce e meiga, características de sua personalidade, com que conduz nossa amizade, fazendo com que estejamos a cada dia que passa mais unidos.

Aos demais amigos, tanto os de mais longas datas, Roberta, Pedro e Milena, como os de datas mais recentes, Artur e Ronaldo, agradeço imensamente pela amizade sincera e o apoio constante.

Enfim, a todas as pessoas que direta ou indiretamente contribuíram para o alcance desse objetivo.

RESUMO

Com o notável crescimento da educação à distância (EAD) nos últimos anos, a incorporação de novas tecnologias computacionais de comunicação possibilitou o desenvolvimento dos ambientes virtuais de aprendizagem (AV) como novos meios de apoio ao aprendizado não presencial. Esses sistemas oferecem um conjunto de tecnologias de informação e comunicação, que permitem desenvolver as atividades no tempo, espaço e ritmo de cada participante. Nesse contexto, foram desenvolvidos os Companheiros Virtuais Inteligentes (CVI), que são agentes pedagógicos com o papel de ajudar o aluno. O CVI engloba a questão da afetividade e é dele a atribuição de acompanhar o aluno na realização de suas tarefas, buscando estabelecer relações de empatia e mantê-lo motivado a persistir no processo de interação. Hoje, no entanto, observamos que a necessidade de se integrar CVIs a AVs deixou de ser exclusiva do meio da educação, passando a encontrá-los em diversos domínios de aplicação, como, por exemplo, na saúde, indústria, prestação de serviços, entre outros. Todavia, ainda há dificuldades na construção desse tipo de solução tecnológica, visto que, a pesquisa em busca de processos, metodologias e técnicas que facilitem o desenvolvimento dele parece ser incipiente. Sendo assim, o objetivo dessa dissertação é propor uma metodologia para construção de CVIs, tomando como base a análise de metodologias tradicionais e ágeis de desenvolvimento de software, bem como de metodologias orientadas a agentes. Uma análise comparativa entre a metodologia proposta e metodologias já existentes também é realizada.

Palavras-chaves: Companheiros Virtuais Inteligentes. Ambientes Virtuais. Metodologias de Desenvolvimento de Software.

ABSTRACT

Based on the growing modality of distance education in the last decades, new computational technology incorporation in communication has enabled the development of virtual learning environments through new support methods to non-presence learning. These softwares offer an array of information Technologies and communication, allowing the development of activities within time, space and each participant´s pace. In this context, Intelligent Virtual Companions (IVC), pedagogical agents whose role is to assist users. The IVC encompasses afectivity and its main attribution is to follow the user in task completion, establishing empathy bonds, by keeping him/her motivated in pursuing the process. Today, however, it is observed the intrinsic need of integrating IVCs and Virtual Environments (VE), not only in the educational field, but also in diverse areas of application, such as health, industry, third sector, among others. All in all, there are still a number of factors in constructing this type of software, once its research lacks a more consistent and coherent analysis. In this way, the aim of this thesis is to present a IVC development methodology, taking into account the comparison between traditional and agile software methodologies, as much as agent-oriented methodologies.

Keywords: Intelligent Virtual Companions. Virtual Environments. Software Development Methodologies.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo introdutório visa apresentar um panorama sobre o tema abordado e trabalhado nesta dissertação, ressaltando, na Seção 1.1, a motivação que originou o desenvolvimento deste trabalho, bem como o contexto atual da pesquisa. Além disso, é neste primeiro capítulo que encontramos os objetivos do trabalho, explicitados na Seção 1.2, a metodologia usada para atingir tais objetivos e a estrutura da dissertação, descritas nas seções 1.3 e 1.4, respectivamente.

1.1 Motivação e Contexto da Pesquisa

Constantemente, formas distintas de comunicação são desenvolvidas e aperfeiçoadas. Inovações recentes ocorreram na Tecnologia de Informação e Comunicação (TIC), com destaque para o desenvolvimento de interfaces cognitivas e a inteligência artificial. Estas mudanças trouxeram a criação de programas de troca de informações entre dois ou mais indivíduos, visando mesmo interesse [Moran, 2000]. Foi nesse contexto em que a Educação à Distância (EaD) [Litto e Formiga, 2009] se aprimorou e cresceu muito nos últimos anos [Moore, 2007]. E os principais fatores que contribuíram para este crescimento foram as facilidades proporcionadas pelo desenvolvimento tecnológico.

A incorporação dessas novas tecnologias computacionais de comunicação possibilitou o desenvolvimento dos Ambientes Virtuais (AV) como novos meios de apoio ao aprendizado à distância. De acordo com Arbex e Bittencourt (2007), os AVs são cenários que possuem interfaces instrucionais para a interação de alunos, além disso, são compostos por ferramentas para atuação autônoma e auto-monitorada, oferecendo recursos para aprendizagem coletiva e individual.

Kenski (2007) apresenta a seguinte definição para os AVs.

> *"são sistemas computacionais disponíveis na internet, destinados ao suporte de atividades mediadas pelas tecnologias de informação e comunicação.Permitem integrar múltiplas mídias, linguagens e recursos, apresentar informações de maneira organizada, desenvolver interações entre pessoas e objetos de conhecimento, elaborar e socializar produções, tendo em vista atingir determinados objetivos"*

. Como exemplos de ambientes virtuais podemos citar o Moodle[1], Teleduc[2], AulaNet[3], entre outros.

Os sistemas computacionais de educação à distância hoje disponibilizados têm como foco centralizar as ações do professor na tarefa de auxiliar no processo de construção-difusão do conhecimento.

Segundo Giraffa (1999), estes softwares oferecem um conjunto de tecnologias de informação e comunicação, que permitem desenvolver as atividades no tempo, espaço e ritmo de cada participante. Agregando formas de disponibilização de conteúdo, acompanhando o progresso do aluno no processo de aprendizagem e permitindo diversas formas de interação entre os usuários.

O uso de AVs no âmbito educacional oferece as seguintes vantagens [Mendonça et al, 2007]:

- melhorar a interação entre o computador e o aluno;

---

1 http://moodle.com

2 http://www.ead.unicamp.br/~teleduc

3 http://aulanet.cinted.ufrgs.br

- a possibilidade de se dar atenção individual ao aluno;
- a possibilidade de o aluno controlar seu próprio ritmo de aprendizagem, assim como a seqüência e o tempo;
- a apresentação dos materiais de estudo de modo criativo, atrativo e integrado, estimulando e motivando a aprendizagem;
- a possibilidade de ser usado para avaliar o aluno.

O fator primordial para a aceitação desse tipo de software é a existência de um relacionamento homem-máquina efetivo e proveitoso. Na EaD, um dos grandes desafios é como fazer a colaboração e cooperação em seus ambientes. A Inteligência Artificial (IA), nesse sentido, é a área que mais tem contribuído com o desenvolvimento de pesquisas e construção de ferramentas no apoio ao processo de ensino-aprendizagem, tentando modelar o desempenho humano para explorar de maneira mais eficiente o conhecimento que se pode adquirir do indivíduo [Rich e Knight, 1993].

De acordo com Giraffa (1999), é crescente o número de pesquisadores na área de IA que têm se preocupado em realizar um trabalho investigativo, devido à qualidade das tecnologias disponibilizadas no mercado.

Foi seguindo essa perspectiva, de crescente busca por meios que ajudem a aprimorar os AVs, que nasceram os Companheiros Virtuais Inteligentes (CVI), que são agentes inteligentes[4] inseridos nos ambientes virtuais com o papel de ajudar o usuário e, conseqüentemente, melhorar a interação entre o homem e computador.

---

[4] Para Jennings (1994), são sistemas computacionais, posicionados em ambientes, que são capazes de agir com autonomia flexível visando atingir os objetivos para os quais foram projetados.

Muitas são as definições sobre CVIs, porém todas descrevem de forma semelhante características e forma de atuação desses agentes inteligentes, como podemos observar no conceito de Hayes-Roth et al (1993) quando diz:

> *"são agentes que executam continuamente três funções: percepção de condições dinâmicas no ambiente; as ações para influenciar o ambiente; e o raciocínio para interpretar percepções, resolver problemas, planejar conclusões e determinar ações."*

Assim como na definição de Maes (1994):

> *"são programas de computador que empregam técnicas de inteligência artificial para prover assistência ativa ao usuário com tarefas baseadas em computadores e adquirem experiência ao auxiliar os usuários em suas tarefas."*

Que reforça o relato de Wooldridge (2002):

> *"é um sistema computacional encapsulado que está situado em algum ambiente e é capaz de ação flexível autônoma neste ambiente, a fim de alcançar seus objetivos de projeto"*

O CVI engloba a questão da afetividade e é dele a atribuição de acompanhar o usuário nos seus estudos (especialmente no âmbito educacional) e na realização de suas tarefas, buscando estabelecer relações de empatia e mantê-lo motivado a persistir apesar das dificuldades no processo [Kampff et al, 2005]. O CVI estabelece vínculo e pretende comprometer, responsabilizar, o usuário com o seu próprio processo de aprendizagem.

Seguindo esse paradigma, hoje, é possível encontrar boas iniciativas que visam a disponibilização de ambientes de educação à distância apoiado por agentes [Geyer et al, 2001].

A necessidade de se integrar companheiros virtuais inteligentes a ambientes virtuais deixou de ser exclusiva da área educacional. Hoje, podemos encontrar agentes inteligentes em diversos domínios de aplicação, como, por exemplo, na saúde, indústria, prestação de serviços, entre outros [Wooldridge, 2002].

Para se ter mais qualidade no produto, é fundamental a existência de métodos, técnicas e ferramentas que guiem a construção do software. No entanto, é notório que ainda há dificuldades no desenvolvimento de companheiros virtuais [Castro et al, 2006], visto que, a pesquisa em busca de métodos e técnicas que facilitem a concepção de CVIs parece ser incipiente. Muitas são as iniciativas de definição e elaboração de companheiros virtuais inteligentes, mas pouca é a preocupação em saber como tais agentes serão concebidos, quais passos são necessários para obtenção do CVI com mais eficiência e eficácia. Sendo assim, faz-se necessária e fundamental uma análise mais profunda das metodologias de desenvolvimento existentes e atualmente utilizadas para suportar a elaboração destes agentes.

Só com uma metodologia[5] de desenvolvimento formalizada, abrangente e bem definida é que se torna possível sistematizar, uniformizar, formalizar e documentar a comunicação entre os envolvidos no processo de construção de um companheiro virtual inteligente; definir o que se quer alcançar; escolher técnicas e tecnologias mais adequadas; dividir competências e responsabilidades necessárias à solução; e, conseqüentemente, obter um produto com o nível de qualidade satisfatório. No caso do desenvolvimento dos

[5] Tartuce (2006) aponta que a metodologia científica trata de método e ciência. Método é o caminho em direção a um objetivo; metodologia é o estudo do método, ou seja, é o corpo de regras e procedimentos estabelecidos para realizar um objetivo.

CVIs, um aspecto importante que deve ser considerado na criação de uma metodologia de desenvolvimento é a interação homem-máquina, bem como a capacidade de relacionamento do usuário, no tocante à cognição e personalidade, já que as ações do agente estão completamente ligadas ao modo de ser do indivíduo que se comunica com o CVI.

A análise realizada no decorrer do trabalho objetiva coletar todos os benefícios advindos das metodologias e ou trabalhos existentes e usados na concepção de CVIs, com o intuito de buscar os principais aspectos positivos de cada uma delas e propor melhorias, reunindo todas essas indicações em uma nova metodologia que será proposta.

Esta metodologia tem como propósito ser usada em qualquer domínio de aplicação. Entende-se por domínio de aplicação um contexto de desenvolvimento, com escopo bem definido para o qual será desenvolvido o CVI . Jackson (1995) define domínio de aplicação da seguinte forma:

> *"Domínio de aplicação é a parte do mundo real na qual o cliente está interessado, com particular relevância para a solução do problema particular. Envolve qualquer coisa ou pessoa que irá interagir com a máquina ou inserir-se no contexto de seu processamento".*

1.2 Objetivos

O desenvolvimento deste trabalho anseia atingir alguns objetivos levantados durante a definição do problema aqui abordado. Os próximos subitens desta seção descrevem tais objetivos.

1.2.1 Objetivo Geral

O objetivo geral da pesquisa é propor uma metodologia para o desenvolvimento de companheiros virtuais inteligentes que possa ser utilizada para qualquer domínio de aplicação.

1.2.2 Objetivos Específicos

Como objetivos específicos têm-se:

- Revisar as metodologias existentes usadas na criação de CVIs;
- Revisar práticas bem sucedidas de metodologias de desenvolvimento de software que possam ser usadas no desenvolvimento de CVIs;
- Analisar perfis de alguns CVIs já existentes;
- Investigar as necessidades previstas na formulação de metodologias para a criação de CVIs;
- Avaliar os efeitos de uma metodologia eficaz na composição de CVIs.

1.3 Metodologia

A elaboração desta dissertação se deu em dois momentos. Inicialmente, foi realizada uma pesquisa bibliográfica acerca de alguns companheiros virtuais inteligentes conhecidos, para entender seus comportamentos e perfis. Em seguida, houve uma coleta bibliográfica relacionada a metodologias de desenvolvimento de software, tradicionais, ágeis e orientadas a agentes usadas na construção de CVIs. Esta atividade teve como foco a busca de boas práticas e características importantes que pudessem ser fundamentais para o propósito do trabalho. Este momento foi basicamente para catalogar e encadear todo o referencial teórico adquirido que serviu de insumo para elaboração da metodologia proposta.

Em seguida, a concepção da metodologia para construção de CVIs se deu de fato, utilizando como base todo material coletado, buscando extrair os principais pontos e idéias positivas no tocante à elaboração de agentes dessa natureza.

1.4 Estrutura do Trabalho

A dissertação é composta por cinco capítulos, estando dividida da seguinte forma:

- **Capítulo 1:** relata a motivação e contexto da pesquisa apresentada no desenvolvimento desta dissertação, além de listar todos os objetivos que se deseja alcançar. Adicionalmente, descreve a metodologia de trabalho utilizada para atingir os objetivos definidos, bem como a forma em que a dissertação está estruturada.

- **Capítulo 2:** apresenta uma fundamentação teórica com relação ao sugimento, conceitos e características dos AVs e CVIs e a integração existente entre eles. Procura-se, também, listar e caracterizar os principais domínios de aplicação em que os companheiros virtuais estão presentes.

- **Capítulo 3:** apresenta uma revisão da literatura acerca de metodologias de desenvolvimento tradicionais e ágeis, bem como metodologias orientadas a agentes usadas na concepção de companheiros virtuais inteligentes e que serviram de objeto de estudo para a definição na metodologia proposta. Além disso, apresenta uma discussão acerca de quais os requisitos necessários que uma metodologia para construção de CVIs deve possuir.

- **Capítulo 4:** apresenta uma metodologia para desenvolvimento de companheiros virtuais inteligentes, que é o objetivo principal deste trabalho, ressaltando, detalhadamente, suas características, boas praticas, fases e respectivas atividades, identificando os papéis envolvidos e os artefatos gerados em cada etapa.

- **Capítulo 5:** diz respeito às conclusões e trabalho futuros, ressaltando uma análise comparativa realizada entre a metodologia proposta e as metodologias orientadas a agentes Prometheus e MaSE, as contribuições, dificuldades e limitações da pesquisa, bem como pontos de extensão para trabalhos futuros.

2 AMBIENTES VIRTUAIS E COMPANHEIROS VIRTUAIS INTELIGENTES

Este capítulo apresenta, inicialmente, na Seção 2.1, uma fundamentação teórica sobre os ambientes virtuais, suas características e funcionalidades. Na Seção 2.2, o ponto focal do capítulo, são apresentados os companheiros virtuais inteligentes, onde serão ressaltadas suas definições, objetivos, comportamentos, pontos positivos, etc. Além disso, serão listados alguns CVIs já existentes, apresentando alguns domínios de aplicações onde eles são comumente utilizados.

2.1 Ambientes Virtuais

Com a globalização, o mundo passou a ter mais necessidade de trocar informações e coletar conhecimentos de diversas formas. Assim sendo, muitos foram os meios de aprendizado que surgiram e/ou se aprimoraram.

Adicionalmente, para Lévy (1993), o advento das nova TIC possibilitou a criação de um novo e amplo espaço de possibilidades para a Educação. Este avanço tecnológico propiciou o surgimento de uma diversidade de ambientes, que fazem uso de diferentes tecnologias como forma de contribuir para o processo de ensino-aprendizagem.

É dentro desse contexto que aparecem os Ambientes Virtuais (AV), criados pela necessidade de se ter um meio de interação mais dinâmico e que pudesse alavancar a Educação à Distância. Schlemmer (2005) define ambientes virtuais de aprendizagem como:

> *"Softwares desenvolvidos para o gerenciamento da aprendizagem. Eles são sistemas que sintetizam a funcionalidade de software para*

> *comunicação mediada por computador e métodos de entrega de material de cursos on-line".*

Outra definição interessante é a dada por Santarosa e Sloczinski (2004):

> *Um ambiente de aprendizagem, suportado pelas tecnologias digitais, pode ser entendido como sendo um ambiente virtual de aprendizagem, ou cenário para a realização de cursos, desde que tenha presente uma proposta educacional identificada com o ambiente e com um paradigma educacional congruente. A realização de cursos, na modalidade à distância, requer estruturas que viabilizem os mesmos, tanto no que diz respeito à acessibilidade e facilidade de uso, quanto às ferramentas interativas e necessárias para propiciar a construção do conhecimento".*

Hoje, os AVs vêm se tornando amplamente usados como ferramenta de ensino/aprendizado, por serem um fácil mecanismo de aquisição de conhecimento.

Entende-se que para se ter uma interação aceitável entre o usuário e computador é necessário que a interface apresente boa usabilidade.

Para Perotto apud Shneiderman (2002), o termo usabilidade, sob o ponto de vista técnico, é a combinação de algumas características orientadas ao usuário:

- Facilidade de aprendizagem;
- Desempenho de alta velocidade de tarefas do usuário;
- Baixa taxa de erros;
- Satisfação subjetiva do usuário;
- Manutenção da atenção do usuário.

Ainda segundo Perotto apud Hix (2002) a usabilidade está diretamente relacionada com a reação do usuário diante da interface, bem como, com a eficiência e eficácia desta. Outro aspecto relevante da usabilidade é a naturalidade da interface com o usuário.

A grande maioria dos trabalhos que propõem uma interação homem-computador sempre objetiva, diretamente ou não, prover ao usuário um alto grau de usabilidade [Perotto apud Hix, 2002].

É notório que não se consegue o desenvolvimento de uma interface de sucesso na primeira tentativa. Um processo de refinamento sucessivo da interface se faz necessário, segundo o ponto de vista de Reis (2000), para avaliar constantemente os resultados a partir das metas que foram definidas no projeto.

Reis (2000) ainda aponta a causa dos problemas de usabilidade que ocorrem, quando diz:

> *"Os pesquisadores mostram que a vasta maioria de problemas de usabilidade vem de uma única causa: a equipe de desenvolvimento não conhece (ou não sabe) qual a peça chave da informação. Se eles tivessem conhecido tais informações mais cedo, teriam projetado o produto para acomodá-lo e o problema de usabilidade nunca teria ocorrido".*

Todas essas considerações acerca de usabilidade são necessárias e importantes no momento da concepção de um AV, visto que o sucesso dessa classe de software é completamente dependente da facilidade de interação entre o usuário e o computador.

Sabe-se que, desde os primórdios da pesquisa em Inteligência Artificial e Educação, a necessidade de inserir mecanismos de apoio à aprendizagem, tais como estratégias e táticas de ensino, baseadas naquelas utilizadas pelos professores em sala de aula, é um dos grandes desafios na pesquisa desta área.

As limitações de hardware e software são fatores que contribuem para que esta tarefa não seja atingida ainda na sua plenitude [Goulart e Giraffa, 2001].

Como características relevantes e que identificam os ambientes virtuais inteligentes, Urretavizcaya (2001) enumera:

> *"1. O conhecimento do domínio está restrito e claramente articulado;*
>
> *2. Possuem conhecimento do usuário que lhes permite dirigir e adaptar a interação;*
>
> *3. A seqüência da interação não esta predeterminada pelo designer instrucional;*
>
> *4. Realizam processos de diagnóstico mais adaptados ao usuário e mais detalhados;*
>
> *5. A comunicação CVI-Usuário melhora, permitindo que o usuário realize perguntas ao companheiro".*

Conforme Gavidia e Andrade apud Jonassen (2003), um ambiente virtual voltado para Educação só é considerado inteligente se passar em três testes:

- O conteúdo do tema ou especialidade deve ser codificado de modo que o sistema possa acessar as informações, fazer inferências ou resolver problemas;
- O sistema deve ser capaz de avaliar a aquisição deste conhecimento pelo usuário;
- As estratégias tutoriais devem ser projetadas para reduzir a discrepância entre o conhecimento do professor e o conhecimento do aluno.

A concepção de AVs no âmbito educacional envolve a integração de um conjunto de recursos, que podem ser classificados de acordo com sua funcionalidade. Jogos, micromundos, simuladores, assistentes, sistemas tutoriais, entre outros, têm sido propostos para softwares educativos. Além disso, os AVs podem incorporar algumas outras classes de ferramentas para facilitar a colaboração e cooperação entre os usuários e mediadores, como aponta Silva et al apud Chaffey (2003):

- Correio eletrônico (e-mail), conferência baseada em texto (síncronas ou assíncronas), vídeo-conferência e outras;
- Software de encontros eletrônicos (do inglês: EMS - Electronic Meetings Software) também chamado de sistema para apoio a decisões de grupo (do inglês: GDSS - Group Decision Support Systems);
- Compartilhamento de documentos e autoria colaborativa;
- Software de gerência de documentos eletrônicos (do inglês: EDMS – Eletronic Document Management Software);
- Sistemas de coordenação de grupos;
- Sistemas gerenciadores do processo de aprendizagem [Silva et al apud Tavares et al, 2003].

Para Rodrigues et al (2005), a construção de ambientes virtuais inteligentes pode ser uma tarefa bastante complexa e lenta, já que existem diversos esforços a serem postos em prática, que vão desde a modelagem do ambiente até a fase de manutenção e testes.

Os ambientes virtuais inteligentes, geralmente, se baseiam em uma arquitetura básica tradicional, composta por quatro componentes, como apresentado por Gavidia e Andrade (2003) na Figura 1.

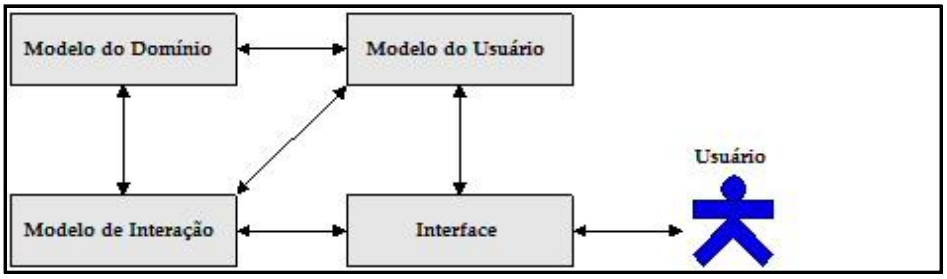

Figura 1 - Arquitetura Básica de um Ambiente Virtual Inteligente [Fonte: Gavidia e Andrade (2003)]

Esta arquitetura é denominada clássica ou função tripartida, se referindo aos três modelos apresentados. Esta proposta trouxe diversos avanços à modelagem de ambientes educacionais, uma vez que propôs a separação do domínio da sua forma de manipulação (utilização), permitindo que estratégias de ensino fossem associadas em função das informações oriundas da modelagem do usuário [Gavidia e Andrade, 2003]. A partir desta arquitetura, surgiram diversas outras mais aprimoradas.

Os AVs oferecem uma forma simples e prática de acesso às informações e promovem um meio onde o aluno adquire uma postura autodidata, que muda completamente a visão de estudo, na qual o professor não participa de forma direta no ensino [Torreão, 2005]. Dessa forma, sem a participação presencial do professor, o estudo passa a se tornar cada vez mais individualizado, e tomando isso como preocupação, é de fundamental importância que esses ambientes possam estimular o estudante a aprender e a interagir com eles, para que o estudo ou qualquer outra forma de interação não caia na monotonia.

Apesar de proporcionar algumas facilidades, o que se vê é que os ambientes virtuais ainda não atendem a todos os requisitos e, contudo, não conseguem atingir o máximo esperado, pois não exploram o quesito de motivação de seus usuários.

De acordo com Hara e Kling (2000), no meio educacional, os conteúdos nos AVs são, simplesmente, disponibilizados de forma impessoal, sem considerar as diversas necessidades dos estudantes, e isso ocasiona o desestímulo para o aprendizado. Sendo assim, o aluno passa a se sentir isolado, acarretando o grande índice de desistência dos cursos à distância.

Dillenbourg (2003) comenta que os problemas das interações homem-máquina são essencialmente um contingente de ordem pedagógica. Ainda segundo Dillenbourg, questões técnicas não são fatores limitadores no estabelecimento de um ambiente colaborativo, uma vez que vários recursos já existentes não apresentam maiores limitações, como por exemplo, a utilização do correio eletrônico e dos chats. O desafio é tornar essas comunicações em um instrumento construtivo para a aprendizagem [Costa e Franco, 2005].

É nesse contexto de deficiências do meio educacional, que os agentes inteligentes despontam como uma solução de melhoramento de tais ambientes, buscando torná-los mais semelhantes e próximos da realidade de cursos presenciais, permitindo um aprendizado mais produtivo, através da disponibilização de um ambiente mais interativo onde o estudante pode ser acompanhado regularmente por um professor [Torreão, 2005]. Observa-se que são notórios os benefícios trazidos ao se utilizar a abordagem de agentes inteligentes com esse propósito.

De acordo com Johnson et al (2000), os agentes inteligentes, quando introduzidos em ambientes educacionais virtuais, são comumente chamados de agentes pedagógicos e evocam diversas características pertinentes ao

comportamento humano que auxiliam na interação e promovem um processo de ensino-aprendizagem mais proveitoso. Essas características podem ser representadas no ambiente de diversas maneiras, como, textos, gráficos, animações, ícones, multimídia ou realidade virtual.

Segundo Elliott et al (1997), agentes pedagógicos são professores mais efetivos quando estes demonstram e compreendem emoções. Os sentimentos do agente, tais como, alegria, surpresa, desapontamento, podem ser expressos em resposta às ações do usuário, no momento da interação entre as partes [Santos et al. 2004]. Esses sentimentos podem ser representados por meio de textos, através da interface gráfica, ou pela combinação de mensagens de texto com um personagem animado [Johnson et al, 2000].

Para Giraffa (1999), quando um agente é utilizado em sistemas que utilizam o paradigma de agentes concebidos para fins educacionais, ele é dito pedagógico. Ainda de acordo com Giraffa, estes companheiros virtuais têm como objetivo apoiar os estudantes no processo de ensino-aprendizagem, interagindo com o usuário e observando suas ações, caracterizando-se como um tutor de conteúdos ou estratégias mais adaptadas ao perfil do usuário.

Como já explicitados anteriormente, podemos citar como exemplos de AVs: o Moodle, o Teleduc, o AulaNet, entre outros. E como exemplos de AVIs podemos destacar o PMK (*Project Management Knowledge Learning Environment*) [Torreão, 2005], ambiente colaborativo, que acoplado ao Victor, ensina gerenciamento de projetos; o ADE (*Advanced Distance Education*) [Johnson et al, 2000], que com o agente Adele auxilia estudantes na área das Ciências Médicas; e o PROPA [Goodman et al, 1997], que com o auxílio da Lucy, ensina habilidades em análise exploratória das atividades de satélites; entre outros.

2.2 Companheiros Virtuais Inteligentes

Na Educação à Distância, existem algumas formas de atuação que o agente pedagógico[6] pode assumir, dentre elas, podemos destacar os papéis de tutor virtual, estudante virtual, ou ainda a roupagem de um personagem que interage explicitamente com o usuário, de modo que, além de agir como um tutor, que atua com estratégias e fornece conteúdos adaptados ao perfil do usuário, pode agir como um companheiro virtual inteligente (CVI) no âmbito da aprendizagem, se assim for definido [Giraffa, 1999].

Os autores procuram definir um companheiro virtual inteligente através de propriedades. Um CVI não necessariamente precisa possuir todas as propriedades para assim ser nomeado, pois cabe aos steakholders que definem o agente incluí-las na aplicação de acordo com as funcionalidades requeridas. Algumas das propriedades mais importantes e que caracterizam um CVI estão listadas e conceituadas na Tabela 1.

Tabela 1 - Propriedades dos CVIs

Propriedades dos CVIs	
Autonomia	É a capacidade que o companheiro virtual tem de agir sozinho, sem a necessidade de orientação de usuários, ou seja, ele possui o controle sobre suas ações no cumprimento de suas metas [Maes, 1994].

[6] De acordo com Wilges (2004), um agente pedagógico pode ser entendido como uma entidade de software que funciona de forma contínua e autônoma em um ambiente de ensino-aprendizagem, capaz de intervir no ambiente de forma flexível e inteligente.

Comunicabilidade	Propriedade que representa a capacidade do CVI de se comunicar com pessoas no ambiente no qual está inserido ou mesmo com outros agentes [Franklin e Graesser, 1996].
Reatividade	É a propriedade que possibilita ao CVI perceber o seu ambiente e atuar sobre ele. É uma propriedade essencial de um agente inteligente. [Nwana, 1996].
Pró-atividade	É por meio dessa propriedade que o companheiro virtual toma iniciativas sobre circunstâncias específicas e não apenas age em resposta ao ambiente. No entanto, para que o agente alcance esta habilidade, é necessário que suas metas estejam claras e bem definidas [Nwana, 1996].
Continuidade	Os agentes precisam estar em processamento contínuo, na forma ativa ou passiva ("sleeping") [Garcindo, 2002].
Aprendizagem	Agentes inteligentes devem ser passíveis de aprendizado, ou seja, devem possuir, no mínimo, a capacidade de alterar seu comportamento baseado em experiências prévias [Franklin e Graesser, 1996].
Inteligência	Esta propriedade é percebida quando o CVI usa sua habilidade de negociação afetiva com ambigüidades e com a capacidade de inferência [Shoham, 1993].
Veracidade	Propriedade que representa a suposição de que um agente não comunicará informações falsas [Garcindo, 2002].

Para Giraffa (1998), os objetivos de um CVI podem ser representados de acordo com seu comportamento – estratégias definidas. Ele ainda aponta que,

os possíveis comportamentos para um agente inserido em um ambiente de ensino são:

- Guia – caracterizado por ser diretivo em suas intervenções e monitorar constantemente o aluno. Possui a propriedade de conduzir o aluno na resolução de problemas em todo o processo de interação;
- Assistente – o agente é menos diretivo e, constantemente, monitora o aluno, fazendo uso de heurísticas sobre resolução do problema naquele domínio, nas intervenções;
- Facilitador – assim como os outros, este CVI monitora o aluno o tempo todo, no entanto não é diretivo. Caracteriza-se por apenas dar dicas sobre a resolução de problemas e intervir apenas quando solicitado.

Um ou vários comportamentos podem ser utilizados pelo tutor, de modo que, com base nas informações passadas pelo usuário e no contexto que se almeja, o foco de atuação pode mudar [Goulart e Giraffa, 2001].

A estrutura interna e funcionamento de um CVI são representados e especificados por sua arquitetura. Dessa forma, para que o desenvolvimento de um CVI seja efetivo e eficaz, é fundamental que sua arquitetura tenha sido concebida com qualidade.

Para Wooldridge e Jennings (1995), as arquiteturas de companheiros virtuais inteligentes podem ser divididas em três áreas:

- **Arquitetura Cognitiva ou Deliberativa** – nesta arquitetura os agentes são vistos como parte de um sistema baseado em conhecimento, onde eles possuem modelos simbólicos explícitos em seus ambientes, de modo que as ações são tomadas via raciocínio lógico. Há duas dificuldades que podem ser encontradas em agentes que possuem

esse tipo arquitetural: em primeiro lugar, traduzir o mundo real em uma descrição simbólica, de forma exata e adequada, é difícil e pode tomar muito tempo; em segundo lugar, há o problema de como representar simbolicamente as informações sobre entidades e processos de forma que os agentes possam raciocinar com essas informações, em tempo hábil.

- **Arquitetura Reativa** – nesta arquitetura não se fez necessária a inclusão de nenhum tipo de modelo do mundo simbólico pré-estabelecido, como também, não faz uso de raciocínio simbólico complexo. A decisão a ser tomada pelo CVI é implementada através de mapeamento do tipo situação-ação.

- **Arquitetura Híbrida** - esta arquitetura representa uma combinação das duas abordagens citadas anteriormente. Nesta arquitetura, também denominada de "arquitetura em camadas", o CVI possui como estrutura dois subsistemas: um deliberativo, que possui o modelo simbólico, e outro reativo, capaz de reagir a eventos que ocorram no ambiente.

Outros autores também especificam as classificações de arquitetura de CVIs de forma semelhante, como a apresentada por Weiss (2000), que se torna muito próxima da proposta por Wooldridge e Jennings, incluindo a arquitetura BDI (agentes com crenças, desejos e intenções) [Gracindo, 2002].

De acordo com Correia (2004), o CVI é composto de um motor de raciocínio e, caso seja assim especificado, pode possuir um personagem animado. Ele é responsável por realizar a interação do sistema com o usuário, tendo suas ações e reações observadas. Na EaD, o CVI também é responsável por apresentar materiais e informações relevantes ao estudante, como por

exemplo, o andamento dos estudos por parte do aluno, textos explicativos de alerta e motivação, questionamentos, entre outros.

Ainda segundo Correia, todas as atividades executadas pelo estudante são monitoradas e acompanhadas pelo CVI, de modo que todo o conteúdo estudado é registrado, promovendo um auxílio constante em situações críticas, dando dicas relevantes no processo de aprendizagem e transmitindo conhecimento. Esse acompanhamento regular dá aos usuários do sistema confiabilidade e segurança no que se refere à interação e às informações que são repassadas, deixando a idéia de ter um amigo (companheiro) que lhes fornecerá ajuda [Correia, 2004].

Torreão (2005) relata que um companheiro virtual inserido em um ambiente virtual:

> *"... pode também incentivar o estudante a articular e a refletir sobre as suas ações passadas, e a discutir suas intenções futuras e suas conseqüências, melhorando assim o aprendizado".*

Assunção et al apud Rissoli (2008) diz que os companheiros virtuais inseridos em ambientes educacionais têm a capacidade de ensinar e aprender, visando adequar estratégias de ensino às diversas necessidades de aprendizagem do estudante. Esta adequação é possível através da combinação mais coerente e dinâmica das informações relacionadas ao estudante, ao conteúdo ou domínio, bem como dos aspectos pedagógicos envolvidos na efetivação do ensino-aprendizagem eficiente.

Para aumentar a motivação dos usuários, que fazem uso de ambientes virtuais para aprender, sugere-se uma interação face-a-face com um personagem animado. Personagens animados em ambientes de aprendizado dão a impressão de serem seres vivos e passam ao usuário uma maior

credibilidade por terem atitudes bem próximas de um tutor humano [Torreão, 2005].

Jaques et al (2001) diz que agentes pedagógicos animados são aqueles que fazem uso de recursos multimídia para apresentar ao usuário um personagem animado que interaja com ele. Além disso, o autor relata que estes personagens usam de vários artifícios para manter a atenção e o interesse do usuário, tais como gestos, olhares e movimentos para aumentar a interação com o estudante, por exemplo, um olhar pode indicar que o estudante deve ter mais atenção, um gesto pode incentivar o estudante a persistir seus estudos, uma frase pode fazer com que o usuário sinta satisfação e motivação em executar as ações perante o sistema.

No entanto, há questões que devem ser observadas com relação aos recursos que o agente animado usa na interação no momento do desenvolvimento do CVI, como relata Johnson et al (2000), quando diz:

> *"Os benefícios que um personagem animado pode trazer são inúmeros, portanto vale lembrar que os projetistas devem ter o cuidado de não promover a distração do estudante durante qualquer animação".*

O companheiro virtual inteligente se configura como o novo paradigma em ambientes virtuais, principalmente, nos de ensino, pois eles podem demonstrar tarefas complexas, utilizar de locomoção e gestos para direcionar a atenção dos usuários para aspectos relevantes das atividades e demonstrar respostas emotivas durante a interação [Johnson et al, 2000].

Wilges (2004) acredita que a concepção de um agente pedagógico animado, que desempenha o papel de assistente em um ambiente educacional, contribui para a melhoria do processo de ensino-aprendizagem de seus usuários. Uma das contribuições seria para que o ensino-aprendizagem não se

baseasse somente no convencional professor/aluno em sala de aula, mas sim através de uma interação à distância entre aluno/máquina.

Assim o aluno é estimulado a buscar uma nova forma de conhecimento/informação, sem, todavia, aumentar o nível de dificuldade e empecilhos neste processo.

A atuação de um agente animado em um ambiente virtual deve ser sensata e ponderada, para que este não se torne por vezes um incômodo para o usuário. É com relação a este aspecto que Wilges (2004) enfatiza:

> *"... a principal função dos agentes animados nesses ambientes é a de facilitar o aprendizado, através de uma interação que possibilite demonstrar aos alunos tarefas e soluções, sem, entretanto, incomodá-lo em momentos desnecessários".*

O consenso entre pesquisadores na área é que os CVIs são a promessa de ampliação da comunicação em ambientes virtuais, incrementando habilidades, prendendo a atenção e motivando o usuário.

Um CVI inserido em um ambiente virtual facilita a cooperação entre usuário e sistema, podendo adaptar o conteúdo e forma de apresentação, de acordo com o desempenho do usuário. Essa é a principal característica, vista como cenário ideal de aprendizagem que proporciona uma interação individualizada [Chou et al, 2003].

Segundo Bocca et al (2003), quando o agente está inserido em um AV, o aluno pode aprender e praticar habilidades no mundo virtual. Através dos agentes, o computador pode interagir com os alunos de diversas formas, através de iniciativa mista, diálogo tutorial (no papel de professor) ou companheiro aprendiz. A comunicação pode ser realizada de forma verbal ou

não verbal. Os gestos podem ser utilizados como artifício facilitador de comunicação.

Bocca et al (2003) ainda aponta que, na comunicação não verbal, o agente animado pode fazer uso de contemplação, afirmação com a cabeça e expressões faciais que podem representar um feedback nas declarações dos alunos, sem a necessidade de interromper o raciocínio do aluno, com comunicação verbal, durante a realização de uma atividade. Enfim, todo esse ferramental não verbal são componentes naturais do diálogo humano que, se aplicados ao agente, surtirão efeitos benéficos no processo de interação homem-computador.

Tornar o comportamento do CVI o mais próximo possível do humano não é uma tarefa simples, porém necessária para uma comunicação mais eficiente pelas vantagens advindas dessa atividade, como relatado por Bocca et al (2003), quando destaca:

> *"O agente pedagógico animado apresenta as vantagens de aumentar a comunicação entre alunos e computadores, e incrementa a habilidade do computador para engajar e motivar o aluno. Estes agentes pedagógicos compartilham aspectos em comum com agentes sintéticos desenvolvidos por aplicações de entretenimento. Eles devem dar ao usuário a impressão de serem naturais e confiáveis. Eles tornam a experiência de ensinar (aprender) mais agradável".*

Como se vê, a utilização de um companheiro virtual inteligente em um ambiente de educação a distância possui diversas vantagens, na medida em que possibilita que os estudantes passem a ser tratados de maneira personalizada, através do acompanhamento do estudante no decorrer dos estudos, na tentativa de motivá-los e fazer com que a experiência seja atrativa [Canuto, 2005].

Apesar da literatura especializada abordar massivamente o sucesso na integração de agentes inteligentes em ambientes virtuais de aprendizagem, percebe-se que este feito não se restringe ao meio educacional. Hoje, é cada vez mais comum encontrarmos CVIs em outros domínios de aplicação, como por exemplo, em sites de e-commerce, onde ajudam o usuário em suas escolhas, opinam sobre compras, motivam o usuário a realizar mais compras, sugerem produtos promocionais; de atendimento ao cliente, onde tiram dúvidas e fornecem informações aos usuários; em aplicações médicas, monitorando pacientes e planos de saúde; em aplicações de entretenimento, especialmente em jogos; entre outros.

2.3 Exemplos de Companheiros Virtuais Inteligentes

Esta seção apresenta, de forma mais detalhada, três companheiros virtuais inteligentes: Adele, Lucy e Victor. Estes agentes foram concebidos em ambientes virtuais de aprendizagem para auxiliar usuários, na tentativa de facilitar o acesso ao conhecimento/informação de forma mais interativa e que fizeram parte do foco no estudo de agentes nesta dissertação.

2.3.1 Adele

Advanced Distance Education (ADE) é um projeto que desenvolve ferramentas para a criação de cursos flexíveis baseados na web que incorporam a Inteligência Artificial. O ADE pertence à equipe do Center of Advanced Research in Technology for Education (CARTE) em USC (Universidade do Sul da Califórnia)/ISI (Instituto de Ciências da Informação).

A equipe do projeto tem desenvolvido simulações que caracterizam agentes pedagógicos - software com personalidade - para ajudar estudantes a trabalhar através dos materiais do curso. Um dos agentes desenvolvidos por eles é nomeado ADELE (*Agent For Distributed Learning Environments*).

ADELE realiza interação com os estudantes e segue sua aprendizagem, enquanto trabalham com os materiais do curso e os exercícios da simulação. ADELE consiste de um agente pedagógico e um personagem animado em 2D e atua como um companheiro de aprendizado, monitorando o estudante constantemente [Johnson et al, 2000].

Os materiais disponibilizados por ADELE são adaptados em consonância com a necessidade de cada aluno. Adicionalmente, o agente fornece sugestões e guia ações do estudante, avaliando o desempenho do mesmo.

O CVI ADELE foi concebido para auxiliar estudantes na área das Ciências Médicas, em duas especialidades: diagnóstico preventivo e trauma. Como o propósito da maioria dos agentes, ele analisa as ações do estudante e dá dicas, explica o raciocínio das ações recomendadas, sugere referências importantes e realiza intervenções, caso o aluno cometa algum erro sério [Torreão, 2005]. A Figura 2 apresenta o agente ADELE interagindo com um aprendiz.

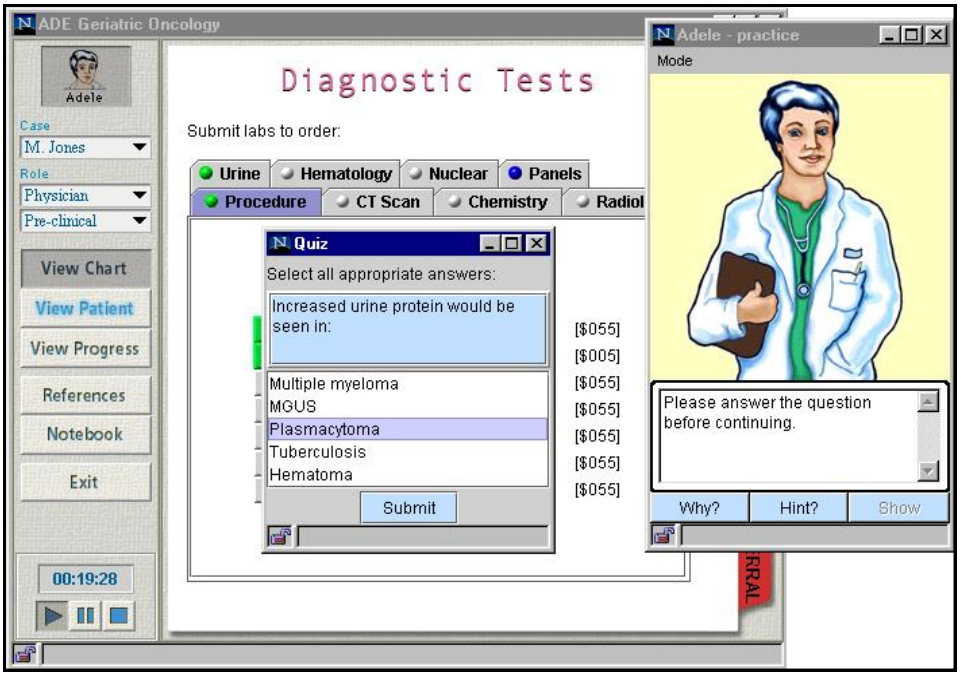

Figura 2 - Adele interagindo com estudante [Fonte: Torreão (2005)]

O ambiente cria para seu usuário simulações de diagnósticos que apresentam casos reais que devem ser analisados. Essas simulações incluem a história do paciente, os resultados dos exames, os testes de laboratório, os raios X e outros métodos de diagnóstico por imagem.

Conforme Johnson et al (1998), o estudante pratica as habilidades diagnósticas examinando e questionando o "paciente virtual" e analisando os dados clínicos. O respectivo autor comenta que ADELE atua no processo de aquisição de conhecimento fornecendo o gabarito e uma revisão do progresso do estudante.

As simulações de trauma procuram ajudar estudantes a desenvolver suas habilidades de resposta a emergências e procedimentos para tratar vítimas de traumas. Tais simulações são projetadas para preparar os discentes para situações complexas, já que médicos da emergência devem responder rapidamente ao desenvolvimento de complicações de pacientes, como por exemplo, a situação em que o paciente desenvolve problemas de respiração ou tem um enfarte no momento do atendimento [Johnson et al, 1998].

Em uma emergência de trauma, os médicos precisam atuar em conjunto, pois executam uma atividade colaborativa com outras pessoas da unidade, cada qual com um papel específico. As simulações de trauma permitem que os estudantes trabalhem junto ao caso, adicionando esta dimensão colaborativa à sua experiência de aprendizagem. O papel selecionado por cada aluno possui tarefas restritas que devem ser executadas. O agente ADELE inclui a noção de "situações". Cada situação é representada por uma descrição de alto nível de um caso que será apresentado ao estudante, juntamente com uma descrição de passos a serem executados para a realização das atividades. Para Johnson et al (1998), esta representação permite que ADELE reconheça características únicas

de cada situação para reagir apropriadamente, com as sugestões e dicas corretas para cada ação que o estudante deve tomar.

2.3.2 Lucy

O projeto de desenvolvimento de LuCy surgiu da necessidade da presença de características dos sistemas tutores inteligentes e de sistemas de aprendizagem colaborativa em um único ambiente.

O CVI LuCy foi desenvolvido para ser integrado a um ambiente já existente, o PROPA, da MITRE Corporation. O domínio de conhecimento em que ele atua é o ensino de habilidades em análise exploratória das atividades de satélites, envolvendo habilidades como formulação de hipóteses, busca por evidências e avaliação da veracidade e significado dessas evidências. A análise exploratória é usada, por analistas de atividades de satélites, para interpretar atividades de satélites baseados em informações incompletas, incertas ou inconsistentes [Goodman et al, 1997].

Para Petry (2005), LuCy foi concebido para dar total apoio ao usuário do ambiente, durante a execução de atividades, quando diz:

> *"o CVI LuCy é um par interativo que responde questões e oferece sugestões da mesma forma que um estudante colaborador real o faria. A vantagem de LuCy comparando com um par real é sua total disponibilidade para com o estudante humano e sua capacidade de apoiá-lo em dúvidas particulares. LuCy incentiva o estudante a justificar suas próprias decisões, considerar e articular ações alternativas, e avaliar suas sugestões".*

Na Figura 3 é apresentada uma situação de interação com o companheiro virtual LuCy.

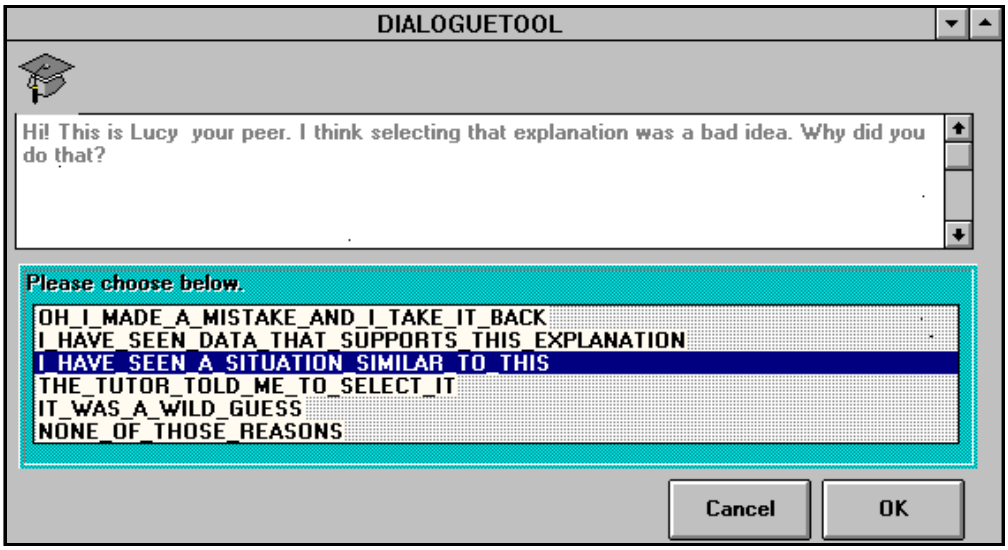

Figura 3 - Interação com o agente LuCy [Fonte: Goodman et al (1997)]

O desenvolvimento de LuCy é baseado em uma arquitetura composta de um parser para tratar as sentenças dos estudantes, o modelo do domínio do conhecimento, o histórico de diálogos, seletor e o gerador de respostas que compõem o modelo do estudante [Petry, 2005].

A interação do usuário com LuCy se dá de acordo com os seguinte passos: Primeiramente, a sentença do estudante é analisada pelo parser para determinar o que o estudante quer. Uma vez determinado o significado da sentença do estudante, ela é armazenada no histórico de diálogos [Goodman et al, 1997]. O próximo passo é selecionar qual o tipo de resposta dar ao aluno. LuCy pode dar uma reposta correta ou uma resposta errada, dependendo do conhecimento que o estudante tem do tema particular em estudo. Por último, a resposta é gerada e enviada ao aluno de acordo com o ritmo de interação com o CVI.

O propósito de criação do projeto de LuCy é meramente promover um diálogo efetivo entres os alunos e o PROPA. O diálogo promovido por LuCy

encoraja o estudante a refletir sobre seu pensamento e a avaliar ações passadas [Petry, 2005].

2.3.3 Victor

O *Project Management Knowledge Learning Environment* (PMK) é um ambiente de ensino que tem como domínio a área de Gerenciamento de Projetos utilizando como literatura o PMBOK® Guide Edição 2000 e outras referências da área. O Victor (*Virtual Intelligent Companion for TutOring and Reflection*) é um CVI que é acoplado ao sistema no intuito de tratar de maneira personalizada cada estudante durante o processo de aprendizado [Canuto, 2005].

Segundo Torreão (2005), o principal intuito do desenvolvimento do Victor é promover uma melhor qualidade no tratamento dos conteúdos através do acompanhamento dos usuários de forma individualizada, sempre se preocupando com o aspecto motivacional dos mesmos, propondo uma visão mais prática e interativa de Gerenciamento de Projetos.

O ambiente também oferece uma preparação para a certificação PMP (*Project Management Professional*). Para isso, ele faz uso de recursos didáticos diversos, como questionários, perguntas freqüentes, exercícios, biblioteca, glossário, lista de figuras e links interessantes [Torreão, 2005].

É disponibilizada, pelo ambiente, uma vasta quantidade de exercícios, de vários tipos (de múltipla escolha, de preenchimento de lacunas e de montagem de diagramas), com dicas de resolução para o modo de estudo. Já no modo simulado, o estudante passa por uma simulação real da prova para obtenção da certificação PMP.

Torreão (2005) apresenta algumas informações relacionadas à interação com o ambiente quando diz:

> *"... são diversos os recursos utilizados pelo PMK para que o estudante possa adquirir conhecimento e interagir com o ambiente, podendo fazer anotações durante todo o aprendizado, visualizando o histórico do conteúdo já estudado e dos simulados realizados, podendo obter informações relativas ao seu desempenho, como taxa de acertos nas questões, e tendo a possibilidade de acessar a qualquer momento outras opções de aprendizado, como por exemplo, glossário, perguntas freqüentes, biblioteca, ajuda, etc".*

O CVI Victor foi concebido no PMK com o propósito de solucionar muitos dos problemas existentes e conhecidos nos ambientes virtuais de aprendizado, dentre eles a questão da falta de motivação que os alunos sentem pela escassez de recursos interativos em muitos dos ambientes existentes, o que acaba acarretando em desistência dos cursos.

De acordo com Correia (2004), Victor é representado por um personagem animado, que interage com o estudante, procurando mantê-lo motivado em seu processo de aprendizado. O respectivo agente, faz uso de imagens e de mensagens de texto, para gerar *feedback* imediato para as ações executadas pelo aprendiz. Adicionalmente, segundo Torreão (2005), ele fornece dicas, sugestões e elogios, de acordo com a dedicação do usuário para com as atividades de estudo.

Tomando como base a informação do parágrafo anterior, é possível considerar o Victor como um agente de sugestão, pois tem a intenção de reforçar o aprendizado do estudante por meio de frases sugestivas.

Victor também faz uso de linguagem natural para responder algumas questões relacionadas ao PMK, atuando assim, como um *chatterbot* [Leitão, 2003].

Diversas táticas pedagógicas são utilizadas por este agente para fornecer resposta imediata. Elogios são dispensados ao estudante, pelo seu esforço, quando as tarefas são feitas corretamente, como também, os erros nos exercícios feitos são apontados quando ocorrem. Conforme Torreão (2005), Victor está sempre motivando o estudante a ser perseverante no estudo.

Os sentimentos do Victor são expressos em resposta às ações do estudante e exibidos em forma de animações. Os movimentos das animações foram projetados de acordo com as ações definidas para o Victor e de expressões faciais definidas para cada ação [Torreão, 2005].

A Figura 4 apresenta momentos de interação de usuário com o Victor. É possível perceber algumas animações criadas para serem expressas em resposta às ações executadas pelo usuário.

Figura 4 - Algumas animações do Victor [Fonte: Torreão (2005)]

Segundo Torreão (2005), a construção do VICTOR se deu em seis etapas: Identificação do Problema, Elicitação de Conceitos Relevantes do Domínio, Conceituação das Tarefas Pedagógicas, Construção da Arquitetura do CVA, Implementação do CVA e Avaliação e Refinamento do CVA.

Conforme Torreão (2005), Victor proporcionou um aprendizado mais individualizado e trouxe facilidade no processo de ensino-aprendizagem, motivando os aprendizes e os auxiliando sempre que preciso.

3 METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

Neste capítulo serão apresentadas algumas metodologias que comumente são usadas no desenvolvimento de software, que serviram como objeto de estudo para a definição da nova metodologia. Primeiramente, na Seção 3.1, é dada uma visão geral sobre o que são metodologias e quais são os benefícios advindos da utilização destas no desenvolvimento de software. A Seção 3.2 trata de metodologias tradicionais, abordando suas principais características e apresentando sua maior referência, o RUP. Em seguida, na Seção 3.3, são caracterizadas as metodologias ágeis, ressaltando suas principais vantagens e apresentando o Scrum e XP para um melhor entendimento. Na Seção 3.4, são apresentadas duas metodologias de desenvolvimento orientadas a agentes: a Prometheus e a MaSE. Por fim, na Seção 3.5, é apresentada um discussão sob o ponto de vista do autor sobre as metodologias estudadas, ressaltando um conjunto de requisitos gerais que uma metodologia para construção de CVI deve possuir.

3.1 Definição e Benefícios

O termo metodologia não possui uma definição amplamente aceita, apesar de ser um conceito bastante utilizado. Para Yourdon (1995), metodologia pode ser entendida como um conjunto de passos e procedimentos a serem seguidos para o alcance do objetivo, neste caso, a construção eficiente e eficaz de um software.

Elaborar softwares sob uma perspectiva menos formal de desenvolvimento tornou-se incipiente, de acordo com experiências iniciais realizadas. Essa conclusão se deu a partir da percepção de que projetos mais

complexos acabavam com a qualidade comprometida, alto custo e prazos apertados [Yourdon, 1995].

Antigamente, não havia a preocupação de se elaborar meios que pudessem controlar e suportar o desenvolvimento de software, como relatado por Santos (2004) na sentença a seguir:

> *"Nos anos 70, quando os gerentes de TI não tinham idéia de como desenvolver um software com qualidade, usando uma metodologia específica e também com uso de técnicas de engenharia de software, a idéia era de produzir softwares que fossem apenas eficientes. Porém, com o passar do tempo, essa visão ficou ultrapassada e, hoje em dia, fica claro que o desenvolvimento de software tem de gerar um produto que por si só seja eficaz, em tempo hábil, e com orçamento previsível".*

A ausência no uso de regras estabelecidas na engenharia de software durante o desenvolvimento pode gerar prejuízos para as organizações.

Ainda para Santos (2004), muitos são os projetos de desenvolvimentos de software que são iniciados e não concluídos. Além disso, existem aqueles que, apesar de terminados, consomem prazos e orçamentos superiores ao definido no começo do projeto. Observa-se, também, que muitos softwares são desenvolvidos com um baixo nível de qualidade. Todos esses fatores evidenciam a necessidade de utilização de uma metodologia de desenvolvimento de software que possa ajudar a agregar qualidade ao produto final em todas as fases desse processo complexo.

Segundo Avison e Fitzgerald (2000) podemos entender como metodologia um conjunto de fases, regras, procedimentos, técnicas, ferramentas, documentação, gerenciamento e treinamento usado na concepção

de sistemas em que o envolvimento do usuário em todas as fases são fundamentais para o sucesso.

Sommerville (2000) define o conceito de metodologia da seguinte forma:

> *"Metodologia de Desenvolvimento é um conjunto de práticas recomendadas para o Desenvolvimento de Softwares, sendo que essas práticas, geralmente, passam por fases ou passos, que são subdivisões do processo para ordená-lo e melhor gerenciá-lo".*

São vários os benefícios advindos da utilização correta de uma metodologia no processo de desenvolvimento de um software. Dentre muitos, podemos destacar os benefícios abaixo:

- **Aumento da qualidade** – A partir do momento que os desenvolvedores do software possuem instrumentos e métodos de levantamento das necessidades dos usuários, torna-se mais fácil construir sistemas com melhor qualidade, bem estruturados e que atendam aos anseios de todos.

- **Independência de indivíduos** - Seguindo corretamente o que preconiza o processo, os softwares ficam bem estruturados e com documentação detalhada, padronizada e organizada. Deste modo, o conhecimento deixa de estar amarrado e exclusivo a um determinado indivíduo que faz parte do projeto, solucionando o problema do caos deixado com a evasão de desenvolvedores.

- **Facilidade de manutenção** – Os motivos que justificam o benefício citado acima, também são adequados para a manutenção dos sistemas, que fica facilitada pela existência de artefatos bem documentados e estruturado.

- **Aumento da produtividade** – A produtividade é naturalmente aumentada, visto que os sistemas bem especificados, possivelmente, são bem projetados e desenvolvidos e, desde modo, possuem mais partes reutilizáveis e se gasta menos tempo com testes para atender ao usuário final.

É com base na importância destacada nos parágrafos acima que surgiu a necessidade de se definir metodologias que pudessem ser usadas para guiar o desenvolvimento de software, na busca de produtos cada vez mais qualificados.

3.2 Metodologias Tradicionais

A divisão em etapas e/ou fases é a principal característica das metodologias consideradas tradicionais, muitas vezes chamadas de "pesadas". As etapas concebidas nesse tipo de metodologia sempre são muito bem definidas e possuem diversas atividades, como Análise, Modelagem, Desenvolvimento e Testes.

O término de uma fase e o início da outra são sempre bem delimitados, visto que a conclusão de uma etapa sempre gera um marco no projeto, que na maioria das vezes é caracterizado pela geração de um artefato (que servirá como entrada para as fases subseqüentes). Documento, protótipo ou versão do sistema são exemplos de artefatos que estabelecem marcos em um desenvolvimento.

O foco principal das metodologias tradicionais é a previsibilidade dos requisitos do sistema, que traz a grande vantagem de tornar os projetos completamente planejados, mantendo sempre uma linha, caracterizando o processo como bastante rigoroso.

De acordo com Pressman (2001), o modelo em cascata, também chamado de Clássico ou Seqüencial, foi o primeiro ciclo de vida de desenvolvimento de

software publicado como tal e, desde então, passou a ser comumente utilizado. É o modelo predominante nas metodologias ditas pesadas.

O modelo em cascata obedece a uma seqüência que deve ser seguida, pelos papéis envolvidos no projeto, de uma etapa a outra. Cada etapa que compõe o processo possui uma documentação padrão associada ao seu término, que, por sua vez, necessita ser aprovada para que a etapa imediatamente posterior esteja apta a ser iniciada [Pressman, 2001].

O controle do projeto é fundamental para a garantia de seu sucesso, todavia a dependência existente entre as etapas do modelo Clássico não facilita a realização desta atividade, visto que para cada alteração realizada em determinado ponto do desenvolvimento do projeto, é necessário uma volta ao início do mesmo para efetivar a alteração [Pressman, 2002]. Isso passa a ser um ponto bastante crítico, uma vez que alterações são muito comuns de ocorrerem durante o desenvolvimento de um projeto.

Autores indicam que o modelo em cascata é ideal para projetos em que os requisitos são bem compreendidos. A seguir, na Figura 5, temos uma ilustração do modelo em cascata [Soares, 2006].

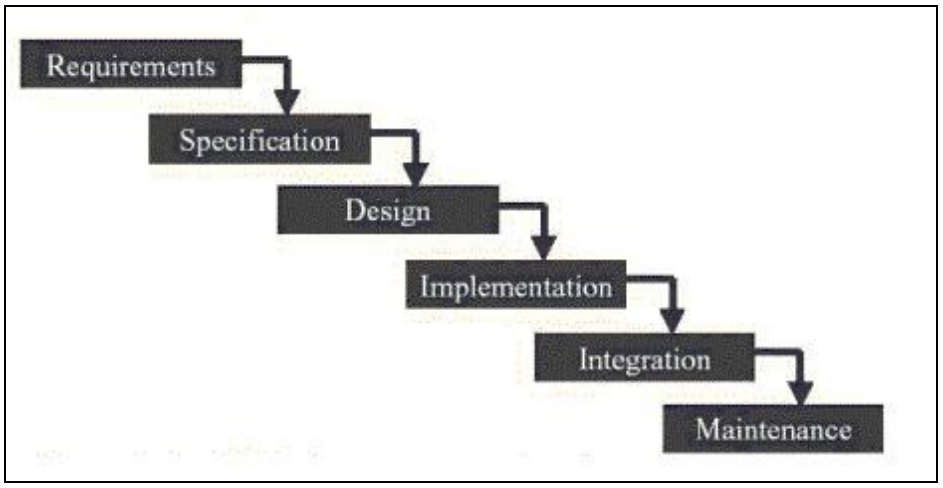

Figura 5 – Modelo de Desenvolvimento de Software em Cascata. [Fonte: Soares (2006)].

Conforme Soares (2006), com a utilização do modelo clássico, pesquisadores e desenvolvedores identificaram e alertaram empresas sobre problemas causados pela adoção dessa visão seqüencial de atividades. Mesmo assim, o modelo clássico foi bastante utilizado.

Dentre algumas críticas que contemplam tais problemas detectados, podemos citar Brooks (1987), quando relata em seu artigo, denominado "*No Silver Bullet: Essence and Accidents of Software Engineering*", que é impossível especificar por completo um software antes do início da fase de implementação. De modo semelhante, Gilb (1999) diz que o desenvolvimento com o modelo clássico apresenta muito risco e menos possibilidade de sucesso.

Segundo pesquisa realizada pelo *Standish Group* em 1995 [Standish Group apud Soares, 2006], tendo como amostra 8380 projetos que utilizaram o modelo cascata, revelou que apenas 16,2% dos projetos foram entregues corretamente, respeitando prazos e custos definidos, bem como todas as funcionalidades especificadas.

Outros dados da mesma pesquisa descrevem que cerca de 31% dos projetos foram cancelados antes do seu final e 52,7% foram entregues em sua completude, porém com prazos estendidos, maiores custos ou apresentando menos funcionalidades do que especificado.

Além disso, também foi constatado que a média de atraso, dos projetos que não foram finalizados de acordo com os prazos e custos especificados, foi de 222%, já a média de custo foi de 189% a mais do que o previsto. A pesquisa também relata que de todos os projetos que foram entregues com o prazo excedido e com o maior custo, apenas 61%, em média, das funcionalidades

especificadas no início do projeto foram incluídas [Standish Group apud Soares, 2006].

Foi percebido também no relato de Standish Group apud Soares (2006) que, mesmo os projetos, cuja entrega é feita respeitando os limites de prazo e custo, possuem qualidade questionável, pois, provavelmente, foram feitos com muita pressão sobre os desenvolvedores, o que pode aumentar consideravelmente o número de erros na fase de testes.

Segundo a pesquisa, as principais razões destas falhas estavam relacionadas com a utilização do modelo Clássico no desenvolvimento de software.

Abaixo, temos algumas vantagens advindas da utilização do modelo em cascata no desenvolvimento de software [Martins, 2003]:

- Estruturação no processo de desenvolvimento. As fases são ordenadas seqüencialmente, onde uma só tem início com o término da anterior;
- As fases que compõem o modelo são fundamentais na produção do software e a ordenação delas é correta;
- A abordagem, hoje, é considerada como norma e, com certeza, persistirá como tal por um bom tempo.

Já como desvantagens percebidas com a utilização do modelo, Martins (2003) destaca:

- Impossibilidade de atualizar e/ou redefinir fases anteriores, bem como inexistência de feedback entre as fases;
- Não suporta modificações nos requisitos;

- A manutenção não é prevista no modelo;
- Não permite a reutilização;
- É excessivamente sincronizado;
- Caso haja algum atraso em uma das fases, todo o processo é afetado;
- O software surge tardiamente;
- A estrutura do modelo direciona a uma forte divisão do trabalho (analistas de requisitos, arquitetos, programadores, analistas de qualidade, programadores de manutenção);
- Ninguém, além do gerente de projeto, consegue ter uma visão global do problema;
- Os papéis envolvidos no projeto não se responsabilizam por problemas ocorridos.

3.2.1 RUP (Rational Unified Process)

Recentemente, mais precisamente em 1998, surgiu uma metodologia denominada *Rational Unified Process* (RUP). É dentro de uma estrutura de desenvolvimento que esse processo de engenharia de software tenta organizar atribuições de tarefas e responsabilidades existentes na elaboração de um software. O RUP, assim como as demais metodologias existentes, possui como propósito – meta - principal garantir a construção do software com qualidade satisfatória, que permita atender as necessidades de seus usuários, levando em consideração um cronograma e orçamento previsível. Hoje, é possível considerar o RUP como a maior referência das metodologias ditas pesadas [Jacobson et al, 1998].

Para Souza (2003), o RUP faz uso de técnicas com o objetivo de aumentar a qualidade dos softwares das empresas de desenvolvimento que fazem uso deste.

Segundo Jacobson et al (1998), outra característica peculiar e que o RUP procura promover é o aumento da produtividade da equipe, através da disponibilização e acesso fácil a bases de conhecimento, para todos os membros, com diretrizes e modelos para atividades de desenvolvimento consideradas críticas. Com o acesso à mesma base de conhecimento, é possível assegurar que todas as pessoas da equipe compartilham de processos únicos e uma linguagem comum, permitindo uma mesma visão do software a ser desenvolvido e unificando os direcionamentos a serem tomados para a elaboração deste.

O RUP foi concebido para ser configurável, pois já se sabe que, na elaboração de um software, apenas um processo não é satisfatório. Ele é adaptável tanto para grandes organizações, quanto para pequenas equipes [Loi, 2007].

A arquitetura do RUP é clara e de fácil entendimento, segundo Loi, podendo ser adaptada para acomodar situações distintas, dependendo da necessidade da organização.

O RUP possui como base o conceito de "melhores práticas", que são regras preconizadas, executadas em todo desenvolvimento do projeto, que buscam redução de risco no projeto, bem como tornar a atividade de construção do software mais eficiente.

As práticas definidas no RUP são as seguintes [RUP, 2001]:

- **Desenvolvimento de software iterativo** – desenvolvimento de ciclos, que facilita dentre outras, a identificação e modificação de requisitos,

integração progressiva do software e melhora na identificação de riscos.

- **Gerenciamento de requisitos** – proporciona uma maneira prática de produzir, organizar, comunicar os requisitos de um projeto.

- **Uso de arquitetura baseada em componentes** – o desenvolvimento é focado na modularidade, por meio da utilização de componentes. Esse direcionamento permite que se crie um software flexível, adaptável, intuitivamente entendível e reutilizável [Ambler, 2005].

- **Modelagem visual de software** – o RUP preconiza a modelagem para facilitar o entendimento da concepção, dimensionamento e complexidade do sistema, bem como para ajudar na identificação e solução de problemas.

- **Verificação contínua da qualidade de software** – a qualidade é atacada tanto com relação ao produto que está sendo desenvolvido, como no processo que guia o projeto de desenvolvimento.

- **Controle de mudanças do software** – como estratégia para garantia da qualidade, é necessário que as mudanças sejam controladas, já que estas ocorrem com freqüência, pois a metodologia possui como prática o desenvolvimento iterativo [Ambler, 2005].

Na Figura 6, temos a ilustração que representa o RUP [RUP, 2003].

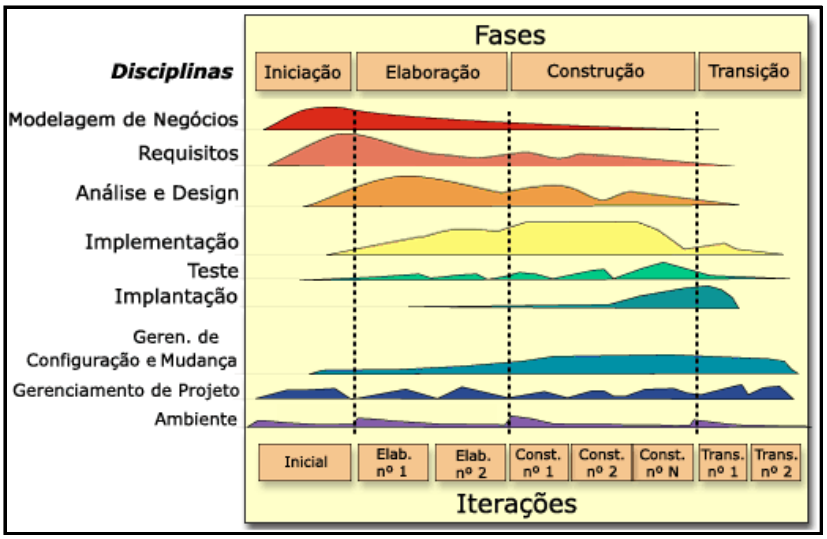

Figura 6 - Visão Geral do RUP [Fonte: RUP (2003)]

A organização do RUP pode ser entendida a partir de duas dimensões:

1. Na horizontal está a dimensão que representa os aspectos relacionados ao fluxo de tempo do processo de desenvolvimento; indica as fases, que por sua vez se dividem em iterações.

2. Na vertical está a dimensão que representa as disciplinas relacionadas ao processo, os fluxos de atividades; é a visão estática, que agrupa atividades e conceitos por assunto.

A conclusão das fases que compõem a visão horizontal se dá com a liberação de um *milestone*. Entende-se por *milestone*, uma etapa, dentro da fase, onde metas são atingidas ou decisões críticas são realizadas. As fases que compõem o RUP são as seguintes [RUP, 2003]:

- **Iniciação**

É a etapa de concepção inicial do sistema. Onde é realizada uma discussão sobre o problema, sobre o escopo e delimitações do projeto, suas fronteiras, determinando os principais casos de uso que vão compor o sistema.

A precisão de elaboração dos casos de uso deve ser necessária para se definir estimativas de prazos e custos. É nesta etapa também que são definidos os atores que irão interagir com o sistema, bem como os recursos que são necessários e os riscos que o projeto pode sofrer.

- **Elaboração**

O objetivo primordial desta fase é fazer uma análise mais refinada do domínio do problema, definir uma arquitetura robusta, elaborar um plano de projeto para o sistema a ser criado e eliminar os elementos de maior risco no projeto. É importante que esses elementos estejam estáveis o suficiente para que se possa prever com precisão os custos e prazos para a conclusão do desenvolvimento.

A fase de Elaboração é considerada a mais importante de todo o RUP, visto que é ao final dela que a engenharia é considerada completa e os custos para modificação do sistema aumentam ao passo que o projeto vai sendo desenvolvido. Após esta fase, o projeto passa ser considerada uma operação de alto custo e alto risco.

- **Construção**

Esta fase diz respeito à modelagem e implementação do sistema em si. É onde efetivamente se tem uma primeira versão do software, de modo que são desenvolvidos todos os componentes do sistema e características não resolvidas nas fases anteriores, realizando testes e integração.

- **Transição**

É nesta fase que é realizada a implantação do sistema para os usuários finais, pois o mesmo já está concluído. É evidente que, ajustes são comuns após a implantação do sistema. Deste modo, esta fase também é responsável pela

realização destes ajustes necessários. Além disso, nesta etapa, é possível realizar atividades paralelas como treinamento de usuários.

Além da visão de fases, como relatado acima, o RUP (2003) apresenta uma visão de fluxos de atividades, que é composta por disciplinas de engenharia de software e disciplinas de suporte, descritas a seguir.

- **Modelagem de Negócios**

Disciplina que tem como objetivo a documentação de processos de negócio, com a finalidade de facilitar a comunicação entre as equipes de engenharia de negócio e engenharia de software.

- **Requisitos**

Tem como propósito descrever todo o funcionamento do sistema, relatando o que ele se propõe a fazer, de modo que os clientes e desenvolvedores estejam de acordo com o descrito. Essa descrição é realizada por meio da elicitação, organização e documentação das funcionalidades e restrições do sistema. Além disso, essa disciplina tem como objetivo rastrear e documentar compromissos e decisões firmadas.

- **Análise e Projeto**

O objetivo principal desta disciplina é dar uma visão de como o sistema será concretizado na implementação. Tem como foco provar que o sistema irá executar todas as tarefas e funções especificadas, satisfazendo os requisitos estabelecidos, possuindo robustez e facilidade de mudanças.

- **Implementação**

Esta disciplina é responsável por organizar o código em camadas, componentes e pacotes; implementar as classes e objetos usando código;

realizar testes unitários nos componentes desenvolvidos, bem como, integrar os resultados para produzir um sistema executável.

- **Testes**

É uma tarefa muito importante e que vai verificar, antes da liberação, se o sistema atende às necessidades do cliente. Responsável por verificar a integração entre objetos, a integração adequada entre os componentes de software e se o sistema atende os requisitos definidos. Adicionalmente, é neste momento que a identificação e correção dos defeitos são realizadas, antes da entrega ao cliente.

- **Instalação**

Responsável pela entrega bem sucedida do software ao cliente por meio da produção de releases, empacotamento, distribuição e instalação do software. É também neste momento que se disponibiliza auxílio aos usuários finais do sistema.

Além dos fluxos de engenharia de software, a visão vertical do RUP apresenta alguns fluxos de suporte. São eles:

- **Gerência de Projeto**

É a atividade responsável por controlar todo o projeto, balanceando os objetivos conflitantes dos envolvidos, de modo a buscar superação dos problemas e entregar o produto com sucesso, satisfazendo as necessidades e anseios dos clientes e usuários.

- **Gerência de Configuração e Mudanças**

Tem como objetivo principal o controle dos artefatos produzidos durante o desenvolvimento do software, garantindo que será possível a realização de atualizações simultâneas e manutenção de múltiplas versões.

- **Gerência de Ambiente**

Atividade responsável por prover o ambiente necessário e adequado para a organização, que forneça as ferramentas e processos capazes de suportar as atividades realizadas pela equipe que desenvolve o sistema.

3.3 Metodologias Ágeis

Souza (2003) relata que, a partir de 2001, especialistas na área de desenvolvimento de software se juntaram para encontrar valores e princípios acerca do desenvolvimento e que pudessem ser capazes de fazer com que as equipes respondessem mais rápido às mudanças constantes nas especificações e o projeto fosse desenvolvido com mais agilidade. Essa atitude foi motivada pela confusão entre os processos existentes na época, por parte dos times de desenvolvimento.

Essa união de profissionais resultou na escrita de um manifesto denominado de "*Manifesto for Agile Software Development*", que destacava quatro valores [Fowler, 2001] [Agile Manifesto, 2004]:

- Indivíduos e iterações ao invés de processos e ferramentas;
- Software funcional ao invés de documentação detalhada;
- Colaboração do cliente ao invés de negociação de contratos;
- Responder às mudanças ao invés de seguir um plano.

Segundo Fowler (2001), o "Manifesto Ágil" não desconsidera os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, apenas tenta mostrar que esses artifícios não têm prioridade de importância diante dos valores pregados por ele: indivíduos e interações, software executável, colaboração do cliente e respostas rápidas a mudanças e alterações. Os conceitos preconizados pelo movimento ágil estão mais próximos

da forma que pequenas e médias organizações trabalham e respondem a mudanças.

Foi a partir desse manifesto que surgiram as metodologias ágeis de desenvolvimento de software que, segundo Soares (2006), são uma resposta às chamadas metodologias tradicionais. Ele evidencia tal fato quando diz:

> *"Metodologias ágeis têm sido apontadas como uma alternativa às abordagens tradicionais para o desenvolvimento de software. As metodologias tradicionais, conhecidas também como pesadas ou orientadas a planejamentos, devem ser aplicadas apenas em situações em que os requisitos do sistema são estáveis e requisitos futuros são previsíveis. Entretanto, em projetos em que há muitas mudanças, em que os requisitos são passíveis de alterações, onde refazer partes do código não é uma atividade que apresenta alto custo, as equipes são pequenas, as datas de entrega do software são curtas e o desenvolvimento rápido é fundamental, não pode haver requisitos estáticos, necessitando então de metodologias ágeis. Além disso, o ambiente das organizações é dinâmico, não permitindo então que os requisitos sejam estáticos".*

Uma característica importante das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. Isso leva a uma adaptação a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento [Cockbum e Highsmith, 2001].

O movimento ágil segue alguns princípios que estão listados a seguir.

1. O cliente está acima de tudo. É preciso satisfazê-lo através da entrega precoce e contínua do software;

2. Utilização das mudanças como uma vantagem competitiva com o cliente;
3. Entregar software em funcionamento com freqüência;
4. Integração entre pessoas de negócio e desenvolvedores em todo desenvolvimento do projeto;
5. Motivar e confiar no trabalho dos indivíduos e proporcionar ambiente e suporte necessários;
6. Comunicação face a face para transmitir informações para e dentro da equipe;
7. A primeira medida de progresso é o software em funcionamento;
8. Processos ágeis promovem um desenvolvimento sustentável, patrocinadores, desenvolvedores e usuários devem estar aptos a manter o passo constante indefinidamente;
9. Maior agilidade através da excelência técnica e bom projeto;
10. Simplicidade é essencial;
11. Equipe se organiza por si só, para criação de melhores arquiteturas, requisitos e projetos;
12. Reflexão constante de como melhorar a eficiência, para possíveis ajustas de comportamento.

Apesar da constante evolução dos computadores, das técnicas e ferramentas nos últimos anos, a produção de software confiável, correto e entregue dentro dos prazos e custos estipulados ainda é muito difícil.

Processos orientados a documentação para o desenvolvimento de software, como o modelo em Cascata, são de certa forma fatores limitadores aos

desenvolvedores. Além disso, muitas organizações não possuem recursos ou inclinação para processos pesados de produção de software. Por esta razão, muitas organizações, particularmente as pequenas, acabam por não usar nenhum processo, o que pode levar a efeitos desastrosos em termos de qualidade de software.

De certo modo, os processos orientados a documentação para o desenvolvimento de software limitam os desenvolvedores, e muitas organizações não possuem recursos para implantação de processos pesados de produção de software. Por esta razão, as organizações pequenas acabam por não usarem nenhum processo [Soares, 2006]. Isto pode levar a efeitos desastrosos na qualidade do produto final, além de dificultar a entrega do software nos prazos e custos predefinidos. Para contornar esta situação, Santos sugere a utilização de metodologias ágeis, que não são orientadas à documentação nem tampouco se preocupam apenas com a codificação.

Para que uma metodologia seja considerada ágil é preciso que ela aceite mudanças ao invés de tentar prever o futuro. Problemas acontecem de qualquer forma, deste modo, a metodologia precisa estar preparada para receber, avaliar e responder às mudanças. Para Soares, agindo assim, há maiores possibilidades de atender aos requisitos do cliente, que muitas vezes são mutáveis.

É necessário que haja uma maior quantidade de projetos que fazem uso de metodologias ágeis para verificar suas reais vantagens, apesar do uso crescente destas. De qualquer forma, os primeiros resultados, em termos de qualidade, confiança, datas de entrega e custo, são promissores.

Dentre as várias metodologias ágeis que existem, destacam-se a *Extreme Programming* e a *Scrum*, que serão detalhadas nas próximas seções deste capítulo.

3.3.1 Extreme Programming

A *Extreme Programming* (XP) é uma metodologia ágil ideal para equipes de médio e pequeno porte que fazem uso de requisitos vagos e mutáveis para desenvolver software [Beck, 1999]. Dentre as principais diferenças da XP em relação às outras metodologias estão:

- *Feedback* constante;
- Abordagem incremental;
- A comunicação entre as pessoas é encorajada.

De acordo com Highsmith el al (2000), o primeiro projeto a usar XP foi o C3, da Chrysler. Após anos de fracasso utilizando metodologias tradicionais, com o uso da XP, o projeto ficou pronto em pouco mais de um ano.

As regras preconizadas pela XP, se adotadas isoladamente, dificilmente surtirão efeito. É justamente o uso em conjunto destas práticas que sustenta a metodologia e proporciona ótimos resultados no desenvolvimento de software [Nonemacher, 2003].

O desenvolvimento rápido do projeto, a garantia da satisfação do cliente e o cumprimento das estimativas são pontos fortemente enfatizados por XP. Para Beck (1999), as regras, práticas e valores da XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores: comunicação, simplicidade, feedback e coragem.

De acordo com Nonemacher (2003), o princípio da comunicação enfatiza que se deve manter melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. Estimula-se, também, a comunicação entre os desenvolvedores e o gerente de projeto.

A simplicidade tem como objetivo estimular a criação de código simples que não deve possuir funções desnecessárias, com o menor número possível de classes e métodos.

A realização de *feedback* constante permite que o desenvolvedor tenha acesso a informações do código (dadas pela execução constante de testes) e cliente (recebidas pela avaliação constante realizada) com freqüência. Sendo assim, a tendência é que o produto final esteja de acordo com as expectativas reais do cliente.

O princípio da coragem é quem permite determinação para implantar os três valores anteriores, pois nem todas as pessoas possuem facilidade de comunicação e têm bom relacionamento. A coragem também proporciona simplicidade, pois assim que a oportunidade de simplificar o software é percebida, a equipe pode experimentar. Além disso, é preciso coragem para obter feedback constante do cliente.

De acordo com Beck (1999), a metodologia XP tem como base 12 práticas que estão descritas abaixo:

- **Planejamento** - consiste em decidir o que é necessário ser feito e o que pode ser adiado no projeto. XP baseia-se em requisitos atuais para desenvolvimento de software, não em requisitos futuros. Os programadores tomam decisões técnicas (estimativas) e os clientes tomam decisões empresariais, selecionando as funcionalidades com a maioria de benefícios.

- **Entregas freqüentes** - visa à construção de um software simples, com entregas freqüentes, onde cada versão entregue deve ter o menor tamanho possível, contendo os requisitos de maior valor para o

negócio. Esta prática aumenta a probabilidade do software final estar de acordo com os requisitos do cliente.

- **Metáfora** - são as descrições de um software sem a utilização de termos técnicos, com o intuito de guiar o desenvolvimento do software.

- **Projeto simples** – em XP, o desenvolvimento deve ser o mais simples possível, para diminuir o custo com manutenção, e satisfazer apenas os requisitos atuais, esquecendo os futuros.

- **Testes** - os programadores desenvolvem os casos de teste antes de escreverem o código e utilizam os testes para focar no que tem que ser alcançado.

- **Programação em pares** - a implementação do código é feita em dupla que compartilha uma única máquina. Enquanto um codifica, o outro analisa atenciosamente o código criado. Esses papéis podem e devem ser alterados continuamente.

- **Refatoração** - é o processo de melhorar o código sem afetar seu comportamento. O código deve ser mantido tão simples quanto possível, pronto para qualquer mudança futura que apareça [Wake, 2000].

- **Propriedade coletiva** – todos os membros da equipe são responsáveis pelo código do projeto e este pertence a todos. Sendo assim, qualquer pessoa pode alterar o código, se achar que vai melhorá-lo, desde que faça a bateria de testes necessária. Isso permite que todos conheçam todas as partes do software, mesmo que não seja de forma detalhada.

- **Integração contínua** - a equipe XP trabalha em etapas curtas e integra o código periodicamente, mantendo os programadores em sintonia,

além de possibilitar processos rápidos. Adicionalmente, esta prática permite que os problemas de integração sejam descobertos logo ao serem criados, desta forma é razoavelmente fácil retificá-los.

- **40 horas de trabalho semanal** - XP assume que não se deve fazer horas extras constantemente. Esta prática procura ratificar o foco nas pessoas e não em processos e planejamentos. Caso seja necessário, os planos devem ser alterados, ao invés de sobrecarregar as pessoas.

- **Cliente presente** - é fundamental a participação do cliente durante todo o desenvolvimento do projeto. O cliente deve estar sempre disponível para sanar todas as dúvidas de requisitos, evitando atrasos e até mesmo construções erradas. Uma idéia interessante é manter o cliente como parte integrante da equipe de desenvolvimento [Hayes, 2001].

- **Código padrão** - padronização na arquitetura do código, para que este possa ser compartilhado entre todos os programadores, já que codificar é uma atividade de equipe.

A Figura 7 apresenta as práticas descritas acima que são preconizadas pela metodologia XP.

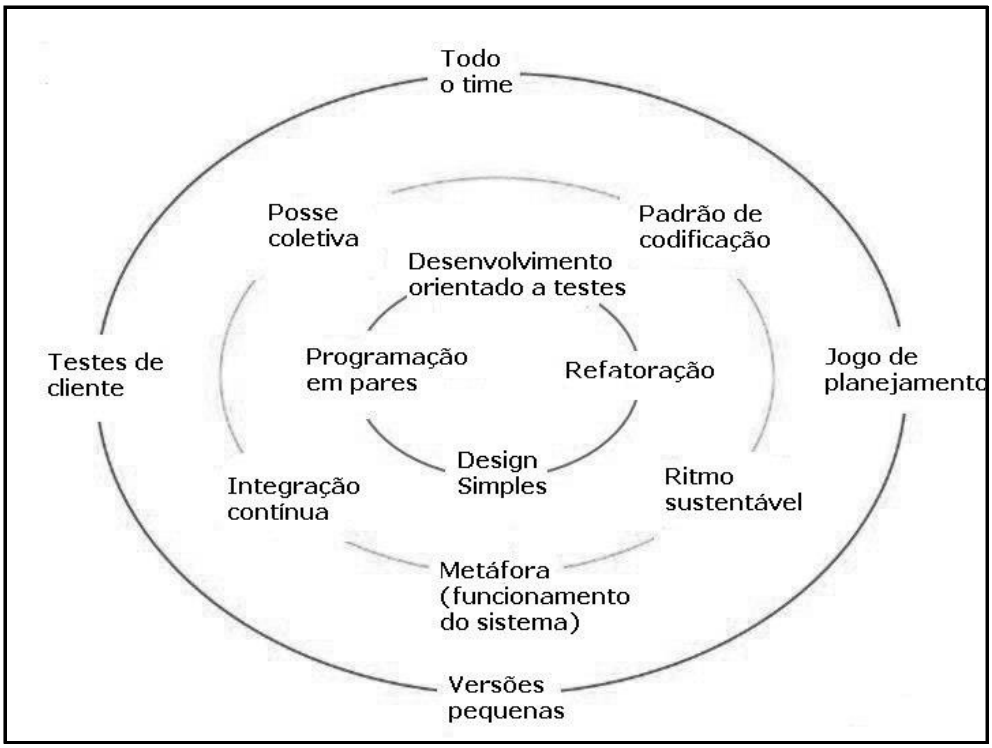

Figura 7 - Práticas da Metodologia XP. [Fonte: Beck (1999)]

3.3.2 Scrum

O *Scrum* é uma metodologia ágil e flexível para gerenciamento de projetos. É caracterizada por definir um processo iterativo e incremental que foca na aceitação das mudanças que podem ocorrer, principalmente em contextos pouco definidos [Ferreira et al, 2004].

Centrado no trabalho em equipe, a metodologia Scrum melhora a comunicação e maximiza a cooperação, possibilitando que cada pessoa envolvida no desenvolvimento faça o melhor de si e se sinta confortável com a atividade que executa. Para Ferreira et al, posteriormente, este comportamento será refletido em aumento de produtividade.

Scrum é aplicável a projetos de qualquer porte, sejam eles grandes ou pequenos. O objetivo principal do Scrum é a facilidade de adaptação constante

em qualquer situação, procurando atender aos interesses e necessidades dos clientes.

O Scrum não requer nem oferece qualquer técnica ou método específico para a fase de desenvolvimento de software. Seu propósito é disponibilizar um conjunto de regras e práticas de gestão que devem ser adaptadas para cada situação, para garantia do sucesso do projeto [Schwaber et al, 2002].

Atividades de monitoramento de *feedback* estão contidas no Scrum. Estas atividades são realizadas através de reuniões rápidas e diárias com toda a equipe e têm como finalidade a identificação e correção de quaisquer deficiências e/ou impedimentos no processo de desenvolvimento. Outro ponto recomendado pelo Scrum é a formação de equipes pequenas de, no máximo, 7 pessoas e sugere que o desenvolvimento seja realizado em iterações curtas, de no máximo 30 dias, também chamadas de Sprints. [Schwaber et al, 2002].

Para Schwaber (2004), na execução do Scrum, há um facilitador chamado de Scrum Master. Ele tem como responsabilidade o monitoramento da execução da metodologia e atua como mediador entre a equipe de desenvolvimento e o *Product Owner*. Este último é o responsável, dentro da equipe do projeto, por representar o patrocinador do projeto. Deste modo, é ele quem prioriza e acompanha as funcionalidades a serem desenvolvidas no projeto, identificando aquelas que agregam mais valor.

O Scrum estabele alguns artefatos que devem ser gerados durante sua execução e que auxiliam no gerenciamento de projetos. Os artefatos estão descritos abaixo [Barros, 2007]:

- **Product Backlog** – é uma lista de funcionalidades que o produto deve conter. A lista é priorizada pelo *Product Owner*.

- **Sprint Backlog** – é uma lista de tarefas, definida em negociação pelo *Product Owner* e a equipe de desenvolvimento, a serem realizadas em cada *sprint* do projeto.

- **Impediment List** – é uma lista composta por todos os problemas identificados pela equipe de desenvolvimento que impedem o alcance dos objetivos. A resolução dos problemas deve ser feita pelo *Scrum Master*.

- **Product Backlog Burn Down** – é um documento que permite a visualização do status do projeto, visto que apresenta as funcionalidades que já foram completadas em um determinado momento.

- **Sprint Backlog Burn Down** – documento que especifica as atividades já concluídas em uma determinada iteração.

O Scrum é composto por fases que procuram auxiliar de forma efetiva o processo de desenvolvimento de software. Abaixo, temos as fases [Barros, 2007]:

1. **Preparação:** É a fase onde são realizadas as definições iniciais do projeto. É nesta ocasião que o artefato *Product Backlog* é criado.

2. **Sprint:**

    a. Encontro de Planejamento do Sprint (*Sprint Planning Meeting*): É o primeiro momento em que os papéis envolvidos no projeto se reúnem no *Sprint*. Após a construção do *Product Backlog* na fase anterior, agora é o momento do *Product Owner* priorizar as funcionalidades listadas no artefato. Além disso, é definido o que deverá ser entregue no *Sprint*;

b. Encontro Diário do Scrum (*Daily Scrum Meeting*): Encontros que devem ser realizados diariamente, onde cada participante externa para os demais as atividades que foram realizadas, bem com as que ainda estão pendentes. Além disso, os participantes relatam os problemas identificados durante a execução das atividades já realizadas;

c. Incremento do Produto (*Product Increment*): Representa o conjunto de funcionalidades que foram acordadas e devem ser entregues em um *Sprint*;

d. Revisão do Sprint (*Sprint Review*): Ocasião em que o *Product Owner* valida o *Product Increment* gerado pela equipe, podendo ou não solicitar mudanças;

e. Retrospectiva do Sprint (*Sprint Retrospective*): Reunião realizada com todos os integrantes da equipe que visa o levantamento dos pontos positivos e negativos detectados durante o último *Sprint* executado;

f. Atualização do Backlog do Produto (*Update Product Backlog*): Revisão de prioridade, realizada pelo *Product Owner*, das funcionalidades a serem entregues no próximo *Sprint* do projeto.

3. **Encerramento:** Após a realização de todos os *Sprints* do projeto, esta fase é realizada com a entrega do produto implementado.

Para uma melhor visão do Scrum, a Figura 8 apresenta uma modelagem de suas fases, papéis e artefatos.

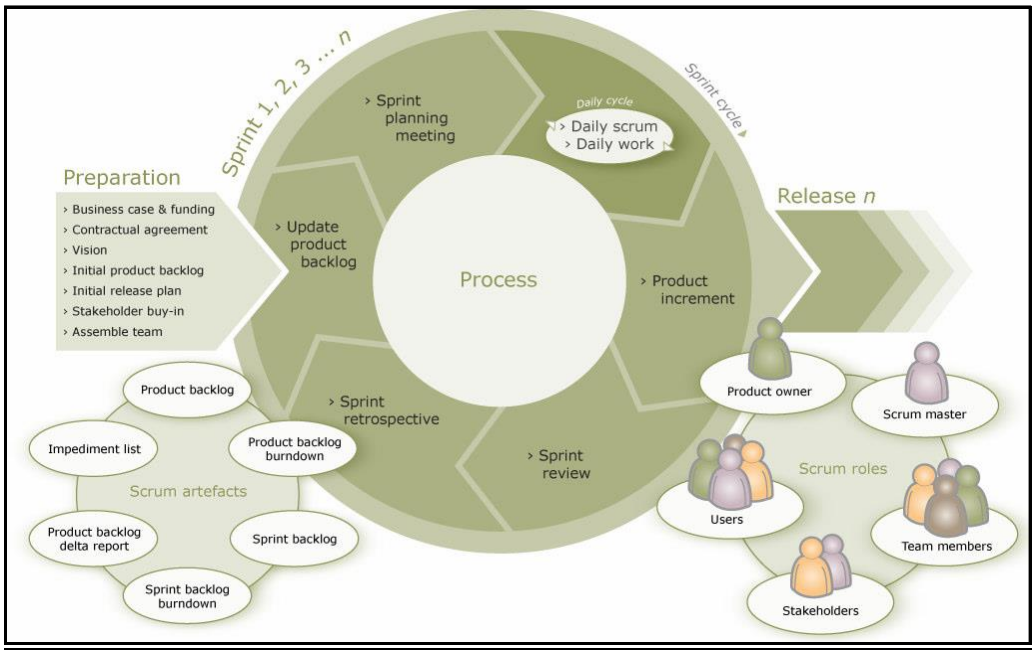

Figura 8 - Visão Geral do Scrum. [Fonte: Barros (2007)]

3.4 Metodologias Orientadas a Agentes

Com o constante crescimento das abordagens orientadas a agentes enquanto modelo promissor na computação, nota-se que para o desenvolvimento de sistemas complexos e distribuídos, tal abordagem tem sido indicada por propiciar um maior nível de abstração se comparada com outras abordagens.

Autonomia e habilidade social são algumas das características dos agentes de software. Comunicando entre si, os agentes coordenam e cooperam para atingir seus objetivos. De acordo com [Dário, 2005], atualmente, os projetistas de software ainda não podem explorar todos os benefícios oferecidos pelo paradigma de agentes devido à ausência de notações diagramáticas, metodologias e ferramentas de projeto conhecidas para o desenvolvimento de sistemas orientado a agentes [Bergenti e Poggi, 2000]. Portanto, a comunidade

acadêmica da Engenharia de Software Orientada a Agentes[7] está interessada na construção de metodologias de suporte a sistemas multi-agentes.

Deste modo, as próximas seções apresentam as metodologias Prometheus e MaSE, que estão sendo amplamente citadas na literatura [Dário, 2005].

3.4.1 Prometheus

Prometheus é uma metodologia da Engenharia de Software Orientada a Agentes para especificar, projetar e implementar sistemas de agentes [Padgham e Winikoff, 2005], que há vários anos está sendo desenvolvida em colaboração com a *Agent Oriented Software* (AOS – Estados Unidos, Reino Unido e Austrália) [Dário, 2005]. A principal meta no desenvolvimento desta metodologia é ter a definição consistente de um processo coerente, de modo a ser utilizado na construção de sistemas de agentes. Prometheus é capaz de lidar com o desenvolvimento de agentes inteligentes que possuem objetivos, crenças, planos e eventos, já que proporciona mecanismos de estruturação hierárquica que permite ser executado em diferentes níveis de abstração.

De acordo com Dário (2005), a metodologia parte do conceito de capacidade, vislumbrada pela idéia de planos, eventos, crenças e outras capacidades que dão habilidades específicas ao agente. O agente é composto por várias capacidades e cada uma delas tem uma função específica.

A metodologia é composta por fases, são elas:

- Especificação do Sistema;

---

[7] Segundo Dam (2003), "Engenharia de Software Orientada a Agentes é a aplicação de agentes na Engenharia de Software em termos de fornecer meios para analisar, projetar e construir sistemas de software".

- Projeto Arquitetural;
- Projeto Detalhado.

Prometheus possui como prática a aplicação processo iterativo em cada fase, seguindo a abordagem do RUP. Outra característica da metodologia é que alguns artefatos gerados são observados enquanto variações da UML.

A Figura 9 apresenta uma visão geral da metodologia, destacando os seus artefatos produzidos em cada fase.

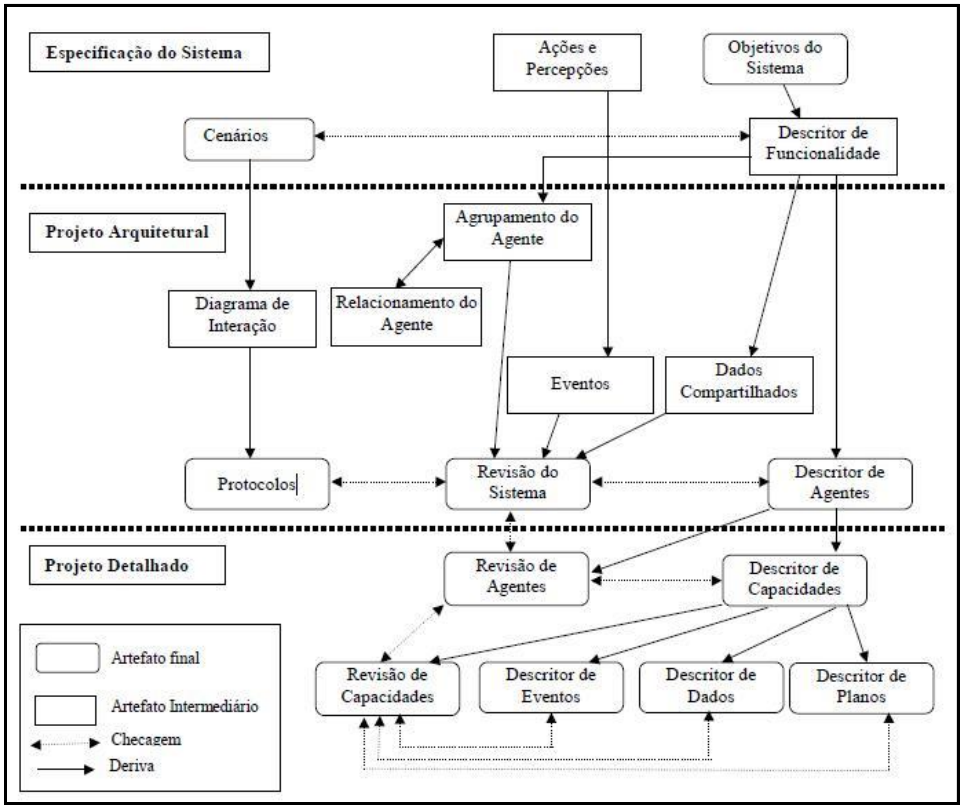

Figura 9 - Visão Geral da Prometheus [Fonte: Dário (2005)]

A seguir, temos mais detalhadamente o propósito de cada fase da metodologia.

1. **Especificação do Sistema**

O conceito de funcionalidade utilizado pelo Prometheus equivale ao conceito de papéis em outras metodologias baseadas em agentes. Portanto, o objetivo dessa fase é detectar as funcionalidades básicas, definindo suas percepções e ações O artefato que guia esta atividade é o "Descritor de Funcionalidade". Segundo Dário (2005), inicialmente, são elencados os objetivos principais do sistema e, em seguida, as funcionalidades que irão atingir esses objetivos. Como as funcionalidades lidam com os aspectos específicos do sistema, faz-se necessário definir os Cenários do Caso de Uso para obter uma visão mais completa do sistema. Um cenário de caso de uso típico consiste de passos que descrevem as percepções, as mensagens enviadas e as ações dos agentes. Outro artefato gerado durante esta fase é o "Descritor de Cenário", que contém os cenários identificados.

2. **Projeto Arquitetural**

Determinar os tipos de agentes, elaborar a estrutura do sistema e definir as interações entre os agentes são os objetivos dessa fase. As funcionalidades são divididas em dois critérios:

a. Coesão: funcionalidades que estão relacionadas (por exemplo, usam o mesmo dado);

b. Ligação: interações significantes entre duas funcionalidades.

As propriedades que permitem a união das funcionalidades que fazem uso do mesmo banco de dados são examinadas pelo "Diagrama de Ligação de Dados". Para avaliar o agrupamento das funcionalidades, Prometheus usa um artefato denominado "Diagrama de Relacionamento do Agente" [Padgham e Winikoff, 2005].

Após a definição de agentes, o passo seguinte é a identificação dos eventos que os agentes responderão e as mensagens produzidas por eles. Estas informações são modeladas com a utilização dos seguintes artefatos: "Descritor de Percepção", "Descritor de Ação", "Descritor de Mensagem" e "Descritor de Tipo de Agente". O passo seguinte é criar o "Diagrama de Revisão do Sistema", que apresenta os agentes, as percepções, as mensagens e as ações dos agentes, representando o principal artefato de projeto [Padgham e Winikoff, 2005].

3. **Projeto Detalhado**

De acordo com Dário (2005), esta última fase da metodologia está focada no desenvolvimento da estrutura interna do agente e em como ele executa suas atividades no sistema, detendo-se na definição das capacidades internas, planos e estruturas de dados detalhadas, para cada tipo de agente elaborado anteriormente. As funcionalidades definidas na fase de especificação se transformam em conjunto de capacidades. As capacidades são refinadas até serem definidos os planos, os eventos e os dados, utilizando o "Diagrama de Revisão do Agente", que fornece uma visão de alto nível da estrutura interna do agente.

Em seguida, Prometheus faz uso do "Diagrama de Revisão de Capacidade" para descrever a estrutura interna de cada capacidade.

Os artefatos finais desta fase são os descritores de plano, de eventos e de estrutura de dados.

Prometheus tem suporte de duas ferramentas [Padgham e Winikoff, 2002]:

1. Jack Development Environment (JDE): para modelagem dos diagramas de revisão existentes na metodologia.

2. Prometheus Design Tool (PDT): que permite analisar a consistência entre os diagramas, além de gerar a documentação do projeto.

3.4.2 MaSE

MaSE (Multiagent Systems Engineering Methodology) é uma metodologia orientada a agentes, desenvolvida pela *Air Force Institute of Technology* (AFIT) [Wood e DeLoach, 2001], que considera a existência prévia da fase de especificação de requisitos para seu início e prossegue com as fases seguintes até sua implementação. O propósito da MaSE é dar suporte ao projetista durante todo o ciclo de vida de desenvolvimento do software.

A metodologia MaSE possui duas fases [Dário, 2005]:

- Análise - responsável pelas atividades de definição dos objetivos do sistema, aplicação de caso de uso e refinamento de papéis.
- Projeto - é dividida na criação de classes de agente, construção de conversa entre os agentes, montagem de classes de agentes e projeto do sistema.

As fases da metodologia podem ser observadas na Figura 10, que apresenta uma visão geral da MaSE [Wood e DeLoach, 2001].

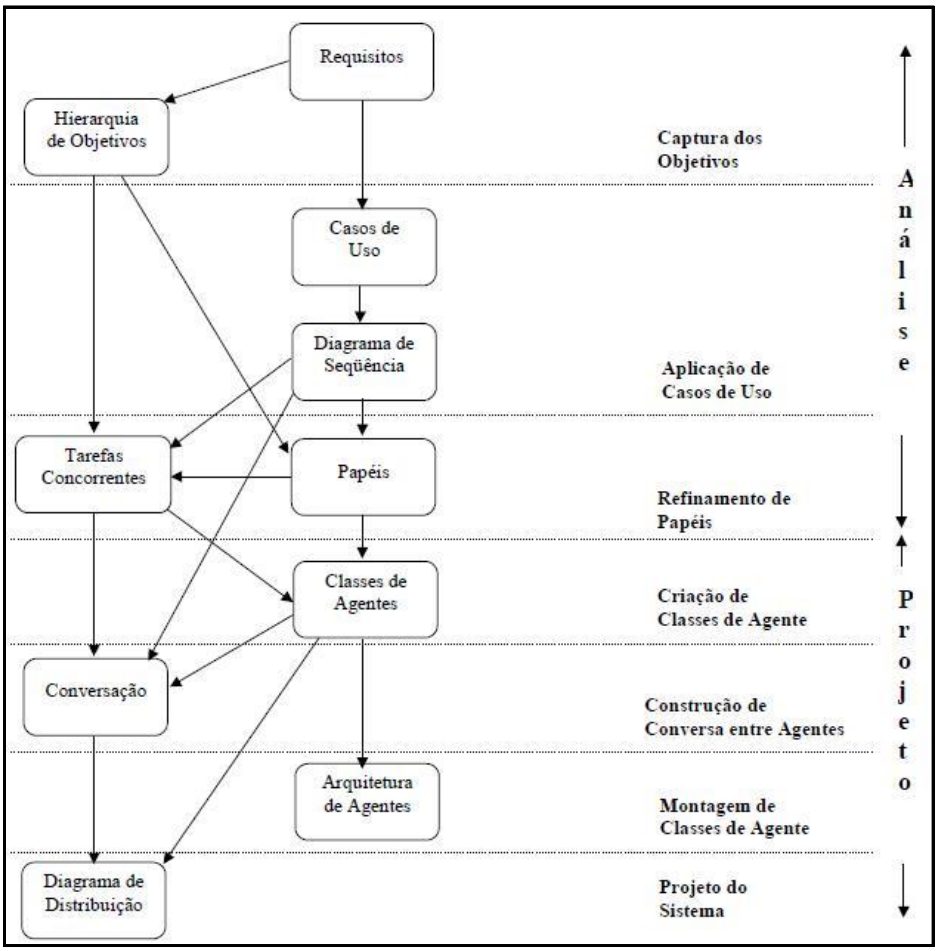

Figura 10 - Visão Geral da MaSE [Fonte: Dário (2005)]

A seguir, temos as fases da metodologia de forma mais detalhada.

1. **<u>Análise</u>**

Ao definir os objetivos de alto-nível do sistema, que representa a primeira atividade a ser realizada, passa-se a extrair os casos de uso e modelá-los no Diagrama de Seqüência, no passo de aplicação de casos de uso. É neste passo que se estabelece um conjunto inicial de papéis do sistema e a comunicação. Os passos referentes à análise estão descritos a seguir [Dário, 2005]:

a. <u>Captura dos objetivos:</u> primeiramente, é feita a identificação de um conjunto de objetivos do sistema, retirado da sua especificação inicial do sistema, assumindo a existência prévia de uma

representação textual dos requisitos. Em um segundo momento, esses objetivos são hierarquicamente organizados, pelo grau e importância para o sistema, em um diagrama chamado Diagrama Hierárquico de Objetivos.

b. Aplicação de Casos de Uso: consiste de duas atividades. Na primeira são identificados os casos de uso, elaborados a partir dos requisitos do sistema. Em seguida, os casos de uso são detalhados da seguinte forma:

   - ✓ Identifica-se um conjunto inicial de papéis baseados nos objetivos e casos de uso;
   - ✓ Os casos de uso são reestruturados através do Diagrama de Sequência, definem a seqüência de eventos entre múltiplos papéis, atividades e estabelecem a comunicação mínima entre eles.

c. Refinamento de Papéis: nesta ocasião é realizado o refinamento do conjunto inicial de papéis que foi definido, mapeando os objetivos em papéis, que devem ser representados no Modelo de Papéis. Em seguida, devem ser criadas as tarefas necessárias para que esses papéis possam cumprir seus objetivos, representadas no Modelo de Papéis detalhado. Durante este passo, pode ser necessário modelar tarefas concorrentes, realizado pelo Diagrama de Tarefas Concorrentes.

2. **Projeto**

Esta fase é composta por quatro passos: criação de classes de agente, construção de conversa entre os agentes, montagem de classes de agentes e projeto do sistema. A seguir, temos o detalhamento de cada passo [Dário, 2005]:

a. Criação de classes de agentes: são criadas classes de agentes a partir dos papéis identificados na fase anterior. No Diagrama de Classes de Agente, artefato gerado neste passo, os relacionamentos representam a comunicação mantida entre as classes de agentes.

b. Construção de conversas de agentes: momento em que são definidas as conversas entre os agentes. A conversação prescinde de um protocolo de coordenação entre as duas classes de agentes que participam da conversa, modelados por dois Diagramas de Classes de Comunicação.

c. Montagem de classes de agentes: ocasião em que a arquitetura é criada. De acordo com Robinson (2000) há cinco tipos diferentes de arquiteturas multi-agentes: crença-desejo-intenção, reativo, planejamento, arquitetura baseada em conhecimento e arquitetura definida pelo usuário.

d. Projeto do Sistema: as classes definidas anteriormente são instanciadas e distribuídas no sistema, pela utilização do artefato Diagrama de Distribuição, responsável por definir a configuração do sistema que será implementado.

3.5 Discussão

A apresentação das metodologias de desenvolvimento de software feita neste capítulo, bem como o estudo geral relacionado a outras metodologias de desenvolvimento, foram fundamentais para a definição da metodologia proposta nesta dissertação.

O estudo feito neste capítulo propiciou a definição de requisitos mínimos para o desenvolvimento de companheiros virtuais inteligentes, em outras palavras, um levantamento das necessidades que uma metodologia voltada para o desenvolvimento de CVIs precisa contemplar para que o processo de construção seja efetivo e alcance os resultados esperados.

Os requisitos gerais identificados de uma metodologia para construção de CVI estão listados a seguir.

1. É importante que a metodologia percorra todo o ciclo de desenvolvimento de um companheiro virtual inteligente, dando suporte e gerando artefatos em todas as fases.

2. A metodologia deve apresentar praticidade na execução de suas atividades.

3. Quanto mais utilizada, mais madura e confiável é a metodologia. Sendo assim, um requisito importante que define uma metodologia é sua maturidade quanto a seu uso em casos reais de desenvolvimento.

4. Um estudo e caracterização do domínio de aplicação do companheiro virtual inteligente são fundamentais, visto que essa atividade facilita e dá insumos para a especificação do agente.

5. Uma boa especificação do companheiro virtual inteligente é um suporte esclarecedor para todo o ciclo de desenvolvimento.

6. A existência de prática de avaliação do companheiro virtual inteligente após sua construção torna-se um diferencial no que diz respeito à garantia da qualidade do produto.

7. Os documentos gerados na metodologia devem ser facilmente identificados e representar um encadeamento das fases.

8. A metodologia deve apresentar práticas de desenvolvimento que podem ser usadas para facilitar o desenvolvimento.

9. A complexidade na construção de companheiros virtuais inteligentes requer uma comunicação eficiente entre os *stakeholders*, para uma melhor disseminação do conhecimento.

Os requisitos listados acima formam o conjunto mínimo ideal para uma metodologia voltada para a construção de CVIs. No entanto, não há a necessidade de se prender apenas aos requisitos definidos. As metodologias podem possuir outros requisitos não contemplados neste trabalho.

4 UMA METODOLOGIA PARA DESENVOLVIMENTO DE COMPANHEIROS VIRTUAIS INTELIGENTES

Este capítulo se configura como o mais relevante, visto que apresenta o foco do trabalho: uma metodologia para construção de companheiros virtuais inteligentes, definida a partir de estudo e análise realizados durante a elaboração desta dissertação. Inicialmente, na Seção 4.1, é apresentada a caracterização da metodologia, onde se reforça a importância de um metodologia que guie o desenvolvimento de CVIs, em seguida, são citados os objetos de estudo que serviram de insumo na concepção da metodologia, além de apontar as principais práticas preconizadas por esta, para a garantia de sucesso do projeto. Na Seção 4.2, é apresentado um fluxograma que representa uma visão geral da metodologia, explicando sucintamente cada uma das fases. Na seção seguinte, a 4.3, são listados e descritos os papéis envolvidos na realização de todas as atividades que representam a metodologia, bem como define-se todos os artefatos de entrada e a saída produzidos. Por fim, na Seção 4.4, são descritas as fases que fazem parte do modelo. Cada fase existe o detalhamento das atividades, os papéis envolvidos, os recursos que são necessários, bem como os respectivos artefatos gerados. Além disso, cada fase apresenta uma tabela que resume todas suas informações, como também, uma representação em forma de diagrama.

4.1 Fundamentos

A necessidade da construção e inserção de um companheiro virtual em um meio é advinda de um problema ainda não solucionado: a deficiência na

interação homem x computador. É nesse sentido, que a metodologia aqui proposta pretende nortear o desenvolvimento destes agentes e, conseqüentemente, amenizar os problemas ainda existentes nos domínios de aplicação.

Como suporte para a definição da metodologia, foi realizado um estudo de metodologias tradicionais e ágeis de desenvolvimento, bem como metodologias orientadas a agentes, na busca de ter uma visão panorâmica destas. Esta análise foi realizada mais fortemente no RUP, XP, Scrum, Prometheus e MaSE [Torreão, 2005].

Na análise realizada foi verificado que no desenvolvimento de companheiros virtuais para serem integrados a ambientes, é necessário e prioritário um estudo sobre o domínio em que este será inserido e integrado. Por esta razão, foi definida uma fase exclusiva para o estudo do domínio, buscando ressaltar seus problemas mais evidentes.

A metodologia proposta apresenta algumas práticas a serem seguidas para o desenvolvimento efetivo e eficaz do software. As práticas aqui listadas foram definidas a partir de uma seleção de outras já consolidadas, que obtiveram sucesso e que têm grandes chances de serem úteis no desenvolvimento de CVIs. A seguir, temos as melhores práticas a serem aplicadas.

4.1.1 Desenvolvimento Iterativo e Incremental

A construção do software (companheiro virtual) é realizada através de ciclos de desenvolvimento. Em cada ciclo são executadas todas as fases definidas na metodologia. Ao final de cada ciclo o produto é incrementado com novas funcionalidades.

Essa característica no desenvolvimento facilita o controle das possíveis alterações em requisitos e dos riscos do projeto. Ao final de cada ciclo é possível realizar o re-planejamento, contemplando tais alterações.

4.1.2 Comunicação Eficiente

Um princípio muito importante, mais evidente nas metodologias ágeis, e que se faz necessário no desenvolvimento de companheiros virtuais inteligentes é a utilização da comunicação como meio de disseminação do conhecimento, bem como de integração dos papéis envolvidos no projeto.

A complexidade na construção de agentes requer uma estratégia de comunicação eficiente, que possa amenizar tal dificuldade por meio de uma integração com troca de experiências. Além disso, a comunicação se faz primordial na relação entre cliente e desenvolvedores [Nonemacher, 2003].

4.1.3 Importância dos Testes

Qualquer software necessita de uma atividade de testes consistente que possa aferir com precisão a qualidade do produto [Black, 2007]. No desenvolvimento de CVIs não é diferente. Sendo assim, a metodologia proposta prima por uma preocupação mais direcionada na validação destes agentes, visto que eles apresentam características e comportamentos humanos que precisam ser cuidadosamente analisados.

4.1.4 Gerenciamento dos Riscos na Construção

Os riscos de um projeto ditam suas possíveis vulnerabilidades. No ciclo de vida de um software, a capacidade de mitigar riscos acentua a possibilidade de sucesso do projeto [Cooper et al, 2005]. Sendo assim, a metodologia orienta que os riscos sejam levantados, priorizados e atacados no início de cada iteração do desenvolvimento.

O gerente de projeto precisa tomar ações corretas para sanar ou minimizar os riscos, por esta razão o gerenciamento se faz necessário.

4.1.5 Caracterização e Modelagem Visual do Domínio

Como a metodologia se propõe a dar suporte ao desenvolvimento de CVIs para qualquer domínio, sua caracterização e modelagem visual ajudam na aquisição do entendimento, dimensão e complexidade do sistema, de modo que se torna possível conhecer todos os problemas existentes no domínio.

4.1.6 Controle de Mudanças

A prática de desenvolvimento iterativo torna o software sujeito a diversas mudanças durante seu desenvolvimento, sendo assim, a metodologia possui como estratégia o controle efetivo destas mudanças para uma melhor garantia da qualidade do produto que está sendo elaborado.

Na metodologia é fundamental que as mudanças sejam priorizadas de acordo com grau de importância estabelecido pelo cliente, visando atendê-las o mais breve possível no início da etapa de construção em cada ciclo.

4.1.7 Arquitetura Baseada em Componentes

O software é dividido em módulos, como estratégia para facilitar o desenvolvimento. A modularização permite que seja criado um software com maior flexibilidade e adaptabilidade. Além disso, a reutilização e entendimento se tornam facilmente compreendidos.

4.2 Visão Geral da Metodologia

A metodologia proposta no desenvolvimento deste trabalho é composta por seis fases. O desenvolvimento de um companheiro virtual inteligente evolui através das fases definidas, que procuram auxiliar e facilitar o processo de construção dessa categoria software por parte de todos os papéis envolvidos [Torreão, 2004].

Cada fase da metodologia aqui apresentada possui algumas atividades que, por sua vez, apresentam passos a serem realizados para o uso efetivo da metodologia.

Os artefatos gerados durante o ciclo de desenvolvimento representam marcos no projeto e determinam o final de uma fase e início da fase seguinte na metodologia, com exceção dos DRI (Documento de Registro de Informações) (documento não formal usado para guiar a discussão e que serve de insumo para elaboração dos artefatos no desenvolvimento) que são gerados constantemente durante todo o desenvolvimento.

Para uma melhor visualização da metodologia, a Figura 11, apresenta todas as fases que a compõe. A modelagem da metodologia foi realizada através da linguagem SPEM 2.0 e a ferramenta EPF *Composer*.

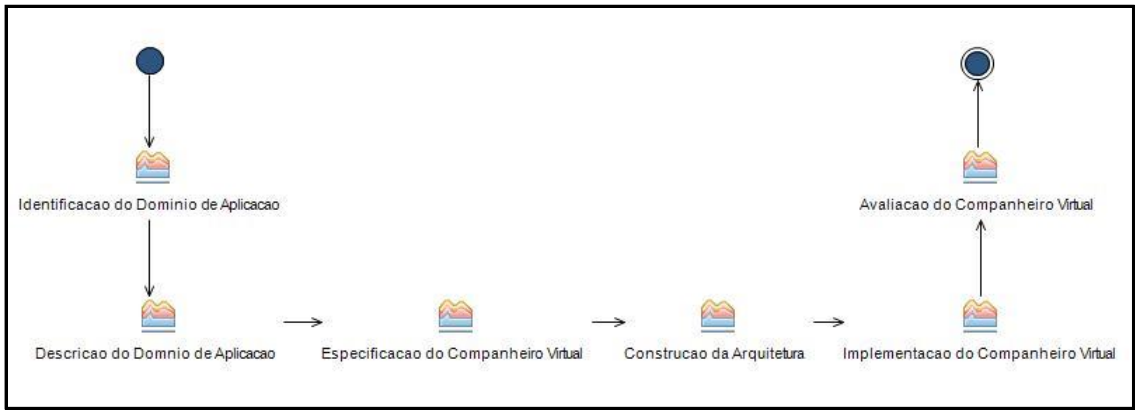

Figura 11 - Visão Geral da Metodologia

Como é possível perceber, a metodologia se inicia com a fase de identificação do domínio de aplicação em que o companheiro virtual inteligente será inserido. Em seguida, com base nas informações captadas, dá-se início a descrição desse domínio através da definição de uma ontologia. Tendo o domínio representado, os *stakeholders* passam a especificar o companheiro virtual, definindo todas as táticas e estratégias de interação inerentes a ele, bem

como suas ações e comportamentos. Com base nas informações do domínio, bem como do perfil do companheiro, dá-se início a construção da arquitetura que irá ditar como será estruturado o agente e a forma que a implementação deve agir. A próxima fase trata da avaliação, por parte dos usuários finais, do agente produzido, simulando cenários que descrevem situações reais de utilização.

Ao final das fases descritas na metodologia, considera-se que um ciclo de desenvolvimento chega ao fim. No entanto, a metodologia suporta a realização de outros ciclos, caso estes sejam necessários.

4.3 Papéis Envolvidos e Artefatos

Esta seção descreve os papéis envolvidos na execução das fases da metodologia, bem como os artefatos de entrada e saída que fazem parte do modelo.

A Tabela 2, apresenta os papéis envolvidos e artefatos pertinentes à metodologia.

Tabela 2 - Papéis Envolvidos e Artefatos da Metodologia

Papéis Envolvidos	Artefatos
• Especialista no Domínio • Gerente de Projeto • Desenvolvedores • Usuários Finais	• DDD (Documento de Descrição do Domínio) • EMD (Especificação de Modelagem de Domínio) • DEC (Documento de Especificação do Companheiro) • MA (Modelo Arquitetural) • Companheiro Virtual Construído • RAP (Relatório de Avaliação do Produto) • DRI (Documento de Registro de Informações)

Para um melhor entendimento de cada item listado na Tabela 2, os subitens abaixo apresentam a definição de cada papel envolvido no modelo e os artefatos de entrada e saída que são utilizados/gerados durante a execução das fases que compõe a metodologia.

4.3.1 Papéis Envolvidos

A execução de um projeto seguindo a metodologia proposta abrange quatro papéis que podem ou não atuar juntos, dependendo da fase.

- **Especialista no Domínio**

Papel responsável por acompanhar quase todas as fases da metodologia executadas durante a etapa de construção do software. Ele é responsável por repassar todo o conhecimento que possui acerca do domínio de aplicação para

uma melhor caracterização do problema, bem como atua como intermediador nas discussões durante esta atividade. De acordo com Arango (1994), os especialistas de domínio são fundamentais na especificação do domínio quando diz:

> *"Especialistas no domínio são excelentes fontes de conhecimento e, geralmente, são a única fonte de justificativas e explicações de "porque as coisas são como são". A memória de um especialista é, geralmente, repleta de preciosas informações históricas que não podem ser encontradas em outras fontes".*

Adicionalmente, o especialista no domínio participa da avaliação final do produto construído.

- **Gerente de Projeto**

Participa de todas as fases de definição do domínio e caracterização do perfil do companheiro virtual inteligente. Sua participação é fundamental para entender como funcionará o companheiro virtual, além de facilitar no controle de algumas variáveis de projeto, como riscos e mudanças.

- **Desenvolvedores**

É deles a responsabilidade efetiva de construir o companheiro virtual, embora participem de outras etapas de caracterização e descrição do problema. Na metodologia o desenvolvedor não é considerado apenas um implementador. A participação dos desenvolvedores nas fases de especificação do domínio e do CVI é fundamental visto que eles precisam obter o máximo de conhecimento possível para proceder com a implementação do CVI de acordo com a definição realizada.

- **Usuários Finais**

Atuam como auxílio em algumas fases da metodologia, proporcionando a visão real da utilização do produto após sua entrega ao cliente. Esta avaliação é importante, visto que ajustes e erros são corrigidos no ato da construção do companheiro e não só após a fase de transição, quando comumente estes usuários passam a interagir com o sistema.

4.3.2 Artefatos

Abaixo, temos descrições sucintas dos artefatos que são utilizados e/ou gerados durante a realização das fases descritas na metodologia.

- **DDD - Documento de Descrição do Domínio**

Documento descritivo que contém as características e problemas conhecidos do domínio de aplicação em que o agente será inserido, além de possuir as necessidades de interação identificadas como importantes durante reuniões realizadas para a caracterização do problema.

- **EMD - Especificação de Modelagem de Domínio**

Este documento é composto pelos conceitos do domínio, gerados a partir das características levantadas. Além disso, contém uma ontologia de domínio modelada, levando em consideração os conceitos estabelecidos.

- **DEC - Documento de Especificação do Companheiro**

É o documento que dita o objetivo e o comportamento do companheiro virtual diante de uma situação de interação. Contém as táticas e estratégias de interação que especificam o perfil do agente, mapeando os objetivos e ações para alcançá-los.

- **MA - Modelo Arquitetural**

É composto pela descrição dos módulos da arquitetura criada, bem como o modelo arquitetural desenvolvido e que servirá de base para a fase de implementação.

- **Companheiro Virtual Construído**

Artefato que trata do produto construído em si. Representa a saída da fase de implementação software que inclui o companheiro virtual. Caracterizado por ser um produto executável e que possa ser utilizado pelos usuários finais para avaliação.

- **RAP - Relatório de Avaliação do Produto**

Documento que afere a qualidade do companheiro virtual construído após rigorosa análise interativa por parte dos usuários finais e especialista no domínio.

- **DRI – Documento de Registro de Informações**

Documento não formal que auxilia em quase todas as fases definidas na metodologia. Sua utilização se dá basicamente nos momentos de reunião(ões), para o registro de pontos importantes que serão utilizados na geração dos artefatos formais a serem criados.

4.4 Fases da Metodologia

Esta seção apresenta as fases da metodologia que evoluem o desenvolvimento de um companheiro virtual inteligente. A distribuição das fases buscou retratar da melhor forma possível uma estratégia eficiente para a construção de CVIs, tentando sanar todos os problemas existentes e tornar o desenvolvimento menos complexo.

4.4.1 Identificação do Domínio de Aplicação

Uma estratégia peculiar da metodologia e que a difere de muitas outras existentes para construção de softwares é a descrição do domínio de aplicação, através da identificação e levantamento de características desse domínio, problemas conhecidos e vivenciados pelas pessoas que lidam com o domínio e detecção das necessidades de interação no domínio são levantados.

Nesta fase inicial, os *stakeholders* do processo de construção do companheiro virtual estão preocupados com a definição das características mais relevantes do problema em questão. Na Tabela 3, temos o quadro demonstrativo referente a esta etapa da metodologia.

Tabela 3 - Fase Identificação do Domínio de Aplicação

Identificação Domínio de Aplicação
Objetivo Realizar o levantamento das principais características e problemas-chave do domínio em que o companheiro virtual será inserido.
Atividades • Identificar Características do Domínio • Identificar Problemas do Domínio • Identificar Necessidades de Interação • Elaborar Documento de Descrição Domínio
Papéis Envolvidos Especialista no domínio, gerente de projeto, desenvolvedores e usuários finais.
Recursos Utilizados Brainstorming, entrevista e questionário.
Artefatos de Entrada

• N/A
Artefatos de Saída • DDD (Documento de Descrição do Domínio).

Esta atividade é realizada por todos os papéis envolvidos na metodologia (especialista no domínio, gerente de projeto, desenvolvedores e usuários finais), pois é a atividade inicial e que provê o embasamento do que será construído nas próximas fases.

Para a concretização desta fase são usadas algumas técnicas de levantamento e identificação de características relevantes, buscando conhecer a maioria dos problemas antes do início da construção do software.

Para a realização de atividades da metodologia que exigem encontros e reuniões é recomendada a utilização de *Brainstorming*, por entender que esta é uma técnica de fácil aplicação, que estimula os participantes a darem suas opiniões de forma criativa.

Pressman (2001) aponta a técnica Brainstorming como facilitadora de criatividade e que acende idéias para solução de problemas, quando diz:

> *"Facilita a criatividade e resolução de problemas através de encontros de livre geração de idéias numa situação de grupo. Tem por intenção alargar as fronteiras do espaço do problema dos participantes e obter soluções não convencionais (pelas manipulações de "quadros de experiência" individuais ou definições internas contextuais de eventos e situações) para permitir aceder a informação e idéias de outra forma indisponível por causa de seus "quadros" da situação atual".*

Vale ressaltar que para se buscar resultados realmente satisfatórios com a utilização do Brainstorming, é necessário que as reuniões iniciais sejam

coordenadas e dirigidas com um participante que execute o papel de intermediador da discussão.

De modo alternativo ou complementar, técnicas como entrevistas e questionários podem ser utilizados na realização das atividades especificadas nesta metodologia.

A Figura 12 apresenta o fluxo de atividades que deve ser obedecido para concretização desta fase.

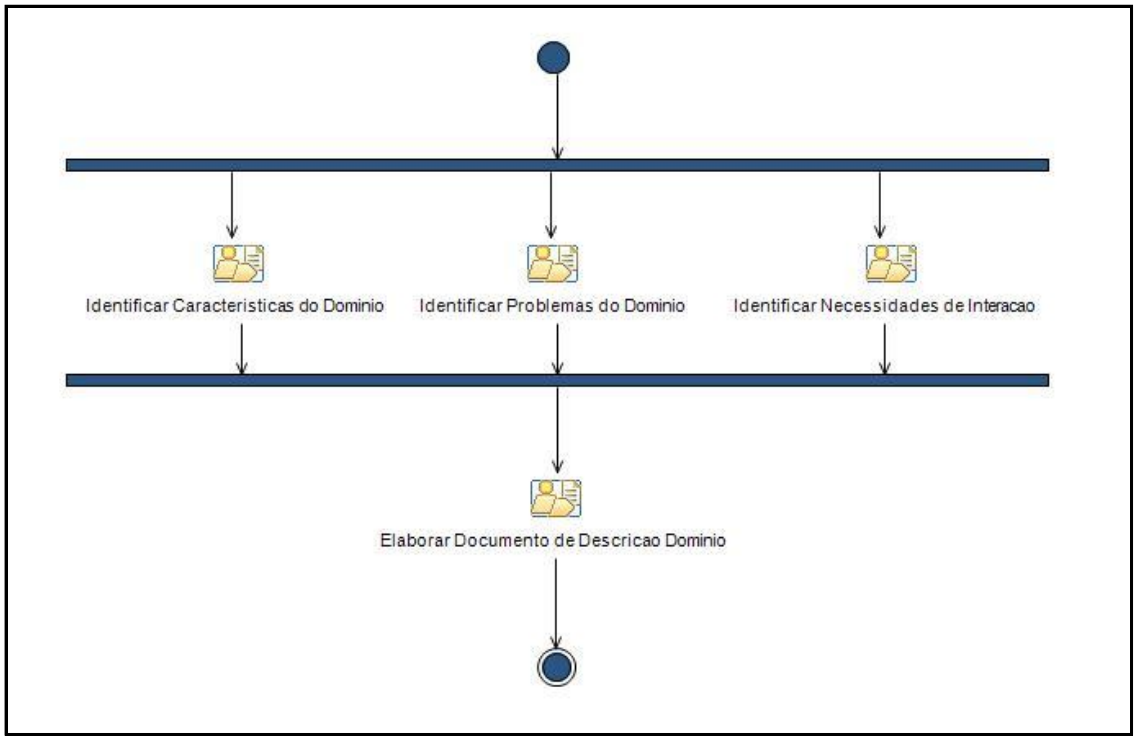

Figura 12 - Fase Identificação do Domínio de Aplicação - Fluxo de Atividades

Como é possível perceber na Figura 12, esta fase é composta por quatro atividades. As atividades "Identificar Características do Domínio", "Identificar Problemas do Domínio" e "Identificar Necessidades de Interação" podem ser realizadas simultaneamente ou não. No entanto, a atividade "Elaborar Documento de Descrição do Domínio" só poderá ser realizada com o término das atividades citadas anteriormente.

As descrições das atividades que fazem parte da fase de identificação do domínio estão relacionadas a seguir.

4.4.1.1 Identificar Características do Domínio

Para a realização desta atividade, o gerente de projeto agenda reunião(ões) para discussão do assunto com os demais envolvidos na fase. Através de brainstorming (e/ou outras técnicas listadas acima), todos os participantes expõem suas respectivas opiniões sobre quais seriam as principais características do domínio em que o companheiro virtual inteligente será inserido. Esta atividade é de suma importância para a construção do software, visto que é possível tomar conhecimento do domínio de aplicação que será trabalhado.

O papel do Especialista no Domínio é de fundamental importância neste momento, uma vez que é ele que detém maior conhecimento sobre o assunto e atua como intermediador nos momentos de discussão.

Todas as características externadas pelos participantes são armazenadas em um DRI que, por sua vez, é refinado através de consenso da maioria para inclusão na seção de Características do Domínio, contida no DDD (Documento de Descrição do Domínio).

4.4.1.2 Identificar Problemas do Domínio

A identificação dos problemas pode também ser realizada em conjunto com a caracterização do domínio (atividade anterior). A definição sobre se estas atividades serão realizadas em paralelo ou em seqüência fica a cargo do gerente de projetos e depende diretamente do custo e cronograma do projeto.

O brainstorming também é um recurso valioso para a boa execução desta atividade. Aliado a ele, podem ser usados questionários respondidos por usuários finais, bem como entrevistas dirigidas realizadas com pessoas que

vivenciaram problemas e que podem servir de insumo para descoberta de deficiências no domínio que o companheiro virtual poderá sanar.

O Especialista no Domínio, como facilitador, registra um DRI de problemas identificados e situações vivenciadas que serão posteriormente cadastradas na seção Problemas do Domínio do DDD (Documento de Descrição do Domínio).

4.4.1.3 Identificar Necessidades de Interação

A identificação das necessidades de interação entre o companheiro virtual e os usuários finais depende do tipo de domínio e da forma de interação desejada.

Em um domínio de educação à distância, por exemplo, é de fundamental importância que um agente possua não só um perfil de tutor, mas também uma personalidade motivadora para que o aluno se mantenha estimulado a persistir nos estudos.

Tendo as características e problemas do domínio definidos, fica mais claro ter uma visão inicial qual(is) tipo(s) de companheiro(s) virtual(is) é(são) necessário(s) para o domínio de aplicação.

As necessidades são identificadas em reuniões onde cada participante relata seu ponto vista, expõe experiências e itens inovadores, tomando como base os DRI gerados nas atividades anteriores.

Um DRI também é criado com as necessidades identificadas pelos stakeholders para ser incluído no DDD, na seção Necessidades de Interação.

4.4.1.4 Elaborar Documento de Descrição Domínio

Esta atividade diz respeito à consolidação de todas as informações coletadas nas atividades anteriores da fase.

Os DRI de características e problemas do domínio, além das necessidades de interação do companheiro x usuário(s) serão refinados e consensualmente aprovados em reunião pelos participantes da fase de identificação do domínio.

O preenchimento do DDD, feito por uma pessoa eleita durante uma das reuniões (geralmente, um desenvolvedor), é realizado com a inclusão de todas estas informações supracitadas e será o documento base e inicial no processo de desenvolvimento do companheiro virtual inteligente.

Este documento deve ser avaliado pelo cliente e, em seguida, divulgado entre todos os papéis envolvidos no processo de construção, para que haja a disseminação do conhecimento adquirido e para que problemas que venham a surgir ou dúvidas levantadas sejam sanados o mais rápido possível antes do início da próxima fase.

4.4.2 Descrição do Domínio de Aplicação

Tendo características e problemas do domínio levantados, faz-se necessária a formalização dos principais conceitos do domínio de aplicação em que o companheiro virtual será inserido.

Esta fase se assemelha a etapa de especificação de requisitos. Na Tabela 4, temos o quadro demonstrativo referente a esta etapa da metodologia.

Tabela 4 - Fase Descrição do Domínio de Aplicação

Descrição do Domínio de Aplicação
Objetivo Representar o domínio de aplicação, tomando como base todas as características levantadas.

Atividades • Representar Domínio • Elaborar Especificação de Modelagem de Domínio
Papéis Envolvidos Especialista no domínio e desenvolvedores.
Recursos Utilizados Frames, redes semânticas [Arango, 1994], ferramenta Prótegé [Chishman et al, 2004], etc.
Artefatos de Entrada • DDD (Documento de Descrição do Domínio).
Artefatos de Saída • EMD (Especificação de Modelagem de Domínio).

Esta atividade é realizada pelo especialista no domínio e os desenvolvedores. É uma fase fundamental para reforçar o entendimento acerca do domínio de aplicação, pois dará aos participantes uma visão modelada do problema.

Os conceitos identificados nesta fase resultarão em uma ontologia que representará o domínio. Escolheu-se essa forma de representação por se tratar a mais adequada para representar a base de conhecimento adquirida e por se tratar de um software dotado de certa inteligência artificial.

Segundo Guimarães (2002), o termo ontologia surgiu com a filosofia, no entanto no início da década de 90 passou a ser utilizado na área de computação, em inteligência artificial, com o objetivo de organizar grandes bases de conhecimento. Em seu contexto, ainda de acordo com Guimarães, a ontologia é responsável por definir a estruturação básica para a construção de uma base de

conhecimentos, objetivando assim facilitar sua compreensão e permitir seu compartilhamento.

De acordo com Souza (2003), no campo da Inteligência Artificial, pode-se descrever uma ontologia de um programa mediante a definição de um conjunto de termos de representação (os conceitos).

São diversas as definições dadas para Ontologia, mas todas vão ao encontro da mesma idéia. Souza apud Duineveld (2003) diz que:

> *"... as ontologias propõem-se a fornecer uma compreensão comum e compartilhada de algum domínio que possa ser entendido por pessoas e computadores".*

Já Gruber (1993) define:

> *"uma ontologia é uma especificação explícita de uma conceitualização".*

E para completar, tem-se a definição de Santi (2000) que define o seguinte conceito:

> *"... tem-se que ontologia é uma investigação dos conceitos que possibilita para as pessoas o conhecimento e determinação dos objetos reais".*

Para a concretização desta fase são usados alguns recursos para modelagem do conhecimento adquirido em forma de ontologia. A metodologia deixa a cargo da equipe de desenvolvimento decidir qual recurso e ferramenta será utilizado para modelagem da ontologia, no entanto sugere como melhor escolha a representação da ontologia através de frames, modelada a partir da ferramenta Prótegé.

De acordo com Chishman et al (2004):

> *"... a ferramenta Prótegé permite que a ontologia seja exportada em diversos formatos, tais como rdf schema, html, texto, entre outros, facilitando sua implementação".*

E ainda diz:

> *"Vale ressaltar que o Protégé é uma ferramenta Java de código aberto, que proporciona uma arquitetura extensível para a criação de ferramentas de bases de conhecimento personalizadas. Para o Protégé, ontologia é um modelo de um campo específico de conhecimento, os conceitos e seus atributos, bem como as relações entre estes conceitos".*

A Figura 13 apresenta o fluxo de atividades que deve ser obedecido para concretização desta fase.

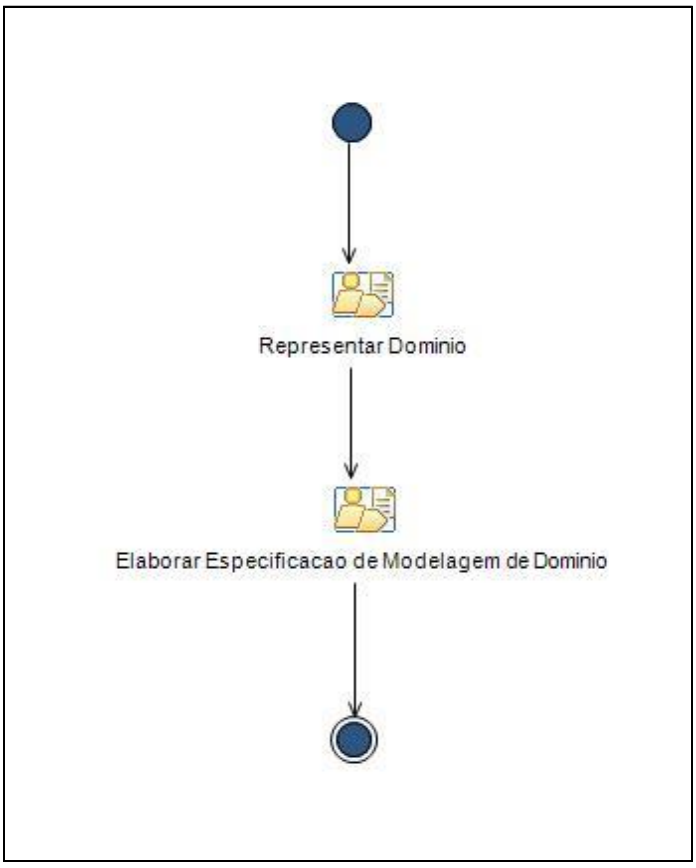

Figura 13 - Fase Descrição Domínio de Aplicação - Fluxo de Atividades

As duas atividades apresentadas são seqüenciais e resultam na geração do modelo de domínio.

As atividades da fase de descrição do domínio de aplicação estão mais detalhadas a seguir:

4.4.2.1 Representar Domínio

Depois das características levantadas, agora, é o momento de se definir os conceitos do domínio. Cada conceito levantado fará parte da ontologia do domínio a ser construída.

Assim como na fase anterior, a lista de conceitos identificados pode ser registrada em um DRI para facilitar o preenchimento do artefato construído ao final da fase. As decisões tomadas nesta fase são realizadas por meio de reuniões com os papéis envolvidos.

Diante dos conceitos listados, pode-se iniciar a construção da ontologia, estabelecendo os relacionamentos existentes entre os diversos conceitos, tomando como base também o DDD (Documento de Descrição do Domínio).

O próximo passo é a formalização do domínio, através da representação da ontologia criada, que pode ser realizada com a utilização de recursos como: frames, redes semânticas, etc. Como dito anteriormente, a escolha da forma de representação fica a cargo da equipe de desenvolvimento, mas também pode ser considerado o nível de familiaridade com determinado tipo de representação para a tomada de decisão.

Ao final desta atividade tem-se o modelo de ontologia do domínio.

4.4.2.2 Elaborar Especificação de Modelagem de Domínio

Esta atividade tem como objetivo criar um documento de especificação de modelagem de domínio, incluindo lista de conceitos levantados para o

domínio e a representação da ontologia descrita. O documento recebe o nome EMD (Especificação de Modelagem de Domínio), deve ser elaborado pelos desenvolvedores e servirá de base para a construção do companheiro virtual.

Para todas as informações coletadas nesta fase, há no EMD seções que contemplam a inclusão destas. Após a elaboração do EMD, este precisa ser validado com o cliente e, assim como o DDD, sua divulgação entre os *stakeholders* deve ser realizada.

4.4.3 Especificação do Companheiro Virtual

A próxima etapa após a modelagem do domínio é a especificação do companheiro virtual inteligente, da forma como ele irá auxiliar o usuário final, por meio da definição do agente que será criado, estabelecendo quais serão suas ações e comportamento para contribuir na interação, e estabelecendo quais suas estratégias e táticas. Abaixo, na Tabela 5, temos o quadro demonstrativo desta fase.

Tabela 5 - Fase Especificação do Companheiro Virtual

Especificação do Companheiro Virtual
Objetivo Definir o objetivo do agente, explicitando suas táticas e estratégias, bem como suas ações e comportamentos diante das diversas situações de interação.
Atividades • Definir Táticas e Estratégias de Interação • Definir Ações e Comportamentos • Elaborar Documento de Especificação do Companheiro Virtual
Papéis Envolvidos Especialista no domínio, gerente de projeto, desenvolvedores e usuários finais.
Recursos Utilizados

Brainstorming e entrevista.
Artefatos de Entrada • DDD (Documento de Descrição do Domínio). • EMD (Especificação de Modelagem de Domínio).
Artefatos de Saída • DEC (Documento de Especificação do Companheiro)

Como é possível identificar na Tabela 5, a fase de especificação do companheiro virtual possui como participantes o especialista no domínio, o gerente de projeto, a equipe de desenvolvimento e, como auxílio, os usuários finais. É nesta ocasião que será definido todo o comportamento e ações do companheiro virtual inteligente, mediante situações de interação, bem como as estratégias e táticas de interação que o CVI deverá possuir.

De acordo com Giraffa (1999) uma estratégia pedagógica está associada a como interagir. Se pensarmos tal conceito no âmbito da Educação, essa interação seria o ato de ensinar. Já as táticas são ações que devem ser executadas para o alcance efetivo da estratégia. Para Torreão (2005), no meio educacional, a escolha da estratégia pedagógica adequada representa a definição de tática para o aprendizado em um domínio específico, considerando informações do estudante.

Ainda conforme Torreão (2005) a escolha da estratégia determina quais táticas devem ser escolhidas. No exemplo citado por ela, para uma estratégia que objetiva o treinamento, uma tática possível seria a de estimular o estudante a repetir o procedimento correto várias vezes.

Torreão (2005) apresenta em seu trabalho algumas estratégias pedagógicas que podem ser utilizadas pelos agentes no momento da interação

com o usuário. Dentre várias podemos destacar três que são interessantes e podem dar uma visão do propósito de uma estratégia no agente:

- Estratégia Baseada em Casos (Case-based teaching) – Representam o domínio através de estudos de caso que incorporam as fórmulas e os princípios do domínio. O objetivo principal é fazer com que o usuário reflita sobre diferentes aspectos do domínio. Esta estratégia é utilizada pelo agente Adele, descrito no Capítulo 2.

- Estratégia Baseada em Explanação – A interação com o usuário se dá através de simulações de casos reais para que o usuário adquira o conhecimento. O companheiro virtual Lucy faz uso dessa estratégia para ensinar aos estudantes fenômenos físicos sobre satélites. É uma estratégia muito usada para aplicações de alto custo e/ou com risco de vida.

- Estratégia Baseada em Treinamento (Coaching) – Leva em consideração a filosofia de "aprender fazendo", onde os usuários constroem seu conhecimento a partir de experiências individuais. O agente Steve possui essa estratégia pedagógica para atuar com os aprendizes.

Além disso, Torreão (2005), aponta que um critério relevante para a escolha da estratégia é considerar o estilo de aprendizado ou absorção do conhecimento de cada indivíduo que interage com o CVI. Para isso, ela sugere o teste *Myers Briggs Type Indicator* (MBTI) para definir o tipo de personalidade do usuário.

Segundo Torreão (2005), o MBTI é baseado na teoria de tipos psicológicos de Carl Jung, fundador da prática e teoria da psicanálise moderna e foi criado pelas psicólogas Briggs e Myers, que expandiram o trabalho de Jung

e dotaram de uma aplicação prática. Elas determinaram que havia quatro escalas de preferências e dezesseis tipos distintos de personalidade.

O comportamento do CVI está diretamente ligado com a estratégia de interação escolhida. Como dito no Capítulo 2, item 2.2, o comportamento do CVI pode assumir, por exemplo, personalidade de guia, assistente ou facilitador.

A Figura 14 apresenta o fluxo de atividades que deve ser obedecido para concretização desta fase.

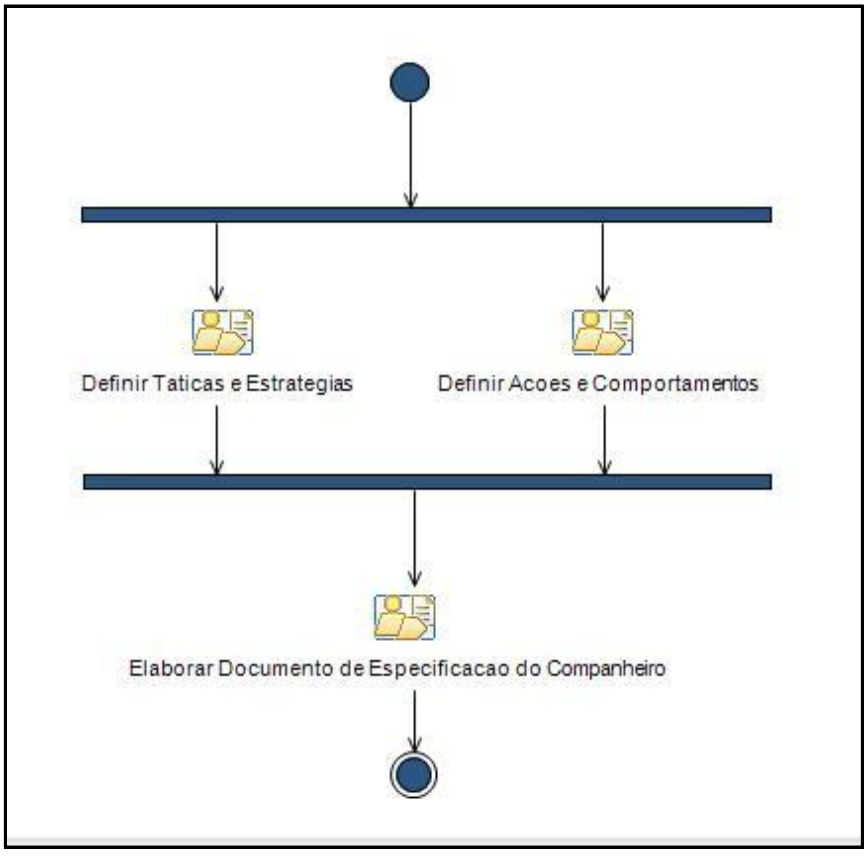

Figura 14 - Fase Especificação do Companheiro Virtual - Fluxo de Atividades

As atividades "Definir Táticas e Estratégias de Interação" e "Definir Ações e Comportamentos" da Figura 14 podem ser executadas em paralelo. Já a

atividade “Elaborar Documento de Especificação do Companheiro Virtual” só pode ser realizada após a finalização das duas anteriores.

A seguir é apresentado o detalhamento de cada atividade desta fase.

4.4.3.1 Definir Táticas e Estratégias de Interação

O recurso do brainstorming é fundamental nesta atividade, visto que é através de reuniões que os participantes especificarão quais estratégias e táticas o CVI deve possuir.

Estratégias de interação são definidas para o companheiro virtual, levando em consideração o domínio de aplicação, bem como o propósito de criação do agente.

É neste momento que são listadas as metas a serem atingidas quando do uso e funcionamento efetivo do companheiro virtual no domínio.

As informações desta fase são registradas no DRI, tentando mapear as estratégias e táticas definidas para o CVI.

4.4.3.2 Definir Ações e Comportamentos

Esta atividade está relacionada à definição de como o agente deve se comportar diante de interações com o usuário. Para isso, são definidas as ações e comportamentos que devem ser executados pelo agente em resposta a outras realizadas pelo usuário.

O levantamento destas informações também é registrado em um DRI que irá conter dados referentes às ações e comportamentos do CVI diante de situações de interação com o usuário.

4.4.3.3 Elaborar Documento de Especificação do Companheiro

Todas as informações levantadas na atividade anterior e listadas em formato de DRI servirão se insumo para a criação de um documento contendo

os objetivos, estratégias, táticas, ações e comportamentos do CVI, especificando, assim, o perfil do companheiro virtual mediante ações do usuário.

Ao final desta fase temos o artefato DEC (Documento de Especificação do Companheiro) elaborado.

4.4.4 Construção da Arquitetura

A arquitetura é a base para o bom entendimento do que será construído. É nesta fase que será definido como o software será estruturado. O domínio da arquitetura de software tem como foco a identificação de importantes propriedades e relacionamentos. A Tabela 6 apresenta o quadro demonstrativo desta fase.

Tabela 6 - Fase Construção da Arquitetura

Construção da Arquitetura
Objetivo Definir arquitetura do companheiro virtual.
Atividades • Definir Componentes • Construir Arquitetura
Papéis Envolvidos Desenvolvedores.
Recursos Utilizados Modelagem arquitetural.
Artefatos de Entrada • EMD (Especificação de Modelagem de Domínio). • DEC (Documento de Especificação do Companheiro)

Artefatos de Saída
• MA (Modelo Arquitetural).

Como é possível identificar na Tabela 6, a fase de construção da arquitetura é realizada apenas pela equipe de desenvolvimento do CVI. Os demais papéis envolvidos na metodologia não participam diretamente desta fase, por se tratar de um momento mais técnico.

O fluxo de atividades da fase está representado na Figura 15.

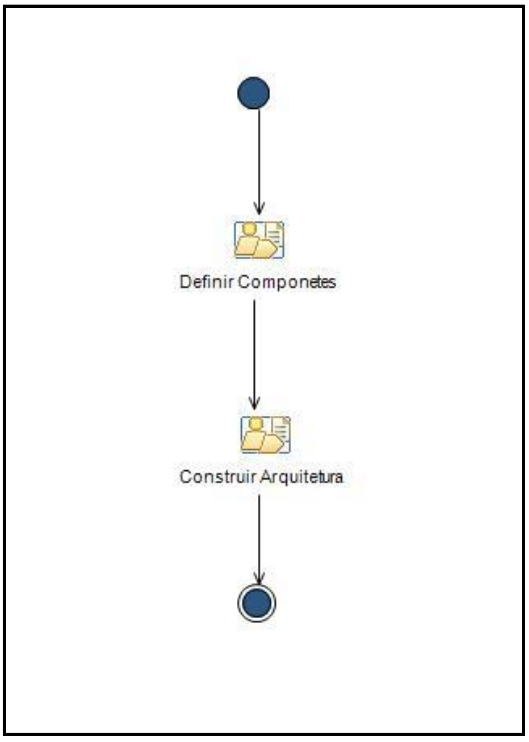

Figura 15 - Fase Construção da Arquitetura - Fluxo de Atividades

O fluxo de atividades, representado pela Figura 15, desta fase apresentam duas atividades que são detalhadas nos subitens a seguir:

4.4.4.1 Definir Componentes

Esta atividade depende basicamente do tipo de companheiro virtual que se quer desenvolver (reativo, deliberativo, etc). Para isso, são usados como base os artefatos: Especificação de Modelagem de Domínio e Documento de Especificação do Companheiro.

No entanto, como sugestão e padrão para qualquer domínio, o seguintes componentes podem ser definidos [Torreão, 2004]:

- Modelo de Usuário – armazena os dados do usuário, seu entendimento do domínio e suas ações no ambiente.
- Modelo de Conhecimento – trata da base de conhecimento do domínio, contendo todos os conceitos do domínio levantados na fase de sua modelagem.
- Modelo de Comportamento – trata do comportamento do companheiro virtual, usando as estratégias e táticas definidas na fase de Especificação do Companheiro Virtual.
- Modelo de Comunicação – trata de como se dará a comunicação entre o companheiro virtual e seus respectivos usuários finais.

Os demais componentes dependerão do tipo de companheiro virtual e poderão ser definidos caso a caso pela equipe de desenvolvimento.

4.4.4.2 Construir Arquitetura

Após definição de todos os componentes que irão compor a arquitetura, esta poderá ser consolidada e modelada a partir de alguma linguagem de modelagem arquitetural, definida a cargo dos desenvolvedores.

A descrição de cada componente da arquitetura e o modelo construído irão compor o documento gerado ao final desta a fase, o MA (Modelo Arquitetural).

4.4.5 Implementação do Companheiro Virtual

Nesta fase, o que foi definido na arquitetura deve ser implementado, com também, a animação do personagem, se houver. A seguir, na Tabela 7, temos o quadro demonstrativo desta fase.

Tabela 7 - Fase Implementação do Companheiro Virtual

Implementação do Companheiro Virtual
Objetivo Construir o companheiro virtual.
Atividades • Construir Companheiro
Papéis Envolvidos Desenvolvedores.
Recursos Utilizados Linguagem de programação, testes.
Artefatos de Entrada • EMD (Especificação de Modelagem de Domínio). • DEC (Documento de Especificação do Companheiro) • MA (Modelo Arquitetural)
Artefatos de Saída • Companheiro virtual construído (software).

Assim como a fase de construção da arquitetura do CVI, fase de impementação do CVI é de responsabilidade dos desenvolvedores.

Apesar da possível complexidade na implementação de CVIs, por se tratarem de Sistemas Especialistas [Torreão apud Turbam, 2005], quando o modelo arquitetural é bem construído, torna-se mais fácil executar a fase de implementação do agente.

A Figura 16 representa o fluxo de atividades da fase de Implementação do Companheiro Virtual.

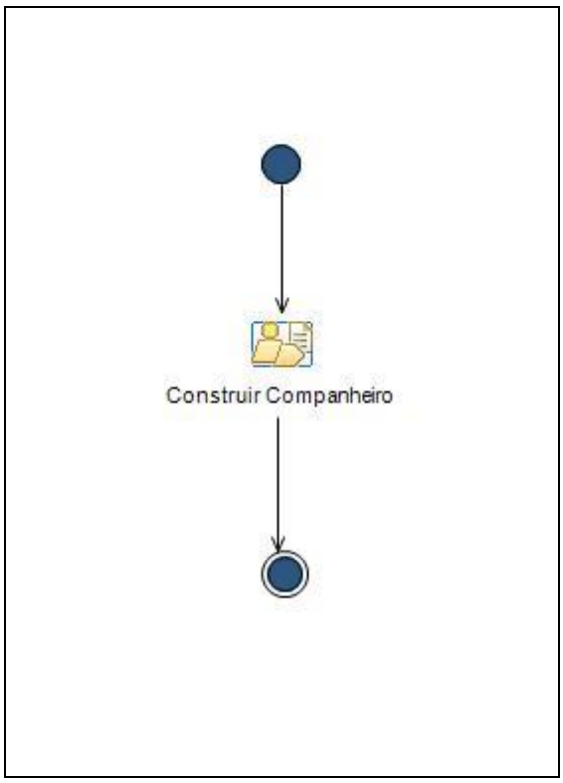

Figura 16 - Fase Implementação do Companheiro Virtual - Fluxo de Atividades

Esta fase é composta por apenas uma atividade, nomeada de "Construir Companheiro", que é descrita abaixo.

4.4.5.1 Construir Companheiro

Como boa prática para esta da fase, pode-se desenvolver um protótipo com um conjunto mínimo de funcionalidades para o sistema rodar [Torreão, 2004].

A partir deste protótipo, testes podem ser realizados para validação do que foi implementado. O protótipo vai sendo alimentado/evoluído com as demais funcionalidades até que se tenha o produto final, tomando como referência os seguintes artefatos: Especificação de Modelagem de Domínio, Documento de Especificação do Companheiro e Modelo Arquitetural.

Todos os módulos definidos na arquitetura serão implementados, incrementalmente, nesta atividade.

Ao final desta etapa tem-se efetivamente o companheiro virtual construído que passa a ser avaliado em simulações de situações de uso real.

4.4.6 Avaliação do Companheiro Virtual

Esta fase é bastante crítica no processo de desenvolvimento do companheiro virtual, visto que é neste momento em que se atesta o produto final em termos de interação com o usuário. Na Tabela 8, temos o quadro demonstrativo desta fase.

Tabela 8 - Fase Avaliação do Companheiro Virtual

Avaliação do Companheiro Virtual
Objetivo Avaliar o companheiro virtual.
Atividades • Realizar Avaliação • Elaborar Relatório de Avaliação

Papéis Envolvidos Especialista do domínio, desenvolvedores e usuários finais.
Recursos Utilizados Técnicas de avaliação.
Artefatos de Entrada • EMD (Especificação de Modelagem de Domínio). • DEC (Documento de Especificação do Companheiro) • Companheiro virtual construído.
Artefatos de Saída • RAP (Relatório de Avaliação do Produto)

Fará parte desta fase o especialista do domínio e usuários finais, interagindo com o CVI construído. Além disso, os desenvolvedores participarão como redatores de inconsistências identificadas nestas interações.

Técnicas de validação de usabilidade são fortemente indicadas para o tipo de avaliação que a fase se propõe a fazer [Ramos, 2004]. Dentre elas, podemos citar as seguintes técnicas:

- Prospectivas - são baseadas na opinião do usuário sobre a interação do sistema – para os usuários finais;
- Diagnósticas – são baseadas no conhecimento / competência do avaliador, tais como a Inspeção de Conformidade, Inspeção Cognitiva – ideais para o especialista no domínio;
- Definitivas (ou empíricas) - são baseadas na observação da interação, como Ensaios de Interação.

Além disso, é fundamental que se ateste a funcionalidade do CVI, por meio da realização de testes funcionais.

A Figura 17 representa o fluxo de atividades da fase de avaliação do companheiro virtual.

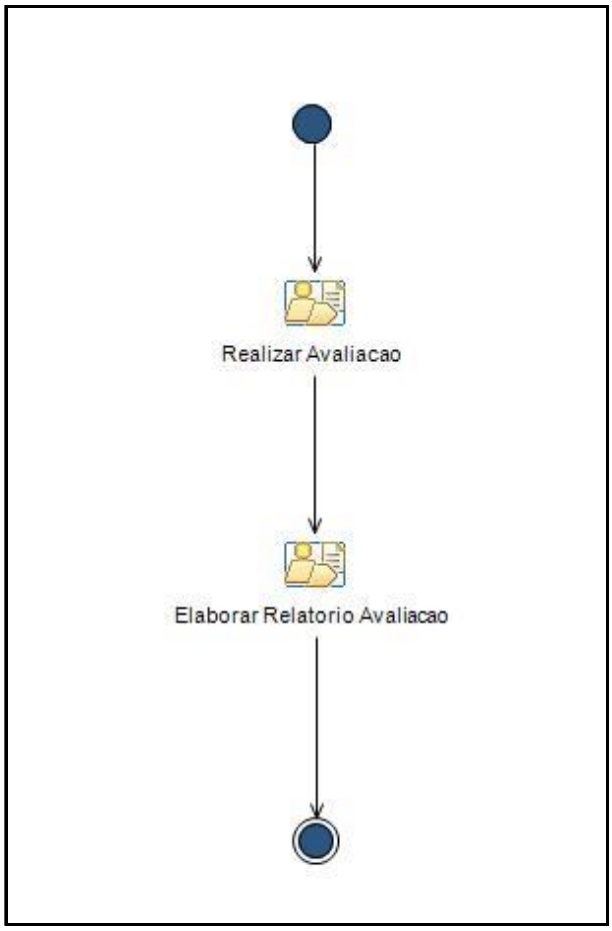

Figura 17 - Fase Avaliação do Companheiro Virtual - Fluxo de Atividades

As atividades desta fase estão descritas nas seções a seguir.

4.4.6.1 Realizar Avaliação

As diversas formas de comportamento do companheiro virtual diante de determinadas situações, definidas no início do processo, são verificadas e atestadas.

Os artefatos Especificação de Modelagem de Domínio, Documento de Especificação do Companheiro e o Companheiro Virtual implementado são usados como entrada nesta atividade.

Nesta atividade são definidas técnicas de validação da usabilidade, como as já citadas nesta fase, Inspeção de Conformidade, Inspeção Cognitiva, Avaliação Heurística, Ensaios de Interação [Ramos, 2004], entre outras, que serão utilizadas no processo de ateste do produto. Fazem parte deste momento o especialista no domínio de aplicação e os usuários finais. Além disso, é necessária a realização de testes funcionais, para garantir que as funcionalidades estão de acordo com o especificado.

Os usuários finais são as pessoas mais importantes para definição da qualidade do produto desenvolvido, pois são eles que vivenciam o uso do software e que poderão simular casos reais de utilização.

Todos os apontamentos feitos acerca da avaliação realizada podem ser registrados em um DRI.

Testes de desempenho, no que diz respeito ao tempo de resposta do companheiro virtual, também são bastante importantes nesta ocasião.

4.4.6.2 Elaborar Relatório de Avaliação

No processo de avaliação, falhas, inconsistências e deficiências são registradas no artefato RAP (Relatório de Avaliação do Produto), que armazenará todas as informações referentes ao processo de avaliação. Caso a atividade anterior tenha produzido DRI, este servirá como insumo para elaboração do artefato final da fase.

Ao final desta atividade, é possível atestar a qualidade do produto desenvolvido, com base nos problemas detectados e os relatos dos papéis envolvidos.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo apresenta uma revisão do trabalho desenvolvido nesta dissertação. Na Seção 5.1 é apresentada uma análise comparativa da metodologia proposta neste trabalho diante das metodologias orientadas a agentes Prometheus e MaSE. A Seção 5.2 aponta as contribuições desta pesquisa, através do alcance dos objetivos definidos. Na Seção 5.3, temos as limitações encontradas durante a pesquisa. Na Seção 5.4 são apresentadas algumas propostas de trabalhos futuros que poderão agregar valor a esta pesquisa.

5.1 Análise Comparativa

Em busca de ressaltar os benefícios trazidos com o desenvolvimento desta pesquisa, nesta seção é apresentada uma análise comparativa entre a metodologia proposta nesta dissertação e as metodologias orientadas a agentes apresentadas no Capítulo 3, a Prometheus e a MaSE.

Para que fosse possível demonstrar uma análise clara e objetiva, foram estabelecidos alguns critérios de comparação, adaptados a partir dos requisitos levantados na Seção 3.5 e identificados como relevantes na avaliação de metodologias, que serviram para indicar suas diferenças e/ou melhorias. A seguir, são apresentados os critérios elencados e utilizados para comparar as metodologias:

- Fases de desenvolvimento – critério que indica se a metodologia suporta todas as fases de desenvolvimento de um companheiro virtual inteligente [Dam, 2003].

- Rastreabilidade – critério que indica se a metodologia permite rastreamento entre seus artefatos e se existe uma ligação clara entre as fases da metodologia [Dam, 2003].

- Domínio de aplicação – critério que indica se a metodologia realiza estudo sobre o domínio de aplicação em que o companheiro virtual inteligente será inserido [Arango e Prieto-Diaz, 1991].

- Comunicação – critério que indica se a metodologia apresenta meios efetivos de interação entre os *stakeholders*, facilitando a disseminação do conhecimento [Nonemacher, 2003].

- Riscos – critério que indica se a metodologia aponta como preocupação o controle e mitigação dos riscos [Cooper et al, 2005] no desenvolvimento de companheiro virtual inteligente.

- Caracterização do CVI – critério que aponta se a metodologia define o agente em desenvolvimento, descrevendo suas ações e comportamentos diante de situações de interação com o usuário [Dam, 2003].

- Avaliação do CVI – critério que aponta se a metodologia realiza alguma avaliação atestando a qualidade do companheiro virtual construído [Torreão, 2005].

- Maturidade – avalia se a metodologia já possui determinada maturidade no desenvolvimento de companheiros virtuais inteligentes.

Vale ressaltar que a comparação utilizou também uma visão subjetiva do autor mediante estudo realizado para o desenvolvimento deste trabalho.

A Tabela 9 apresentada a seguir mostra o resultado da comparação entre as metodologias. Foram assumidos três valores possíveis na análise: a expressão "Atende" significa que a metodologia atende ao critério estabelecido, a expressão "Atende Parcialmente" significa que a metodologia atende parcialmente ao critério e a expressão "Não Atende" significa que a metodologia não atende ao critério definido.

Tabela 9 - Análise Comparativa entre Metodologias

Critério	**Prometheus**	**MaSE**	**Metodologia Proposta**
Caracterização do CVI	Atende	Atende	Atende
Avaliação do CVI	Não Atende	Não Atende	Atende
Domínio de aplicação	Não Atende	Não Atende	Atende
Rastreabilidade	Atende	Atende	Atende
Comunicação	Não Atende	Não Atende	Atende Parcialmente
Riscos	Não Atende	Não Atende	Atende Parcialmente
Fases de desenvolvimento	Atende Parcialmente	Atende Parcialmente	Atende Parcialmente
Maturidade	Atende Parcialmente	Atende Parcialmente	Não Atende

Na comparação realizada entre as metodologias, percebeu-se que todas atendem ao critério de definição do companheiro virtual inteligente, uma vez que apresentam a caracterização, definição de ações, comportamentos que devem ser tomados pelo CVI diante de situações de interação com o usuário. Os artefatos elaborados em cada metodologia contemplam tais informações de maneira satisfatória.

O critério que analisa se a metodologia contempla uma fase ou preconiza alguma prática de avaliação do companheiro virtual inteligente após a sua construção só é coberto pela metodologia proposta neste trabalho. A metodologia Prometheus não cita em momento algum a realização de avaliação do sistema, através de simulação de casos de interação com o usuário. A MaSE, por não cobrir todas as fases de desenvolvimento de um companheiro virtual inteligente, também não atende a este critério. A metodologia proposta na dissertação apresenta uma fase exclusiva para avaliação do CVI, composta por atividades e artefatos gerados ao final do desenvolvimento.

O critério de definição e descrição do domínio de aplicação também só é percebido na metodologia que foi resultado da dissertação, que possui atividades específicas para tal propósito. Como já citado anteriormente, é importante que se faça um estudo realizado sobre o domínio de aplicação, identificando seus problemas e características existentes, bem como as necessidades de interação com o usuário. Isso ajuda no momento da concepção do agente, na definição de seu comportamento e ações. As metodologias Prometheus e MaSE só trabalham com a definição do agente, não tendo atividades e práticas que caracterizem o domínio de aplicação.

Os artefatos de todas as metodologias possuem um encadeamento lógico que permite entender o fluxo de atividades realizado, atendendo ao critério de rastreabilidade. Os artefatos na metodologia proposta representam marcos que delimitam o término de uma fase e o início da seguinte, possibilitando uma clara percepção da mudança de fases no ciclo de desenvolvimento. As demais metodologias analisadas possuem comportamentos semelhantes.

A comunicação é fundamental em qualquer projeto de software para que problemas sejam sanados e deficiências sejam trabalhadas. Deste modo, o exercício da comunicação é primordial para a disseminação do conhecimento

durante a construção de um companheiro virtual inteligente, visto que os CVIs são considerados sistemas complexos. A metodologia proposta define algumas práticas a serem seguidas para auxiliar o desenvolvimento do CVI, dentre elas, é preconizada a prática de comunicação efetiva, por meio de reuniões que ocorrem nas fases do desenvolvimento, ajudando assim na propagação do conhecimento entre os *stakeholders*. Sendo assim, considera-se que, com relação ao critério comunicação, a metodologia proposta atende parcialmente, por entender que seria mais eficiente a definição de uma fase ou modelo que pudesse gerenciar de forma mais profunda a comunicação no projeto. As demais metodologias analisadas não preconizam prática ou modelo que facilite a comunicação no projeto, não atendendo ao critério de comunicação estabelecido.

Outra prática que a metodologia proposta preconiza é a de gerenciamento dos riscos na construção do CVI. A complexidade na construção desse tipo de sistema acentua a probabilidade de ocorrerem riscos no projeto. Por esta razão, este trabalho sugere que o gerente de projeto disponibilize mais atenção para os riscos que possam ocorrer e tomar ações para mitigá-los sempre que necessário. Por isso, a metodologia proposta atende parcialmente ao critério, por também entender que seria mais eficiente a definição de uma fase ou modelo que pudesse gerenciar de forma mais contundente os riscos no projeto. As metodologias Prometheus e MaSE não fazem referência à preocupação que se deve ter com os riscos, não satisfazendo ao critério definido para a análise comparativa.

O critério que indica se as metodologias abrangem todas as fases de desenvolvimento de um companheiro virtual inteligente é atendido parcialmente por todas elas. A metodologia proposta e a Prometheus não tratam de forma eficiente o gerenciamento de projeto. Já a metodologia MaSE não percorre todo ciclo de desenvolvimento.

O critério que avalia a maturidade da metodologia é baseado na experiência em aplicação desta no desenvolvimento de companheiros virtuais inteligentes. Apesar de muito usadas no desenvolvimento de agentes, foi considerado que as metodologias Prometheus e MaSE atendem parcialmente a este critério, pois estas estão em constante evolução a têm muito a ser melhoradas. Já a metodologia proposta, recebeu o conceito "Não Atende" por se tratar de uma primeira versão e nunca ter sido avaliada em um caso real de desenvolvimento de CVI.

Com base no que foi exposto, podemos considerar que a metodologia proposta atende a uma boa parte dos critérios estabelecidos e, deste modo, possui alguns diferenciais não identificados nas outras metodologias estudadas e que facilitam o desenvolvimento de CVIs, como a definição de práticas e atividades que trabalham com o domínio de aplicação do companheiro virtual inteligente.

5.2 Contribuições

Desenvolver sistemas não é uma tarefa trivial e elaborá-los sob uma perspectiva informal, com ausência de regras estabelecidas na Engenharia de Software, pode gerar diversos prejuízos para as organizações. Essa forma de desenvolvimento permite que os projetos acabem por ter a qualidade comprometida, prazos apertados e altos custos.

Por esta razão é importante que sistemas sejam construídos através da utilização de metodologias de desenvolvimento. O uso correto de metodologias no desenvolvimento de software traz diversos benefícios, como o aumento da qualidade, a independência de indivíduos, a facilidade de manutenção e o aumento da produtividade.

Esta mesma orientação vale para o desenvolvimento de companheiros virtuais inteligentes. No entanto, a pesquisa na elaboração de métodos,

processos e técnicas que norteiem o desenvolvimento destes agentes ainda é insuficiente, embora esteja crescendo progressivamente. Este crescimento se deve à necessidade de se criar companheiros virtuais inteligentes para diversos domínios de aplicação, facilitando e melhorando a interação dos ambientes com os usuários finais. Dessa maneira, é fundamental a existência de metodologias padronizadas que auxiliem na construção sistemática de tais aplicações.

Com o objetivo de tentar atender essa necessidade, a principal contribuição deste trabalho é a proposta de uma metodologia para o desenvolvimento de companheiros virtuais inteligentes. Para um melhor entendimento, a seguir, temos detalhadamente as contribuições deste trabalho.

- Estudo sobre ambientes virtuais, identificando seus principais problemas;
- Estudo sobre companheiros virtuais inteligentes e quais os benefícios trazidos para a interação homem x computador quando estes são integrados a ambientes virtuais;
- Revisão sobre metodologias tradicionais e ágeis de desenvolvimento de software, bem como duas metodologias orientadas a agentes, a Prometheus e MaSE.
- Proposta de uma metodologia de desenvolvimento de companheiros virtuais inteligentes, elaborada a partir da revisão realizada nas metodologias de desenvolvimento tradicionais, ágeis e orientadas a agentes, destacando boas práticas, fases, atividades, papéis e artefatos criados.
- Análise comparativa entre a metodologia proposta e as metodologias orientadas a agentes Prometheus e MaSE, utilizando critérios de comparação definidos previamente.

A metodologia proposta está fundamentada no estudo de metodologias tradicionais e ágeis de desenvolvimento, que instigaram a idéia de definição de boas práticas a serem seguidas no desenvolvimento de companheiros virtuais inteligentes. Além disso, as metodologias orientadas a agentes, apresentadas no Capítulo 3, serviram de suporte para a definição das fases, atividades e artefatos descritos na metodologia proposta.

A Tabela 10 apresenta um resumo da metodologia proposta nesta dissertação.

Tabela 10 - Resumo da Metodologia Proposta

<table>
<tr><th colspan="3">Boas Práticas</th></tr>
<tr><td colspan="3">• Desenvolvimento Iterativo e Incremental
• Comunicação Eficiente
• Importância dos Testes
• Gerenciamento dos Riscos na Construção
• Caracterização e Modelagem Visual do Domínio
• Controle de Mudanças
• Arquitetura Baseada em Componentes</td></tr>
<tr><th>Fases</th><th>Atividades</th><th>Artefatos</th></tr>
<tr><td>Identificação do Domínio de Aplicação</td><td>Identificar Características do Domínio
Identificar Problemas do Domínio
Identificar Necessidades de Interação
Elaborar Documento de Descrição Domínio</td><td>DDD (Documento de Descrição do Domínio)</td></tr>
<tr><td>Descrição do Domínio de Aplicação</td><td>Representar Domínio
Elaborar Especificação de</td><td>EMD (Especificação de Modelagem de Domínio)</td></tr>
</table>

	Modelagem de Domínio	
Especificação do Companheiro Virtual	Definir Táticas e Estratégias de Interação Definir Ações e Comportamentos Elaborar Documento de Especificação do Companheiro Virtual	DEC (Documento de Especificação do Companheiro)
Construção da Arquitetura	Definir Componentes Construir Arquitetura	MA (Modelo Arquitetural)
Implementação do Companheiro Virtual	Construir Companheiro	Companheiro virtual construído
Avaliação do Companheiro Virtual	Realizar Avaliação Elaborar Relatório de Avaliação	RAP (Relatório de Avaliação do Produto)

5.3 Limitações

Algumas limitações impactaram no fluxo normal de execução do trabalho, fazendo com que fossem tomadas algumas medidas alternativas para que o resultado final fosse obtido. A seguir, são apresentadas as limitações que essa pesquisa enfrentou.

- As informações obtidas sobre as metodologias usadas no desenvolvimento dos agentes Adele e Lucy foram insuficientes para contribuírem de forma mais expressiva na elaboração da metodologia proposta. O material disponível na rede sobre tal as metodologias supracitadas ainda é escasso. Para amenizar esta falta, além de metodologias tradicionais e ágeis, foram pesquisadas e analisadas as metodologias orientadas a agentes Prometheus e MaSE, para serviram de suporte na definição da metodologia proposta.

- A falta de um estudo de caso real de aplicação, impediu que houvesse uma avaliação mais profunda da metodologia. Na tentativa de suprir tal necessidade, foi realizada uma análise comparativa entre a metodologia proposta e as metodologias orientadas a agentes Prometheus e MaSE.

5.4 Trabalhos Futuros

Certamente este trabalho agregou valor à pesquisa em busca de métodos para facilitar o desenvolvimento de companheiros virtuais inteligentes, bem como acrescentou mais informações acerca da engenharia de software orientada a agentes. No entanto, sabemos que, por limitações, dificuldades e/ou deficiências, não foi possível incrementar o trabalho com outras propostas interessantes. Deste modo, estas propostas ficam como sugestões a serem realizadas no futuro:

- Estudos de caso: neste trabalho, não houve aplicação da metodologia em estudo de caso, tanto pelo pouco tempo restante para finalização do trabalho como pela inexistência de estudo de caso real de aplicação. No entanto, é de suma importância utilizar a metodologia em diversas aplicações de desenvolvimento, para uma efetiva validação desta;

- Avaliação crítica: avaliar companheiros virtuais inteligentes não é uma tarefa fácil, devido à complexidade e abstração que eles possuem. Deste modo, uma pesquisa detalhada com relação a técnicas de avaliação de companheiros virtuais inteligentes seria primordial para que problemas fossem detectados antes da entrega ao cliente;

- Refinar artefatos: o refinamento/aprimoramento dos artefatos é interessante para que estes possam refletir o mais fielmente possível o

companheiro virtual inteligente. Uma sugestão seria incluir mais modelos visuais nos artefatos;

- Criação de atividade: a criação de uma atividade para acompanhamento de variáveis no desenvolvimento do projeto (como riscos, mudanças, etc) talvez fosse mais interessante do que considerar boas práticas para o gerenciamento destes problemas, como é feito na metodologia proposta;

- Desenvolvimento de ferramenta: a criação de uma ferramenta que suporte todo o ciclo de vida da metodologia, dando visibilidade a todas fases e atividades, permitindo a elaboração rastreabilidade de todos os artefatos propostos, seria um grande diferencial e facilitaria sua utilização.

REFERÊNCIAS

Agile Manifesto, Disponível em: http://agilemanifesto.org/, 2004, acessado em 03/02/2009.

Ambler, S., "A Manager's Introduction to The Rational Unified Process (RUP)", Ambysoft, 2005.

Arango, G., Prieto-Diaz, R., "Domain analysis concepts and research direction", IEEE Computer Society Press, 1991.

Arango, G., "Software Reusability - Domain analysis methods", Ellis HorWood, 1994.

Arbex, D. e Bittencourt, D.,"Estratégias para o Desenvolvimento de um Ambiente Virtual de Aprendizagem: Um Estudo de Caso realizado na Unisul Virtual", Associação Brasileira de Educação à Distância", Revista Brasileira de Aprendizagem Aberta e a Distância, São Paulo, 2007.

Avison, D., Fitzgerald, G. "Information Systems Development – Methodologies, Techniques and Tools", 2nd Edition, London, The Alden Press, 2000.

Assunção, B., Lopes, E. e Rissoli, V. "Sistema Tutor Inteligente integrado a Monitoria Estudantil para elaboração de um Assistente Virtual de Ensino Inteligente", XXVIII Congresso da SBC, Workshop sobre Informática na Escola, Belém, Brasil, 2008.

Barros, R., "Análise de Metodologias de Desenvolvimento de Software aplicadas ao Desenvolvimento de Jogos Eletrônicos", Trabalho de Graduação, Universidade Federal de Pernambuco, Recife, 2007.

Beck, K., "Programação Extrema Explicada", Bookman, 1999.

Bergenti, F. e Poggi, A., "Exploiting UML in the Design of Multi-Agent Systems", In Engineering Societies in the Agent World, First International Workshop, ESAW 2000, Lecture Notes in Computer Science, Vol. 1972, Germany, 2000.

Black, R., "Pragmatic Software Testing: Becoming An Effective And Efficient Test Professional", John Wiley and Sons LTD, 2007.

Bocca, E., Jaques, P. e Vicari, R. "Modelagem e Implementação da Interface para Apresentação de Comportamentos Animados e Emotivos de um Agente Pedagógico Animado", Instituto de Informática, PPGC, UFRGS, Porto Alegre, Brasil, 2003.

Brooks, F. "No Silver Bullet: Essence and Accidents of Software Engineering", Proc. IFIP, IEEE CS Press, 1987.

Canuto, E. "Victor-P: Um CVA Chatterbot com Personalidade", Trabalho de Graduação, Centro de Informática, Universidade Federal de Pernambuco, Recife, Brasil, 2005.

Castro, J., Alencar, F. e Silva, C., "Livro das Jornadas de Atualização em Informática", Capítulo 5 - Engenharia de Software Orientada a Agentes, Campo Grande, Brasil, 2006.

Chishman, R., Alves, I. e Bertoldi, A., "O Conhecimento Semântico Representado em Ontologias Aplicadas à Busca e Extração de Informação na Web", UNISINOS, 2004.

Cockburn, A. e Highsmith, J. "Agile Software Development: The Business of Innovation", IEEE Computer, 2001.

Chou, C., Chan, T-W. e Lin, C. "Redefining The Learning Companion: The Past, Present, And Future Of Educacional Agents", Computers & Education, 255-269, 2003.

Cooper, D., Grey, S., Raymond, G. e Walker, P., "Project Risk Management Guidelines", John Wiley & Sons Ltda, 2005.

Correia, A. "O Vivo no Mundo Digital", Trabalho de Graduação, Centro de Artes e Comunicação, Universidade Federal de Pernambuco, Recife, Brasil, 2004.

Costa, L. e Franco, S. "Ambientes Virtuais de Aprendizagem e suas Possibilidades Construtivistas", Brasil, 2005, http://www.cinted.ufrgs.br/renote/maio2005/artigos/ a25_ambientesvirtuais.pdf.

Dam, K., "Evaluating and Comparing Agent-Oriented Software Engineering Methodologies", Master Thesis of Applied Science in Information Technology, RMIT University, Australia, 2003.

Dário, C., "Uma Metodologia Unificada para o Desenvolvimento de Sistemas Orientados a Agentes", Dissertação de Mestrado, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2005.

Dillenbourg, P. "Virtual Learning Environment", 2003, http://tecfa.unige.ch/tecfa/publicat/dil-papers-2/Dil.7.5.18.pdf.

Elliott, C., Rickel, J. e Lester, J. "Integrating affective computing into animated tutoring agents", In: Ijcai Workshop on Animated Interface Agents: Making them Intelligent, Nagoya, Japão, 1997.

Ferreira, D., Costa, F., et al. "SCRUM Um Modelo Ágil para Gestão de Projectos de Software", Universidade do Porto, Porto, Portugal, 2004.

Franklin S., Graesser, A. "Is it a Agent, or just a Program? A Taxonomy for Autonomous Agents", In: Proceedings of the 3rd international workshop on agent theories, Springer-Verlag, 1996.

Garcindo, L. "Uma Abordagem sobre o Uso da Hipermídia Adaptativa em Ambientes Virtuais de Aprendizagem", Programa de Pós-Graduação em Engenharia de Produção, Tese de Doutorado, Universidade Federal de Santa Catarina, Brasil, 2002.

Gavidia, J. e Andrade, L. "Sistemas Tutores Inteligentes", COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2003.

Geyer, C., Ferrari, D., et al. "SEMEAI - SistEma Multiagente de Ensino e Aprendizagem na Internet", In: XII SBIE 2001 – Simpósio Brasileiro de Informática na Educação, Vitória, ES, Brasil, 2001.

Gilb, T., "Principles of Software Engineering Management", Addison-Wesley, 1988.

Giraffa, L. "Fundamentos de Sistemas Tutores Inteligentes", CPGCC, UFRGS, Porto Alegre, Brasil, 1998.

Giraffa, L. "Uma Arquitetura de Tutor Utilizando Estados Mentais", Tese de Doutorado, CPGCC, UFRGS, Porto Alegre, Brasil, 1999.

Goodman, B., Soller, A., Linton, F. e Gaimari R. "Encouraging Student Reflection and Articulation using a Learning Companion", In: Proceedings

of the 8th World Conference on Artificial Intelligence in Education (AI-ED 97), Kobe, Japan, 151-158, 1997, http://www.cscl-research.com/Dr/documents/LuCy-AI-ED97.doc.

Goulart, R. e Giraffa, L. "Arquiteturas de Sistemas Tutores Inteligentes", Technical Report Series, Number 011, Faculdade de Informática, UFRGS, Porto Alegre, Brasil, 2001.

Gruber, T., "A translation approach to portable ontologies", In: Knowledge Acquisition, 1993.

Guimarães, F., "Utilização de ontologias no domínio B2C", Mestrado em Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2002.

Hara, N. e Kling, R. "Students' Distress with a Web-based Distance Education Course: An Ethnographic Study of Participants' Experiences", 2000, http://www.slis.indiana.edu/CSI/WP/wp00-01B.html.

Hayes, S. "An Introduction to Extreme Programming", 2001.

Hayes-Roth B., Lalanda P., Morignot P., Pfleger, K. e Blabanovic, M., "Plans and Behavior in Intelligent Agents", Technical Report KSL-93-43, Knowledge Systems Laboratory, Computer Science Department, Stanford University, 1993.

Jackson, M., "Software Requirements and Specifications", Addison Wesley, 1995.

Jacobson, I., Booch, G. e Rumbaugh, J., "The Unified Software Development Process", Addison Wesley, 1998.

Jaques, P. et al. "Interação com Agentes Pedagógicos Animados: Um Estudo Comparativo". In: Workshop sobre Fatores Humanos em Sistemas Computacionais, Florianópolis, SC, Brasil, 2001.

Jennings, N., "Cooperation in Industrial Multi-agent Systems", World Scientific, 1994.

Johnson, W., Shaw, E. e Ganeshan, R. "Pedagogical Agents on the Web", ITS'98 - Conference on Intelligent Tutoring Systems Workshop on Pedagogical Agents and Workshop on Intelligent Tutoring Systems on the Web, 1998.

Johnson, W., Rickel, J. e Lester, J. "Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learing Environments", International Journal of Artificial Intelligence in Education, 2000.

Kampff, A., Lira, A., Reitz, D., Gomes, F., Fonseca, L., Machado, N. e Bercht, M. "Relação entre o Perfil do Usuário e a Escolha do Perfil do Tutor", 2005, http://www.cinted.ufrgs.br/renote/maio2005/artigos/a38_perfilusuario_revisado.pdf.

Kenski, V., "Educação e tecnologias: O novo ritmo da informação", Campinas, Papirus, 2007.

Leitão, D. "Um Chatterbot para um ambiente de ensino de gerência de projetos", Trabalho de Graduação, Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, Recife, Brasil, 2003.

Lévy, P. "As Tecnologias da Informação: O futuro do pensamento na era da Informática" Rio de Janeiro, Brasil, 1993.

Litto, F. e Formiga, M., "Educação à Distância: O estado da arte", Pearson Education, 1ª edição, 2009.

Loi, L., "Comparação e Avaliação entre o Processo RUP de Desenvolvimento de Software e a Metodologia Extreme Programming", Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2007.

Maes, P. "Agents that Reduce Work and Information Overload", Communications of the ACM, Vol. 37, No. 7, 1994.

Martins, P., "Modelo Cascata ou Clássico", Curso de Engenharia da Computação, Universidade do Algarve, Portugal, 2003.

Mendonça, A., Ribeiro, E. e Mendonça, G. "A Importância dos Ambientes Virtuais de Aprendizagem na EAD", CEFET-GO, Brasil, 2007.

Moore, M., "Educação à Distância: Uma Visão Integrada", Editora Thomson, 1ª edição, Brasil, 2007.

Moran, J. "Ensino e Aprendizagem Inovadores com Tecnologias", Revista Informática na Educação: Teoria & Prática, PGIE, UFRGS, Porto Alegre, Brasil, 137 – 144, 2000.

Nonemacher, M., "Comparação de Avaliação entre o Processo RUP de Desenvolvimento de Software e a Metodologia Extreme Programming", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis, 2003.

Nwana, H. "Software Agents: An Overview". Knowledge Engineering Review, v. 11, n. 3, 1-40, 1996.

Padgham, L., e Winikoff, M., "Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents", In Proceedings of the Workshop on Agent-Oriented Methodologies at Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), Seattle, 2002.

Padgham, L e Winikoff, M., "Prometheus: A Practical Agent-Oriented Methodology", Chapter 5 in Agent-Methodologies, edited by B. Henderson-Sellers and P.Giorgini, Idea Group, 2005.

Pereira, A., "AVA: Ambientes Virtuais de Aprendizagem em Diferentes Contextos", Ciência Moderna, 1ª edição, Brasil, 2007.

Perotto, M. "Estudo de uma Metodologia Orientada a Agentes: Um Protótipo para um Ambiente Virtual", Programa de Pós-Graduação em Engenharia de Produção, Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2002.

Petry, P. "Um Sistema para o Ensino e Aprendizagem de Algoritmos Utilizando um Companheiro de Aprendizagem Colaborativo", Dissertação de Mestrado, Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2005.

Pressman, R. "Engenharia de Software", Rio de Janeiro, McGraw-Hill,2001.

Pressman, R. "Engenharia de Software", 5ª Edição, Rio de Janeiro, McGraw-Hill, 2002.

Ramos, E., "Avaliação de Usabilidade de IHC", Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2004.

Rich, E. e Knight, K. "Inteligência Artificial", São Paulo, Makron Books, 1993.

Rodrigues, M., Novais, P. e Santos, M. "Future challenges in intelligent tutoring systems - a framework", In m-ICTE2005 3rd International Conference on

Multimedia and Information and Communication Technologies in Education, 2005.

RUP, "Rational Unified Process: Best Practices for Software Development Teams", Rational Software White Paper, TP026B, Rev 11/01, 2001.

RUP, "Rational Unified Process", 2003, Disponível em <http://www.wthreex.com/rup/>.

Santarosa C. e Sloczinski, H., "Aprendizagem coletiva em curso mediado pela web", Anais do VII Congresso Iberoamericano de Informática Educativa, México, 2004.

Santi, S., "Ontologias – Abordagens de Construção e Aplicações", UFRGS, 2000.

Santos, L., Bergmann, U. e Choren, R. "Uma Proposta para Levantamento de Requisitos e Derivação de Modelos de Análise para Sistemas Multi-Agentes", Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2004.

Schlemmer, E., "Metodologias para educação à distância no contexto da formação de comunidades virtuais de aprendizagem", In: Ambientes virtuais de aprendizagem, Porto Alegre, Artmed, 2005.

Schwaber, K., "Agile Project Management with Scrum", Microsoft Press, 1ª edição, 2004.

Shoham, Y. "Agent oriented programming", Artificial Intelligence, v. 60, n.1, 51-92, 1993.

Silva, A., Brito, S., Favero, E., Domínguez, A., Tavares, O. e Francês, C. "Uma arquitetura para desenvolvimento de ambientes interativos de aprendizagem baseado em agentes, componentes e framework", In: XIV Simpósio Brasileiro de Informática na Educação, IM/UFRJ, Rio de Janeiro, Brasil, 2003.

Soares, M., "Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software", Universidade Presidente Antônio Carlos, 2006.

Souza, L., "Regras de Raciocínio Aplicadas a Ontologias por Meio de Sistema Multiagente para Apoio a Decisões Organizacionais", Pontifícia Universidade Católica do Paraná, Mestrado em Informática, Curitiba, 2003.

Sommerville, I., “Software Engineering”, 6th Edition, Addison-Wesley, Reading Massachusetts, 2000.

Tartuce, T. J. A., “Métodos de pesquisa”, Fortaleza, UNICE – Ensino Superior, 2006.

Torreão, P. “Project Management Learning Environment: Ambiente Inteligente de Aprendizado para Educação em Gerenciamento de Projetos”, Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Recife, Brasil, 2005.

Urretavizcaya, L. “ Sistemas Inteligentes en el âmbito de la educación”, Revista Iberoamericana de Inteligência Artificial, n. 12, 2001.

Wake, W., “Extreme Programming Explored”, 2000.

Weiss, G. “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”, MIT Press, 2000.

Wilges, B., Lucas, J. e Silveira, R. “Um Agente Pedagógico Animado Integrado a um Ambiente de Ensino a Distância”, CINTED, UFRGS, Porto Alegre, Brasil, 2004.

Wood, M. e DeLoach, S., “An Overview of the Multiagent Systems Engineering Methodology”, In Proceedings of the First International Workshop on Agent-Oriented Software Engineering, Lecture Notes in Computer Science, Vol. 1957, 2001.

Wooldridge, M. e Jennings, N. “Intelligent agents: theory and practice”, The Knowledge Engineering Review, 1995.

Wooldridge, M. “An Introduction to Multiagent Systems”, John Wiley and Sons, England, 2002.

Yourdon, E. “Declínio e Queda dos Analistas e Programadores”, São Paulo, Makron Books, 1995.

APÊNDICE A – TEMPLATES DA METODOLOGIA PROPOSTA

A seguir são apresentados os *templates* dos documentos que constituem a metodologia proposta nesta dissertação.

Documento de Descrição do Domínio

Histórico de Alterações

Data	Versão	Descrição	Autor

1. Introdução

*[A introdução do **Documento de Descrição do Domínio (DDD)** deve fornecer uma visão geral de todo o documento.]*

2. Descrição Geral do Domínio de Aplicação

[Especifique de um modo geral o domínio de aplicação que será trabalhado.]

3. Características do Domínio de Aplicação

[Nesta seção devem ser incluídas todas as características do domínio de aplicação que foram especificadas durante reunião que contou com todos os papéis definidos na metodologia. O DRI gerado ajuda no preenchimento desta seção.]

4. Problemas do Domínio de Aplicação

[Nesta seção devem ser incluídos todos os problemas conhecidos do domínio de aplicação que foram levantados durante reunião que contou com todos os papéis definidos na metodologia. O DRI gerado ajuda no preenchimento desta seção.]

5. Necessidade de Interação

[Nesta seção devem ser incluídas todas as necessidades de interação entre companheiro virtual e usuários finais que são fundamentais para domínio de aplicação e que foram levantadas durante reunião que contou com todos os papéis definidos na metodologia. O DRI gerado ajuda no preenchimento desta seção.]

Especificação de Modelagem do Domínio

Histórico de Alterações

Data	Versão	Descrição	Autor

1. Introdução

*[A introdução da **Especificação de Modelagem do Domínio (EMD)** deve fornecer uma visão geral de todo o documento.]*

2. Conceitos do Domínio de Aplicação

[Nesta seção devem ser incluídos todos os conceitos do domínio de aplicação que foram especificados durante reunião realizada entre o especialista no domínio e desenvolvedores. O DRI gerado ajuda no preenchimento desta seção.]

3. Representação do Domínio de Aplicação (Ontologia)

[Nesta seção deve ser incluída a representação gráfica da ontologia criada para o domínio de aplicação. Esta imagem deve ser obtida da ferramenta utilizada para modelagem, como a Prótegé. Além disso, devem ser descritos os recursos utilizados para elaboração do modelo, como também, uma descrição detalhada da modelagem em questão.]

Documento de Especificação do Companheiro

Histórico de Alterações

Data	Versão	Descrição	Autor

1. Introdução

*[A introdução do **Documento de Especificação do Companheiro (DEC)** deve fornecer uma visão geral de todo o documento.]*

2. Estratégias e Táticas do Companheiro Virtual Inteligente

[Nesta seção devem ser incluídas todas as estratégias e táticas do companheiro virtual inteligente em construção, definidas durante reunião realizada com a presença de todos os perfis envolvidos no desenvolvimento. O DRI gerado ajuda no preenchimento desta seção.]

Estratégias	Táticas
Estratégia 1	*Tática 1.1*
	Tática 1.2
Estratégia 2	*Tática 2.1*
...	*...*

3. Comportamentos e Ações do Companheiro Virtual Inteligente

[Nesta seção devem ser incluídos todos os comportamentos e ações em resposta a ações do usuário do companheiro virtual inteligente em construção, definidos durante reunião

realizada com a presença de todos os perfis envolvidos no desenvolvimento. O DRI gerado ajuda no preenchimento desta seção.]

Comportamentos	**Ações**
Comportamento 1	*Ação 1.1*
	Ação 1.2
Comportamento 2	*Ação 2.1*
...	*...*

Modelo Arquitetural

Histórico de Alterações

Data	Versão	Descrição	Autor

1. Introdução

*[A introdução do **Modelo Arquitetural (MA)** deve fornecer uma visão geral de todo o documento.]*

2. Componentes da Arquitetura

[Nesta seção devem ser definidos e descritos os componentes da arquitetura do companheiro virtual inteligente em construção.]

3. Modelagem Arquitetural

[Nesta seção deve ser incluída a modelagem arquitetural do companheiro virtual inteligente, criada a partir de alguma linguagem de modelagem arquitetural e ferramenta escolhida pelos stakeholders envolvidos no projeto.]

Relatório de Avaliação do Produto

Histórico de Alterações

Data	Versão	Descrição	Autor

1. Introdução

*[A introdução do **Relatório de Avaliação do Produto (RAP)** deve fornecer uma visão geral de todo o documento.]*

2. Técnicas de Validação Utilizadas

[Nesta seção devem ser descritas as técnicas de validação que foram definidas e utilizadas para avaliação do companheiro virtual inteligente construído.]

3. Procedimentos Realizados

[Esta seção deve descrever como foi realizada a validação do companheiro virtual inteligente diante de cada técnica de validação definida. Além disso, deve descrever quais procedimentos foram realizados para a realização da validação.]

4. Problemas Encontrados

[Esta seção deve conter todos os erros e/ou melhorias detectadas durante a validação do companheiro virtual inteligente diante de situações de interação com realizadas com os usuários finais.]

Tipo	Descrição
□ *Erro* □ *Melhoria*	*Texto que descreve o erro / melhoria*
...	...

Documento de Registro de Informações

Fase: □ *Identificação do Domínio de Aplicação*
□ *Descrição do Domínio de Aplicação*
□ *Especificação do Companheiro Virtual*
□ *Construção da Arquitetura*
□ *Implementação do Companheiro Virtual*
□ *Avaliação do Companheiro Virtual*

Atividade: _____

Data da Reunião: _____

Paraticipantes: _____
_____
_____
_____
_____

1. Itens levantados em Reunião (*Brainstorming*)

[Esta seção deve apresentar todos itens e observações feitas durante reunião para determinada atividade de construção do companheiro virtual inteligente.]

□ *Item 1*
□ *Item 2*

....

□ *Item N*