UNIVERSIDADE ANHEMBI MORUMBI
CÉSAR SANZ GUTIERREZ
EDUARDO YUKIO MIYAKE
FÁBIO HENRIQUE PEREIRA LIMA
NICK LAZUR

ENGENHARIA DE REQUISITOS NA METODOLOGIA ÁGIL

São Paulo
2009

CÉSAR SANZ GUTIERREZ
EDUARDO YUKIO MIYAKE
FÁBIO HENRIQUE PEREIRA LIMA
NICK LAZUR

ENGENHARIA DE REQUISITOS NA METODOLOGIA ÁGIL

Trabalho de Conclusão de Curso apresentado como exigência parcial para a obtenção de título de Graduação do Curso de Ciências da Computação, da Universidade Anhembi Morumbi

Orientador(a): Dra. Judith Pavón

São Paulo
2009

UNIVERSIDADE ANHEMBI MORUMBI

CÉSAR SANZ GUTIERREZ

EDUARDO YUKIO MIYAKE

FÁBIO HENRIQUE PEREIRA LIMA

NICK LAZUR

ENGENHARIA DE REQUISITOS NA METODOLOGIA ÁGIAL

Trabalho de Conclusão de Curso apresentado como exigência parcial para a obtenção de título de Graduação do Curso de Ciências da Computação, da Universidade Anhembi Morumbi

Aprovado em

_____

Prof. Alessandro Biagi Costa
Universidade Anhembi Morumbi

_____

Prof. Augusto Mendes Gomes Júnior
Universidade Anhembi Morumbi

_____

Prof. Judith Pavón De Mendoza
Universidade Anhembi Morumbi

AGRADECIMENTOS

Gostaríamos de agradecer em primeiro lugar aos nossos pais e familiares que sempre nos apoiaram e deram motivação durante o estudo de toda a carreira, permitindo que pudéssemos dedicar de forma especial e obter forças para ante qualquer adversidade, conseguir chegar até o final do curso de graduação.

Agradecemos também a todos os nossos mestres que, durante esse período de formação, tiveram a paciência e dedicação de ensinar de grande forma as metas e objetivos de cada curso ao longo de todos estes quatro anos.

Agradecimentos especiais aos nossos professores Judith Pavón e Marcelo Couto, que se dedicaram, nos incentivaram, estiveram sempre presentes em nosso trabalho e nos conduziram da melhor forma possível para a entrega deste trabalho de conclusão de curso com a melhor qualidade possível e dentro dos prazos exigidos.

“Planejar é decidir de antemão qual é, e como será a sua vitória.”

(Rhandy di Stefano)

RESUMO

A Engenharia de Software é composta por etapas essenciais para o processo de desenvolvimento de software, sendo a Engenharia de Requisitos uma das mais importantes disciplinas deste processo. Neste contexto, suas atividades são principalmente dedicadas a identificar, definir e gerenciar o escopo do produto a ser desenvolvido. As filosofias ágeis surgiram pela necessidade de diminuir os processos e artefatos dos métodos tradicionais de desenvolvimento de software, baseado em um conjunto de atividades predefinidas, descritas como processos prescritivos. Entretanto, os métodos tradicionais são considerados em geral muito burocráticos, mas eficientes para projetos que não sofrem muitas mudanças em seus requisitos. Este trabalho de conclusão de curso tem como objetivo abordar as etapas da Engenharia de Requisitos na Metodologia Ágil, definindo sua viabilidade e a melhor forma de aplicação dos conceitos de Engenharia de Requisitos na filosofia ágil. Para tanto, um estudo de caso será desenvolvido, de forma a demonstrar como é aplicada a Engenharia de Requisitos dentro do ciclo de vida de software proposto por uma metodologia ágil.

Palavras chave: Engenharia de Requisitos, Documento de Especificação de Requisitos, Metodologias Ágeis.

ABSTRACT

Software Engineering is composed of essential steps in the process of software development, requirements engineering is one of the most important disciplines in this process. In this context, its activities are mainly devoted to identify, define and manage the scope of the product to be developed. The agile philosophy arose by the need to reduce the processes and artifacts of traditional software development, based on a set of predefined activities, processes described as prescriptive. However, traditional methods are generally considered very bureaucratic, but effective for projects that do not suffer many changes in its requirements. This TCC aims to address the steps of Requirements Engineering in Agile Methodology, defining its feasibility and how best to apply the concepts of requirements engineering the agile philosophy. Thus, a case study will be developed in order to demonstrate how it is applied to engineering requirements within the life cycle of software proposed by an agile methodology.

Keywords: Requirements Engineering, User Requirements Document, Agile Methodologies.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASD *Adaptive Software Development* - Desenvolvimento Adaptável de Software.

CRF *Change Request Form*. Formulário de solicitação de mudança.

ER Engenharia de Requisitos.

ETel Empresa de telecomunicação

FDD *Feature Driven Development*. Desenvolvimento orientado a características.

JAD *Joint Application Design*.

MA Modelagem Ágil.

OOAD Análise, Desenho e Programação Orientada a Objetos.

RAD *Radical Software Development*.

ROI Retorno do investimento.

RUP *Rational Unified Process.*

TI Tecnologia da Informação.

UDI Universo de Informações.

UML *Unified Modeling Language.*

US *User Stories.*

VORD *Viewpoint-oriented requirements definition.* Definição de requisitos orientado a ponto de vista.

XP *Extreme Programming* - Programação Extrema.

SUMÁRIO

1 INTRODUÇÃO

Este trabalho de conclusão de curso apresenta um estudo da engenharia de requisitos (ER), fase inicial e de extrema importância da "Engenharia de Software". Posteriormente, são apresentadas as características da filosofia ágil e os métodos mais conhecidos no mercado: *Extreme Programming* (XP), SCRUM, *Feature Driven Development* (FDD) e *Adaptive Software Development* (ASD), fazendo uma análise comparativa. A partir desta análise comparativa, são escolhidas aquelas que melhores atendam às necessidades, sendo detalhada cada uma das etapas da "Engenharia de Requisitos" nos métodos escolhidos. Ao final, por meio de um estudo de caso, é demonstrada a viabilidade da aplicação da ER sob a perspectiva da filosofia ágil.

1.1 OBJETIVO

O objetivo deste trabalho é abordar as etapas da ER na metodologia ágil, definindo sua viabilidade e a melhor forma de aplicação dos conceitos de ER na abordagem ágil. Para tanto, um estudo de caso foi desenvolvido, de forma a demonstrar como é aplicada a ER dentro do ciclo de vida de software proposto por uma metodologia ágil.

1.2 JUSTIFICATIVA

Os processos tradicionais de engenharia de requisitos exigem um considerável esforço na definição do escopo do produto a ser entregue durante o ciclo de vida do software. Havendo mudanças no escopo do produto, estas precisam ser avaliadas e negociadas sob as perspectivas de esforço, prazo, custo e qualidade. Este cenário traz consigo um dos principais problemas enfrentados pelas equipes de desenvolvimento de sistemas e gestão das mudanças. Os custos destas mudanças aumentam em razão de quão cedo elas são efetuadas dentro do ciclo de vida de software (LEFFINGWELL ; WIDRIG, 2000). Esta realidade tem sido a principal razão da emergência de um conjunto de métodos ágeis que procuram enfrentar da melhor forma estas mudanças e os riscos relacionados (HIGHSMITH ; COCKBURN, 2001).

A filosofia das metodologias ágeis visa gerar o menor número de artefatos, ter maior comunicação verbal, existir pelo menos um representante do cliente na equipe de desenvolvimento (*on-site customer*), ter pequenos releases e ser adaptável a mudanças

iterativas (refatoração), isso tem feito crescer a aplicação destes métodos nas empresas. Entretanto, é grande o desconhecimento dos profissionais sobre como lidar com ER nestas metodologias.

Desta forma, o desenvolvimento deste trabalho, motiva-se pela necessidade das empresas de otimizar os processos compostos pela ER, fase principal da engenharia de software e fase onde falhas detectadas permitem um ganho de tempo muito importante no processo de desenvolvimento de software e pelo fato de apresentar um estudo que situe os profissionais em como lidar com a ER nas metodologias ágeis.

1.3 ABRANGÊNCIA

Este trabalho compreende ao estudo dos processos da ER, sua importância e sua aplicação dentro do contexto das metodologias ágeis. Estão sendo consideradas as seguintes metodologias: *Extreme Programming* (XP), Scrum, *Feature Driven Development* (FDD) e *Adaptive Software Development* (ASD), revisando suas principais práticas, princípios e participações dos perfis envolvidos.

Foi elaborada uma análise comparativa entre as metodologias estudadas, suas principais vantagens e limitações. Como resultado, pretende-se orientar aos profissionais sobre como devem ser tratados os processos de ER nestas metodologias. Os resultados serão refletidos com a implementação de um caso prático.

1.4 ESTRUTURA DO TRABALHO

No decorrer dos capítulos é apresentado, na ordem:

a) O estudo realizado sobre ER que está dividido em duas partes. A primeira abrange o processo de desenvolvimento de requisitos e é composta por: levantamento de requisitos, análise de requisitos, documentação de requisitos e validação de requisitos. A segunda abrange o processo de gestão de requisitos, composta por: gestão de qualidade de requisitos e gestão de configuração e mudança de requisitos;

b) O estudo das metodologias ágeis. Divididas em metodologias de desenvolvimento (*Extreme Programming*) e Metodologias de gerenciamento (Scrum, *Feature Driven Development e Adaptive Software Development*), serão apresentadas informações como definição, valores, princípios e práticas;

c) Foram abordados artigos relacionados de forma crítica que serviram como embasamento para o desenvolvimento do trabalho de conclusão de curso.

d) Foi apresentado o estudo de caso, análise dos métodos ágeis, estudo de viabilidade, aplicação das metodologias ágeis e as melhores práticas como resultado do estudo.

e) Foi realizada a conclusão do trabalho, e sugeridos possíveis trabalhos futuros.

2 ENGENHARIA DE REQUISITOS

Compreender a natureza dos problemas pode ser muito difícil, especialmente se o sistema for novo, portanto, é difícil estabelecer com absoluta certeza o que o sistema deve fazer. A descrição das funções e restrições são os requisitos para o sistema. Um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar, para atingir seus objetivos (PFLEEGER, 2004, p. 111).

Pode-se definir a ER como o processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos de sistema (SOMMERVILLE, 2005, p. 103). Este processo abrange os processos de levantamento de requisitos, análise e negociação, documentação e validação.

A ER tem como papel realizar a interação entre as pessoas que requisitam serviços ou impõem restrições, tais como usuários, clientes e desenvolvedores, entre "o que" deve ser feito e "como" deve ser feito (SOARES, 2008). É necessário nesta etapa, levantar, analisar conflitos, validar, priorizar, modificar e reusar requisitos, rastreá-los considerando sua origem, os componentes arquiteturais e o código que os implementam, dentre outras tarefas.

Depois de detalhar alguns conceitos que definem a ER e sua importância dentro do processo de desenvolvimento de software, a continuação, e de uma forma geral se ilustra como se classifica a mesma, conforme a figura 1:

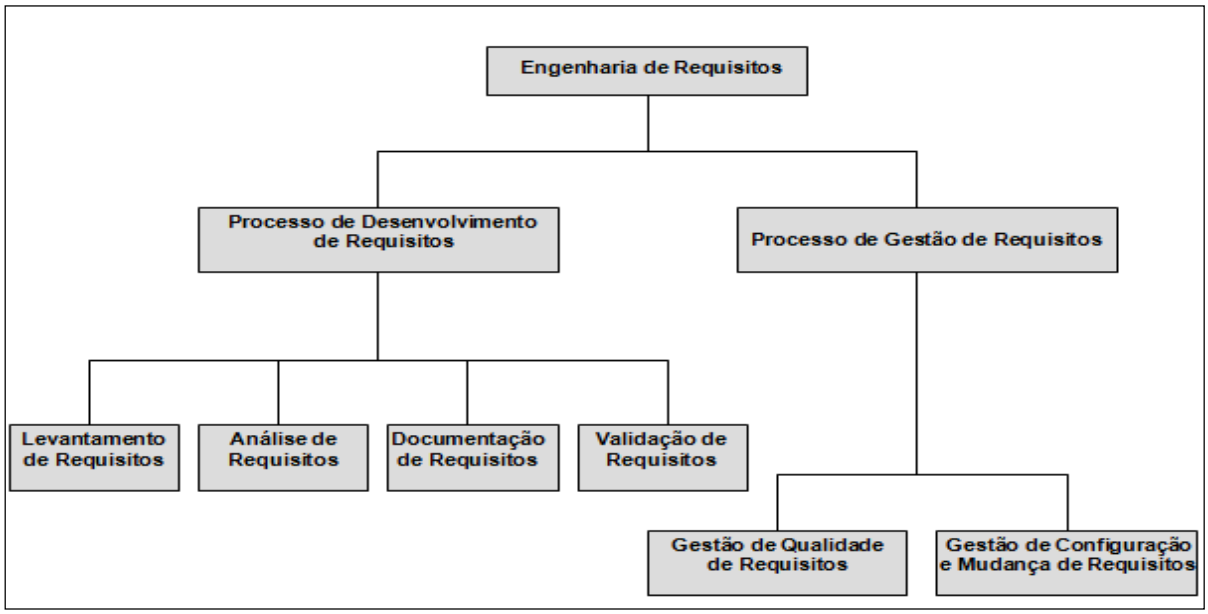

Figura 1 - Fluxograma de classificação da ER. Fonte: SOARES (2008).

2.1 NÍVEIS DE REQUISITOS

Alguns dos problemas que surgem durante o processo de ER são resultados da falta de entendimento e separação entre os níveis de requisitos (requisitos de usuários, requisitos de sistema). A continuação segue a definição destes níveis de requisito (SOMMERVILLE, 2005):

a) Requisitos de usuário: são declarações, em linguagem natural e também em diagramas. Contudo, vários problemas podem surgir quando os requisitos são descritos em linguagem natural:

Falta de clareza: dificuldade na utilização da linguagem de maneira precisa e sem ambigüidade, sem produzir um documento de difícil leitura.

Confusão de requisitos: os requisitos funcionais e não funcionais, os objetivos do sistema e as informações sobre o projeto podem não estar claramente definidos.

Fusão de requisitos: requisitos diferentes podem ser expressos juntos como um único requisito.

b) Requisitos de sistema: estabelecem detalhadamente o que o sistema deverá fazer e não como deve ser implementado. Linguagem natural é utilizada freqüentemente para escrever especificações de requisitos de sistema, mas podem surgir os seguintes problemas:

Compreensão da linguagem natural: depende do uso das mesmas palavras para o mesmo conceito, pelos leitores e por quem escreve as especificações. Isso leva a divergências, devido à ambigüidade da linguagem natural.

Especificação de requisitos em linguagem natural: é muito flexível, sendo possível expressar a mesma idéia de formas ou modos diferentes. Fica por conta do leitor descobrir quando os requisitos são os mesmo e quando são diferentes.

Dificuldade para padronizar os requisitos de linguagem natural: pode ser difícil encontrar todos os requisitos relacionados, sendo possível deparar com situações lugar de revisar um grupo de requisitos relacionados, tenha que ser revisado cada requisito.

Na figura 2 é importante ter claro que um único requisito de usuário pode gerar um ou mais requisitos de sistema.

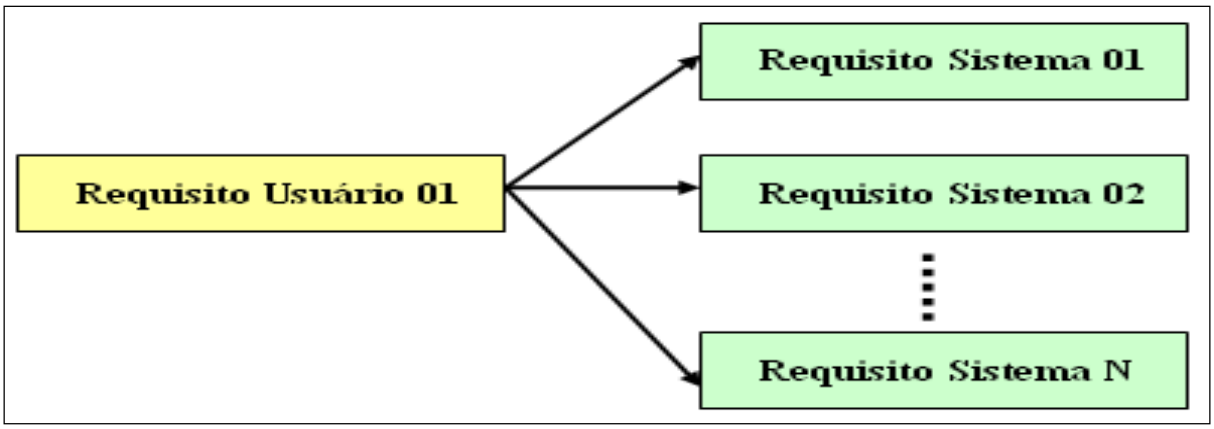

Figura 2 – Requisitos de Usuário por Requisito de Sistema. Fonte: SOMMERVILLE (2005)

2.2 TIPOS DE REQUISITOS DE SISTEMA

Os requisitos de sistema são classificados como funcionais, não funcionais ou requisitos de domínio (SOMMERVILLE, 2005), assim como segue na figura 3:

Figura 3 – Classificação dos Requisitos. Fonte: SOMMERVILLE (2005).

a) Requisitos funcionais: estão intimamente ligados às funcionalidades propostas pelo sistema, e que serão usadas na resolução do problema do contratante, e atenderão todas suas necessidades funcionais. Resumidamente, descrevem serviços e funções de sistema e como o sistema deve se comportar quando determinadas entradas são fornecidas. Exemplo: controle de reserva de salas e equipamentos.

Em geral os requisitos funcionais estão associados a:

- Funcionalidades do sistema;
- Serviços que o sistema deve prover;
- Comportamento do sistema a determinadas entradas;

- Funções que o sistema não deve suportar.

b) Requisitos não funcionais: estão geralmente ligados a qualidade do produto como, por exemplo, robustez, segurança ou integridade. Então descreve uma restrição imposta ao sistema. Exemplo: tempo de resposta, uso de linguagem de programação específica. Os requisitos não-funcionais definem, em geral, restrições aos sistemas propostos. Muitas dessas restrições refletem restrições do usuário ao processo de desenvolvimento do software como restrições organizacionais, orçamentárias, legais, etc.

Os requisitos não funcionais são associados à qualidade, desempenho, portabilidade, precisão, confiabilidade, segurança. Por sua vez os requisitos não funcionais se classificam em:

- Requisitos de Produto: especificam o comportamento do produto.
- Requisitos Organizacionais: são procedentes de políticas e procedimentos nas organizações do cliente e desenvolvedor.
- Requisitos Externos: são fatores externos do sistema ou a seu processo de desenvolvimento.

2.3 PROCESSO DE DESENVOLVIMENTO DE REQUISITOS

Como já se definiu na figura 1, existem quatro atividades do processo de desenvolvimento de requisitos, levantamento de requisitos, análise e negociação, documentação e validação, as mesmas serão tratadas de forma mais detalhada:

2.3.1 Levantamento de Requisitos

Nesta fase são obtidos com detalhes os requisitos do cliente. Descobrem-se os requisitos do sistema através da observação de sistemas existentes, da discussão com usuários e da análise da tarefa. O levantamento de requisitos tem como objetivo obter conhecimento relevante para o problema. Prever o mais correto entendimento de o que é esperado do software.

Dentro das atividades envolvidas para o levantamento de requisitos, é necessário compreender o domínio da organização e do projeto para que haja uma facilidade na comunicação com o requisitante. Uma ação importante é conversar com os usuários do sistema, pois eles poderão passar as necessidades de forma mais detalhada e clara, achando

assim problemas e buscando soluções para o mesmo. As maiores dificuldades que surgem não são computacionais, mas de comunicação, pois o objetivo é extrair do usuário o que ele espera do sistema a ser desenvolvido. Caso ele não possa passar as informações de forma detalhada, ou não consiga se articular poderá haver um erro ou problema no entendimento. Recai no engenheiro de requisitos a tarefa de discernir o que é relevante entre as informações dadas pelo usuário, bem como lidar adequadamente com as declarações vagas do usuário a respeito do que o mesmo espera de sistemas computacionais.

Assim a tarefa de levantar requisitos não é trivial. Também é necessário analisar os requisitos em relação a inconsistências e incompletudes, bem como negociar, solucionando conflitos, de forma que um conjunto de requisitos seja acordado. Estas atividades são realizadas, na maioria de vezes, paralelamente e/ou de forma intercalada. O objetivo é delimitar um conjunto de requisitos que atendam os desejos dos usuários.

O levantamento de requisitos é desafiador. Ele exige a colaboração de vários participantes com diferentes experiências. Em geral os clientes e usuários são especialistas em seu domínio e têm uma idéia geral do que o sistema deverá ser, mas tem pouca ou nenhuma experiência no desenvolvimento de aplicações.

Muito importante ter conhecimento de que erros introduzidos durante o levantamento de requisitos são tardiamente encontrados e têm um grande custo pra serem corrigidos, por isso, a obrigação de ter especial cuidado, alto grau de análise e domínio da situação no momento do levantamento.

Entre as técnicas para levantar requisitos pode-se apontar (SOMMERVILLE, 2005):

a) Entrevistas: neste caso, não se pode haver qualquer tipo de interferência por parte do entrevistador. É preciso deixar o entrevistado “despejar” informação e somente depois questionar ou agregar informações.

Por requisitarem planejamento, as entrevistas devem ser planejadas: objetivo, organização da entrevista, seleção dos participantes, questionário prévio, *feedback*, etc.

Como resultado das entrevistas deve ser gerado um documento de compromisso com os participantes.

b) Cenários: os cenários são muito utilizados de forma a apresentar ao usuário o comportamento do sistema. As pessoas geralmente acham mais fácil relacionar exemplos da vida real do que descrições abstratas. Elas podem compreender e criticar um cenário de como poderiam interagir com um sistema de software. Os engenheiros de requisitos podem utilizar as informações obtidas com essa discussão para formular os requisitos reais do sistema.

Os cenários podem ser particularmente úteis para acrescentar detalhes a um esboço da descrição de requisitos. Estas são exemplos de sessões de interação. Cada cenário aborda um pequeno número de possíveis interações. Diferentes tipos de cenários foram desenvolvidos, e eles fornecem diferentes tipos de informações, com diferentes tipos de detalhes sobre o sistema.

A obtenção de requisitos com base em cenários pode ser utilizada informalmente, e o engenheiro de requisitos trabalha com usuários para identificar cenários e captar detalhes desses cenários. Como alternativa, uma abordagem mais estruturada, como os cenários de eventos ou casos de uso, pode ser empregada.

c) Etnografia: é uma técnica de observação que pode ser utilizada para compreender os requisitos sociais e organizacionais. Um analista se insere no ambiente de trabalho em que o sistema será utilizado. O trabalho diário é observado e são anotadas as tarefas reais em que os participantes estão envolvidos. O valor da etnografia é que ela ajuda a descobrir requisitos de participantes implícitos, que refletem os processos reais, em vez dos processos formais, em que as pessoas estão envolvidas.

A etnografia é particularmente eficaz na descoberta de dois tipos de requisitos:
Os requisitos derivados de maneira como as pessoas realmente trabalham, em vez da maneira pela qual as definições de processo dizem como elas deveriam trabalhar.
Os requisitos derivados da cooperação e conscientização das atividades de outras pessoas.

d) Levantamento orientado a pontos de vista: existindo dentro de uma organização vários tipos de usuários finais, podem existir muitos pontos de vista que devem ser considerados. Os diferentes pontos de vista a respeito de um problema “vêem” o problema de modos diferentes. Contudo, suas perspectivas não são inteiramente independentes, mas em geral apresentam alguma duplicidade, de modo que apresentam requisitos comuns.

As abordagens orientadas a pontos de vista, na ER, reconhecem esses diferentes pontos de vista e os utilizam para estruturar e organizar o processo de levantamento e os próprios requisitos.

Para o levantamento orientado a pontos de vista podemos utilizar, entre outros, o método VORD (*viewpoint-oriented requirements definition* – definição de requisitos orientada a pontos de vista) e o *Brainstorming*.

e) Prototipação: lida basicamente em oferecer ao usuário as interfaces do sistema para que ele experimente o aplicativo verificando os seus requisitos e encontrando novos.

Apresenta um custo alto de desenvolvimento (protótipos são desenvolvimentos parciais que são descartados depois).

Técnicas de desenvolvimento ágil vêm sistematicamente substituindo protótipos por entregas parciais completas.

2.3.2 Análise de Requisitos

Na fase de análise acontece a classificação dos requisitos, essa atividade considera o conjunto não estruturado de requisitos e os organiza em grupos coerentes.

Quando múltiplos usuários estão envolvidos, os requisitos devem apresentar conflitos. Esses conflitos precisam ser encontrados e solucionados. Esta atividade também permitirá definir os requisitos por nível de importância. Os usuários são envolvidos para poder definir quais são os requisitos mais importantes.

Os requisitos são verificados, a fim de se descobrir se eles são completos e consistentes, e se estão em concordância com o que os usuários realmente desejam do sistema. Muito importante saber que erros de requisitos podem sair muito caro para a empresa, podendo sair até mais caro que um erro na implementação, por isso da importância de uma correta análise de requisitos, isto por que requisitos mal especificados podem levar a grandes custos de retrabalho quando esses erros são descobertos durante o desenvolvimento ou depois que o sistema está em operação. A razão disso é que uma mudança nos requisitos, em geral, significa que o projeto de sistema e a implementação também devem ser modificados e que o sistema tem de ser novamente testado.

Varias técnicas podem ser utilizados para este fim. Entre elas destacam-se:

a) Lista de checagem de análise: é uma lista de questões, as quais o analista pode usar para avaliar cada requisito. Cada item serve de guia na avaliação do requisito. No final da checagem. Pode obter-se uma lista de problemas associados com requisitos. Estes problemas podem ser solucionados através de negociação ou se necessário com novo levantamento de requisitos.

b) Matrizes de interação: é utilizada para descobrir as interações entre requisitos, apontando possíveis conflitos entre requisitos. A atividade de negociação pode corrigir estes conflitos.

c) Prototipação: os protótipos na etapa de levantamento de requisitos podem ser aperfeiçoados na etapa de análise, possibilitando uma análise mais completa dos requisitos.

Protótipos facilitam o envolvimento dos diferentes usuários no levantamento, análise e negociação de requisitos.

Após a análise de requisitos, sendo descobertos conflitos ou problemas, ocorre o processo de negociação de requisitos. Esta atividade visa solucionar problemas que foram advindos do conflito entre os diversos usuários, os quais podem atribuir diferentes prioridades aos requisitos. A negociação consiste em que todos os usuários entrem em consenso em relação a um conjunto de requisitos definidos bem como em relação às prioridades definidas para os mesmos.

2.3.3 Documentação de Requisitos

Um documento de requisito é uma descrição oficial e detalhada dos requisitos de um sistema para os clientes, usuários e desenvolvedores, a qual também deve ser de fácil entendimento para os mesmo. Organiza e formaliza os requisitos obtidos do usuário. Qualquer falta de entendimento desses requisitos deverá ser solucionado nesta fase. Transforma a informação obtida durante a etapa de análise em um documento que define um conjunto de requisitos realmente desejados pelo cliente.

O processo de ER é geralmente guiado por um método adotado para a realização das atividades. Estes métodos possuem uma abordagem sistemática para produzir modelos do sistema. Quando se modela requisitos, produzem-se modelos, os quais podem pertencer a abordagens tais como: modelagem de fluxo de dados, abordagens orientada a objetos e métodos formais.

A atividade de documentação de requisitos é intercalada em muitos momentos com as atividades de levantamento, análise e negociação de requisitos. O resultado da documentação de requisitos inclui os requisitos acordados representados através de um documento em linguagem natural bem como uma especificação dos requisitos do sistema.

Esta especificação em geral consiste em diagramas e modelos pertencentes à metodologia adotada em desenvolvimento.

Em (IEEE Std. 830, 1998; SOMMERVILLE, 2005; VOLERE, 2006) são definidos *templates* para o documento de requisitos. Eles apresentam estruturas para organizar os requisitos no documento, ver figura 4.

Volere (2006)	IEEE Std. 830
PROJETO	1. Introdução
1. O propósito do produto.	1.1 Propósito
2. Clientes, fregueses e outros interessados	1.2 Escopo
3. Usuários do produto.	1.3 Definições, acronimos e abreviações
RESTRIÇÕES DO PROJETO	1.4 Referencias
4. Restrições obrigatórias.	1.5 Visão geral
5. Convenções para nomes e definições.	2. Descrição global
6. Fatos relevantes e suposições	2.1 Perspectivas do produto
REQUISITOS FUNCIONAIS	2.2 Funções do produto
7. O escopo do trabalho	2.3 Características dos usuários
8. O escopo do produto	2.4 Restrições
9. Requisitos funcionais e de dados	2.5 Suposições e dependências
REQUISITOS NÃO FUNCIONAIS	3. Requisitos específicos
10. Requisitos de amigabilidade	Apêndices
11. Requisitos de usabilidade	
12. Requisitos de performance	**Sommerville (2005)**
13. Requisitos operacionais	
14. Requisitos de manutembilidade e portabilidade	1. Prefácio
15. Requisitos de segurança	2. Introdução
16. Requisitos culturais e políticos	3. Glossario
17. Requisitos legais	4. Requisitos do usuário
ASSUNTOS DE PROJETO	5. Arquitetura do sistema
18. Assuntos em aberto	6. Requisitos do sistema
19. Soluções de prateleira	7. Modelo do sistema
20. Novos problemas	8. Evolução do sistema
21. Tarefas	9. Apêndices
22. Agenda	10. Índice
23. Riscos	
24. Custos	
25. Documentação para o usuário e treinamento	
26. Requisitos para as próximas versões	
27. Ideias para soluções	

Figura 4 – Exemplos de *templates* para documentos de requisitos. Fonte: IEEE Std. 830 (1998); SOMMERVILLE (2005); VOLERE (2006).

De maneira geral, um documento de requisito deve definir, no mínimo, um glossário onde são definidos o UDI (Universo de informações), que contenha toda a informação de domínio do problema, compreendendo os agentes (atores, usuários) mais outras fontes de informação, e uma lista de sentenças de requisitos, geralmente, em linguagem natural não estruturada e organizada de diferentes maneiras.

2.3.4 Validação de Requisitos

Ocupa-se de validar se os requisitos realmente definem o sistema que o cliente deseja (SOMMERVILLE, 2005, p. 115). Quem valida os requisitos é o gestor/cliente ou usuários finais. Tem muito em comum com a análise de requisitos, uma vez que se preocupa em descobrir problemas nos requisitos. Mas são processos distintos, já que a validação deve se ocupar da elaboração de um esboço completo de documentos de requisitos, enquanto a análise envolve trabalhar com requisitos incompletos.

Durante o processo de validação de requisitos, diferentes tipos de verificações devem ser realizados sobre os requisitos no documento de requisitos. Entre os principais tipos de verificação destacam-se os seguintes (SOMMERVILLE, 2005, p. 116):

a) Verificações de validade: um usuário pode pensar que um sistema é necessário para realizar certas funções. Contudo, mais estudos e análises podem identificar funções adicionais ou diferentes, que são exigidas. Os sistemas têm diversos usuários com necessidades diferentes e qualquer conjunto de requisitos é inevitavelmente uma solução conciliatória da comunidade de usuários.

b) Verificações de consistência: os requisitos em um documento não devem ser conflitantes, ou seja, não devem existir restrições contraditórias ou descrições diferentes para uma mesma função de sistema.

c) Verificações de completeza: o documento de requisitos deve incluir requisitos que definam todas as funções e restrições exigidas pelo usuário do sistema.

d) Verificações de realismo: utilizando o conhecimento da tecnologia existente, os requisitos devem ser verificados, a fim de assegurar que eles realmente possam ser implementados.

e) Facilidade de verificação: para reduzir o potencial de divergências entre cliente e fornecedor, os requisitos do sistema devem sempre ser escritos de modo que possam ser verificados. Isso significa que um conjunto de verificações pode ser projetado para mostrar que o sistema entregue cumpre com esses requisitos.

Existe uma série de técnicas de validação de requisitos que podem ser usadas em conjunto ou individualmente (SOMMERVILLE, 2005, p. 116), entre as mais importantes encontram-se:

a) Revisão de requisitos: é a técnica mais comum de validação de requisitos, na revisão os usuários e desenvolvedores se reúnem para discutir o documento de requisitos em busca de conflitos, contradições, erros e omissões. A partir dessas descobertas, podem-se adotar medidas que solucionem os conflitos. Problemas típicos encontrados nas revisões incluem: incompletude das descrições dos requisitos, ambigüidade, bem como a não obediência a padrões da organização.

b) Prototipação: protótipos são desenvolvidos com o intuito de permitir uma melhor representação dos requisitos do sistema. Enquanto tradicionalmente, o usuário tem de esperar até etapas no final do processo de desenvolvimento, para visualizar uma versão executável do sistema, com o desenvolvimento de protótipos, usuários podem ter uma idéia antecipada de

como o sistema executável funcionará. É observado que a técnica de prototipação geralmente é usada para ajudar nas atividades de levantamento e análise de requisitos. Contudo, ela também pode ser usada para validar requisitos. Após o documento de requisitos já ter sido definido e acordado, pode se aperfeiçoar o protótipo desenvolvido no levantamento e análise e utilizá-lo para validar os requisitos, com um auxílio mais efetivo dos usuários/clientes.

c) Geração de casos de teste: desenvolver testes para os requisitos a fim de verificar testabilidade. Se o teste for difícil ou impossível de implementar, pode significar que os requisitos precisam ser revistos. Por exemplo, técnicas baseadas em cenários podem ser usadas para levantar, analisar requisitos e criar casos de teste para os cenários desenvolvidos. Casos de teste podem ser aplicados de forma simulada nos cenários desenvolvidos. Se surgirem dificuldades para criar os casos de teste bem como para executá-los através de simulação, há uma grande probabilidade de existirem problemas com os requisitos. É menos oneroso e traumático descobrir estes problemas na atividade de validação do processo de ER ao invés das etapas finais do processo de desenvolvimento de software.

d) Validação de Modelos: geralmente, quando se documenta requisitos, os mesmos são descritos através de linguagem natural ou diagramas. Tipicamente, cria-se um documento em linguagem natural descrevendo os requisitos do sistema. Para detalhar melhor o funcionamento do sistema são desenvolvidos modelos de sistema. O desenvolvimento destes modelos está associado á abordagem de desenvolvimento de software adotada. Estes modelos precisam ser validados para demonstrar que os mesmos são consistentes tanto internamente como externamente. Eles devem representar os reais requisitos do usuário.

2.4 PROCESSO DE GESTÃO DE REQUISITOS

Projetos de desenvolvimento de software têm como principal objetivo construir sistemas que atendem às necessidades dos seus clientes em acordo com os objetivos do negócio.

Devido às constantes mudanças no negócio e a novas necessidades do cliente é preciso ter uma adequada gestão sob os requisitos já definidos durante o processo de desenvolvimento de requisitos.

Grandes sistemas de software são desenvolvidos para melhorar ou aperfeiçoar processos do dia a dia. Ao atualizar ou substituir um sistema, é difícil prever o impacto que essa alteração pode causar.

Um sistema é complexo já que diferentes usuários podem utilizar, sendo que cada um tem uma necessidade diferente. Os requisitos finais são uma conciliação entre todos os requisitos para todos os usuários.

Os requisitos são solicitados por uma pessoa ou um grupo de pessoas com algumas restrições de acesso ou até mesmo restrições financeiras, o que na maioria das vezes acaba divergindo com as necessidades dos usuários.

Até mesmo quando a parte física de um software ou as prioridades são alteradas, o sistema tem que ser alterado para atender a essas modificações. Alterações físicas acabam afetando os Requisitos não funcionais.

O gerenciamento de requisitos é o processo de compreender e controlar as mudanças nos requisitos de sistemas. Este processo é realizado em conjunto com outros processos da ER. O Planejamento se inicia ao mesmo tempo em que o levantamento inicial de requisitos e o gerenciamento ativo dos requisitos devem começar assim que um esboço da versão do documento de requisitos estiver disponível (SOMMERVILLE, 2005, p. 118).

2.4.1 Gestão da qualidade de requisitos

Para que um software tenha uma boa qualidade, normas são aplicadas durante o processo de desenvolvimento. Todas essas normas são aplicadas desde o inicio até a finalização do processo.

Para obter uma boa qualidade, dois padrões podem ser aplicados:

a) Padrão de produto: aplicado ao produto no desenvolvimento, padrão de documentação e de codificação.

b) Padrão de processo: como o processo tem que ser executado durante seu desenvolvimento, especificações, testes e descrição dos documentos gerados classificam padrão de processo.

Os dois padrões são muito próximos, já que o padrão de processo é voltado às saídas de processos e na maioria das vezes inclui atividades específicas onde o padrão de produto é seguido.

Importância dos padrões de software:

a) Encapsulamento das melhores práticas ou mais adequadas.

b) Melhor prática, onde é garantido que a qualidade foi aplicada com os padrões corretos.

c) Melhor continuidade quando uma pessoa da seqüência ao trabalho iniciado por outra pessoa.

As equipes de garantia de qualidade que estiverem desenvolvendo padrões, normalmente, deverão basear os padrões organizacionais nos padrões nacionais e internacionais. Com a utilização desses padrões como ponto de partida, a equipe de garantia deve elaborar um manual de padrões, que deve definir aqueles apropriados para sua organização. (SOMMERVILLE, 2005, p. 461)

Algumas vezes, esses padrões são classificados como burocráticos e irrelevantes para os engenheiros de software, principalmente quando as atividades têm que ser oficializadas em formulários. Por esses motivos, alguns engenheiros acabam ignorando ou até mesmo afirmando que esses padrões não são aplicáveis ao projeto que eles irão desenvolver.

Algumas etapas que devem ser seguidas:

a) Entender o motivo e a importância desses padrões e aplicar os mesmos.

b) Rever ou até mesmo alterar para que a qualidade fique mais aplicável e segura.

c) Fornecer software que auxilie nos padrões de qualidade.

2.4.2 Gestão de Configuração e Mudanças de Requisitos

Os sistemas de software acabam sofrendo mudanças ao longo de sua utilização, modificando sua vida útil. Para que essas mudanças sejam feitas, registradas e aplicadas ao sistema de maneira que o custo não seja elevado, um processo de gerenciamento de mudança e as ferramentas *case* associadas serão aplicados.

Logo que um software está em testes ou já está sendo utilizado pelo cliente, o processo de gerenciamento de mudanças pode ser aplicado. Esse processo deve ser aplicável para garantir que essas alterações tenham um bom custo e benefícios, mas sempre tendo o processo sob controle.

Para se iniciar uma mudança, alguns passos têm que ser seguidos. Preenchimento de um formulário de solicitação de mudança, *Change Request Form* – CRF (ver anexo A), onde é definida a mudança que será feita. Este formulário deve ser preenchido pela pessoa que identifica a mudança, uma vez preenchido deverá passar por um comitê de avaliação da solicitação, onde se determina o impacto que pode ocasionar dita mudança sob o requisito ou requisitos já definidos. Uma vez avaliado, o comitê através de seu responsável determina a aprovação ou não da mudança.

No CRF também é registrado o valor dessas mudanças, as datas, aprovações, validações, a implementação e o custo das alterações. Ele também pode conter uma seção onde o engenheiro de manutenção faz um rascunho de como a mudança deverá ser implantada. Todas as mudanças serão registradas no banco de dados de configuração.

Quando o formulário de mudanças é preenchido, ele passa por uma análise, onde será confirmado se a mudança é aceitável. Alguns formulários podem ser recusados se houver erros de compreensão ou até mesmo problemas já conhecidos. Após a análise ser efetuada, e for constatado que a alteração foi feita em outra mudança, a alteração tem que ser recusada, enviando para a pessoa que fez a solicitação.

Se tudo estiver correto, é iniciado então o processo de avaliação das mudanças e do custo relacionado. Será verificado se a mudança causará impacto no resto do sistema e como implementar essa(s) mudança(s). Em seguida, é necessário calcular seu custo. Os desenvolvedores só começam a trabalhar nas mudanças quando todo o processo já está aprovado. Após as mudanças passarem pela equipe de desenvolvedores, a equipe de testes verifica se o sistema não possui nenhum tipo de erro.

3 METODOLOGIAS ÁGEIS

A indústria de software sempre contou com métodos cujo processo de desenvolvimento era baseado em um conjunto de atividades predefinidas, descritas como processos prescritivos (AMBLER 2004), onde muitas vezes, o trabalho começava com o levantamento completo de um conjunto de requisitos, seguido por um projeto de alto-nível, de uma implementação, de uma validação e, por fim, de uma manutenção (SOMMERVILLE 2003). Entretanto estes métodos são considerados muito burocráticos e eficientes para projetos grandes e que não sofram muitas mudanças em seus requisitos (FOWLER 2001).

A partir da década de 90, novos métodos começaram a surgir de forma a trabalhar em cima de uma abordagem de desenvolvimento ágil. Estes métodos surgiram como uma reação aos métodos tradicionais, que utilizam processos prescritivos, ganhando ano após ano novos adeptos.

Em fevereiro de 2001 um grupo composto por 17 metodologistas, dentre eles Robert C. Martin, Jim Highsmith, Kent Beck, Mike Beedle, Alistair Cockburn, Martin Fowler, Steve Mellor, Ken Schwaber, Jeff Sutherland, dentre outros, formaram a aliança para o desenvolvimento ágil de software, ou também conhecida como Aliança Ágil (Ágile Alliance).

A aliança ágil formulou um manifesto contendo um conjunto de princípios que definem critérios para os processos de desenvolvimento ágil de software (AMBLER 2004).

O manifesto ágil baseia-se em quatro valores, sendo eles:

a) Indivíduos e interações valem mais que processos e ferramentas;

b) Um software funcionando vale mais que uma documentação extensa;

c) A colaboração do cliente vale mais que a negociação de contrato;

d) Responder as mudanças vale mais que seguir um plano.

O manifesto ágil tem por base os quatro valores, entretanto, de forma a facilitar seu entendimento, os membros da aliança ágil, refinaram as filosofias contidas em doze princípios, sendo eles (BECK 2001):

a) A prioridade é satisfazer ao cliente através de entregas de software de valor contínuas e freqüentes;

b) Receber bem as mudanças de requisitos, mesmo em uma fase avançada, dando aos clientes vantagens competitiva;

c) Entregar software em funcionamento com freqüência de algumas semanas ou meses, sempre na menor escala de tempo;

d) As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente durante todo o projeto;

e) Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessários para a realização do trabalho;

f) A maneira mais eficiente da informação circular dentro da equipe é através de uma conversa *face-a-face*;

g) Ter o software funcionando é a melhor medida de progresso.

h) Processos ágeis promovem o desenvolvimento sustentável. Todos envolvidos devem ser capazes de manter um ritmo de desenvolvimento constante.

i) Atenção contínua a excelência técnica e a um bom projeto aumentam a agilidade.

j) Simplicidade é essencial.

k) As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas.

l) Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz e então se ajustar e adaptar seu comportamento.

Basicamente os métodos ágeis se diferem dos tradicionais em dois aspectos:

a) São adaptativos ao invés de prescritivos;

b) São orientados às pessoas ao invés dos processos.

É importante destacar que os métodos ágeis não são contra os modelos de processos de desenvimento tradicionais. Trata-se de uma alternativa à burocracia existente.

Nas próximas seções são apresentadas as principais características da Modelagem Ágil e dos métodos ágeis XP, SCRUM, FDD e ASD, sendo divididos em metodologias de desenvolvimento e de gerenciamento de sistemas.

3.1 METODOLOGIA DE DESENVOLVIMENTO DE SISTEMAS

Foram abordadas as metodologias ágeis Extreme Programming (XP) e Modelagem Ágil. Classificadas como metodologias de desenvolvimento de sistemas, tem como foco a aplicação das melhores práticas, apresentando formas de extrair o melhor resultado no desenvolvimento de sistemas.

3.1.1 Modelagem Ágil

Introduzida por Scott W. Ambler em 2002, a modelagem ágil ou *ágile modeling* tem por objetivo a geração de documentos e modelos eficazes, que dêem suporte ao desenvolvimento de sistemas.

Trata-se de uma coleção de práticas com princípios e valores podendo ser aplicados por profissionais de software no dia a dia (AMBLER 2002).

A modelagem ágil não define procedimentos detalhados de como criar um modelo e sim dá orientações de como o modelador poderá ser mais efetivo.

Uma observação importante a se ressaltar é o fato da modelagem ágil não ser um método completo. Mesmo apresentando a importância das atividades de programação, testes, gerência, suporte ao sistema, dentre outros, ela não as contempla em seu escopo.

A modelagem ágil deve ser utilizada em conjunto com outros métodos de desenvolvimento (AMBLER 2004).

3.1.1.1 Definição

Trata-se de um complemento aos processos existentes, focando na modelagem e em segundo plano na documentação. De forma resumida, a modelagem ágil é flexível o suficiente para deixar aberto, por exemplo, a necessidade da utilização de ferramentas case ou não, se a ferramenta case poderá deixar o processo mais simples e eficiente então pode ser usada.

3.1.1.2 Valores

Segundo AMBLER (2002), a metodologia que mais se aproxima dos valores da modelagem ágil, é o método *Extreme Programming* (XP). Do conjunto de cinco valores, quatro possuem similaridades entre as metodologias, sendo elas: comunicação, simplicidade, *feedback* e coragem. O quinto valor surgiu como uma necessidade da modelagem ágil, descrita como humildade.

A humildade refere-se à forma que os envolvidos no projeto se relacionam. Os mesmos devem se respeitar, entender os mais variados pontos de vista e ter humildade suficiente para admitir que necessitam de ajuda em determinado momento.

3.1.1.3 Princípios

A modelagem ágil possui princípios estabelecidos

Princípios fundamentais ou centrais (AMBLER, 2002):

a) Software é seu principal objetivo: software que funcione;

b) Habilitar seu próximo esforço é um objetivo secundário: pensar sempre nas próximas funcionalidades;

c) Viaje com pouca bagagem: menos documentos durante o projeto - escolher documentos a serem mantidos durante o processo de desenvolvimento;

d) Assuma Simplicidade;

e) Aceite a Mudança;

f) Aplique Mudanças Incrementais;

g) Modele com um propósito: para atender a realidade, para melhorar a comunicação;

h) Construa Múltiplos Modelos;

i) Trabalhe com Qualidade;

j) Obtenha rápido f*eedback*;

k) Maximize o investimento do s*takeholder* (pessoa chave que representa a empresa);

Princípios suplementares:

a) Conteúdo é mais importante que representação;

b) Cada um tem algo a aprender com o outro;

c) Conheça seus modelos;

d) Conheça suas ferramentas;

e) Adapte o modelo à organização.

3.1.1.4 Práticas

A Modelagem Ágil também possui dois tipos de práticas, as práticas centrais e as práticas suplementares.

Práticas fundamentais ou centrais

a) Modelagem iterativa e incremental:

- Aplique o artefato correto;
- Crie vários modelos em paralelo;

- Itere entre diferentes artefatos;
- Modele em pequenos incrementos.

b) Trabalho em Equipe:

- Modelar com outras pessoas;
- Participação ativa do Stakeholder;
- Conhecimento coletivo (nunca deixe somente uma pessoa dominar todo o processo, pois se a mesma morrer, acabou o projeto);
- Exiba modelos publicamente (colocar em painéis, parede, etc.).

c) Simplicidade:

- Crie conteúdo simples;
- Descreva modelos simples;
- Use a ferramenta mais simples.

d) Validação:

- Considere a testabilidade;
- Prove com código.
- Práticas suplementares

e) Produtividade:

- Utilize padrões e normas de modelagem;
- Aplique padrões (design patterns) com sabedoria;
- Reutilize recursos existentes.

f) Documentação:

- Descarte Modelos temporários;
- Formalize os modelos de contrato "*Contract Models*";
- Atualize apenas quando dói (para que o modelo não fique inconsistente).

g) Propósito:

- Modele para entender;
- Modele para comunicar.

h) Boas Idéias:

- Conheça bem suas ferramentas;
- Refatoração (Refactoring);
- *Test-First Design.*

3.1.2 *Extreme Programming* (XP)

Tendo por criador Kent Beck, o método foi aplicado pela primeira vez em 1996 em um projeto chamado Chrysler *Comprehensive Compensation* (C3) para a empresa DaimlerChrysler.

Trata-se de um método eficiente, flexível e de baixo risco para equipes pequenas e médias que desenvolvem requisitos dinâmicos ou em constante mudança.

3.1.2.1 Valores do XP

Composto por quatro valores são divididos em:

a) Comunicação: tem por objetivo uma comunicação rápida e eficaz entre *stakeholders*;

b) Simplicidade: tem por objetivo simplificar continuamente o software. Comunicação e simplicidade caminham juntas, pois quanto maior a comunicação fica mais fácil identificar o que realmente necessita ser feito;

c) *Feedback:* tem por objetivo evidenciar o problema o mais rápido possível, seja pela equipe de desenvolvimento ou seja pelo cliente.

d) Coragem: tem por objetivo apresentar a importância da Coragem para se pedir ajuda, para reconstruir códigos, para informar sobre eventuais atrasos, dentre outros.

3.1.2.2 Práticas do XP

As práticas são divididas em doze elementos, sendo eles:

a) Jogo do planejamento: Composto por Cliente e Programadores, as responsabilidades se dividem em:

- Clientes: decisão sobre escopo, prioridade, composição e datas das entregas.
- Programadores: verificar estimativa de tempo de desenvolvimento para cada funcionalidade, avaliação dos riscos técnicos e decisão sobre o processo de trabalho.

O jogo do planejamento é dividido em duas etapas, planejamento da entrega e planejamento da iteração.

- Planejamento da entrega: trata-se de uma atividade onde o cliente apresenta as funcionalidades desejadas aos programadores, que por sua vez verificam e apresentam sua dificuldade. Uma vez conhecidas as funcionalidades e a dificuldade do mesmo, o cliente elabora um plano para o projeto.
- Planejamento da iteração: trata-se de uma atividade onde a equipe de desenvolvedores recebe orientações através do *user stories* ou histórias de usuários, que são compostas pelas informações das funcionalidades do sistema. Durante o Planejamento da Iteração, o cliente informa também as *user stories* que deseja para a próxima iteração. As iterações para a equipe de desenvolvimento construir o software variam de 1 a 4 semanas, sendo necessário entregar um software executável.

b) Entregas freqüentes: a cada iteração deve haver uma entrega contendo as *user stories* mais importantes. Assim a equipe entrega um software executável, sendo possível ao cliente a utilização do mesmo para avaliação ou para os usuários finais.

c) Metáfora: ajuda de forma a manter toda a equipe em sintonia com o projeto. Segundo ASTELS (2002), cada projeto do XP deve fazer uso de no mínimo uma metáfora para que seja possível orientar a equipe e fornecer um contexto compartilhado.

d) Projeto simples: o XP mantém o projeto o mais simples possível. Desta forma, quanto mais simples mais ágil ele se torna. A equipe XP não desenvolve um grande projeto inicial, mas projeta continuamente durante todo o tempo (FOWLER, 2001). Por isso da importância das iterações.

e) Testes: os testes são divididos em duas partes, sendo uma de responsabilidade do cliente (testes de aceitação) e outra de responsabilidade do programador (testes de unidade). Os testes de unidade têm por função, obrigar os programadores a analisar melhor o que será efetivamente codificado, sendo que para cada *user story* é escrito no mínimo um teste de aceitação.

f) *Refactoring:* Trata-se de uma técnica empregada na reconstrução do código, cujo principal objetivo é fazer com que o mesmo fique mais reutilizável e compreensível (FOWLER, 2001). O *Refactoring* deve ser feito sempre que possível e onde possível.

g) Programação em pares: "Todo software produzido em XP é construído por 2 programadores, sentados lado a lado, na mesma máquina. Desta forma, pode-se assegurar que todo código produzido é revisado por, pelo menos, outro programador" (JEFFRIES, 2001). É importante ressaltar que existem dois papéis em cada par, ou seja, o desenvolvedor que está com o teclado e com o mouse é responsável por pensar na melhor forma de se implementar o

método corrente. Já o outro desenvolvedor, deve analisar estratégicamente se a abordagem adotada irá funcionar, se existe algum outro teste a ser executado ou se existe alguma forma de simplificar e otimizar o código (BECK, 2000).

h) Propriedade coletiva: em um projeto XP, todos são responsáveis por todo o sistema, ou seja, no decorrer do projeto, qualquer um que perceba uma oportunidade de agregar valor a alguma parte do código, pode fazê-lo (BECK 2000).

i) Integração contínua: As equipes XP mantêm o sistema integrado todo o tempo. Sendo assim, o código é integrado e testado depois de algumas horas ou no máximo um dia após o desenvolvimento. Esta integração contínua evita ou descobre problemas de compatibilidade cedo.

j) Semana de 40 horas: XP não recomenda horas extras por períodos superiores à uma semana, além de passar noites trabalhando. Nestas condições, a possibilidade de ocorrer erros aumenta significativamente. A sobrecarga de trabalho é sintoma de sérios problemas no projeto (BECK 2000).

k) Cliente presente: O cliente deve estar sempre presente nas atividades, fazendo parte da equipe. A presença do cliente é importante, pois o mesmo poderá fornecer detalhes do sistema quando surgirem dúvidas. Entretanto, algumas vezes não é possível a presença do cliente, no mesmo local que os programadores, de qualquer forma é possível trabalhar com XP, desde que se tenha uma forma de comunicação.

l) Padrões de codificação: As equipes de XP seguem um padrão de codificação. Desta forma, todo código apresenta-se de forma familiar. A padronização do código mantém o mesmo consistente e fácil para o entendimento de toda a equipe. O padrão deve ser adotado voluntariamente por toda a equipe (BECK 2000).

3.1.2.3 Fases do processo do XP

O projeto ideal XP divide-se em: curta fase de desenvolvimento, anos de produção e refinamentos e encerra quando o projeto não possui mais sentido (BECK, 2000). A figura 5 indica as fases do Processo XP.

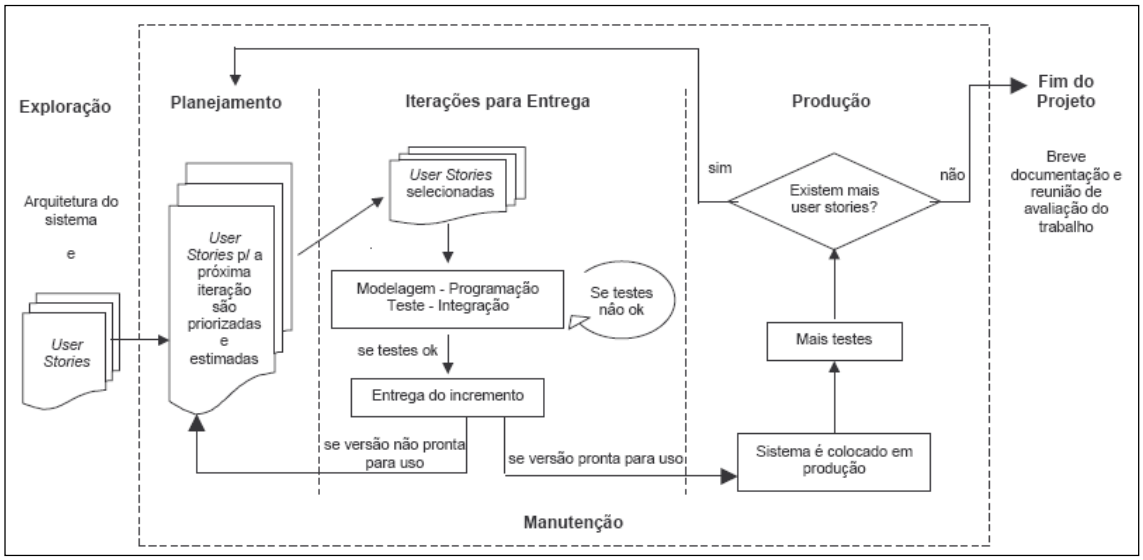

Figura 5 - **Fases do Processo do XP. FONTE: AUTORES (2009)**

a) Exploração

Tem por objetivo entender o real escopo do sistema e se inicia com o cliente escrevendo as *user stories*, figura 6.

Data: 22/12/2000 **Nova:** X **Dificuldade:** Média

Número da História: 005 - Criar Recibo **Prioridade do Usuário**: Alta **Técnico:** João Camargo

Referência Anterior: 001 – Efetuar Venda **Risco:** **Estimativa do Técnico:** 20 h

Descrição: Manter um recibo aberto com uma descrição breve de cada item escaneado e seu preço. Quando a venda for concluída, incluir o total na parte inferior do recibo.

Notas:

Acompanhamento da tarefa

Data	Estado	A realizar	Comentário

Figura 6 - Exemplo de User Story. FONTES: AUTORES (2009)

Paralelamente às *user stories*, os programadores testam e utilizam diferentes tecnologias e configurações, explorando as possibilidades para arquitetura do sistema. Tais testes devem ser feitos de várias maneiras, ou seja, diferentes pares devem testar diferentes

tecnologias e comparar. Caso o período não seja suficiente, então a mesma deve ser classificada como um risco ao projeto.

Esta pequena exploração permite que os programadores tenham idéia da implementação quando o usuário apresentar suas *user stories* (BECK 2000).

b) Planejamento

Tem por objetivo definir a menor data e o maior conjunto de *user stories* que serão realizadas na primeira entrega (BECK, 2000).

O cliente é quem decide quais *user stories* são mais importantes e que devem fazer parte da primeira entrega. Caso haja uma boa preparação durante a fase de exploração, são necessários somente alguns dias para a fase de planejamento.

Passos para auxiliar a fase de planejamento (WAKE, 2002):

User stories classificadas pelo cliente por valor: Alto, Médio e Baixo.

User stories classificadas pelos programadores por risco: Alto, Médio e Baixo.

Desenvolvimento de *user stories* com tempo estimado pelos programadores.

User stories escolhidas pelo cliente para a próxima entrega.

c) Iterações para entrega

É nesta fase que ocorre o maior trabalho de desenvolvimento, que incluem modelagem, programação, escrita e execução dos testes de unidade e aceitação e integração (AMBLER 2004).

Uma característica importante do XP é que sejam implementadas somente as funcionalidades que estejam marcadas para a iteração corrente (WELLS, 2001).

O prazo final de cada iteração deve ser respeitado, permitindo analisar qual a real velocidade do projeto durante a mesma.

Desta forma, o conjunto de *user stories* que farão parte da nova iteração estará melhor estimado. Ao final de cada iteração, o cliente terá completado a execução de todos os testes de aceitação e os programadores executam os testes de unidade escritos antes da implementação.

d) Produção

Tem início ao final da primeira iteração.

Novos testes devem ser executados de forma a provar a estabilidade do mesmo e verificar a possibilidade de se entrar em produção. Talvez seja necessário realizar alguns ajustes, que serão facilitados pelo fato de se possuir maior conhecimento do projeto.

Esta fase não significa que o sistema esteja concluído, o software estará em processo de evolução, atendendo assim novas necessidades ou eventuais alterações informadas pelo cliente, reiniciando assim a fase de planejamento (BECK 2000).

e) Manutenção

Trata-se do estado normal de um projeto XP, compreendendo as fases de planejamento, iterações para a entrega e produção, até a entrega final do sistema.

f) Fim do Projeto

Uma vez que não se tem mais o que agregar ao projeto e que o cliente esteja satisfeito é sinal que o mesmo chegou ao fim. Sendo assim, é necessário escrever algumas páginas sobre suas principais funcionalidades, que auxiliarão futuros desenvolvedores a entender determinados problemas.

Para a finalização do projeto é necessária uma reunião com todos os envolvidos a fim de reavaliar tudo que se passou.

3.1.2.4 Equipe do XP

Divididos em programadores, clientes, testadores, rastreadores, treinadores, consultores, gerentes, possuem responsabilidades específicas, conforme podemos verificar abaixo:

a) Programadores: são responsáveis por estimar prazos das *user stories*, escrever e executar os testes de unidade, implementar o sistema, trabalhar em par, fazer *refactoring* sempre que necessário, solicitar esclarecimento de um cliente.

b) Clientes: são responsáveis por definir as funcionalidades do software, escrever as *user stories*, definir prioridades para as *user stories*, escrever e executar os testes de aceitação e esclarecer dúvidas.

c) Testadores: são responsáveis por auxiliar os clientes a escrever os testes de aceitação, Executar os testes que forem solicitados e informar sobre o resultado à equipe.

d) Rastreadores: são responsáveis por coletar sinais do projeto, 1 ou 2 vezes por semana.

e) Treinador: são responsáveis por processo de desenvolvimento, Notificação de pessoas que por ventura estejam desviando do foco, Garantir que o projeto permaneça extremo, Ajudar com o que for necessário, Manter a visão do projeto, Formular e comunicar uma tarefa.

f) Consultor: são responsáveis por auxiliar a equipe na resolução de problemas específicos.

g) Gerente: são responsáveis por gerenciar a equipe e seus problemas, agendar as reuniões de planejamento, garantir que as reuniões fluam como planejado, escrever o que foi definido nas reuniões, manter o rastreador informado dos acontecimentos das reuniões e buscar recursos.

3.2 METODOLOGIA DE GERENCIAMENTO

No decorrer deste capítulo, serão abordadas as Metodologias Ágeis, SCRUM, *Adaptive Software Development* (ASD) e *Feature Driven Development*(FDD). Classificadas como metodologias de gerenciamento, tem como foco a aplicação das melhores práticas, apresentando formas de extrair o melhor resultado.

3.2.1 Scrum

O termo SCRUM surgiu a partir de uma tática de jogo de rugby, onde os integrantes se unem de forma a recuperar a bola perdida, alinhando-se e atacando todos ao mesmo tempo. (Takeuchi e Nonaka 2008).

Esta é a representação concreta do que é o SCRUM, trata-se de um trabalho de equipe onde todos caminham rumo ao mesmo objetivo.

Desta forma, não diferente do dia-a-dia de projetos de desenvolvimento de software, todos os integrantes devem trabalhar com o espírito de equipe, devem correr sempre para o mesmo lado e estarem alinhados com cada mudança ou problema apresentado.

3.2.1.1 Equipe *Scrum*

Existem quatro papéis importantes no Scrum, cada um com sua respectiva responsabilidade, sendo eles:

Scrum Master: tem por responsabilidades organizar as reuniões diárias, gerenciar todo o processo do *Scrum*, assegurar que o projeto está sendo desenvolvido de acordo com as práticas do *Scrum*.

Product Owner: é responsável pela tomada de decisão final relacionada ao *product Backlog*, escolha e liberação dos itens que serão *tarbalhados* no *Sprint Backlog* e cálculo de esforço.

Scrum Team: atuam de forma a atingir os objetivos de cada *Sprint*. Composto por no máximo pessoas, tem por responsabilidades determinar a criação do *Sprint Backlog*, Revisar os itens do *Product Backlog*.

Cliente: possui participação ativa na fase de elaboração dos itens que irão compor o *Product Backlog*;

3.2.1.2 Práticas *Scrum*

Scrum é um método ágil que possui como principal objetivo a entrega de um software de qualidade.

Sendo formada por intervalos denominados "*sprints*", que possuem aproximadamente um mês de duração, estabelece um conjunto de regras e práticas gerenciais utilizando métodos como "*product backlog, sprint, sprint backlog*, reunião de planejamento de *sprint*, reuniões diárias do scrum e revisão do *sprint*,".

Não existem restrições quanto ao tamanho e complexidade do projeto, entretanto, podem ocorrer problemas em relação ao gerenciamento das mudanças dos requisitos, equipes de desenvolvimento e trocas de pessoal.

a) *Product Backlog*: Trata-se de tudo aquilo que baseado no conhecimento atual, será necessário no produto final. Sendo o ponto de partida do Scrum, serão definidas as funcionalidades, as prioridades, a tecnologia e as estratégias. Os elementos do "*product backlog*" devem possuir as seguintes informações: Descrição Sucinta, estimativa, um responsável e uma prioridade (muito alta, alta e média).

b) *Sprint*: fase onde são executados os itens definidos no *product backlog*. Tratam-se de reuniões periódicas que tem por objetivo coordenar a realização das atividades existentes, devendo esta durar no máximo 30 dias. Ao final de cada *sprint*, Ao final de cada *sprint*, deve-se entregar uma versão do software funcionando atendendo os requisitos pré-definidos no *sprint*. Na *sprint* existe a integração da parte desenvolvida do software com outras partes implementadas e são feitos testes, sendo possível acompanhar o progresso das atividades. O desenvolvimento dos itens dentro de uma *sprint* não segue nenhum processo pré- definido

(SCHWABER 1995). A utilização de fases tradicionais, como: análise, projeto, implementação, testes e entrega (ABRAHAMSON 2002).

c) *Sprint Backlog*: trata-se de um conjunto de funcionalidades selecionadas do "*product sprint*" para serem implementadas durante o "*sprint*". A partir do momento que todos os itens do *sprint backlog* estiverem prontos, um novo incremento do sistema é entregue. Essas decisões são tomadas em conjunto, para tanto, existe a participação do *scrum team*, do *scrum máster* e o dono da produção.

d) Reunião de planejamento da *sprint*: cada *sprint* é iniciado com uma reunião denominada "reunião de planejamento da *sprint*", que tem por objetivo analisar os ítens do *product backlog*, priorizando e definindo o *sprint backlog*.

e) Reunião diária do scrum: existem reuniões diárias ou em dias alternados, com duração de 15 à 30 minutos cada uma. Esta reunião tem por objetivo a identificação de problemas e para discussões para resolução de obstáculos. Nestas reuniões são levantadas três questões principais para cada membro da equipe (SCHWABER e BEEDLE 2002), sendo elas:
O que foi finalizado desde a última reunião?
Quais as dificuldades encontradas durante o trabalho?
Quais atividades pretendem-se realizar até a próxima reunião?

Além disso, tais reuniões possuem grande importância, tendo se em vista a possibilidade de todas as pessoas fiquem informadas sobre o progresso e as dificuldades levantadas.

f) Revisão da *sprint*: o *scrum master*, juntamente com o *scrum team* apresentam ao cliente no último dia de cada *sprint* o incremento. Assim sendo, os participantes avaliam o incremento e decidem sobre as próximas atividades.

3.2.1.3 Fases Do Processo do SCRUM

As etapas do processo do scrum são compostas por definição dos requisitos, desenvolvimento e entrega final. A figura (7) indica o processo scrum.

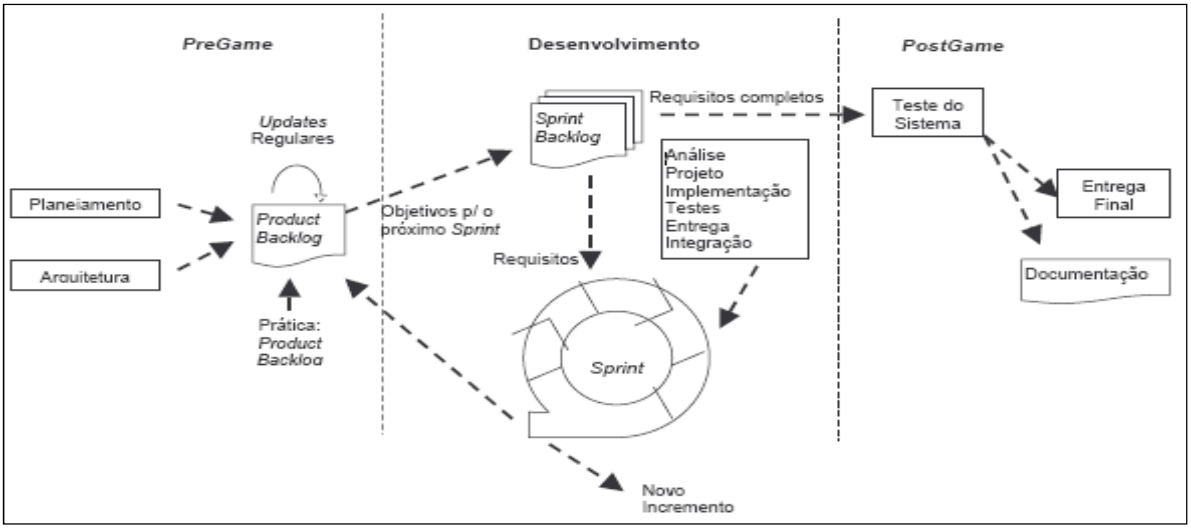

Figura 7 – Etapas do Processo Scrum. FONTE: AUTORES (2009)

As três fases estão representadas por pré *game*, desenvolvimento e pós *game*.

a) Pré Game: esta fase é composta por outras duas sub fases, sendo elas "planejamento e arquitetura".

Planejamento: inclui a definição do sistema que está sendo desenvolvido. Para tanto, o "*product backlog*" deve conter todos os requisitos que são conhecidos até o momento, devendo ser constantemente atualizado. Além disso, deve-se incluir a definição da equipe, ferramentas e outros recursos, avaliação dos riscos e treinamentos necessários. A cada iteração, o *product backlog* atualizado, deve ser revisado pelo *scrum team*.

Arquitetura: baseado nos itens atuais do *Product Backlog* é feito um projeto do sistema. Caso alguma alteração seja efetuada nos itens atuais, são identificadas as mudanças necessárias para implementar os itens do "*product backlog*", juntamente com os possíveis problemas que poderão surgir a partir de tais alterações. Feito isto, uma reunião é realizada para revisar o projeto e reavaliar a proposta, uma vez avaliada, são tomadas decisões.

b) Desenvolvimento: conhecida como *gamephase*, trata-se da fase onde o sistema é desenvolvido através de vários *sprints*, sendo que as funcionalidades são desenvolvidas ou modificadas em ciclos iterativos com a finalidade de produzir novos incrementos. Nesta fase são utilizadas as práticas da *sprint, sprint backlog*, reunião de planejamento da *sprint*, reuniões diárias do scrum e revisão da *sprint*.

c) Post Game: trata-se da entrega final do software, sendo necessário realizar mais testes de sistema e gerar uma breve documentação.

3.2.2 *Feature Driven Development* (FDD)

Feature Driven Development é um dos métodos de gerenciamento de desenvolvimento de software criado a partir das técnicas de modelagem ágil. O termo FDD – *Feature Driven Development* trás em seu significado o desenvolvimento induzido ou guiado por funcionalidades, o FDD é focado em pequenas interações que geralmente duram 2 semanas, ao final ocorre a entrega de uma parte do sistema funcionando (HIGHSMITH 2002). Uma característica é uma funcionalidade definida pelo cliente (PALMER 2002).

Estas características são funções valiosas para o cliente que são agrupadas em uma lista. A FDD fornece maior ênfase em gestão de projeto e qualidade do que outras abordagens (PRESSMAN 2006).

A FDD nasceu entre 1997 e 1998, em Singapura onde um banco internacional solicitou um sistema de empréstimos. Após dois anos de uma determinada consultoria, 3500 páginas de casos de uso e um modelo de objetos com centenas de classes foi avaliado como impossível. Decidiram então implantar as metodologias OOAD (Análise, Desenho e Programação Orientada a Objetos) de Peter Coad e de gerência de projetos de Jeff De Luca.

Em 15 meses, a dupla resultou, 2000 funcionalidades entregues por uma equipe de 50 pessoas.

As funcionalidades definidas pelo cliente devem ser de tamanho suficiente para serem implementadas em no máximo duas semanas, quando passam desse prazo, devem ser divididas em funcionalidades menores, podendo ser pequenas o bastante para serem implementadas em poucas horas (HARTMANN, 2002).

Segundo Higsmith (2002), o FDD afirma que:

- Em um sistema para construir sistema é necessário uma ordem escalar para acompanhar projetos maiores;
- Um processo simples, bem definido, funciona melhor;
- Bons processos vão para um plano de fundo, então o foco são os resultados;
- Ciclos de vida pequenos e iterativos são melhores para o FDD.

3.2.2.1 Práticas do FDD

O FDD, como os demais métodos ágeis, possui um conjunto de práticas que servem para garantir a utilização do método em um projeto de software.

Dentre essas práticas observa-se a gerência de configuração, inspeções e processos que orientam o desenvolvimento.

Como todo processo de desenvolvimento de software, o FDD é construído através de um conjunto de melhores práticas tratadas de maneira diferente das que já existem. Cada qual complementa e reforça outras, podendo a equipe escolher implementar apenas uma ou duas delas, entretanto, não terá com retorno o total benefício que teria usando os processos do FDD (PALMER ; FELSING 2002).

Segundo Palmer e Felsing (2002), as práticas do FDD são:

- Modelagem dos objetos de domínio: construção de diagramas que revelam os objetos relevantes dentro do domínio do problema e seus relacionamentos. O modelo de negócios de domínio fornece uma visão global do qual o sistema será construído, guiando os programadores-chege para um melhor projeto para as funcionalidades a serem implementadas, pois auxilia na manutenção da integridade conceitual do sistema, formando um entendimento global sobre o problema a ser resolvido (HARTMAN 2003).

- Desenvolvimento por características: o desenvolvimento no FDD tem nesta prática a elaboração de uma Lista de funcionalidades que são divididas em grupos de forma hierárquica (feature list), que podem ser indicadas pelo cliente, onde são identificadas as características do sistema através de casos de uso. Esta lista de funcionalidades indicará o caminho de desenvolvimento e mostrará seu progresso conforme a implementação das mesmas. O prazo máximo para a implementação de cada grupo de funcionalidades é de 2 semanas, passando desse prazo as funcionalidades são redistribuídas de forma que possam ser implementadas dentro deste tempo.

- Propriedade de classes de código: há duas idéias principais sobre a propriedade de código, uma é a posse individual do código, outra é posse coletiva. O FDD defende a posse individual do código fonte, que é dividido em pequenos pedaços, onde cada pedaço do código é uma classe. Estas classes devem possuir o menor tamanho possível e são de responsabilidade de um único programador, que se torna o proprietário dela, cuidando de sua manutenção e construção. Assim os proprietários de classes viram especialistas nelas, podendo explicar para a equipe o seu

funcionamento, e ainda podendo fazer atualizações mais rápidas na classe (PALMER ; FELSING 2002).

Algumas desvantagens da propriedade de classes como a necessidade de mudança de classe por um programador que não é o desenvolvedor dela, ter que aguardar o proprietário dessa classe para que haja uma sintonia, isso pode acarretar atrasos. Na necessidade de um proprietário abandonar o projeto e outro desenvolvedor fazer todo o estudo do que já foi desenvolvido para acompanhar o entendimento e dar seqüência àquela classe.

Equipes de características: São as equipes que desenvolvem os grupos de funcionalidades da lista, os desenvolvedores tomam suas decisões conforme foi reportado os problemas pelo grupo. Cada grupo pode ter de 3 a 6 pessoas, dependendo da necessidade daquele grupo de funcionalidades que será designado, podendo ser remanejado caso houver necessidade.

Inspeções: São realizadas inspeções de forma a corrigir falhas durante ou ao término de cada iteração, a fim de eliminar erros e disseminar a cultura de desenvolvimento com convenções de modificações.

Construções Regulares: Construção regular é a realização de que a cada iteração, ou implementação de um grupo de funcionalidades, cria-se um executável, podendo ser documentado. Permitindo a antecipação de erros e sempre haverá uma versão atual para ser apresentada ao cliente.

Administração ou Gerência de configuração: é a documentação de todo histórico de versões, com relatórios de modificações que permite a identificação de quem executou e quando através de um rastreamento.

Relatório de Resultados: O FDD, particularmente forte na área de resultados (PALMER ; FELSING, 2002), tem como característica reportar, em todos os níveis inclusive ao cliente, todos os resultados ocorridos durante o projeto.

3.2.2.2 Os Processos no Fluxo do Método FDD

A figura 8 indica os processos no fluxo FDD. SIQUEIRA(2002).

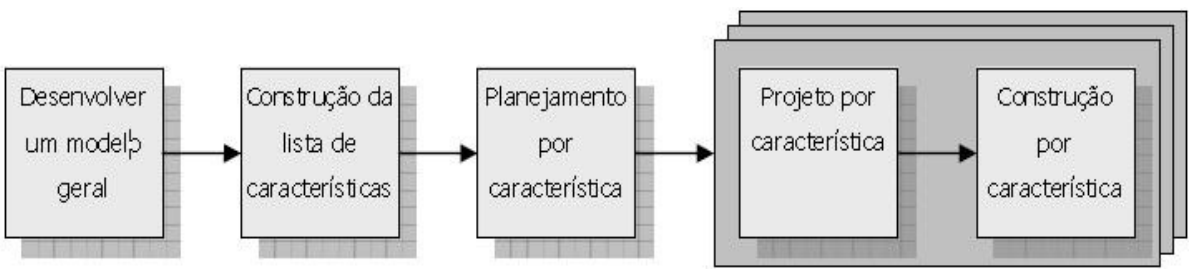

Figura 8 – Processos no Fluxo do Método FDD. Fonte: SIQUEIRA(2002)

Inicia-se o desenvolvimento de um sistema abrangente a partir dos requisitos iniciais do cliente. É elaborada uma lista de funcionalidades (*feature list*), por ela é planejado todo o desenvolvimento. Podemos chamar esses processos de Concepção e Planejamento do fluxo FDD.

A partir daí, há um detalhamento por funcionalidades para sua construção, que será elaborada como pacotes de trabalho dependendo das funcionalidades. Cada pacote de funcionalidades é direcionado uma equipe pequena para realizá-lo, após terem completado as equipes se reúnem para acoplar todos os pacotes desenvolvidos.

Para FDD, *feature* é uma funcionalidade ou característica pequena o suficiente para ser implementada no máximo em uma iteração. Cada funcionalidade tem um valor para o cliente, é definida com uma fórmula simples que permite ser repetível e confiável: ARO
Exemplo:
Ação : Calcular o total de uma venda;
Resultado : Autorizar uma transação com cartão de um cliente;
Objeto: Enviar uma nota fiscal.

3.2.2.3 Características FDD

Fornece a estrutura suficiente para equipes maiores, pois trabalha com módulos de desenvolvimento por funcionalidades. Cada equipe recebe um módulo, que é testado individualmente. Posteriormente eles são adaptados e testados simultaneamente. Isso implica na qualidade de produção de software que gera resultados antecipados tangíveis e funcionais.

Com a elaboração de um sistema abrangente, desde o início o trabalho é significativo antes mesmo de tornar-se iterativo. As informações de estado e progresso são elaboradas de forma clara, simples e compreensível. Todas essas características do método FDD agradam desde os desenvolvedores até o cliente do projeto.

3.2.2.4 Os Cargos

Os cargos são divididos em:

a) Principais:

Gerente de projeto
Arquitetura líder
Gerente de desenvolvimento
Programador líder
Proprietário de classe
Especialistas do domínio
Gerente de domínio
b) De apoio:
Gerente de Versão
Guru de linguagem
Engenheiro de construção
Ferramenteiro
Administrador de Sistemas
c) Adicionais:
Testadores
Instaladores
Escritores técnicos

3.2.3 *Adaptive Software Development* (ASD)

O *Adaptive Software Development* (Desenvolvimento adaptável de software) foi desenvolvido por Jim Highsmith, que se utilizou do RAD (*Radical Software Development*), tomando como princípios métodos iterativo e incremental.

O ASD encoraja o desenvolvimento incremental e iterativo. Fundamentalmente, seu objetivo é prover um processo para auxiliar no desenvolvimento ágil de software. Este é um processo dedicado à aprendizagem contínua, dirigido as mudanças, reavaliações e grande colaboração entre desenvolvedores e clientes (HIGHSMITH, 2002).

3.2.3.1 Práticas do ASD

O ASD não possui um conjunto de práticas definidas e sim um conjunto de propriedades que caracterizam o processo de desenvolvimento adaptativo (HIGHSMITH, 2002).

3.2.3.2 Propriedades do ASD (ABRAHAMSSON, 2002)

Dirigido a missões: para cada iteração do ciclo de desenvolvimento justifica-se através de uma missão, que pode mudar ao longo do projeto;

Baseado em características: desenvolvimento orientado a características, desenvolvimento do software em pequenas partes;

Iterativo: desenvolvimento de pequenos ciclos (iterações), com o objetivo de resultar em uma implementação satisfatória para cada missão definida por iteração.

Prazos pré-determinados: Fixação de prazos para evitar ambigüidade em projetos, com prazos tangíveis que fará com que a equipe do projeto tome decisões que podem representar riscos no seu início;

Tolerância a mudanças: característico do método, as mudanças são freqüentes e a equipe tem de estar pronta para adaptá-las e controlá-las, fazendo uma constante avaliação do que pode mudar;

Orientado a riscos: tudo que é considerado características de alto risco tem seu desenvolvimento priorizado.

3.3 COMPARAÇÃO E ANÁLISE DOS MÉTODOS ÁGEIS

Neste ponto será realizada uma análise em relação às Metodologias Ágeis propostas em todo este capítulo. A partir desta análise, será possível identificar as semelhanças existentes entre os métodos , assim como também, a identificação de práticas e papéis necessários para que estas atividades sejam realizadas de modo a serem incorporadas nas atividades que vão fazer parte do estudo de caso.

3.3.1 Identificação e análise das atividades

Será tomado como base, para a identificação e análise das atividades, o desenvolvimento incremental, uma vez que esta abordagem serviu de “inspiração” para o desenvolvimento ágil.

Os métodos ágeis são fundamentados no desenvolvimento incremental (SOMMERVILLE, 2005). E conforme pode ser observado, as atividades sugeridas durante seu processo de desenvolvimento são bastante semelhantes aos princípios ágeis.

No desenvolvimento incremental, ver figura 9, os clientes inicialmente identificam, em um esboço, os requisitos do sistema e selecionam quais são os mais e os menos importantes. Em seguida é definida uma série de iterações de entrega, onde em cada uma é fornecido um subconjunto de funcionalidades executáveis, dependendo de suas prioridades.

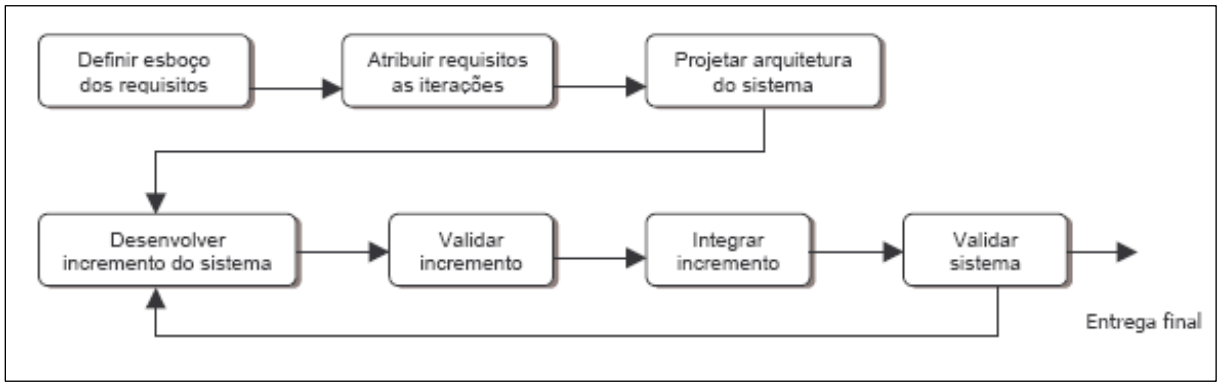

Figura 9 – Atividades do desenvolvimento incremental. Fonte: SOMERVILLE (2005)

Após a identificação dos incrementos, as funcionalidades a serem entregues na primeira iteração são detalhadas e desenvolvidas. Paralelamente a este desenvolvimento, outras funcionalidades podem ser analisadas para fazerem parte dos outros incrementos. Uma vez que cada incremento é concluído e entregue, os clientes podem colocá-lo em operação. O fato dos clientes poderem experimentar o sistema gradualmente facilita o esclarecimento das funcionalidades para os incrementos subseqüentes e à medida que novos incrementos são concluídos, eles são integrados às iterações existentes, de modo que o sistema melhora a cada novo incremento entregue (SOMMERVILLE, 2005).

Por tanto, será apresentada a seguir, um estudo contendo as atividades realizadas pelos métodos ágeis XP, Scrum, FDD, ASD e MA, juntamente com as práticas e papéis adotados pelas mesmas de acordo com as atividades propostas pelo desenvolvimento incremental.

Para cada uma das atividades propostas pelo desenvolvimento incremental será apresentado em uma tabela um resumo da atividade proposta, na qual será explicado as práticas e os papéis associados à atividade para cada um dos métodos ágeis estudados.

3.3.2 Quadro comparativo das metodologias ágeis

Para cada uma das atividades propostas pelo desenvolvimento incremental serão apresentadas as principais características de cada metodologia, definindo assim uma comparação entre as metodologias de desenvolvimento de sistemas e as metodologias de gerenciamento, nas mais variadas fases. Nela é possível verificar pontos fortes e fracos e definir qual metodologia mais se aplica a uma determinada necessidade.

O quadro comparativo das metodologias ágeis (ver APÊNDICE R) compreende cada uma das atividades propostas pelo desenvolvimento incremental, e para cada atividade, se define um resumo da atividade proposta associada a cada um dos métodos ágeis estudados.

Abaixo segue um detalhamento da análise comparativa, que servirá como base para definir quais métodos serão aproveitados nas fases da Engenharia de Requisitos.

Na atividade de definição de esboço dos requisitos (primeira linha do apêndice R), no XP a definição preliminar dos requisitos é feito na fase de exploração, onde são escritas as *user stories* com o cliente. Práticas como metáfora, jogo de planejamento e cliente presente, auxiliam os clientes na definição das *user stories*. No Scrum, a definição do esboço dos requisitos acontece durante a subfase de planejamento durante a fase de pré-game, onde os requisitos conhecidos até o momento são listados dando origem ao *product backlog*. No FDD, os requisitos já conhecidos são listados, definidos e documentados através de casos de uso ou *user stories* durante a fase desenvolver um modelo global que consiste na primeira etapa do processo FDD. A prática desenvolver através de características sugere que além dos artefatos gerados para a documentação dos requisitos sejam construídos um diagrama de classe UML e diagramas de sequência com o objetivo de ajudar na compreensão do projeto. O ASD não apresenta nenhuma prática em relação à definição preliminar dos requisitos, mas durante a fase de início de projeto são relaizadas sessões JAD com a presença de todos os envolvidos no projeto, incluindo os reperesentantes do cliente, com o objetivo de identificar os requisitos definidos até o momento. Finalmente, MA sugere a criação do documento de requisitos para apresentar informações sobre os mesmos. Entre os artefatos propostos para fazerem parte do documento de requisitos estão: modelos contendo as regras de negócio, casos de uso, user stories, e protótipos.

Na atividade de atribuição dos requisitos as iterações (segunda linha do Apêndice R), no XP é feita a fase de planejamento, durante a adoção da prática jogo de planejamento, onde acontece a definição das user stories que serão implementadas durante cada iteração. Também

é adotada a prática de cliente presente, durante a escolha das *user stories* que serão implementadas durante a iteração. No Scrum, os requisitos que serão desenvolvidos em cada *sprint* são definidos a partir da lista de requisitos contida no *product backlog*. Estes requisitos são selecionados durante a reunião de planejamento da *sprint* de acordo com sua prioridade, dando origem ao *sprint backlog*, que consiste numa lista de requisitos que será desenvolvida em cada *Sprint* (iteração). Aqui são analisados os requisitos selecionados por *sprint* e a equipe scrum determina a complexidade, e calcula tempo e esforço para os itens selecionados. No FDD, esta atividade é definida durante a fase construir uma lista de características, onde as características (requisitos) são agrupadas e priorizadas de acordo com a importância e dependência dando origem aos conjuntos de características que serão desenvolvidos durante as iterações. E também durante a fase planejar a construção por características, onde é feito o planejamento para o desenvolvimento destes requisitos, como também são atribuídos os responsáveis por cada uma das características. Durante a subfase de planejamento do ciclo adaptável na fase de especulação de ASD, são definidas as quantidades de iterações e os requisitos que serão implementados em cada uma delas. A prática de prazos pré-fixados é adotada para garantir que o tempo definido para cada iteração deve ser respeitado e cumprido. A MA por não apresentar um modelo de desenvolvimento para o desenvolvimento de software, não tem nenhum aporte relevante a este respeito.

Na atividade de projeto da arquitetura do sistema (terceira linha do Apêndice R), o XP propõe que paralelamente à escrita das *user stories*, seja realizado o projeto de arquitetura de sistema. XP adota a prática de projeto simples, que define que o projeto tem que ser o mais simples possível e refinado durante todo o processo de desenvolvimento de software. Na subfase de arquitetura do pré-game do scrum é feito um projeto geral do sistema baseado nos itens do *product backlog*. Este projeto de arquitetura do sistema tem a função essencial de ajudar a atingir os objetivos de cada *Sprint*. O FDD sugere a través da prática Modelagem dos objetos de domínio que seja construído um diagrama de classes da UML para representar a arquitetura do sistema durante a fase desenvolver um modelo global. Podem também ser complementados com diagramas de seqüência da UML. O método ASD não sugere nenhuma atividade. Finalmente, o MA sugere a elaboração de um documento chamado Documento do Sistema, que representa uma visão geral da arquitetura da técnica e da arquitetura de negócio do sistema.

Na atividade de desenvolver o incremento do sistema (quarta linha do Apêndice R), no XP a atividade de desenvolvimento dos requisitos durante as iterações é realizada na fase

de iterações para Entrega, onde sugere para cada conjunto de user stories selecionadas para fazerem parte da iteração sejam realizadas as atividades de modelagem e programação. Para suportar estas atividades XP utiliza as práticas, tais como, padrões de codificação, programação em pares, propriedade coletiva, refatoração, cliente presente e entregas freqüentes. No Scrum está fase acontece durante a execução do *sprint*, onde são implementados os requisitos contemplados no *sprint backlog*. Scrum não segue nenhum procedimento pré-definido para a execução desta atividade, só as práticas já definidas dentro do método, tais como, reuniões diárias. Já no FDD o desenvolvimento dos conjuntos de características acontece durante as fases projetar e construir cada característica, onde acontecem as atividades de análise da documentação existente, geração de diagramas de seqüência para o conjunto de características, refinamento do modelo elaborado durante as etapas de definição de requisitos e projeto de arquitetura de sistema e finalmente da implementação das classes e métodos correspondentes às características que serão desenvolvidas em cada uma das iterações. Durante a fase de colaboração do ASD, os requisitos que fazem parte de cada uma das iterações são desenvolvidos. No ASD o desenvolvimento das diferentes iterações pode ser realizado simultaneamente, dado que seu foco é o resultado final. MA sugere alguns documentos a serem gerados durante e ao final desta atividade. Os documentos são: documentação de operações e um documento contendo as decisões de projeto.

Na atividade de validar o incremento (quinta linha do Apêndice R), no XP a fase de validação de requisitos acontece na fase de Iterações para a Entrega, a través da execução de testes de aceitação e dos testes de unidade. A prática cliente presente coloca como que o mesmo deve estar sempre presente e disponível para escrever e realizar os testes de aceitação. Já no Scrum esta fase de validação de requisitos é realizada ao final de cada *Sprint*. Scrum não adota nenhum processo pré-definido para está etapa. Podendo ser utilizados processos de validação de outros métodos disponíveis. No FDD acontecem ao final de cada iteração na fase construir cada característica, onde o conjunto de características implementado é testado pelos responsáveis de cada classe. FDD adota práticas de inspeções para garantir a qualidade do desenvolvimento. Por outro lado, ASD não adota nenhuma prática relacionada à validação dos requisitos, mas coloca que a validação dos requisitos desenvolvidos durante a fase de colaboração acontece na subfase revisões de qualidade na fase de aprendizado, onde a qualidade funcional do sistema gerado na iteração é verificada através da definição de grupos de clientes para revisar e testar a aplicação. Também a qualidade técnica é verificada.

Finalmente, MA não sugere a geração de documentos relacionados a atividades de testes de validação.

Na atividade de integrar o incremento (sexta linha do Apêndice R), no XP a atividade de integração acontece ao longo da fase de iterações para a entrega através da prática integração contínua, que sugere que as equipes devem manter o sistema integrado todo o tempo. Isto é, à medida que o código vai sendo gerado ele vai sendo integrado. A integração continua permite descobrir problemas de compatibilidade cedo. A integração do resultado de cada uma das *sprints* acontece ao final de cada uma delas na fase de desenvolvimento. O scrum não sugere nenhuma prática em relação à atividade de integração, também nem como devem ser realizados, podendo ser utilizado outros processos de métodos disponíveis. No FDD a integração do conjunto de características implementado durante a iteração corrente com os outros, acontece ao final de cada uma das iterações durante a fase de Construir cada Característica, depois dos testes. Tanto ASD como MA não sugerem nenhuma atividade específica para integração.

Na atividade de validar sistema (sétima linha do Apêndice R), XP sugere que as integrações devem acontecer paralelamente às implementações das *user stories* e conseqüentemente aos testes, porém, o mesmo não sugere práticas, papéis, nem atividades específicas de validação depois dos incrementos integrados. XP recomenda que a fase de produção seja posta em prática dentro da organização, onde será avaliado e poderá sofrer alterações de acordo com as observações levantadas pelos clientes. O scrum valida a integração dos incrementos, ao final de cada *Sprint* durante a fase de desenvolvimento através da prática revisão da *sprint*, onde são revisados os incrementos e se decide pelas atividades seguintes. No FDD a validação é feita na fase construir cada característica, através das inspeções e dos testes de integração realizados pelas pessoas responsáveis. O método não sugere nenhuma prática com respeito à validação dos incrementos integrados. Tanto ASD como MA não sugerem nenhuma atividade específica para integração.

Na atividade de entrega final (última linha do Apêndice R), os métodos não sugerem nenhuma prática nem papéis para a atividade de entrega final, com exceção de MA que propõe a elaboração dos documentos de operações, de suporte e finalmente o documento de usuários.

Como pôde ser observada, a análise realizada possibilitou a identificação de cada uma das atividades propostas pelos métodos ágeis, além de permitir a identificação dos seus pontos comuns. No capítulo 5, estudo de caso, é tomada a decisão sob qual ou quais métodos

serão selecionados para sua aplicação dentro dos processos da engenharia de requisitos e assim pode-se aplicar as melhores práticas e princípios da filosofia ágil baseando-se nesses métodos.

4 TRABALHOS RELACIONADOS

Neste capítulo se apresenta uma relação de trabalhos relacionados que serviram de embasamento ao desenvolvimento do trabalho de conclusão de curso. Ao longo das pesquisas, foi possível detectar uma quantidade relativamente grande de artigos abordando o tema de ER, Modelagem e Métodos Ágeis. Entretanto, de sua totalidade somente alguns poderiam ser utilizados, visto à idoneidade do conteúdo apresentado.

A atividade iniciou-se com a monografia de pós-graduação “Framework para comparação e análise de métodos ágeis” (FAGUNDES, 2005), que tem por objetivo propor um framework para os métodos ágeis.

Neste trabalho mostra como, à medida que os métodos ágeis são estudados, é possível perceber a grande semelhança existente entre eles e também suas diferenças. Por este motivo, esta monografia propõe a criação de um framework, apresentando as atividades sugeridas por cada um dos métodos, de forma a serem utilizadas dependendo da necessidade do desenvolvedor.

Sendo assim, o desenvolvedor não precisará conhecer todos os métodos e sim verificar qual deles adéqua-se à sua necessidade e utilizá-lo.

A comparação dos métodos baseia-se no ciclo do desenvolvimento incremental. Através do estudo fase por fase são feitas comparações entre as metodologias, apresentando o que cada uma realiza para determinada fase do desenvolvimento incremental. Este comparativo permite esclarecer como se desenvolve determinada metodologia dentro dos processos de desenvolvimento de software principalmente, considerando vantagens e desvantagens e selecionando as melhores práticas para uma determinada fase que momento depois é aplicado num estudo de caso.

Segundo o autor, os métodos mais completos e utilizados, que atendem à maior parte das fases do desenvolvimento incremental são o extreme programming e o Scrum. Com focos diferentes, o XP apresenta uma filosofia voltada ao desenvolvimento. Já o Scrum foca sua filosofia no gerenciamento como forma de agilizar a entrega do produto final.

Como aportes para o trabalho de conclusão e curso podem realçar a metodologia utilizada para realizar a análise comparativa e como são listadas, de forma organizada e consistente, as atividades sugeridas pelos métodos ágeis estudados.

O artigo “Projeto e implementação de uma ferramenta para gerência de requisitos em metodologias ágeis” (GONÇALVES, 2007), indica a importância do “manifesto ágil”, esclarece que documentação, processos e ferramentas têm importância secundária quando comparado com a importância dos valores do manifesto.

Em base à análise comparativa define-se a importância de cada um dos métodos analisados e propõe um método “híbrido”, o qual tem seu foco principal no método FDD, devido à capacidade gerencial satisfatória dos seus processos. É perfeitamente adequada para equipes pequenas, além de ter um ciclo de vida focado no desenvolvimento do negócio. Junto com as práticas e papéis do FDD, a metodologia é complementada com XP, com as *user stories*, e Scrum com suas práticas gerenciais, tais como, *backlogs*, *sprints*, reuniões e retrospectivas. Aplicar três metodologias ágeis parece um pouco redundante, visto que o FDD e Scrum, segundo explicação neste trabalho, são métodos gerenciais semelhantes e a nosso entender seria preciso de só um deles para complementar o XP. Além de serem estes dois métodos, os mais conhecidos do mercado atual.

Adaptações dos artefatos típicos das metodologias utilizadas, como por exemplo, *user story* de XP, deixam como ensinamento a possibilidade de fazer o mesmo tipo de adaptações em artefatos, ou mesmo em alguma fase de determinada metodologia dentro da execução do estudo de caso, isto permitirá cobrir de uma melhor forma todos os processos da ER.

Para Elias Müller em seu trabalho de conclusão de curso “Métodos ágeis: uma aplicação de scrum no simuplan” considera a análise comparativa dos métodos fundamental para poder alcançar o objetivo do trabalho. Para isto, foi realizada uma série de comparações de forma muito detalhada, que tiveram com resultado encontrar os pontos fortes e fracos, vantagens e desvantagens dos métodos que estão sendo estudados neste trabalho, tais como: XP, Scrum, FDD e ASD.

Como resultado final foi selecionado o método Scrum para a aplicação no Projeto Simuplan. Scrum a pesar de ser o método com segunda melhor pontuação no resultado da análise comparativa, se decidiu na sua escolha por que busca organizar da melhor forma os projetos e assim obter um maior controle dos mesmos. Para o autor, métodos ágeis são mais indicados para processos com mudanças contínuos e com requisitos incertos e voláteis, ou seja, que podem estar em constante mudança. Na prática define-se um passo a passo de como é aplicado o método Scrum dentro do projeto, compartilhando o detalhe de sua aplicação e os resultados obtidos. Esta última abordagem é muito importante para o andamento do trabalho

de conclusão de curso por que este exemplo prático nós ajudará a entender de uma melhor forma como os métodos ágeis são aplicados e permitirá aproveitar dita prática.

Outro trabalho relacionado importante é "Engenharia de Requisitos Ágil" (SANTANA, 2008), neste trabalho se compara o método tradicional, em este caso RUP, com a filosofia da abordagem ágil. Grande importância é dada às vantagens da filosofia ágil, tais como, comunicação verbal, ênfase na parte teórica, isto é, utiliza os valores do manifesto ágil aplicados a ER e não um método ágil definidos neste trabalho.

Importante é que este trabalho define como são aplicados estes conceitos, muito importante para o entendimento da filosofia ágil, como por exemplo, definindo o escopo do produto e a partir dele estabelecer *baselines* (pontos de controle) que nos permita controlar as mudanças. Também define o documento de requisitos como um contrato entre cliente e desenvolvedores que permite fixar o limite do sistema. A priorização dos requisitos para poder implementar desde os pontos mais relevantes e assim no sucessivo, servindo também para definir os entregáveis para o cliente. Adaptação às mudanças através de refatoração minimizando custo de mudanças, também permite flexibilidade ao alterar o escopo. O controle de mudanças através de documentos complementares ou feitos no mesmo código pode dificultar o mapeamento dos requisitos.

Na palestra "Scrum: Construindo o sucesso em um projeto para a Petrobrás" ministrada na Segunda Conferência Latino-Americana sobre Metodologias Ágeis foi comentado que o sistema web para gerenciamento do processo de licenciamento ambiental da Petrobrás, apresentava problemas como a falta de coordenação e de metodologia, sendo agravado com o crescente número de usuários que solicitavam novas funcionalidades e correções de *bugs*.

Rafael Sabbagh (Scrum *Master*) e Marcos Garrido (*Product Owner*) reergueram o projeto, utilizando a abordagem do scrum, efetuando entregas freqüentes a fim de mostrar ao cliente que o sistema estava ativo novamente.

Desta forma, é possível observar o impacto positivo causado pela adoção do scrum. Com relação ao trabalho de conclusão de curso, a possibilidade de acompanhar casos práticos aplicando scrum, auxiliou a equipe em tomadas de decisões durante a execução dos *sprints*. Casos como este são comuns no ciclo de desenvolvimento de software. Levando-se em consideração a engenharia de requisitos, é possível encontrar cenários como este já em produção, sendo necessário trabalhar na gestão do documento de requisitos.

5 ESTUDO DE CASO

Neste capítulo será apresentado um estudo de caso com o objetivo de identificar os processos e os artefatos da ER no contexto das filosofias ágeis. O estudo de caso foi realizado a partir das necessidades detectadas na Empresa de telefonia (ETel).

5.1 O PROBLEMA

Esta necessidade está baseada na problemática atual que acontece em muitas empresas, centros de estudo, dentre outros. Consiste na reserva de equipamentos e salas, um assunto aparentemente simples, mas que ocasiona muito desconforto e problema.

O problema concentra-se na falta de gestão dos equipamentos e salas de reunião disponíveis na (ETel). Atualmente, este tipo de gestão é feito de forma manual, o que ocasiona incômodo entre os funcionários. Sendo assim, os funcionários são obrigados a irem até a sala onde deseja fazer a reunião, anotar seu nome e horário de reserva, caso tenha horário disponível ou dirigir-se para outra sala mediante indisponibilidade, e assim sucessivamente até encontrar alguma disponível, sendo que atualmente o número de salas de reuniões diminuiu para somente quatro.

No caso de equipamentos, o problema principal encontrado foi na reserva do equipamento retroprojetor, pelo fato de não se possuir nenhum controle, pode ser requerido por qualquer pessoa. Isto faz com que no momento da reunião, o equipamento possa não estar disponível. Importante mencionar também que um 50% dos funcionários possui notebook, o que gera problemas devido a que qualquer um deles pode pegar o equipamento dado a facilidade de utilização em conjunto.

A gerência de consultoria propõe aliviar estes problemas, para isto determinou a automatização do processo de reserva de equipamento e salas. A preferência adotada para o desenvolvimento da aplicação é C Sharp e o banco de dados Access. Inicialmente irá suportar a reserva de equipamentos e salas de reunião, mas espera-se que seja o primeiro passo para automatizar vários dos processos manuais da empresa e também será um piloto para migrar a aplicação para ambiente web.

5.2 PLANEJAMENTO

Para poder atingir o objetivo deste trabalho de conclusão de curso foi decidido, em primeiro lugar, estabelecer uma estratégia para a abordagem do estudo de caso, para isto, contará com três etapas principais: exploração, execução, onde ocorrem os *sprints*, e produção.

Para poder abordar estes problemas e baseados em nossa análise comparativa decidiu-se utilizar as práticas de dois métodos ágeis previamente estudados, *Extreme Programming* (XP) e Scrum para adaptá-los aos processos da ER. Estes dois métodos foram escolhidos por serem os mais utilizados no mercado, e o intuito é mostrar aos analistas como acontecem os processos de ER nestas metodologias. A figura 10 reproduz graficamente o ciclo dos métodos XP e Scrum, e como serão utilizados.

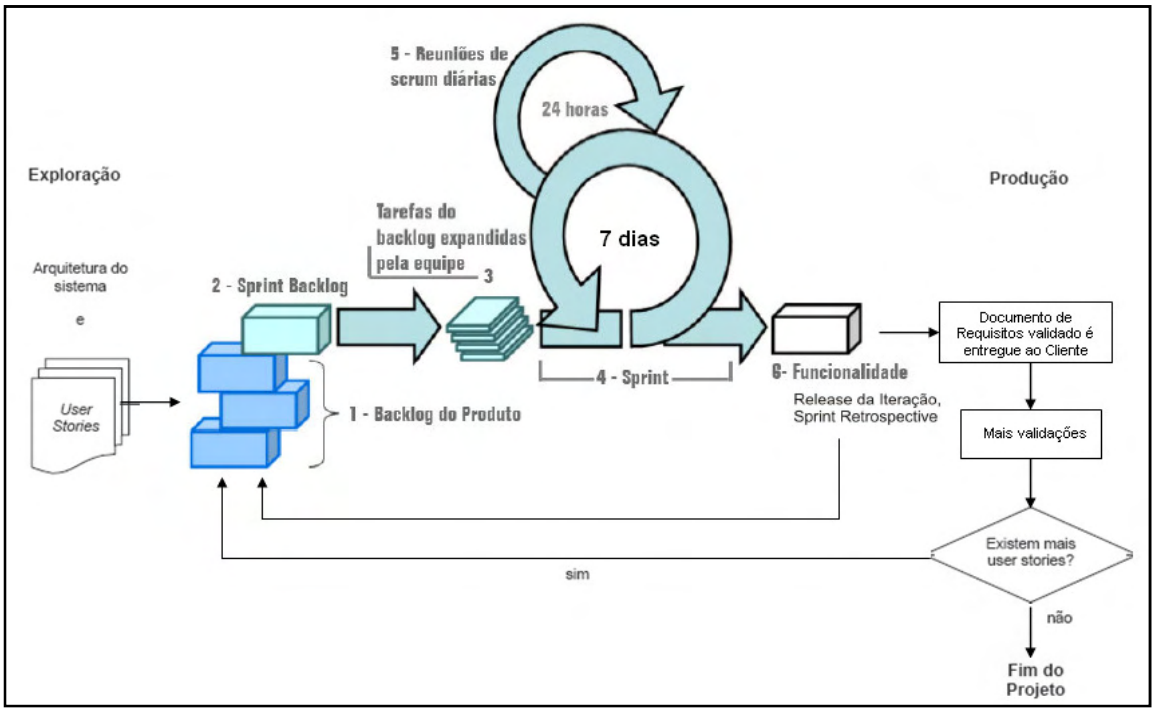

Figura 10 – Método Extreme Programming e Scrum unificados. FONTE: AUTORES (2009)

A etapa de exploração, extraída do método XP, corresponde dentro do processo de desenvolvimento de requisitos, o equivalente à atividade de levantamento de requisitos. Esta etapa permitirá, em conjunto com o cliente, listar e complementar os requisitos necessários para poder atender as necessidades do cliente. Os requisitos serão previamente definidos no formato de *user story*, ver tabela 01. Não será necessária criação de grande quantidade de *user*

stories para que possa dar início à etapa de execução. Basicamente, em cada *user story* será detalhado cada requisito identificado durante este processo, dando um código de identificação a cada requisito, definindo uma breve descrição, que deverão ser detalhados durante os *sprints*. Também se identificarão as tarefas a ser desenvolvidas para cumprir com cada requisito, a pessoa responsável para atender determinado requisito e, muito importante, os testes de aceitação que permitirão aprovar ou não o requisito definido na *user story*.

O modelo de *user story* deverá ter uma seção onde será registrado o histórico de alterações, a razão desta adaptação, deve-se há um melhor acompanhamento nos requisitos no momento de realizar mudanças, que podem acontecer tanto na retrospectiva de *sprint*, assim como também, no processo de gestão de requisitos.

Tabela 1 – Modelo *User Story* para Estudo de Caso. FONTE: AUTORES (2009)

Número de US: US01
Descrição:
Regras de Negócio: RN1.1
Testes de Aceitação: TA1.1
Estimativa Inicial:
Histórico de Alterações: HA1.1

Os pontos 1 ao 6 da figura 10, são compostos pelas fases do SCRUM denominadas pré-*sprint*, *sprint* e pós-*sprint* do scrum. Estas fases equivalem aos processos de Análise, Documentação e Validação de requisitos, adaptando-as conforme processos listados abaixo.

Os pontos 1, 2 e 3 da figura 10 formam a fase de pré-*sprint*. Nesta fase, a partir das *user stories* levantadas entre o *product owner* e o coordenador da empresa de telefonia, é gerado o *product backlog*, composto pelos requisitos que deverão ser atendidos pelo sistema. Definido o *product backlog*, o scrum *master* em conjunto com o *product owner*, devem definir quais os requisitos de maior complexidade e de maior urgência por parte do usuário.

Definida esta lista, separam-se as de maior importância formando o *sprint backlog*. Com o início do *sprint backlog*, o scrum *master* distribui as tarefas entre os membros da equipe, verificando quem possui maior conhecimento em cada tarefa.

Os pontos 4 ao 5 representam a fase de *sprint* e correspondem aos processos de análise e documentação de requisitos. O *sprint*, no scrum, representa o ciclo de desenvolvimento, onde serão realizadas as tarefas especificadas no *sprint backlog* e terá duração de sete dias.

No decorrer deste período, serão realizadas reuniões diárias que permitirão o debate entre os membros da equipe sobre o andamento do projeto. Nesta fase, também é gerando e detalhado em conjunto com o cliente o documento de requisitos, sendo considerado um dos entregáveis.

No ponto 6 da figura, pós-*Sprint* é onde será apresentado por parte do *Scrum Master* ao cliente, usuário final e responsáveis, o produto da iteração, que será o documento de requisitos, para ser validado por meio de prototipação e revisão de requisitos.

Durante o ciclo do Scrum atividades são trabalhadas em conjunto, dentre elas, podemos destacar o levantamento, a análise e a validação de requisitos. Para o processo de validação, com a criação de protótipos no decorrer do ciclo, é possível estabelecer em conjunto com o Líder de Consultoria e Implantação da empresa de telecomunicação, o *product owner* e o scrum *master*, melhorias ou eventuais correções de maus entendimentos. Além desta validação, é feita uma revisão dos requisitos. Este tipo de validação torna-se mais minuciosa, detectando falhas logo no início do projeto, evitando assim desgastes futuros e maiores gastos.

Esta validação servirá de apoio para a qualidade no desenvolvimento da documentação e conseqüentemente do software. A validação, por meio de prototipação e revisão de requisitos, possibilitará o desenvolvimento do formulário de validação, conforme figura abaixo:

Tabela 2 - Formulário de Validação de Requisitos. AUTORES(2009)

Formulário de Validação				
Doc. De Requisitos (Nome, Versão): Nome do documento de requisitos e qual a versão que se encontra				
Elaborador (Papel, Nome, E-mail, Ramal): Papel, nome, e-mail e ramal do elaborador do formulário de validação				
Revisor (Papel, Nome, E-mail, Ramal): Papel, nome, e-mail e ramal do revisor do formulário de validação				
Data Revisão (dd/mm/aaaa): Dia, Mês e Ano		**Tempo revisão (Horas):** Quantidade de horas para efetuar revisão		
Desconformidades encontrada				
Número	**Página**	**Linha**	**Requisito**	**Descrição**
Número: Regra de Negócio/ Teste Aceitação	Número	Número	Título do Requisito	Breve descrição
Observações Gerais dos revisores				
Observações gerais a serem inseridaspelo revisor.				

Este formulário fará o registro de desconformidades apresentadas durante o processo de validação e servirá de apoio na definição de uma possível retrospectiva. Esta retrospectiva ou *sprint retrospective*, conforme definido no scrum, ocorre nesta fase. Este *sprint* ocorre mediante correções ou necessidades apresentadas como resultado do *sprint* e do resultado do processo de validação de requisitos.

Para tanto, o documento de requisitos, depois de validado, poderá não ter satisfeito alguma necessidade do cliente, o requisito ou requisitos poderão sofrer alterações no seu alcance e devem voltar para ser inclusos no *product backlog*, ou também aparecer novos requisitos, recomeçando o processo de ER novamente.

Esta etapa poderá ocorrer quantas vezes for necessária, levando-se em consideração a demanda da quantidade de requisitos, definição do *product backlog*, mudanças que apareçam através da execução dos *sprint*s (retrospectiva) e ao processo de gestão de requisitos.

Como resultado de cada *sprint* teremos os seguintes artefatos, considerados entregáveis do estudo de caso para o processo de desenvolvimento de requisitos:

User stories;

Documento de requisitos;

Funcionalidades destinadas aos *sprint*s (Executáveis);

Uma vez entregue estas documentações, passa-se ao processo de gestão de requisitos, que será atendido na etapa de produção.

Nesta etapa, serão feitas adaptações do método XP, onde, baseados nos resultados de cada um dos *sprints* serão realizados, pelo cliente e/ou equipe de projeto, novas revisões, validações e controles de qualidade sobre o documento de requisitos, podendo ter como resultado alterações ou criação de novos requisitos. Importante indicar que tais alterações serão solicitadas através do *Change Request Form*. Conhecido como CRF (*Change Request Form*), sua utilização implica na mudança ou inclusão de funcionalidades vistas como necessárias em um dado momento. Para sua utilização, existe um processo de aprovação.

Este ciclo inicia-se com um comitê, onde serão debatidos itens como sua viabilidade, incluindo custo, prazo e grau de dificuldade, sendo registradas nas *user stories*, controladas através da seção de histórico de alterações.

Desta forma, é possível verificar que na estrutura planejada a fase de exploração é representada pelo método XP, a fase de execução é representada pelo método SCRUM e no final a fase de produção representada pelo método XP.

5.3 EXECUÇÃO DA ESTRATÉGIA

O início da execução de estratégia definida para alcançar o objetivo deste trabalho de conclusão de curso, teve início no dia vinte e um de setembro de dois mil e nove (21/09/2009), sendo apresentados os métodos a serem aplicados aos responsáveis na (ETel).

Nesta data foi dada uma breve explanação da filosofia ágil, os métodos SCRUM e XP e como a Engenharia de Requisitos poderia ser aplicada nesta filosofia.

A continuação foi apresentada a equipe e responsáveis para o projeto. O responsável por ser o *Scrum Master* foi o Sr. Eduardo Miyake, o *Product Owner* foi atribuído ao Sr. César Sanz e o time Scrum ficou composto pelos Sr. Fabio Lima e Sr. Nick Lazur. Por parte da ETel ficou como responsável o líder de consultoria e implantação.

Uma vez dado o *kick-off* do projeto, iniciou-se as etapas da estratégia já definidas no capítulo de planejamento.

5.3.1 Exploração

Uma vez definida e apresentada a equipe de projeto, continuamos com as reuniões entre scrum máster, *product owner* e o responsável do projeto por parte da ETel, nestas

reuniões foram explicadas a estratégia de trabalho, onde destaca a metodologia que utilizaríamos para esta primeira fase de projeto.

Considerando que é utilizado RUP como metodologia dentro do cliente, foi acordado utilizar a ER na metodologia ágil. Depois da aprovação do método, começamos com o processo de levantamento de requisitos, a técnica utilizada foi por meio de entrevistas (*face-to-face*). Na primeira reunião foram definidas já as principais necessidades onde o cliente tinha mais carência. Durante este processo de levantamento, cada requisito identificado foi preenchido numa *user story*, que será um de nossos entregáveis ao final de cada *sprint* e será utilizado como insumo para a elaboração do documento de requisitos e também para controlar a gestão das mudanças nos requisitos.

Como resultado deste levantamento se concluiu que a seguinte lista de requisitos e suas respectivas funcionalidades devem ser atendidas e priorizadas pelo sistema a ser desenvolvido, importante realçar que devido a este processo de ER estar sendo definidos no contexto das filosofias ágeis, os requisitos identificados neste primeiro levantamento poderá sofrer alterações ao longo da execução de este estudo de caso:

Lista de Requisitos:

Manutenção de salas de reunião (Apêndice A);

Manutenção de equipamento (Apêndice B);

Solicitação de salas de reunião (Apêndice C);

Solicitação de equipamentos (Apêndice D);

Manutenção de funcionário (Apêndice E);

Relatório de reserva de sala por semana (Apêndice F);

Esta lista de requisitos servirá de base para a elaboração do *product backlog* para o sistema de reservas de equipamentos e salas da ETel. Importante também mencionar que a lista de requisitos, para este momento do processo, não necessita estar listada em ordem de prioridade, esta priorização corresponde à etapa de execução.

Esta etapa de levantamento por meio de *user stories* é continuamente elaborada pelo scrum *master* em conjunto com o *product owner*, que disponibiliza os mesmos para o detalhamento de cada funcionalidade do *product backlog*.

A continuação e conforme definido, segue cada requisito com sua *user story* associada (ver tabelas nos apêndices), num primeiro momento só será detalhado o requisito, deixando para fase de execução o complemento de preenchimento de cada uma das *user stories* definidas neste estudo de caso.

Todas as *user stories* serão detalhadas no decorrer da fase de execução, conforme as reuniões aplicadas para o detalhamento de cada funcionalidade. Neste detalhamento serão definidas as tarefas e testes de aceitação para cada requisito.

5.3.2 Execução

Na fase de Execução, é elaborada uma reunião entre o Scrum *Master* e o Scrum *Team* para a distribuição de tarefas, sendo detalhadas aos integrantes suas respectivas funções. Concluída esta etapa, começa então o ciclo de desenvolvimento baseado nos *sprints*.

Nesta fase, serão utilizadas as etapas dos ciclos do Scrum com a finalidade de se fazer somente o necessário, ou seja, o que é mais importante para alcançar o objetivo e também gerar resultados ao cliente.

5.3.2.1 Primeiro *Sprint*

O primeiro *sprint* teve início no dia vinte e oito de setembro de dois mil e nove (28/09/2009), com o levantamento de requisitos utilizando as *user stories*. No decorrer do processo de levantamento de requisitos o *product owner* iniciou a seleção dos requisitos que, divididos por tarefas, fariam parte do primeiro *sprint backlog*, sendo gerada a tabela abaixo:

Tabela 3 - Requisitos levantados e priorizados. AUTORES(2009)

Prioridade	Item	Descrição	Tempo (horas)	Responsável
Muito Alta	1	Módulo de Solicitação de Sala		Fábio Lima
	2	Módulo de solicitação de equipamentos		Nick Lazur
	3	Cadastro de Equipamentos		Eduardo Miyake
	4	Cadastro de Salas de Reunião		Cesar Sanz
Alta	5	Cadastro de Funcionários		Eduardo Miyake
	6	Módulo de Relatórios		Cesar Sanz

Os requisitos foram priorizados de acordo com as necessidades apresentadas pelo cliente, sendo classificados como muito alta, alta e baixa. Os requisitos com prioridade “muito alta” são aqueles que possuem grande influência no sistema e um alto valor de negócio, demandando maior tempo para desenvolvimento. As demais foram caracterizadas como alta, pela complexidade e pelo valor de negócio e baixa por não agregarem tanto valor de negócio e não possuir urgência.

Antes do início das atividades, o Scrum *Master* convocou uma reunião com o Scrum *Team* para apresentar informações sobre as tarefas, o prazo de duração do *sprint* (sete dias) e a existência de reuniões diárias.

Para o primeiro *Sprint*, foram selecionados os requisitos com prioridade muito alta, conforme tabela abaixo:

Tabela 4 – *Sprint Backlog* – *Sprint* 1. FONTE: AUTORES(2009)

Prioridade	Descrição	Tarefas	Tempo Estimado (h)	Tempo de Conclusão (h)	Responsável	Status
Muito alta	Manutenção de salas de reunião	Cadastrar sala de reunião	4hrs	5h e 15 minutos	Fábio Lima	Concluído
		Alterar sala de reunião	4hrs	5h e 25 minutos		Concluído
		Editar sala de reunião	4hrs	3h e 20 minutos		Concluído
		Excluir sala de reunião	4hrs	2h		Concluído
	Manutenção de equipamentos	Cadastrar equipamento	4hrs	5h	Nick Lazur	Concluído
		Alterar equipamento	4hrs	3h e 15 minutos		Concluído
		Editar equipamento	4hrs	4h		Concluído
		Excluir equipamento	4hrs	3h e 45 minutos		Concluído
	Solicitação de salas de reunião	Solicitar sala de reunião	4hrs	4h e 35 minutos	Eduardo Miyake	Concluído
		Editar solicitação de sala de reunião	4hrs	4h		Concluído
		Cancelar solicitação de sala de reunião	4hrs	3h e 25 minutos		Concluído
	Solicitação de equipamentos	Solicitar equipamento	4hrs	4h	Cesar Sanz	Concluído
		Editar solicitação de equipamento	4hrs	4h		Concluído
		Cancelar solicitação de equipamento	4hrs	4h		Concluído

Os requisitos da tabela acima foram detalhados pela prioridade, descrição, tarefas, tempo estimado, tempo de conclusão, responsável e status.

Este detalhamento pode ser mais bem visto nas *user stories* associadas (Ver APÊNDICES - G ao J)

Baseando-se nas *user stories* detalhadas, será possível desenvolver o documento de requisitos composto pelo detalhamento das tarefas. Para o desenvolvimento do documento, foi necessário levantar e ter bem claro qual o objetivo, o âmbito, as funções do produto e os requisitos específicos. Levantadas estas informações, foram definidos os requisitos funcionais e as regras de negócio, auxiliando assim na elaboração do protótipo (APÊNDICE K). Uma vez definido o documento de requisitos, iniciou-se, em conjunto com o responsável do projeto da empresa de telefonia, a validação do documento elaborado. Em paralelo também foi desenvolvida a primeira parte do sistema, sendo também validado pelo cliente.

Como resultado do processo de validação, conforme descrita na seção de planejamento, através do formulário de validação, foi detectada a necessidade de incorporar as seguintes regras de negócio e funcionalidades:

a) Regras de negócio

Reserva de uma única sala pode ser feita somente por um funcionário para o mesmo horário.

Reserva será permitida somente se houver uma semana de antecedência à data em que se está realizando a reserva atual, apresentando uma mensagem de restrição caso exceda o tempo.

No momento da escolha do equipamento deverá ser apresentada uma mensagem de estoque esgotado caso não haja equipamento disponível.

b) Novos requisitos

Relatório de reserva de sala por funcionário.

Relatório de reserva de equipamento por funcionário.

Com estes resultados encerrou-se o primeiro *sprint* tendo como entregáveis as *user stories* devidamente preenchidas, o documento de requisitos do primeiro *sprint*, já validado pelo cliente, e com melhorias já detectadas que serão incluídas no segundo *sprint*.

Além desses documentos, constará como um dos entregáveis o software de reserva de salas de reuniões e equipamentos em produção de acordo com o especificado no primeiro *sprint*. Abaixo seguem as principais telas do sistema em produção:

a) Tela de menu de cadastro para equipamento e salas de reuniões, sendo que a de cadastro de funcionários está como uma das prioridades do segundo *sprint*.

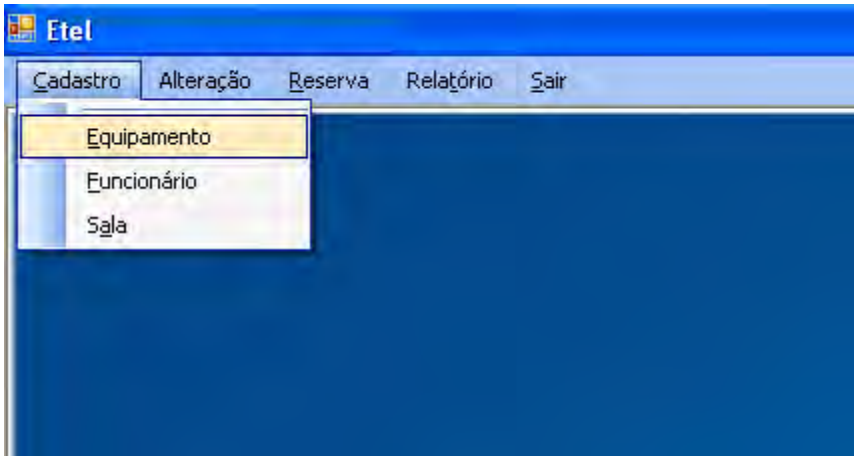

Figura 11- Menu de cadastros. FONTE: AUTORES(2009)

b) Tela de menu de alteração para equipamento e salas de reuniões, sendo que a de alteração de funcionários está como uma das prioridades do segundo *sprint*.

Figura 12 - Menu de alterações. FONTE: AUTORES(2009)

c) Tela de reserva de sala ou equipamento:

Figura 13 - Reserva de sala ou equipamento. FONTE: AUTORES(2009)

d) Tela de cadastro de equipamento:

Figura 14 - Cadastro de equipamentos. FONTE: AUTORES(2009)

e) Tela de cadastro de sala:

Figura 15 - Cadastro de sala. FONTE: AUTORES(2009)

Segundo *Sprint*

O começo do segundo *Sprint* ocorreu no dia trinta de outubro de dois mil e nove (30/10/2009). O atraso deveu-se a pouca disponibilidade do responsável do projeto por parte do cliente.

Como resultado da etapa de exploração, em conjunto com o primeiro *Sprint*, foi possível listar os seguintes requisitos a serem incluídos no segundo *Sprint*, conforme tabela abaixo:

Tabela 5 - Requisitos levantados para inclusão no segundo *sprint*. FONTE: AUTORES(2009)

Resultado Exploração	Resultado primeiro *Sprint*
Cadastrar funcionário	Relatório de reserva de sala por funcionário
Relatório de reserva de sala por semana	Relatório de reserva de equipamento por funcionário.

Também, como partes do resultado do primeiro *Sprint* foram identificadas as seguintes regras de negócio:

Reserva de uma sala de reunião não deve constar na mesma data e hora

Reserva de sala de reunião deve ser registrado com uma semana de antecedência.

Controle de estoque deve ser feito na reserva de equipamento

Cada nova funcionalidade resultado do primeiro *Sprint* foi associada à uma *User Story* (ver APÊNDICES L ao O). As regras de negócio foram identificadas e associadas durante a fase de análise de requisitos.

No caso de alguma regra de negócio nova alterar alguma *User Story* já elaborada, cada uma destas alterações deverão ser registradas na seção de histórico de alterações da *User Story* correspondente, identificando a data, hora e usuário responsável pelo pedido de modificação.

Isto permitiu um adequado controle nas mudanças sobre os requisitos identificados.

Antes do início das atividades, o *Scrum Master* convocou uma reunião com o *Scrum Team* para apresentar informações sobre as tarefas, o prazo de duração do *sprint* (sete dias) e a permanência de reuniões diárias.

Para o segundo *Sprint*, foram selecionados os requisitos com prioridade alta, conforme tabela abaixo:

Tabela 6– *Sprint* Backlog – *Sprint* 2. FONTE: AUTORES(2009)

Prioridade	Descrição	Tarefas	Tempo Estimado (h)	Tempo de Conclusão (h)	Responsável	Status
Alta	Manutenção de funcionários	Cadastrar funcionário	4hrs	5h e 15 minutos	Fábio Lima	Concluído
		Alterar funcionário	4hrs	5h e 25 minutos		Concluído
		Editar funcionário	4hrs	3h e 20 minutos		Concluído
		Excluir funcionário	4hrs	2h		Concluído
	Relatório de reserva de sala por semana	Gerar relatório	4hrs	5h	Nick Lazur	Concluído
		Disponibilizar relatórios	4hrs	3h		Concluído
	Relatório de reserva de sala por funcionário	Gerar relatório	4hrs	4h	Eduardo Myake	Concluído
		Disponibilizar relatórios	4hrs	2h		Concluído
	Relatório de reserva de equipamento por funcionário	Gerar relatório	4hrs	3h e 45 minutos	Cesar Sanz	Concluído
		Disponibilizar relatórios	4hrs	2h		Concluído

Os requisitos da tabela acima foram detalhados pela prioridade, descrição, tarefas, tempo estimado, tempo de conclusão, responsável e status.

Este detalhamento pode ser visto melhor nas *user stories* associadas (Ver APÊNDICES de P ao S).

Baseando-se nas *user stories* detalhadas, será possível desenvolver o documento de requisitos composto pelo detalhamento das tarefas. Para o desenvolvimento do documento, foi necessário levantar e ter bem claro qual o objetivo, o âmbito, as funções do produto e os requisitos específicos. Levantadas estas informações, foram definidos os requisitos funcionais e as regras de negócio, auxiliando assim na elaboração do protótipo (APÊNDICE U). Uma vez definido o documento de requisitos, iniciou-se, em conjunto com o responsável do projeto da empresa de telefonia, a validação do documento elaborado. Em paralelo, foi desenvolvida a parte final do sistema, sendo também validado pelo cliente.

Com estes resultados encerrou-se o segundo *sprint* tendo como entregáveis as *user stories* devidamente preenchidas, o documento de requisitos do segundo *sprint*, já validado pelo cliente.

Além desses documentos, constará como um dos entregáveis o software de reserva de salas de reuniões e equipamentos em produção de acordo com o especificado no segundo *sprint* . Abaixo seguem as principais telas do sistema em produção:

a) Tela de menu de cadastro de funcionário:

Figura 16 - Tela de cadastro de funcionário. FONTE: AUTORES(2009)

b) Tela de menu de alteração de funcionário:

Figura 17 - Tela de alteração de funcionário. FONTE: AUTORES(2009)

c) Tela de cadastro de Equipamento:

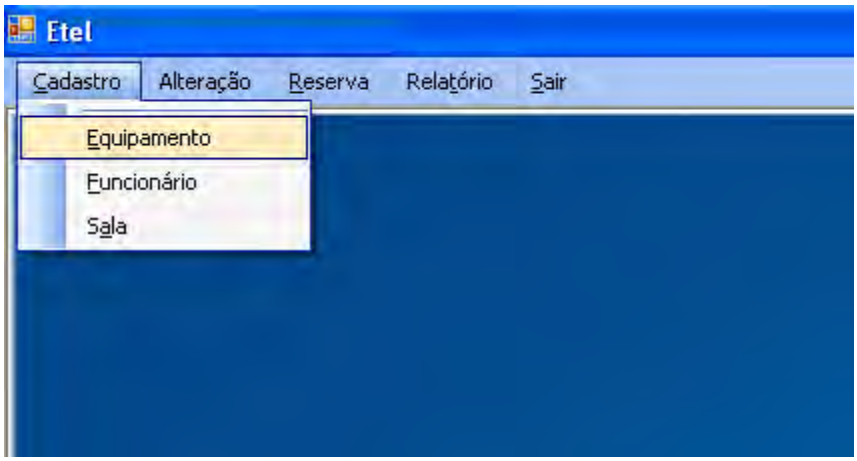

Figura 18 - Tela de cadastro de funcionário. FONTE: AUTORES(2009)

d) Tela de relatório por sala:

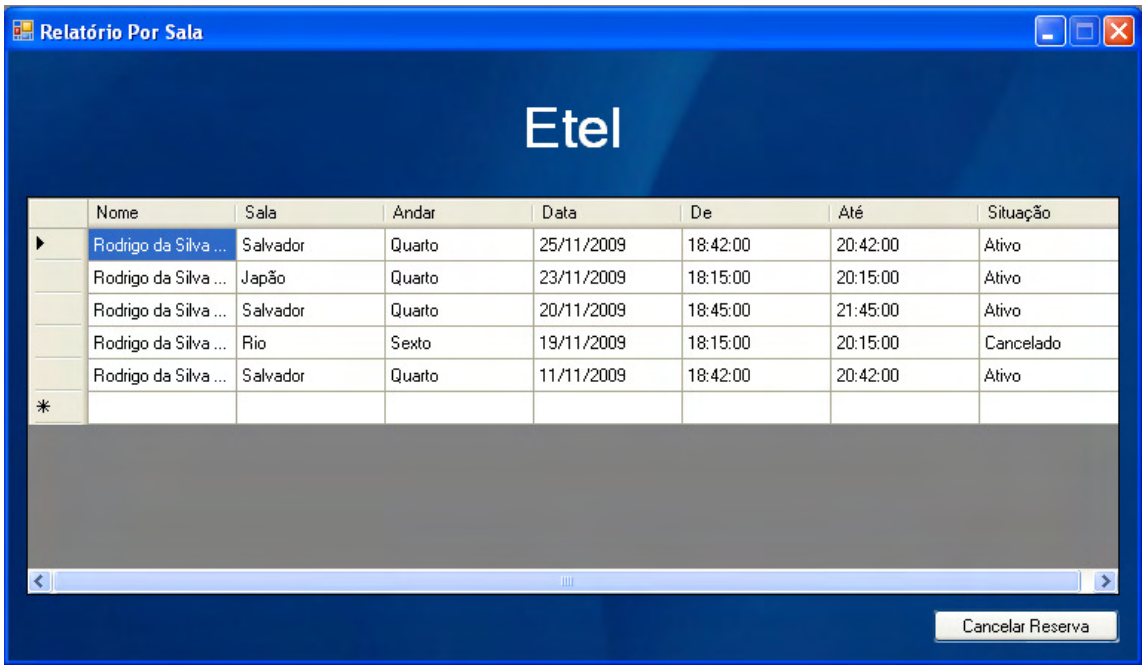

Figura 19 - Tela de relatório por sala. FONTE: AUTORES(2009)

Durante a execução deste *Sprint* foram identificados requisitos não-funcionais, que em acordo com o cliente decidiu-se colocá-los diretamente no documento de requisitos, dado que as *User Stories* só permitem guardar requisitos funcionais. Abaixo segue a lista destes requisitos não-funcionais:

[RNF01] Segurança

O sistema deve considerar a liberação de acesso aos usuários, com seus respectivos perfis, para este caso cada funcionário deve ter um usuário e senha de acesso ao sistema.

[RNF02] Usabilidade

Dado que o usuário não possui tempo disponível para aprender como utilizar o sistema, a interface com o usuário é de vital importância para o sucesso do sistema. O sistema terá uma interface amigável ao usuário primário sem se tornar cansativa aos usuários mais experientes.

[RNF03] Desempenho

Embora não seja um requisito essencial ao sistema, deve ser considerada por corresponder a um fator de qualidade de software.

[RNF04] Hardware e Software

Visando criar um produto com maior extensibilidade, reusabilidade e flexibilidade, deve ser adotar como linguagem principal de desenvolvimento Power Builder. Entretanto, outras linguagens também poderão ser usadas quando indicações técnicas recomendem.

O uso da linguagem *Power Builder* deve-se a que atualmente existem outros sistemas também internos no cliente, que utilizam está linguagem, e seria uma forma de padronizar este tipo de sistemas. O sistema operacional deve ser Windows XP, sem restrições de máquina. No entanto, essa máquina deverá se comunicar com um sistema de banco de dados.

5.3.3 PRODUÇÃO

Depois da execução do primeiro *Sprint* e com a primeira versão do sistema funcionando, os usuários começaram a utilizá-lo nas atividades do dia-a-dia. Com isto, novos requisitos e regras de negócio foram detectados de forma a satisfazer as necessidades do cliente.

Para tanto, decidiu-se colocar estes requisitos primeiro nas *User Stories*, atualizando os já existentes ou criando novas *User Stories*, para posteriormente refleti-las no documento de requisitos.

O curto prazo entre os *Sprints* não permitiu entrar com maiores detalhes no processo de gestão de requisitos, além da impossibilidade de incluí-las no segundo *Sprint*. Para tanto, seria necessário realizar um terceiro *Sprint* para atender a solicitação do cliente.

O terceiro *Sprint* seria executado dentro da mesma filosofia em que foram concebidos os dois primeiros, seguindo os processos propostos para este estudo de caso.

As seguintes necessidades foram identificadas por parte dos usuários:

a) Regras de negócio

Se uma reserva de uma sala de reunião já tiver sido efetuada para um determinado horário e outro funcionário desejar realizar uma reserva na mesma data e hora, a reserva poderá ser realizada com status “Em Espera”, possibilitando diante de um eventual cancelamento, a efetivação da primeira reserva após a cancelada.

A mesma regra deve ser aplicada no caso de reserva de equipamento.

b) Novos requisitos

Relatório de histórico de cancelamento de reserva de salas de reunião por semana.

Relatório de histórico de cancelamento de reserva de equipamento por semana.

Com estes levantamentos a definição das necessidades ficaram conforme listado abaixo:

a) As regras de negócio serão adicionadas, tanto na *user story* 03 - Solicitação de salas de reunião, e na *user story* 04 - Solicitação de Equipamento. Também será registrada em cada uma das u*sersStories* na seção de histórico de alterações a data, hora e o usuário que está solicitando a mudança

b) Para os relatórios foi decidido criar uma *User Story* por cada um. Seguindo com o processo normal desde a criação do *User Story*.

5.4 ANÁLISE DE RESULTADOS

Durante a execução do estudo de caso se conseguiu identificar que utilizando princípios e praticas ágeis nos processos da engenharia de requisitos, se consegue fazer um trabalho de muita qualidade, permitindo obter ótimos resultados na elaboração do documento de requisitos e uma adequada gestão do mesmo.

A continuação se faz uma análise crítica, detalhando os resultados obtidos em cada uma das etapas da estratégia executada durante o estudo de caso, indicando as vantagens e desvantagens encontradas para poder alcançar o objetivo deste trabalho.

Na etapa de exploração conseguiu-se definir os requisitos em alto nível através das *User Stories*, com isto, foi possível para as etapas seguintes, ter um melhor controle nos requisitos, tanto no detalhamento, que foi conseguido quando se complementou com as regras de negócio e a estimativa inicial para a realização de cada *User Story*, e também com a gestão de mudanças através do histórico de alterações, que foi uma adaptação na *User Story* para ajudar na rastreabilidade de qualquer mudança ou alteração de um requisito já existente.

Práticas ágeis como *on-site customer* e reuniões *face-to-face* permitiram que a comunicação com o cliente pudesse ser mais próxima.

A problemática encontrada existiu com a identificação dos requisitos não-funcionais, dado que estes requisitos descrevem restrições que devem ser impostas ao sistema, e que em geral se aplicam a várias funcionalidades. Como as *User Stories* foram utilizadas para especificar os requisitos funcionais, decidiu-se refletir este tipo de requisito no mesmo documento de requisitos.

Posteriormente, na etapa de execução, conseguiu-se mediante práticas ágeis, primeiro, com reuniões de equipe, sempre com a participação do cliente, completar as *User Stories* com suas respectivas regras de negócio gerando uma estimativa inicial para o desenvolvimento de cada funcionalidade.

Embora as filosofias ágeis coloquem ênfase no fato de ser melhor ter um software funcionando do que muita documentação foi necessário a elaboração do Documento de Requisitos. Isto auxilia a reflexão de uma forma mais detalhada, todas as necessidades do cliente, definindo os requisitos funcionais, não funcionais, protótipos, testes de aceitação, todos em um só documento, ao qual será muito mais fácil de gerenciar.

Através de pequenos releases foi possível definir as maiores carências do Cliente, entregando para o mesmo, o sistema funcionando em versões determinadas na priorização dos requisitos e detalhados no resultado de cada um dos *Sprints*.

Terminando com a retrospectiva, que é uma prática do Scrum que foi utilizada para realizar ou identificar melhorias no sistema, se conseguiu ter refinamento nos requisitos, que identificados são introduzidos nos releases seguintes e assim sucessivamente até satisfazer as necessidades do cliente através de um software de qualidade.

Portanto, aceitar mudanças iterativas e haver a participação ativa dos *stakeholders* é os princípios e práticas ágeis dentro do processo da ER.

Na etapa de produção, onde se faz a gestão dos requisitos na qualidade ou no controle de mudanças, com a colaboração do cliente, possibilitou observar melhorias no sistema já em operação. Novos requisitos ou melhorias devem aparecer, voltando aos processos já identificados e explanados durante as etapas de exploração e execução.

6 CONCLUSÕES

A engenharia de requisitos na metodologia ágil permite trabalhar de forma paralela todas as fases do ciclo da engenharia de requisitos, seja na elaboração do documento de requisitos, permitindo através de um adequado levantamento, maior flexibilidade na sua elaboração, conseguindo entregar ao cliente resultados concretos referentes às suas necessidades de negócio de forma antecipada.

Com esta filosofia, o risco de serem implementados requisitos desnecessários é reduzido de forma significativa, tendo como resultado a possibilidade de se obter uma redução de riscos e custos durante o processo de desenvolvimento de software.

Para conseguir estas vantagens para a engenharia de requisitos, foi preciso utilizar práticas ágeis, tais como, mudanças iterativas, *on-site customer*, pequenos releases e comunicação verbal, aperfeiçoando os processos de desenvolvimento e gestão de requisitos.

Neste trabalho, foram apresentados os principais ganhos provenientes da engenharia de requisitos na metodologia ágil, além do aproveitamento de artefatos já definidos nos métodos tradicionais, tais como documento de requisitos, formulário de validação, dentre outros, e algumas sugestões de melhoria baseadas nas práticas tradicionais de engenharia de requisitos quanto a aspectos relacionados à interação com usuário, gerenciamento de mudanças e o levantamento dos requisitos não-funcionais.

Este trabalho serve de base para trabalhos futuros visto a importância do assunto para a prática da engenharia de software nas empresas. Composto por um bom embasamento teórico permitirá o conhecimento de alguns métodos e o aprofundamento em outros, além dos aplicados no estudo de caso.

REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, S. W. **Modelagem Áil**, São Paulo: Bookman, 2002.

BECK, K., **Manifesto for agile software development**. Disponível em: http://www.agilemanifesto.org. Acessado em: 18 abr. 2009.

BECK, K. **Extreme Programming Explained: Embrace Change.** Addison Wesley, 2000.

ENGENHARIA DE REQUISITOS ÁGIL. Disponivel em: http://www.cin.ufpe.br/~in1020/arquivos/monografias/2008-1/Monografia_AgileER.doc. Acessado em: 15 mai. 2009.

FOWLER, Martin. **The New Methodology. 2001.** Disponivel em: http://www.martinfowler.com/articles/newMethodology.html. Acessado em: 12 mai. 2009.

HARTMANN, Julio. Estudo sobre a aplicação de métodos ágeis no desenvolvimento e gerenciamento de projetos de software. Monografía (Pós-graduação em Computação) - Universidade Federal de Rio Grande do Sul - UFRGS, Porto Alegre, 2003.

HIGHSMITH, J. **Agile Software Development Ecosystems.** Addison Wesley, 2002.

HIGHSMITH, J.; COCKBURN, A. **Agile Software Development: The Business of Innovation.** IEEE Computer, 2001.

IEEE Std 830-1998 IEEE. Recommended Practice for Software Requirements Specifications. Disponível em: <http://standards.ieee.org/reading/ieee/std_public/description/se/830-1998_desc.html>. Acessado em: 30 mai. 2009.

METODOLOGIA ÁGIL. Projeto e implementação de uma ferramenta para gerência de requisitos em metodologias ágeis. Disponível em: <http://www.seminfo.com.br/anais/2008/pdfs/seminfo/8-50723.pdf>. Acessado em: 15 mar. 2009.

MÉTODOS ÁGEIS. Framework para comparação e análise de métodos ágeis. Disponível em:<http://www.tede.ufsc.br/teses/PGCC0662.pdf>. Acessado em: 30 abr. 2009.

MÉTODOS ÁGEIS. Uma aplicação do Scrum no SIMUPLAN. Disponível em: <http://inf.upf.br/gepes/downloads/TC_Elias_Muller_M%E9todos_%C1geis.pdf>. Acessado em: 27 mar. 2009.

PALMER, S. R.; FELSING, J. M. **A Practical Guide to Feature-Driven Development.** Prentice Hall, 2002

PFLEEGER, S. L., **Engenharia de Software**, 2. ed. São Paulo: Pearson, 2004

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

SCRUM. Scrum. Disponível em: <www.brod.com.br/?q=node/396>. Acessado em: 01 abr. 2009.

SCRUM. Scrum. Disponível em: <http://oglobo.globo.com/blogs/tecnologia/posts/2009/01/08/treinamento-de-scrum-algumas-ideias-que-definem-metodo-148904.asp>. Acessado em: 01 abr. 2009.

SCRUM. Uma proposta de extensão de um Método Ágil para Gerência e Desenvolvimento de Requisitos visando adequação ao CMMI. Disponível em: <http://www.tede.ufsc.br/teses/PGCC0651.pdf>. Acessado em: 10 abr. 2009.

SIQUEIRA, F. L., **Método de Comparação e análise de metodologias para o desenvolvimento de um sistema de discussão e colaboração.** Escola Politécnica 2002.

SOARES, A. L., **Engenharia de Requistos - 1. Conceitos Fundamentais**. Disponível em: < http://paginas.fe.up.pt/~jmpc/courses/erss0809/>. Acessado em: 30 mai. 2009.

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

TAKEUCHI, Hirotaka e NONAKA, Ikujiro. **Gestão do Conhecimento**. Bookman, 2008.

VOLERE REQUIREMENTS SPECIFICATION TEMPLATE. Extracts and Samples from the Template. By James & Suzanne Robertson principals of the Atlantic Systems Guild. Disponível em: < http://www.volere.co.uk/template.htm>. Acessado em: 30 mai. 2009.

WAKE, William C. Extreme Programming Explored. Reading, Massachusetts: Ed. Addison-Wesley, 2002.

ANEXOS

ANEXO A - CHANGE REQUEST FORM (CRF)

CHANGE REQUEST — PARTE I			
Nome do Cliente:			
Nome do Projeto:		**Projeto:**	
Fase do Projeto:		**Data Atual:**	
Gerente Projeto:		**Data Estimada:**	
Nome Requerimento:		**Requisito #:**	
Motivo da Mudança:		Preparado por:	
Descrição da Mudança:		Preparado por:	
Custo Aproximado:		Preparado por:	

Ramificações (e.g., schedule and staffing):		Preparado por:	
(A)provado/(R)ejeitado/(C)ancelado		(A)provado/(R)ejeitado/(C)ancelado	
Organização de Entrega:		Cliente:	
Nome:		Nome:	
Assinatura:		Assinatura:	
Data:		Data:	

CHANGE REQUEST — PARTE II			
Nome do Projeto:		**Projeto:**	
Nome Requerimento:		**Requisito #:**	

Custos	Quantidade
Atualizar entregáveis originais (Preparar uma lista detalhada).	
Atualizar documentações internas, tais como, Planos de Teste (Preparar uma lista detalhada).	
Mudança de itens de configuração de software, tais como, códigos, banco de dados, telas (Preparar uma lista detalhada).	
Gerenciamento de Configuração.	
Testes (incl. Unitarios, Integrados, Sistemas, Aceitação)	
Procedimentos e revisões.	
Modificações de Banco de Dados	
Correções de dados	
Tempo de orientação	
Liderança de Equipe e Gestão de Projeto	
Viagens e despensas	
Custo de preparação de Requerimento de mudança	
Custo de Hardware e Software	
Implicações de Arquitetura (espaço em disco, CPU, etc.)	

Custo de facilidades		
Impacto da programação de pagamento		
Outros		
Contigencias		
CUSTO TOTAL		
Estimação Preparado por:		
Estimação Revisado por:		

APÊNDICES

APÊNDICE A – User story 01 - Manutenção de salas de reunião

US01
Manutenção de salas de reunião, onde o responsável pela administração do sistema poderá fazer o cadastro, edição e exclusão de salas de reunião.
RN1.1
TA1.1 - Cadastrar uma sala de reunião com todos os dados corretos (Cadastro efetuado com sucesso). TA1.2 - Cadastrar uma sala de reunião sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA1.3 - Cadastrar uma sala de reunião dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA1.4 - Editar uma sala de reunião com dados corretos (Edição deve ser concluída). TA1.5 - Editar uma sala de reunião com dados incorretos (Edição não pode ser concluída). TA1.6 - Editar uma sala de reunião retirando dados obrigatórios (Edição não pode ser concluída) TA1.7 - Excluir uma sala de reunião (Exclusão deve ser concluída).
Estimativa Inicial:
HA1.1

APÊNDICE B – User story 02 - Manutenção de equipamentos

US02
Manutenção de equipamentos, onde o responsável pela administração do sistema poderá fazer o cadastro, edição e exclusão de equipamentos.
RN2.1 -
TA2.1 - Cadastrar um equipamento com todos os dados corretos (Cadastro efetuado com sucesso). TA2.2 - Cadastrar um equipamento sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA2.3 - Cadastrar um equipamento dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA2.4 - Editar um equipamento com dados corretos (Edição deve ser concluída). TA2.5 - Editar um equipamento com dados incorretos (Edição não pode ser concluída). TA2.6 - Editar um equipamento retirando dados obrigatórios (Edição não pode ser concluída) TA2.7 - Excluir um equipamento (Exclusão deve ser concluída).
Estimativa Inicial:
HA2.1

APÊNDICE C – User story 03 –Solicitação de sala de reunião

US03
Solicitação de sala de reunião, onde qualquer funcionário com acesso ao sistema poderá registrar uma solicitação, editar e cancelar a solicitação
RN3.1 -
TA3.1 - Cadastrar uma solictação de sala com todos os dados corretos (Cadastro efetuado com sucesso). TA3.2 - Cadastrar uma solictação de sala sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA3.3 - Cadastrar uma solictação de sala dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA3.4 - Editar uma solictação de sala com dados corretos (Edição deve ser concluída). TA3.5 - Editar uma solictação de sala com dados incorretos (Edição não pode ser concluída). TA3.6 - Editar uma solictação de sala retirando dados obrigatórios (Edição não pode ser concluída) TA3.7 - Excluir uma solictação de sala (Exclusão deve ser concluída).
Estimativa Inicial:
HA3.1

US04
Solicitação de equipamento, onde qualquer funcionário com acesso ao sistema poderá registrar uma solicitação, editar e cancelar a solicitação.
RN4.1 -
TA4.1 - Cadastrar uma solictação de equipamento com todos os dados corretos (Cadastro efetuado com sucesso). TA4.2 - Cadastrar uma solictação de equipamento sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA4.3 - Cadastrar uma solictação de equipamento dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA4.4 - Editar uma solictação de equipamento com dados corretos (Edição deve ser concluída). TA4.5 - Editar uma solictação de equipamento com dados incorretos (Edição não pode ser concluída). TA4.6 - Editar uma solictação de equipamento retirando dados obrigatórios (Edição não pode ser concluída) TA4.7 - Excluir uma solictação de equipamento (Exclusão deve ser concluída).
Estimativa Inicial:
HA4.1

APÊNDICE E – User story 05 – Manutenção de funcionários

US05
Manutenção de funcionário, onde se poderá fazer o cadastro, edição e exclusão de funcionários.
RN5.1 -
TA5.1 - Cadastrar um funcionário com todos os dados corretos (Cadastro efetuado com sucesso). TA5.2 - Cadastrar um funcionário sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA5.3 - Cadastrar um funcionário dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA5.4 - Editar um funcionário com dados corretos (Edição deve ser concluída). TA5.5 - Editar um funcionário com dados incorretos (Edição não pode ser concluída). TA5.6 - Editar um funcionário retirando dados obrigatórios (Edição não pode ser concluída) TA5.7 - Excluir um funcionário (Exclusão deve ser concluída).
Estimativa Inicial:
HA5.1

APÊNDICE F – User story 06 – Relatório de reserva de sala por semana

US06
Relatório de reserva de sala por semana, permite fazer a consulta e impressão do relátório definido.
RN6.1 -
TA6.1 - Geração de relatório com opção de impressão. TA6.2 - Disponibilizar exportação do relatório.
Estimativa Inicial:
HA6.1

US01
Manutenção de salas de reunião, onde o responsável pela administração do sistema poderá fazer o cadastro, edição e exclusão de salas de reunião.
RN1.1 - Cadastro de sala de reunião não pode ser efetuado sem informar todos os dados obrigatórios RN1.2 - Cadastro de sala de reunião dentro de um horário e data existentes não deve ser efetuado RN1.3 - Alteração de uma sala de reunião com dados corretos não deve ser concluída RN1.4 - Alteração de uma sala de reunião com dados incorreto não deve ser concluída RN1.5 - Alteração de uma sala de reunião retirando dados obrigatórios não pode ser concluída
TA1.1 - Cadastrar uma sala de reunião com todos os dados corretos (Cadastro efetuado com sucesso). TA1.2 - Cadastrar uma sala de reunião sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA1.3 - Editar uma sala de reunião com dados corretos (Edição deve ser concluída). TA1.4 - Editar uma sala de reunião com dados incorretos (Edição não pode ser concluída). TA1.5 - Editar uma sala de reunião retirando dados obrigatórios (Edição não pode ser concluída) TA1.6 - Excluir uma sala de reunião (Exclusão deve ser concluída).
Estimativa Inicial: 16h
HA1.1

APÊNDICE H – User story 02 – Manutenção de equipamentos

US02
Manutenção de equipamentos, onde o responsável pela administração do sistema poderá fazer o cadastro, edição e exclusão de equipamentos.
RN2.1 - Cadastro de equipamentos não pode ser efetuado sem a informação de todos os dados obrigatórios RN2.2 - Cadastro de equipamentos não pode ser efetuado em horário ou data já existentes RN2.3 - Alteração de equipamento não pode ser efetuada com dados incorretos RN2.4 - Alteração de equipamento não pode ser concluída quando dados obrigatórios são retirados
TA2.1 - Cadastrar um equipamento com todos os dados corretos (Cadastro efetuado com sucesso). TA2.2 - Cadastrar um equipamento sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA2.3 - Editar um equipamento com dados corretos (Edição deve ser concluída). TA2.4 - Editar um equipamento com dados incorretos (Edição não pode ser concluída). TA2.5 - Editar um equipamento retirando dados obrigatórios (Edição não pode ser concluída) TA2.6 - Excluir um equipamento (Exclusão deve ser concluída).
Estimativa Inicial: 16h
HA2.1

APÊNDICE I – User story 03 – Solicitar salas

US03
Solicitação de sala de reunião, onde qualquer funcionário com acesso ao sistema poderá registrar uma solicitação, editar e cancelar a solicitação
RN3.1 - Solicitação de sala não pode ser efetuado sem informar todos os dados obrigatórios RN3.2 - Solicitação de sala dentro de um horário ou data já existentes não pode ser concluída RN3.3 - Alteração de solicitação com dados incorretos não pode ser concluída RN3.4 - Alteração de solicitação retirando dados obrigatórios não pode ser concluída. RN3.5 - Reserva de uma sala de reunião não deve constar na mesma data e hora RN3.6 - Reserva de sala de reunião deve ser registrado com uma semana de antecedência.
TA3.1 - Cadastrar uma solictação de sala com todos os dados corretos (Cadastro efetuado com sucesso). TA3.2 - Cadastrar uma solictação de sala sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA3.3 - Cadastrar uma solictação de sala dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA3.4 - Editar uma solictação de sala com dados corretos (Edição deve ser concluída). TA3.5 - Editar uma solictação de sala com dados incorretos (Edição não pode ser concluída). TA3.6 - Editar uma solictação de sala retirando dados obrigatórios (Edição não pode ser concluída) TA3.7 - Cancelar uma solictação de sala (Cancelamento deve ser concluído). TA3.8 - Cadastrar uma solictação de sala de reunião numa data e hora já existente (Cadastro não deve ser efetuado). TA3.9 - Cadastrar uma solicitação de sala de reunião com duas semanas de antecedência (Cadastro não deve ser efetuado) TA3.10 - Cadastrar uma solicitação de sala de reunião com uma semana de antecedência (Cadastro efetuado com sucesso)
Estimativa Inicial: 12h
HA3.1

APÊNDICE J – User story 04 – Solicitar equipamentos

US04
Solicitação de equipamento, onde qualquer funcionário com acesso ao sistema poderá registrar uma solicitação, editar e cancelar a solicitação.
RN4.1 - Solicitação de equipamento não deve ser efetuado sem informar os dados obrigatórios RN4.2 - Solicitação de equipamento dentro de um horário/ data já existentes não deve ser efetuado RN4.3 - Solicitação com dados incorrretos não pode ser efetuada RN4.4 - Alterar uma solicitação retirando dados obrigatórios RN4.5 - Controle de estoque deve ser feito na reserva de equipamento
TA4.1 - Cadastrar uma solictação de equipamento com todos os dados corretos (Cadastro efetuado com sucesso). TA4.2 - Cadastrar uma solictação de equipamento sem informar todos os dados obrigatórios (Cadastro não deve ser efetuado). TA4.3 - Cadastrar uma solictação de equipamento dentro de um horario e data já existentes (Cadastro não deve ser efetuado). TA4.4 - Editar uma solictação de equipamento com dados corretos (Edição deve ser concluída). TA4.5 - Editar uma solictação de equipamento com dados incorretos (Edição não pode ser concluída). TA4.6 - Editar uma solictação de equipamento retirando dados obrigatórios (Edição não pode ser concluída) TA4.7 - Cancelar uma solictação de equipamento (Cancelamento deve ser concluído). TA4.8 - Cadastrar uma solictação quando não tiver estoque disponivel (Cadastro não deve ser efetuado)
Estimativa Inicial: 12h
HA4.1

APÊNDICE K – Formulário de validação

Formulário de Validação	
Doc. De Requisitos (Nome, Versão):	
Nome Arquivo:	
Elaborador (Papel, Nome, E-mail, Ramal):	
Revisor (Papel, Nome, E-mail, Ramal):	
Data Revisão (dd/mm/aaaa):	Tempo revisão (Horas):

Desconformidades encontrada				
Número	Página	Linha	Requisito	Descrição

Observações Gerais dos revisores

APÊNDICE L – User Story 05 – Execução – Manutenção de funcionário

US05
Manutenção de funcionário, onde se poderá fazer o cadastro, edição e exclusão de funcionários.
RN5.1 - Cadastro de funcionário não pode ser efetuado sem informar todos os dados obrigatórios RN5.2 - Alteração de registro de funcionário com todos os dados preenchidos corretamente. RN5.3 - Alteração de um registro de funcionário com dados incorretos
TA5.1 - Cadastrar um funcionário com todos os dados corretos (Cadastro efetuado com sucesso). TA5.2 - Cadastrar um funcionário sem informar todos os dados obrigatórios (Cadastro no deve ser efetuado). TA5.3 - Editar um funcionário com dados corretos (Edição deve ser concluída). TA5.4 - Editar um funcionário com dados incorretos (Edição não pode ser concluída). TA5.5 - Editar um funcionário retirando dados obrigatórios (Edição não pode ser concluída) TA5.6 - Excluir um funcionário (Exclusão deve ser concluída).
Estimativa Inicial: 8h
HA5.1

APÊNDICE M – User Story 06 – Execução – Relatório de reserva de sala por semana

US06
Relatório de reserva de sala por semana, permite fazer a consulta e impressão do relátório definido.
RN6.1 - Geração de relatório deve conter dados requeridos, tais como sala de reunião, data, dia da semana, hora início, hora fim e funcionário.
TA6.1 - Geração de relatório com opção de impressão. TA6.2 - Disponibilizar exportação do relatório.
Estimativa Inicial: 8h
HA6.1

APÊNDICE N – User Story 07 – Execução – Relatório de reserva de sala por funcionário

US07
Relatório de reserva de sala por funcionário, permite fazer a consulta e impressão do relátório definido.
RN7.1 - Geração de relatório deve conter dados requeridos, tais como data, hora início, hora fim, sala de reunião e funcionário.
TA7.1 - Geração de relatório com opção de impressão. TA7.2 - Disponibilizar exportação do relatório.
Estimativa Inicial: 8h
HA7.1

APÊNDICE O – User Story 08 – Execução – Relatório de reserva de equipamento por funcionário

US08
Relatório de reserva de equipamento por funcionário, permite fazer a consulta e impressão do relátório definido.
RN8.1 - Geração de relatório deve conter dados requeridos, tais como data, hora início, hora fim, equipamento e funcionário.
TA8.1 - Geração de relatório com opção de impressão. TA8.2 - Disponibilizar exportação do relatório.
Estimativa Inicial: 8h
HA8.1

Documento de Requisitos do Sistema

Reserva de Salas de Reunião e Equipamentos

Versão 1.0

Histórico de Alterações

Data	Versão	Descrição	Autor
09/10/2009	1.0	Documento de Requisitos do sistema de reserva de salas de reunião e equipamentos, como resultado do Primeiro *Sprint*.	Eduardo Miyake

INTRODUÇÃO

Este documento especifica os requisitos do sistema de Reserva de Salas de reunião e Equipamentos, fornecendo aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação do sistema.

OBJETIVO

Especificar os requisitos levantados na etapa de exploração, classificando os tipos de requisitos e detalhando-os.

ÂMBITO

O Sistema proposto para a empresa de telecomunicações é uma ferramenta para controle de solicitações de salas de reunião e equipamentos.

Atualmente o processo de reserva é efetuado de forma manual, gerando transtorno aos usuários. Para reserva de uma sala o funcionário precisa se deslocar até a sala desejada e colocar seu nome e horário numa planilha. Na indisponibilidade do mesmo, será necessário dirigir-se à outra até localizar uma livre. Mais desorganizado é a solicitação de equipamentos. Não há um controle para o mesmo, havendo a possibilidade de no momento da reunião não haver o equipamento disponível.

O sistema atenderá este cenário, otimizando o dia-a-dia dos funcionários.

VISÂO GERAL DO DOCUMENTO

- **Na seção 2** é apresentada uma visão geral do sistema, caracterizando qual é o seu escopo e listando as funcionalidades requeridas.
- **Na seção 3** são enumerados todos os requisitos funcionais, não funcionais e regras de negócio para os requisitos levantados.
- **Na seção 4** são propostos os protótipos para o sistema de reserva de salas de reunião e equipamentos.

- **Na seção 5** são definidos os testes de aceitação que serão utilizados para aprovação dos requisitos.

VISÃO GERAL DO SISTEMA

Nesta etapa serão descritos passos importantes que ajudaram no entendimento dos requisitos e a especificá-los posteriormente.

ABRANGÊNCIA DO SISTEMA

O sistema de Reserva de Salas de Reunião e Equipamentos permitirá aos usuários poder elaborar uma adequada gestão das salas de reunião e equipamentos que formam parte da Empresa, evitando desconforto e ajudando a levar a gestão de uma forma simples mas eficaz.

Este Documento pretende ser o Documento de Requisitos, Análise e Projeto do Sistema de Reserva de Salas e Equipamentos, abrangendo desde o levantamento de Requisitos, passando por seu refinamento e discussão.

FUNÇÕES DO PRODUTO

As funcionalidades consideradas durante a execução do primeiro *Sprint* são as seguintes:

a) Manutenção de Equipamentos;

b) Manutenção de Salas de Reunião;

c) Solicitação de Sala;

d) Solicitação de Equipamentos;

ESPECIFICAÇÃO DE REQUISITOS

Neste capítulo os requisitos serão detalhados e divididos em requistos funcionais, requisitos não-funcionais e regras de negócio.

USER STORIES

As *user stories* criadas foram tratadas conforme tabela abaixo:

User Story	Descrição
US01	Manutenção de salas de reunião, onde um responsável por administrar o sistema poderá cadastrar, alterar e excluir salas.
US02	Manutenção de equipamentos, onde um responsável por administrar o sistema poderá cadastrar, alterar e excluir equipamentos.
US03	Solicitação de salas de reunião, onde qualquer funcionário com acesso ao sistema poderá cadastrar, alterar e cancelar solicitação.
US04	Solicitação de Equipamentos, onde qualquer funcionário com acesso ao sistema poderá cadastrar, alterar e cancelar solicitação.

REQUISITOS FUNCIONAIS

Identificação e detalhamento das funcionalidades do sistema extraídas das *user stories.*

Manutenção de Salas de Reunião (US01)

[RF01] – Cadastrar uma sala com todos os dados corretos

Descrição: Cadastrar uma sala com todos os dados corretos. O administrador do sistema fará o cadastramento da sala mediante informações corretas (Nome da sala e Localização).

[RF02]- Alterar uma sala de reunião com dados corretos

Descrição: Alterar uma sala de reunião com dados corretos. O administrador do sistema fará a edição da sala de reunião mediante informações corretas (Nome da sala e Localização).

[RF03] – Excluir cadastro de sala de reunião

Descrição: Excluir cadastro de sala de reunião. O administrador do sistema fará a exclusão de uma sala de reunião.

Manutenção de Equipamentos (US02)

[RF04] – Cadastrar equipamento com os dados corretos

Descrição – Cadastrar um equipamento com todos os dados corretos. O administrador do sistema irá informar o tipo de equipamento (Datashow, Notebook, Retroprojetor ou quadro branco), o modelo, número de série e observações.

[RF05] – Editar equipamento com dados corretos

Descrição – Alterar equipamento com dados corretos. O administrador do sistema fará a alteração dos dados do equipamento que estiverem corretos, sendo necessário informar o motivo.

[RF06] – Excluir um equipamento

Descrição – Excluir um equipamento. O administrador do sistema fará a exclusão de um equipamento, sendo necessário informar o motivo.

Solicitação de Sala de Reunião (US03)

[RF07] – Solicitar sala de reunião

Descrição – Solicitar sala com todos os dados corretos. Qualquer funcionário com acesso à funcionalidade poderá solicitar uma sala, preenchendo os campos com os dados corretos.

[RF08] – Alterar uma solicitação com dados corretos

Descrição - Alterar uma solicitação com dados corretos. Qualquer funcionário com acesso à funcionalidade poderá alterar uma solicitação, preenchendo os campos com os dados corretos.

[RF09] – Cancelar uma solicitação

Descrição – Cancelar uma solicitação. Qualquer funcionário com acesso à funcionalidade poderá alterar uma solicitação, preenchendo os campos com os dados corretos.

Solicitação de equipamento (US04)

[RF10] – Solicitar equipamento

Descrição – Solicitar equipamento. Qualquer funcionário com acesso ao sistema poderá efetuar a solicitação mediante preenchimento com todos os dados corretos.

[RF11] – Alterar uma solicitação

Descrição – Alterar uma solicitação. Qualquer funcionário com acesso ao sistema poderá efetuar uma alteração de solicitação com todos os dados corretos.

[RF12] – Cancelar uma solicitação

Descrição – Cancelar uma solicitação. Qualquer funcionário com acesso ao sistema poderá efetuar o cancelamento.

Requisitos Não Funcionais

Não foram definidos requisitos não funcionais durante a execução do primeiro *Sprint*.

Regra de Negócio

A continuação segue a identificação e detalhamento das regras de negócio do sistema levantadas para as funcionalidades definidas para o primeiro *Sprint*.

Manutenção de salas de reunião (US01)

[RN01] – Cadastro de sala de reunião não pode ser efetuado sem informar todos os dados obrigatórios

Descrição: Cadastro de sala de reunião não pode ser efetuado sem informar todos os dados obrigatórios.

[RN02] – Cadastro de sala de reunião dentro de um horário e data existentes não deve ser efetuado

Descrição: Cadastro de sala de reunião dentro de um horário e data existentes não deve ser efetuado. Caso uma data ou horário solicitado não esteja disponível, é apresentada uma mensagem de indisponibilidade e uma nova data é solicitada.

[RN03] – Alteração de uma sala de reunião com dados corretos não deve ser concluída

Descrição: Alteração de uma sala de reunião com dados corretos não deve ser concluída.

[RN04] – Alteração de uma sala de reunião com dados incorretos não deve ser concluída

Descrição: Alteração de uma sala de reunião com dados incorreto não deve ser concluída.

[RN05] – Alteração de uma sala de reunião retirando dados obrigatórios não pode ser concluída

Descrição: Alteração de uma sala de reunião retirando dados obrigatórios não pode ser concluída. Mediante ausência de dados obrigatórios, a alteração não poderá ser concluída.

Manutenção de equipamentos (US02)

[RN06] – Cadastro de equipamentos não pode ser efetuado sem a informação de todos os dados obrigatórios

Descrição - Cadastro de equipamentos não pode ser efetuado sem a informação de todos os dados obrigatórios. Será feita validação nos campos Tipo de equipamento, modelo, número de série e observações.

[RN07] – Cadastro de equipamentos não pode ser efetuado em horário ou data já existente

Descrição: Cadastro de equipamentos não pode ser efetuado em horário ou data já existente. O cadastro do equipamento se faz possível somente na ausência de conflito entre horário e data pré-existente.

[RN08] – Alteração de equipamento não pode ser efetuada com dados incorretos

Descrição: Alteração de equipamento não pode ser efetuada com dados incorretos. Mediante apresentação de dados incorretos, será apresentada uma janela informando o campo a ser corrigido.

[RN09] – Alteração de equipamento não pode ser concluída quando dados obrigatórios são retirados

Descrição: Alteração de equipamento não pode ser concluída quando dados obrigatórios são retirados. Mediante retirada de dados obrigatórios, será apresentada uma janela informando sobre necessidade do campo retirado.

Solicitação de Salas de Reunião (US03)

[RN10] – Solicitação de sala não pode ser efetuada sem informar todos os dados obrigatórios

Descrição: Solicitação de sala não pode ser efetuada sem informar todos os dados obrigatórios. Caso os campos não tenham sido preenchidos de forma correta, ou na ausência do preenchimento de algum campo, não será possível concluir a solicitação.

[RN11] – Solicitação de sala dentro de um horário ou data já existente não pode ser concluída

Descrição: Solicitação de sala dentro de um horário ou data já existente não pode ser concluída. Na escolha de uma sala com data ou horário já reservados, será apresentada uma janela informando que para o período estipulado não é possível concluir, solicitando uma nova data.

[RN12] – Alteração de solicitação com dados incorretos não pode ser concluída

Descrição: Alteração de solicitação com dados incorretos não pode ser concluída. No preenchimento incorreto, será apresentada uma janela informando sobre o preenchimento incorreto, solicitando correção do item.

[RN13] – Alteração de solicitação retirando dados obrigatórios não pode ser concluída

Descrição: Alteração de solicitação retirando dados obrigatórios não pode ser concluída.

Solicitação de Equipamento (US04)

[RN14] – Solicitação de equipamento não deve ser efetuada sem informar os dados obrigatórios

Descrição: Solicitação de equipamento não deve ser efetuada sem informar os dados obrigatórios. No não preenchimento de dados obrigatórios, a solicitação não poderá ser efetuada, sendo apresentada uma mensagem de não preenchimento.

[RN15] – Solicitação de equipamento dentro de um horário ou data já existente não deve ser efetuada

Descrição: Solicitação de equipamento dentro de um horário ou data já existente não deve ser efetuada. Mediante seleção de equipamento para um período indisponível, será apresentada mensagem informando indisponibilidade e solicitando um novo período.

[RN16] – Solicitação com dados incorretos não pode ser efetuada

Descrição: Solicitação com dados incorretos não pode ser efetuada. Mediante apresentação de dados incorretos, não será dado andamento à solicitação, sendo apresentada mensagem de dados incorretos.

[RN17] – Alterar uma solicitação retirando dados obrigatórios

Descrição: Alterar uma solicitação retirando dados obrigatórios. Mediante alteração de retirada de dados obrigatórios, será apresentada mensagem de alteração não aceita.

PROTÓTIPO DA INTERFACE COM O UTILIZADOR

Constam os protótipos desenvolvidos no decorrer do primeiro *Sprint*.

Figura 1 – Tela Inicial

Figura 2 – Tela de Cadastro de Sala

Figura 3 – Tela de Cadastro de Equipamento.

PLANO DE TESTES DE ACEITAÇÃO

O plano de testes de aceitação define um conjunto de casos de teste das funcionalidades descritas como *user stories*.

As principais vantagens são:

- Garantir que os requisitos especificados são verificáveis;
- Esclarecer a especificação de requisitos, interpretando os requisitos e descobrindo ambigüidades e omissões.
- Servir como contrato concreto entre o fornecedor e o cliente.
- Os mesmos dados de exemplo usados nos protótipos podem ser usados nos testes de aceitação.

Numa abordagem baseada em *user stories* ou casos de utilização, os teste de aceitação relativos aos requisitos funcionais devem ser baseados em cenários de utilização. A continuação segue os testes de aceitação para cada uma das funcionalidades.

Manutenção de Salas de Reunião (US01)

[TA01] – Cadastrar uma sala com todos os dados corretos

Descrição: Cadastrar uma sala com todos os dados corretos. O administrador do sistema fará o cadastramento da sala mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA02] – Cadastrar uma sala de reunião sem informar todos os dados obrigatórios

Descrição: Cadastrar uma sala sem todos os dados obrigatórios. O administrador do sistema fará o cadastramento da sala de reunião omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA03] – Editar uma sala de reunião com dados corretos.

Descrição: Editar uma sala de reunião com dados corretos. O administrador do sistema fará a edição de dados cadastrados de uma sala de reunião com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA04] – Editar uma sala de reunião com dados incorretos.

Descrição: Editar uma sala de reunião com dados incorretos. O administrador do sistema fará a edição de dados cadastrados de uma sala de reunião com dados incorretos. O resultado deve ser edição não concluída.

[TA05] – Editar uma sala de reunião retirando dados obrigatórios.

Descrição: Editar uma sala de reunião retirando dados obrigatórios. O administrador do sistema fará a edição de dados cadastrados de uma sala de reunião retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA06] – Excluir uma sala de reunião cadastrada.

Descrição: Excluir uma sala de reunião já cadastrada. O administrador do sistema fará a seleção e exclusão de uma sala de reunião já cadastrada. O resultado deve ser exclusão concluída.

Manutenção de Equipamentos (US02)

[TA07] – Cadastrar um equipamento com todos os dados corretos

Descrição: Cadastrar um equipamento com todos os dados corretos. O administrador do sistema fará o cadastramento de equipamento mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA08] – Cadastrar um equipamento sem informar todos os dados obrigatórios

Descrição: Cadastrar um equipamento sem todos os dados obrigatórios. O administrador do sistema fará o cadastramento de equipamento omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA09] – Editar um equipamento com dados corretos.

Descrição: Editar um equipamento com dados corretos. O administrador do sistema fará a edição de dados cadastrados de um equipamento com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA10] – Editar um equipamento com dados incorretos.

Descrição: Editar um equipamento com dados incorretos. O administrador do sistema fará a edição de dados cadastrados de um equipamento com dados incorretos. O resultado deve ser edição não concluída.

[TA11] – Editar um equipamento retirando dados obrigatórios.

Descrição: Editar um equipamento retirando dados obrigatórios. O administrador do sistema fará a edição de dados cadastrados de um equipamento retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA12] – Excluir um equipamento cadastrado.

Descrição: Excluir um equipamento já cadastrado. O administrador do sistema fará a seleção e exclusão de um equipamento já cadastrado. O resultado deve ser exclusão concluída.

Solicitação de Sala de Reunião (US03)

[TA13] – Cadastrar uma solicitação de sala de reunião com todos os dados corretos

Descrição: Cadastrar uma solicitação de uma sala de reunião com todos os dados corretos. Um funcionário fará o cadastramento de uma solicitação de sala mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA14] – Cadastrar uma solicitação de sala de reunião sem informar todos os dados obrigatórios

Descrição: Cadastrar uma solicitação de sala sem todos os dados obrigatórios. Um funcionário fará o cadastramento de uma solicitação de sala de reunião omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA15] – Cadastrar uma solicitação de sala de reunião dentro de um horário e data já registrados.

Descrição: Cadastrar uma solicitação de sala de reunião dentro de um horário e data já existentes. Um funcionário fará o cadastramento de uma solicitação de sala de reunião selecionando um horário e data já existentes para a sala selecionada. O resultado deve ser cadastro não efetuado.

[TA16] – Editar uma solicitação de sala de reunião com dados corretos.

Descrição: Editar uma solicitação de sala de reunião com dados corretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de sala de reunião com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA17] – Editar uma solicitação de sala de reunião com dados incorretos.

Descrição: Editar uma solicitação de sala de reunião com dados incorretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de sala de reunião com dados incorretos. O resultado deve ser edição não concluída.

[TA18] – Editar uma solicitação de sala de reunião retirando dados obrigatórios.

Descrição: Editar uma solicitação de sala de reunião retirando dados obrigatórios. Um funcionário fará a edição de dados cadastrados de uma solicitação de sala de reunião retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA19] – Cancelar uma solicitação de sala de reunião cadastrada.

Descrição: Excluir uma solicitação de sala de reunião já cadastrada. Um funcionário fará a seleção e cancelamento de uma solicitação de uma sala de reunião já cadastrada. O resultado deve ser cancelamento concluído.

Solicitação de equipamento (US04)

[TA20] – Cadastrar uma solicitação de equipamento com todos os dados corretos

Descrição: Cadastrar uma solicitação de um equipamento com todos os dados corretos. Um funcionário fará o cadastramento de uma solicitação de equipamento mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA21] – Cadastrar uma solicitação de equipamento sem informar todos os dados obrigatórios

Descrição: Cadastrar uma solicitação de equipamento sem todos os dados obrigatórios. Um funcionário fará o cadastramento de uma solicitação de equipamento omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA22] – Cadastrar uma solicitação de equipamento dentro de um horário e data já registrados.

Descrição: Cadastrar uma solicitação de equipamento dentro de um horário e data já existentes. Um funcionário fará o cadastramento de uma solicitação de equipamento selecionando um horário e data já existentes para o equipamento selecionado. O resultado deve ser cadastro não efetuado.

[TA23] – Editar uma solicitação de equipamento com dados corretos.

Descrição: Editar uma solicitação de equipamento com dados corretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de equipamento com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA24] – Editar uma solicitação de equipamento com dados incorretos.

Descrição: Editar uma solicitação de equipamento com dados incorretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de equipamento com dados incorretos. O resultado deve ser edição não concluída.

[TA25] – Editar uma solicitação de equipamento retirando dados obrigatórios.

Descrição: Editar uma solicitação de equipamento retirando dados obrigatórios. Um funcionário fará a edição de dados cadastrados de uma solicitação de equipamento retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA26] – Cancelar uma solicitação de equipamento já cadastrado.

Descrição: Cancelar uma solicitação de equipamento já cadastrado. O administrador do sistema fará a seleção e cancelamento de uma solicitação de equipamento já cadastrado. O resultado deve ser cancelamento concluído.

Referências e leituras recomendadas

- Kurt Bittner, Ian Spence; Use Case Modelling , Addison-Wesley, 2003.
- IEEE recommended practice for software requirements specifications (IEEE Std 830-1998) Disponível em http://ieeexplore.ieee.org/.
- Gerald Kotonya and Ian Sommerville, Requirements Engineering: Processes and Tecnhiques, 1998

Documento de Requisitos do Sistema
Reserva de Salas de Reunião e Equipamentos
Versão 2.0

Histórico de Alterações

Data	Versão	Descrição	Autor
09/10/2009	1.0	Documento de Requisitos do sistema de reserva de salas de reunião e equipamentos, como resultado do Primeiro *Sprint*.	Eduardo Miyake
02/11/2009	2.0	Documento de Requisitos do sistema de reserva de salas de reunião e equipamentos, como resultado do Segundo *Sprint*.	César Sanz

INTRODUÇÃO

Este documento especifica os requisitos do sistema de reserva de salas de reunião e equipamentos, fornecendo aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação do sistema.

OBJETIVO

Especificar os requisitos levantados na etapa de exploração, classificando os tipos de requisitos e detalhando-os.

ÂMBITO

O Sistema proposto para a Empresa de Telecomunicações é uma ferramenta para controle de solicitações de salas de reunião e equipamentos.

Atualmente o processo de reserva é efetuado de forma manual, gerando transtorno aos usuários. Para reserva de uma sala o funcionário precisa se deslocar até a sala desejada e colocar seu nome e horário numa planilha. Na indisponibilidade do mesmo, será necessário dirigir-se à outra até localizar uma livre. Mais desorganizado é a solicitação de equipamentos. Não há um controle para o mesmo, havendo a possibilidade de no momento da reunião não haver o equipamento disponível.

O sistema atenderá este cenário, otimizando o dia-a-dia dos funcionários.

VISÂO GERAL DO DOCUMENTO

- Na seção 2 apresenta uma visão geral do sistema, caracterizando qual é o seu escopo e listando as funcionalidades requeridas.
- Na seção 3 são enumerados todos os requisitos funcionais, não funcionais e regras de negócio para os requisitos levantados.
- Na seção 4 são propostos os protótipos para o sistema de reserva de salas de reunião e equipamentos.
- Na seção 5 são definidos os testes de aceitação que serão utilizados para aprovação dos requisitos.

VISÃO GERAL DO SISTEMA

Nesta etapa serão descritos passos importantes que ajudaram no entendimento dos requisitos e a especificá-los posteriormente.

ABRANGÊNCIA DO SISTEMA

O sistema de Reserva de Salas de Reunião e Equipamentos permitirá aos usuários poder elaborar uma adequada gestão das salas de reuniões e equipamentos que fazem parte da empresa, evitando desconforto e ajudando a levar a gestão de uma forma simples mas eficaz.

Este documento pretende ser o documento de requisitos, análise e projeto do sistema de reserva de salas e equipamentos, abrangendo desde o levantamento de requisitos, passando por seu refinamento e discussão.

FUNÇÕES DO PRODUTO

As funcionalidades consideradas durante a execução do primeiro *Sprint* são as seguintes:

a) Manutenção de Equipamentos;

b) Manutenção de Salas de Reunião;

c) Solicitação de Sala;

d) Solicitação de Equipamentos;

e) Manutenção de Funcionários;

f) Relatórios de reserva de sala de reunião por semana;

g) Relatório de reserva de sala de reunião por funcionário;

h) Relatório de reserva de equipamento por funcionário.

ESPECIFICAÇÃO DE REQUISITOS

Neste capítulo os requisitos serão detalhados e divididos em Requisitos Funcionais, Requisitos Não-Funcionais e Regras de Negócio.

USER STORIES

As *user stories* criadas foram tratadas conforme tabela abaixo:

User Story	Descrição
US01	Manutenção de salas de reunião, onde um responsável por administrar o sistema poderá cadastrar alterar e excluir salas.
US02	Manutenção de equipamentos, onde um responsável por administrar o sistema poderá cadastrar alterar e excluir equipamentos.
US03	Solicitação de salas de reunião, onde qualquer funcionário com acesso ao sistema poderá cadastrar alterar e cancelar solicitação.
US04	Solicitação de Equipamentos, onde qualquer funcionário com acesso ao sistema poderá cadastrar alterar e cancelar solicitação.
US05	Manutenção de funcionário, onde se poderá fazer o cadastro, edição e exclusão de funcionários.
US06	Relatório de reserva de sala por semana permite fazer a consulta e impressão do relatório definido.
US07	Relatório de reserva de sala por funcionário permite fazer a consulta e impressão do relatório definido.
US08	Relatório de reserva de equipamento por funcionário, permite fazer a consulta e impressão do relatório definido.

Requisitos

Os requisitos gerados foram tratados da seguinte forma:

Requisitos Funcionais

Identificação e detalhamento das funcionalidades do sistema extraídas das *user stories.*

Manutenção de Salas de Reunião (US01)

[RF01] – Cadastrar uma sala com todos os dados corretos

Descrição: Cadastrar uma sala com todos os dados corretos. O administrador do sistema fará o cadastramento da sala mediante informações corretas (Nome da sala e Localização) .

[RF02]- Alterar uma sala de reunião com dados corretos

Descrição: Alterar uma sala de reunião com dados corretos. O administrador do sistema fará a edição da sala de reunião mediante informações corretas (Nome da sala e Localização).

[RF03] – Excluir cadastro de sala de reunião

Descrição: Excluir cadastro de sala de reunião. O administrador do sitema fará a exclusão de uma sala de reunião.

Manutenção de Equipamentos (US02)

[RF04] – Cadastrar equipamento

Descrição – Cadastrar um equipamento com todos os dados corretos. O administrador do sistema irá informar o tipo de equipamento (DataShow, Notebook, Retroprojetor ou quadro branco), o modelo, número de série e observações.

[RF05] – Editar equipamento com dados corretos

Descrição – Alterar equipamento com dados corretos. O administrador do sistema fará a alteração dos dados do equipamento que estiverem corretos, sendo necessário informar o motivo.

[RF06] – Excluir um equipamento

Descrição – Excluir um equipamento. O administrador do sistema fará a exclusão de um equipamento, sendo necessário informar o motivo.

Solicitação de Sala de Reunião (US03)

[RF07] – Solicitar sala de reunião

Descrição – Solicitar sala com todos os dados corretos. Qualquer funcionário com acesso à funcionalidade poderá solicitar uma sala, preenchendo os campos com os dados corretos.

[RF08] – Alterar uma solicitação com dados corretos

Descrição - Alterar uma solicitação com dados corretos. Qualquer funcionário com acesso à funcionalidade poderá alterar uma solicitação, preenchendo os campos com os dados corretos.

[RF09] – Cancelar uma solicitação

Descrição – Cancelar uma solicitação. Qualquer funcionário com acesso à funcionalidade poderá alterar uma solicitação, preenchendo os campos com os dados corretos.

Criar módulo de solicitação de equipamento (US04)

[RF10] – Solicitar equipamento

Descrição – Solicitar equipamento. Qualquer funcionário com acesso ao sistema poderá efetuar a solicitação mediante preenchimento com todos os dados corretos.

[RF11] – Alterar uma solicitação

Descrição – Alterar uma solicitação. Qualquer funcionário com acesso ao sistema poderá efetuar uma alteração de solicitação com todos os dados corretos.

[RF12] – Cancelar uma solicitação

Descrição – Cancelar uma solicitação. Qualquer funcionário com acesso ao sistema poderá efetuar o cancelamento.

Manutenção de Funcionário (US05)

[RF13] – Cadastrar funcionário

Descrição: Cadastrar um funcionário com todos os dados corretos. O administrador do sistema fará o cadastramento de um funcionário mediante informações corretas.

[RF14]- Alterar cadastro de funcionário

Descrição: Alterar um cadastro de funcionário com dados corretos. O administrador do sistema fará a edição de um cadastro de funcionário mediante informações corretas.

[RF15] – Excluir cadastro de funcionário

Descrição: Excluir cadastro de um funcionário. O administrador do sistema fará a exclusão do cadastro de um funcionário.

Relatório de sala de reunião por semana (US06)

[RF16] – Geração de relatório com opção de impressão

Descrição: Permitir a geração do relatório de sala de reunião por semana, onde se poderá identificar as reservas de sala de reunião por semana, cada dia com seu respectivo funcionário e hora da reserva.

[RF17] – Disponibilizar a exportação do relatório

Descrição: Permitir a exportação do relatório para formato pdf.

Relatório de sala de reunião por funcionário (US07)

[RF18] – Geração de relatório com opção de impressão

Descrição: Permitir a geração do relatório de sala de reunião por funcionário, onde cada funcionário poderá identificar suas reservas feitas.

[RF19] – Disponibilizar a exportação do relatório

Descrição: Permitir a exportação do relatório para formato pdf.

Relatório de equipamento por funcionário (US08)

[RF20] – Geração de relatório com opção de impressão

Descrição: Permitir a geração do relatório de equipamento por funcionário, onde cada funcionário poderá identificar suas reservas feitas..

[RF21] – Disponibilizar a exportação do relatório

Descrição: Permitir a exportação do relatório para formato pdf.

Requisitos Não Funcionais

Não foram definidos requisitos não funcionais durante a execução do primeiro *Sprint*.

Regras de Negócio

A continuação segue a identificação e detalhamento das regras de negócio do sistema levantadas para as funcionalidades definidas para o primeiro *Sprint*.

Manutenção de salas de reunião (US01)

[RN01] – Cadastro de sala de reunião não pode ser efetuado sem informar todos os dados obrigatórios

Descrição: Cadastro de sala de reunião não pode ser efetuado sem informar todos os dados obrigatórios.

[RN02] – Cadastro de sala de reunião dentro de um horário e data existentes não deve ser efetuado

Descrição: Cadastro de sala de reunião dentro de um horário e data existentes não deve ser efetuado. Caso uma data / horário solicitado não esteja disponível, é apresentada uma mensagem de indisponibilidade e uma nova data é solicitada.

[RN03] – Alteração de uma sala de reunião com dados corretos não deve ser concluída

Descrição: Alteração de uma sala de reunião com dados corretos não deve ser concluída.

[RN04] – Alteração de uma sala de reunião com dados incorretos não deve ser concluída

Descrição: Alteração de uma sala de reunião com dados incorreto não deve ser concluída.

[RN05] – Alteração de uma sala de reunião retirando dados obrigatórios não pode ser concluída

Descrição: Alteração de uma sala de reunião retirando dados obrigatórios não pode ser concluída. Mediante ausência de dados obrigatórios, a alteração não poderá ser concluída.

Manutenção de equipamentos (US02)

[RN06] – Cadastro de equipamentos não pode ser efetuado sem a informação de todos os dados obrigatórios

Descrição - Cadastro de equipamentos não pode ser efetuado sem a informação de todos os dados obrigatórios. Será feita validação nos campos Tipo de equipamento, modelo, número de série e observações.

[RN07] – Cadastro de equipamentos não pode ser efetuado em horário ou data já existentes

Descrição: Cadastro de equipamentos não pode ser efetuado em horário ou data já existentes. O cadastro do equipamento se faz possível somente na ausência de conflito entre horário e data pré existentes.

[RN08] – Alteração de equipamento não pode ser efetuada com dados incorretos

Descrição: Alteração de equipamento não pode ser efetuada com dados incorretos. Mediante apresentação de dados incorretos, será apresentado uma janela informando o campo a ser corrigido.

[RN09] – Alteração de equipamento não pode ser concluída quando dados obrigatórios são retirados

Descrição: Alteração de equipamento não pode ser concluída quando dados obrigatórios são retirados. Mediante retirada de dados obrigatórios, será apresentada uma janela informando sobre necessidade do campo retirado.

Solicitação de Salas de Reunião (US03)

[RN10] – Solicitação de sala não pode ser efetuada sem informar todos os dados obrigatórios

Descrição: Solicitação de sala não pode ser efetuada sem informar todos os dados obrigatórios. Caso os campos não tenham sido preenchidos de forma correta, ou na ausência do preenchimento de algum campo, não será possível concluir a solicitação.

[RN11] – Solicitação de sala dentro de um horário ou data já existente não pode ser concluída

Descrição: Solicitação de sala dentro de um horário ou data já existente não pode ser concluída. Na escolha de uma sala com data ou horário já reservados, será apresentada uma

janela informando que para o período estipulado não é possível concluir, solicitando uma nova data.

[RN12] – Alteração de solicitação com dados incorretos não pode ser concluída

Descrição: Alteração de solicitação com dados incorretos não pode ser concluída. No preenchimento incorreto, será apresentada uma janela informando sobre o preenchimento incorreto, solicitando correção do item.

[RN13] – Alteração de solicitação retirando dados obrigatórios não pode ser concluída

Descrição: Alteração de solicitação retirando dados obrigatórios não pode ser concluída.

[RN14] – Reserva de uma sala de reunião na mesma data e hora.

Descrição Um funcionário poderá fazer a reserva de uma única sala de reunião para uma mesma data e hora já registrada.

[RN15] – Reserva de sala de reunião como uma semana de antecedência.

Descrição: Uma reserva será permitida somente se houver uma semana de antecedência à data em que se está realizando a reserva atual, apresentando uma mensagem de restrição caso exceda o tempo.

Solicitação de Equipamento (US04)

[RN16] – Solicitação de equipamento não deve ser efetuada sem informar os dados obrigatórios

Descrição: Solicitação de equipamento não deve ser efetuada sem informar os dados obrigatórios. No não preenchimento de dados obrigatórios, a solicitação não poderá ser efetuada, sendo apresentada uma mensagem de não preenchimento.

[RN17] – Solicitação de equipamento dentro de um horário/ data já existentes não deve ser efetuado

Descrição: Solicitação de equipamento dentro de um horário/ data já existentes não deve ser efetuado. Mediante seleção de equipamento para um período indisponível, será apresentada mensagem informando indisponibilidade e solicitando um novo período.

[RN18] – Solicitação com dados incorreto não pode ser efetuada

Descrição: Solicitação com dados incorretos não pode ser efetuada. Mediante apresentação de dados incorretos, não será dado andamento à solicitação, sendo apresentada mensagem de dados incorretos.

[RN19] – Alterar uma solicitação retirando dados obrigatórios

Descrição: Alterar uma solicitação retirando dados obrigatórios. Mediante alteração de retirada de dados obrigatórios, será apresentada mensagem de alteração não aceita.

[RN20] – Controle de estoque de equipamento na reserva

Descrição: No momento da escolha do equipamento deverá ser apresentada uma mensagem de estoque esgotado caso não haja equipamento disponível.

Manutenção de funcionário (US05)

[RN21] – Cadastro de funcionário não pode ser efetuado sem informar todos os dados obrigatórios

Descrição: Cadastro de funcionário não pode ser efetuado sem informar todos os dados obrigatórios, o sistema deve validar pelo preenchimento destes campos.

[RN22] – Alteração de registro de funcionário com todos os dados preenchidos corretamente.

Descrição: Alteração de funcionário com falta de dados preenchidos corretamente deve ser concluída com sucesso, tendo que ser validado pelo sistema.

[RN23] – Alteração de um registro de funcionário com dados incorretos.

Descrição: Alteração de um registro de funcionário com dados incorretos não deve ser concluída, tendo que ser validado pelo sistema no momento da aceitação.

Relatório de sala de reunião por semana (US06)

[RN24] – Geração de relatório com dados requeridos

Descrição – Permitir a geração de relatório de reserva de sala de reunião por semana, indicando dias da semana, horário de inicio e fim, e funcionário com a reserva associada.

Relatório de sala de reunião por funcionário (US07)

[RN25] – Geração de relatório com dados requeridos

Descrição – Permitir a geração de relatório das reservas de sala de reunião por funcionário, indicando dias da semana, sala de reunião, horário de inicio e fim da reserva associada.

Relatório de equipamento por funcionário (US08)

[RN26] – Geração de relatório com dados requeridos

Descrição – Permitir a geração de relatório das reservas de equipamento por funcionário, indicando o equipamento, horário de inicio e fim da reserva associada.

PROTÓTIPO DA INTERFACE COM O UTILIZADOR

Neste capítulo constam os protótipos desenvolvidos no decorrer do primeiro e segundo *Sprint*s.

Figura 1 – Tela Inicial

Figura 2 – Tela de Solicitação de Sala

Figura 3 – Tela de Solicitação de Equipamento

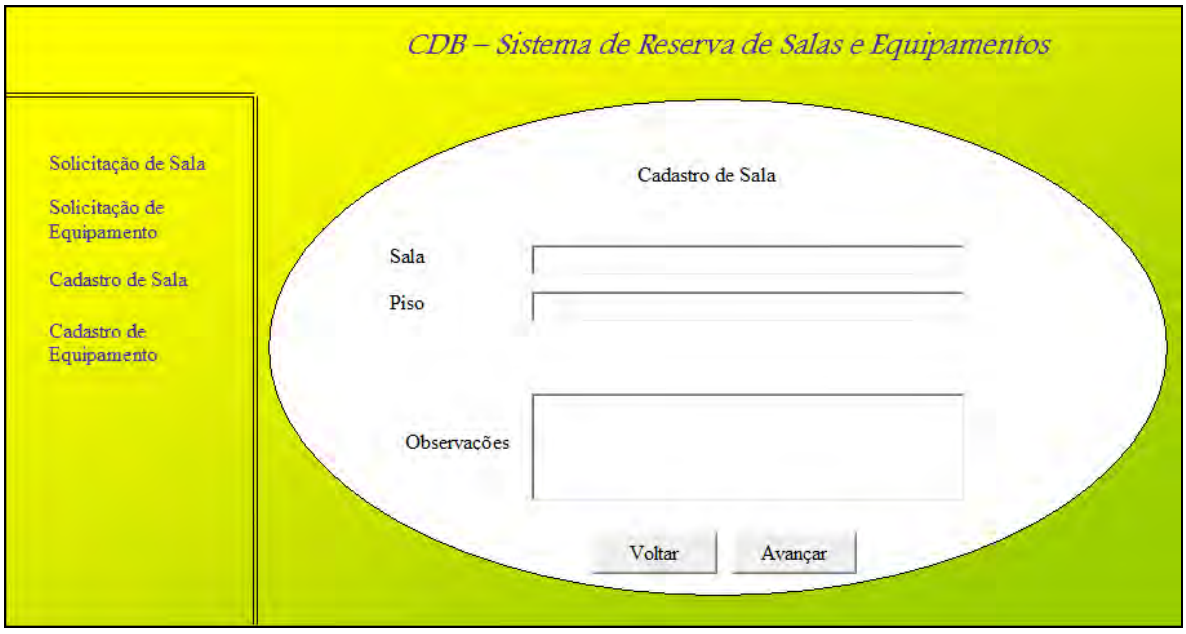

Figura 4 – Tela de Cadastro de Sala

Figura 5 – Tela de Cadastro de Equipamento.

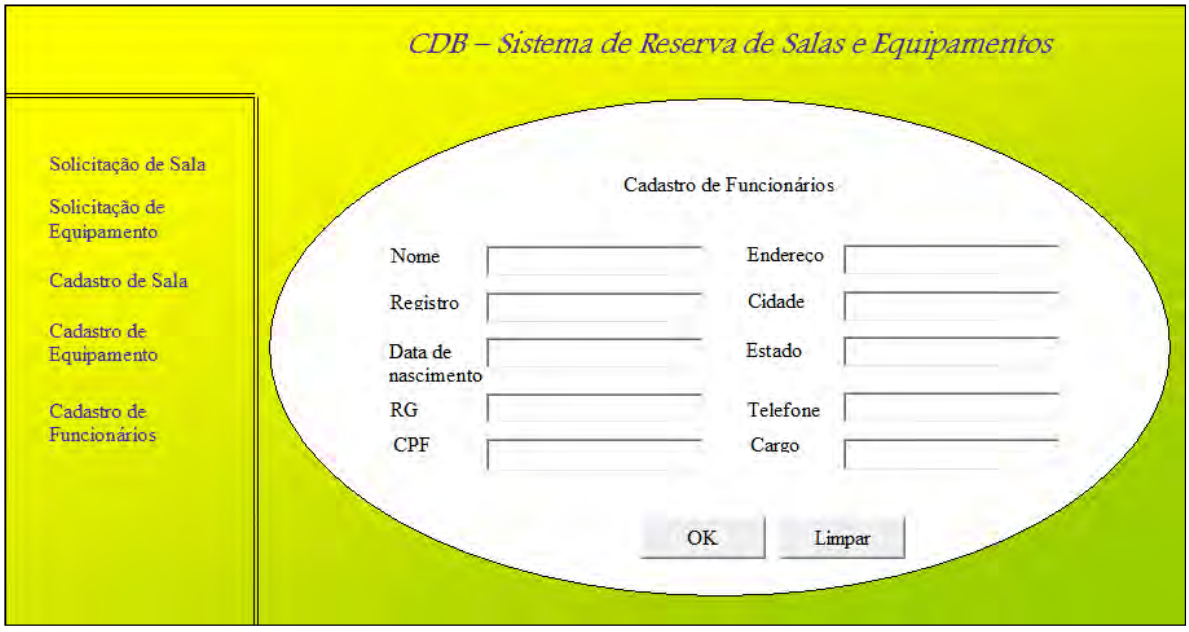

Figura 6 – Tela de Cadastro de Funcionário.

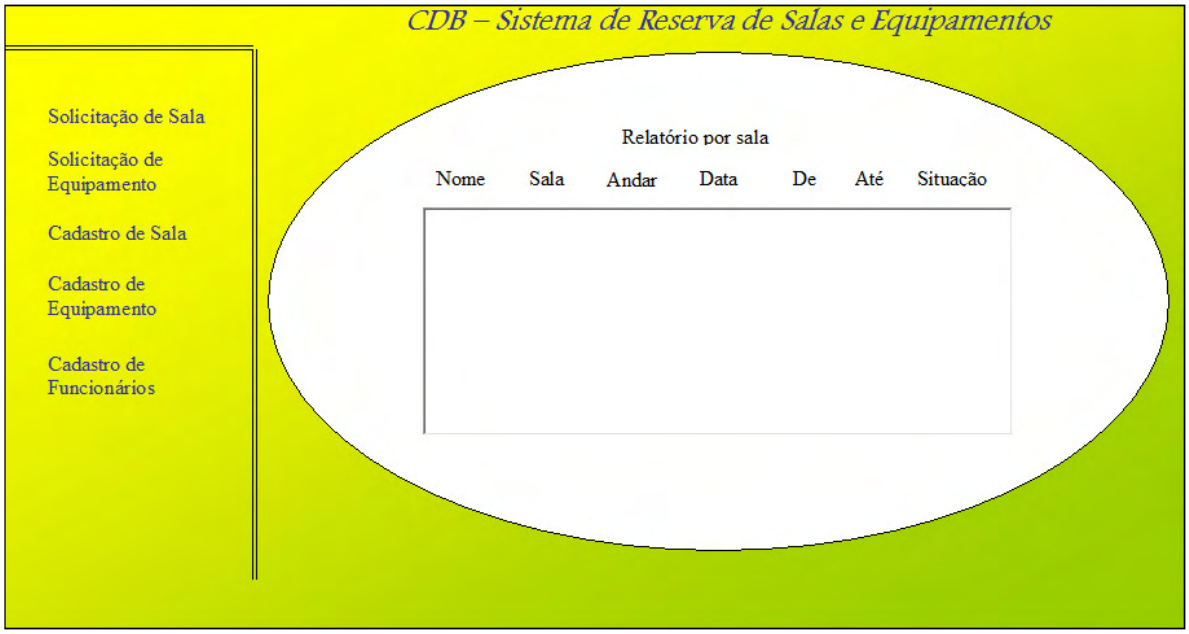

Figura 7 – Tela de Relatório por Sala

PLANO DE TESTES DE ACEITAÇÃO

O plano de testes de aceitação define um conjunto de casos de teste das funcionalidades descritas como *user stories*.

As principais vantagens são:

- Garantir que os requisitos especificados são verificáveis;
- Esclarecer a especificação de requisitos, interpretando os requisitos e descobrindo ambigüidades e omissões.
- Servir como contrato concreto entre o fornecedor e o cliente.
- Os mesmos dados de exemplo usados nos protótipos podem ser usados nos testes de aceitação.

Numa abordagem baseada em *user stories* ou casos de utilização, os teste de aceitação relativos aos requisitos funcionais devem ser baseados em cenários de utilização. A continuação segue os testes de aceitação para cada uma das funcionalidades.

Manutenção de Salas de Reunião (US01)

[TA01] – Cadastrar uma sala com todos os dados corretos

Descrição: Cadastrar uma sala com todos os dados corretos. O administrador do sistema fará o cadastramento da sala mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA02] – Cadastrar uma sala de reunião sem informar todos os dados obrigatórios

Descrição: Cadastrar uma sala sem todos os dados obrigatórios. O administrador do sistema fará o cadastramento da sala de reunião omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA03] – Editar uma sala de reunião com dados corretos.

Descrição: Editar uma sala de reunião com dados corretos. O administrador do sistema fará a edição de dados cadastrados de uma sala de reunião com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA04] – Editar uma sala de reunião com dados incorretos.

Descrição: Editar uma sala de reunião com dados incorretos. O administrador do sistema fará a edição de dados cadastrados de uma sala de reunião com dados incorretos. O resultado deve ser edição não concluída.

[TA05] – Editar uma sala de reunião retirando dados obrigatórios.

Descrição: Editar uma sala de reunião retirando dados obrigatórios. O administrador do sistema fará a edição de dados cadastrados de uma sala de reunião retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA06] – Excluir uma sala de reunião cadastrada.

Descrição: Excluir uma sala de reunião já cadastrada. O administrador do sistema fará a seleção e exclusão de uma sala de reunião já cadastrada. O resultado deve ser exclusão concluída.

Manutenção de Equipamentos (US02)

[TA07] – Cadastrar um equipamento com todos os dados corretos

Descrição: Cadastrar um equipamento com todos os dados corretos. O administrador do sistema fará o cadastramento de equipamento mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA08] – Cadastrar um equipamento sem informar todos os dados obrigatórios

Descrição: Cadastrar um equipamento sem todos os dados obrigatórios. O administrador do sistema fará o cadastramento de equipamento omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA09] – Editar um equipamento com dados corretos.

Descrição: Editar um equipamento com dados corretos. O administrador do sistema fará a edição de dados cadastrados de um equipamento com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA10] – Editar um equipamento com dados incorretos.

Descrição: Editar um equipamento com dados incorretos. O administrador do sistema fará a edição de dados cadastrados de um equipamento com dados incorretos. O resultado deve ser edição não concluída.

[TA11] – Editar um equipamento retirando dados obrigatórios.

Descrição: Editar um equipamento retirando dados obrigatórios. O administrador do sistema fará a edição de dados cadastrados de um equipamento retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA12] – Excluir um equipamento cadastrado.

Descrição: Excluir um equipamento já cadastrado. O administrador do sistema fará a seleção e exclusão de um equipamento já cadastrado. O resultado deve ser exclusão concluída.

Solicitação de Sala de Reunião (US03)

[TA13] – Cadastrar uma solicitação de sala de reunião com todos os dados corretos

Descrição: Cadastrar uma solicitação de uma sala de reunião com todos os dados corretos. Um funcionário fará o cadastramento de uma solicitação de sala mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA14] – Cadastrar uma solicitação de sala de reunião sem informar todos os dados obrigatórios

Descrição: Cadastrar uma solicitação de sala sem todos os dados obrigatórios. Um funcionário fará o cadastramento de uma solicitação de sala de reunião omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA15] – Cadastrar uma solicitação de sala de reunião dentro de um horário e data já registrados.

Descrição: Cadastrar uma solicitação de sala de reunião dentro de um horário e data já existentes. Um funcionário fará o cadastramento de uma solicitação de sala de reunião selecionando um horário e data já existentes para a sala selecionada. O resultado deve ser cadastro não efetuado.

[TA16] – Editar uma solicitação de sala de reunião com dados corretos.

Descrição: Editar uma solicitação de sala de reunião com dados corretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de sala de reunião com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA17] – Editar uma solicitação de sala de reunião com dados incorretos.

Descrição: Editar uma solicitação de sala de reunião com dados incorretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de sala de reunião com dados incorretos. O resultado deve ser edição não concluída.

[TA18] – Editar uma solicitação de sala de reunião retirando dados obrigatórios.

Descrição: Editar uma solicitação de sala de reunião retirando dados obrigatórios. Um funcionário fará a edição de dados cadastrados de uma solicitação de sala de reunião retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA19] – Cancelar uma solicitação de sala de reunião cadastrada.

Descrição: Excluir uma solicitação de sala de reunião já cadastrada. Um funcionário fará a seleção e cancelamento de uma solicitação de uma sala de reunião já cadastrada. O resultado deve ser cancelamento concluído.

[TA20] – Cadastrar uma solicitação de sala de reunião numa data e hora já existente Descrição: Um funcionário poderá fazer só uma única reserva de sala de reunião para uma data e hora definida. O resultado deve ser cadastro não efetuado.

[TA21] – Cadastrar uma solicitação de sala de reunião com duas semanas de antecedência Descrição: O usuário fará um cadastro de uma solicitação de uma sala de reunião com duas semanas de antecedência. O resultado será cadastro não deve ser efetuado.

[TA22] – Cadastrar uma solicitação de sala de reunião com uma semana de antecedência Descrição: O usuário fará um cadastro de uma solicitação de uma sala de reunião com uma semana de antecedência. O resultado será cadastro efetuado com sucesso.

Solicitação de equipamento (US04)

[TA23] – Cadastrar uma solicitação de equipamento com todos os dados corretos

Descrição: Cadastrar uma solicitação de um equipamento com todos os dados corretos. Um funcionário fará o cadastramento de uma solicitação de equipamento mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA24] – Cadastrar uma solicitação de equipamento sem informar todos os dados obrigatórios

Descrição: Cadastrar uma solicitação de equipamento sem todos os dados obrigatórios. Um funcionário fará o cadastramento de uma solicitação de equipamento omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA25] – Cadastrar uma solicitação de equipamento dentro de um horário e data já registrados.

Descrição: Cadastrar uma solicitação de equipamento dentro de um horário e data já existentes. Um funcionário fará o cadastramento de uma solicitação de equipamento selecionando um horário e data já existentes para o equipamento selecionado. O resultado deve ser cadastro não efetuado.

[TA26] – Editar uma solicitação de equipamento com dados corretos.

Descrição: Editar uma solicitação de equipamento com dados corretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de equipamento com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA27] – Editar uma solicitação de equipamento com dados incorretos.

Descrição: Editar uma solicitação de equipamento com dados incorretos. Um funcionário fará a edição de dados cadastrados de uma solicitação de equipamento com dados incorretos. O resultado deve ser edição não concluída.

[TA28] – Editar uma solicitação de equipamento retirando dados obrigatórios.

Descrição: Editar uma solicitação de equipamento retirando dados obrigatórios. Um funcionário fará a edição de dados cadastrados de uma solicitação de equipamento retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA29] – Cancelar uma solicitação de equipamento já cadastrado.

Descrição: Cancelar uma solicitação de equipamento já cadastrado. O administrador do sistema fará a seleção e cancelamento de uma solicitação de equipamento já cadastrado. O resultado deve ser cancelamento concluído.

[TA30] – Cadastrar uma solicitação quando não tiver estoque disponível.

Descrição: Cadastrar uma solicitação de equipamento quando não tiver estoque disponível. O resultado deve mostrar uma mensagem de “Estoque indisponível” e não deve realizar o cadastro.

Manutenção de funcionário (US05)

[TA31] – Cadastrar um funcionário com todos os dados corretos

Descrição: Cadastrar um funcionário com todos os dados corretos. O administrador do sistema fará o cadastramento de funcionários mediante informações corretas. O resultado deve ser cadastro efetuado com sucesso.

[TA32] – Cadastrar um funcionário sem informar todos os dados obrigatórios

Descrição: Cadastrar um funcionário sem todos os dados obrigatórios. O administrador do sistema fará o cadastramento de funcionário omitindo algum dado obrigatório. O resultado deve ser cadastro não efetuado.

[TA33] – Editar o registro de um funcionário com dados corretos.

Descrição: Editar o registro de um funcionário com dados corretos. O administrador do sistema fará a edição de dados cadastrados de um funcionário com os dados corretos. O resultado deve ser edição realizada com sucesso.

[TA34] – Editar o registro de um funcionário com dados incorretos.

Descrição: Editar o registro de um funcionário com dados incorretos. O administrador do sistema fará a edição de dados cadastrados de um funcionário com dados incorretos. O resultado deve ser edição não concluída.

[TA35] – Editar o registro de um funcionário retirando dados obrigatórios.

Descrição: Editar o registro de um funcionário retirando dados obrigatórios. O administrador do sistema fará a edição de dados cadastrados de um funcionário retirando dados obrigatórios. O resultado deve ser edição não concluída.

[TA36] – Excluir um funcionário cadastrado.

Descrição: Excluir um funcionário já cadastrado. O administrador do sistema fará a seleção e exclusão de um funcionário já cadastrada. O resultado deve ser exclusão concluída.

Relatório de sala de reunião por semana (US06)

[TA37] – Geração de relatório com opção de impressão.

Descrição: Permitir a geração de relatório de reserva de sala de reunião por semana, ativando a opção de impressão. O resultado deve ser relatório gerado com sucesso.

[TA38] – Disponibilizar exportação de relatório.

Descrição: Depois de gerado o relatório de reserva de sala de reunião por semana, permitir a exportação do relatório no formato pdf.

Relatório de sala de reunião por funcionário (US07)

[TA39] – Geração de relatório com opção de impressão.

Descrição: Permitir a geração de relatório de reserva de sala de reunião por funcionário, ativando a opção de impressão. O resultado deve ser relatório gerado com sucesso.

[TA40] – Disponibilizar exportação de relatório.

Descrição: Depois de gerado o relatório de reserva de sala de reunião por funcionário, permitir a exportação do relatório no formato pdf.

Relatório de equipamento por funcionário (US08)

[TA41] – Geração de relatório com opção de impressão.

Descrição: Permitir a geração de relatório de reserva de equipamento por funcionário, ativando a opção de impressão. O resultado deve ser relatório gerado com sucesso.

[TA42] – Disponibilizar exportação de relatório.

Descrição: Depois de gerado o relatório de reserva de equipamento por funcionário, permitir a exportação do relatório no formato pdf.

Referências e leituras recomendadas

- Kurt Bittner, Ian Spence; Use Case Modelling , Addison-Wesley, 2003.
- IEEE recommended practice for software requirements specifications (IEEE Std 830-1998) Disponível em http://ieeexplore.ieee.org/.
- Gerald Kotonya and Ian Sommerville, Requirements Engineering: Processes and Tecnhiques, 1998

APÊNDICE R - Tabela Comparativo dos Métodos Ágeis

TCC	Métodos Ágeis				
Desenvolvimento Incremental	**XP**	**SCRUM**	**FDD**	**ASD**	**MA**
Definição do esboço dos requisitos	Clientes escrevem as user stories	Definição do Product Backlog	Geração de artefatos para a documentação dos requisitos	Requisitos definidos durante as sessões JAD	Geração de : Documento de Requisitos Visão Geral Executiva Visão Geral do Projeto
Atribuição dos requisitos às iterações	Equipe e cliente definem as user stories que serão desenvolvidas nas iterações, que duram de 1 a 4 semanas.	Definição do Sprint Backlog. As sprints (iterações) duram no máximo 30 dias	As características são agrupadas, priorizadas e distribuídas aos responsáveis pelo seu desenvolvimento. As iterações duram no máximo 2 semanas.	Definição do número de iterações. As iterações duram de 4 a 8 semanas.	Refinamento do Documento - Visão Geral Executiva
Projeto da arquitetura do Sistema	Acontece paralelo a escrita da user stories, sem prescrever como deve ser projetado.	Desenvolver projeto baseado nos itens do Product Backlog	Construir diagrama de classes em UML para representar a arquitetura do sistema, e podem ser gerados diagramas de sequência da UML	Não prevê modo de projetar nessa metodologia	Geração de uma documentação do sistema contendo: Visão geral da arquitetura técnica Visão geral da arquitetura do negócio do sistema.
Desenvolver o Incremento do Sistema	Implementação das users stories que fazem parte da iteração corrente por duplas de programadores.	Implementação dos requisitos contemplados no Sprint Backlog para a Sprint corrente	Análise da documentação existente, geração de Diagrama de Sequência da UML, refinamento do modelo gerado nas atividades anteriores e implementação das características que serão desenvolvidas durante a iteração corrente	Implementação de requisitos que fazem parte da iteração corrente	Geração de uma Documentação de Operações, contemplando: As dependências em que o sistema está envolvido As diretrizes para a resolução de problemas. Documento contendo as decisões de projeto tomadas durante o desenvolvimento.
Validar o incremento	Os programadores executam os testes de unidade e os clientes executam os testes de aceitação	Não há definição de nenhum processo de validação	Os testes e inspeções são executados pelos próprios programadores após a implementação.	São verificadas a qualidade técnica e funcional do sistema	....
Integrar o incremento	A integração acontece paralelamente ao desenvolvimento das users stories	Atividade realizada ao final de cada Sprint	Atividade realizada após os testes no incremento	Não há definição de integração de incremento no ASD	....
Validar sistema	O sistema é disponibilizado ao cliente para que o mesmo realize validações	O cliente valida o sistema integrado em uma reunião no último dia da Sprint	Esta atividade ocorre através das inspeções e dos testes de integração	Não há definição de validação do sistema no ASD	....
Entrega final	Cliente satisfeito com o sistema	Somente se todos os itens do Product Backlog foram desenvolvidos	O sistema é entregue após todos os conjuntos de características serem implementados	Se todos os requisitos foram desenvolvidos	....