UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Um Processo Ágil de Engenharia de Requisitos com Apoio de Padrões de Software"

DANIEL EDUARDO FUNABASHI DE TOLEDO

São Carlos/SP
Janeiro/2008

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

T649pa

Toledo, Daniel Eduardo Funabashi de.
Um processo ágil de engenharia de requisitos com apoio de padrões de software / Daniel Eduardo Funabashi de Toledo. -- São Carlos : UFSCar, 2008.
119 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2008.

1. Engenharia de requisitos. 2. Padrões de software. 3. Avaliação de processo de software. 4. Métodos ágeis. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

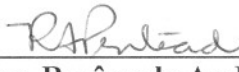
Programa de Pós-Graduação em Ciência da Computação

"Um Processo Ágil de Engenharia de Requisitos com Apoio de Padrões de Software"

DANIEL EDUARDO FUNABASHI DE TOLEDO

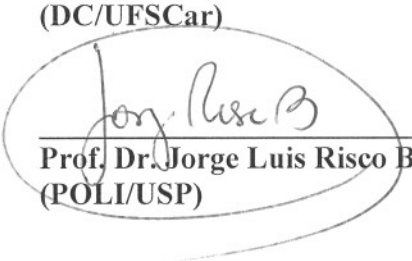
Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:



Profa.Dra. Rosângela Ap.Dellosso Penteado
(Orientadora - DC/UFSCar)

Prof. Dr. Antonio Carlos do Santos
(DC/UFSCar)



Prof. Dr. Jorge Luis Risco Becerra
(POLI/USP)

São Carlos
Janeiro/2008

Dedico este trabalho aos meus pais, Vail e Rosa, e a minha esposa Fernanda que, com amor, fizeram com que cada passo da minha vida fosse ainda mais especial.

Agradecimentos

À Profa. Dra. Rosângela Aparecida Dellosso Penteado, pela orientação, apoio, compreensão e paciência, sem as quais não seria possível a realização deste trabalho;

À empresa que proporcionou a realização do estudo de caso, sem a qual não conseguiríamos realizar nossas pesquisas nas áreas escolhidas;

Aos companheiros que aceitaram realizar o estudo de caso. Aliás, sem vocês, minha tarefa de coleta e medição teria sido impossível;

Aos professores da banca de qualificação, pelas valiosas sugestões;

À Secretaria de pós-graduação e à coordenadora Profa. Dra. Heloisa, por depositarem confiança no meu trabalho e fornecerem tudo de que precisei para a realização deste projeto;

À Dra. Nelkis de la Orden Medina , pelas orientações e sugestões em boa hora;

Ao Prof. Dr. Fernão Stella R. Germano, pelos valiosos ensinamentos e orientações;

A todos os colegas da pós-graduação, pelos momentos de descontração e ajuda mútua;

Aos amigos, Marcelo e Murilo, pelas festas e churrascos proporcionados;

À Rita e ao Toninho, pelo apoio, estímulo e confiança;

Aos meus irmãos, Larissa e Lucas, pelo carinho, estímulo e confiança;

Aos meus pais, pelo 'porto seguro' que sempre representaram na minha vida, por todo o apoio, força, amor e confiança que sempre me dedicaram e que jamais conseguirei retribuir;

À Fernanda, que, mais do que ter me apoiado, esteve comigo durante todas as etapas do meu trabalho, principalmente nas horas mais difíceis, sempre me ouvindo e ajudando, com muita paciência e carinho;

A todos que não foram citados, mas que estiveram presentes nestes anos de trabalho, auxiliando-me de uma forma ou de outra;

Por fim, agradeço a Deus, não apenas por mais esta realização em minha vida, mas também por ter colocado todas essas pessoas maravilhosas no meu caminho.

A TODOS, MUITO OBRIGADO!

"É necessário abrir o coração à bondade,
o cérebro à compreensão, a existência ao trabalho,
o passo ao bem, o verbo à fraternidade".

Emmanuel (Chico Xavier)

Resumo

A engenharia de requisitos tem evoluído historicamente ligada ao modelo de processo cascata de desenvolvimento de sistemas. Para processos que têm a participação efetiva dos clientes na tomada de decisões e na evolução do desenvolvimento, chamados de processos ágeis, seu valor continua a ser essencial para desenvolver sistemas que realmente satisfazem às necessidades do cliente. O modelo de processo iterativo e com alta interação com o cliente é a chave para preservar essa importância. Padrões de requisitos e organizacionais podem ser usados para guiar o engenheiro de requisitos para obter processos eficientes a fim de desenvolver sistemas seguindo os princípios e abordagens do desenvolvimento ágil. No contexto de uma organização real de desenvolvimento de sistemas, uma proposta de um processo ágil para desenvolvimento de sistemas iterativa e interativamente usando padrões organizacionais e de requisitos foi formulada e avaliada em um estudo de caso real. Essa proposta foi modela no meta-modelo SPEM (OMG, 2005) e considera o adiantamento da aplicação de métricas de software para a fase de requisitos a fim de melhor planejar as iterações seguintes. A equipe que trabalhou na avaliação foi recrutada internamente na organização e o autor da proposta e desta dissertação também é membro da organização. Ele treinou seus colegas e acompanhou e registrou a avaliação conduzida. Um questionário foi formulado e aplicado à equipe e os resultados dessa aplicação são aqui relatados. Algumas das características do processo proposto foram incorporadas à prática de desenvolvimento de sistemas da organização e algumas estão sendo consideradas para incorporação como a adaptação necessária. Elas são aqui comentadas juntamente com a apreciação da equipe sobre a modelagem em SPEM e o adiantamento da aplicação das métricas de software.

Abstract

Requirements engineering has evolved historically connected to the waterfall process model of system development. For agile processes its value continues to be essential to develop systems that really satisfy the customer needs. The iterative process model with high interaction with the customer is the key to preserve its importance. Organizational and requirements patterns can be used to guide the requirements engineer to obtain efficient processes to develop systems following principles and approaches of agile development. In the context of a real systems development organization a proposal of an agile process for system development iteratively and interactively using organizational and requirements patterns has been formulated and evaluated in a real case study. This proposal has been modeled in the SPEM meta-model and considers advancing the application of software metrics to the requirements phase in order to better plan following iterations. The team that worked in the evaluation has been recruited internally in the organization and the author of the proposal and of this dissertation has also been a member of the organization. He trained his colleagues and followed and recorded the evaluation conducted. A questionnaire has been formulated and applied to the team and the results of this application are reported here. Some of the characteristics of the proposed process have been incorporated to the organization systems development practice and some are being considered for incorporation with the necessary adaptation. These are here commented together with the team opinion about the SPEM modeling and advanced application of software metrics.

Lista de Tabelas

Lista de Figuras

Lista de Quadros

Sumário

Capítulo 1 – Introdução

1.1 - Considerações Iniciais

A engenharia de software é citada por Pressman (2006) como uma tecnologia em camadas, cuja base tem o enfoque na qualidade e as outras camadas tratam do processo, métodos, técnicas e ferramentas a serem utilizados tanto para o desenvolvimento quanto para a manutenção de software.

Mesmo com a constante evolução de métodos, técnicas e ferramentas, a entrega de software em prazos e custos estabelecidos nem sempre é conseguida, bem como, muitas vezes, o produto desenvolvido nem sempre corresponder às reais necessidades dos clientes (LARMAN, 2007). Várias razões podem ser atribuídas a esses problemas, mas o fator principal é a manipulação inadequada de requisitos (KOTONYA; SOMMERVILLE, 1998).

Requisitos são definidos como condição indispensável que deve ser atendida para alcançar os objetivos e resolver os problemas e as necessidades de clientes durante o desenvolvimento do software (IEEE, 1990). Para ajudar engenheiros de software a melhor compreender esses problemas e necessidades, a engenharia de requisitos fornece mecanismos apropriados para entender o que o cliente realmente deseja, analisando as suas necessidades, negociando uma condição, especificando a solução de modo não ambíguo e gerindo os requisitos durante todo o desenvolvimento do software (PRESSMAN, 2006). Assim, a engenharia de requisitos é uma etapa da engenharia de software essencial para a concepção de um novo sistema e está presente na maioria dos modelos de processo de desenvolvimento de software, porém, com diferentes enfoques (FOWLER, 2005). Em alguns modelos de processo tradicionais, como o cascata, a engenharia de requisitos deve ser completada no início do processo de desenvolvimento do software (SOMMERVILLE, 2003). Nos modelos de processo ágeis a engenharia de requisitos ocorre de forma iterativa com ativa participação do cliente na especificação dos requisitos (ABRAHAMSSON et al., 2002). O cliente também participa ativamente das tomadas de decisões durante o projeto, pois modelos de processo ágeis têm a habilidade de serem flexíveis e reagir prontamente às modificações que necessitam ser apropriadamente geridas para proporcionar o desenvolvimento correto das reais necessidades do cliente (KRUCHTEN, 2001).

Dessa forma, os métodos ágeis abordam o processo de engenharia de requisitos diferente dos modelos de processo tradicionais. A principal diferença está na forma como as

modificações em requisitos são tratadas durante o desenvolvimento do software. Os modelos de processo tradicionais adotam a estratégia de previsibilidade. Utilizam técnicas para compreender o domínio do problema e elicitar todos os requisitos antes de iniciar o desenvolvimento. Após a elicitação e especificação dos requisitos um planejamento é realizado para que as modificações possam ser controladas no decorrer do processo de desenvolvimento do software. Por outro lado os métodos ágeis optam pela adaptabilidade, os requisitos são elicitados aos poucos e o planejamento é contínuo para que adaptações às mudanças possam ocorrer (FOWLER, 2005).

Embora sejam flexíveis, se adaptem à medida que as necessidades dos clientes aparecem, os modelos ágeis de processo de desenvolvimento de software devem ser utilizados de forma organizada, estabelecendo diretrizes a serem seguidas para realizar as suas atividades (CREEL, 2006).

Em busca de modos eficientes para apoiar de forma efetiva a realização da engenharia de software nos métodos ágeis, o uso de padrões de software (software *patterns*) vem se mostrando eficaz. Padrões de software fornecem soluções de sucesso comprovado para problemas recorrentes em um determinado contexto e são classificados em categorias (COPLIEN; HARRISON, 2004).

Dentre as várias categorias existentes, os padrões de requisitos e os organizacionais propõem soluções para o contexto da engenharia de requisitos. Esses padrões documentam práticas de sucesso comprovado para auxiliar na solução de problemas que ocorrem na realização das atividades de engenharia de requisitos e podem ser utilizados para organizar e guiar o seu processo (SHOEMAKER, 2007).

1.2 - Motivação e Objetivo

A demanda cada vez maior por produtos de software que necessitam ser liberados de forma mais rápida e com qualidade faz com que projetos e organizações, embora com características particulares, reutilizem práticas de sucesso para realizar as atividades de engenharia de requisitos.

Com práticas de sucesso comprovado para o contexto da engenharia de requisitos, os padrões de requisitos e organizacionais foram pesquisados na literatura para auxiliarem na realização das atividades inerentes a esta etapa do desenvolvimento de software. Assim, a principal motivação deste trabalho consiste em buscar apoio nos padrões de requisitos e

organizacionais para que possam ser utilizados de forma eficiente na obtenção e gerenciamento dos requisitos do sistema nos métodos ágeis, sem que a agilidade seja afetada.

Nesse contexto, este trabalho tem como objetivo utilizar padrões de requisitos e organizacionais que possam de forma efetiva organizar, conduzir e auxiliar a realização das atividades do processo de engenharia de requisitos, quando realizado de acordo com os princípios preconizados pela metodologia ágil.

Para atingir esse objetivo, um processo ágil de engenharia de requisitos é proposto. Esse processo é modelado por meio do meta-modelo SPEM (*Software Process Engineering Metamodel*) (OMG, 2005) para mostrar o relacionamento e a ordenação dos seus elementos e teve os padrões de requisitos e organizacionais integrados por meio do estereótipo "guia" do SPEM. Para corroborar o processo elaborado, sua aplicação ocorreu em um estudo de caso, realizado em uma empresa real de desenvolvimento de software, para analisar as vantagens e desvantagens por ele proporcionados, bem como pela sua modelagem em SPEM juntamente com os padrões utilizados.

1.3 - Organização do Trabalho

Este trabalho está organizado em cinco capítulos além deste. O Capítulo 2 apresenta alguns dos assuntos relevantes utilizados durante este trabalho, como por exemplo: processos de software, meta-modelo SPEM, engenharia de requisitos, métricas de software, Pontos de Casos de Uso, métodos ágeis e padrões de software.

O Capítulo 3 apresenta alguns padrões de requisitos e organizacionais existentes na literatura com as respectivas modificações que foram realizadas para que fossem utilizados neste trabalho.

O Capítulo 4 apresenta o processo ágil de engenharia de requisitos proposto. Esse processo é modelado por meio do meta-modelo SPEM, e utiliza padrões de requisitos e organizacionais para guiar a realização de suas atividades.

O Capítulo 5 apresenta o resultado da aplicação do processo proposto em um estudo de caso desenvolvido em uma empresa real de desenvolvimento de software e também apresenta as análises realizadas com base em um questionário aplicado à equipe que participou desse estudo de caso para analisar as vantagens e desvantagens do processo proposto.

O Capítulo 6 apresenta as conclusões deste trabalho evidenciando os resultados obtidos, as contribuições e limitações, além de sugestões para trabalhos futuros.

Capítulo 2 – Assuntos Relacionados

2.1 - Considerações Iniciais

Organizações que desenvolvem software necessitam realizar um conjunto de atividades para gerenciar, desenvolver e manter o software (ACUÑA; FERRÉ, 2001). Esse conjunto de atividades também chamado de processo de engenharia de software possibilita o desenvolvimento racional do software por meio de tecnologias e práticas (SOMMERVILLE, 2003).

Para direcionar efetivamente o trabalho de engenharia de software, modelos de processo devem ser utilizados. Esses modelos representam o relacionamento entre os elementos do processo e procuram direcionar a seqüência de passos a serem realizados facilitando a visualização e entendimento do processo por parte das pessoas envolvidas (SOMMERVILLE, 2003). A maioria dos modelos de processo da engenharia de software começa com a especificação das necessidades dos usuários e termina com a elaboração e manutenção do software. Essas necessidades de usuários, também chamadas de requisitos, devem ser adquiridas e geridas, por meio de um conjunto de atividades, durante todo o processo de desenvolvimento do software. Esse conjunto de atividades corresponde ao processo de engenharia de requisitos, o qual pode ser realizado iterativamente para obter, analisar, validar, documentar e gerenciar os seus requisitos (THAYER; DORFMAN, 1997).

Na atividade de elicitação de requisitos, as necessidades dos usuários são obtidas de forma geral, ou seja, em alto nível de abstração. Depois, os requisitos são analisados e ajustados para compor um documento de requisitos, o qual é validado pelo usuário que detém o conhecimento das regras de negócio do sistema. A atividade de gerenciamento de requisitos fica responsável por qualquer interrupção ou mudanças em requisitos durante todo o desenvolvimento do software (SAWYER; KOTONIA, 2001). Os requisitos podem ter o tempo estimado para serem atendidos durante o desenvolvimento do software. Estimativas de tamanho é um processo pelo qual uma pessoa ou um grupo de pessoas estima o tamanho de um projeto de software (ANDRADE; OLIVEIRA, 2004). O tamanho de um projeto de software é uma das primeiras estimativas a ser realizada, pois dessa dimensão depende a definição de esforço, custos e prazos necessários para a definição do plano de desenvolvimento do software (ANDRADE; OLIVEIRA, 2004).

Existem diferentes metodologias de desenvolvimento de sistemas de software. As metodologias tradicionais têm o enfoque na estratégia de previsibilidade, tentam planejar grande parte do software em muitos detalhes com bastante antecedência, sendo resistentes a mudanças (FOWLER, 2005). Por outro lado, os métodos ágeis abordam os requisitos ao longo de todo o processo de desenvolvimento do software (TOMAYKO, 2002). Esses métodos têm o enfoque na adaptabilidade, isto é, os envolvidos com o projeto aceitam a idéia de que os requisitos do software vão ser conhecidos à medida que o processo de desenvolvimento avança (AMBLER, 2007b).

Para auxiliar a realização e organização das atividades da engenharia de software, padrões de software podem ser utilizados. Esses padrões propõem soluções de sucesso comprovado e possibilitam o reúso de suas soluções no contexto da engenharia de software.

Este capítulo contém os assuntos necessários para o desenvolvimento deste trabalho e está organizado da seguinte forma: a Seção 2.2 apresenta conceitos de processo e modelagem de processo de software, com destaque para o meta-modelo SPEM (OMG, 2005); a Seção 2.3 aborda conceitos sobre engenharia de requisitos, a Seção 2.4 apresenta os conceitos de métricas de software, com destaque para a técnica de Pontos de Casos de Uso; a Seção 2.5 apresenta os conceitos de metodologia ágil; a Seção 2.6 apresenta conceitos de padrões de software e, na Seção 2.7, estão as considerações finais.

2.2 - Processo de software

Processo de software pode ser compreendido como um conjunto de atividades, parcialmente ordenadas, e resultados associados cuja realização produzem um produto de software (SOMMERVILLE, 2003). De acordo com o IEEE (*Institute of Electrical and Electronic Engineers*) (1990) processo é definido como uma seqüência de passos realizados para um determinado propósito. Trata-se do fundamento da Engenharia de Software, possibilitando o desenvolvimento racional do software através da efetiva utilização de um conjunto de atividades, práticas, métodos e tecnologias visando desenvolver e manter o software (PRESSMAN, 2006; SOMMERVILLE, 2003).

Dentre os objetivos que um processo de desenvolvimento de software deve alcançar, Tyrrell (2000) destaca:

- **Eficácia**: Um processo eficaz auxilia o desenvolvimento do produto correto. Deve auxiliar os desenvolvedores quanto: ao levantamento das necessidades do cliente, a

produzir o que o cliente necessita e, principalmente, a verificar se o que foi produzido é realmente o que foi requisitado pelo cliente.

- **Manutenibilidade**: Um dos objetivos de um bom processo é expor o raciocínio usado pelos projetistas e programadores para que identifiquem os defeitos e mudanças que devem ser realizadas. Dessa forma, evita-se a perda de informações relevantes com a saída de um membro da equipe de desenvolvimento que detenha um determinado conhecimento sobre o projeto. Conseqüentemente o trabalho decorrente de alterações de requisitos também ficará facilitado.
- **Preditibilidade**: O desenvolvimento de um novo produto precisa ser planejado, de forma que sejam alocados recursos necessários, como tempo e pessoas.
- **Repetitividade**: Um processo deve ser replicável em outros projetos. Os processos elaborados sob demanda (*ad-hoc*) raramente podem ser reusados se não forem mantidas as mesmas condições e as mesmas pessoas trabalhando no novo projeto. Há perda de tempo e sobrecarga para produzir um processo sempre a partir do zero.
- **Garantia de Qualidade**: O objetivo de um processo definido é possibilitar que engenheiros de software desenvolvam produtos com alta qualidade. Qualidade pode ser definida como a adequação do produto aos requisitos do sistema. O processo deve prover um relacionamento entre o requisitado pelo cliente e o produto desenvolvido.
- **Perfectibilidade** (Melhoria Contínua): Um processo deve ser aperfeiçoado. Não se pode esperar que um processo esteja tão perfeito que não possa ser melhorado. Tecnologias, ambiente de trabalho, estratégias de desenvolvimento e os produtos requisitados estão em constante mudança, tornando o processo passível de modificação e em busca de melhoria contínua.
- **Rastreabilidade**: Um processo deve possibilitar que gerente, desenvolvedores e clientes acompanhem o projeto e localizem os problemas e seus responsáveis.

Os processos de software são formados por diversos elementos e pela definição específica de seus relacionamentos. Os elementos comuns em processos de software, de acordo com Conradi et al. (1994) e Acuña e Ferré (2001) são:

- **Agente ou Ator:** é a entidade que participa da execução do processo.

- **Papel:** descreve as responsabilidades e habilidades necessárias para realizar uma atividade específica do processo de software.
- **Atividade:** representa o trabalho a ser realizado por um ator ou grupo de atores em determinado ponto de execução do processo.
- **Artefato:** é o subproduto produzido e mantido no processo. Podem ser gerados, consumidos e modificados em uma atividade, e, assim, podem ter várias versões ao longo do processo.

2.2.1 - Modelagem de processo de software

Modelos de processo de software definem um conjunto distinto de atividades, tarefas, marcos e produtos de trabalho necessários para realizar a engenharia de software. Esses modelos representam o relacionamento entre os elementos do processo e procuram direcionar efetivamente o trabalho de engenharia de software (ACUÑA; FERRÉ, 2001; PRESSMAN, 2006). As vantagens em se modelar um processo, segundo Curtis et al. (1992), são:

- **Facilitar o entendimento e comunicação:** A modelagem do processo permite a estruturação dos seus elementos mostrando seus inter-relacionamentos e tornando mais fácil a visualização e entendimento do processo por parte das pessoas envolvidas.
- **Gestão e controle do processo:** permite e facilita o monitoramento, gerenciamento e coordenação das atividades do processo.
- **Suporte para melhoria de processo:** todos os envolvidos têm a possibilidade de sugerir adaptações e melhorias devido ao maior conhecimento do processo.
- **Suporte para a automação de partes do processo:** uma vez que se tenha definido um modelo para representar processos, ferramentas podem ser criadas para automatizar o processo.

Sommerville (2003) comenta que as atividades descritas a seguir são comuns aos modelos de processo de software:

- **Especificação de software:** é necessário definir a funcionalidade do software e as restrições para a sua operação.

- **Projeto e implementação de software:** o software deve ser produzido de modo que atenda a sua especificação.
- **Validação de software:** o software precisa ser validado para garantir que ele faz o que o cliente deseja.
- **Evolução de software:** o software precisa evoluir para atender às necessidades de modificações em requisitos do sistema.

Com a modelagem de processos de software consegue-se determinar os inter-relacionamentos entre os elementos que fazem parte do processo, retratando uma visão do processo de software (SAMPAIO, 2004). Existem diversas linguagens e meta-modelos que possibilitam modelar processos de software. Entretanto, devido ao surgimento de várias linguagens de modelagem de processo de software a OMG (*Object Management Group*) (OMG, 2005) decidiu submeter uma proposta de requisição por uma linguagem de modelagem para processos de software. Diversas empresas como: IBM, Unisys, Alcatel se uniram para elaborar um meta-modelo que atendesse aos requisitos propostos pela OMG. Assim, a meta-linguagem *Software Process Engineering Metamodel* - SPEM, baseada na UML, foi oficializada como um padrão da OMG e é apresentada de forma resumida na próxima Subseção.

2.2.1.1 - SPEM - Software Process Engineering Metamodel

O SPEM é baseado no MOF (*Meta-Object Facility*), tecnologia para definir meta-dados e representá-los como objetos, que utiliza a abordagem orientada a objetos para modelar processos e a UML como notação para a modelagem. É estruturado como um perfil UML (UML *profile)*, ou seja, uma variante da UML que utiliza mecanismos de extensão (estereótipos) com a finalidade de normalizar sua utilização para um domínio particular (OMG, 2005).

Os princípios fundamentais do SPEM são provenientes da idéia de que o processo é uma colaboração entre entidades abstratas ativas, chamadas de **Papel** de processos (*process role*), que realizam operações chamadas de **Atividade** (*activity*) em entidades chamadas **Artefato** (*work product*), como mostra a Figura 2.1 (OMG, 2005).

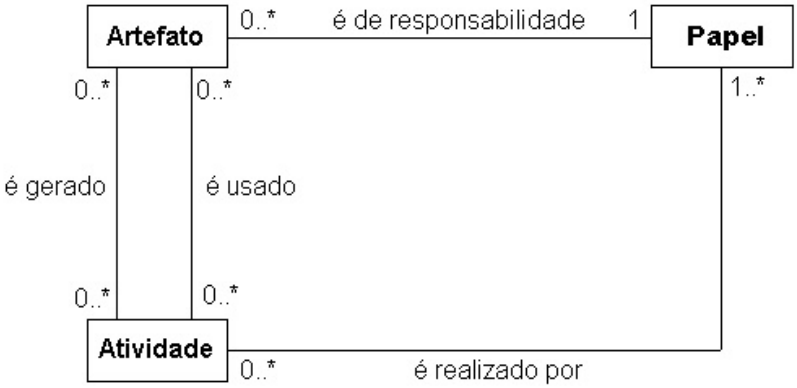
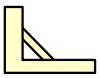
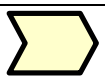
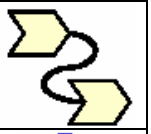


Figura 2.1 – Modelo conceitual do SPEM (adaptado de OMG, 2005).

Essa figura apresenta o relacionamento entre os estereótipos fundamentais de processo, porém, o SPEM define outros para modelagem de processo. Os estereótipos utilizados neste trabalho são apresentados de forma resumida na Tabela 2.1. A primeira coluna mostra o nome do estereótipo, a segunda apresenta a sua descrição e a terceira mostra a notação utilizada para representá-lo.

Tabela 2.1 – Estereótipos utilizados pelo SPEM (adaptado de OMG, 2005 apud Costa, 2006).

Estereótipo	Descrição	Notação
Artefato (*WorkProduct*)	É qualquer coisa produzida, consumida ou modificada por um processo. Um artefato pode pertencer a uma categoria específica de artefato como, por exemplo, Documento, Modelo UML, entre outras.	
Guia (*Guidance*)	Provê informações adicionais sobre o elemento do processo ao qual está associado, como: Normas, Técnicas, "Templates", entre outros. Um guia pode pertencer a uma categoria de guia.	
Atividade (*Activity*)	Descreve parte do trabalho que deve ser realizado por um papel durante o processo.	
Pacotes de processo (*Process Package*)	É um conjunto de elementos de processo agrupados de forma coerente, como: Engenharia de Requisitos, Implementação, entre outros.	
Processo (*Process*)	É a descrição completa de um processo de desenvolvimento de software.	
Documento (*Document*)	É uma categoria de artefato (*WorkProductKind*) e representa um documento produzido, visualizado ou alterado ao longo do processo.	
Conjunto de Trabalho (*WorkDefinition*)	Descreve o trabalho realizado no processo e pode ser criado para representar partes compostas do trabalho que serão futuramente separadas.	
Ator	Define a responsabilidade sobre os produtos de trabalho específicos e o papel de um determinado indivíduo no processo.	

Cada estereótipo do SPEM possui uma notação que pode ser utilizada com diagramas UML para representar diferentes perspectivas do processo de software. Segundo a OMG (2005), as notações apresentadas na Tabela 2.1 são usadas com os seguintes diagramas UML para representar o processo de software: Diagrama de Classes; Diagrama de Pacotes; Diagrama de Atividades; Diagrama de Casos de Uso; Diagrama de Seqüência; Diagrama de Estados (OMG, 2005).

Diagrama de atividades representa a perspectiva dinâmica do processo, descrevendo quais artefatos são gerados ou consumidos pelas atividades e a seqüência em que essas atividades devem ocorrer.

2.3 - Engenharia de Requisitos

Apesar dos avanços tecnológicos dos últimos anos os problemas de custo, manutenção e qualidade em software permanecem (NUSEIBEH; EASTERBROOK, 2000; THAYER; DORFMAN, 1997). Young (2002) aponta que 85% das causas de falhas em projetos de software estão relacionadas às deficiências em determinar os requisitos, sendo que os tipos mais comuns de erros são suposições incorretas, omissões, inconsistências e ambigüidades.

Segundo Boehm et al. (2002), erros em requisitos de software são até 20 vezes mais caros de se corrigir do que qualquer outro tipo de erro quando só detectados depois do software implementado. Em outros estudos sobre falhas em requisitos de software, Lamsweerde (2000), aponta que os problemas relacionados aos requisitos do sistema afetam boa parte das organizações que desenvolvem e usam sistemas de software.

Inúmeras definições para o termo requisitos são encontradas na literatura. Segundo o IEEE (1990), requisitos são definidos como uma condição que deve ser cumprida ou possuída por um sistema de software para resolver problemas ou alcançar objetivos de usuários, satisfazendo um contrato, uma norma, uma especificação ou outros documentos formalmente impostos.

Embora o enfoque da definição de requisitos de software incida sobre as necessidades dos usuários, ao defini-los também deve-se considerar as necessidades de todos os interessados no sistema (*stakeholders)*, ou seja, todas as pessoas que têm ligação direta ou indireta com o software: equipe de desenvolvimento, clientes, usuários finais, gerentes, analistas de sistemas, mantenedores, etc. (KOTONYA; SOMMERVILLE, 1998).

Dahlstedt (2003) afirma que os requisitos possuem três elementos que o caracterizam: a origem, o motivo e o objeto de sua realização, como ilustra a Figura 2.2.

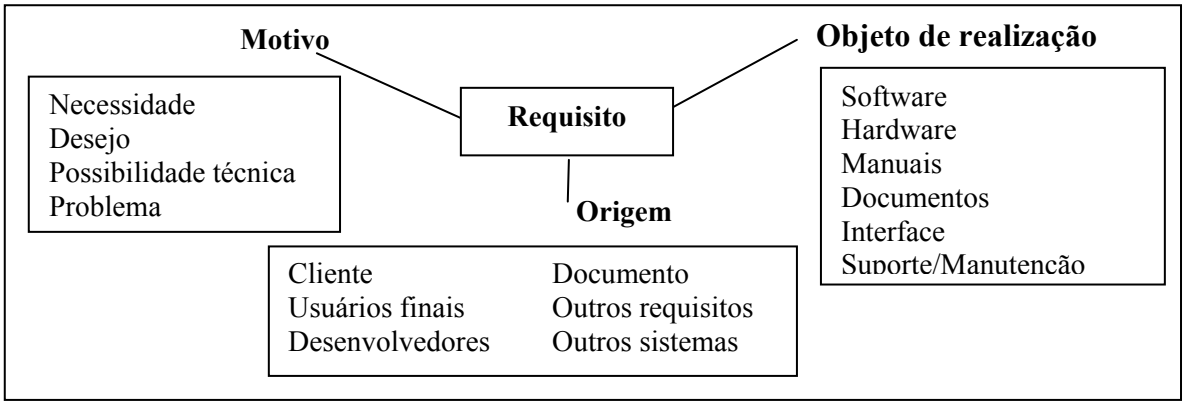


Figura 2.2 – Elementos de um requisito (adaptado de DAHLSTEDT, 2003).

A origem de um requisito pode ser qualquer pessoa interessada no sistema, um documento de negócio, requisitos especificados anteriormente ou originados de interações com outros sistemas. O motivo pelo qual um requisito existe é, geralmente, para realizar ou satisfazer algo para os interessados no sistema, como: uma função que o usuário precisa para desempenhar seu trabalho, uma possibilidade técnica que realiza certa atividade de negócio de modo mais eficiente, uma necessidade para apoiar as estratégias de negócio da organização. Por fim, um requisito tem um ou mais objetos de realização, o mais usual é o software, mas também pode ser hardware, manuais, interface, documentação do sistema, etc. (DAHLSTEDT, 2003).

As organizações que desenvolvem software necessitam de um conjunto de atividades a ser seguido para obter, analisar, validar, documentar e gerenciar os requisitos do sistema (THAYER; DORFMAN, 1997). Esse conjunto estruturado de atividades, também chamado de processo de engenharia de requisitos, pode ser realizado de maneira iterativa durante o desenvolvimento do software (SOMMERVILLE, 2003).

A Figura 2.3 mostra como essas atividades são desenvolvidas pelo modelo de processo iterativo de engenharia de requisitos. O processo inicia-se na atividade de elicitação de requisitos, a partir de necessidades dos usuários, por exemplo. Ao término dessa atividade o processo possui os requisitos de maneira informal. Assim, inicia-se a atividade de análise de requisitos, na qual os requisitos serão analisados e ajustados. Seguindo no modelo espiral, inicia-se a atividade de especificação de requisitos, onde os requisitos ajustados passam a compor um documento de requisitos. Esse documento é validado na atividade de validação de

requisitos e ao término dessa atividade, decide-se se o documento gerado será aceito ou se será realizado novo ciclo no espiral. Já a atividade de gerenciamento de requisitos fica responsável por qualquer interrupção ou mudanças dos requisitos durante desenvolvimento do sistema. As atividades são repetidas em ciclos até que um documento de especificação de requisitos seja produzido de forma aceitável pelo engenheiro de requisitos e pelo cliente (SAWYER; KOTONIA, 2001).

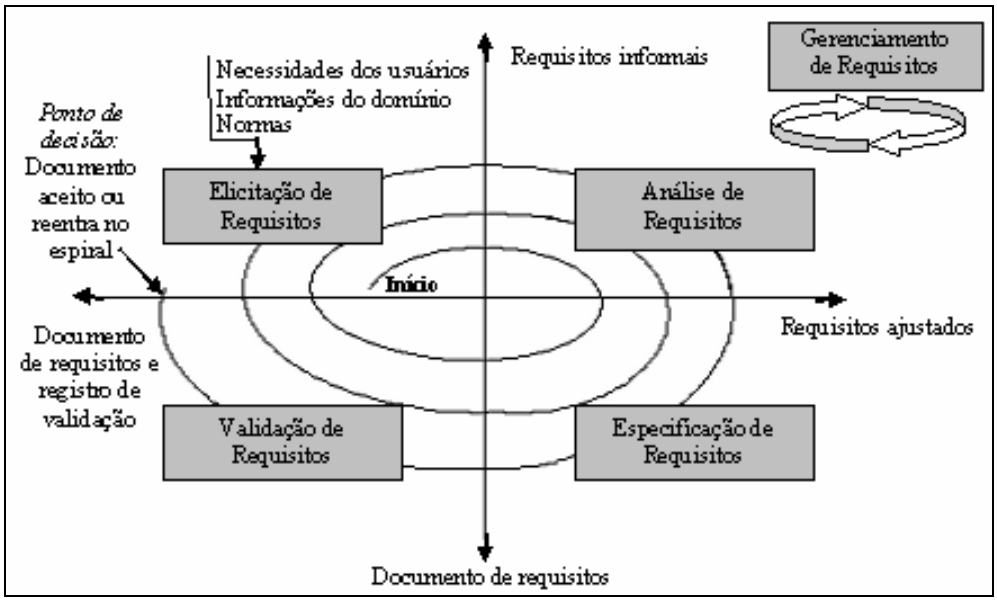


Figura 2.3 – Modelo espiral do processo de engenharia de requisitos (adaptado de SAWYER; KOTONIA, 2001).

O uso do termo "engenharia" implica que conceitos, práticas e técnicas devem ser utilizados para garantir que os requisitos do sistema sejam completos, consistentes e relevantes (SOMMERVILLE; SAWYER, 1997). Assim, Carvalho e Chiossi (2001) definem que engenharia de requisitos é o processo de transformação de idéias que estão na mente dos usuários (entrada) em um documento (saída), que fornece a todas as partes entendimento escrito do problema descrevendo "*o quê"* o produto a ser desenvolvido deverá fazer sem, entretanto, descrever "*como"* deve ser feito. Isso pode ser conseguido por meio de cenários de usuário, listas de funções e de características, modelos de análise ou uma especificação.

Thayer e Dorfman (1997), ressalvam que a engenharia de requisitos auxilia os engenheiros de software fornecendo um processo para entender as necessidades dos usuários, analisar essas necessidades, avaliar e negociar as soluções propostas para a construção do software.

2.3.1 - Atividades da Engenharia de Requisitos

Para as atividades da engenharia de requisitos, Siddiqi e Shekaran (1996) comentam que há pouca uniformidade quanto aos nomes. Na Tabela 2.2 são apresentadas algumas denominações para essas atividades de acordo com diversos autores. A primeira linha apresenta os autores e as colunas apresentam os conceitos por eles usados.

Tabela 2.2 – Denominações para as atividades da engenharia de requisitos de acordo com diferentes autores.

PRESSMAN, 2006	SOMMERVILLE, 2003	DAVIS E HICKEY, 2002	SAWYER E KOTONIA, 2001	THAYER E DORFMAN, 1997
Concepção	Viabilidade	Elicitação	Elicitação	Elicitação
Elicitação	Elicitação			
Elaboração	Análise	Modelagem	Análise	Análise
Negociação		Triagem		
Especificação	Especificação	Especificação	Especificação	Especificação
Validação	Validação	Validação	Validação	Validação
Gestão	Gerenciamento		Gerenciamento	Gerenciamento

Apesar das diferentes representações, as atividades de elicitação, análise, especificação, validação e gerenciamento de requisitos podem ser a síntese de todas as outras atividades (THAYER; DORFMAN, 1997). A seguir são descritos cada uma dessas atividades.

2.3.1.1 - Elicitação de requisitos de software

Define o escopo e a natureza do problema a ser resolvido. Começa quando a necessidade de negócio é identificada dentro de uma organização. Têm a intenção de estabelecer o entendimento básico dos problemas, das pessoas que estão envolvidas com as soluções e da comunicação e colaboração efetiva entre clientes e desenvolvedores (PRESSMAN, 2006).

A necessidade de troca de informações durante a elicitação de requisitos faz com que essa atividade seja caracterizada por estreita interação entre clientes e desenvolvedores (CARVALHO; CHIOSSI, 2001).

Depois de definido o escopo do problema, essa fase tem como objetivo ajudar os interessados no sistema a definir e entender os requisitos do domínio da aplicação (SOMMERVILLE, 2003).

Uma estimativa pode ser feita para verificar se as necessidades do usuário podem ser satisfeitas usando as tecnologias de software e hardware corrente. É decidido se o sistema

proposto terá um custo efetivo do ponto de vista do negócio e se ele poderá ser desenvolvido conforme as restrições de orçamento (SOMMERVILLE, 2003).

Para ajudar a controlar problemas dessa atividade, algumas técnicas podem ser utilizadas para apoiar o processo de elicitação de requisitos, como: Entrevistas, Questionários, Cenários de Uso, Observação Direta, Protótipo, Análise de Documento, *Workshop*, *Brainstorming*, PIECES (*Performance, Information e data, Economy, Control, Efficiency e Service*) e JAD (*Joint Application Development*) (CARVALHO; CHIOSSI, 2001; GOGUEN; LINDE, 1993; KOTONYA; SOMMERVILLE, 1998).

2.3.1.2 - Análise de requisitos de software

Nessa atividade os requisitos são expandidos, refinados, modificados e priorizados. Os requisitos faltantes, conflitantes, ambíguos, e que se sobrepõem são, normalmente, identificados. Estimativas de tamanho, custo e esforço de desenvolvimento podem ser realizadas para avaliar o impacto desses requisitos no custo do projeto e no prazo de entrega (SAWYER; KOTONYA, 2001). Estimativas serão abordadas na Subseção 2.4.1.

A atividade de análise também possui tarefas de modelagem e refinamento de requisitos. Guiados pela criação de modelos de análise que descreverão como os usuários vão interagir com o sistema, esses modelos podem ser baseados em: cenários do usuário, atividades, classes, comportamentos, entre outros (PRESSMAN, 2006). Kotonya e Sommerville (2003), também sugerem o uso de listas de verificação (*checklists*) para analisar, priorizar e classificar os requisitos; e o uso de matrizes de interação para encontrar conflitos e sobreposições.

2.3.1.3 - Especificação de requisitos de software

Nessa atividade um documento contendo as descrições dos requisitos do software deve ser desenvolvido (SOMMERVILLE, 2003). Esse documento, denominado Documento de Requisitos, pode ser escrito em: Linguagem Natural, Modelo Gráfico, Modelo Matemático Formal, Modelos de Casos de Uso, Estórias de Usuário, Modelo de Características, Modelos de Interface ou qualquer combinação desses (PRESSMAN, 2006). Nesse documento os requisitos e características do sistema devem ser especificados de maneira completa, precisa e verificável (IEEE, 1990). Os modelos de Casos de Uso são utilizados em larga escala para a

especificação de requisitos (RUP, 2007). Os casos de uso captam as funcionalidades básicas do sistema do ponto de vista dos usuários e podem ser utilizados para representar todos os requisitos do sistema e/ou partes deles (COCKBURN, 2001). Requisitos não descritos com casos de uso, como os requisitos não funcionais, podem ser especificados textualmente, em linguagem natural (LARMAN, 2007).

Em termos gerais, o documento de requisitos representa uma ponte de comunicação entre clientes e os desenvolvedores, descrevendo o que o cliente espera que o sistema faça e os parâmetros de desempenho desse sistema (IEEE, 1998b).

Com essas descrições, o documento de requisitos deve atender a diferentes pessoas interessadas no sistema (SAWYER; KOTONYA, 2001). A Figura 2.4 ilustra alguns dos possíveis usuários desse documento e como eles o utilizam.

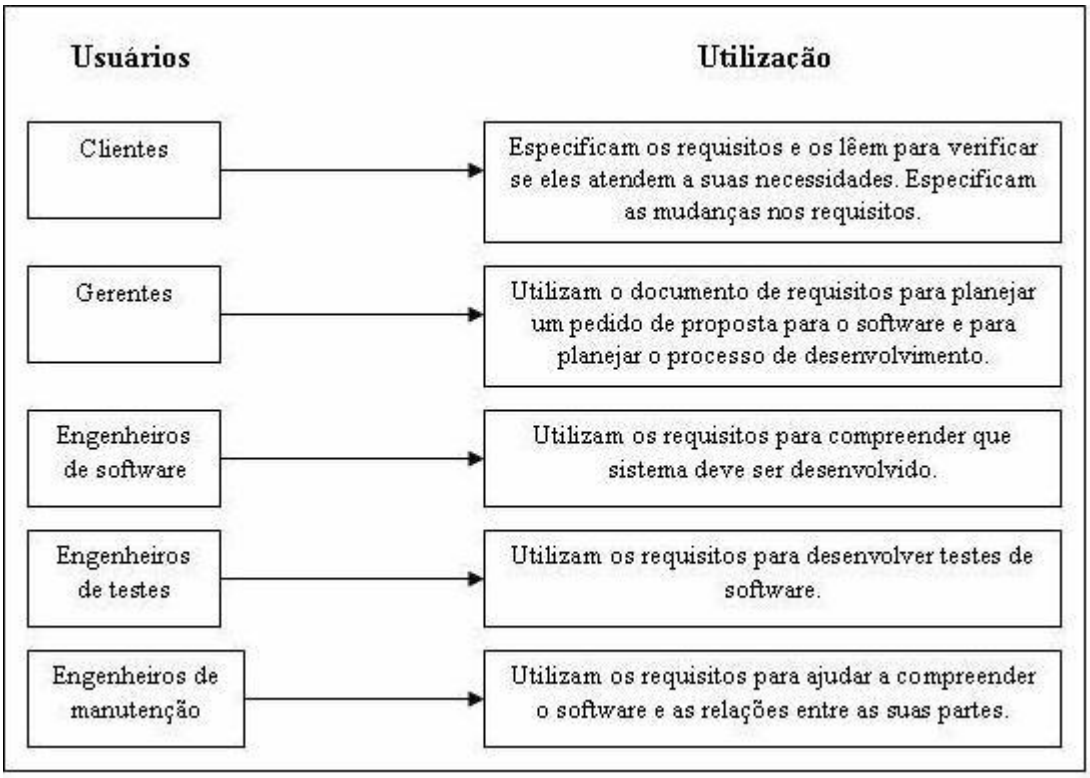


Figura 2.4 – Usuários do documento de requisitos (KOTONYA; SOMMERVILLE, 1998).

Como esse documento é para ser usado para especificar o que o software é solicitado a fazer, os defensores de notações formais argumentam que as representações formais trazem benefícios na comunicação com os desenvolvedores de sistema, em termos de entendimento da aplicação, da ausência de ambigüidade e da identificação de abstrações úteis ao desenvolvimento (DAMIAN, 2000).

Por outro lado, se o documento de requisitos é também usado na comunicação com os clientes e usuários do sistema, representações informais e linguagem natural são mais prováveis de ser bem sucedidas, já que clientes e usuários geralmente não são familiarizados com notações formais (SOMMERVILLE, 2003). Desse modo, a utilização de descrições em linguagem natural, combinados com modelos gráficos como os casos de uso, são tidos como suficientes para especificar requisitos (PRESSMAN, 2006). Já em sistemas onde os clientes possuem conhecimento de notações formais, as representações formais podem ser utilizadas para auxiliar no entendimento dos requisitos.

2.3.1.4 - Validação de requisitos de software

A validação de requisitos é definida como o processo que certifica que o documento de requisitos gerado esteja consistente com as necessidades e intenções de clientes e usuários, autenticando os artefatos que servirão de base para as fases subseqüentes do processo de desenvolvimento do software (PRESSMAN, 2006).

Nessa atividade o documento de requisitos é analisado para garantir que todos os requisitos do software estejam descritos de maneira não ambígua, que problemas de inconsistências, omissões e erros tenham sido detectados e corrigidos (SOMMERVILLE, 2003).

A validação não é só aplicada ao documento final de requisitos, mas durante as versões intermediárias geradas ao longo do processo de engenharia de requisitos (PRESSMAN, 2006).

2.3.1.5 - Gerenciamento de requisitos de software

O gerenciamento de requisitos auxilia os engenheiros de requisitos desde a obtenção até a validação de requisitos. Através do rastreamento de requisitos durante o processo de desenvolvimento do software, as modificações em requisitos podem ser realizadas de forma gerenciável. O rastreamento é um fator importante para gerenciar os requisitos do sistema, pois identifica, controla e os modifica à medida que o processo de desenvolvimento avança (PRESSMAN, 2006). Além de fornecer a documentação completa dos requisitos, o rastreamento ajuda no processo de manutenção de software (KOTONYA; SOMMERVILLE, 1998).

Todos os requisitos que serão modificados durante o desenvolvimento do software devem passar por um processo de gerenciamento de mudanças. A vantagem de utilizar um processo para esse gerenciamento é que todas as propostas de mudanças são tratadas de modo consistente e que as mudanças no documento de requisitos são feitas de maneira controlada (SOMMERVILLE, 2003). A Figura 2.5 ilustra as atividades do processo de gerenciamento de mudanças de requisitos.

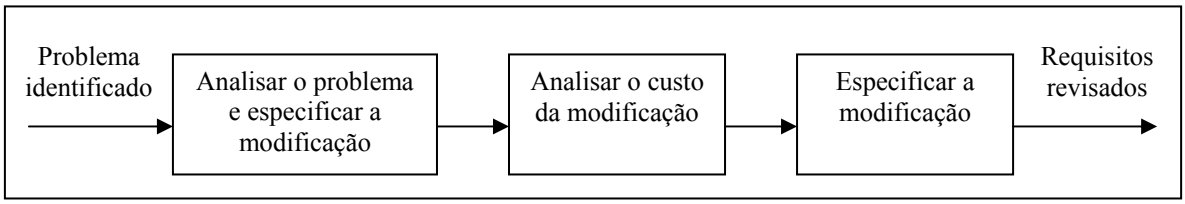


Figura 2.5 – Processo de gerenciamento de mudanças de requisitos (SOMMERVILLE, 2003).

O processo inicia com a identificação do problema com os requisitos. É realizada a análise do problema a fim de verificar a sua validade e uma proposta de mudança nos requisitos pode então ser efetuada. O efeito dessa mudança é avaliado, utilizando-se do rastreamento dos requisitos. O custo da mudança é estimado e, uma vez concluído essa análise, é tomada a decisão sobre prosseguir ou não com a alteração de requisitos. Se decidido pela modificação, os requisitos devem ser atualizados no documento de requisitos e disponibilizados para o processo de desenvolvimento do software.

2.4 - Métricas de software

Métricas são metodologias de mensuração cujo principal objetivo é medir alguma propriedade do sistema de software ou da sua especificação (ANDRADE; OLIVEIRA, 2004). Pressman (2006) define métrica como sendo uma medida quantitativa para avaliar a qualidade de atributos internos do produto, antes do produto ser desenvolvido.

As métricas podem ser classificadas em várias categorias, como apresentado a seguir. Entretanto, para obter resultados significativos, todas as abordagens de métricas requerem alguns dados históricos similares ao projeto proposto, incluindo: o domínio da aplicação, o nível de maturidade do processo de desenvolvimento, a experiência da equipe, taxas de produtividade, tipos de tecnologia e de ferramentas utilizadas e o grau de reuso (FENTON; NEIL, 2000; McGARRY et al., 2001).

- **Métricas de processo**: Medem as atividades realizadas durante todo o desenvolvimento de software.
- **Métricas de produto**: Medem os artefatos, produtos e documentos gerados pelo processo.
- **Métricas de recursos**: Medem as entidades requeridas para a execução do processo.
- **Métricas objetivas**: Podem ser quantificadas e medidas através de expressões numéricas ou representações gráficas de expressões numéricas contadas a partir do código fonte, projeto, dados de teste, e outras informações do software.
- **Métricas subjetivas**: São medidas baseadas em estimativas pessoais ou de grupo, geralmente obtidas através de conceitos como excelente, regular, bom e ruim.
- **Métricas diretas**: São aquelas que não dependem da medida de outro atributo, mas da quantificação de um fator observado no produto.
- **Métricas indiretas**: Envolvem as medidas de um ou mais atributos relacionados.

Dentro da categoria de métricas direta, McGarry et al. (2001) destacam algumas abordagens de métricas de estimativas de projeto de software como os modelos paramétricos. Esses modelos assumem que existe um relacionamento matemático entre o tamanho, o esforço, o cronograma e a qualidade. A seguir serão apresentadas as métricas de estimativa de tamanho de software que, em geral, são baseadas em modelos paramétricos.

2.4.1 - Estimativas de tamanho de software

A estimativa de tamanho de software é um processo pelo qual uma pessoa ou um grupo de pessoas estima o tamanho de um projeto de software. O tamanho de um projeto de software é uma das primeiras estimativas a ser realizada, pois dessa dimensão depende a definição de esforço, custos e prazos necessários para a definição do plano de desenvolvimento do software (McGARRY et al., 2001).

Cada projeto pode ser estimado de acordo com o tamanho físico (que são medidos através da especificação de requisitos, análise, projeto e código), com base nas funções que o usuário obtém, na complexidade do problema que o software irá resolver e no reuso do

projeto, que mede o quanto o produto será copiado ou modificado a partir de um outro produto existente (ANDRADE; OLIVEIRA, 2004).

Várias métricas de tamanho de software têm sido desenvolvidas e melhoradas desde o final da década de 1960, entre elas: Linhas de código (LOC), Análise de Pontos de Função (APF), *Bang*, *Feature Points, Mark II*, Pontos de Casos de Uso (PCU), *Full Function Point* (GARMUS; HERRON, 2000).

A Análise de Pontos de Função é uma das métricas utilizada em sistemas de informação que mede o tamanho das funções do software sob o ponto de vista do usuário, utilizando a documentação gerada durante todo o processo de desenvolvimento do software, independentemente da tecnologia usada para implementá-lo. Desenvolvida em 1979 por Allan Albrecht e posteriormente refinada pelo *International Function Point Users Group* (IFPUG) essa métrica relaciona-se diretamente às fases de análise e projeto de software (IFPUG, 2000; IFPUG, 2006; LONGSTREET, 2002).

A partir do uso da tecnologia Orientada a Objetos para o desenvolvimento de projetos de software, diversos estudos têm sido realizados propondo o mapeamento da APF para Orientação a Objetos ou desenvolvendo novas métricas, como os Pontos de Casos de Uso (PCU) (KUSUMOTO et al., 2000).

Os PCU permitem realizar estimativas no início do projeto de desenvolvimento de software com base no modelo de casos de uso (ANDRADE; OLIVEIRA, 2004). PCU é um método de estimativa de tamanho de projeto de software orientado a objetos criado por Karner (1993) e fundamentado na Análise de Pontos de Função, Mark II e no Modelo de casos de uso.

Apesar do PCU ter sido definido com base na Análise de Pontos de Função, estas métricas diferem em vários aspectos conforme mostra a Tabela 2.3. A primeira coluna apresenta as características comparadas, a segunda coluna apresenta as informações de acordo com a APF e a terceira coluna apresenta as características dos PCU.

Tabela 2.3 – Características e diferenças entre APF e PCU (adaptado de ANDRADE; OLIVEIRA, 2004).

Características	APF	PCU
Criada em	1979	1993
Padrão internacional	2002	-
Maturidade	Alto	Em crescimento
Institutos formais	IFPUG	-
Técnicas baseadas	Independente de tecnologia	Modelo de casos de uso
Visão do usuário	Sim	Sim
Complexidade no uso	Alta	Baixa
Momento de uso adequado	Análise e Projeto	Requisitos

A principal diferença entre APF e PCU está relacionada ao momento da coleta de informações para a estimativa de tamanho inicial do projeto. A APF possui resultados melhores à medida que se tem mais informação da análise e do projeto do software (tabelas, campos, associações, etc). Os PCU têm como proposta serem utilizados logo no início do ciclo de desenvolvimento, na fase de definição dos requisitos, com base nos modelos de casos de uso (DAMODARAN; WASHINGTON, 2002; GARMUS; HERRON, 2000).

2.4.1.1 - Pontos de casos de uso

Pontos de Casos de Uso (PCU) estimam o custo/prazo da função do software de acordo com a visão do usuário final e com base em uma unidade de medida, o PCU e o seu processo de contagem compõem-se de seis etapas (KARNER, 1993):

i. **Contar atores e atribuir o grau de complexidade**: Identificar e multiplicar o total de atores de acordo com o tipo de complexidade (simples, médio ou alto) pelo respectivo peso (1, 2 ou 3), conforme Tabela 2.4 e somar os produtos para obter o total de atores não ajustados.

Tabela 2.4 – Peso dos atores (adaptado de KARNER, 1993).

Complexidade do Ator	Descrição	Peso
Simples	Representa um outro sistema com Interface definida de Programas.	1
Médio	Representa um outro sistema que interage através de protocolos ou quando há interação humana através de terminal.	2
Alto	É uma pessoa que interage através de Interface Gráfica ou página Web.	3

ii. **Contar os casos de uso e atribuir o grau de complexidade**: A complexidade é baseada no número de transações. De acordo com o número de transações, multiplicar cada caso de uso pelo respectivo peso conforme Tabela 2.5.

Tabela 2.5 – Peso dos casos de uso (adaptado de KARNER, 1993).

Complexidade dos Casos de Uso	Descrição	Peso
Simples	Tem até 3 transações incluindo os passos alternativos e deve ser realizado com menos de 5 classes de análise.	5
Médio	Tem de 4 a 7 transações incluindo os passos alternativos e deve ser realizado com 5 a 10 classes de análise.	10
Alto	Tem acima de 7 transações incluindo os passos alternativos e deve ser realizado com pelo menos 10 classes de análise.	15

iii. **Calcular os PCU não ajustados:** De acordo com a seguinte fórmula: **PCU não ajustado = ∑ Peso dos Atores + ∑ Peso dos Casos de Uso**

iv. **Determinar o fator de complexidade técnica** (Tabela 2.6): Os fatores de complexidade técnica variam numa escala de 0 a 5, de acordo com o grau de dificuldade do sistema a ser construído. O valor 0 indica pouca criticidade e baixa complexidade (irrelevante para o projeto) e o valor 5 indica alta criticidade e complexidade (essencial). Após determinar o valor dos fatores, multiplicar pelo respectivo peso, somar o total dos fatores, com os respectivos pesos, e aplicar a seguinte fórmula: **Fator de complexidade técnica (FT) = 0,6 + (0,01 * ∑ fator técnico).**

Tabela 2.6 – Fatores de complexidade técnica (adaptado de KARNER, 1993).

Fatores	Peso
Sistemas distribuídos	2,0
Desempenho da aplicação	1,0
Eficiência do usuário final (on-line)	1,0
Processamento interno complexo	1,0
Reusabilidade do código em outras aplicações	1,0
Facilidade de instalação	0,5
Usabilidade (facilidade operacional)	0,5
Portabilidade	2,0
Facilidade de manutenção	1,0
Concorrência	1,0
Características especiais de segurança	1,0
Acesso direto para terceiros	1,0
Facilidades especiais de treinamento	1,0

v. **Determinar a eficiência do fator ambiental**: Os fatores ambientais indicam a eficiência do projeto e estão relacionados ao nível de experiência dos profissionais. A esses fatores, primeira coluna da Tabela 2.7, são aplicados os valores da escala de 0 a 5, onde 0 indica baixa habilidade (pouca experiência no domínio da aplicação), 3 indica média experiência e 5 indica alta experiência. Após determinar um valor para cada fator deve-se multiplicar pelo respectivo peso, segunda coluna da Tabela 2.7. Por exemplo, para o fator "dificuldade da linguagem de programação", pode-se atribuir o valor 5, ou seja, os envolvidos possuem muita dificuldade com a linguagem de programação. Em seguida, aplica-se a multiplicação pelo respectivo peso, nesse exemplo -1,0, o que resultaria em -3,0. Após aplicar os valores para todos os fatores e multiplica-los pelos respectivos pesos, deve-se aplicar a seguinte fórmula para o

cálculo do fator ambiental: **Fator de complexidade ambiental (FA) = 1,4 + (-0,03 * ∑ do fator ambiental).**

Tabela 2.7 – Fatores de complexidade ambiental (adaptado de KARNER, 1993).

Fatores	Peso
Familiaridade com o Processo de Desenvolvimento de Software	1,5
Experiência na Aplicação	0,5
Experiência com OO, na Linguagem e na Técnica de Desenvolvimento	1,0
Capacidade do Líder de Projeto	0,5
Motivação	1,0
Requisitos estáveis	2,0
Trabalhadores com dedicação parcial	-1,0
Dificuldade da Linguagem de Programação	-1,0

vi. **Calcular os PCU´s ajustados**: Esse cálculo é realizado com base na multiplicação dos PCU não ajustados, na complexidade técnica e na complexidade ambiental, através da seguinte fórmula: **PCU ajustados = PCU não ajustado * FT * FA**

Para realizar o calculo da estimativa de horas, Karner (1993) propõe a produtividade de 20 homens-hora por unidade PCU, através da seguinte fórmula: **Estimativa de horas = PCU ajustados * homens-hora por unidade de PCU.**

2.5 - Metodologia Ágil

A metodologia ágil foi desenvolvida principalmente para produzir software com qualidade e em curto espaço de tempo, a fim de atender a demanda relacionada à construção de software com rapidez (COCKBURN; HIGHSMITH, 2001).

Tornou-se popular em 2001 através da criação da "Aliança Ágil" e da publicação de seu manifesto. Dezessete especialistas da indústria de desenvolvimento de software representando os métodos: XP (*eXtreme Programming*) (BECK, 2000), Scrum (SCHWABER; BEEDLE, 2002), DSDM (*Dynamic System Development*) (STAPLETON, 1997), Crystal (COCKBURN, 2002), FDD (*Feature-Driven Development*) (COAD et al., 1999), Programação Pragmática (HUNT; THOMAS, 1999) e outros (BECK et al, 2001); estabeleceram princípios comuns compartilhados por todos esses métodos com a finalidade de criar melhores práticas de desenvolvimento de software em relação aos modelos de processo de software "tradicionais" (ABRAHAMSSON et al., 2002).

Em um esforço para vencer as fraquezas reais e percebidas da engenharia de software convencional (PRESSMAN, 2006), o manifesto ágil foi detalhado por meio de quatro valores e doze princípios que sustentam esses valores, os quais podem ser encontrados na maioria dos métodos ágeis (BECK et al., 2001). Esses valores são:

- Indivíduos e interações em vez de processos e ferramentas.
- Software executável em vez de documentação abrangente.
- Colaboração do cliente em vez de negociação de contratos.
- Respostas rápidas a modificações em vez de seguir um plano.

Em geral, esses valores determinam que o foco nas pessoas e o compartilhamento de conhecimentos entre elas é o fator chave para que o projeto obtenha sucesso. Usar software operacional permite verificar com rapidez os resultados produzidos e fornecer "*feedback"* rápido. A documentação deve ser curta e relatar somente o essencial para o projeto. O contato com o cliente é fundamental para o sucesso do projeto sendo necessário que haja constante troca de informações, idéias e pensamentos entre a equipe de desenvolvimento e o cliente. Ter a habilidade de assimilar todo tipo de modificação durante a execução do projeto, desde mudanças na elicitação de requisitos, quanto mudanças na equipe de desenvolvimento e tecnologia a ser utilizada (ABRAHAMSSON et al., 2002; HIGHSMITH; COCKBURN, 2001).

2.5.1 - Princípios e abordagens da metodologia ágil

Os doze princípios que sustentam a metodologia ágil são: *i)* a maior prioridade é satisfazer o cliente desde o início com a entrega rápida e continua de software operacional; *ii)* fazer mudanças nos requisitos do sistema é bem vinda em qualquer fase do desenvolvimento; *iii)* fornecer versões do software com freqüência, as quais funcionem adequadamente; *iv)* clientes e desenvolvedores devem trabalhar juntos ao longo de todo o projeto; *v)* fazer o projeto com pessoas motivadas, disponibilizar o ambiente e a infra-estrutura necessária e confiar que elas possam fazer o trabalho; *vi)* o método mais eficiente e efetivo para obter informações sobre o projeto é pela conversa face a face com o cliente; *vii)* a principal medida de progresso é o software funcionando; *viii)* processos ágeis promovem um desenvolvimento sustentável, cliente e desenvolvedores devem descobrir seu ritmo de trabalho e mantê-lo constantemente; *ix)* atenção contínua a boas práticas e bom projeto promovem a agilidade; *x)*

simplicidade de trabalho é essencial; *xi)* permitir que equipes da própria organização participem da definição da arquitetura, requisitos e projeto do sistema; *xii)* em intervalos regulares, a equipe refletirá em como se tornar mais eficiente, podendo ajustar o seu comportamento conforme as necessidades.

De posse dos valores e princípios, um conjunto de abordagens deve ser considerado no processo ágil de desenvolvimento de software: *i)* a comunicação entre indivíduos com constante interação e cooperação entre eles (AMBLER, 2007b; HIGHSMITH; COCKBURN, 2001); *ii)* facilidade de incorporar e adaptar mudanças de requisitos durante todo o desenvolvimento; *iii)* fornecer versões de software funcional e operacional em intervalos freqüentes do ciclo de vida de projeto (MILLER, 2001); *iv)* desenvolvedores e representantes de usuários devem ser bem informados e autorizados a tomar decisões, com base na documentação suficiente para entender e criar o projeto (COCKBURN; HIGHSMITH, 2001) e *v)* a equipe de desenvolvimento deve ser pequena, normalmente não ultrapassa dez indivíduos (ABRAHAMSSON et al., 2002).

2.5.2 - Processos ágeis

Baseado nos valores, princípios e nas abordagens descritas acima, Abrahamsson et al. (2002), indicam que para um processo ser ágil, esse processo deve ser: incremental, cooperativo, direto e adaptável:

- **Incremental**: versões pequenas do software, com ciclos de desenvolvimento curtos e rápidos;
- **Cooperativo**: clientes e desenvolvedores trabalhando constantemente juntos e com boa comunicação;
- **Direto e Simples**: o processo fica bem documentado, fácil de aprender e de modificar:
- **Adaptável**: capaz de atender mudanças a qualquer momento do desenvolvimento;

A metodologia ágil aborda o processo de desenvolvimento de software de maneira diferente dos modelos preconizados pela engenharia de software. A principal diferença está na forma como são tratadas as rápidas mudanças durante o desenvolvimento (COSTA et al., 2005). Essas mudanças podem ser: de mercado, de requisitos, de sistemas, de tecnologias de

implementação e de equipes de projeto dentro do período de desenvolvimento (COCKBURN; HIGHSMITH, 2001).

As metodologias tradicionais têm o enfoque na estratégia de previsibilidade, tentam planejar grande parte do software em muitos detalhes com bastante antecedência, sendo resistentes a mudanças. Já a metodologia ágil tem o enfoque na adaptabilidade, isto é, mudanças de requisitos durante o desenvolvimento são bem-vindas e tendem a ajustar o desenvolvimento do software e são orientadas às pessoas, consideram a habilidade da equipe de desenvolvimento com constante interação entre os envolvidos no projeto (FOWLER, 2005).

O foco na adaptabilidade ocorre porque de acordo com os valores e práticas ágeis, normalmente no início do projeto de desenvolvimento, nem clientes nem desenvolvedores possuem total conhecimento dos requisitos e funcionalidades do sistema. Dessa forma, os envolvidos com o projeto aceitam a idéia de que os requisitos do software serão conhecidos à medida que o processo de desenvolvimento avança (AMBLER, 2007b; HIGHSMITH, 2002).

Por ser orientada a pessoas a metodologia ágil destaca a comunicação efetiva entre clientes e equipe de desenvolvimento. Essa interação se faz necessária durante todo o desenvolvimento do software, deve ser realizada face a face e ser vista como um canal de comunicação (DAVIES, 2005). Para que essa comunicação seja eficiente, é enfatizado que o cliente deve estar "*on-site*", ou seja, deve fazer parte da equipe de desenvolvimento (DAVIES, 2005). Dessa forma, o cliente deve disponibilizar um representante capaz de tomar decisões tais como: priorização de requisitos, dúvidas relativas ao domínio do sistema e direcionamento do desenvolvimento a partir de constante "*feedback"* (COCKBURN; HIGHSMITH, 2001). Esse envolvimento do cliente deve ocorrer durante todo o desenvolvimento do software (EBERLEIN; LEITE, 2002) e é importante para que a equipe de desenvolvimento compreenda os detalhes do que ele realmente necessita (DAVIES, 2005).

2.5.3 - Métodos ágeis

Para apoiar os valores e práticas propostas pela abordagem ágil, diversos métodos ágeis têm surgido nos últimos anos. Abrahamsson et al. (2002), apresentam nove métodos ágeis dividindo-os em diferentes focos. Os métodos XP (*eXtreme Programming*), Programação Pragmática (PP) e Modelagem Ágil (AM) têm enfoque nas práticas da metodologia ágil; o método Scrum trata de gerenciamento; e os métodos Desenvolvimento

Adaptativo de Software (ASD), Crystal Methods, Método de Desenvolvimento Dinâmico de Sistema (DSDM), Desenvolvimento Guiado por Características (FDD)) têm enfoque em processo. Porém, alguns métodos ágeis não fornecem cobertura completa no ciclo de vida do software, ou seja, desde a concepção do projeto até o sistema em uso, e necessitam de técnicas e ferramentas efetivas para apoiar a realização das atividades, práticas e a geração dos artefatos (ABRAHAMSSON et al., 2002).

2.6 - Padrões em desenvolvimento de software

Os primeiros conceitos de padrões surgiram na área de arquitetura de construções e planejamento urbano com os trabalhos de Christopher Alexander. Ele e seus colaboradores definem que um padrão é a descrição de um problema que ocorre repetidamente em um ambiente descrevendo uma solução para esse problema de tal forma que seja possível usar essa solução milhões de vezes (ALEXANDER et al., 1977; ALEXANDER, 1979). Assim, um padrão deve expressar o relacionamento entre um determinado contexto, um problema que se repete periodicamente (com objetivos e restrições) e uma solução recorrente para esse problema nesse contexto (APPLETON, 2000).

Segundo Lea (1997), o uso desse conceito é aplicado em várias outras disciplinas, incluindo a Engenharia de Software. Os padrões em desenvolvimento de software são utilizados para tratar de problemas periódicos enfrentados durante todo o seu processo de desenvolvimento (APPLETON 2000). Seu uso teve início em um trabalho de Cunningham e Beck (1987), os quais decidiram utilizar algumas das idéias de Alexander (1979) para direcionar novos programadores de Smaltalk. Posteriormente, Coplien (1991) introduziu a linguagem de padrões para C++. Em meados de 1994, com o lançamento do livro "*Design Patterns: Elements of Reusable Object-Oriented Software*" de Eric Gamma, Richard Helm, Ralph Johnson e John Vlissides, conhecidos por GoF - "Gang of Four", os padrões de software começaram a ser largamente discutidos.

Padrões não são nem apenas prática e nem apenas teoria, mas ambos. Quando teoria e prática se encontram, reforçam e complementam um ao outro, mostrando que a estrutura descrita por um padrão deve ser útil e usável (APPLETON, 2000).

Porém, nem toda solução, algoritmo ou boas práticas, constituem um padrão. Se uma solução apresentar todos os requisitos de um padrão, ele será considerado um proto-padrão (candidato a padrão) até que tenha sido comprovada a sua utilização em fenômenos periódico,

em no mínimo três diferentes sistemas, o que é freqüentemente chamado de “regra dos três” (APPLETON, 2000). Além disso, essa solução deve passar por rigorosos exames e ser revista por pessoas diferentes antes de ser declarada um padrão.

2.6.1 - Elementos de um padrão

Padrões devem ser documentados textualmente e geralmente são organizados em seções ou elementos que definem o que se chama de formato do padrão. Apesar do uso de diferentes formatos de padrões, comumente encontramos em um padrão elementos essenciais como os descritos abaixo (APPLETON, 2000). Esses formatos são apenas sugestões do que cada padrão deve conter e não há obrigatoriedade de usar todos os elementos em todos os padrões (VLISSIDES, 1998):

- **Nome**: deve ser significativo e geralmente referencia o problema ou a solução. Pode ser uma única palavra ou uma frase curta. Às vezes um padrão pode ter mais de um nome, geralmente usado ou reconhecível na literatura. Nesse caso é prática comum para documentar esses apelidos ou sinônimos por Também Conhecido Como.

- **Problema**: descreve o problema a ser resolvido apresentando a sua intenção, suas metas e objetivos alcançáveis dentro do contexto dado.

- **Contexto:** descreve em que circunstâncias o problema surge e a aplicabilidade do padrão. Mostra a configuração inicial do sistema antes do padrão ser aplicado.

- **Forças:** revelam as complexidades de um problema. Descreve as considerações positivas e negativas a serem avaliadas a fim de definir por quê (e se) a solução proposta deve ser empregada.

- **Solução:** descreve claramente o que é necessário para resolver o problema. A solução não só descreve a estrutura estática, mas também o comportamento dinâmico. A estrutura estática mostra a forma e a organização do padrão, e a dinâmica, mostra o comportamento, ou seja, o que o padrão faz. A descrição da solução do padrão pode indicar diretrizes para se evitar problemas já ocorridos.

- **Exemplo:** um ou mais exemplos da aplicação do padrão. Exemplos ajudam o leitor a entender o uso do padrão e sua aplicabilidade, mostrando como o padrão é aplicado e como transforma um determinado contexto. Pode ser completado por uma implementação para mostrar como a solução poderia ser conseguida.

- **Contexto Resultante:** descreve o estado ou configuração do sistema depois da aplicação do padrão, incluindo as conseqüências. Às vezes é chamado de "resolução de forças" porque descreve quais forças foram atendidas e quais não, e quais padrões podem ser aplicados. A documentação do contexto resultante ajuda a relacionar o contexto inicial de outros padrões a fim realizar uma tarefa ou projeto, já que um único padrão é freqüentemente só um passo para alcançar um projeto maior.

- **Análise Racional:** explicação das regras ou passos do padrão, em termos de como e porque ele soluciona as forças para alcançar os objetivos desejados. Esse elemento do padrão mostra como ele funciona, porque funciona, e porque é "bom".

- **Padrões Relacionados:** descreve como o padrão está relacionado com outros padrões que se referem ao mesmo problema e compartilham forças em comum. Pode referenciar outros padrões usados em conjunto ou outras soluções para o problema, ou ainda, variações do padrão. Podem ser: padrões antecessores (cuja aplicação conduz a esse), e padrões sucessores (cuja aplicação segue a desse padrão). Em alguns formatos é apresentado como "**Veja Também**".

- **Usos Conhecidos:** descreve ocorrências conhecidas do padrão e sua aplicação dentro de sistemas existentes. Isto ajuda a validar um padrão verificando que realmente é uma solução comprovada de um problema que ocorre periodicamente.

A Tabela 2.8 apresenta alguns formatos utilizados para descrever padrões. As colunas apresentam o nome pelo qual os padrões são conhecidos e as linhas apresentam os elementos que são utilizados na sua descrição.

Tabela 2.8 – Formatos dos padrões de software.

GAMMA et al., 1994	APPLETON, 2000	COPLIEN, 1995	BUSCHMANN et al., 1996
• Nome • Classificação • Intenção • Também Conhecido Como • Motivação • Aplicabilidade • Estrutura • Participantes • Colaborações • Implementação • Código Exemplo • Conseqüências • Usos Conhecidos • Padrões Relacionados	• Nome • Problema • Contexto • Força • Solução • Exemplo • Contexto Resultante • Análise Racional • Padrões Relacionados • Usos Conhecidos	• Nome • Problema • Contexto • Forças • Solução • Esboço • Contexto Resultante • Argumentação	• Nome • Problema • Contexto • Também Conhecido Como • Exemplo • Solução • Estrutura • Dinâmica • Implementação • Exemplo Resolvido • Variantes • Usos Conhecidos • Conseqüências • Veja Também

Para a descrição de padrões de software neste trabalho, foi utilizado um formato reduzido contendo os elementos nome, problema e solução.

2.6.2 - Catálogos, sistemas e linguagens de padrões

Padrões podem ser agrupados de acordo com o grau das suas estruturas e interação, podendo ser catálogos de padrões, sistemas de padrões ou linguagens de padrões. Catálogos de padrões representam um conjunto de padrões relacionados com algumas referências cruzadas entre si. Um sistema de padrões representa um conjunto de padrões com características semelhantes que trabalham juntos, porém não precisam ser computacionalmente completos (morfologicamente e funcionalmente) (BUSCHMANN et al., 1996).

Linguagem de padrões é uma coleção de padrões associados que apóiam uns aos outros para transformar requisitos e restrições numa arquitetura completa (COPLIEN, 1998) guiando tanto o desenvolvimento de um novo sistema como também a evolução de um sistema já existente (ANDRADE, 2001). Essas linguagens devem ser aplicadas para solucionar problemas que um padrão individual não pode dado o tamanho ou a complexidade do problema em um dado contexto (APPLETON, 2000), e devem ser completas morfologicamente (seus padrões se combinam para formar uma estrutura sem lacunas) e funcionalmente (quaisquer novas forças introduzidas pelos padrões são resolvidas por eles mesmos) (HANMER, 2003).

Hanmer (2003) sugere que uma linguagem de padrão deve conter: *i)* a intenção da linguagem com uma descrição curta do que é pretendido que a linguagem realize; *ii)* um diagrama ou grafo que mostra um exemplo de como os padrões se relacionam um ao outro; *iii)* uma descrição de como a linguagem de padrão é morfologicamente e funcionalmente completa; e *iv)* identificação dos padrões que compõem a linguagem.

A fim de facilitar a utilização dos padrões de software, pode-se classificá-los em diversas categorias, como apresentado na próxima Subseção.

2.6.3 - Categorias de padrões

- **Padrões de Análise**: têm o foco no conhecimento do domínio da aplicação. Seu objetivo é concentrar-se na fase de análise (no ciclo de desenvolvimento de software) com um modelo semântico de alto nível que endereça estruturas de negócio conceituais em lugar de implementações de software. Esse modelo afeta a flexibilidade e reusabilidade do sistema resultante (FOWLER; KENDALL, 2000).

- **Padrões de Processo:** são coleção de técnicas, ações, e tarefas (atividades) que descrevem comportamentos comprovados e de sucesso para o desenvolvimento de software. Quando aplicados junto a outros padrões, podem ser usados para construir um processo de software para organizações. Uma característica importante dos padrões de processo é que é possível a sua aplicação para todos os aspectos de desenvolvimento. Um único padrão de processo pode variar de uma visão de alto nível, de como são desenvolvidas as aplicações, a uma visão mais detalhada de uma parte específica do processo de desenvolvimento do software (AMBLER, 1998).

- **Padrões de Arquitetura:** expressam uma organização estrutural para o sistema de software (APPLETON, 2000). Fornecem uma estrutura básica para sistemas de software e provêem um conjunto de subsistemas pré-definidos que especifica suas responsabilidades e inclui regras para organizar os relacionamentos entre eles (BUSCHMANN et al., 1996).

- **Padrões de Projeto:** descrevem uma estrutura para solucionar problemas de projeto de software (BUSCHMANN et al., 1996). Esta é a categoria de padrões mais encontrada na literatura, em especial para a orientação a objeto (GAMMA et al., 1994).

- **Padrões de Idiomas:** é um padrão considerado de baixo-nível, específico para uma linguagem de programação. Descrevem regras gerais de estilo de programação e soluções de como implementar partes específicas de determinada linguagem (BUSCHMANN et al., 1996).
- **Padrões de Testes:** descrevem situações que testadores de software podem conseguir reusar quando abordam o teste de algum sistema novo ou revisado (DELANO; RISING, 1997; RISING, 2000).

Essa classificação de padrões não é rigorosa e pode haver padrões que se encaixam em mais do que uma categoria de modo a promover sua recuperação e uso (BRAGA et al., 2001). Os padrões de requisitos e os organizacionais são apresentados no Capítulo 3.

2.7 - Considerações Finais

Este capítulo apresentou conceitos de processo e modelagem de processos, os quais representam a maneira de construir e modelar sistemas de software. O SPEM (OMG, 2005) foi apresentado como um meta-modelo para representar processos de desenvolvimento.

Foram apresentados também conceitos de engenharia de requisitos, que é a fase inicial do processo de desenvolvimento de software (SOMMERVILLE, 2003) e corresponde ao processo de aquisição, refinamento e verificação das necessidades do cliente para um sistema de software (IEEE, 1998a). O conjunto de atividades de engenharia de requisitos, bem como a maneira com que essas atividades são realizadas ao longo do desenvolvimento de software também foi apresentado.

Embora não seja solução à prova de falhas, o processo de engenharia de requisitos é uma abordagem sólida para assegurar que as especificações de requisitos de software estejam de acordo com as necessidades dos clientes e que satisfarão suas expectativas (PRESSMAN, 2006). Durante a realização das atividades da engenharia de requisitos, os requisitos do sistema podem ter os tempos estimados para serem atendidos. Dessa forma, métricas de tamanho de software também foram abordadas neste Capítulo, com ênfase na métrica de Pontos de Casos de Uso.

Ainda neste capítulo, foi apresentada a metodologia ágil de desenvolvimento de software, que é uma estratégia emergente a qual enfatiza aspectos humanos e respostas rápidas a mudanças contínuas durante todo o processo (FAVARO, 2002). Os valores e

práticas que norteiam os métodos ágeis foram apresentados com ênfase na adaptabilidade das modificações em requisitos durante as iterações do desenvolvimento do software e participação ativa do cliente.

Conceitos sobre padrões de software também foram apresentados, bem como foram descritos alguns elementos e formatos de padrões e como é a interação das estruturas dos padrões de software.

A importância desses estudos está no conhecimento das técnicas e ferramentas usadas pelos engenheiros de software, visando à aplicação desses conhecimentos neste trabalho.

Capítulo 3 – Padrões de requisitos e padrões organizacionais

3.1 - Considerações Iniciais

Os padrões de software têm se tornado um instrumento fundamental de garantia de qualidade e produtividade no desenvolvimento de projetos de software. Além de fornecer um vocabulário comum para expressar soluções, os padrões permitem a criação de uma biblioteca de soluções para ajudar na resolução de problemas recorrentes incentivando a documentação e reúso de "boas práticas" nos modelos de processo de desenvolvimento de software (ANDRADE et al., 2007).

Com o intuito de facilitar o reúso dos padrões de software em modelos de processo, cada etapa desses modelos pode ser auxiliada por uma categoria de padrão. Porém não deve haver ortogonalidade nessa relação, pois podem existir categorias de padrões que se encaixam em mais do que uma etapa nos modelos de processo de desenvolvimento de software (ANDRADE, 2001).

No domínio da engenharia de requisitos padrões de requisitos e padrões organizacionais podem auxiliar engenheiros de requisitos a realizarem as atividades da engenharia de requisitos (HAGGE; LAPPE, 2005; SHOEMAKER, 2007). Esses padrões documentam as necessidades dos usuários e descrevem o comportamento funcional de sistemas de software em alto nível de abstração (ANDRADE, 2001). Padrões de requisitos auxiliam a realização de atividades de elicitação, análise, especificação, validação e gerenciamento de requisitos, e padrões organizacionais têm o objetivo de representar estruturas utilizadas com sucesso em organizações que desenvolvem software e aprimorar a comunicação entre integrantes da equipe (COPLIEN, 1995; KOTULA, 1998).

Além desta Seção, este capítulo apresenta alguns padrões de requisitos na Seção 3.2 e alguns padrões organizacionais na Seção 3.3. Esses padrões são apresentados em um formato reduzido, com os elementos: **nome**, **problema** e **solução**. Para reutilizar as práticas propostas pelos padrões de requisitos e organizacionais neste trabalho, que está enfocado na engenharia de requisitos dos métodos ágeis, algumas modificações foram necessárias, como apresenta a Seção 3.4. Por fim, na Seção 3.5 são apresentadas às considerações finais do capítulo.

3.2 - Padrões de requisitos

Possibilitam a transferência de conhecimento entre engenheiros de requisitos e reutilizam práticas e técnicas já comprovadas para auxiliar na realização das atividades da engenharia de requisitos (FERDINANDI, 2002; HAGGE; LAPPE, 2005).

Lappe et al. (2004) apresentam um catálogo com quatorze padrões para a engenharia de requisitos. Cinco padrões desse catálogo são utilizados por este trabalho e são apresentados na Tabela 3.1. A primeira coluna apresenta o nome do padrão e na segunda o padrão é descrito resumidamente com os elementos problema e solução. Os outros padrões desse catálogo não são utilizados por serem característicos de equipes distribuídas, controles formais de modificações e ausência do cliente para tomadas de decisão, perdendo o enfoque da metodologia ágil.

Tabela 3.1 – Padrões de Requisitos (adaptado de LAPPE et al., 2004).

Nome do Padrão	Resumo do Padrão
Empregue um Engenheiro de Requisitos como encarregado *(Employ a Requirements Engineer as a Care Taker)*	**Problema**: Envolver os diversos membros do projeto e selecionar um para desempenhar a tarefa de gerenciar requisitos.
	Solução: Eleja dentre todos os membros disponíveis, um que possua conhecimento dos métodos de engenharia de requisitos. Essa pessoa, chamada de engenheiro de requisitos, deve tomar as decisões referentes ao gerenciamento dos requisitos.
Crie uma diretriz de especificação acompanhando o trabalho de um analista *(Create a Specification Guideline by Tracking How an Analyst Work)*	**Problema**: Especificações de requisitos são desenvolvidas de acordo com habilidades individuais das pessoas em cada projeto, dificultando que esses requisitos sejam compreendidos por diferentes envolvidos no projeto.
	Solução: Crie uma diretriz para guiar a maneira de realizar a especificação de requisitos. Essa diretriz deve acompanhar os passos dos analistas de requisitos para fazerem suas especificações.
Agrupe requisitos às características *(Bundle Requirements to Features)*	**Problema**: Os envolvidos no projeto necessitam entender os requisitos do sistema em diferentes níveis de abstração. Gerentes de projeto necessitam de requisitos de alto-nível para o entendimento da visão do projeto em um primeiro passo, enquanto que desenvolvedores precisam de requisitos técnicos detalhados.
	Solução: Agrupe os requisitos do sistema de acordo com suas características. Identifique as semelhanças efetivas dos requisitos e reúna-os em uma lista de características. O resultado é uma estrutura hierárquica que decompõe o produto em características que agem como ligação entre requisitos de alto-nível e detalhados.
Forneça declaração de objetivos para cada requisito *(Provide Statement of Objective With Each Requirement)*	**Problema**: Durante a modelagem do projeto do sistema, verifica-se que existem requisitos incoerentes, os quais comprometem a execução do projeto.
	Solução: Encontre solução exeqüível para os requisitos e negocie com o Cliente essa solução.
Detalhe a especificação escrevendo casos de teste *(Detail the Specification by Writing Test)*	**Problema**: A especificação de requisitos não está clara o suficiente ou necessita ser alterada, porém os requisitos já estão congelados para a implementação.
	Solução: Deixe a especificação de requisitos como está e detalhe nos casos de teste as soluções para a especificação.

Bramble et al., (2002) apresentam a linguagem de padrões para criar casos de uso efetivos (*Patterns for Effective Use Cases*) (trinta e um padrões) e Coram (1996) apresenta uma linguagem de padrões para a construção e demonstração de software (*Demo Prep*) (sete padrões). Alguns desses padrões podem ser relacionados à categoria de padrões de requisitos, pois possuem características que orientam a realização das atividades da engenharia de requisitos. De acordo com Bramble et al., (2002), um dos maiores causadores dos problemas relacionados a requisitos é a comunicação entre clientes e desenvolvedores. Para minimizar esse problema, casos de uso e protótipos podem ser utilizados para especificar, validar e mostrar os requisitos do sistema ao cliente. Os casos de uso capturam e modelam os cenários que o sistema deve tratar e os protótipos permitem demonstrar ao cliente o que está sendo desenvolvido.

Três padrões para criar casos de usos efetivos (BRAMBLE et al., 2002) e um padrão para construção e demonstração de software (CORAM, 1996) são utilizados por este trabalho e estão apresentados, respectivamente, nas Tabela 3.2 e Tabela 3.3. A primeira coluna das tabelas apresenta o nome do padrão e a segunda a descrição resumida com os elementos problema e solução proposta por eles.

Tabela 3.2 – Padrões para criação de casos de uso efetivos (adaptado de BRAMBLE et al., 2002).

Nome do Padrão	Resumo do Padrão
Amplitude antes de profundidade *(Breadth Before Depth)*	**Problema**: Detalhar os casos de uso no início do projeto, antes de conhecer os objetivos do sistema.
	Solução: Desenvolva primeiramente uma visão geral dos casos de uso e à medida que o desenvolvimento avança adicionar os detalhes.
Desenvolvimento espiral *(Spiral Development)*	**Problema**: Desenvolver todos os casos de uso de uma única vez dificulta o acréscimo e modificações em requisitos.
	Solução: Desenvolva os casos de uso iterativamente, com cada iteração aperfeiçoando a precisão de um conjunto de casos de uso.
Equipe de escrita pequena *(Small Writing Team)*	**Problema**: Dificuldade de consenso quando muitas pessoas descrevem os casos de uso.
	Solução: Na escrita dos casos de uso, limite o número de pessoas para dois ou três.

Tabela 3.3 – Padrões para construção e demonstração de software (adaptado de CORAM, 1996).

Nome do Padrão	Resumo do Padrão
Identificação de elemento *(Element Identification)*	**Problema**: Selecionar as funções do sistema que preocupam o cliente e que devem ser apresentadas a ele para validação.
	Solução: Conversar com o cliente e identificar as funções do sistema que o preocupam. Essas funções devem ser demonstradas a ele por meio de protótipos.

3.3 - Padrões organizacionais

Padrões organizacionais podem ser usados não apenas para auxiliar no entendimento de organizações já existentes, mas também para modelar uma nova organização, seu processo de desenvolvimento de software e aprimorar a comunicação entre integrantes da equipe (COPLIEN, 1995). De acordo com Harrison (1996), um conjunto de pessoas não pode ser agrupadas e esperar que elas trabalhem como equipe espontaneamente, é necessário organizar a forma de trabalho dessas pessoas.

Coplien e Harrison (2004) descrevem, por meio de quatro linguagens, noventa e dois padrões que combinam estruturas organizacionais com práticas de desenvolvimento de software. Esses padrões descrevem como conduzir uma organização e o seu processo de desenvolvimento de software abordando a estruturação do trabalho por meio de cronogramas, processos e tarefas, e a estrutura de relacionamentos dos papéis na organização (COPLIEN; HARRISON, 2004). Quatro padrões dessas linguagens são utilizados por este trabalho e apresentados na Tabela 3.4. A primeira coluna apresenta o nome do padrão e a segunda apresenta um resumo contendo o problema e a solução proposta para ele.

Tabela 3.4 – Padrões organizacionais (adaptado de COPLIEN; HARRISON, 2004)

Nome do Padrão	Resumo do Padrão
Envolver o cliente *(Engage Customers)*	**Problema**: Modificações no sistema podem ocorrer a qualquer momento e o cliente necessita estar presente para as tomadas de decisões.
	Solução: Envolva o cliente com o projeto de forma que ele faça parte da equipe de desenvolvimento do sistema.
Unidade de Propósito *(Unity Of Purpose)*	**Problema**: Os membros da equipe estão começando a trabalhar junto e possuem diferentes conhecimentos e experiência, deve-se constituir entre eles uma visão comum sobre o projeto.
	Solução: O líder do projeto deve expor, para todos os membros da equipe, uma visão comum e o propósito do projeto.
Equipe auto selecionadora *(Self Selecting Team)*	**Problema**: Indicar pessoas para desempenharem os papéis do projeto, selecionando a equipe de desenvolvimento.
	Solução: As pessoas disponíveis para o projeto devem opinar na escolha de suas equipe e papéis que querem trabalhar.
Cenários definem o problema *(Scenarios Define Problem)*	**Problema**: Descrever os requisitos funcionais do sistema e validá-los com o cliente.
	Solução: Utilize casos de uso para descrever e apresentar os requisitos funcionais do sistema ao cliente.

3.4 - Padrões adaptados para serem utilizados no processo proposto

Os padrões de requisitos de Lappe et al. (2004), os de Bramble et al. (2002) e o de Coram (1996) e os padrões organizacionais de Coplien e Harrison (2004), descrevem soluções para auxiliar o processo de desenvolvimento do software, desde a interação entre pessoas até a aquisição e completo controle dos requisitos do sistema. Para o efetivo uso desses padrões neste trabalho foram necessárias modificações em dois desses padrões. Essas modificações foram realizadas para que esses padrões pudessem ser utilizados no contexto da engenharia de requisitos com métodos ágeis e estão descritos a seguir:

- Padrão *Identificação de elemento*, apresentado na Tabela 3.3, menciona que as funções que preocupam o cliente devem ser demonstradas e validadas por meio de protótipos. Entretanto, de acordo com Bramble et al. (2002), modelos de casos de uso é uma técnica efetiva para especificar os requisitos do sistema e proporciona, a todos os envolvidos com o projeto, entendimento dos requisitos a serem atendidos facilitando a validação desses requisitos pelo cliente. No contexto do desenvolvimento ágil o cliente também é participante ativo da priorização dos requisitos, os quais são atendidos nas entregas de versões de software funcional (AMBLER, 2007b). Assim, a validação dos requisitos do sistema ocorre junto à realização da especificação dos requisitos (cliente participa dessa atividade) e a demonstração das funções prioritárias, as quais preocupam os clientes (cliente participa da priorização dos requisitos), ocorrerá por meio da entrega rápida de software funcional.

- Padrão *Unidade de Propósito*, apresentado na Tabela 3.4, menciona que deve ser estabelecido um consenso, sobre o propósito do projeto, entre os membros da equipe que estão começando a trabalhar juntos. Entretanto, independente da experiência e tempo de trabalho juntos dos integrantes da equipe, todos eles devem ter conhecimento uniforme dos requisitos a serem atendidos no desenvolvimento do sistema. Assim, antes das atividades de projeto de software é necessário que o Engenheiro de Requisitos apresente a todos os envolvidos no projeto os requisitos a serem atendidos (AMBLER, 2007b).

Todos os padrões utilizados neste trabalho, os modificados e os originais, foram adicionados do elemento denominado **Passos da Solução**. Esse elemento é utilizado para apresentar detalhadamente os passos a serem seguidos para alcançar a solução proposta por cada padrão. Os padrões são apresentados nos Quadros de 3.1 a 3.13, com o **Nome** do padrão,

a descrição do **Problema** que o padrão pretende resolver, a **Solução** para o problema e os **Passos da Solução** proposta. Em particular, o padrão *Agrupe requisitos a característica*, apresentado no Quadro 3.9, possui também o elemento **Exemplo** para melhor apresenta-lo:

Quadro 3.1 – Padrão Amplitude antes de profundidade (adaptado de BRAMBLE et al., 2002).

Nome: Amplitude antes de profundidade
Problema: Descrever detalhadamente os objetivos do sistema com casos de uso.
Solução: Descrever os objetivos do sistema de forma geral e por meio de casos de uso.
Passos da Solução: 1. Identificar as fronteiras do sistema, os atores envolvidos com o sistema e os casos de uso realizados por esses atores. As seguintes perguntas devem ser feitas ao Cliente: Quais grupos de usuários necessitam de ajuda do sistema para executar tarefas? Quais grupos de usuários são necessários para executar as funções básicas do sistema? Quais grupos de usuários deverão executar funções secundárias, como manutenção e administração do sistema? O sistema interagirá com algum sistema externo de hardware ou software? Quais são as principais tarefas a serem executadas pelo sistema? 2. Criar o diagrama de casos de uso com os objetivos do sistema. 3. Descrever os casos de uso de forma geral com: Nome que identifique o caso de uso; Resumo das ações do caso de uso; Atores envolvidos com o caso de uso e Fluxo principal do caso de uso, para que forneçam a visão dos objetivos do sistema. Os casos de uso serão detalhados na especificação dos requisitos.

Quadro 3.2 – Padrão Envolver o cliente (adaptado de COPLIEN; HARRISON, 2004).

Nome: Envolver o cliente
Problema: Obter o comprometimento do cliente no projeto.
Solução: Envolver o cliente com o projeto de forma que ele faça parte da equipe do projeto.
Passos da Solução: 1. Comprometer o Cliente com o projeto de forma que faça parte da equipe. Ele deve participar ativamente das atividades de elicitação, especificação e priorização de requisitos e nas tomadas de decisões de modificações em requisitos do início ao fim do projeto. Isso facilitará também a interação entre o cliente e os desenvolvedores do sistema.

Quadro 3.3 – Padrão Equipe auto selecionadora (adaptado de COPLIEN; HARRISON, 2004).

Nome: Equipe auto selecionadora
Problema: Indicar os atores para desempenharem os papéis do projeto.
Solução: Permitir que as pessoas opinem na escolha dos papéis a desempenhar no projeto.
Passos da Solução: 1. Reunir todas as pessoas envolvidas no projeto e apresentar os objetivos do sistema a eles. 2. Permitir que as pessoas opinem sobre quais papéis querem desempenhar. 3. Indicar quais profissionais irão desempenhar os papéis do projeto. Levar em consideração os interesses das pessoas e as habilidades de cada um para desempenharem os papéis. a. Para o papel de Engenheiro de Requisitos, use o padrão *Empregue um Engenheiro de Requisitos como encarregado*, apresentado no Quadro 3.4.

Quadro 3.4 – Padrão Empregue um Engenheiro de Requisitos como encarregado (adaptado de LAPPE et al., 2004).

Nome: Empregue um Engenheiro de Requisitos como encarregado
Problema: Indicar um ator para desempenhar o papel de Engenheiro de Requisitos.
Solução: Atribuir a uma pessoa com conhecimentos em engenharia de requisitos, o papel de Engenheiro de Requisitos.
Passos da Solução: 1. Indicar uma pessoa da equipe para desempenhar o papel de Engenheiro de Requisitos. Essa pessoa deve ter as seguintes características: habilidades de comunicação, conhecimentos no domínio do negócio e da tecnologia envolvidos no projeto, e conhecimento nas técnicas de elicitação, análise e especificação de requisitos.

Quadro 3.5 – Padrão Desenvolvimento espiral (adaptado de BRAMBLE et al., 2002).

Nome: Desenvolvimento espiral
Problema: Desenvolver todos os requisitos do sistema.
Solução: Aperfeiçoar os requisitos iterativamente, aumentando progressivamente a precisão de um conjunto de requisitos em cada iteração.
Passos da Solução: 1. Desenvolver os requisitos do sistema de forma iterativa e evolutiva. Nos primeiros ciclos de elicitação e especificação deve-se realizar os requisitos relacionados às funções gerais do sistema. À medida que os requisitos são conhecidos deve-se desenvolver as funções especializadas do sistema. 2. Repetir a elicitação e especificação de requisitos até que os requisitos conhecidos estejam detalhados.

Quadro 3.6 – Padrão Crie uma diretriz de especificação acompanhando o trabalho de um analista (adaptado de LAPPE et al., 2004).

Nome: Crie uma diretriz de especificação acompanhando o trabalho de um analista
Problema: Especificar os requisitos do sistema de acordo com as habilidades individuais das pessoas em cada projeto.
Solução: Elaborar diretrizes para conduzir a especificação de requisitos.
Passos da Solução: 1. Especificar os requisitos funcionais de acordo com o padrão *Cenários definem o problema*, apresentado no Quadro 3.7. 2. Especificar os requisitos não funcionais por meio de descrição textual. Deve-se atribuir um número seqüencial seguido de um nome que identifique cada requisito e descrevê-los detalhando todas as suas ações, restrições e características. 3. Agrupar os requisitos do sistema de acordo com suas características para melhorar a visibilidade e organização dos requisitos especificados. Use o padrão *Agrupe requisitos a características*, apresentado no Quadro 3.9.

Quadro 3.7 – Padrão Cenários definem o problema (adaptado de COPLIEN; HARRISON, 2004).

Nome: Cenários definem o problema
Problema: Especificar os requisitos funcionais do sistema.
Solução: Especificar os requisitos funcionais por meio de casos de uso.
Passos da Solução: 1. Restringir o número de pessoas que irão especificar os requisitos com casos de uso, pois quando há muitas pessoas para realizar essa atividade há dificuldade de estabelecer consenso entre elas. Use o padrão *Equipe de escrita pequena*, apresentado no Quadro 3.8. 2. Descrever os casos de uso com: Número seqüencial seguido de um nome que identifique o caso de uso; Resumo das ações do caso de uso; Atores envolvidos com o caso de uso; Pré-condições do caso de uso; Fluxo principal do caso de uso; Fluxos alternativos do caso de uso; Pós-condições do caso de uso. 3. Retornar ao passo 2 do Quadro 3.6.

Quadro 3.8 – Padrão Equipe de escrita pequena (adaptado de BRAMBLE et al., 2002)

Nome: Equipe de escrita pequena
Problema: Restringir o número de pessoas para a tarefa de especificar requisitos funcionais.
Solução: Limitar o número de pessoas que irão especificar os requisitos funcionais com casos de uso.
Passos da Solução: 1. Restringir a três o número de pessoas a realizar a especificação dos requisitos com casos de uso. Dentre essas três, obrigatoriamente uma pessoa deve ser o Engenheiro de Requisitos e outra deve ser o representante do Cliente. 2. Retornar ao passo 2 do Quadro 3.7.

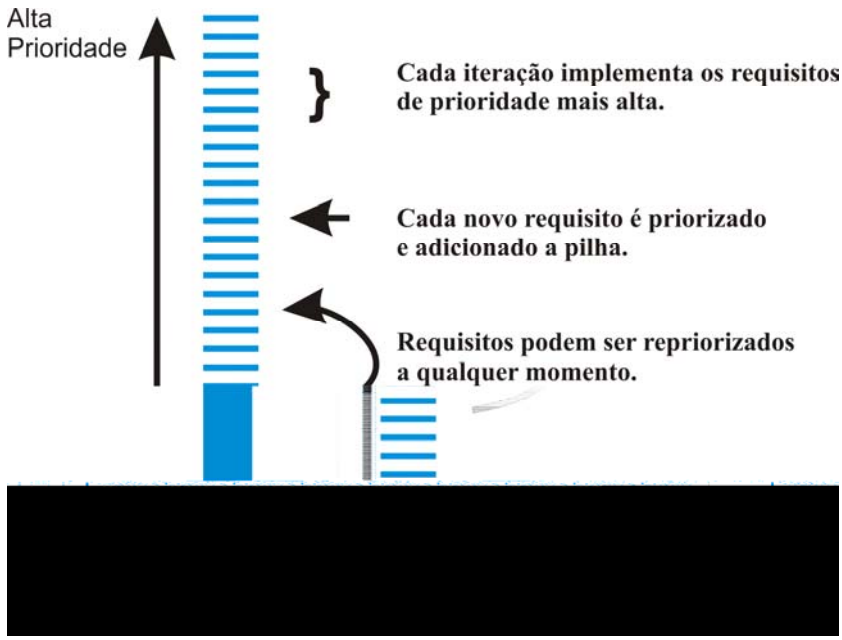
Quadro 3.9 – Padrão Agrupe requisitos às características (adaptado de LAPPE et al., 2004)

Nome: Agrupe requisitos às características
Problema: Proporcionar visibilidade de todos os requisitos do sistema entre os envolvidos com o projeto.
Solução: Reunir e organizar os requisitos de acordo com características gerais do sistema.
Passos da Solução: 1. Estabelecer características que representem as funcionalidades gerais do sistema. 2. Identificar semelhanças efetivas entre os requisitos. 3. Agrupar os requisitos semelhantes de acordo com as características estabelecidas no passo 1. A seguir é apresentado um exemplo dos passos da solução desse padrão:

Exemplo:
Sistema: Controle de Estoque.
Características gerais: Cadastros, Controle de Entrada e Saída e Relatórios.
Requisitos do sistema:
Caso de Uso – Cadastrar Cliente.
Caso de Uso – Movimentar Entrada de Item.
Caso de Uso – Cadastrar Fornecedor.
Caso de Uso – Gerenciar Estoque Mínimo.
Caso de Uso – Movimentar Saída de Item.
Caso de Uso – Relação do movimento de Saída.
Requisitos Agrupados:
Cadastros: Cadastrar Cliente e Cadastrar Fornecedor;
Controle de Entrada e Saída: Movimentar Entrada de Item, Movimentar Saída de Item e Gerenciar Estoque Mínimo.
Relatórios: Relação do movimento de Saída.
Resultado:

Sistema	Características	Requisitos
Controle de Estoque	Cadastros	• Cadastrar Cliente • Cadastrar Fornecedor
	Controle de Entrada e Saída	• Movimentar Entrada de Item • Movimentar Saída de Item • Gerenciar Estoque Mínimo
	Relatórios	• Relação do movimento de Saída.

Quadro 3.10 – Padrão Identificação de elemento (modificado de CORAM, 1996).

Nome: Identificação de elemento
Problema: Priorizar os requisitos do sistema para atender primeiramente as preocupações do cliente.
Solução: Priorizar os requisitos do sistema por meio da técnica de pilha de requisitos.
Passos da Solução: 1. O representante do cliente deve informar quais os requisitos do sistema que ele gostaria que fossem atendidos primeiramente. 2. O Engenheiro de Requisitos deve verificar se existem requisitos com prioridade maior que dependem de outro com prioridade menor. Caso haja, o Engenheiro de Requisitos deve negociar as prioridades com o representante do cliente até que os requisitos de maior prioridade não sejam dependentes de requisitos com menor prioridade. 3. Empilhar os requisitos do sistema de forma que no topo da pilha permaneçam os requisitos de alta prioridade. 4. Gerenciar a pilha de requisitos de forma que a cada novo requisito que for adicionado, removido ou modificado, a pilha deve ser reorganizada e atualizada de forma que os requisitos prioritários permaneçam sempre no topo da pilha e não possuam dependências, como descrito no passo 2. A Figura 3.1 exibe as propriedades da pilha de requisitos.    **Figura 3.1 –** Pilha de requisitos (adaptado de AMBLER, 2007c)

Quadro 3.11 – Padrão adaptado Unidade de Propósito (modificado de COPLIEN; HARRISON, 2004).

Nome: Unidade de Propósito
Problema: Possibilitar conhecimento uniforme dos requisitos a todos envolvidos no projeto.
Solução: Apresentar a todos os requisitos do sistema.
Passos da Solução: 1. Reunir todos os envolvidos com o projeto. 2. Apresentar a todos os requisitos do sistema. Essa apresentação deve ser realizada sem formalidades e com duração de no máximo trinta minutos.

Quadro 3.12 – Padrão Forneça declaração de objetivos para cada requisito (adaptado de LAPPE et al., 2004).

Nome: Forneça declaração de objetivos para cada requisito
Problema: Necessidade de modificações em requisitos.
Solução: Analisar a necessidade de modificação e propor soluções para elas.
Passos da Solução: 1. Analisar se os problemas encontrados podem ser solucionados sem modificar a especificação dos requisitos. Caso haja a necessidade de modificações, o Engenheiro de Requisitos e o representante do Cliente devem propor soluções para as modificações identificadas, observando se essas soluções terão impacto no tempo ou no custo de desenvolvimento da versão do sistema. a. Se as modificações forem solucionadas sem impactar no tempo ou no custo da versão em desenvolvimento, a especificação dos requisitos deve ser atualizada de acordo com o padrão *Detalhe a especificação escrevendo casos de teste*, apresentado no Quadro 3.13. b. Senão, as modificações não possam ser realizadas sem impactar em tempo e custo, o Gerente de Projetos deve interromper o ciclo de desenvolvimento e retornar para o conjunto de trabalho *Elicitar e Especificar Requisitos*, do pacote de processo **Controlar Requisitos**.

Quadro 3.13 – Padrão Detalhe a especificação escrevendo casos de teste (adaptado de LAPPE et al., 2004).

Nome: Detalhe a especificação escrevendo casos de teste
Problema: Modificar a especificação dos requisitos durante o desenvolvimento da versão do sistema.
Solução: Atualizar os requisitos por meio dos casos de teste do sistema.
Passos da Solução: 1. Atualizar os requisitos descrevendo-os nos casos de teste do sistema. Os casos de teste devem ser anexados ao artefato ***Documento de Especificação de Requisitos Priorizado, Estimado e Plano da Versão***.

3.5 - Considerações Finais

Este capítulo apresentou alguns padrões de requisitos e organizacionais e mostrou como esses padrões podem facilitar o reúso de suas soluções nas atividades de engenharia de requisitos e na forma de interação entre as pessoas envolvidas nessas atividades.

Existem vários padrões de requisitos e organizacionais disponíveis na literatura porém, apenas os que estão relacionadas com este trabalho foram apresentados nas Seções 3.2 e 3.3. Os padrões foram mostrados de forma individual e com os elementos nome, problema e solução. Dois desses padrões foram modificados para uso neste trabalho, que tem o enfoque na engenharia de requisitos e nos métodos ágeis. Todos os padrões, originais e modificados, foram adicionados ao elemento **Passos da Solução**, o qual apresenta descrições passo a passo das soluções de cada padrão. Esses padrões foram apresentados nos Quadros 3.1 a 3.13 e são utilizados para organizar e guiar a realização das atividades do processo ágil de engenharia de requisitos proposto por este trabalho, apresentado no Capítulo 4.

Capítulo 4 – Um Processo Ágil de Engenharia de Requisitos apoiado por Padrões de Software

4.1 - Considerações Iniciais

A maioria dos modelos de processo de desenvolvimento de software anteriores à década de 2000 utilizam recursos para que todos os requisitos do sistema sejam elicitados e especificados antes de sua implementação, com pouca participação do cliente na sustentação das informações. Dessa forma, com a constante modificação de requisitos por parte do cliente, ou seja, inserção de novos requisitos, atributos ou funções para manipula-los, torna-se difícil obter uma documentação completa do sistema, na qual todos os recursos consumidos estejam planejados logo ao início da concepção do sistema. Essa forma de desenvolvimento muitas vezes dificulta as possíveis alterações que possam ocorrer com relação aos requisitos do sistema.

Por outro lado, os métodos ágeis optam pela adaptabilidade, empregam técnicas de iteratividade na realização e desenvolvimento dos requisitos com ativa participação do cliente. Assim, enfatizam que o cliente participe da elicitação e especificação dos requisitos do sistema para proporcionar o desenvolvimento de suas reais necessidades e que interaja constantemente com a equipe de projeto para as tomadas de decisões quando ocorrerem modificações em requisitos (FOWLER, 2005; ORR, 2004).

Ainda que seja um modelo adaptativo, as atividades do processo de desenvolvimento de software nos métodos ágeis devem ser realizadas de forma organizada e podem ser apoiadas por práticas e técnicas convencionais da engenharia de software (CREEL, 2006). Dentre essas práticas e técnicas, padrões de requisitos e organizacionais podem ser utilizados para apoiar na organização e direcionar a realização das atividades da engenharia de requisitos nos métodos ágeis, pois esses padrões propõem soluções de sucesso comprovado.

Para evidenciar a realização dessas atividades, este Capítulo apresenta uma forma ordenada de realizá-las. A documentação do processo é feita com a sua modelagem em SPEM e com o apoio dos padrões de requisitos e organizacionais, apresentados na Seção 3.4.

Além desta Seção, este Capítulo apresenta uma visão geral do processo proposto na Seção 4.2, as atividades do processo apoiadas por padrões de requisitos e organizacionais estão detalhadas na Seção 4.3 e a as considerações finais estão na Seção 4.4.

4.2 - Visão geral do processo proposto

O processo proposto foi concebido para estruturar organizadamente a realização das atividades da engenharia de requisitos nos métodos ágeis. Esse processo é fundamentado no modelo de ciclo de vida proposto por Ambler (2006) e nos princípios da metodologia ágil: os requisitos são elicitados e especificados iterativamente, detalhados e refinados de modo evolutivo; modificações em requisitos podem ocorrer durante o desenvolvimento e devem ser controladas de forma a proporcionar a realização apropriada das versões do sistema; há participação ativa do cliente nas tomadas de decisão desde a elicitação até o gerenciamento dos requisitos e há entrega rápida de software funcional, proporcionando *feedback* do cliente com relação ao sistema que está sendo desenvolvido.

Para mostrar e detalhar as atividades da engenharia de requisitos, o processo proposto foi dividido em:

- **Inicialização**: ocorre o comprometimento do cliente com o projeto, define-se a equipe que pode participar do desenvolvimento do sistema e inicia-se a identificação das necessidades do cliente em alto nível de abstração, fornecendo a visão geral dos objetivos do sistema.

- **Controle de Requisitos:** as atividades da engenharia de requisitos são consolidadas de forma iterativa, pois de acordo com Boehm (2002), os envolvidos com o projeto possuem dificuldades em identificar e compreender todos os requisitos logo no seu início. Nos primeiros ciclos de elicitação e especificação de requisitos, os requisitos gerais do sistema são obtidos para proporcionar melhor entendimento das funções do sistema que devem ser atendidas e à medida que detalhes dessas funções são descobertos, os requisitos são refinados, sempre com a participação do cliente para a validação dos requisitos. Nessa etapa ocorre também a estimativa dos requisitos por meio da técnica Pontos de Casos de Uso, proporcionando determinar o esforço, custo e prazo necessários para atender os requisitos conhecidos.

- **Gestão de Versão do Sistema**: ocorre o planejamento e desenvolvimento das versões de software funcional e acontece a entrega dessas versões ao cliente. Há o gerenciamento de modificações em requisitos, já que modificações podem ocorrer a qualquer momento do desenvolvimento e devem ser controladas de forma que os desenvolvedores trabalhem sempre com os requisitos atualizados. Essa etapa também mostra as atividades de projeto, codificação e teste de software como “caixa preta”, ou

seja, essas atividades não são detalhadas por não fazerem parte do escopo deste trabalho.

Processos de software são formados por diversos elementos e pela definição de seus relacionamentos (ACUNÃ; FERRE, 2001). No processo proposto os elementos presentes são: atividades, artefatos, papéis e guias. Para mostrar o relacionamento entre as etapas do processo e entre os seus elementos, o processo proposto foi modelado com o meta-modelo SPEM (OMG, 2005), com a notação UML e os seus mecanismos de extensão (estereótipos) (OMG, 2001). No SPEM, o processo é organizado em pacotes de processo, que representam as suas etapas.

A Figura 4.1 apresenta os pacotes de processo considerados: **Inicialização**, **Controle de Requisitos** e **Gestão de Versão do Sistema**.

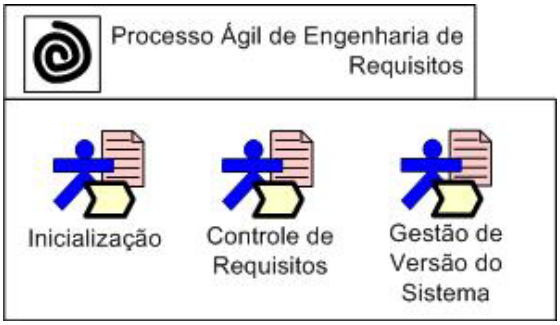


Figura 4.1 – Pacotes de processo do processo proposto.

Para mostrar o relacionamento ordenado e dinâmico existente entre os pacotes de processo, ou seja, a ordem de execução dos pacotes de processo e a iteratividade existente entre eles, é utilizado o diagrama de atividades da UML como apresenta a Figura 4.2. O processo inicia-se com o pacote de processo **Inicialização** e é finalizado após o pacote de processo **Gestão de Versão do Sistema**.

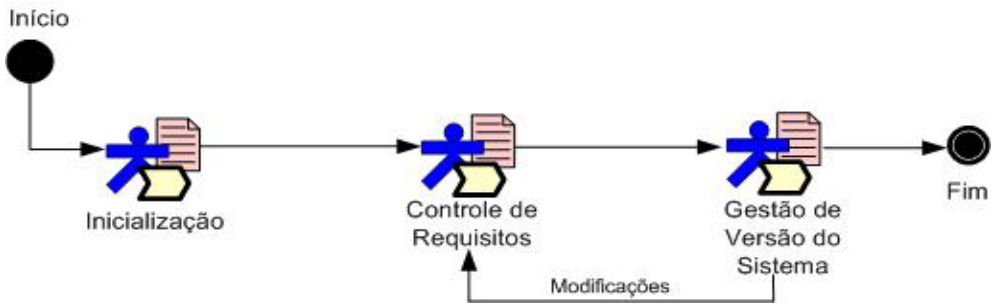


Figura 4.2 – Diagrama de atividades dos pacotes de processo do processo proposto.

A Figura 4.3 exibe a estrutura de relacionamentos entre os pacotes de processo e os elementos do processo, na representação do SPEM, por meio de um diagrama de classes da UML, no qual somente os nomes das classes e os relacionamentos entre elas são apresentados.

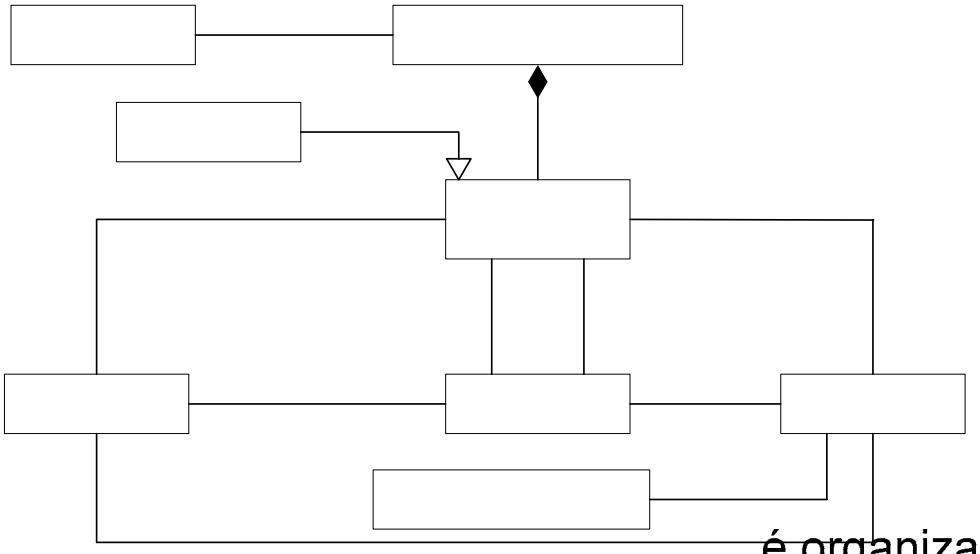


Figura 4.3 – Elementos do processo proposto.

O processo é organizado em pacotes de processo os quais possuem conjuntos de trabalho que se relacionam com papéis, artefatos e guias. Os conjuntos de trabalho são desempenhados por pessoas representadas por um papel, podem gerar ou consumir artefatos e possuir ou não guias para auxiliar a sua realização (OMG, 2005). Atividades podem herdar dos conjuntos de trabalho os relacionamentos com papéis, artefatos e guias.

Os padrões de requisitos e organizacionais, apresentados na Seção 3.4, foram integrados ao processo proposto e classificados como um tipo de guia, pois podem ser utilizados para guiar a organização e a realização de uma atividade, de um papel ou um artefato no processo de desenvolvimento de software (FERDINANDI, 2002).

A Tabela 4.1 apresenta os elementos que compõem cada pacote de processo. A primeira coluna apresenta os pacotes de processo, a segunda apresenta os conjuntos de trabalho ou atividades pertencentes a cada pacote de processo, a terceira apresenta os papéis, a quarta apresenta os artefatos e a quinta coluna apresenta os padrões como guia.

O detalhamento de todos os pacotes de processo e de seus respectivos elementos (conjuntos de trabalho ou atividades, artefatos, papéis e guias) são apresentados na próxima Seção.

Tabela 4.1 – Composição dos pacotes de processo por meio de seus elementos

Pacote de Processo	Conjunto de Trabalho/Atividade	Papéis	Artefatos	Padrões representados como Guia
▪ Inicialização	▪ Atividade: Identificar Escopo	▪ Cliente ▪ Gerente de Projetos	▪ Escopo do Projeto	▪ *Amplitude antes de profundidade* ▪ *Envolver o cliente* ▪ *Equipe auto selecionadora* ▪ *Empregue um Engenheiro de Requisitos como encarregado.*
▪ Controle de Requisitos	▪ Conjunto de Trabalho: Elicitar e Especificar Requisitos.	▪ Engenheiro de Requisitos ▪ Colaborador	▪ Documento de Requisitos	▪ *Desenvolvimento espiral* ▪ *Crie uma diretriz de especificação acompanhando o trabalho de um analista* ▪ *Cenários definem o problema* ▪ *Equipe de escrita pequena* ▪ *Agrupe requisitos a características*
	▪ Conjunto de Trabalho: Priorizar e Estimar Requisitos	▪ Gerente de Projetos ▪ Engenheiro de Requisitos ▪ Colaborador	▪ Documento de Requisitos Priorizado e Estimado	▪ *Identificação de elemento*
▪ Gestão de Versão do Sistema	▪ Atividade: Definir Versão	▪ Gerente de Projetos ▪ Engenheiro de Requisitos ▪ Colaborador	▪ Documento de Requisitos Priorizado, Estimado e Plano da Versão	▪ *Unidade de Propósito*
	▪ Conjunto de Trabalho: Planejar Testes	▪ Engenheiro de Requisitos ▪ Desenvolvedores	▪ Casos de Teste	▪ *Detalhe a especificação escrevendo casos de teste*
	▪ Conjunto de Trabalho: Desenvolver Iteração	▪ Desenvolvedores	▪ Versão	▪ *Forneça declaração de objetivos para cada requisito*
	▪ Conjunto de Trabalho: Finalizar Versão	▪ Desenvolvedores	▪ Versão	

4.3 - Detalhamento dos pacotes de processo e seus elementos

4.3.1 - Pacote de processo Inicialização

Tem a finalidade de estabelecer o escopo do projeto e é composto da atividade *Identificar Escopo*, Figura 4.4.

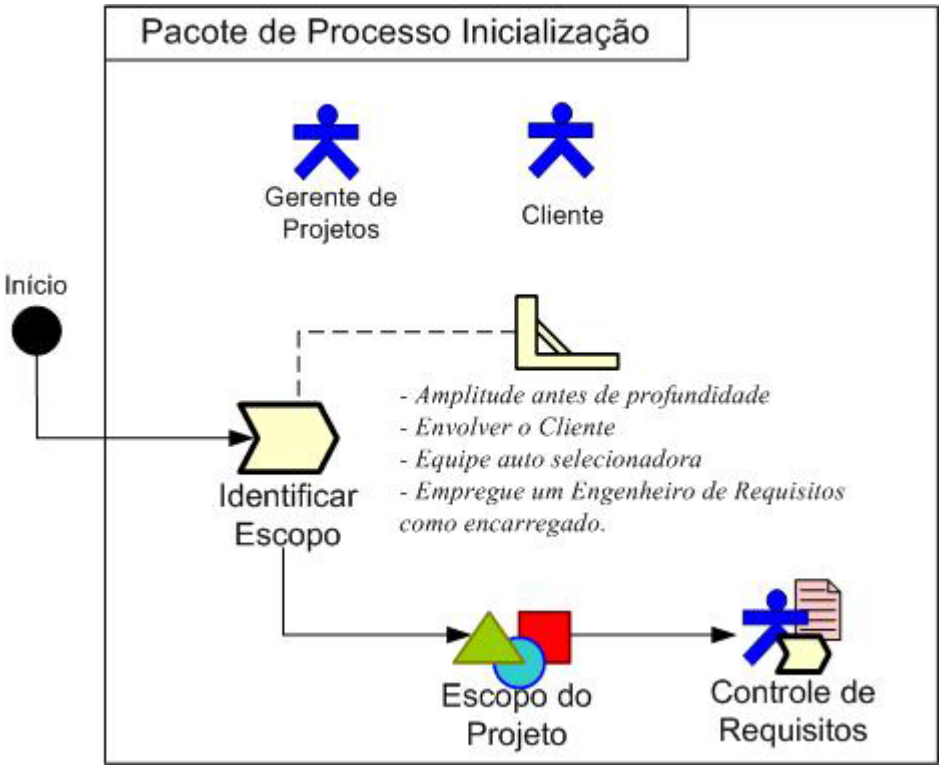


Figura 4.4 – Diagrama de atividades do pacote de processo Inicialização.

A atividade *Identificar Escopo* é de responsabilidade do Gerente de Projetos, representado pelo papel de mesmo nome, o qual define o planejamento e acompanha a realização das atividades do projeto. Junto com o Cliente o Gerente de Projetos deve identificar e definir os objetivos do sistema captando o contexto no qual está inserido, obter o comprometimento do Cliente no projeto e identificar atores para desempenharem os outros papéis durante a realização do projeto. Além do papel Gerente de Projetos, o processo proposto é constituído dos papéis:

- **Colaborador**: é responsável por representar o Cliente nas tomadas de decisões durante o projeto e participar da elicitação, especificação, validação e priorização de requisitos. Esse papel pode ser representado por mais de um ator.

- **Engenheiro de Requisitos**: é o responsável por todas as ações referentes aos requisitos do sistema.

- **Desenvolvedor**: é o responsável por realizar atividades referentes a projeto, codificação e teste de software. Esse papel pode ser representado por mais de um ator.

Os papéis Colaborador, Engenheiro de Requisitos e Desenvolvedor serão relacionados ao processo no momento em que realizarem alguma atividade no processo. Assim, o Colaborador e o Engenheiro de Requisitos serão relacionados no pacote de processo **Controle de Requisitos**, apresentado na Subseção 4.3.2, e o papel Desenvolvedor ao pacote de processo **Gestão de Versão do Sistema**, apresentado na Subseção 4.3.3.

Os objetivos do sistema podem ser obtidos a partir dos problemas e necessidades do Cliente, que podem ser descritos com casos de uso (RUP, 2001).

Obter o comprometimento do Cliente com o projeto não é tarefa fácil porém, é essencial para que sejam identificados todos os requisitos necessários para o sistema (AMBLER, 2007b).

Para realizar essa atividade os seguintes passos são necessários:

1. O Gerente de Projetos deve, até que o Cliente esteja totalmente de acordo como os objetivos do sistema, reunir-se com ele para:
   i. Identificar os objetivos do sistema. As seguintes perguntas podem ser feitas ao Cliente: Qual é o problema? Qual é a sua necessidade?
   ii. Descrever os objetivos do sistema com casos de uso gerais. Os detalhes devem ser descritos apenas na especificação dos requisitos. Use o padrão *Amplitude antes de profundidade*, apresentado no Quadro 3.1.
   iii. Obter acordo do Cliente com as descrições dos objetivos do sistema.
2. O Gerente de Projetos deve obter o comprometimento do Cliente para as tomadas de decisões no projeto, conforme o padrão *Envolver o cliente*, apresentado no Quadro 3.2.
3. O Cliente deve indicar no mínimo um ator para desempenhar o papel de Colaborador.
4. O Gerente de Projetos deve indicar os outros atores da equipe que irão desempenhar os papéis do projeto, de acordo com o padrão *Equipe auto selecionadora*, apresentado no Quadro 3.3.
5. O Gerente de Projetos deve atribuir um nome significativo para o projeto.

Ao final dessa atividade o artefato ***Escopo do Projeto*** é gerado com as informações apresentadas no Quadro 4.1.

Quadro 4.1 – Gabarito do Artefato Escopo do Projeto.

Escopo do Projeto
1 – Nome Projeto: *<conjunto de caracteres que representa o nome do projeto>*
2 – Colaborador: *<nome do(s) representante(s) do cliente no projeto>*
3 – Objetivos do sistema: *<casos de uso representando os objetivos do sistema>*
3.1 – Diagrama de casos de uso: *<diagrama de casos de uso representando os objetivos do sistema>*
3.2 – Descrição geral dos casos de uso: *<descrição geral dos casos de uso representando os objetivos do sistema>*
4 – Papéis: *<identificação dos atores que irão desempenhar os papéis no projeto>*

O artefato ***Escopo do Projeto*** é entrada para o conjunto de trabalho *Elicitar e Especificar Requisitos* do pacote de processo **Controle de Requisitos**, descrito na próxima Subseção.

4.3.2 - Pacote de processo Controle de Requisitos

Tem a finalidade de realizar a elicitação, especificação, validação, priorização e estimativas dos requisitos do sistema, sendo composto de dois conjuntos de trabalho como mostra a Figura 4.5.

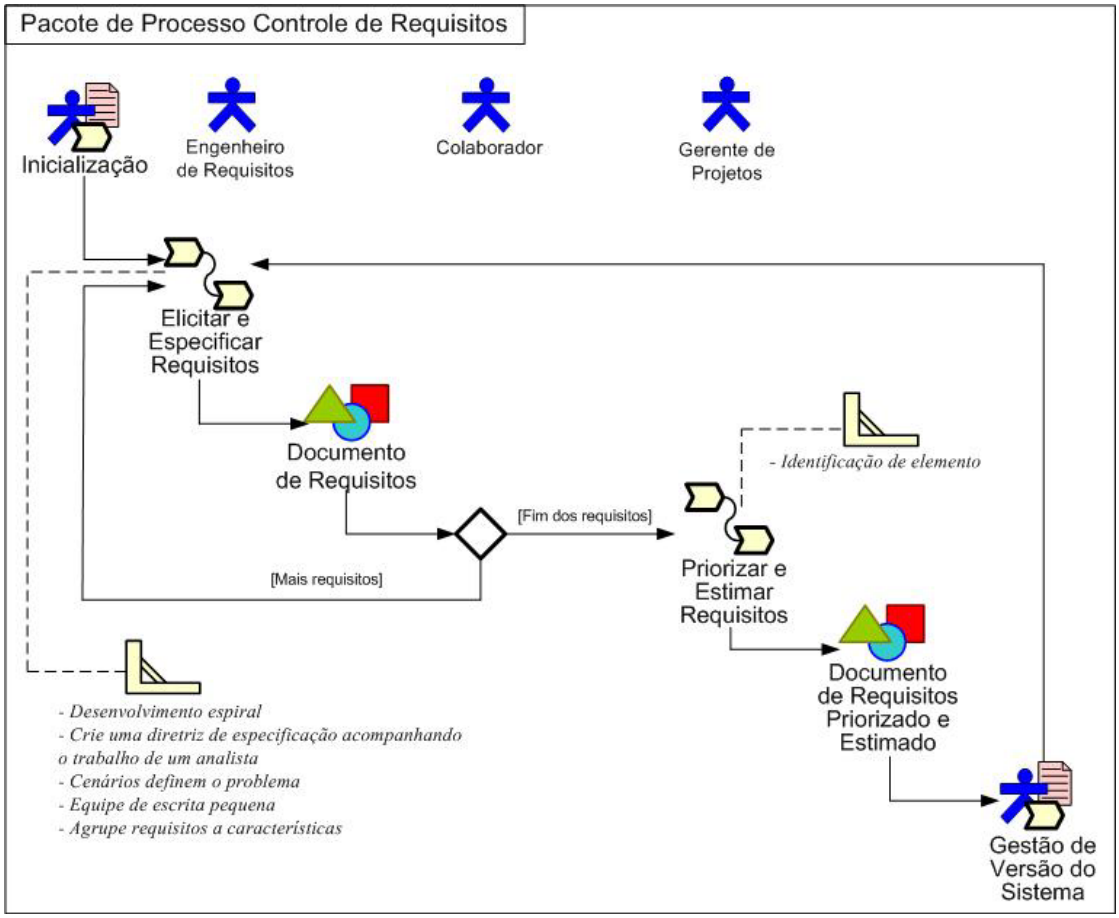


Figura 4.5 – Diagrama de atividades do pacote de processo Controle de Requisitos.

O conjunto de trabalho *Elicitar e Especificar Requisitos* é de responsabilidade do Engenheiro de Requisitos que, junto com o Colaborador e a partir do escopo do projeto, devem obter os requisitos do sistema e especificá-los evitando o uso de jargões técnicos.

Como exibido na Figura 4.5, esse conjunto de trabalho deve ser realizado iterativamente, ou seja, os requisitos do sistema devem ser elicitados e especificados em ciclos repetitivos até que todos os requisitos conhecidos estejam detalhados e, ao término de cada ciclo, o artefato ***Documento de Requisitos*** deve ser produzido ou atualizado. Esse conjunto de trabalho deve ser realizado de acordo com o padrão *Desenvolvimento espiral*, apresentado no Quadro 3.5.

Para a elicitação dos requisitos técnicas convencionais podem ser utilizadas. Alguns exemplos dessas técnicas são: entrevistas, questionários, *workshop*, *brainstorming* e observação direta (ROBERTSON; ROBERTSON, 2006). Robertson e Robertson apontam *workshop* e *brainstorming* como as mais indicadas para a elicitação de requisitos, pois conseguem envolver o cliente na geração de idéias.

A especificação dos requisitos funcionais pode ser realizada, por exemplo, com a elaboração de modelos de casos de uso (RUP, 2001). Robertson e Robertson enfatizam que os casos de uso facilitam o entendimento "do que" deve ser realizado. Requisitos não funcionais, os quais não devem ser descritos com casos de uso devem ser descritos textualmente (LARMAN, 2007).

Para realizar esse conjunto de trabalho os seguintes passos são necessários:

1. O Engenheiro de Requisitos deve se reunir com o Colaborador para:
    i. Identificar fontes de informações para a elicitação dos requisitos, que podem ser pessoas, objetos e outros sistemas. Se existir um modelo na organização para identificar os requisitos, esse deve ser usado. Caso contrário, algumas perguntas que podem ser feitas ao Colaborador são: Quem são os usuários do sistema? Quem mais será afetado pelas saídas produzidas pelo sistema? Quem avaliará e aprovará o sistema quando for liberado e implantado? Existem outros usuários internos ou externos do sistema cujas necessidades precisam ser abordadas? Quem manterá o novo sistema?
    ii. Elicitar os requisitos do sistema a partir das fontes de informações levantadas no passo anterior. Técnicas convencionais de elicitação de requisitos podem ser utilizadas.

iii. Utilizar a técnica de observação direta para elicitar os requisitos relacionados a objetos e outros sistemas, se existirem.

iv. Especificar os requisitos do sistema. Para que as especificações de requisitos não sejam realizadas de acordo com as habilidades individuais das pessoas em cada projeto, dificultando o entendimento dos requisitos pelos integrantes da equipe, use o padrão *Crie uma diretriz de especificação acompanhando o trabalho de um analista*, apresentado no Quadro 3.6.

O Quadro 4.2 apresenta um gabarito das informações que devem constituir o artefato ***Documento de Requisitos.***

Quadro 4.2 – Gabarito do Artefato Documento de Requisitos

Documento de Requisitos	
Casos de uso	
Nome do Caso de Uso	*<número seqüencial seguido de um nome que identifique o caso de uso>.*
Resumo	*<descrição resumida das ações do caso de uso>.*
Atores	*<atores participantes do caso de uso>.*
Pré-condições	*<lista das pré-condições para o caso de uso iniciar>.*
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
<ações dos atores no caso de uso>.	*<ações do sistema no caso de uso>.*
Fluxo Alternativo	
Ações do Ator	**Ações do Sistema**
<ações dos atores no caso de uso>.	*<ações do sistema no caso de uso>.*
Pós-condições	*<lista das pós-condições ao término do caso de uso>.*

Requisitos não Funcionais
Identificador do Requisito: *<número seqüencial seguido de um nome que identifique o requisito>.*
Descrição: *<descrição detalhada do requisito>.*

Requisitos agrupados de acordo com características		
Sistema	**Características**	**Requisitos**
<nome do sistema>.	*<nome das característica do sistema>.*	*<identificador do requisito ou caso de uso>.*
	<nome das característica do sistema>.	*<identificador do requisito ou caso de uso>.*
		<identificador do requisito ou caso de uso>.

Após a conclusão da elicitação e especificação dos requisitos inicia-se o conjunto de trabalho *Priorizar e Estimar Requisitos*. Esse conjunto de trabalho é de responsabilidade do Gerente de Projetos e do Engenheiro de Requisitos, que junto com o Colaborador devem priorizar e estimar o esforço necessário para atender os requisitos do sistema. O Engenheiro de Requisitos deve priorizar os requisitos do sistema até que esses requisitos estejam totalmente ordenados e sem dependências.

Para ordenar os requisitos do sistema Ambler (2007c), propõe o uso da técnica pilha de requisitos, na qual os requisitos são empilhados sendo que os de maior prioridade, os quais preocupam o Cliente e serão atendidos primeiramente, ficam no topo da pilha. Após a priorização dos requisitos, o Gerente de Projetos e o Engenheiro de Requisitos devem estimar o esforço necessário para que os requisitos sejam atendidos. A técnica Pontos por Casos de Uso (PCU), apresentada na Subseção 2.4.1.1 pode ser utilizada.

Para realizar esse conjunto de trabalho os seguintes passos são necessários:

1. O Engenheiro de Requisitos e o Colaborador devem realizar a priorização dos requisitos do sistema de acordo com o padrão *Identificação de elemento*, apresentado no Quadro 3.10.
2. O Gerente de Projetos e o Engenheiro de Requisitos devem estimar o esforço de desenvolvimento para os requisitos do sistema. As estimativas devem ser realizadas utilizando a técnica de Pontos de Casos de Uso, apresentada na Subseção 2.4.1.1.

Quando o conjunto de trabalho *Priorizar e Estimar Requisitos* for concluído, o artefato ***Documento de Requisitos***, apresentado no Quadro 4.2 é acrescido da priorização e estimativas dos requisitos, produzindo o artefato ***Documento de Requisitos Priorizado e Estimado***. Um gabarito desse artefato é apresentado no Quadro 4.3.

Quadro 4.3 – Gabarito do artefato Documento de Requisitos Priorizado e Estimado

Documento de Requisitos (Quadro 4.2)	
Requisitos Priorizados e Estimados	
Pilha de Requisitos	**Esforço Estimado**
<identificador do requisito ou caso de uso>	*<tempo estimado em horas>*
<identificador do requisito ou caso de uso>	*<tempo estimado em horas>*

4.3.3 - Pacote de processo Gestão de Versão do Sistema

Tem a finalidade de, a partir do artefato ***Documento de Requisitos Priorizado e Estimado***, gerir o desenvolvimento das versões do sistema. É composto de uma atividade e três conjuntos de trabalho, como mostra a Figura 4.6.

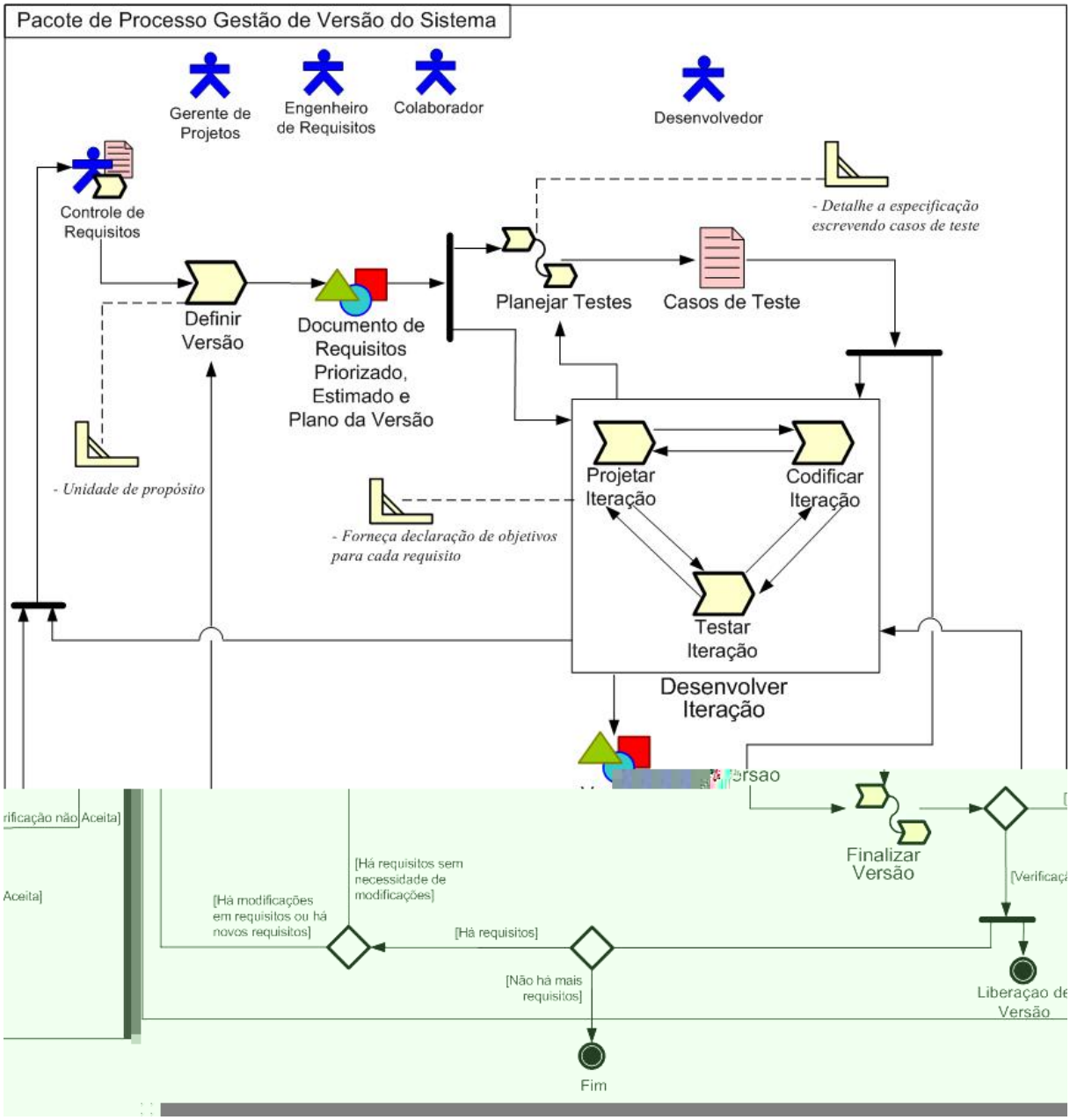


Figura 4.6 – Diagrama de atividades do pacote de processo Gestão de Versão do Sistema.

A Figura 4.6 mostra atividades de Projeto de Software, Codificação, Teste de Software e Integração de Versões sendo representadas pelos conjuntos de trabalho *Planejar Testes*,

Desenvolver Iteração e *Finalizar Versão*. Entretanto essas atividades não serão detalhadas por não fazerem parte do escopo deste trabalho.

A atividade *Definir Versão* é de responsabilidade do Gerente de Projetos que junto com o Engenheiro de Requisitos e o Colaborador retiram um conjunto de requisitos da pilha de requisitos e os aloca em versões a serem desenvolvidas até que a pilha esteja vazia. Cada versão do sistema deve durar, de acordo com Schwaber e Beedle (2002), no máximo trinta dias.

A necessidade de retirar os requisitos da pilha serve para que eles sejam "congelados" durante a realização de uma iteração do ciclo de desenvolvimento. Esse congelamento é necessário para que os desenvolvedores trabalhem com requisitos estáveis durante a iteração e as modificações em requisitos não atrapalhem o andamento do processo de desenvolvimento, mas se torne um benefício para a qualidade do sistema (AMBLER, 2007c).

Para realizar essa atividade os seguintes passos são necessários:

1. Caso seja o primeiro ciclo do processo, o Gerente de Projetos deve:
    i. Estipular a duração máxima, em dias, para as versões a serem desenvolvidas.
2. O Engenheiro de Requisitos e o Colaborador devem possibilitar conhecimento uniforme dos requisitos do sistema a todos os envolvidos no projeto, conforme o padrão *Unidade de Propósito*, apresentado no Quadro 3.11.
3. O Gerente de Projetos deve:
    i. Informar no artefato ***Documento de Requisitos Priorizado e Estimado e Plano da Versão*** o número da versão que será desenvolvida.
    ii. Somar os tempos estimados dos requisitos do topo da pilha de requisitos, até que a somatória não ultrapasse a duração máxima estipulada no passo 1. Ressalva-se que a somatória das estimativas deve ser considerada de acordo com as horas trabalhadas da equipe, ou seja, se a equipe que participará do projeto trabalhar oito horas diárias deve-se somar os tempos estimados para os requisitos e dividi-los por oito.
    iii. Retirar da pilha os requisitos a serem desenvolvidos na versão e alocá-los no artefato ***Documento de Requisitos Priorizado e Estimado e Plano da Versão***.

Após a realização dessa atividade o artefato ***Documento de Requisitos Priorizado e Estimado e Plano da Versão*** é produzido conforme o gabarito apresentado no Quadro 4.4.

Quadro 4.4 – Gabarito do artefato Documento de Requisitos Priorizado e Estimado e Plano da Versão

Plano da Versão	
	Nº da Versão *<número da versão>*
Requisito	**Esforço Estimado**
<identificador do requisito ou caso de uso>	*<tempo estimado em horas>*
<identificador do requisito ou caso de uso>	*<tempo estimado em horas>*

Após a definição dos requisitos que serão atendidos na versão do sistema, os conjuntos de trabalho *Planejar Testes* e *Desenvolver Iteração* podem ser realizados em paralelo.

Esses conjuntos de trabalho, bem como o *Finalizar Versão* são apresentados apenas para mostrar que durante a realização deles podem ocorrer modificações em requisitos e essas modificações devem ser analisadas e gerenciadas.

No conjunto de trabalho *Planejar Testes*, o qual é de responsabilidade do Desenvolvedor, são produzidos os casos de testes de unidade e versão do sistema, no artefato **Plano de Teste**. Casos de teste de unidade são utilizados na atividade *Testar Iteração* do conjunto de trabalho *Desenvolver Iteração* e casos de teste de versão para serem utilizados no conjunto de trabalho *Finalizar Versão*.

No conjunto de trabalho *Desenvolver Iteração* o Desenvolvedor é responsável por produzir as versões do sistema iterativamente realizando atividades de projeto (atividade *Projetar Iteração*), codificação (atividade *Codificar Iteração*) e executando testes de unidade na versão produzida (atividade *Testar Iteração*).

Durante a realização desse conjunto de trabalho o Desenvolvedor pode identificar problemas e incoerências nas especificações dos requisitos. A existência desses problemas deve ser imediatamente comunicada ao Gerente de Projetos, o qual deve analisá-los tendo como guia o padrão *Forneça declaração de objetivos para cada* requisito, apresentado no Quadro 3.12.

O resultado do conjunto de trabalho *Desenvolver Iteração* é o artefato **Versão**, o qual é entrada para o conjunto de trabalho *Finalizar Versão*. Esse conjunto de trabalho tem o objetivo de executar os casos de teste de versão do sistema e realizar a integração das versões, caso existam. Caso sejam verificadas inconsistências na versão, deve-se retornar para o conjunto de trabalho *Desenvolver Iteração* a fim de corrigí-las. Caso contrário ocorre a entrega da versão do sistema ao Cliente.

O Gerente de Projetos deve verificar com o Colaborador e com o Engenheiro de Requisitos se ainda existem requisitos a serem atendidos. Caso não haja, o processo é finalizado. Caso existam, os seguintes passos devem ser realizados:

1. O Gerente de Projetos, o Engenheiro de Requisitos e o Colaborador devem:
    i. Verificar se esses requisitos necessitam de modificações ou se são novos requisitos a serem elicitados.
    ii. Caso seja necessário modificações ou sejam novos requisitos, deve-se:
        a. Retornar ao conjunto de trabalho *Elicitar e Especificar Requisitos*, do pacote de processo **Controlar Requisitos** apresentado na Subseção 4.3.2.
    iii. Caso contrário, deve-se:
        a. Realizar a atividade *Definir Versão* para iniciar novo ciclo de desenvolvimento de versão do sistema.

4.4 - Considerações Finais

Este capítulo apresentou um processo ágil de engenharia de requisitos fundamentado nos princípios da metodologia ágil e no modelo de ciclo de vida proposto por Ambler (2006). O processo proposto foi modelado por meio do meta-modelo SPEM (OMG, 2005) para organizar e mostrar o relacionamento entre todos os elementos do processo, o qual é composto de três pacotes de processo: Inicialização, Controle de Requisitos e Gestão de Versão do Sistema. Utilizando-se do diagrama de atividades da UML foi possível também mostrar de forma objetiva o relacionamento ordenado e dinâmico existente entre os pacotes de processo e entre os elementos.

Por meio do estereótipo guia do SPEM, os padrões de requisitos e organizacionais apresentados nos Quadros 3.1 à 3.13 foram integrados ao processo proposto e proporcionaram a organização e a realização de suas atividades.

As atividades referentes ao Projeto, Codificação e Testes de Software não foram detalhadas no processo proposto e foram abordadas como "*caixa preta*", pois este trabalho está enfocado na engenharia de requisitos. Para a realização dessas atividades, a organização que fizer uso do processo proposto deverá adaptá-las de acordo com as suas necessidades.

O Capítulo 5 mostra a aplicação do processo ágil de engenharia de requisitos proposto em um estudo de caso realizado com uma empresa real de desenvolvimento de software.

Capítulo 5 – Estudos de Caso de aplicação do processo proposto

5.1 - Considerações Iniciais

O processo de obtenção de requisitos é uma preocupação constante dos desenvolvedores de software, pois muitos dos problemas encontrados no produto final advêm da fase de engenharia de requisitos e a correção desses problemas se torna mais trabalhosa à medida que se avança no processo de desenvolvimento.

O desenvolvimento ágil vem sendo utilizado em larga escala por profissionais que desenvolvem software e a forma de controlar os requisitos é diferente dos métodos tradicionais, pois requisitos devem ser obtidos de forma iterativa, com a participação do cliente e modificações podem ocorrer e serem tratadas durante todo o desenvolvimento do sistema (AMBLER, 2006). Para apoiar e organizar o processo de engenharia de requisitos proposto no Capítulo 4, padrões de requisitos e padrões organizacionais foram utilizados. Para mostrar que padrões de software integrados à engenharia ágil de requisitos auxiliam efetivamente o desenvolvimento de software, este capítulo apresenta um estudo de caso realizado com uma empresa real desenvolvedora de software.

Este capítulo está organizado da seguinte forma: a Seção 5.2 apresenta a descrição do ambiente em que foi realizado o estudo de caso; na Seção 5.3 é apresentada a condução desse estudo de caso; na Seção 5.4 são apresentados os resultados do questionário aplicado à equipe que participou desse estudo de caso e na Seção 5.5 estão às considerações finais.

5.2 - Descrição do ambiente do Estudo de Caso

O estudo de caso foi realizado em uma empresa real de tecnologia da informação, produtora de software no domínio da Internet. Essa empresa, denominada “Empresa T”, situada na cidade de Campo Grande, Estado de Mato Grosso do Sul, disponibilizou uma equipe com cinco integrantes para a realização desse estudo de caso: A, B, C, D e E, com diferentes conhecimentos e experiências em desenvolvimento de software. Quatro são formados na área de informática (A, B, C, D) e um (E) está cursando o último semestre do curso de Tecnologia em Sistemas para a Internet. Dos quatro primeiros, dois são recém formados (C e D), um está formado há mais de três anos (B) e o outro há mais de seis anos e tem Mestrado em Ciências da Computação (A).

O autor desta dissertação trabalha nessa empresa na área de processo de desenvolvimento de software e teve a anuência da gerência de processos para a aplicação do processo de engenharia de requisitos proposto neste trabalho.

A partir de entrevista com os integrantes da equipe foi possível verificou-se que os integrantes da equipe possuíam conhecimentos variados com relação aos assuntos abordados no processo proposto. Todos os integrantes da equipe tinham conhecimentos sobre processo de desenvolvimento de software e engenharia de requisitos já que a Empresa T produz software seguindo o modelo em cascata e diretrizes definidas de acordo com o RUP (RUP, 2001). A fase de engenharia de requisitos é realizada em ciclos repetidos de levantamento, análise, especificação e validação de requisitos até que se obtenha todos os requisitos do sistema e esses requisitos são especificados com descrição textual e casos de uso, quando necessário.

Apenas dois integrantes da equipe (A e B) possuíam conhecimentos sobre métodos ágeis e padrões de projeto (*design patterns*). Assim, pode-se classificar o nível de conhecimento de cada um dos integrantes da equipe nos assuntos relacionados ao processo da seguinte forma: **desconhecido** representa que o integrante não tem conhecimento algum sobre o assunto, nada ouviu falar sobre ele; conhecimento **básico** representa que o integrante já ouviu falar, mas nunca fez uso de diretrizes, técnicas e conceitos envolvidos; conhecimento **intermediário** representa que o integrante conhece e já fez uso (algumas vezes) de diretrizes, técnicas e conceitos envolvidos; e conhecimento **avançado** representa que o integrante conhece e faz uso constante de diretrizes, técnicas e conceitos envolvidos. A Tabela 5.1 apresenta, nas linhas, o nível de conhecimento dos integrantes da equipe do projeto e nas colunas, são apresentados os assuntos relacionados ao processo.

Tabela 5.1 – Nível de conhecimento dos integrantes da equipe

Integrantes	Processo de Software	Engenharia de Requisitos	Métodos ágeis	Padrões de Software
A	Avançado	Avançado	Intermediário	Básico
B	Avançado	Intermediário	Básico	Básico
C	Intermediário	Avançado	Desconhecido	Desconhecido
D	Intermediário	Intermediário	Desconhecido	Desconhecido
E	Intermediário	Básico	Desconhecido	Desconhecido

A fim de proporcionar conhecimento a todos os integrantes da equipe do projeto, sobre os assuntos relacionados ao processo proposto, anteriormente ao início da realização do estudo de caso foi realizada uma exposição, pelo autor desta dissertação, aos integrantes da

equipe. Em seguida, foi apresentado o processo proposto no Capítulo 4. A próxima Seção comenta a condução do estudo de caso.

5.3 - Condução do Estudo de Caso

Os requisitos do sistema foram desenvolvidos com a utilização das ferramentas MS-Visio 2003 (MICROSOFT, 2007a) e MS-Word 2003 (MICROSOFT, 2007b), por já serem utilizadas pela Empresa T. O sistema desenvolvido é de propriedade da Empresa T, o Cliente é o próprio presidente da empresa e trata de venda de produtos pela internet, sem abordar o controle de estoque. A descrição do sistema é apresentada no Quadro 5.4.

O projeto teve início com o papel de Gerente de Projetos atribuído ao integrante A, que se reuniu com o Cliente para a definição do escopo do projeto de acordo com a atividade *Identificar Escopo*, apresentada na Subseção 4.3.1. Para a definição do escopo do projeto, o Gerente de Projetos identificou os objetivos do sistema por meio dos problemas e necessidades do Cliente, obteve o comprometimento dele com o projeto por meio da indicação do ator Colaborador e atribuiu os outros papéis do projeto aos integrantes da equipe.

Os objetivos apresentados no Quadro 5.2 foram descritos de acordo com os passos da solução do padrão *Amplitude antes de profundidade,* apresentado no Quadro 3.1. Apesar do diagrama de casos de uso e de suas descrições gerais, o Gerente de Projetos decidiu pela elaboração de um texto descrevendo a visão do sistema.

O Cliente aprovou o diagrama e as descrições dos casos de uso e a descrição textual da visão do sistema e conforme o padrão *Envolver o cliente,* apresentado no Quadro 3.2, indicou uma pessoa para desempenhar o papel de Colaborador, o integrante A, que passou a acumular papéis (Gerente de Projetos e Colaborador). O Cliente indicou o Gerente de Projetos para desempenhar o papel de Colaborador pelo fato de que ele não podia participar do projeto e a única pessoa com conhecimentos nas regras de negócio do sistema a ser desenvolvido e com a capacidade de tomar decisões no lugar dele era o próprio Gerente de Projetos.

O Gerente de Projetos de acordo com os passos da solução do padrão *Equipe auto selecionadora,* apresentado no Quadro 3.3, reuniu todos os integrantes da equipe e apresentou os objetivos do sistema. Os integrantes da equipe opinaram quais os papéis que gostariam de desempenhar: os integrantes B, D e E manifestaram interesse em desempenhar o papel de Desenvolvedor e o integrante C em desempenhar o papel de Engenheiro de Requisitos. Para a atribuição do papel Engenheiro de Requisitos, o Gerente de Projetos observou as habilidades

dos integrantes de acordo com os passos da solução do padrão *Empregue um Engenheiro de Requisitos como encarregado,* apresentado no Quadro 3.4. Assim, o Gerente de Projetos atribuiu as responsabilidades do projeto de acordo com as habilidades e os interesses dos integrantes, como apresenta o Quadro 5.4. O autor desta dissertação atuou como apoio ao processo, ou seja, acompanhando a aplicação do processo proposto e coletando informações junto à equipe.

Ao término da atividade *Identificar Escopo* o artefato ***Escopo do Projeto*** foi produzido utilizando o gabarito apresentado no Quadro 4.1, adicionado do item Visão do Projeto. Um trecho desse artefato é apresentado de forma decomposta nos Quadros: 5.1 com o nome do projeto e a atribuição do papel Colaborador; 5.2 com os objetivos do sistema representados pelo diagrama de casos de uso; 5.3 com a descrição geral do caso de uso **Realizar pedido**; e 5.4 com a visão do sistema e as atribuições do projeto.

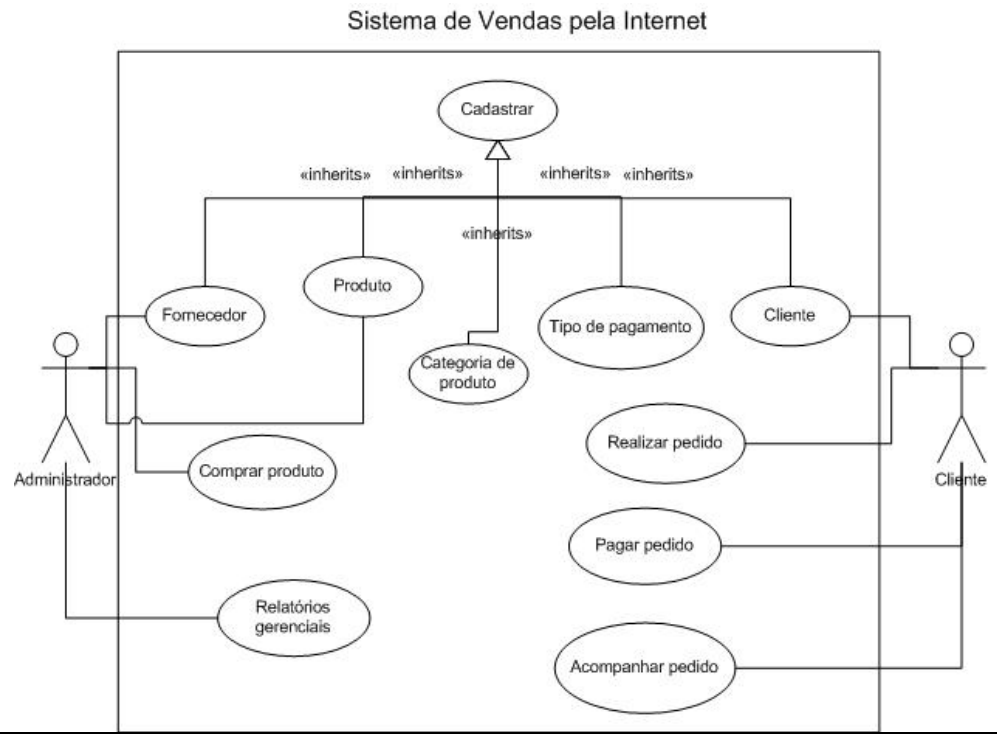
Quadro 5.1 – Nome do projeto e papel Colaborador

1 - Nome Projeto: Sistema de Vendas pela Internet
2 - Colaborador: Integrante A

Quadro 5.2 – Trecho dos objetivos do sistema representados por diagrama de casos de uso

3 – Trecho dos Objetivos do sistema:

3.1 – Diagrama de casos de uso:



Quadro 5.3 – Trecho da descrição geral do caso de uso Realizar pedido

3 – Trecho dos Objetivo do sistema:

3.2 – Trecho da Descrição geral dos casos de uso:

Casos de uso	
Nome do Caso de Uso	**Realizar pedido**
Resumo	O cliente seleciona um produto realiza um pedido desse produto.
Atores	Cliente
Pré-condições	01- Os produtos devem estar cadastrados no sistema.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
01- O cliente escolhe uma categoria de produtos.	
	02- O sistema mostra os produtos da categoria escolhida.
03- O cliente seleciona um produto.	
	04- O sistema exibe as informações gerais do produto.
05- O cliente adiciona o produto ao carrinho de compras.	
06- O cliente finaliza o seu pedido.	

Quadro 5.4 – Visão do sistema e atribuições do projeto

4 - Visão do Sistema

O sistema de comércio eletrônico (*e-commerce*) deve controlar o fluxo de compras e vendas em lojas virtuais. Não deve ser desenvolvido para uma loja específica e sim de modo genérico para ser comercializado e customizado de acordo com as necessidades dos que o adquirirem. Não é do escopo do sistema abordar o controle de estoque de produtos, pois a Empresa T apenas realiza a comercialização, ou seja, faz o anúncio de produtos e quando ocorre uma venda há a necessidade de efetuar a correspondente compra de seus fornecedores. Assim, o sistema deve possibilitar o cadastro de: fornecedores, produtos de diversas categorias e de clientes. Clientes efetuam compras, fazem o seu cadastro no sistema, realizam o pagamento do seu pedido e recebem seus produtos em um prazo máximo de sete dias úteis (atendendo apenas ao território nacional). O sistema também deve disponibilizar um ambiente para que os clientes possam acompanhar o atendimento de seus pedidos. Diversos relatórios são produzidos para que o administrador da loja possa acompanhar as compras e vendas realizadas.

5 – Papéis:

Engenheiro de Requisitos: integrante C;
Desenvolvedor: integrantes B, D e E.

Seguindo o fluxo do processo como apresentado na Figura 4.2, o pacote de processo **Controle de Requisitos** foi realizado como descrito a seguir. Durante o conjunto de trabalho *Elicitar e Especificar Requisitos*, apresentado na Subseção 4.3.2, o Engenheiro de Requisitos e o Colaborador realizaram atividades de elicitação e especificação de requisitos de acordo com o padrão *Desenvolvimento espiral,* apresentado no Quadro 3.5, ou seja, os requisitos inicialmente identificados no artefato ***Escopo do Projeto*** foram elicitados e especificados em três iterações, com cada iteração aperfeiçoando e detalhando as funções do sistema.

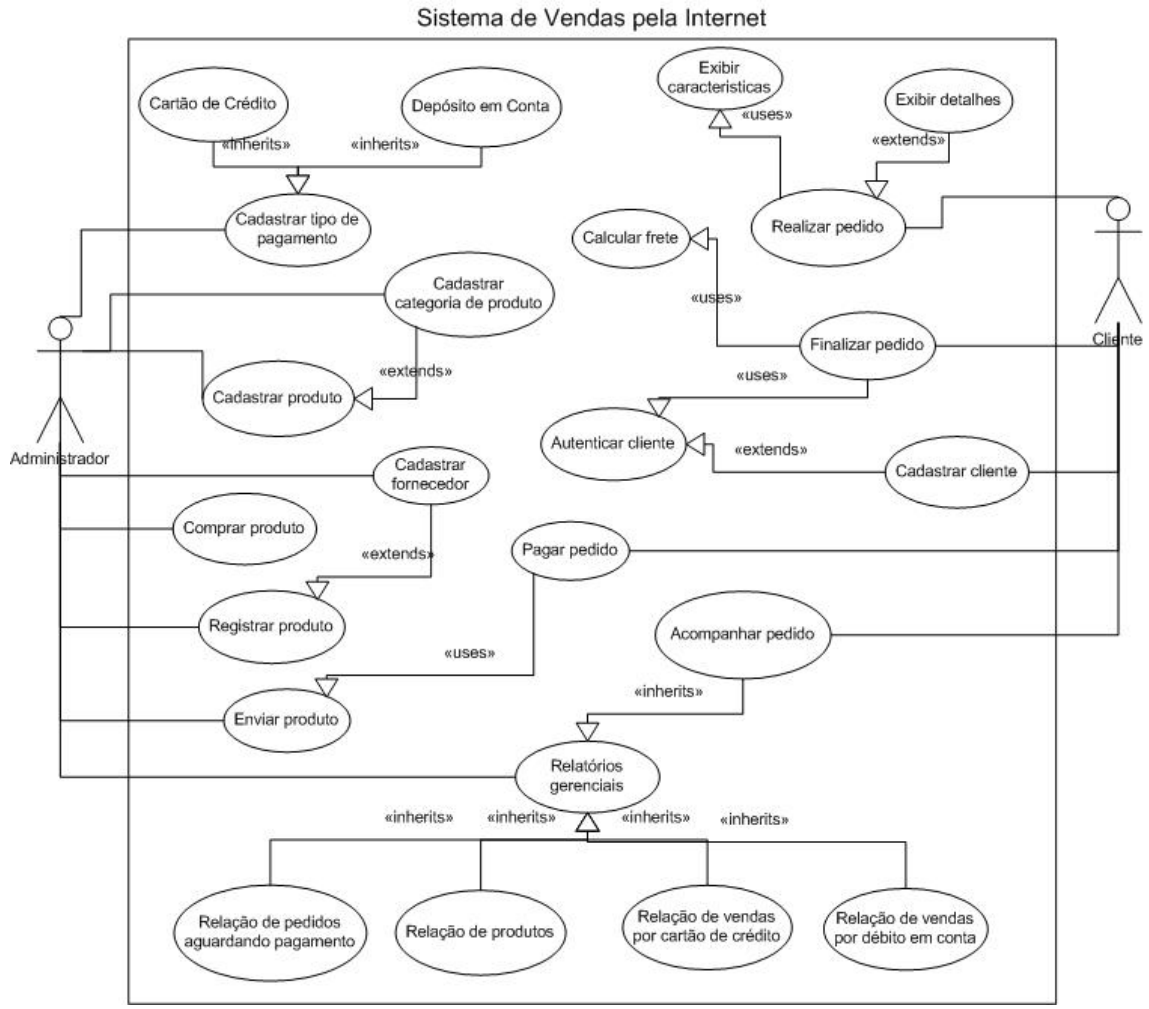
O Engenheiro de Requisitos identificou com o Colaborador as possíveis fontes de informações para elicitar requisitos. Duas pessoas externas à equipe foram identificadas e o Engenheiro de Requisitos utilizou a técnica de entrevista para realizar a elicitação dos requisitos, por ser adequada quando há poucas fontes de informações. Após entrevistar as duas pessoas identificadas como fontes de informações, o Engenheiro de Requisitos e o Colaborador especificaram os requisitos do sistema de acordo com o padrão *Crie uma diretriz de especificação acompanhando o trabalho de um analista,* apresentado no Quadro 3.6. Os requisitos funcionais foram especificados por meio de casos de uso, de acordo com os passos da solução do padrão *Cenários definem o problema*, apresentado no Quadro 3.7, e os requisitos não funcionais por meio de descrição textual.

O Engenheiro de Requisitos e o Colaborador, de acordo com os passos da solução do padrão *Agrupe requisitos às características,* apresentado no Quadro 3.9, identificaram características representativas das funções gerais do sistema: cadastros do sistema, pedidos de venda de produtos, compra de produtos de fornecedores e relatórios gerenciais. Assim, todos os requisitos conhecidos foram agrupados de acordo com essas características ao término de cada uma das três iterações de elicitação e especificação de requisitos.

Ao término de cada ciclo do conjunto de trabalho *Elicitar e Especificar Requisitos*, o artefato ***Documento de Requisitos*** foi produzido e aperfeiçoado utilizando o gabarito apresentado no Quadro 4.2. Um trecho desse artefato, referente ao término da primeira iteração, é apresentado de forma decomposta nos Quadros: 5.5 com o diagrama de casos de uso; 5.6 com a descrição do caso de uso **Realizar pedido**; 5.7 com a descrição do requisito não funcional **RNF14 - Tempo de resposta para exibir detalhes de produtos**; e 5.8 com o agrupamento dos requisitos de acordo com as características do sistema.

Quadro 5.5 – Trecho do diagrama de casos de uso da primeira iteração

Diagrama de casos de uso:



Quadro 5.6 – Trecho da descrição do caso de uso Realizar pedido

Nome do Caso de Uso	**UC09 - Realizar pedido**
Resumo	O cliente seleciona um produto e o adiciona ao carrinho de compras.
Atores	Cliente
Pré-condições	01- Os produtos devem estar cadastrados no sistema.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1- O cliente escolhe uma categoria de produto.	
	2- O sistema mostra a foto e o nome dos produtos da categoria escolhida em ordem alfabética.
3- O cliente seleciona um produto.	
	4- O sistema exibe as características gerais do produto.

Continuação do **Quadro 5.6** - Trecho da descrição do caso de uso Realizar pedido

Fluxo Principal	
Ações do Ator	**Ações do Sistema**
5- O cliente escolhe uma opção de visualizar detalhes do produto.	
	6- O sistema exibe os detalhes do produto.
7- O cliente informa a quantidade de itens e adiciona o(s) produto(s) ao carrinho de compras.	
Pós-condições	O carrinho de compras do cliente é adicionado do produto escolhido.

Quadro 5.7 – Descrição do requisito não funcional Tempo de resposta para exibir detalhes de produtos

Requisitos não Funcionais
Identificador do Requisito: RNF14 - Tempo de resposta para exibir detalhes de produtos.
Descrição: O tempo de resposta desejável para que o sistema exiba os detalhes dos produtos deve ser menor que 10 segundos.

Quadro 5.8 – Trecho dos requisitos da primeira iteração agrupados de acordo com as características do sistema

Requisitos agrupados de acordo com características		
Sistema	**Características**	**Requisitos**
Vendas pela Internet	Cadastros do sistema	UC01 – Cadastrar cliente UC02 – Cadastrar categoria de produto UC03 – Cadastrar produto UC04 – Cadastrar fornecedor UC05 – Cadastrar tipo de pagamento UC05.1 - Cartão de crédito UC05.2 - Depósito em conta
	Pedidos de venda de produtos	UC09 - Realizar pedido UC09.1 - Exibir caracteristicas UC09.2 - Exibir detalhes UC10 - Finalizar pedido RNF11 - Tempo de resposta para exibir detalhes de produtos. UC12 - Calcular frete UC13 – Pagar pedido UC14 - Autenticar cliente
	Compra de produtos de fornecedores	UC06 – Comprar produto UC07 – Registrar produto UC08 – Enviar produto
	Relatórios gerenciais	UC15 - Acompanhamento de pedido UC016 – Relação de produtos UC020 – Relação de pedidos aguardando pagamento UC021 – Relação de vendas por cartão de crédito UC022 – Relação de vendas por débito em conta

A Figura 5.1 exemplifica o aperfeiçoamento dos requisitos durante as três iterações de elicitação e especificação, exemplificando a evolução do caso de uso **UC09 - Realizar**

pedido. Esse caso de uso foi inicialmente descrito no artefato *Escopo do Projeto*, como apresentado no Quadro 5.3 e durante a primeira iteração houve a necessidade de desmembra-lo em dois casos de uso: **UC09 - Realizar pedido,** apresentado no Quadro 5.9 e **UC10 - Finalizar pedido**.

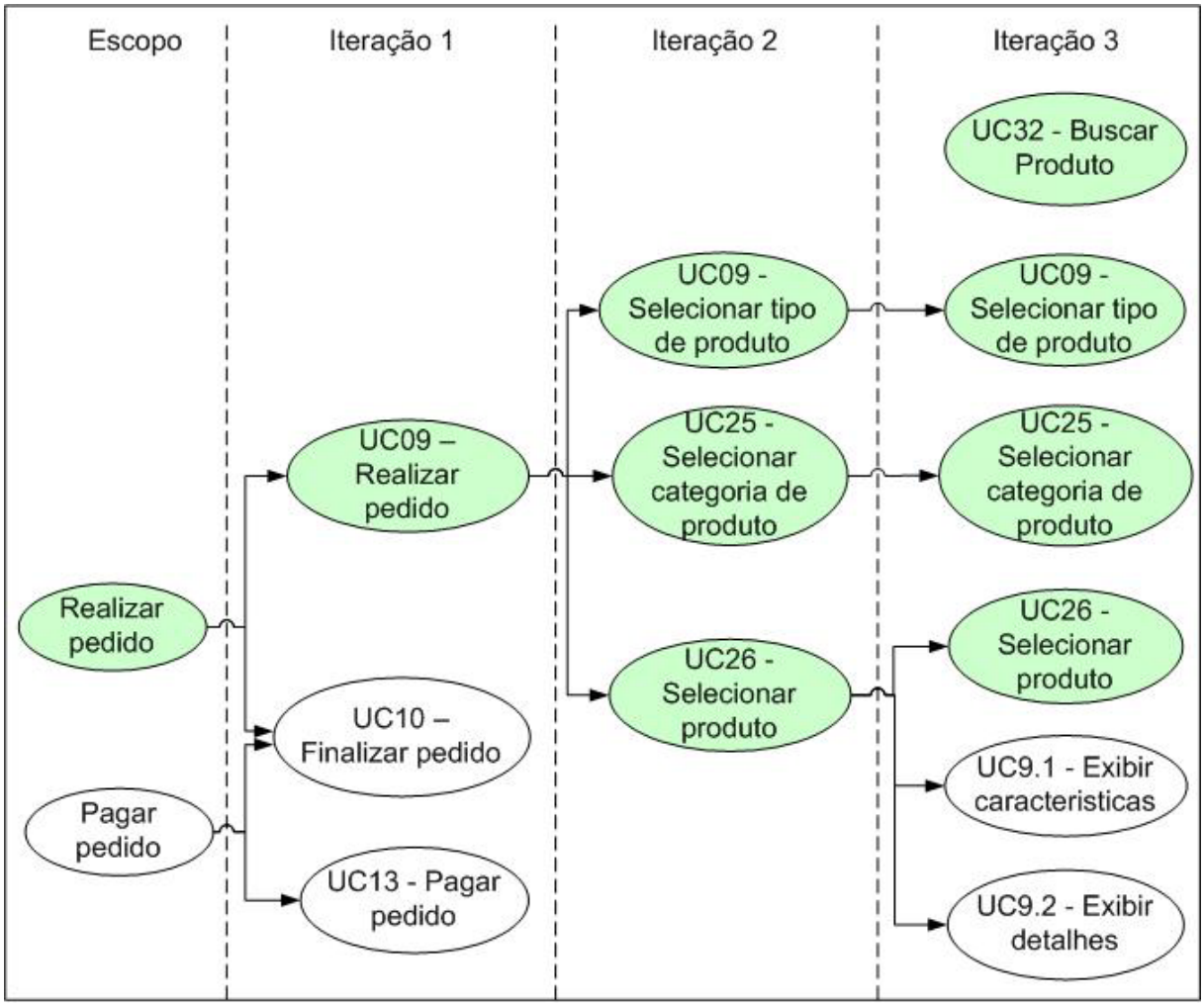


Figura 5.1 – Progresso do caso de uso Realizar pedido.

Na segunda iteração o caso de uso **UC09 - Realizar pedido** foi desmembrado nos casos de uso: **UC09 - Selecionar tipo de produto**, **UC25 - Selecionar categoria de produto** e **UC26 - Selecionar produto**, respectivamente apresentados nos Quadros 5.9, 5.10 e 5.11.

Quadro 5.9 – Trecho da descrição do caso de uso UC09 - Selecionar tipo de produto

Nome do Caso de Uso	**UC09 - Selecionar tipo de produto**
Resumo	O cliente seleciona um tipo de produto e o sistema exibe as categorias e os produtos desse tipo.
Atores	Cliente
Pré-condições	01- Os produtos devem estar cadastrados no sistema.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
01- O cliente seleciona um tipo de produto.	
	02- O sistema exibe a foto e o nome dos produtos do tipo escolhido, com os dez produtos mais vendidos sendo exibidos primeiramente e os demais produtos exibidos em ordem alfabética.
	03- O sistema mostra as categorias do tipo de produto selecionado.
04- O cliente seleciona uma categoria.	
	05- Executa o caso de uso **UC25 - Selecionar categoria de produto**
Fluxo Alternativo 1	
Ações do Ator	**Ações do Sistema**
	01- O sistema exibe a foto e o nome dos produtos do tipo escolhido, com os dez produtos mais vendidos sendo exibidos primeiramente e os demais produtos exibidos em ordem alfabética.
02- O cliente seleciona um produto.	
	03- Executa o caso de uso **UC26 - Selecionar produto**
Pós-condições	Os produtos e as categorias do tipo de produto selecionado são exibidos.

Quadro 5.10 – Trecho da descrição do caso de uso UC25 - Selecionar categoria de produto

Nome do Caso de Uso	**UC25 - Selecionar categoria de produto**
Resumo	O cliente seleciona uma categoria do tipo de produto e o sistema exibe os produtos dessa categoria.
Atores	Cliente
Pré-condições	01- Os produtos devem estar cadastrados no sistema. 02- Foi escolhido um tipo de produto.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
01- O cliente seleciona uma categoria.	
	02- O sistema exibe a foto e o nome dos produtos da categoria escolhida, com os dez produtos mais vendidos sendo exibidos primeiramente e os demais produtos exibidos em ordem alfabética.

Continuação do **Quadro 5.10** – Trecho da descrição do caso de uso UC25 - Selecionar categoria de produto

Fluxo Alternativo 1	
Ações do Ator	**Ações do Sistema**
	01- O sistema exibe a foto e o nome dos produtos da categoria escolhida, com os dez produtos mais vendidos sendo exibidos primeiramente e os demais produtos exibidos em ordem alfabética.
02- O cliente seleciona um produto.	
	03- Executa o caso de uso **UC26 - Selecionar produto**
Pós-condições	Os produtos da categoria selecionada são exibidos.

Quadro 5.11 – Trecho da descrição do caso de uso UC26 - Selecionar produto

Nome do Caso de Uso	**UC26 - Selecionar produto**
Resumo	O cliente seleciona um produto e o adiciona ao carrinho de compras.
Atores	Cliente
Pré-condições	01- Os produtos devem estar cadastrados no sistema.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
01- O cliente seleciona um produto.	
	02- O sistema exibe as características gerais do produto.
03- O cliente escolhe a opção para visualizar detalhes do produto.	
	04- O sistema exibe os detalhes do produto.
05- O cliente informa a quantidade de itens a ser comprado do produto.	
06- O cliente adiciona o(s) item(ns) do produto ao carrinho de compras.	
Fluxo Alternativo 3	
Ações do Ator	**Ações do Sistema**
01- O cliente escolhe a opção de visualizar detalhes do produto.	
	02- O sistema informa que não há detalhes do produto cadastrado no sistema.
Pós-condições	O carrinho de compras do cliente é adicionado do produto escolhido.

Na terceira iteração, requisitos referentes à busca por produtos foram elicitados e especificados por meio do caso de uso **UC32 – Buscar produto** como apresenta o Quadro 5.12, os casos de uso **UC09 - Selecionar tipo de produto, UC25 - Selecionar categoria de produto e UC26 - Selecionar produto** também foram adicionados do fluxo alternativo que faz referencia a busca de produtos e foram especificadas as funções relacionadas as características gerais e detalhes do produto.

Quadro 5.12 – Trecho da descrição do caso de uso UC32 - Buscar produto.

Nome do Caso de Uso	**UC32 - Buscar produto**
Resumo	O cliente informa uma palavra chave, o sistema realiza busca e exibe os produtos relacionados.
Atores	Cliente
Pré-condições	01- Os produtos devem estar cadastrados no sistema.
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
01- O cliente informa uma palavra chave no campo busca e seleciona a opção de pesquisa.	
	02- O sistema realiza uma busca nos nomes e descrições dos produtos que coincidem com a palavra informada.
	03- O sistema exibe os produtos encontrados.
04- O cliente seleciona um produto.	
	05- Executa o caso de uso **UC26 - Selecionar produto**
Fluxo Alternativo 1	
Ações do Ator	**Ações do Sistema**
	01- O sistema realiza uma busca nos nomes e descrições dos produtos que coincidem com a palavra informada.
	02- O sistema informa que não há produtos cadastrados com a palavra informada.
Pós-condições	Os produtos pesquisados são exibidos ao cliente.

Após a especificação dos requisitos com casos de uso realizou-se o conjunto de trabalho *Priorizar e Estimar Requisitos*, apresentado na Subseção 4.3.2 e de acordo com o padrão *Identificação de elemento,* apresentado no Quadro 3.10. O Colaborador informou os requisitos do sistema que ele gostaria que fossem atendidos primeiramente, priorizando-os na seguinte ordem: requisitos relacionadas aos cadastros do sistema, aos pedidos de venda de produtos, a compra de produtos de fornecedores e por fim os requisitos relacionadas aos relatórios gerenciais. O Engenheiro de Requisitos e o Gerente de Projetos, de acordo com os passos da solução do padrão *Identificação de elemento,* apresentado no Quadro 3.10, alocaram os requisitos priorizados em formato de pilha com os requisitos de maior prioridade no topo da pilha, como mostra o exemplo da Figura 5.2. Nesse exemplo alguns requisitos relacionados às funções de cadastros são mostrados no topo da pilha, alguns relacionados aos pedidos de venda de produtos aparecem logo abaixo seguidos dos requisitos relacionados à compra de produtos de fornecedores, e no fim da pilha aparecem alguns requisitos relacionados aos relatórios gerenciais.

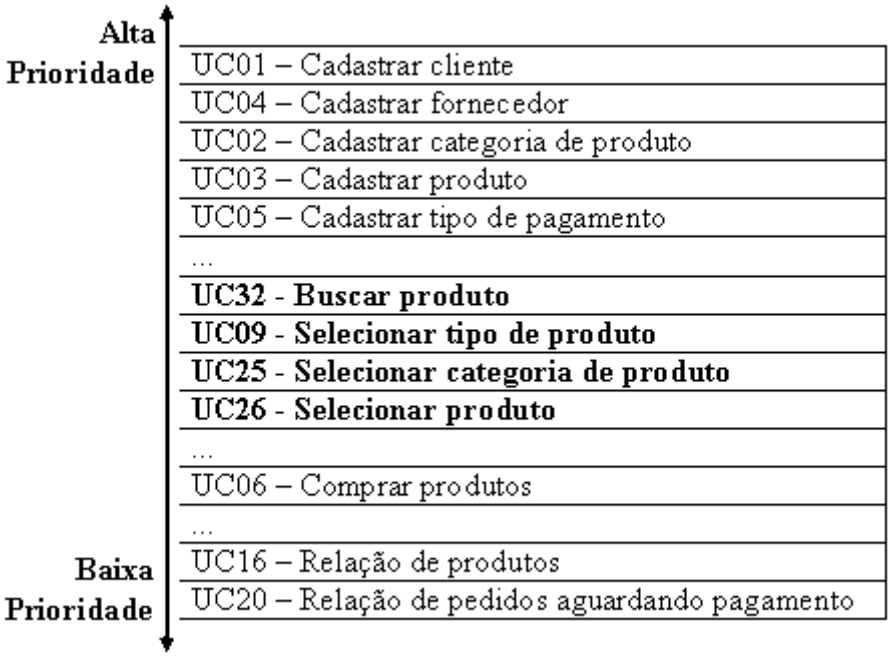


Figura 5.2 – Exemplo da pilha de requisitos

Após alocar os requisitos na pilha o Gerente de Projetos e o Engenheiro de Requisitos estimaram os tempos de desenvolvimento dos requisitos através da técnica Pontos de Casos de Uso (PCU), apresentada na Subseção 2.4.1.1. Entretanto, após as estimativas o Gerente de Projetos ajustou os tempos estimados considerando o histórico de projetos semelhantes já desenvolvidos pela Empresa T. Esse ajuste ocorreu ao término das contagens dos PCU no momento em que os tempos, em horas, já estavam calculados.

Para exemplificar o cálculo das estimativas, a seguir são apresentados os passos da estimativa por PCU do caso de uso **UC26 - Selecionar produto**. A Figura 5.3 apresenta o trecho do diagrama de casos de uso necessário para a visualização dos relacionamentos dos atores e dos casos de uso.

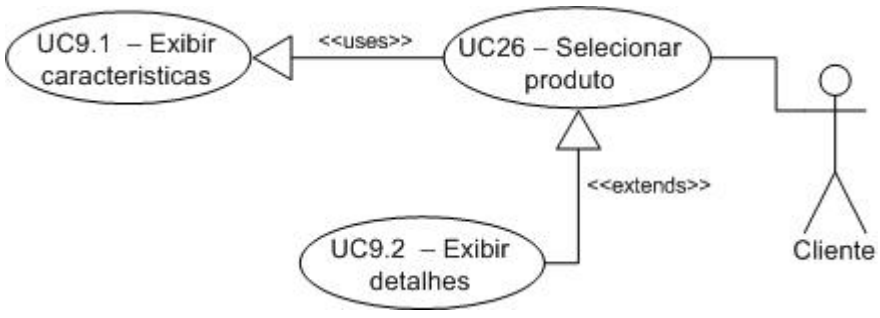


Figura 5.3 – Trecho do diagrama de casos de uso referente ao UC26 - Selecionar produto.

Como apresentado na Figura 5.3, o caso de uso **UC26 - Selecionar produto** possui relacionamento com apenas um ator. Esse ator é classificado, de acordo com a Tabela 2.4, com complexidade alta, pois esse ator interage com o caso de uso é através de páginas da web. Assim, temos a contagem de um ator, o qual recebe o peso igual a três de acordo com a Tabela 2.4.

Na Figura 5.3 é verificado também que o caso de uso **UC26 - Selecionar produto** está relacionado a outros dois casos de uso, obtendo um total de três casos de uso relacionados. Esses casos de uso são classificados, de acordo com a Tabela 2.5, com complexidade simples, pois esses casos de uso devem ser realizados com menos de cinco classes de análise. Assim, temos a contagem de três casos de uso, os quais recebem o peso igual a cinco de acordo com a Tabela 2.5.

De posse das informações das contagens e pesos dos atores e casos de uso, realiza-se o cálculo do valor. O valor é o resultado da multiplicação do peso pela contagem. Após o cálculo dos valores, realiza-se o calculo do total somando-se todos os valores. Para os atores, chega-se ao total de três e para os casos de uso ao total de quinze. A Tabela 5.2 apresenta a contagem do atores e dos casos de uso envolvidos.

Tabela 5.2 – Contagem de atores e casos de uso

Complexidade	Atores			Casos de Uso		
	Peso	**Contagem**	**Valor**	**Peso**	**Contagem**	**Valor**
Simples	1	0	0	5	**3**	15
Médio	2	0	0	10	0	0
Complexo	3	**1**	3	15	0	0
		Total	**3**		**Total**	**15**

Após a contagem dos atores e dos casos de uso, foi calculado os Pontos de Casos de Uso Não Ajustados (PCUNA), somando-se o total dos atores com o total de casos de uso: **PCUNA = ∑ Peso dos Atores + ∑ Peso dos Casos de Uso**, que resultou em 18. Realizou-se a contagem dos fatores técnicos do projeto observando os pesos apresentados na Tabela 2.6, resultando em 12. Foi então realizado o cálculo do Fator de Complexidade Técnica (FCT): **(FCT) = 0,6 + (0,01 * ∑ fator técnico)**, o que resultou em 0,72.

Realizou-se a contagem dos fatores ambientais observando os pesos apresentados na Tabela 2.7, resultando em 34. Foi então realizado o cálculo do Fator de Complexidade Ambiental (FCA): **(FCA) = 1,4 + (-0,03 * ∑ do fator ambiental)**, o que resultou em 0,38.

Realizou-se o cálculo dos PCU ajustados, seguindo a fórmula: **PCU ajustados = PCU não ajustado * FT * FA**, que resultou em 4,92. Utilizou-se a produtividade de 20 pessoas-hora, conforme Karner (1993), para obter a estimativa em horas do caso de uso **UC26 - Selecionar produto** e aplicou-se a fórmula: **Estimativa de horas (ES) = PCU ajustados * pessoas-hora por unidade de PCU,** resultando em 98,50 pessoas-horas. Como a equipe é composta de cinco integrantes, realizou-se a divisão de ES por cinco, que resultou em 19,70 horas para o desenvolvimento desse caso de uso.

O ajuste realizado pelo Gerente de projetos foi de 8,5% para menos, ou seja, o caso de uso de **UC26 - Selecionar produto** passou para o tempo estimado de 18,02 horas.

Ao término do conjunto de trabalho *Priorizar e Estimar Requisitos* o artefato ***Documento de Requisitos Priorizado e Estimado*** foi produzido utilizando o gabarito apresentado no Quadro 4.3. Um trecho desse artefato é apresentado na Tabela 5.3, com a primeira coluna mostrando os requisitos empilhados de acordo com as prioridades e a segunda o esforço estimado, em horas, para o desenvolvimento de cada requisito.

Tabela 5.3 – Trecho do artefato Documento de Requisitos Priorizado e Estimado

Requisitos Priorizados e Estimados	
Pilha de Requisitos	**Esforço Estimado (horas)**
...	...
UC03 – Cadastrar produto	8,20
UC05 – Cadastrar tipo de pagamento	12,04
...	...
UC26 - Selecionar produto	18,02

Seguindo o fluxo do processo como apresentado na Figura 4.2, o pacote de processo **Gestão de Versão do Sistema** foi realizado como descrito a seguir. Durante a atividade *Definir Versão*, apresentada na Subseção 4.3.3 o Gerente de Projetos investigou na literatura quais os tempos médios de desenvolvimento de versões de software funcional e estipulou que cada versão do sistema não poderia gastar mais do que quinze dias de desenvolvimento. O tempo médio de quinze dias foi definido de acordo com a pesquisa realizada em Ambler (2007), em que organizações com equipes e projetos semelhantes ao da Empresa T, realizam suas iterações em quinze dias. Como a equipe trabalha oito horas diárias, os tempos estimados de cada requisito foram divididos por oito horas para determinar quantos dias cada requisito levaria para ser atendido. O Gerente de Projetos e o Engenheiro de Requisitos consultaram a pilha de requisitos e somaram os dias estimados dos requisitos do topo da pilha, até que não

ultrapassasse os quinze dias estipulados. Como exemplo, o requisito **UC26 - Selecionar produto** que foi estimado em 18,02 horas teve esse tempo dividido por oito horas resultando em 2,25 dias para ser atendido, incluindo etapas de projeto, codificação, teste e entrega.

O Gerente de Projetos alocou então os requisitos na primeira versão do sistema, a qual teve os requisitos estimados em quatorze dias e tratou das funções referentes aos cadastros do sistema. Após a alocação dos requisitos na primeira versão, o Engenheiro de Requisitos de acordo com o os passos da solução do padrão *Unidade de Propósito*, apresentado no Quadro 3.11*,* reuniu todos os envolvidos com o projeto e realizou uma apresentação geral dos requisitos do sistema durante vinte minutos.

Ao término da atividade *Definir Versão,* o artefato ***Documento de Requisitos Priorizado, Estimado e Plano da Versão*** foi produzido utilizando o gabarito apresentado no Quadro 4.4. Um trecho desse artefato é apresentado na Tabela 5.4, com a primeira coluna mostrando a pilha de requisitos, a segunda o esforço estimado, em horas, e a terceira o número da versão em que os requisitos foram sendo alocados para serem atendidos.

Tabela 5.4 – Trecho do artefato Documento de Requisitos Priorizado, Estimado e Plano da Versão.

Documento de Requisitos Priorizado e Estimado e Plano da Versão		
Requisito	**Esforço Estimado**	**Nº da Versão**
...	...	...
UC03 – Cadastrar produto	8,20	01
UC05 – Cadastrar tipo de pagamento	12,04	01
...	...	...
UC26 - Selecionar produto	18,02	02

Após o planejamento de cada versão do sistema, as atividades de projeto, codificação e teste de software foram realizadas seguindo diretrizes do processo já utilizado pela Empresa T. Entretanto, durante o desenvolvimento do projeto de software da primeira versão, o Colaborador solicitou modificações no requisito **UC05 – Cadastrar tipo de pagamento**. Esse requisito foi inicialmente especificado com as formas de pagamento por cartão de crédito e depósito em conta corrente, entretanto o Colaborador reportou a necessidade do sistema disponibilizar também pagamentos por boleto bancário. O Gerente de Projetos e o Engenheiro de Requisitos analisaram a solicitação de modificação nesse requisito de acordo com os passos da solução do padrão *Forneça declaração de objetivos para cada requisito*, apresentado no Quadro 3.12 e propuseram soluções para esse requisito observando que a

solução proposta não teria impacto no prazo e no custo da versão em desenvolvimento. Assim, o requisito **UC05 – Cadastrar tipo de pagamento** foi modificado conforme as necessidades do Colaborador e reescrito no artefato ***Documento de Requisitos*****,** pois o Gerente de Projetos e o Engenheiro de Requisitos argumentaram que esse artefato não havia entrado em controle formal de mudanças, podendo ser alterado.

Houve a entrega da primeira versão do sistema ao Cliente e ainda que existissem requisitos a serem desenvolvidos na pilha de requisitos, modificações no requisito **UC13 – Pagar pedido** foi necessário. Conforme o fluxo do processo apresentado na Figura 4.6, realizou-se novamente o pacote de processo **Controle de Requisitos**, como descrito a seguir. Durante o conjunto de trabalho *Elicitar e Especificar Requisitos,* apresentado na Subseção 4.3.2, o Engenheiro de Requisitos e o Colaborador modificaram o requisito **UC13 – Pagar pedido** para atender também ao pagamento por boleto bancário e as duas pessoas identificadas como fontes de informações foram novamente entrevistadas.

Novos requisitos foram então elicitados e especificados por meio de casos de uso, de acordo com os passos da solução do padrão *Cenários definem o problema*, apresentado no Quadro 3.7. Esses requisitos, relacionados a relatórios gerenciais, foram elicitados e especificados em apenas uma iteração e foram agrupados de acordo com os passos da solução do padrão *Agrupe requisitos a características,* apresentado no Quadro 3.9. Ao término desse conjunto de trabalho o artefato ***Documento de Requisitos*** foi atualizado e realizou-se o conjunto de trabalho *Priorizar e Estimar Requisitos,* apresentado na Subseção 4.3.2.

O Colaborador informou as prioridades dos requisitos novos e dos modificados e o Gerente de Projetos e o Engenheiro de Requisitos, de acordo com os passos da solução do padrão *Identificação de elemento*, apresentado no Quadro 3.10, alocaram esses requisitos na pilha de requisitos, reorganizando-a, e estimaram o esforço de desenvolvimento desses requisitos. Ao término das estimativas, o artefato ***Documento de Requisitos Priorizado e Estimado*** foi atualizado e conforme apresentado na Figura 4.5, realizou-se o pacote de processo **Gestão de Versão do Sistema**, como descrito a seguir.

Durante a atividade *Definir Versão*, apresentada na Subseção 4.3.3 o Gerente de Projetos e o Engenheiro de Requisitos consultaram a pilha de requisitos e somaram os dias estimados dos requisitos do topo da pilha até que não ultrapassasse os quinze dias estipulados para o desenvolvimento de versões do sistema. O Gerente de Projetos alocou então os requisitos na segunda versão do sistema, a qual teve os requisitos estimados em quinze dias e tratou das funções referentes a pedidos de venda de produtos e compra de produtos de fornecedores.

Após a alocação dos requisitos na segunda versão o Engenheiro de Requisitos, de acordo com o os passos da solução do padrão *Unidade de Propósito*, apresentado no Quadro 3.11, reuniu todos os envolvidos com o projeto e realizou uma apresentação geral dos requisitos novos e dos modificados durante três minutos. Ao término da atividade *Definir Versão,* o artefato ***Documento de Requisitos Priorizado, Estimado e Plano da Versão*** foi atualizado e realizou-se as atividades de projeto, codificação e testes de software de acordo com o processo já utilizado pela Empresa T. Durante essas atividades, não houve a necessidade modificações.

O sistema, com a segunda versão, foi então entregue ao Cliente e havia requisitos na pilha de requisitos a serem desenvolvidos, sem a necessidade de modificações ou novo ciclo de elicitação e especificação de requisitos. O Gerente de Projetos alocou na terceira versão do sistema todos os requisitos restantes da pilha, cujo tempos estimados somados foi de nove dias. A terceira versão do sistema tratou dos requisitos referentes a relatórios gerenciais.

O sistema, com a terceira versão, foi então entregue ao Cliente e houve a necessidade de elicitar novos requisitos, pois o Colaborador solicitou novos relatórios gerenciais. Assim, de acordo com o fluxo do pacote de processo **Gestão de Versão do Sistema** apresentado na Figura 4.6, realizou-se o pacote de processo **Controle de Requisitos**.

Requisitos relacionados a relatórios gerenciais para o sistema foram elicitados e especificados em duas iterações. Realizou-se a priorização e as estimativas dos requisitos e a pilha de requisitos foi atualizada. O Gerente de Projetos alocou na quarta versão do sistema todos os requisitos restantes da pilha de requisitos, cujo dias estimados somados foi de cinco dias. Esses requisitos foram apresentados a todos os envolvidos com o projeto durante cinco minutos e a quarta versão do sistema foi realizada sem a necessidade de modificações nos requisitos. O sistema, com a quarta versão, foi entregue ao Cliente finalizando o projeto.

5.4 - Resultados da aplicação do questionário a equipe

Para avaliação do processo proposto e da integração dos padrões de requisitos e organizacionais ao processo, um questionário foi elaborado, apresentado no Apêndice A e respondido por todos os integrantes da equipe. Esse questionário investigou as vantagens e desvantagens do processo proposto, a integração das práticas propostas pelos padrões ao processo e a modelagem do processo com o SPEM. Algumas perguntas do questionário possuem características que podem variar de acordo com o papel desempenhado pelos

integrantes da equipe. Por exemplo, o Gerente de Projetos possui uma visão diferente dos Desenvolvedores com relação à atividade *Identificar Escopo*, apresentada na Subseção 4.3.1, pois os Desenvolvedores não participaram da realização dessa atividade. Assim, para responder as perguntas 1a, 2a, 2d, 2e, 2f, 3a e 3b do questionário foi solicitado aos integrantes que considerassem os papéis que eles desempenharam no projeto. As perguntas 1b, 2b e 2c, que são de âmbito geral foram respondidas de forma conjunta pelos integrantes da equipe.

As respostas do questionário aplicado aos integrantes da equipe são apresentadas na próxima Subseção.

5.4.1 - Respostas do questionário aplicado

5.4.1.1 - Respostas da pergunta 1a

Pergunta 1a: Quais as principais vantagens e desvantagens do processo proposto com relação ao processo já utilizado pela Empresa T? Justifique sua resposta.

Essa pergunta teve algumas respostas semelhantes por todos os integrantes da equipe, quanto às vantagens do processo proposto:

- **Desenvolvimento do sistema em versões de software funcional**: proporciona motivação aos integrantes da equipe, pois eles visualizam o sistema funcionando. De acordo com dados históricos de projetos anteriores desenvolvidos na Empresa T, os integrantes das equipes tendem a ficar desmotivados quando um projeto se estende por mais de seis semanas de desenvolvimento.
- **Participação ativa do Cliente nas tomadas de decisões**: proporciona desenvolver o sistema de acordo com as reais necessidades do cliente, minimizando retrabalhos na identificação, especificação e controle de requisitos.
- **Facilidade com que modificações em requisitos podem ser tratadas durante o desenvolvimento**: proporciona a análise e tomada de decisões rapidamente, sem a necessidade de controles formais de mudanças.

Os integrantes da equipe também mostraram diferentes opiniões relacionadas às vantagens e desvantagens do processo:

- **Gerente de Projetos**

- **Vantagens:**
  - Uso da técnica PCU e realização das estimativas durante a engenharia de requisitos proporcionaram antecipação nas estimativas dos requisitos do sistema, pois nos projetos desenvolvidos pela Empresa T as estimativas são realizadas durante as atividades de Projeto de Software.
  - Aumento da produtividade e comprometimento dos integrantes da equipe em relação aos projetos anteriores realizados na empresa.
- **Desvantagens:**
  - Dificuldade de estabelecer um compromisso com o Cliente por meio da indicação de uma pessoa para desempenhar o papel de Colaborador, pois nem sempre o Cliente possui essa conveniência;
  - Os objetivos do sistema descritos apenas com casos de uso não transmitem todas as informações necessárias para o Cliente, quando esse não conhece a técnica de modelos de casos de uso.

- **Colaborador**
  - **Vantagem:**
    - Apontou a vantagem da sua participação na priorização dos requisitos, pois o sistema é entregue em versões desenvolvidas em curto espaço de tempo (máximo de quinze dias) e de acordo com as suas preocupações.
  - **Desvantagem:**
    - Falta de conhecimento na técnica de casos de uso pode dificultar o seu relacionamento com a especificação dos requisitos do sistema.

- **Engenheiro de Requisitos**
  - **Vantagens:**
    - Descrição dos objetivos do sistema com casos de uso facilita a compreensão do sistema a ser desenvolvido e proporciona o reúso dos casos de uso nas especificações dos requisitos;
    - Participação do Cliente na especificação dos requisitos proporciona que o sistema atenda as reais necessidades do Cliente.
  - **Desvantagem:**

- Falta de um controle de rastreabilidade entre requisitos dependentes, evidenciando que esse controle facilitaria controlar modificações em requisitos.

- **Desenvolvedores**
  - **Vantagens:**
    - A especificação dos requisitos com casos de uso, pois proporciona melhor entendimento dos requisitos do sistema;
    - Apresentação dos requisitos antes de realizar atividades de Projeto de Software, pois proporciona entendimento uniforme dos requisitos que devem ser atendidos.
  - **Desvantagens:**
    - Não apontaram desvantagens.

5.4.1.2 - Respostas da pergunta 1b

Pergunta 1b: Quais conjuntos de trabalho/atividades foram utilizados e quais não foram utilizados? Justifique sua resposta.

Nem todos os conjuntos de trabalho/atividades do processo proposto foram utilizados para o desenvolvimento do sistema, pois a Empresa T utilizou-se do seu processo habitual para desenvolver atividades de projeto, codificação e testes de software. A Tabela 5.5 apresenta todos os conjuntos de trabalho/atividades utilizados e não utilizados.

Tabela 5.5 – Conjuntos de trabalho/atividades utilizados e não utilizados

	Conjuntos de Trabalho/Atividades
Utilizados	- Atividade *Identificar Escopo*, apresentada na Subseção 4.3.1; - Conjunto de trabalho *Elicitar e Especificar Requisitos*, apresentado na Subseção 4.3.2; - Conjunto de trabalho *Priorizar e Estimar Requisitos*, apresentado na Subseção 4.3.2; - Atividade *Definir Versão*, apresentada na Subseção 4.3.3.
Não Utilizados	- Conjunto de trabalho *Planejar Testes*, apresentado na Subseção 4.3.3; - Conjunto de trabalho *Desenvolver Iteração, apresentado na* Subseção 4.3.3; - Conjunto de trabalho *Finalizar Versão*, apresentado na Subseção 4.3.3.

5.4.1.3 - Respostas das perguntas 2a e 2b

Pergunta 2a: Você acredita que os padrões inseridos auxiliaram o processo? Justifique sua resposta.

Todos os integrantes da equipe classificaram que os padrões auxiliaram o processo proposto, pois os padrões estabeleceram diretrizes a serem seguidas para guiar a realização dos conjuntos de trabalho/atividades do processo proposto.

Pergunta 2b: Quais padrões não foram utilizados? Justifique sua resposta.

- **Padrão** *Equipe de escrita pequena*, apresentado no Quadro 3.8, pois esse padrão menciona o fato de limitar a três o número de pessoas para a especificação dos requisitos e o projeto possuía apenas duas pessoas para a realização dessa tarefa;
- **Padrão** *Detalhe a especificação escrevendo casos de teste*, apresentado no Quadro 3.13, pois esse padrão menciona que caso haja modificações em requisitos durante atividades de projeto de software, essas modificações devem ser especificadas nos casos de teste do sistema, mas o Gerente de Projetos e o Engenheiro de Requisitos argumentaram que as modificações poderiam ser especificadas no artefato **Documento de Requisitos**, tendo em vista que esse artefato não havia entrado em controle formal de mudanças.

5.4.1.4 – Respostas da pergunta 2c

Pergunta 2c: Quais as principais vantagens e desvantagens que os padrões integrados ao processo e utilizados no desenvolvimento do sistema, proporcionaram para o projeto? Justifique sua resposta.

A Tabela 5.6 apresenta, na primeira coluna o nome do padrão e na segunda suas principais vantagens e desvantagens.

Tabela 5.6 – Vantagens e desvantagens dos padrões integrados ao processo proposto.

Padrão	Principais Vantagens e Desvantagens
Amplitude antes de profundidade, apresentado no Quadro 3.1	**Vantagens**: criar casos de uso gerais no início do projeto proporciona melhor entendimento dos objetivos do sistema, do que se esses objetivos fossem descritos apenas de forma textual. **Desvantagens**: necessidade do Gerente de Projetos em conhecer a técnica de modelos de casos de uso e do Cliente em entender os objetivos descritos com essa técnica.
Envolver o cliente, apresentado no Quadro 3.2	**Vantagens**: proporciona a interação entre o Cliente e a equipe do projeto durante todo o desenvolvimento do sistema. **Desvantagens**: nem sempre o cliente tem disponibilidade para atuar como Colaborador e não tem outra pessoa disponível para indicar.
Equipe auto selecionadora, apresentado no Quadro 3.3	**Vantagens**: quando há equipes com várias pessoas com habilidades comuns, esse padrão facilita na atribuição dos papéis, pois considera a opinião de cada um. **Desvantagens**: para equipes pequenas esse padrão não colabora com a tomada de decisão para a atribuição de responsabilidades, pois normalmente existem poucas pessoas com habilidades comuns.
Empregue um Engenheiro de Requisitos como encarregado, apresentado no Quadro 3.4	**Vantagens**: proporciona a realização e controle das atividades da engenharia de requisitos por pessoas com conhecimentos na área. **Desvantagens**: não foram apontadas desvantagens para esse padrão.
Desenvolvimento espiral, apresentado no Quadro 3.5	**Vantagens**: desenvolver os requisitos iterativamente facilita o desenvolvimento, pois o detalhamento dos requisitos aumenta aos poucos. **Desvantagens**: não foram apontadas desvantagens para esse padrão.
Crie uma diretriz de especificação acompanhando o trabalho de um analista, apresentado no Quadro 3.6	**Vantagens**: ter diretrizes para realizar e a especificação dos requisitos facilita a execução das atividades do Engenheiro de Requisitos e proporciona uma especificação semelhante para todos os projetos que seguem o mesmo processo. **Desvantagens**: não foram apontadas desvantagens para esse padrão.
Cenários definem o problema, apresentado no Quadro 3.7	**Vantagens**: casos de uso melhoram o entendimento dos requisitos a serem desenvolvidos. **Desvantagens**: a participação do colaborador na descrição dos casos de uso pode atrapalhar o desenvolvimento, pois nem sempre essa pessoa possui conhecimentos nessa técnica.
Agrupe requisitos às características, apresentado no Quadro 3.9	**Vantagens**: melhora da visibilidade dos requisitos do sistema por todos os envolvidos no projeto. **Desvantagens**: não foram apontadas desvantagens para esse padrão.
Identificação de elemento, apresentado no Quadro 3.10	**Vantagens**: desenvolvimento das funções que preocupam o cliente, proporcionando rápido *feedback* à equipe de desenvolvimento. **Desvantagens**: o cliente pode priorizar funções que são dependentes de requisitos ainda não especificados.
Unidade de Propósito, apresentado no Quadro 3.11	**Vantagens**: compartilhar a visão do Engenheiro de Requisitos, com relação ao entendimento dos requisitos, com os Desenvolvedores facilita o estabelecimento de consenso entre os envolvidos no projeto. **Desvantagens**: não foram apontadas desvantagens para esse padrão.
Forneça declaração de objetivos para cada requisito, apresentado no Quadro 3.12	**Vantagens**: facilidade na tomada de decisão para modificar um requisito em qualquer etapa do desenvolvimento do sistema. **Desvantagens**: não foram apontadas desvantagens para esse padrão.

5.4.1.5 – Respostas da pergunta 2d

Pergunta 2d: Quais padrões foram mais e menos relevantes para o desenvolvimento do sistema? Justifique sua resposta.

Os integrantes da equipe classificaram, dentre todos os padrões integrados ao processo e utilizados no desenvolvimento do sistema, os menos relevantes da seguinte forma: o **Gerente de Projetos**, o **Colaborador**, o **Engenheiro de Requisitos** e os **Desenvolvedores** (representados pelos integrantes **B** e **E**), classificaram o padrão *Equipe auto selecionadora,* apresentado no Quadro 3.3, argumentando que as pessoas que irão desempenhar um papel no projeto sempre opinarão de acordo com suas habilidades e tarefas que já estão acostumados a realizar na empresa; e o padrão *Empregue um Engenheiro de Requisitos como encarregado,* apresentado no Quadro 3.4, apontado também pelo **Desenvolvedor** representado pelo integrante **D**, pois esse padrão foi classificando como um procedimento inerente à engenharia de software e não uma necessidade que pode ou não ser realizada.

Já os padrões mais relevantes foram apontados pelos integrantes da equipe da seguinte forma:

- **Gerente de Projetos:**
  - Padrão *Desenvolvimento espiral*, apresentado no Quadro 3.5, pois proporciona a realização dos requisitos conforme os detalhes do sistema são conhecidos;
  - Padrão *Agrupe requisitos a características*, apresentado no Quadro 3.9, pois proporciona a visualização de todos os requisitos em grupos de características gerais do sistema;
  - Padrão *Forneça declaração de objetivos para cada requisito*, apresentado no Quadro 3.12, pois proporciona realizar modificações em requisitos durante todo o processo, sem controles formais de mudanças.
- **Engenheiro de Requisitos:**
  - Padrão *Envolver o cliente,* apresentado no Quadro 3.2, pois a presença do cliente durante a especificação dos requisitos favorece a realização dos requisitos conforme as reais necessidades dele;
  - Padrão *Amplitude antes de profundidade,* apresentado no Quadro 3.1, pois a descrição dos objetivos do sistema com casos de uso facilita a compreensão da finalidade do sistema a ser desenvolvido;

- Padrão *Cenários definem o problema,* apresentado no Quadro 3.7, pois reutiliza as descrições dos casos de uso definidos nos objetivos do sistema e proporciona que os Desenvolvedores entendam com clareza os requisitos a serem desenvolvidos, pois casos de uso são técnicas conhecidas por eles.

- **Colaborador:**
  - Padrão *Identificação de elemento*, apresentado no Quadro 3.10, pois proporciona o desenvolvimento das funções do sistema de acordo com as suas prioridades.

- **Desenvolvedores:**
  - Padrão *Crie uma diretriz de especificação acompanhando o trabalho de um analista,* apresentado no Quadro 3.6, pois proporciona que todos os projetos desenvolvidos possuam a mesma forma de especificação de requisitos;
  - Padrão *Cenários definem o problema,* apresentado no Quadro 3.7, pois possibilita melhor entendimento dos requisitos a serem desenvolvidos, minimizando problemas de ambigüidades que ocorrem quando os requisitos são escritos apenas textualmente;
  - Padrão *Unidade de propósito,* apresentado no Quadro 3.11, pois proporciona uniformizar o entendimento dos requisitos entre todos os Desenvolvedores e o Engenheiro de Requisitos.

5.4.1.6 - Respostas das perguntas 2e e 2f

Pergunta 2e: Como você classifica as descrições das soluções dos padrões, em relação ao problema/solução que o padrão se propõe a resolver?

( ) Mal Relacionado: As descrições não condizem com o problema e solução dos padrões.

( ) Razoavelmente Relacionado: As descrições condizem razoavelmente com o problema e solução dos padrões.

( ) Bem Relacionado: As descrições condizem exatamente com o problema e solução dos padrões.

Todos os integrantes da equipe afirmaram que os passos da solução estão bem relacionados e condizem exatamente com o problema e a solução dos padrões.

Pergunta 2f: Com que facilidade pode-se compreender a descrição dos passos que representam à solução dos padrões utilizados?

( ) Mal Definida: As descrições estão confusas e incompletas. Não se pôde compreender "o que" deveria ser feito.

( ) Razoavelmente Definida: As descrições estão razoavelmente confusas ou incompletas. Não se pôde compreender com precisão "o que" deveria ser feito.

( ) Bem Definido: As descrições estão bem definidas e completas. Foi possível compreender com precisão "o que" deveria ser feito.

- **Gerente de Projetos, Engenheiro de Requisitos e os Desenvolvedores (integrantes D e E**): classificaram que as descrições estão bem definidas e possibilitou a compreensão do que deveria ser realizado.
- **Desenvolvedor (integrante B)**: classificou como razoavelmente definida sem justificar a sua resposta.

5.4.1.7 - Respostas das perguntas 3a e 3b

Pergunta 3a: Facilidade de Compreensão: Com que facilidade se pode compreender o processo por meio da modelagem realizada com SPEM?

( ) Mal Definido: As atividades do processo estão mal definidas. Não se tem idéia da seqüência em que as atividades devem ocorrer.

( ) Razoavelmente Definido: As atividades do processo estão razoavelmente definidas e explicadas. Não se tem idéia detalhada da seqüência em que as atividades devem ocorrer.

( ) Bem definido: As atividades do processo estão bem definidas e explicadas. Tem-se idéia clara da seqüência em que as atividades devem ocorrer.

- **Gerente de Projetos**: classificou como razoavelmente definido, pois argumentou que os conjuntos de trabalho/atividades que cada papel deve desempenhar não estão totalmente claros na modelagem do processo.
- **Engenheiro de Requisitos e Desenvolvedores**: classificaram que o processo está bem definido, pois o processo define claramente as atividades que devem ocorrer no

projeto, os artefatos gerados e utilizados em cada atividade, bem como a seqüência que em que devem ocorrer.

Pergunta 3b: Facilidade de Manutenção: Seria fácil realizar alterações no processo para integrar novas atividades, artefatos, papéis e guias?

( ) Difícil de Alterar: Devido a sua definição inadequada seria difícil alterá-lo para atender às necessidades particulares de um projeto ou organização.

( ) Fácil de Alterar: Devido ao processo estar bem definido seria fácil alterá-lo para atender às necessidades particulares de um projeto ou organização.

Todos os integrantes da equipe classificaram o processo como fácil de alterar para atender às necessidades particulares de um projeto ou organização.

5.5 - Considerações Finais

Este capítulo apresentou, por meio de um estudo de caso realizado com uma empresa real desenvolvedora de software, que padrões de requisitos e organizacionais podem organizar, conduzir e auxiliar efetivamente a realização do desenvolvimento de software, se integrados à engenharia ágil de requisitos. Por meio desse estudo de caso e do questionário aplicado aos integrantes da equipe foi possível verificar vantagens e desvantagens do processo proposto no desenvolvimento de software da Empresa T, como apresentado a seguir. Algumas dessas vantagens já foram adotadas pela empresa e outras estão sendo avaliadas para serem integradas ao processo atualmente utilizado por ela.

No estudo de caso realizado o autor desta dissertação observou que a equipe esteve motivada durante todo o desenvolvimento do sistema. Esse fato foi atribuído, por todos os integrantes da equipe, ao desenvolvimento do sistema consistir em versões de software funcional, o que proporcionou a visualização dos resultados do trabalho da equipe em no máximo quinze dias. A prática de desenvolver software em versões funcionais está sendo cuidadosamente estudada e avaliada pela gerência de processos da Empresa T, pois essa prática modifica toda a metodologia utilizada atualmente pela empresa.

O mesmo acontece com as práticas relacionadas às modificações em requisitos durante todo o desenvolvimento, pois no processo utilizado atualmente na empresa as etapas do desenvolvimento de software acontecem de acordo com o modelo cascata e ao término de

cada etapa ocorre a criação da "*baseline*" e modificações necessitam ser controladas formalmente.

A participação do representante do Cliente durante o desenvolvimento do sistema, desempenhado pelo papel Colaborador, foi classificada pelos integrantes da equipe como uma prática que melhorou a conformidade do sistema com as reais necessidades do Cliente. Entretanto, durante a realização do estudo de caso o autor desta dissertação pôde observar que o fato do Cliente não ser real, ou seja, o Cliente era a própria Empresa T e o Integrante A acumulou os papéis Gerente de Projetos e Colaborador, dificultou a análise de alguns resultados relacionados às atividades de especificação e priorização de requisitos. Durante a realização dessas atividades, notou-se a dificuldade do Integrante A em separar o que era decisões de gerência das de cliente.

Em comparação aos projetos anteriores desenvolvidos na Empresa T, modelo de casos de uso foi considerado, por todos os integrantes da equipe, como fator fundamental no sucesso do estudo de caso, pois a descrição dos objetivos do sistema com casos de uso proporcionou reutilizar essas descrições na especificação dos requisitos, e a descrição de todos os requisitos funcionais com casos de uso melhorou o entendimento dos requisitos a serem atendidos e minimizou problemas de ambigüidades nas descrições. A prática de utilizar casos de uso, tanto para descrever os objetivos do sistema como para especificar os requisitos, já foi adotada no processo utilizado pela Empresa T.

Por meio do padrão *Desenvolvimento espiral*, apresentado no Quadro 3.5, as atividades de elicitação e especificação de requisitos foram realizadas iterativamente minimizando re-trabalho na especificação de requisitos, pois primeiramente foram descritos os requisitos gerais do sistema e a medida que foram sendo conhecidos os requisitos especializados, eles foram descritos em detalhes. Essa prática já foi adotada pela Empresa T. O mesmo ocorre com a prática de agrupar requisitos de acordo com as características do sistema, que também foi classificada pelos integrantes da equipe como vantagem do processo proposto permitindo que diferentes pessoas envolvidas no projeto verificassem todas as funções do sistema em diferentes níveis de abstração.

A técnica Pontos de Casos de Uso já está sendo utilizada pela Empresa T para estimar os tempos de desenvolvimento dos requisitos. Essa técnica foi classificada como fundamental pelos integrantes da equipe por proporcionar antecipação nas estimativas, em relação aos projetos anteriormente desenvolvidos pela empresa.

A exposição dos requisitos a serem desenvolvidos a todos os integrantes da equipe no início do desenvolvimento também foi adotada como prática pela Empresa T.

A modelagem do processo proposto com o SPEM foi classificada, pelos integrantes da equipe, como eficaz e proporcionou entendimento das seqüências em que as atividades devem ocorrer e serem realizadas por papéis. Quanto à facilidade em alterar o processo proposto, todos os integrantes da equipe classificaram o processo como bem definido e fácil de ser adaptado para atender às necessidades particulares de um projeto ou organização que desenvolve software.

Com a aplicação do questionário aos integrantes da equipe pôde-se verificar que houve discrepância de opiniões, pois cada integrante respondeu de acordo com o papel que desempenhou no processo proposto. Nas perguntas de âmbito geral, em que os integrantes responderam de forma conjunta há uniformidade nas opiniões, como por exemplo, perguntas 1b, 2b e 2c. Para as perguntas 2e, 2f, 3a, 3b apenas um Desenvolvedor (integrante B) divergiu dos demais. (Verificar, pois há algo estranho nesta frase, pois houve discrepância e vc comenta que não houve. O que corrigi está correto? – Daniel – Colocar as outras perguntas, evidenciando q nelas as opiniões foram diferentes, ou seja, houve discrepância em algumas, a de âmbito particular de cada papel. Isso deve ser colocado no início da frase.)

Capítulo 6 – Conclusão

6.1 - Considerações Iniciais

Devido à importância de melhorar constantemente o processo de desenvolvimento de software para que produtos de qualidade sejam entregues em conformidade com as reais necessidades dos clientes, este trabalho utilizou as práticas de sucesso comprovado dos padrões de requisitos e organizacionais para auxiliar a realização das atividades de engenharia de requisitos nos métodos ágeis.

Para estruturar e organizar essas atividades, um processo de engenharia ágil de requisitos modelado pelo meta-modelo SPEM foi elaborado e proposto. Esse processo está fundamentado em alguns princípios ágeis, tais como iteratividade e participação ativa do cliente na obtenção e gerenciamento de requisitos, e no modelo de ciclo de vida proposto por Ambler (2006).

Por meio do estereótipo guia do SPEM, alguns padrões de requisitos e organizacionais foram modificados e integrados ao processo proposto para auxiliar na organização e conduzir a realização de suas atividades.

O processo proposto foi aplicado em um estudo de caso realizado em uma empresa real de desenvolvimento de sistemas de software para analisar as suas vantagens e desvantagens com relação ao processo já utilizado pela empresa. Por meio da aplicação de um questionário à equipe que participou do estudo de caso e de observações feitas pelo autor desta dissertação, pôde-se concluir que o processo proposto trouxe contribuições significativas ao processo de engenharia de software utilizado atualmente pela empresa. Entretanto, foram observadas também algumas restrições do processo proposto em relação ao estudo de caso.

Além desta Seção, este Capítulo está dividido da seguinte forma: a Seção 6.2 apresenta as contribuições e restrições desta pesquisa e a Seção 6.3 apresenta as perspectivas de pesquisas futuras.

6.2 - Contribuições e restrições da pesquisa

Com a realização do estudo de caso e a aplicação do questionário à equipe, verificou-se que as práticas utilizadas no processo proposto proporcionaram contribuições efetivas ao

processo de desenvolvimento de software da Empresa T, porém restrições também foram observadas. Algumas dessas práticas já foram incorporadas ao processo de desenvolvimento de software da empresa e outras estão sendo analisadas para também serem adotadas. Entretanto, as já adotadas não implicam em modificações no modelo de desenvolvimento de software utilizado atualmente pela empresa, que apenas introduziu o uso dessas práticas e técnicas em seu processo, substituindo algumas já utilizadas. As contribuições, restrições e adoção das práticas do processo proposto são resumidamente apresentadas a seguir e já detalhadas na Seção 5.4.

- **Desenvolvimento do software em versões funcionais**.
  - **Contribuições**:
    a. Integrantes da equipe motivados durante todo o desenvolvimento do sistema, pois podiam visualizar, em no máximo quinze dias, o resultado do seu trabalho;
    b. Desenvolvimento das versões do sistema de acordo com as reais necessidades do Cliente, pois ele forneceu rápida apreciação com relação ao que estava sendo produzido.
  - Essa prática ainda não foi adotada pela empresa por exigir mudança para um modelo de desenvolvimento de software incremental com entregas rápidas de software funcional, o que ainda está sendo analisado.

- **Possibilidade de modificações em requisitos durante o desenvolvimento**
  - **Contribuição**: Proporcionou rápida tomada de decisão para modificar requisitos e facilidade em alterar os requisitos do sistema sem procedimentos burocráticos como são realizados nos outros projetos da Empresa T.
  - Essa pratica ainda não foi adotada pela empresa.

- **Participação ativa do cliente durante a especificação dos requisitos**.
  - **Contribuições**:
    a. Atendimento das reais necessidades do Cliente, pois ele participou da elicitação e da especificação dos requisitos;
    b. Melhor produtividade dos integrantes da equipe em relação aos outros projetos realizados na Empresa T.

- **Restrições**:
  a. Dificuldade do Cliente em indicar uma pessoa para desempenhar o papel de Colaborador, pois ele não podia desempenhar esse papel e não tinha outra pessoa para representá-lo;
  b. A melhora de produtividade da equipe está diretamente condicionada à participação ativa do Gerente de Projetos no desenvolvimento do sistema, pois ele desempenhou também o papel de Colaborador. Nos projetos anteriormente desenvolvidos na empresa, o Gerente de Projetos apenas realizava atividades de planejamento e ficava mais distante dos outros integrantes da equipe.
- Essa prática já foi adotada pela empresa, mas não é obrigatória. Há também a restrição de que não pode haver acúmulos de papéis para desempenhar o papel de Colaborador.

- **Utilização de modelo de casos de uso durante a definição dos objetivos do sistema e durante a especificação dos requisitos.**
  - **Contribuições**:
    a. Melhor compreensão, por parte de todos os envolvidos no projeto, dos objetivos e requisitos do sistema;
    b. Reutilização dos modelos de casos de uso na especificação dos requisitos.
  - **Restrição**: Apesar do modelo de casos de uso melhorar a compreensão dos objetivos do sistema, o Gerente de Projetos optou por continuar também com a representação textual descritiva para os objetivos de sistema, pois ele alegou que nem sempre o Cliente entende todos os objetivos representados com modelos de casos de uso.
  - Na especificação dos requisitos, o uso de casos de uso tornou-se obrigatório.

- **Realização gradativa da especificação dos requisitos, conforme são conhecidos os detalhes do sistema.**
  - **Contribuição**: Diminuição do re-trabalho do Engenheiro de Requisitos durante a especificação dos requisitos, pois primeiramente foram especificados os requisitos gerais do sistema e os detalhes foram especificados á medida que se conheciam melhor as necessidades do Cliente.
  - Essa prática já foi adotada pela empresa como guia para realização da especificação dos requisitos.

- **Agrupamento de requisitos de acordo com características do sistema.**
  - **Contribuição**: Melhor visibilidade dos requisitos do sistema, pois todos os envolvidos com o projeto visualizaram todas as funções do sistema agrupadas e em diferentes níveis de abstração.
  - Essa prática já foi adotada pela empresa, tornando-se obrigatória.

- **Priorização dos requisitos pelo Colaborador.**
  - **Contribuição**: Possibilitou que as versões do sistema fossem entregues de acordo com as preocupações e necessidades do Cliente.
  - Essa prática ainda não foi adotada pela empresa, pois o modelo de processo atualmente utilizado não prevê entregas de software funcional em versões.

- **Utilização de pontos de casos de uso para realizar as estimativas.**
  - **Contribuição**: Proporcionou a antecipação da realização das estimativas, pois em projetos anteriores as estimativas eram realizadas apenas na etapa de Projeto de Software.
  - Essa prática já foi adotada pela empresa, tornando-se obrigatória.

- **Exposição dos requisitos a todos os integrantes da equipe.**
  - **Contribuição**: Proporcionou uniformidade quanto ao entendimento do que deve ser realizado pelos integrantes da equipe, melhorando a conformidade do sistema em desenvolvimento com as necessidades do Cliente.
  - Essa prática já foi adotada pela empresa, tornando-se obrigatória.

A Tabela 6.1 apresenta uma comparação das práticas e técnicas adotadas antes e depois da realização do estudo de caso. A primeira coluna apresenta as características do processo e a segunda e a terceira colunas como elas foram tratadas.

Tabela 6.1 – Comparação das abordagens utilizadas na Empresa T antes e depois da realização do estudo de caso

Características	Antes	Depois
Objetivos do Sistema	- Descritos apenas textualmente.	- Descritos textualmente e com modelo de casos de uso.
Especificação dos Requisitos	- Descritos textualmente e com casos de uso, quando necessário; - Sem a participação do cliente; - Especificados de acordo com a experiência do engenheiro de requisitos.	- Descritos com modelo de casos de uso; - Com a participação ativa do cliente; - Primeiramente especificam-se os requisitos gerais e à medida que se conhece o sistema, os detalhes são descritos.
Visibilidade dos requisitos	- Dificuldade de visualizar quais requisitos pertencem a determinadas características do sistema.	- Fácil visualização dos requisitos pertencentes a determinadas características do sistema.
Estimativas	- Realizadas com o uso da técnica Análise de Pontos por Função durante a etapa de projeto de software.	- Realizadas com o uso da técnica Pontos de Casos de Uso durante a engenharia de requisitos.
Apresentação dos requisitos a todos os integrantes da equipe	- Todos os integrantes da equipe tinham acesso ao documento de especificação de requisitos e o engenheiro de requisitos apenas esclarecia as dúvidas dos integrantes, quando necessário.	- Há apresentação dos requisitos do sistema a todos os integrantes da equipe, antes do desenvolvimento do projeto de software.

Com relação à facilidade de compreensão e manutenção no processo proposto os integrantes da equipe classificaram o processo como fácil de alterar para atender às necessidades particulares de um projeto ou organização e a modelagem do processo, em SPEM, foi classificada como eficiente e proporcionou o entendimento da ordem de execução dos elementos do processo.

As contribuições do processo proposto, observadas no estudo de caso, também podem ser influenciadas pelo ambiente no qual ele foi realizado. De acordo com pesquisa apresentada em Ambler (2007a), dentre as organizações que fazem uso da entrega rápida de software funcional, 83% têm seus ciclos de desenvolvimento com duração entre uma e quatro semanas e 32% dessas organizações utilizam duas semanas. O maior índice de sucesso no desenvolvimento de sistemas de software em relação ao tamanho médio das equipes é para aquelas compostas por um a cinco integrantes. Assim, o ambiente no qual foi realizado o estudo de caso favoreceu aos resultados satisfatórios obtidos, pois como apresentado no Capítulo 5, considerando uma equipe com cinco integrantes e que os ciclos de desenvolvimento foram de no máximo duas semanas.

A principal restrição quanto à aplicação do processo proposto no estudo de caso realizado na Empresa T está relacionada ao acúmulo de papéis desempenhados pelo integrante A: Gerente de Projetos e Colaborador, pois esse integrante é um profissional experiente em desenvolvimento de software e no uso da técnica modelos de casos de uso.

Durante a realização do estudo de caso o Gerente de Projetos reportou que ocorreu uma diminuição de 13% no tempo gasto para a realização das atividades de elicitação e especificação de requisitos, em comparação com a média dos projetos anteriores desenvolvidos pela Empresa T. Entretanto, a diminuição de tempo para a realização dessas atividades está relacionada ao aumento do número de pessoas, com conhecimentos na técnica modelos de casos de uso, que estavam realizando as atividades de elicitação e especificação de requisitos. Nos outros projetos desenvolvidos pela empresa essas atividades eram realizadas por apenas uma pessoa.

Dessa forma, pode-se concluir que apesar do processo proposto ser aplicado em uma empresa real de desenvolvimento de sistemas de software, o cliente não foi real.

Além da elaboração e modelagem de um processo ágil de engenharia de requisitos, este trabalho também tem como principais contribuições acadêmicas a identificação de alguns padrões de requisitos e organizacionais existentes, os quais estão relacionados com os princípios preconizados pelos métodos ágeis e a identificação de práticas e técnicas utilizadas na realização da engenharia de requisitos.

Por meio do estudo de caso realizado pode-se comprovar a eficiência desses padrões quando considerados como guias de processo, para apoiar a realização de atividades e tarefas em um processo ágil de desenvolvimento de software. Para verificar se os padrões de requisitos e organizacionais serão úteis em atividades de outras metodologias de processos de desenvolvimento de software ou em outros contextos, diferente do aplicado nesse trabalho, esses padrões devem ser relacionados às guias de outros processos de desenvolvimento e novos estudos de caso devem ser conduzidos. Neste trabalho esses padrões foram aplicados em um contexto em que a equipe do projeto era pequena, o cliente não possuía total conhecimento dos requisitos do sistema no início do desenvolvimento e o processo proporcionou facilidade de incorporar e adaptar mudanças de requisitos durante todo o desenvolvimento havendo constante interação e cooperação entre os envolvidos no projeto. Além disso, considerou-se que a equipe tinha alta experiência na aplicação proposta, a equipe estava altamente motivada com o projeto e os requisitos não modificava muito durante o projeto, sendo considerados com estabilidade média.

6.3 - Perspectivas de pesquisas futuras

A seguir, são apresentadas algumas sugestões para trabalhos futuros:

- Verificar a existência de outros padrões de requisitos e organizacionais que possam ser integrados ao processo proposto para melhorá-lo quanto à realização da elicitação de requisitos.

- Verificar a existência de padrões de processo que podem ser utilizados para auxiliar nas diretrizes do processo proposto, sem afetar sua flexibilidade.

- Realizar outros estudos de caso para convalidar os resultados encontrados.

Referências Bibliográficas

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. Agile software development methods. **VTT Publications,** n. 478, 2002.

ACUÑA, S.; FERRÉ, X. Software process modeling. In: **Proceedings of The 5th World Multiconference on Systemics, Cybernetics and Informatics**, 2001. p. 1-6.

ALEXANDER, C. **The timeless way of building**. New York: Oxford University Press, 1979.

ALEXANDER, C.; ISHIKAWA, S.; MURRAY, S. **A pattern language:** towns, buildings, construction. New York: Oxford University Press, 1977.

AMBLER, S. W. **Agile adoptions Survey.** AmbySoft, 2007a. Disponível em: <http://www.ambysoft.com/surveys/agileMarch2007.html>. Acesso em: 13 dez. 2007.

AMBLER, S. W. **Agile requirements best practices.** AmbySoft, 2007b. Disponível em: <http://www.agilemodeling.com/essays/agileRequirementsBestPractices.htm>. Acesso em: 13 dez. 2007.

AMBLER, S. W. **Agile requirements change management.** AmbySoft, 2007c. Disponível em: <http://www.agilemodeling.com/essays/changeManagement.htm>. Acesso em: 13 dez. 2007.

AMBLER, S. W. **The agile system development lifecycle.** AmbySoft, 2006. Disponível em: <http://www.ambysoft.com/essays/agileLifecycle.html>. Acesso em: 13 dez. 2007.

AMBLER, S. W. **An introduction to process patterns**. AmbySoft, 1998. Disponível em: <http://www.ambysoft.com/downloads/processPatterns.pdf>. Acesso em: 21 jun. 2006.

ANDRADE, E.L.P.; OLIVEIRA, K. M. Uso combinado de análise de pontos de função e casos de uso na gestão de estimativa de tamanho de projetos de software orientado a objetos. In: **III Simpósio Brasileiro de Qualidade de Software**. Brasília, 2004.

ANDRADE, R. M. C. **Capture, reuse, and validation of requirements and analysis patterns for mobile systems.** 2001. Tese (Doutorado) - School of Information Technology and Engineering, University of Ottawa, Canadá, 2001.

ANDRADE, T. C.; FREITAS, F.G.; SOUZA, J.T. Modelo de melhoria do processo de software para micro e pequenas empresas baseado em padrões. In: **Proceedings of the SugarLoafPlop**, 2007. p. 162-177. Porto de Galinhas, PE, Brasil. Disponível em: < http://sugarloafplop.dsc.upe.br>. Acesso em: 13 dez. 2007.

APPLETON, B. **Patterns and software**: **essential concepts and terminology**. 2000. Disponível em: <http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html>. Acesso em: 21 jun. 2006.

BECK, K. **Extreme programming explained:** embrace change. 2. ed. Addison-Wesley, 2000.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. **Manifesto for agile software development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 21 jun. 2006.

BOEHM, B. W.; ERDOGMUS, H.; HARRISON, W.; REIFER, D.J.; SULLIVAN, K.J. Software engineering economics: background, current practices, and future directions**.** In: **International conference on software engineering**, 24, 2002. p. 683-684, Florida.

BRAGA, R. T. V.; GERMANO, F. E. R.; MASIERO, P. C.; MALDONADO, J. C. Introdução aos padrões de software. Disponível em: <http://sugarloafplop2005.icmc.usp.br/NotasDidaticas_Padroes.pdf>. Acesso em: 21 jun. 2006.

BRAMBLE, P.; COCKBURN, A.; POLS, A.; ADOLPH, S. **Patterns for effective use cases**. Reading, MA: Addison-Wesley, 2002.

BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAND, P.; STAL, M. **Pattern-oriented software architecture**: a system of patterns. Wiley & Sons, 1996.

CARVALHO, A. M. B. R.; CHIOSSI, T. C. S. **Introdução à engenharia de software.** Campinas: Unicamp, 2001.

COAD, P.; LEFEBVRE, E.; DELUCA, J. **Java modeling in color with UML**. Prentice-Hall, 1999.

COCKBURN, A. **Agile software development**. Addison-Wesley, 2002.

COCKBURN, A. **Writing effective use cases.** Addison-Wesley, 2001.

COCKBURN, A. **Surviving object-oriented projects:** a manager's guide. Addison-Wesley, 1998.

COCKBURN, A.; HIGHSMITH, J. Agile software development: the people factor. **IEEE Computer**, v. 34, n. 11, p. 131-133, 2001.

CONRADI, R.; FERNSTRÖM, C.; FUGGETTA, A. **Concepts for evolving software process, in software process modeling and technology**. Research Studies Press, 1994.

COPLIEN, J. O. **A pattern definition**. 1998. Disponível em: <http://hillside.net/patterns/definition.html>. Acesso em: 21 jun. 2006.

COPLIEN, J. O. A generative development-process pattern language. In: COPLIEN, J.; SCHMIDT, D. **Pattern Languages of Program Design**. USA: Addison-Wesley, 1995. p. 183-237.

COPLIEN, J. O. **Advanced C++ programming styles and idioms**. Addison-Wesley, 1991.

COPLIEN, J. O.; HARRISON N. B. **Organizational Patterns of Agile Software Development**. 1. ed. Prentice Hall, 2004.

CORAM, T. Demo Prep: a pattern language for the preparation of software demonstrations. In: VLISSIDES, J.; COPLIEN, J.; KERTH, N. **Pattern Languages of Program Design 2**. Addison-Wesley, 1996. p. 407-416.

COSTA, E. G.; PENTEADO, R. A. D.; SILVA, J. C. A.; BRAGA, R., T. V. **Padrões e métodos ágeis**: agilidade no processo de desenvolvimento de software. In: **Proceedings of the SugarLoafPlop**, 5, 2005. p. 156-169. Campos do Jordão, SP, Brasil.

COSTA, E. G. **Integração de padrões organizacionais e de processo ao método ágil scrum**. Dissertação de Mestrado. Universidade Federal de São Carlos, São Carlos, 2006.

CREEL, C. **Requirements by patterns**, 2006. Disponível em: <http://www.ddj.com/architect/196600223>. Acesso em: 12 dez. 2007.

CUNNINGHAM, W.; BECK, K. **Using pattern languages for object-oriented programs**. 1987. Disponível em: <http://c2.com/doc/oopsla87.html>. Acesso em: 21 jun. 2006.

CURTIS, B.; KELLNER, M.; OVER, J. **Process Modeling**. Communications of the ACM, p.75-90, v.35, n.9, 1992.

DAHLSTEDT, A. **Requirements engineering**. University of Skövde, 2003. Disponível em: <http://www.ida.his.se/ida/kurser/informationsystems_engineering/kursmaterial/forelasningar/Chapter11_2003.pdf>. Acesso em: 21 jun. 2006.

DAMIAN, D. E. H. **Challenges in requirements engineering**. Department of Computer Science, University of Calgary, Relatório Técnico n. T2N 1N4.1999-645-08, 2000.

DAMODARAN, M; WASHINGTON, A. **Estimation using use case points**. Computer Science Program. Texas, Victoria: University of Houston. S.d. Disponível em:

<http://bfpug.com.br/Artigos/UCP/Damodaran-Estimation_Using_Use_Case_Points.pdf.> Acesso em: 12 dez. 2007.

DAVIES, R. Agile requirements. **Methods & Tools**, v.13, n. 3, 2005.

DAVIS, A. M.; HICKEY, A. M. Requirements elicitation and elicitation technique selection: a model for two knowledge-intensive software development processes. In: **International conference on system sciences**, 36., 2002, Hawaii.

DELANO, D.; RISING, L. A pattern language for system test. In: **Pattern languages of program design** 3., 1997, Addison-Wesley.

EBERLEIN, A.; LEITE, J. C. S. P. Agile requirements definition: a view from requirements engineering. In: **Time Constrained Requirements Engineering**, 2002, Essen, Alemanha. 2002. Disponível em: <http://www2.enel.ucalgary.ca/tcre02/> Acesso em: 12 dez. 2007.

FAVARO, J. Managing requirements for business value. **IEEE Software**, v. 19, p. 15-17, 2002.

FENTON, N.; NEIL, M. **Software Metrics:** roadmap. Future of software engineering. Limerick Ireland ACM. p. 359-370, 2000.

FERDINANDI, P. L. A **Requirements Pattern: Succeeding in the Internet Economy**. Addison-Wesley, 2002.

FOWLER, M. **The new methodology.** 2005. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>. Acesso em: 21 jun. 2006.

FOWLER, M.; KENDALL, S. **UML distilled:** a brief guide to the standard object modeling language. 2. ed. Addison-Wesley, 2000.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. M. **Design patterns:** elements of reusable object-oriented software. Addison-Wesley, 1994.

GARMUS, D., HERRON, D. **Function point analysis:** measurement practices for successful software projects. Addison-Wesley: EUA. 2000. 363 p.

GOGUEN, J.A.; LINDE, C. Techniques for requirements elicitation. In**: Proceedings of IEEE International Symposium on requirements engineering**, 1993. p. 152-164.

HAGGE, L.; LAPPE, K. Sharing requirements engineering experience using patterns. **IEEE Software,** v. 22, n. 1, p. 24-31, 2005.

HANMER, R. Introduction to pattern languages. In: **Sugarloafplop**, 2003, Porto de Galinhas, PE, Brasil. Disponível em: <http://www.cin.ufpe.br/~sugarloafplop/patLang.ppt>. Acesso em: 13 dez. 2007.

HARRISON, N. B. Organizational Patterns for Teams. In: VLISSIDES, J.; COPLIEN, J.; KERTH, N. **Pattern Languages of Program Design 2**. Addison-Wesley, 1996, p. 345-352.

HIGHSMITH, J. **Agile software development ecosystems**. Boston: Addison- Wesley, 2002.

HIGHSMITH, J.; COCKBURN, A. Agile software development: the business of innovation. **IEEE Computer.** v. 34, n. 9, p. 120-122, 2001.

HUNT, A.; THOMAS, D. **The pragmatic programmer**. Addison-Wesley, 1999.

IEEE. Institute of Electrical Electronics Engineers. **IEEE Standard Glossary of Software Engineering Terminology**. IEEE Std 610.12-1990, New York, 1990.

IEEE. Institute of Electrical Electronics Engineers. **IEEE Recommended Practice for Software Requirements Specifications**. IEEE Std 830-1998, New York, 1998a.

IEEE. Institute of Electrical Electronics Engineers. **IEEE Guide for Developing System Requirements Specifications**. IEEE Std 1233-1998, New York, 1998b.

IFPUG, International Function Point Users' Group. **About Function Point Analysis,** 2006 <http://www.ifpug.org/about/about.htm>. Acesso em: 12 Dez. 2007.

IFPUG. International Function Point Users' Group**. Function Point Counting Practices Manual.** Release 4.1. Ohio, 2000. Disponível em: <http://www.ifpug.org/publications/manual.htm>. Acesso em: 12 Dez. 2007.

INTHURN, C. **Qualidade & teste de software**. Visual Books, 2001.

KARNER, G. **Use case points**: resource estimation for Objectory projects. Objective Systems SF AB (copyright owned by Rational/IBM), 1993.

KOTONYA, G.; SOMMERVILLE, I. **Requirements engineering:** processes and techniques. Chichester: John Wiley & Sons, 1998.

KOTULA, J. Using patters to create component documentation. **IEEE Software**. v. 15, n. 2, p. 84-92, 1998.

KRUCHTEN, P. Agility with the RUP. **Cutter IT Journal**, Arlington, v.14, n.12, p. 27- 33, 2001.

KUSUMOTO, S; INOUE, K.; KASIMOTO, T. Function Point Measurement for Object-Oriented Requirements Specification. In: **Annual international computer software and aplications conference**, 24., 2000. **IEEE,** 6p.

LAMSWEERDE, A. V. Requirements engineering in the year 00**:** a research perspective. In: **International conference on software engineering**, 22., 2000, Ireland.

LAPPE, K.; CZIHARZ, T.; DREIER, H.; FAHNEY, R.; GUMM, D.; HAGGE1, L.; HÖHNE, G.; HOUDEK, F.; ITTNER, J.; JANZEN, D.; PAECH, B. **Requirements engineering patterns**: an approach to capturing and exchanging requirements engineering experience, 04–223, Alemanha: DESY, 2004.

LARMAN, C. **Utilizando uml e padrões**: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed. Bookman, 2007.

LEA, D. - **Christopher alexander:** an introduction for object-oriented designers. 1997. Disponível em: <http://gee.cs.oswego.edu/dl/ca/ca/ca.html>. Acesso em: 12 dez. 2007.

LEFFINGWELL, D.; WIDRIG, D. **Managing software requirements** - a use case approach. 2. ed., 2003.

LONGSTREET, D. **Fundamentals of function points analysis**. Blue Springs: Longstreet Consulting Inc., 2002. Disponível em: <http://www.math.unipd.it/~tullio/IS-1/2004/Approfondimenti/Function_Point_Analysis.pdf>. Acesso em: 12 dez. 2007.

MCGARRY, J.; CARD, D.; JONES, C.; LAYMAN, B.; CLARK, E.; DEAN, J.; HALL, F. **Practical software measurement**: objective information for decision makers. Boston: Addison-Wesley. 2001. 277p.

MICROSOFT. **Microsoft visio**. 2007a. Disponível em <http://office.microsoft.com/pt-br/visio/default.aspx>. Acesso em: 12 Dez. 2007.

MICROSOFT. **Microsoft word**. 2007b. Disponível em <http://office.microsoft.com/pt-br/word/default.aspx>. Acesso em: 12 dez. 2007.

MILLER, G. G. The characteristics of agile software process**.** In: **International conference of object-oriented languages and systems**, 39., 2001, Santa Bárbara.

NUSEIBEH, B; EASTERBROOK, S. Requirements engineering: a roadmap. In: **International conference on software engineering**, 22., 2000, Ireland.

OMG - Object Management Group. **Software Process Engineering Metamodel Specification,** 2005.

OMG - Object Management Group. **UML Specification, 1.4.,** 2001.

ORR, K. **Agile requirements**: Opportunity or oxymoron? **IEEE Software**, vol. 21, 2004.

PRESSMAN, R. S. **Engenharia de software**. 6. ed. São Paulo: McGraw Hill, 2006.

RISING, L. **The pattern almanac**. Addison-Wesley, 2000.

ROBERTSON, S.; ROBERTSON, J. **Mastering the requirements process**. 2. ed. Addison-Wesley, 2006.

RUP - **Rational Unified Process**. 2001. Disponível em: <http://www.wthreex.com/rup/>. Acesso em: 12 dez. 2007.

SAMPAIO, A. **Xwebprocess**: um processo ágil para o desenvolvimento de aplicações web.2004. Dissertação de Mestrado. Universidade Federal de Pernambuco, Pernambuco, 2004.

SAWYER, P.; KOTONYA, G. Software requirements. In: **Guide to the software engineering body of knowledge, swebok**, 2001. Disponível em: <http://www.swebok.org/stoneman/ trial_1_00.htm>. Acesso em: 21 jun. 2006.

SCHWABER, K.; BEEDLE, M. **Agile software development with scrum**. Prentice-Hall, 2002.

SHOEMAKER, M, L. **Requirements patterns and antipatterns:** best (and worst) practices for defining your requirements paperback. Addison-Wesley Professional, 2007. 352 p.

SIDDIQI, J.; SHEKARAN, M. Requirements engineering: The Emerging Wisdom. **IEEE Software**, vol. 13, 1996.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. São Paulo: Addison Wesley, 2003.

SOMMERVILLE, I.; SAWYER, P. **Requirements engineering**: a good practice guide. New York: John Wiley & Sons, 1997.

STAPLETON, J. **Dsdm dynamic systems development method:** the method in practice. Addison-Wesley, 1997.

THAYER R.H.; DORFMAN, M. **Software requirements engineering**. 2. ed. IEEE Computer Society Press, 1997. Disponível em: <http://www.sei.cmu.edu/news-at-sei/features/1999/mar/Background.mar99.pdf.>. Acesso em: 21 jun. 2006.

TOMAYKO, J. E. Engineering of unstable requirements using agile methods. In: **Time Constrained Requirements Engineering**, 2002, Essen, Alemanha. 2002. Disponível em: < http://www-di.inf.puc-rio.br/~julio/tcre-site/p1.pdf>. Acesso em: 13 dez. 2007.

TURK, D.; FRANCE, R.; RUMPE, B. Limitations of agile software processes. In: **International conference on extreme programming and agile processes in software engineering**, 3., 2002, Italy.

TYRRELL, S. **The many dimensions of the software process.** ACM Crossroads.

VLISSIDES, J. **Patterns:** the top ten misconceptions. 1998. Disponível em: <http://www.research.ibm.com/designpatterns/pubs/top10misc.pdf>. Acesso em: 21 jun. 2006.

YOUNG, R.R. Recommended requirements gathering practices. Crosstalk The Journal of Defense Software Engineering, 2002.

Apêndice A – Questionário

Questionário aplicado ao final do estudo de caso para verificar o processo de Engenharia de Requisitos proposto.

1) **Com relação ao processo de engenharia de requisitos proposto:**
    a. Quais as principais vantagens e desvantagens do processo proposto com relação ao processo já utilizado pela Empresa T? Justifique sua resposta.
    b. Quais conjuntos de trabalho/atividades foram utilizados e quais não foram utilizados? Justifique sua resposta.

2) **Com relação aos padrões integrados ao processo:**
    a. Você acredita que os padrões inseridos auxiliaram o processo? Justifique sua resposta.
    b. Quais padrões não foram utilizados? Justifique sua resposta.
    c. Quais as principais vantagens e desvantagens que os padrões integrados ao processo e utilizados no desenvolvimento do sistema, proporcionaram para o projeto? Justifique sua resposta.
    d. Quais padrões foram mais e menos relevantes para o desenvolvimento do sistema? Justifique sua resposta.
    e. Como você classifica as descrições das soluções dos padrões, em relação ao problema/solução que o padrão se propõe a resolver?
        i. ( ) Mal Relacionado: As descrições não condizem com o problema e solução dos padrões.
        ii. ( ) Razoavelmente Relacionado: As descrições condizem razoavelmente com o problema e solução dos padrões.
        iii. ( ) Bem Relacionado: As descrições condizem exatamente com o problema e solução dos padrões.
    f. Com que facilidade pode-se compreender a descrição dos passos que representam a solução dos padrões utilizados?

i. ( ) Mal Definida: As descrições estão confusas e incompletas. Não se pôde compreender “o que” deveria ser feito.
ii. ( ) Razoavelmente Definida: As descrições estão razoavelmente confusas ou incompletas. Não se pôde compreender com precisão “o que” deveria ser feito.
iii. ( ) Bem Definido: As descrições estão bem definidas e completas. Foi possível compreender com precisão “o que” deveria ser feito.

3) Com relação à modelagem do processo proposto com o SPEM:

a. Facilidade de Compreensão: Com que facilidade se pode compreender o processo por meio da modelagem realizada com SPEM?
   i. ( ) Mal Definido: As atividades do processo estão mal definidas. Não se tem idéia da seqüência em que as atividades devem ocorrer.
   ii. ( ) Razoavelmente Definido: As atividades do processo estão razoavelmente definidas e explicadas. Não se tem idéia detalhada da seqüência em que as atividades devem ocorrer.
   iii. ( ) Bem definido: As atividades do processo estão bem definidas e explicadas. Tem-se idéia clara da seqüência em que as atividades devem ocorrer.
b. Facilidade de Manutenção: Seria fácil realizar alterações no processo para integrar novas atividades, artefatos, papéis e guias?
   i. ( ) Difícil de Alterar: Devido a sua definição inadequada seria difícil alterá-lo para atender às necessidades particulares de um projeto ou organização.
   ii. ( ) Fácil de Alterar: Devido ao processo estar bem definido seria fácil alterá-lo para atender às necessidades particulares de um projeto ou organização.

Apêndice B - Formulário de Consentimento

"Engenharia de requisitos utilizando padrões"

Eu declaro ter mais de 18 anos de idade e concordar em participar de um estudo de caso conduzido por Daniel Eduardo Funabashi de Toledo, como parte das atividades para obtenção do titulo de mestre, no programa de Pós Graduação da Universidade Federal de São Carlos. Esse estudo visa compreender a aplicabilidade da abordagem: "Um Processo Ágil de Engenharia de Requisitos com Apoio de Padrões de Software" na tarefa de identificar melhorias para a Engenharia de Requisitos nos métodos ágeis.

Procedimento

Este estudo acontecerá em duas etapas. Primeiro é apresentado o aparato necessário para o desenvolvimento da Engenharia de Requisitos, Metodologia Ágil e Padrões de Software, bem como instruções para a realização das tarefas e documentos que serão utilizados neste estudo. Na segunda etapa os participantes deverão utilizar o aparato apresentado previamente para a validação do uso das práticas e técnicas apresentadas. Eu entendo que, uma vez o experimento tenha terminado, os trabalhos que desenvolvi, serão estudados visando a entender a eficiência dos procedimentos das práticas e das técnicas que me foram ensinadas.

Confidencialidade

Toda informação coletada neste estudo é confidencial, e meu nome e da empresa na qual trabalho, não serão identificados em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das práticas, técnicas e documentos apresentados e que fazem parte do experimento.

Benefícios, liberdade de desistência

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a utilização da abordagem proposta "Um Processo Ágil de Engenharia de Requisitos com Apoio de Padrões de Software" para o apoio na engenharia de requisitos.

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Estou ciente que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de práticas, técnicas e processos para a engenharia de requisitos de software.

Equipe:

Pesquisador Responsável: Daniel Eduardo Funabashi de Toledo

Professora Responsável: Prof. Dr Rosângela Aparecida Dellosso Penteado

Programa de Pós-graduação em Ciência da Computação. Universidade Federal de São Carlos.

Nome (em letra de forma): _____

Assinatura: _____ data: _____