CENTRO PAULA SOUZA
GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Americana
Curso Superior de Bacharelado em
Análise de Sistemas e Tecnologia da Informação

A IMPORTÂNCIA E A NECESSIDADE DA ENGENHARIA DE REQUISITOS

Amaury Borges Souza

Americana, SP
2013

CENTRO PAULA SOUZA
GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Americana
Curso Superior de Bacharelado em
Análise de Sistemas e Tecnologia da Informação

A IMPORTÂNCIA E A NECESSIDADE DA ENGENHARIA DE REQUISITOS

Amaury Borges Souza
amaury.souza@fatec.sp.gov.br

Trabalho de conclusão de curso apresentado ao Curso Superior de Bacharelado em Análise de Sistemas e Tecnologia da Informação da Faculdade de Tecnologia de Americana – FATEC, como requisito parcial para obtenção do título de bacharel em Análise de Sistemas e Tecnologia da Informação sob orientação do Professor Especialista Fernando José Ignácio.

Área: Engenharia de Software

Americana, SP
2013

BANCA EXAMINADORA

Orientador: _____
Fernando José Ignácio
Especialista, Faculdade de Tecnologia de Americana.

Professor Convidado: _____
Rogério de Freitas
Especialista, Faculdade de Tecnologia de Americana.

Professor Convidado: _____
Nivaldo Tadeu Marcusso
Especialista, Faculdade de Tecnologia de Americana.

Dedico este trabalho aos meus amigos de trabalho, do Linux Technology Center - IBM, que reconheceram meu esforço e dedicação durante o desenvolvimento da monografia.

AGRADECIMENTOS

Agradecer em primeiro lugar a Deus por não me deixar desanimar e me dar forças para realização desse trabalho.

A todos os professores que contribuíram para a minha formação e no desenvolvimento desse trabalho em especial ao meu orientador Prof. Fernando Ignácio.

A minha família que soube entender minha ausência durante o desenvolvimento desse trabalho.

E por fim, aos meus amigos Tamara Campidelli, Tiago Altomare e Lucas Custódio Alves que fizeram parte dessa longa jornada e que ficaram marcados para sempre como “grandes amigos”.

RESUMO

Este trabalho tem como objetivo mostrar e alertar a todos os envolvidos em projetos de software, a importância e a necessidade das etapas que envolvem a engenharia de requisitos. Segundo uma pesquisa realizada no Instituto de Tecnologia de Massachusetts, foi provado um crescente custo em projetos com pouca especificação dos requisitos na fase inicial e também em projetos com pouca documentação durante a fase de desenvolvimento. Com essa pesquisa observa-se que, os profissionais da área focam mais no desenvolvimento do sistema, enquanto deixa de lado o processo da engenharia de requisitos, algo que parece simples, mas que na verdade é necessário para o sucesso do projeto e a satisfação do cliente. Quero deixar claro que se deve focar mais na documentação do projeto, ou seja, analisar, documentar e verificar os requisitos do sistema. Com base nessas atividades durante o processo de engenharia de requisitos, é possível comprovar a real importância da mesma durante o desenvolvimento do projeto. Existem técnicas que são necessárias para um alcançar um projeto de sucesso, como a prototipação de software, a modelagem do sistema e os cenários, entre outras, são essenciais, pois é através dessas técnicas que o alto custo e os prazos observados nos projetos de software podem ser explicados e reduzidos. Sabemos que se os requisitos não forem analisados e verificados durante a fase inicial do projeto, provavelmente isso implicará nas etapas posteriores, fazendo com que aumente o trabalho do time, surgindo custos mais altos, prazos ultrapassados e por fim a insatisfação do cliente para com o sistema desenvolvido.

Palavras-chave: engenharia de requisitos, projeto, custos, documentação.

ABSTRACT

This work aims to show and warn all those involved in software projects, the importance and necessity of steps involving requirements engineering. According to a survey conducted at the Massachusetts Institute of Technology, has been proven an increasing cost on projects with little specification of requirements in the initial phase and in projects with little documentation during the development phase. With this research we observed that the professionals focus more on development of the system, while leaving aside the requirements engineering process, something that seems simple, but actually it is necessary for project success and customer satisfaction . I want to make clear that it should focus more on project documentation, ie, analyze, document and verify the system requirements. Based on these activities during the requirements engineering process, it is possible to prove the real importance of it during the project development. There are techniques that are necessary for achieving a successful project, such as software prototyping, system modeling and scenarios, among others, are essential because it is through these techniques that the high cost and the time observed in software projects can be explained and reduced. We know that the requirements are not analyzed and verified during the initial phase of the project, it will probably involve in the later stages, so that increases the work of the team, appearing higher costs, exceeded deadlines and ultimately customer dissatisfaction toward system developed.

Key-words: requirements engineering, project, cost, documentation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

UML: Unified Modeling Language

JAD: Joint Application Design

RAD: Rapid Application Development

CASE: Computer Aided Software Engineering

T.I: Tecnologia da Informação

SUMÁRIO

1. INTRODUÇÃO

Como podemos perceber a tecnologia está sempre inovando, ganhando cada vez mais usuários e lojas focadas em algum serviço que a tecnologia oferece. Com essa grande demanda de pessoas querendo produtos novos, mais baratos, mais inteligentes, cresce também a fabricação de produtos softwares no mercado para suprir essa necessidade. Porém, com tanta informação no dia a dia, muitos profissionais quando vão projetar um software, acabam esquecendo-se de realizar detalhadamente algumas etapas do ciclo de vida do software, ou seja, a etapa de requisitos do sistema, parte essencial para a qualidade do projeto.

Os sistemas de software são conhecidos como ativos estratégicos para varias organizações. Visto que esses sistemas, em especial os sistemas de informação, têm uma função principal no apoio aos processos de negócio das empresas, é de grande importância que os sistemas funcionem seguindo os requisitos definidos. Neste contexto, uma importante tarefa no desenvolvimento de software é a identificação e o entendimento dos requisitos dos negócios que os sistemas vão apoiar (AURUM; WOHLIN, 2005).

O desenvolvimento de software é uma tarefa difícil por natureza. Um dos motivos para essa afirmação é que de fato não tem uma única solução para cada esboço de desenvolvimento. Além disso, a todo tempo vários profissionais estão envolvidos com pessoas, o que torna o sucesso do projeto muito relacionado à competência da equipe e a maneira como trabalham, e para complicar um pouco mais, diversas vezes não fazendo o uso de um processo bem focado e sistemático para auxiliar as tarefas do projeto. (SPÍNOLA, 2008).

De fato, o mercado de tecnologia está em alta para a construção de sistema de informação, e levando em consideração todo o processo de desenvolvimento de um software, é necessário ver a definição da Engenharia de Requisitos nesse processo, como explica Sommerville (2003):

> A engenharia de requisitos é um processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos do sistema. Existem quatro atividades genéricas de processo de engenharia de requisitos que são de alto nível, ou seja, o estudo da viabilidade do sistema, a obtenção e a análise de requisitos, a especificação de requisitos e sua documentação e, finalmente, a validação desses requisitos. (SOMMERVILLE, 2003)

Segundos estudos realizados por profissionais da área de T.I, mostra que uma enorme quantidade de projetos de softwares é cancelada por não levarem em consideração as necessidades dos clientes. De fato, não a nada que possa explicar tal fenômeno, mas uma coisa é certa, muitos trabalhos mostram requisitos sem correções, inconsistentes, sendo esses uns dos principais motivos de fracassos em projetos. Portanto, é necessário e importante seguir corretamente os processos dos requisitos bem como as técnicas e abordagens definidas pela engenharia de requisitos.

Para concluir esse processo, envolvendo regras de negócios, estudos de viabilidade, análise de requisitos entre outros. O presente trabalho busca mostrar essas características que são essenciais para um software ser aceito pelo cliente, focando no processo da engenharia de requisitos e em técnicas para manter os custos baixos e assim aumentando a qualidade do projeto.

1.1. Delimitação

Esse trabalho abordará o tema a importância e a necessidade da engenharia de requisitos, e mostrará como deve ser feito as atividades do processo de engenharia de requisitos, descrevendo também fatores importantes nesse processo, como o relacionamento com clientes e o papel do analista exercido durante o processo.

Abordará ainda uma breve introdução das técnicas mais utilizadas durante a engenharia de requisitos. Por último será mostrado o diagrama de casos de uso, na

qual, será modelado um sistema de locação de veículos, mostrando algumas das etapas de análise e levantamento de requisitos.

1.2. Objetivos

O objetivo dessa monografia é mostrar de maneira simples, a importância que tem cada atividade da engenharia de requisitos, e mostrar também a necessidade de algumas técnicas utilizadas no processo de requisitos. Será desenvolvido um estudo de caso, necessário para demonstrar um modelo de sistema, usando um diagrama da UML, com o objetivo de expor para o leitor uma forma mais clara de entender os requisitos do sistema.

Este trabalho foi realizado para conclusão do curso de Análise de Sistemas e Tecnologia da Informação e acima de tudo para um aprendizado profissional e pessoal, sobre conceitos relativos ao tema, à importância e a necessidade da engenharia de requisitos.

1.3. Objetivos específicos

Os objetivos específicos deste trabalho acadêmico iniciam com os conceitos das atividades da engenharia de requisitos, algumas técnicas de requisitos e terá:

a) Apresentação das diferenças entre os requisitos funcionais e não funcionais.

b) Esclarecimento da importância da comunicação com os clientes.

c) Apresentação básica sobre dados referentes aos custos de projetos de software.

1.4. Problematização

Atualmente como o mercado de tecnologia está em alta, o desenvolvimento de produtos de software representa um importante e crescente segmento da indústria de software. Desenvolver software é uma tarefa complexa por natureza, ao longo do projeto encontram-se pessoas que tem outra visão, positiva ou negativa dos possíveis requisitos do sistema, o que torna o sucesso do projeto dependente da competência da equipe que faz a análise e o levantamento dos requisitos do sistema.

As etapas de requisitos do processo de desenvolvimento de software são necessárias para o sucesso do projeto, é preciso seguir corretamente as regras de negócios da empresa, fazendo uma análise correta dos requisitos, da documentação e sempre manter reuniões com as pessoas envolvidas, como o programador, o analista, e o cliente, para que vejam como está a situação do mesmo, visto que, existe uma importância para cada etapa, para que não haja futuramente consequências em relação ao prazo e o custo final do sistema.

1.5. Justificativa

Segundo Pressman, (1995) a engenharia de requisitos é a etapa mais complexa do processo de desenvolvimento de software e a que mais absorve os defeitos difíceis de serem corrigidos em etapas posteriores do ciclo de vida do software. Para Kristel e Kang, (2003) se colocarmos um custo de $1,00 para cada alteração feita em um software durante a fase de análise e levantamento de requisitos, este custo cresce 1,5 a 6 vezes durante a fase de desenvolvimento e de 60 a 100 vezes, do custo original que foi atribuído ao cliente:

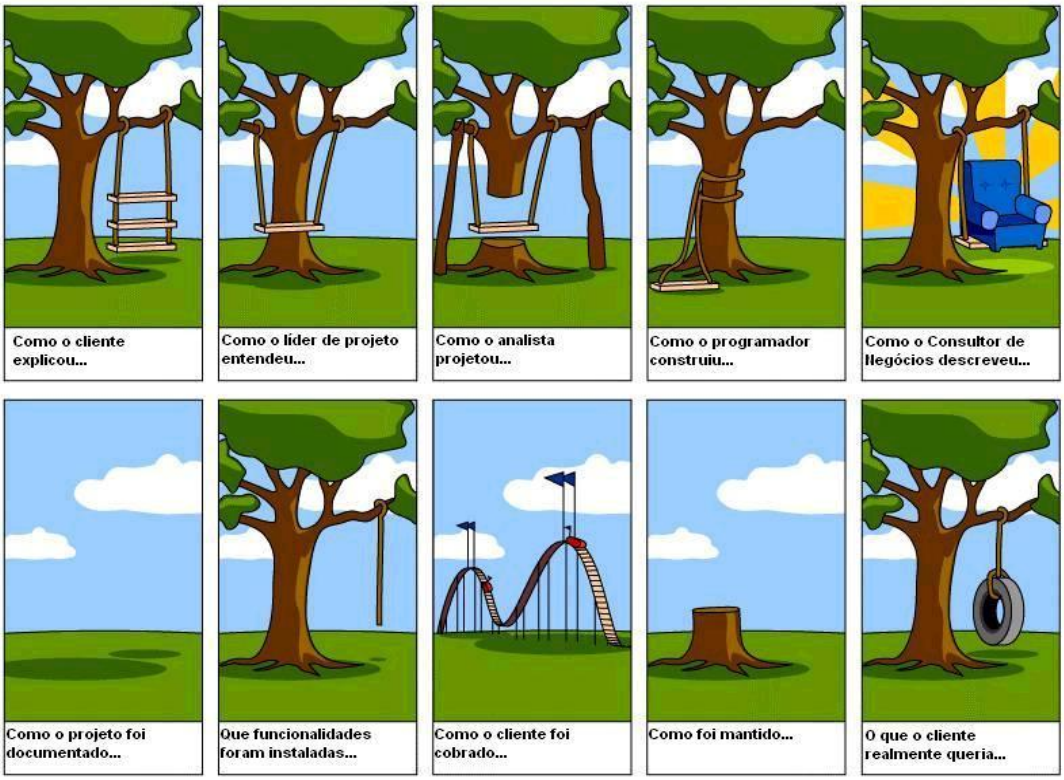

Figura 1 - Representação das etapas do Processo de desenvolvimento ligada à engenharia de requisitos
Fonte: SPÍNOLA (2008, p.)

A Figura 1 representa aquele velho desenho do balanço, baseado nas etapas do processo de engenharia de requisitos, exemplificando de maneira básica o quão é importante os requisitos do sistema, a implementação do software e o relacionamento entre os membros da equipe juntamente com o cliente. Considerando a relevância dessas etapas no processo de desenvolvimento de software, sem uma correta definição e validação dos requisitos do projeto é quase certo que o projeto terá o seu sucesso comprometido resultando em produtos de software de baixa qualidade, consequentemente frustrando as expectativas do cliente, e comprometendo as metas da empresa contratante.

1.6. Metodologia

A metodologia utilizada para realizar esse trabalho foi através de revisão de livros com pesquisas em fontes bibliográficas de autores renomados para entendimento dos conceitos dos assuntos abordados no trabalho, como o processo da engenharia de requisitos, por exemplo.

Essa monografia será desenvolvida com base em algumas das atividades da engenharia de requisitos, mostrando a importância e a necessidade de cada etapa no desenvolvimento do sistema, relacionado aos requisitos de software.

O trabalho foi dividido em etapas, primeira parte introdutória, seguida pelos conceitos referentes aos assuntos abordados. Por fim, o estudo de caso de um sistema de locação de veículos, com base no diagrama de casos de uso.

Para a realização do estudo de caso foi necessário modelar o sistema usando um software chamado Astah, para assim seguir os passos de análise e levantamento dos requisitos, por exemplo, o caso de uso Manter Clientes que será mostrado no capitulo cinco.

1.7. Estrutura do trabalho

Para a realização deste trabalho acadêmico, foi identificada a necessidade de dividir o estudo em etapas, primeira etapa o capitulo 1 que abordará superficialmente o tema em questão, descrevendo a introdução e os objetivos propostos do trabalho.

No capítulo 2 será feita uma apresentação básica sobre a importância e a necessidade dentro do contexto da engenharia de requisitos.

No capítulo 3 será feita de maneira mais aprofundada uma abordagem direcionada as atividades da engenharia de requisitos, demonstrando também o papel dos profissionais envolvidos no projeto, bem como a diferença entre os tipos de requisitos.

No capítulo 4 serão apresentados basicamente alguns dados sobre os custos, que devem ser levado em consideração, também será mostrado algumas técnicas

da engenharia de requisitos, essas técnicas serão abordadas de maneira simples a fim de mostrar para o leitor a importância dessas para o processo da engenharia de requisitos.

E finalmente, no capitulo 5 será feito um estudo de caso utilizando um diagrama da UML, muito usado na engenharia de requisitos, na qual, será projetado a documentação e a especificação dos requisitos de um sistema de locadora de veículos.

A última parte será feita uma conclusão, para demonstrar os resultados que foram adquiridos sobre o processo dos requisitos, as técnicas estudadas e estudo de caso com base nos objetivos definidos por esse trabalho acadêmico.

2. A IMPORTÂNCIA DA ENGENHARIA DE REQUISITOS

Pode se perceber que cada dia é mais constante encontrar produtos softwares inseridos em uma gama de outros produtos, como carros, telefones, sistemas empresariais, entre outros. Como de um produto precisa-se preço e qualidade, o software neste caso deve ter o patamar de qualidade exigido. De acordo com Leite (2007), está é exatamente a função da engenharia, encontrar sistemas de software com mais qualidade e possivelmente dentro de um valor aceitável com a qualidade desejada, otimizando a redução de custos.

De acordo com pesquisas, quanto mais tarde encontrar características erradas no desenvolvimento mais caro será o sistema. O processo da engenharia de requisitos mostra de diversas maneiras os fundamentos para um sistema, também registra e analisa sistematicamente os requisitos do sistema, melhorando a comunicação entre o cliente e o pessoal que desenvolve o projeto de software.

Segundo Ávila (2008), um dos objetivos mais adequados à engenharia de requisitos é a qualidade de software e a produtividade no desenvolvimento, tanto nas operações quanto na manutenção do sistema e garantir que profissionais tenham compreensão de alguns fatores do processo, como custos, prazos e possíveis níveis de qualidade.

Para Sommerville (2003), a engenharia de requisitos é uma parte da engenharia de software, que tem como objetivo analisar, especificar o processo de definição dos requisitos, essa análise é necessária porque todos os sistemas possui certo grau de complexidade, com isso, é essencial focar mais na parte de definição de requisitos, ou seja, aonde o projeto tem mais fragilidade.

Também se pode dizer que a engenharia de requisitos, fornece técnicas e ferramentas para que se possa fazer uma especificação mais a funda dos requisitos, possibilitando que o cliente entenda o que a equipe está planejando. Esse processo da engenharia de software é fundamental, pois ajuda os profissionais envolvidos no projeto a analisar melhor os problemas que possam aparecer no documento de requisitos e em outras partes do projeto.

Para Castro (2006), um projeto bem sucedido é aquele que se aplicam os seguintes aspectos:

- Cliente satisfeito;
- Colaborou na obtenção de metas para de regras de negócios;
- O sistema foi executado de acordo com os princípios adotados;
- Atendeu as especificações de qualidade e desempenho e
- Seguiu os prazos e custos propostos no começo.

É necessário atender também as necessidades dos clientes, a engenharia de requisitos auxilia o analista a identificar o sistema de forma mais simples, com técnicas, métodos e ferramentas para a análise e levantamento de requisitos.

No capítulo cinco será mostrada uma ferramenta muita utilizada por profissionais da área, que facilita a compreensão do sistema, fazendo com que o cliente possa entender os requisitos que foram especificados no projeto e que ajuda também os engenheiros de software a projetar o sistema.

3. ESCLARECENDO O QUE DEVE SER FEITO E COMO EXECUTAR CADA ATIVIDADE DO PROCESSO DA ENGENHARIA DE REQUISITOS

A engenharia de requisitos é um ramo da engenharia de software que compreende todas as atividades relacionadas com a definição dos requisitos de software de um sistema (KOTONYA, 1998).

O processo de engenharia de requisitos envolve muita criatividade, interação de pessoas, conhecimento para transformar as informações sobre a organização, as regras de negócios e o próprio sistema em possíveis documentos e modelos que mostrem o caminho para o desenvolvimento de software (SOMMERVILLE, 2003). A engenharia de requisitos é necessária, pois possibilita levantar uma prévia do custo e de tempo de maneira simples, mas precisas e também para melhorar os requisitos de software do sistema.

Os processos de engenharia de requisitos podem variar de acordo com a empresa, em função das características do projeto. A definição de um certo processo à organização traz vários benefícios, visto que uma boa descrição do mesmo fornecerá orientações e reduzirá o possível esquecimento de uma execução superficial. Mas por começo não há necessidade de falar em processos exatos, ou executar algum e implementá-lo na organização. Ao contrário disso, as empresas devem começar com um processo básico e ajustá-lo para um mais estudado, que de fato seja adequado as reais necessidades da empresa (SOMMERVILLE, 2003).

Segundo Kotonya (1998), pode se verificar na figura 2 que os *inputs* para o processo de engenharia de requisitos incluem: informações sobre sistemas já existentes, necessidades dos *stakeholders*, padrões da organização, regulamentações e informações de domino da aplicação. Todas estas informações são utilizadas para a realização das etapas do processo. Como resultado *output* obtêm os requisitos acordados, uma especificação de requisitos e modelos do sistema. Este modelo do processo mostra que para a realização das etapas do processo de engenharia de requisitos, fatores humanos e técnicos têm de ser tratados corretamente, fazendo com que dessa forma cada resultado do processo, seja o mais completo e claro possível:

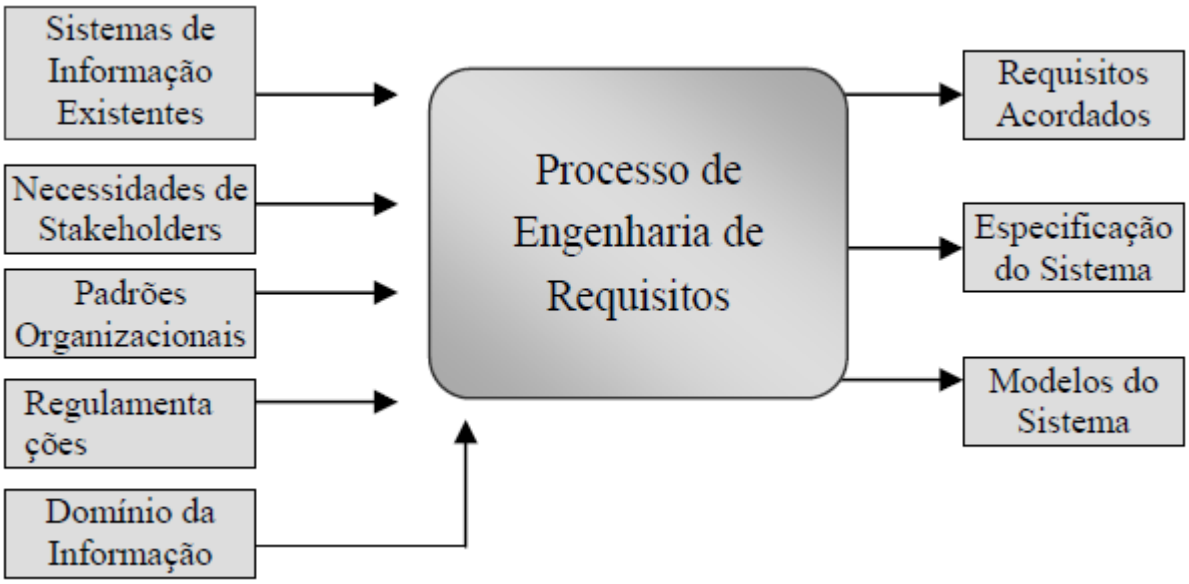

Figura 2 - Inputs e outputs do processo de engenharia de requisitos
Fonte: Kotonya et al (1997)

O processo de engenharia de requisitos é composto basicamente pelas seguintes atividades:

- Estudos de viabilidade
- Análise e levantamento de requisitos;
- Especificação de requisitos;
- Validação de requisitos;
- Gerenciamento de requisitos.

Segundo Sommerville (2003), essas etapas não seguem rigorosamente uma sequência correta de realização no processo da engenharia. Tipicamente inicia-se com o estudo de viabilidade dos requisitos, juntamente com uma análise dos requisitos. Posteriormente, são documentados e validados. Paralelamente a todas estas atividades é realizado o gerenciamento de requisitos, no qual objetiva-se controlar a evolução e as alterações nos requisitos.

As maiores dificuldades surgem na execução destas atividades. Diversos fatores dificultam o desenvolvimento de documentos de requisitos que realmente satisfaçam os usuários. Alguns problemas com requisitos encontrados durante o processo de engenharia de requisitos são relatados em (KOTONYA ET AL, 1998):

- Os requisitos não refletem as reais necessidades do cliente em relação ao sistema a ser desenvolvido;
- Os requisitos são inconsistentes ou incompletos.
- É dispendioso fazer mudanças após os requisitos terem sido acordados entre as partes (cliente e equipe de desenvolvimento);
- É comum ocorrer interpretação errônea entre clientes e equipe de desenvolvimento.

Desta forma, para atender corretamente os requisitos dos usuários, é necessário que as etapas do processo de engenharia de requisitos, sejam realizadas de forma clara e corretas possíveis e com um suporte computacional adequado. Isto leva a processos com maior maturidade, o que permite que uma organização obtenha softwares com qualidades.

3.1. O gerenciamento de projetos

O gerenciamento de projetos é um grande e importante fator que esta envolvido em todos os projetos de software. E é interessante destacar esse tema, pois, está ligado com a engenharia de requisitos, que como visto foca nos custos e prazos de produtos softwares.

Segundo Sommerville (2003), o gerenciamento é uma essencial diferença entre o desenvolvimento profissional e a implementação amadora. Dentro desse processo existe também o gerente de projetos, que é de fundamental importância para projetar e controlar as atividades de gerenciamento. Para Sommerville (2003), os gerentes de projetos tem o papel de programar e verificar todo o processo de desenvolvimento do projeto de software. São responsáveis por assegurar que este seja feito dentro de padrões normas e posteriormente, se este esta dentro do prazo e do custo acordado.

3.1.2 Atividades do gerenciamento

É difícil compreender uma etapa ou descrição de um trabalho efetivo de um gerente de projetos. O trabalho não é padrão, ele muda conforme as regras de negocio da empresa, padrões, etc. Mas em geral, existe algumas atividades que são de responsabilidade dos gerentes ao longo de determinados projetos:

- Elaboração das propostas;
- Planejamento e programação de projetos;
- Custo do projeto e
- Monitoramento e revisões.

Segundo Sommerville (2003), quando se começa um projeto é interessante focar na elaboração de proposta, para fazer o projeto. Essa proposta é importante, pois, mostra os objetivos e todo o desenvolvimento do projeto. Já o planejamento do projeto tem o objetivo de encontrar os marcos, os documentos a serem feitos durante o projeto. Os custos é uma atividade importante, pois, busca mostrar todo o orçamento dos recursos utilizados pelo projeto. Quanto ao monitoramento do projeto, está é uma atividade constante. É papel do gerente de projetos fazer um monitoramento da situação do projeto e sempre comparar custos e etapas feitas com as que antes tinham sido elaborados.

Outro fator importante para auxiliar nessas atividades é o recrutamento de pessoas para trabalhar na equipe do gerente de projetos. É ele quem faz essas escolhas, o ideal é que a equipe seja experiente e esteja apta a trabalhar no projeto, mas às vezes, os gerentes têm de contentar com uma equipe não tão experiente.

Para Sommerville (2003), algumas razoes são:

O orçamento do projeto pode não ajudar suficientemente para contratar esses profissionais bem pagos;

Um time com a experiência adequada pode não estar livre no momento do projeto e

Um último e importante fator, é que muitas vezes a própria organização queira treinar seus funcionários para o projeto, nesse caso, seria uma equipe come menos

experiência, para que com essa possibilidade aprenda mais com o gerente de projetos e o projeto em si.

3.2. Classificação dos requisitos

Para Sommerville (2003), os requisitos de software são classificados como funcionais, não funcionais e requisitos de domínio, segundo ele, uma das melhores definições para cada tipo de requisitos, pode ser:

> "Requisitos funcionais: são as declarações de funções que o sistema deve fornecer como o sistema deve reagir a entradas especificas e como deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também ser explicitamente declarar o que o sistema não deve fazer.
> Requisitos não funcionais: são restrições sobre os serviços ou as funções oferecidos pelo sistema. Entre eles destacam-se restrições de tempo, restrições sobre o processo de desenvolvimento, padrões, entre outros. Requisitos de domínio: são requisitos que se originam do domínio de aplicação do sistema e que refletem características desse domínio. Podem ser requisitos funcionais ou não funcionais."

3.2.1 Requisitos funcionais

Segundo Soares (2006), tais requisitos mostram as funcionalidades que o sistema possa fornecer, no momento em que estiver pronto, como entradas, saídas e exceções.

As especificações de requisitos funcionais devem ter característica como, bem completa e consistente, todos os métodos pedidos pelo usuário devem estar definidos e essas definições não podem ser contraditórias. Na prática, de maneira geral a maioria dos softwares complexos e grandes, é quase impossível conseguir a consistência dos requisitos.

Um dos motivos são as próprias ideias da equipe, que baseado no encontro de problemas que são descobertos em etapas seguintes do processo, deve se fazer novamente a correção do documento dos requisitos.

Um ponto importante sobre esse tipo de requisito, é que este não deve ser misturado com outros tipos de tecnologia da informação, como, linguagens de programação, SGBD e bibliotecas em geral.

3.2.2 Requisitos não funcionais

Para Sommerville (2003), os requisitos não funcionais são aqueles que não estão ligados às funções especificas pelo sistema ou software.

De maneira geral os requisitos não funcionais abordam o sistema inteiro de software, e não apenas a determinados processos, ou características do sistema. Tanta complexidade mostra que de fato estes requisitos são mais importantes que os requisitos funcionais, que no caso abordam características individuais do sistema.
Os requisitos não funcionais são descobertos seguindo a necessidade do cliente, em casos específicos, com orçamentos, politicas da empresa, etc.

3.3. Estudo de viabilidade

Segundo Sommverville (2003), a primeira etapa a ser realizada no processo de engenharia de requisitos é o estudo de viabilidade, que significa um esclarecimento inteiro do sistema e também como o mesmo será utilizado pela empresa. Este processo é importante porque, é através dele que se pode verificar se compensa ou não fazer o processo de engenharia de requisitos como também o de desenvolvimento do sistema.

Também é levado em conta se o sistema contribuirá para com os objetivos da empresa, visto que, sem a contribuição dos objetivos para a organização, tais objetivos não terão valor algum para a empresa.

Para a preparação de um estudo de viabilidade, consiste em avaliar e coletar informações e fazer relatórios. A primeira fase que é a de avaliação deve-se levar em conta algumas perguntas básicas para o andamento do processo, como:

- O sistema trará vantagem para os objetivos da empresa?
- O sistema tem a capacidade de ser projetado/implementado seguindo a tecnologia atual, obedecendo a fatores que envolvem custos e prazos?
- O sistema pode interagir com demais sistemas em operação?

3.4 Levantamento e análise dos requisitos

Segundo Sommerville (2003), logo na sequência, a próxima atividade do processo de engenharia de requisitos é o levantamento e análise dos requisitos. Nesta etapa toda equipe de desenvolvimento do sistema faz um breve encontro com clientes e usuários do possível sistema, para adquirir informações sobre as tecnologias envolvidas no software em relação ao desempenho do mesmo, como hardware, software e os serviços que serão aceitos.

Nessa atividade de levantamento e análise dos requisitos existem vários tipos de pessoas que podem se envolver na empresa. O termo *stakeholder* pode ser entendido como, qualquer pessoa que terá algum contato com os requisitos do sistema, conforme Figura 3:

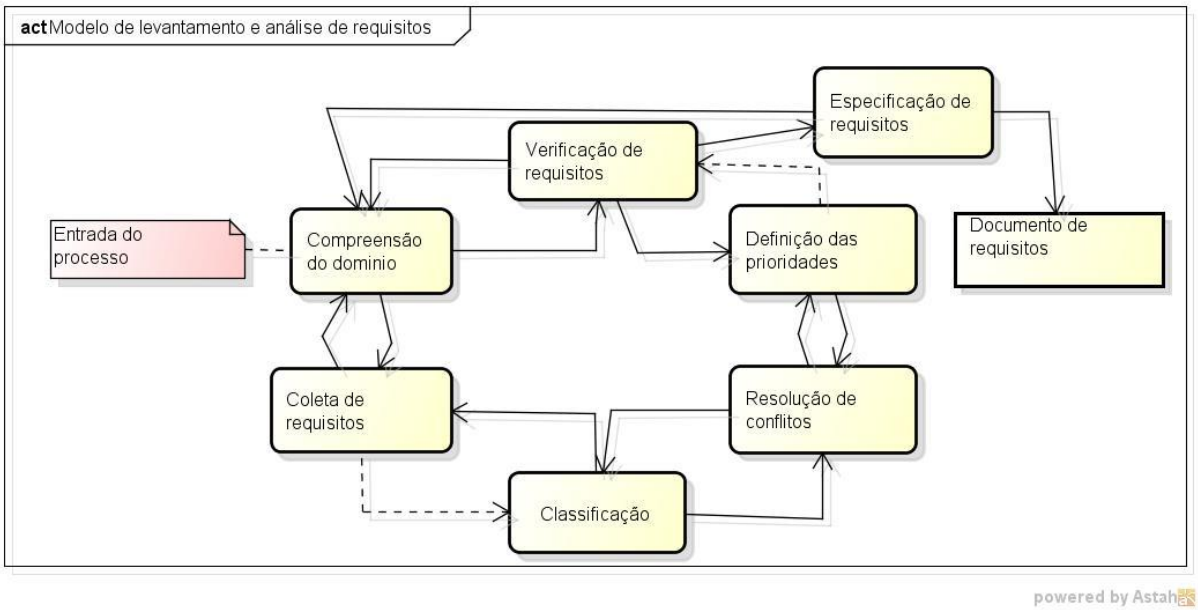

Figura 3 - Representação do modelo de levantamento e análise dos requisitos (Autoria própria).

Os *stakeholders* englobam diversas pessoas como os usuários finais do sistema, os engenheiros de software que implementam o projeto, os gerentes de negócios, os especialistas, os analistas, todos esses estão relacionados com o termo *stakeholders.*

Segundo Sommerville (2003), a Figura 3 apresenta claramente um modelo básico do processo de levantamento e análise dos requisitos. De acordo com a Figura 3 fica notável que esse processo fornece um retorno para cada atividade de levantamento dos requisitos, começando pela compreensão do domínio e finalizando na verificação de requisitos.

A seguir será abordado o papel do analista de sistemas, em relação ao processo da engenharia de requisitos, que tem grande importância no mesmo e também serão mostradas três técnicas de levantamento e análise de requisitos que são essenciais para o sucesso do projeto.

3.4.1 O Analista de Sistemas

Como mostra a Figura 4, segundo Pressman (1995), o papel do Analista segue esse modelo, na qual, o mesmo coordena e projeta cada atividade

relacionada à análise de requisitos. Durante essa etapa ele deve fazer a comunicação com os usuários para esclarecer o que o sistema deverá ter como características. Ele tem um papel importante no desenvolvimento, mais focado na especificação dos requisitos, e também participando de revisões que venham a acontecer:

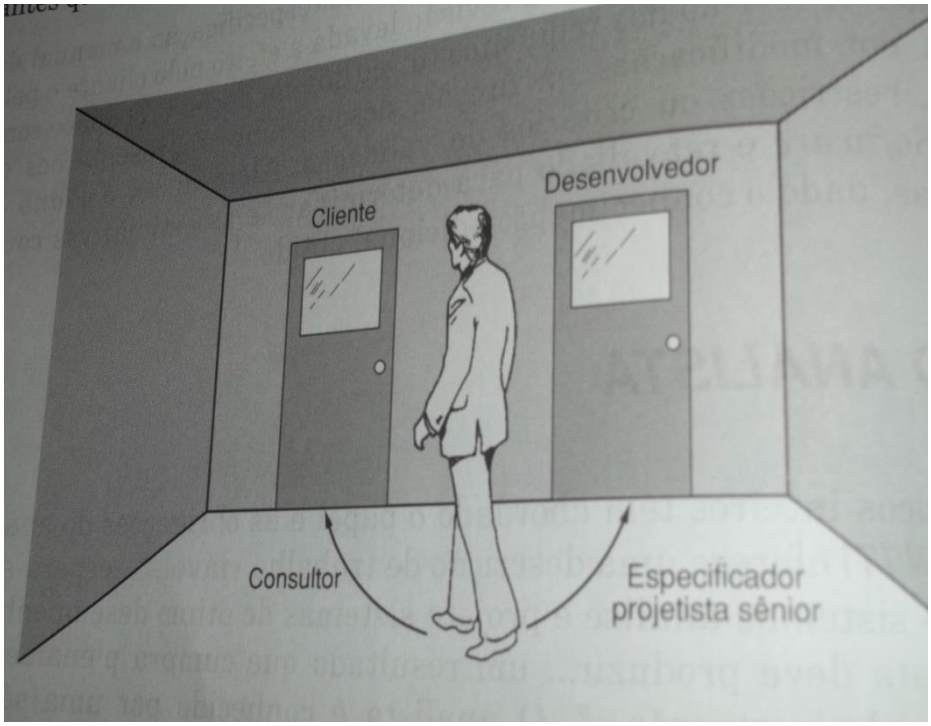

Figura 4 - Representação da função do analista. Fonte: Pressman (1995).

Para Pressman (1995), é crucial perceber que o analista deve entender todos os paradigmas da engenharia de software e também conhecer as etapas da engenharia de software que são executadas para esses paradigmas.

O papel do Analista é fundamental para a atividade de levantamento e análise dos requisitos, bem como, para todas as atividades, pois é ele quem mantem o contato com os possíveis clientes/usuários do sistema, para que com isso possa ser feita um processo dos requisitos de forma correta, assim, tornando o projeto mais eficiente e com uma possível aceitação do cliente final.

3.4.2. A necessidade do relacionamento com os clientes

Para o sucesso de um projeto de software, é necessária também a participação do cliente juntamente com o profissional da área. É um fator que caminha junto com as atividades da engenharia de requisitos, pois a fase de requisitos necessita do envolvimento do cliente para o entendimento de suas necessidades.

Segundo Pádua (2003), umas das principais causas de atrasos e custos elevados em projetos de software, foi diagnosticada como sendo a falta de informação sobre clientes, requisitos inconsistentes e incorretos, esses pontos negativos podem ser corrigidos através de algumas técnicas de envolvimento de clientes e usuários.

Para Pádua (2003), diversos problemas em projetos de software são iniciados por expectativas irreais, ainda mais nos prazos dos projetos. Um estudo mostrou que 11% de quase todos os projetos são impactados por expectativas irreais, o fornecimento de expectativas em relação a prazos, segundo o estudo é um dos erros mais fáceis de serem cometidos.

A Figura 5 a seguir mostra um possível encontro de pessoas, para uma breve reunião:

Figura 5 - Representação de um encontro entre o cliente e analista.

3.4.3. Levantamento orientado a ponto de vista

Para Sommerville (2003), para todo sistema, independente do tamanho do mesmo, em vários casos existe diversos usuários finais. Vários *stakeholders* preferem determinar os requisitos do sistema, ou seja, essa relação mostra que de qualquer forma o sistema sendo básico, há vários tipos de pontos de vista que devem ser estudados.

Os processos orientados a pontos de vista na engenharia de requisitos interpretam esses pontos de vista e utilizam para organizar o processo de levantamento e os próprios requisitos (SOMMERVILLE, 2003).

Listo abaixo o que pode ser ponto de vista em relação a alguns métodos:

- Para Sommverville (2003), uma fonte ou “filtro” de dados, nesse caso, os pontos de vista são focados mais na produção ou consumo dos dados. A análise dos requisitos consiste em diagnosticar esses pontos de vista, mostrando quais dados são produzidos e que tipo de processamento é feito.
- Um *framework* de representação, um ponto de vista, é visto como um tipo de modelo de sistema. Por exemplo, analistas implementam um modelo de relacionamento de entidades e um modelo de estados. Isso deixa claro, pois cada abordagem da análise encontra aspectos sobre o determinando sistema SOMMERVILLE (2003).

Cada um desse modelo de ponto de vista mostra diferentes aspectos positivos como negativos. Os pontos de vista como fonte de dados ou “filtros” e as representações são importantes para a descoberta de conflitos entre os possíveis requisitos. SOMMERVILLE (2003). Com isso, tais modelos não são adequados para a escolha e estruturação no processo de análise dos requisitos, visto que, não existe relações básicas entre os pontos de vista e os *stakeholders* envolvidos no sistema.

3.4.4. Cenários

Nessa outra técnica de levantamento de requisitos, percebe-se que logo pelo nome, podemos entender que pode ser uma modelagem, modelo de um sistema.

Segundo Sommerville (2003), os usuários podem ser mais úteis para incrementar características a um determinado esquema de requisitos. São descrições sobre exemplos de uma interação. No levantamento de requisitos os cenários aborda um número de interações.

A seguir será mostrado o que o cenário pode incluir do sistema, como:

- Dados do estado do sistema ou o começo do cenário;
- Dados de eventos do cenário;
- Dados do que poderá ser encontrado de errado e dados de como corrigir e lidar com isso, e
- Dados de estado do sistema quando o cenário estiver terminando.

No processo de levantamento e analise dos requisitos os analistas trabalham juntos com os *stakeholders* para encontrar possíveis cenários e conseguir algumas características desses cenários. (SOMMERVILLE, 2003).

3.4.4.1. A importância dos Use Cases

São técnicas ligadas aos cenários para a descoberta de requisitos. Atualmente os use-cases se transformou em uma possível notação da UML (UnifiedModeling Language), serve para esclarecer possíveis modelos de sistemas orientados a objetos (PRESSMAN, 1995).

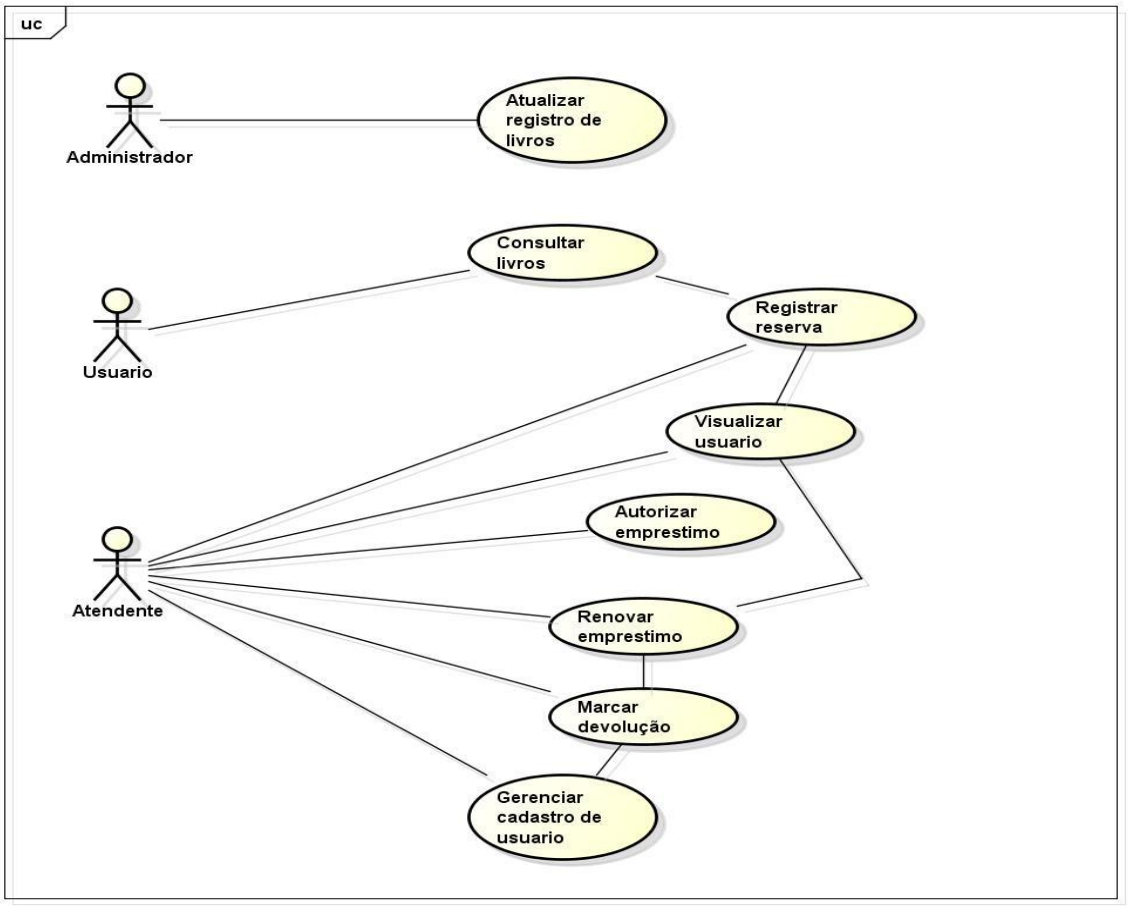

Figura 6 - Representação de um use-case de biblioteca (Autoria própria).

Na Figura 6 podemos ver um use-case de biblioteca que mostra os agentes, que são os bonecos e as classes que estão representadas pelas elipses seguidos com um nome. Os use-cases representa a interações que o sistema oferece de acordo com os requisitos do software.

3.4.5. Etnografia

Para Sommerville (2003), a etnografia consiste em uma técnica de vistoria que é utilizada para esclarecer os requisitos sociais e organizacionais da empresa.

Esse processo é observado todo dia e também são analisadas as tarefas que as pessoas estão envolvidas. A etnografia ajuda ainda a encontrar requisitos de software implícitos, na qual, o mesmo mostra os processos e reais e não os processos formais na qual, as pessoas trabalham (SOMMERVILLE, 2003).

A etnografia é usada mais precisamente para encontrar dois tipos de requisitos do sistema, como citado por Sommerville (2003), na qual, é interessante mostrar, pois retrata de forma clara o objetivo e a necessidade da etnografia:

> Os requisitos derivados de maneira como as pessoas realmente trabalham, em vez da maneira pela qual as definições de processo dizem como elas derivam trabalhar. Por exemplo, os controladores de trafego aéreo podem desligar um sistema de alerta de conflito aéreo, que destaca aeronaves voando em rota de colisão.
>
> Os requisitos derivados da cooperação e conscientização das atividades de outras pessoas. Por exemplo, os controladores de trafego aéreo podem utilizar a conscientização do trabalho dos outros controladores para prever o número de aeronaves que entrarão no seu setor de controle. (SOMMERVILLE, 2003).

3.5. Validação de requisitos

Para Sommerville (2003), a validação de requisitos é essencial devido ao surgimento de erros em documentos de requisitos, isso faz com que leve a custos enormes direcionados ao retrabalho, no momento da ocorrência desses erros normalmente no desenvolvimento ou logo depois que o sistema está rodando. O valor do custo de se realizar uma alteração no sistema, referente a um problema de requisitos é bem maior do que corrigir erros de um projeto ou até de problemas de implementações.

A seguir estão listados segundo Sommerville (2003), alguns tipos de verificações que existem e que devem ser feitas no processo de validação de requisitos, dentre elas destacam-se:

- Verificação de validade: o usuário às vezes pensa que um determinado sistema pode fazer todas as funções que deseja, com isso, de acordo com pesquisas e analises no sistema, facilita a identificação de outras funções que possam ser exigidas.
- Verificação de consistência: esse tipo de verificação mostra que os requisitos de um documento não pode haver choque, ou seja, não pode haver restrições para as mesmas funções que estão descritas no sistema.
- Verificação de completeza: consiste em verificar se no documento de requisitos estão incluídos requisitos que abordam todas as possíveis funções exigidas pelo usuário.

Para Pressman (1995), normalmente não se deve deixar de lado as dificuldades de validação de requisitos. Em geral, o processo de validação de requisitos raramente encontra todos os problemas envolvidos nos requisitos, visto que, tais problemas e erros depois da entrega e aceitação do projetista, o documento de requisitos não tem mais correção e não pode ser checado novamente.

3.6. Gerência de requisitos

Segunda Spínola (2008), o gerenciamento de requisitos consiste nas mudanças feitas ao longo do processo, como por exemplo, nas mudanças em mercado, mudanças de legislação, entre outras que os requisitos estão a sofrer. Todos esses fatores faz com que seja necessário mudar os requisitos. Mas para isso, essas alterações, mudanças devem ser coordenadas de maneira adequada para que não haja descontrole sobre os prazos de entrega e os custos do projeto.

A seguir são apresentadas algumas técnicas necessárias nesse processo de gerenciamento de requisitos.

3.6.1. Controle de mudanças

Consiste em manter o controle e planejamento das mudanças que possam ocorrer ao longo do projeto. Segundo Spínola (2008), existe uma maneira bem simples para ajudar a manter esse controle de mudanças que é a "*baseline*" de requisitos, que oferece opções de como era os requisitos, o que foi alterado/introduzido e o que não foi necessário. Para Pressman (1995), é inevitável não ter como mudar o processo de desenvolvimento de um software, diversas vezes os requisitos podem estar incompletos ou inconsistentes segundo a visão do cliente, ou até mesmo a tecnologia que muda.

Uma maneira para definir, gerenciar essas mudanças durante a construção do sistema, seria preparar um controle de mudanças diante da gerência de requisitos. Os requisitos de software se alteram na maioria das vezes em 2% ao mês na fase de implementação e pode aumentar na fase de correção dos requisitos. (PRESSMAN, 1995). O engenheiro de requisitos auxilia que o mais importante é estabelecer uma prática sistemática dessa mudança nos requisitos, deve-se seguir um processo de aceitação do time de desenvolvimento e entre os usuários.

3.6.2. Rastreabilidade

A rastreabilidade é importante porque ela acompanha a "vida" de um requisito de acordo com o projeto, independente da situação, por exemplo, indo do requisito e chegando no projeto ou indo do projeto e chegando no requisito. (SPÍNOLA, 2008).

Para Spínola (2008), realizar a rastreabilidade geralmente deve seguir um conjunto acompanhado da especificação de requisitos em artefato com o nome de matriz de rastreabilidade, na qual, o objetivo é mapear os rastros dos requisitos segundo a especificação, que podem ser:

- Pré-rastreabilidade: que são os rastros seguidos das atas de reunião com o cliente, regras de negócios, que organizaram os requisitos.

- Pós-rastreabilidade: que visa rastrear os requisitos que se formaram a partir da criação dos requisitos, como, a documentação, o código-fonte, entre outros.

3.7. Esclarecimento da importância que tem as ferramentas CASE na engenharia de requisitos

Um importante artefato que auxilia no processo da engenharia de requisitos, são as ferramentas CASE, utilizadas como suporte para desenvolver um projeto de software. Para Sommerville (2003), varias dessas ferramentas fornecem um amplo conjunto de serviços, para ajudar as atividades do processo da engenharia de requisitos, assim apoiando também na qualidade do produto e no tempo de implementação.

Existem algumas vantagens que as ferramentas CASE oferece para o processo da engenharia de requisitos, podemos mencionar, o aumento da produtividade para a equipe de desenvolvimento, diminuição dos custos do software, melhor gerenciamento dos requisitos e facilidade na manutenção das etapas durante o desenvolvimento.

De maneira geral, as ferramentas CASE, ajudam a automatizar um enorme conjunto de tarefas que existem na engenharia de software e na engenharia de requisitos, como, a parte de geração de código, nos testes do software, na documentação dos requisitos e também nas atividades da engenharia de requisitos, principalmente na atividade de gerência de requisitos.

4. PROTOTIPAÇÃO E ESPECIFICAÇÃO DE REQUISITOS: ESSENCIAIS PARA O SUCESSO DO PROJETO

Para início do capítulo será mostrado um fator que está ligado à engenharia de requisitos, os custos envolvidos na manutenção de software, algo que pode parecer simples, mas que ultimamente traz muita preocupação para empresas e os próprios profissionais da área.

Recentemente, segundo Ávila (2008), foi diagnosticada a real necessidade de analisar mais a fundo e de forma mais sistemática os estudos relacionados às atividades de engenharia de requisitos. Um dos destaques é o grande crescimento envolvendo as correções de requisitos, que com isso se tornam inadequadamente elicitados, ou seja, mal corrigidos, mal analisados, com isso gerando implicações com os usuários que acabam insatisfeitos com o sistema, e também não se aplicando as suas reais necessidades.

Para Sommerville (2003), os custos de correções de erros de requisitos em etapas posteriores do processo de engenharia de requisitos podem ser muito elevados.

4.1. Custos

Segundo Spínola (2007), a Figura 7 apresenta claramente os possíveis custos relacionados à engenharia de requisitos. Percebe-se que os custos envolvidos com a análise de requisitos são baixos, mas em geral, é nesta atividade que muitos dos defeitos são inseridos. Para Spínola (2008), analisando a primeira linha dessa atividade fica claro também que os custos desses problemas são relativamente baixos. De acordo com a Figura conclui-se que de fato existe a necessidade de focar nas atividades envolvidas com a especificação dos requisitos.

	% do Custo de Desenvolvimento	% dos erros introduzidos	% dos erros encontrados	Custo relativo de correção
Análise de Requisitos	5	55	18	1
Projeto	25	30	10	1 - 1.5
Códificação e teste de unidade	50			
Teste	10	10	50	1 - 5
Validação e Documentação	10			
Manutenção		5	22	10 - 100

Figura 7 - Representação dos custos relacionados às etapas da engenharia de requisitos. Fonte: Spínola (2007).

4.2. Análise crítica

Segundo Spínola (2007), focando na importância que as atividades relacionadas aos requisitos seguem no mercado de tecnologia, uma pesquisa feita pela Standish Group analisando 320 companhias e 7.000 projetos de softwares, em 1996, obteve o seguinte resultado conforme Figura 8:

Figura 8 - Distribuição de término dos projetos de software. Fonte: Spínola (2008).

Para Spínola (2007), 16% dos projetos de softwares são finalizados com sucesso, pois, segue alguns fatores relevantes, como o tempo e o custo determinado.

Segundo Ávila (2008), 50% dos projetos estão em fase problemática, ou seja, não realizando claramente os processos da engenharia de requisitos, assim, tendo um elevado custo e possivelmente fora do prazo especificado.

E o outro restante, comprova que 30% dos projetos de softwares são abortados no meio do processo da engenharia de requisitos, ou até na etapa de implementação do mesmo (SPÍNOLA, 2007).

4.3. Prototipação de software

A prototipação é feita em diversas fases do processo de engenharia de requisitos, como por exemplo, na identificação, análise e validação dos requisitos, mas será abordado apenas duas atividades neste trabalho.

Segundo Sommerville (2003), um protótipo é um modelo inicial de um sistema, na qual, serve para esboçar conceitos e testar características do projeto e também para saber se existem problemas e as possíveis correções do mesmo.

Para Sommerville (2003), a implementação de um protótipo logo no inicio de um projeto é importante, pois se podem controlar os possíveis custos do projeto e também para que os usuários possam testar algumas funcionalidades do sistema. Já segundo Pressman (1995), mostra que um protótipo de software ajuda em duas atividades do processo de engenharia de requisitos, que são:

- Levantamento de requisitos: onde com o protótipo do sistema pronto, faz com que os usuários possam fazer experiências de como o sistema contribui para seu serviço. Isso é importante, pois é nessa atividade que são analisados os requisitos, com isso os usuários podem descobrir pontos positivos e negativos do sistema e posteriormente até negociar novos requisitos.
- Validação de requisitos: o protótipo do sistema pode descobrir erros nos requisitos que foram testados e validados. Nessa atividade do processo de engenharia de requisitos é comum os usuários pedirem algumas funções para serem feitas no projeto e através do protótipo que a especificação do sistema pode ser alterada pelos usuários, possibilitando até mudanças dessas funções que antes os próprios achavam importante no sistema.

Para Pressman (1995), a prototipação faz parte do processo de engenharia de requisitos, a prototipação também possui alguns benefícios além de auxiliar os usuários na especificação de requisitos, como:

- A partir das funções feitas no sistema, é possível descobrir alguns erros ou equívocos da equipe de desenvolvimento e também dos usuários.
- Quando o protótipo está pronto é possível também encontrar requisitos incompletos, indefinidos através da equipe de desenvolvimento.

De fato, fazer a implementação de um protótipo de software ajuda a melhorar a especificação de requisitos trazendo outra vantagem, segundo Sommerville (2003):

- Treinamento do usuário: consiste no treinamento do usuário antes do sistema efetivo estar disponível.

4.3.1. Prototipação no processo de software

De acordo com Pressman (1995), é complexo para os usuários prever como eles iram testar os sistemas para ajudar os mesmos no trabalho do dia a dia, ainda mais se esses sistemas foram grandes, possivelmente será difícil realizar esse teste antes do sistema ficar pronto.

Um jeito de encarar esse problema é usar o protótipo evolucionário para a implementação do sistema. Isso quer dizer compartilhar com o usuário um sistema indefinido e posteriormente alterá-lo e ampliá-lo para que os requisitos do usuário fiquem mais objetivos (SOMMERVILLE, 2003).

Para Pressmam (1995), outra forma é construir um protótipo descartável para apoiar nas atividades de análise e validação de requisitos. Logo após a avaliação, o protótipo é descartado e um possível sistema com requisitos definidos, na qual, a qualidade do mesmo poderá aumentar.

Para Sommerville (2003), a prototipação evolucionária inicia-se com um sistema simples, mas que possui os requisitos mais essenciais do usuário. Tal sistema é modificado de acordo com novos requisitos encontrados. No final o sistema acaba virando o que era necessário quando o usuário estava negociando. A

prototipação evolucionária é recentemente e vem sendo uma técnica muito usada para desenvolvimento de web sites e comercio eletrônico.

Segundo Sommerville (2003), a prototipação descartável tem o objetivo de ajudar na análise da especificação do sistema. O protótipo é descrito, analisado e alterado. Depois de verificado ele mostra a especificação detalhada do sistema, com isso é inserido no documento de requisitos e por fim é descartado, tornando assim sem utilidade.

É proposto que Sommerville (2003), existe diferenças entre o dever da implementação evolucionária e da implementação descartável, que são:

- O foco da prototipação evolucionária é depositar um sistema real aos usuários, que dizer, o processo começa com os requisitos do usuário, visto que, são mais analisados e possui mais importância e são feitos apenas com a permissão do usuário.
- Já para a prototipação descartável o foco é validar os requisitos, é necessário iniciar com os requisitos que estão indefinidos, sem a absoluta certeza do mesmo, pois de fato, é interessante retirar mais informações desses requisitos.

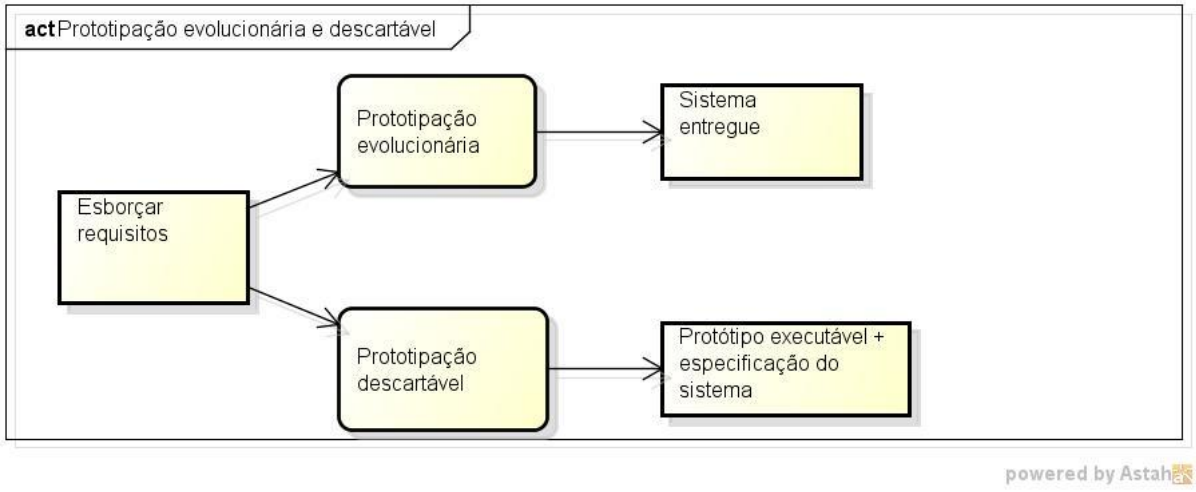

Figura 9 - Representação da prototipação evolucionária e a prototipação descartável (Autoria propria).

A Figura 9 acima mostra o modelo de prototipação evolucionária e descartável.

4.3.2. Prototipação Evolucionária

Para Sommerville (2003), o modelo de prototipação evolucionária foca no objetivo de programar um modelo inicial do sistema, mostrando anotações dos usuários e desenvolvendo-a por varias etapas, até que um possível sistema correto seja criado. A prototipação evolucionária antigamente era bastante usada para sistemas de inteligência artificial, que normalmente era quase impossível ou complexo de se realizar a especificação de software.

Hoje em dia a prototipação evolucionaria já é uma técnica usada em grande parte de sistemas, para o desenvolvimento de software. Segundo Sommerville (2003), essa técnica esta incluída junto a técnicas como a RAD, focada na implementação rápida e no JAD, focada na implementação conjunta de aplicações. Para Spínola (2007), JAD (*joint application development*) consiste em uma técnica que faz a interação entre os envolvidos que precisam tomar uma atitude que proporciona vários danos em diversas áreas de uma empresa. Essa técnica engloba tarefas de preparação para reuniões com os usuários, reuniões de workshop, entre outras.

Segundo Spínola (2007), o JAD tem que ser utilizado em casos que necessita de um proposito entre os usuários, pois, ajuda os envolvidos do projeto terem uma visão ampla do sistema.

Segundo Pressman (1995), o principal objetivo do JAD é elevar o comprometimento e o envolvimento do usuário para com isso, obter dados para a elaboração do documento de especificação de requisitos para o projeto com o entendimento de todos. Essa técnica é dividida em quatro fases, a Figura 10 mostra de forma clara essas fases:

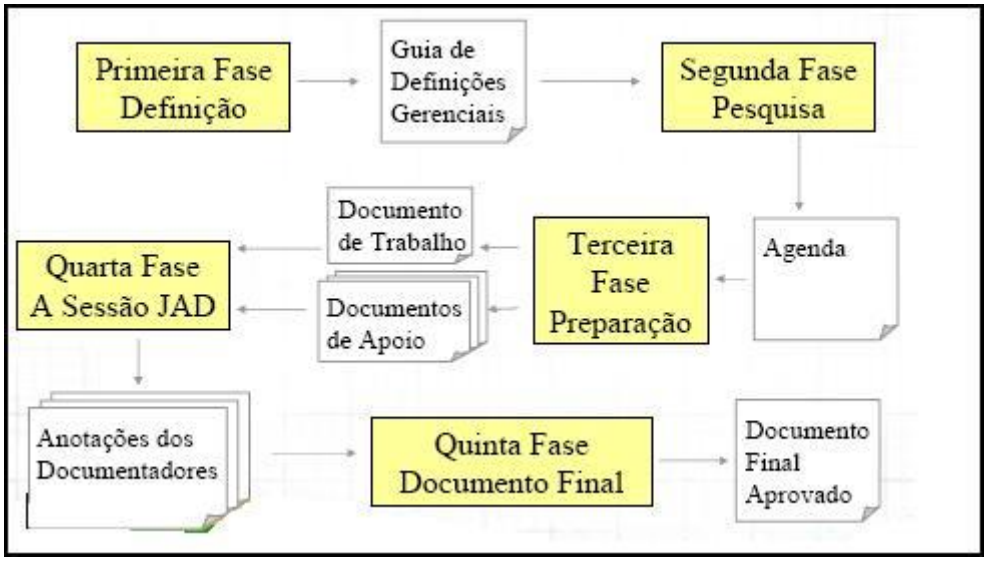

Figura 10 - Fases da técnica para levantamento de requisitos JAD. Fonte: Pressman (1995).

Para Sommerville (2003), existem duas vantagens importantes para utilizar essa técnica no desenvolvimento de software que são:

1- A rápida resposta do sistema: a alteração na regra de negócios mostra que é importante o apoio ao sistema e que seja fornecido rapidamente. Em determinados casos, essa rápida resposta junto com a facilidade de uso são de fato mais essencial que a funcionalidade de manutenção de software.

2- Envolvimento do usuário com o sistema: a participação do usuário com o processo de desenvolvimento de software, não significa apenas que o sistema atenderá corretamente aos requisitos, mas também quer dizer que os usuários finais adotaram certo compromisso e diretamente irão ajudar para o funcionamento dele. Abaixo é mostrado o modelo da prototipação evolucionária na Figura 11:

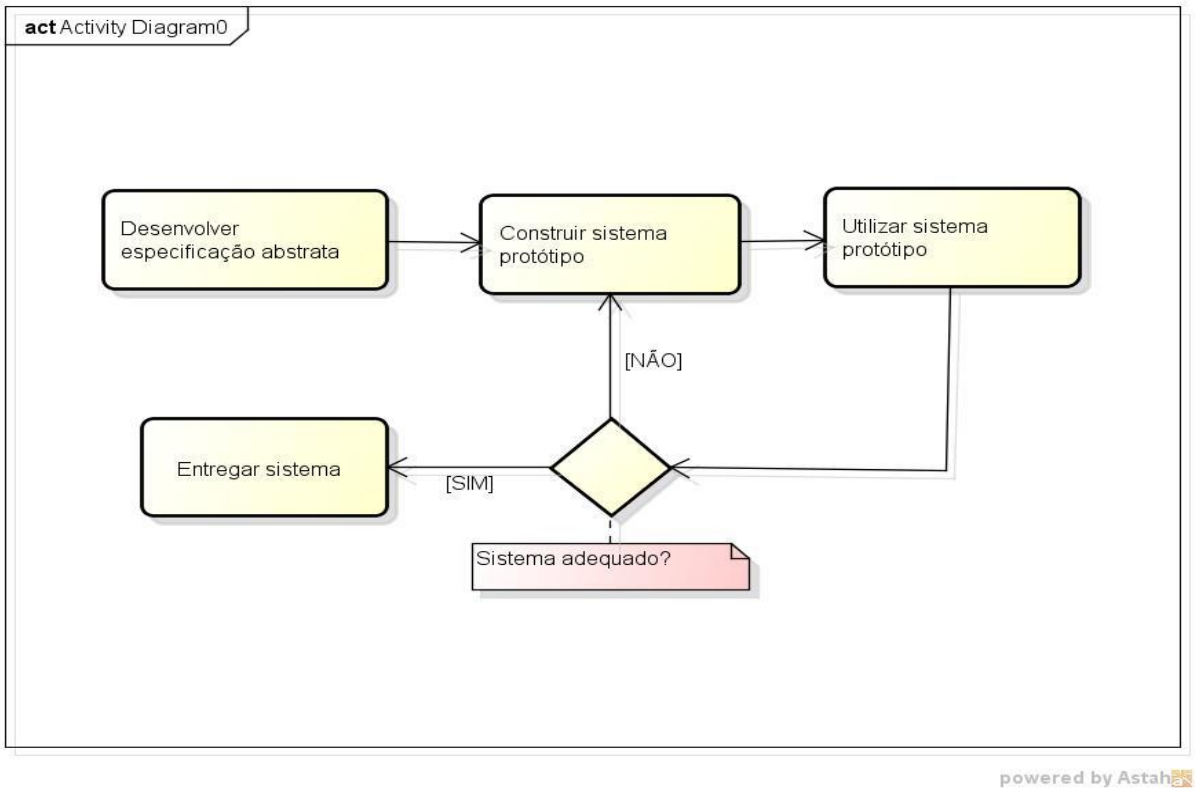

Figura 11 - Representação da prototipação evolucionária. (Autoria própria).

Segundo Sommerville (2003), as aplicações com foco na prototipação evolucionária e na especificação para a implementação de software se diferenciam em sua aplicação de verificação e validação. Na verificação é onde se pode checar se o programa contribui para as especificações. Já a validação deve apresentar que o programa é correto ao consenso previsto entre os profissionais.

Contudo, a verificação e a validação de um projeto que foi implementado utilizando a prototipação evolucionária, nesse caso, podem apenas checar se o sistema está correto, ou seja, se é bom o suficiente para o proposito indicado.

4.3.3. Prototipação Descartável

Para Sommerville (2003), uma abordagem usando a prototipação descartável ajuda a elevar o processo de análise de requisitos, com o intuito de diminuir os

custos envolvidos ao longo do ciclo de vida. O principal objetivo da prototipação descartável é mostrar de maneira clara os requisitos e prover informações para os engenheiros de software analisar o risco do processo. Logo depois dessa análise o protótipo é descartado e ele não é utilizado como base para a implementação seguinte do sistema. Esse modelo do processo de software focado no estágio essencial da prototipação está a seguir na Figura 12:

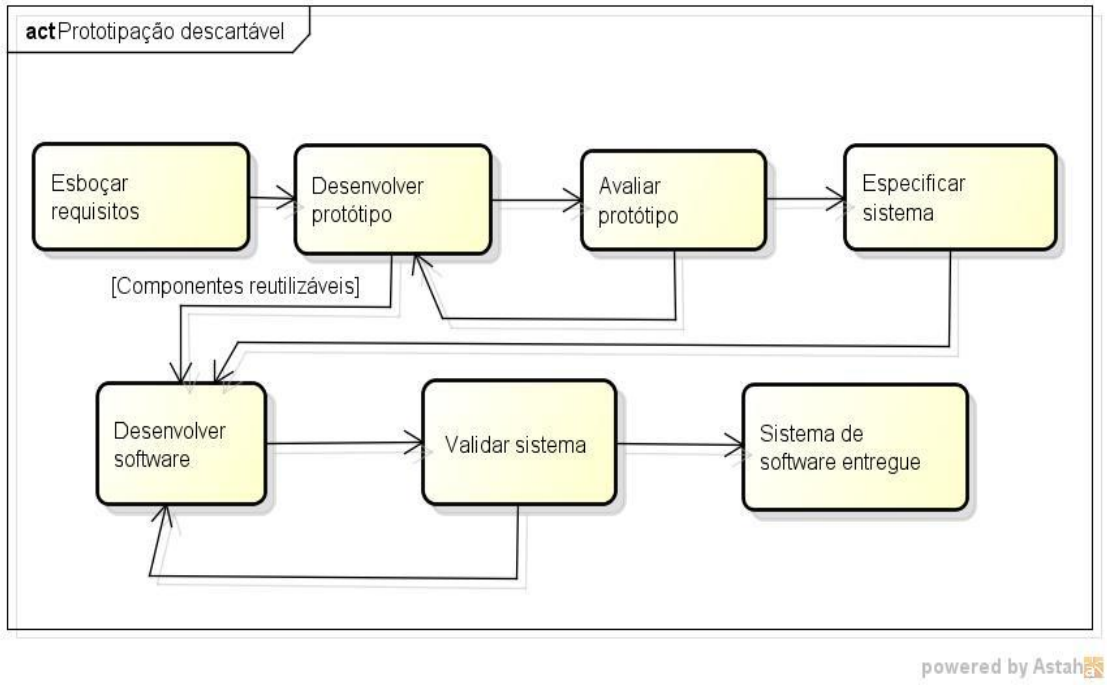

Figura 12 - Representação de um processo de software com prototipação descartável. (Autoria própria).

Segundo Pressman (1995), a prototipação descartável não costuma ser utilizada para a validação do sistema e sim para ajudar a implementar os requisitos do sistema. O sistema deve ser programado de maneira rápida para que os usuários possam passar um retorno da sua experiência com o protótipo descartado.

Para Sommerville (2003), o modelo apresentado na Figura 12 leva em conta que o protótipo é implementado após um resumo da especificação do projeto.

Segundo Pressman (1995), a prototipação descartável também tem uma desvantagem, a maneira de utilização do protótipo talvez não se identifique de acordo com o sistema final a ser entregue, ou seja, na sua utilização depois do

desenvolvimento. E também existe o fato do treinamento, que durante a avaliação do protótipo talvez possa ser insuficiente para o andamento da equipe do sistema.

4.4. Especificação

Segundo Pressman (1995), a especificação de software tem como objetivo aprimorar a qualidade do projeto. Os analistas ultimamente trabalham com especificações inconsistentes, com pouco grau de certeza e como consequência acontece resultados na frustação e incerteza do documento.

Para Pressman (1995), as técnicas utilizadas no processo de análise ajuda a levar uma especificação mais clara em papel, sendo implementada utilizando ferramentas CASE, que possui anotações especificas dos requisitos e também tem o fato da prototipação que fornece um executável, ou seja, esse protótipo executável ajuda a dar apoio a apresentação dos requisitos. Abaixo é mostrada na Figura 13, uma pequena representação da especificação juntamente com o projeto:

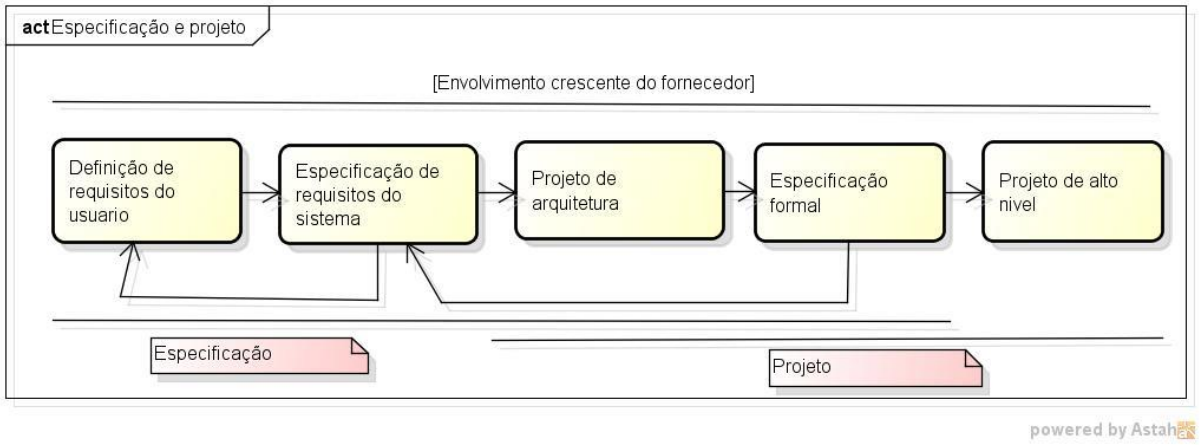

Figura 13 - Representação da especificação e do projeto (Autoria própria).

4.4.1. Especificação formal no processo de software

Segundo Sommerville (2003), existem três níveis de especificação de software que pode ser implementado ao longo do projeto, tais como, os requisitos do usuário, os requisitos do sistema e a especificação do projeto. Agora esclarecendo

mais sucintamente os níveis de especificação de software, o requisito de usuários consiste em uma especificação feita com qualidade e a especificação de projeto é a com mais detalhes.

Para Sommerville (2003), no processo de especificação é normal que o contato do usuário para com o retorno com o sistema seja menor, mas logo no começo do desenvolvimento, é de grande importância o envolvimento do usuário e a especificação deve ser implementada de acordo com a visão do cliente, ou seja, com baixo grau de complexidade da parte dos engenheiros de software. Posteriormente, em contra partida é necessário que no final do processo, a especificação seja voltada para os fornecedores, nesse caso tornando a especificação mais completa e concisa.

A seguir são mostradas na Figura 14, as etapas dessa especificação de software podendo ser realizada paralelamente:

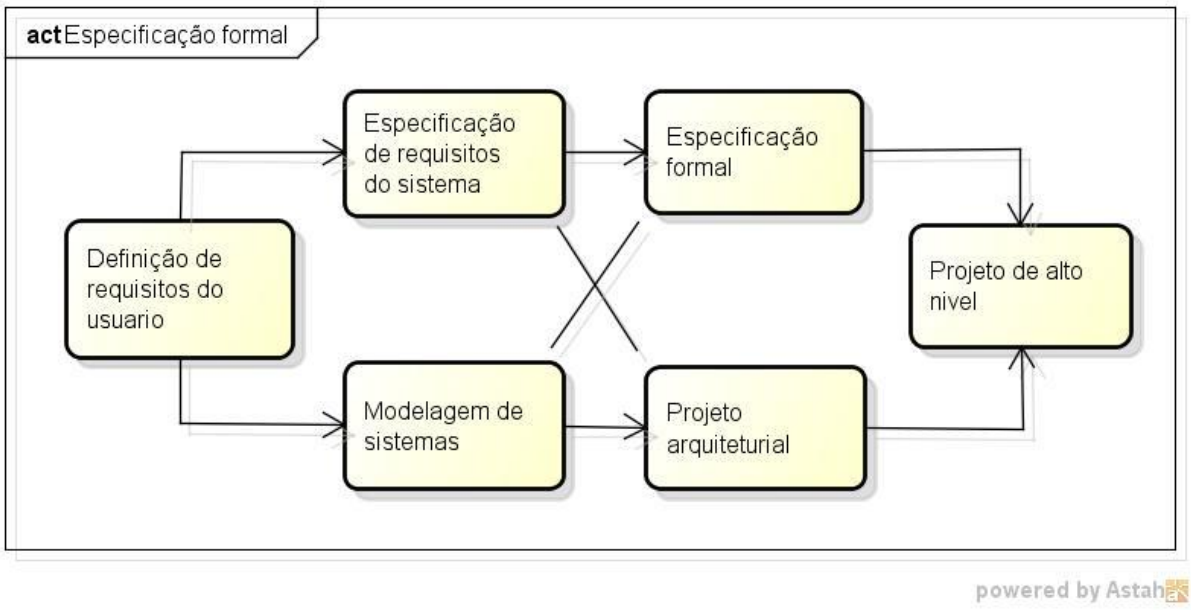

Figura 14 - Especificação formal no processo de software. (Autoria própria).

Ressalta Pressman (1995), que esse processo da especificação não precisa necessariamente ser desenvolvido segundo as etapas da Figura 14, como mostrado acima, tal processo pode se realizar em paralelo.

Para Sommerville (2003), implementar e fazer a análise de uma especificação formal excede custos de desenvolvimento no projeto. A Figura 15 mostra claramente os tais custos do desenvolvimento em algumas etapas durante a produção de software, que são afetados pela utilização da especificação. Mesmo com a especificação formal durante o projeto, os custos da especificação e do desenvolvimento são iguais e já os custos da validação são minimamente diminuídos.

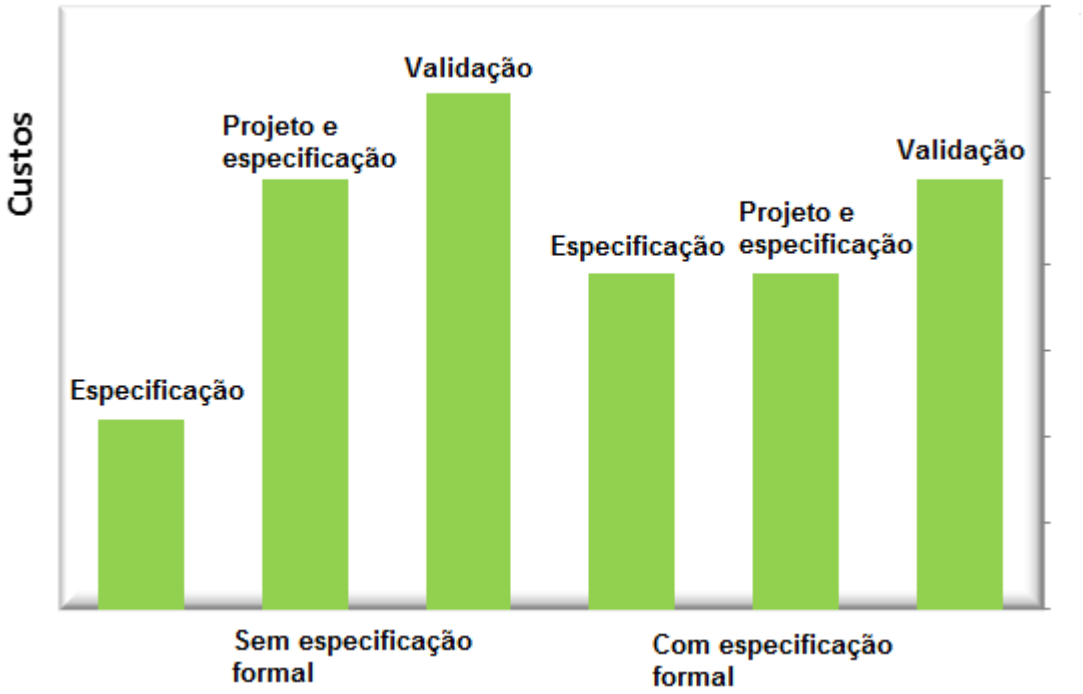

Figura 15 - Custos da implementação de software usando a especificação formal. Fonte: Sommerville (2003).

4.4.2. Revisão da Especificação

Segundo Pressman (1995), a etapa de revisão de requisitos deve ser considerada tanto pelos desenvolvedores quanto ao cliente, que estão envolvidos no sistema. Essa revisão é analisada como um todo, primeiro deve ser objetiva, precisa e completa.

Para Pressman (1995), existem algumas questões que devem ser levadas em conta, para garantir um bom nível de qualidade na especificação:

- Os diagramas são objetivos? Cada um pode sustentar-se sozinho, sem anotações a parte?
- O contato com o cliente foi seguido dentro do esperado?
- As interfaces relevantes aos elementos do sistema foram descritas?

Para atender a todos essas perguntas acima, uma revisão de especificação deve focar em um nível mais detalhado do sistema.

Aponta Sommerville (2003), algumas diretrizes para serem seguidas na revisão da especificação detalhada:

- Busque achar declarações que indiquem um grau de certeza (por exemplo, “sempre”, “todos”, “nunca”).
- Quando um cálculo for detalhado, desenvolva dois exemplos no mínimo.
- Tomar cuidado com verbos vagos, como por exemplo, “manuseado”, “pulado”, “eliminado”, pois há diversas maneiras de analisa-los.

Segundo Pressman (1995), depois da revisão ser concluída, a especificação de requisitos é “carimbada” pelo cliente e pelo analista, com isso, a especificação vira-se um “termo de contrato” da implementação do software.

Portanto, ainda com esses procedimentos corretos da revisão, existem sempre problemas durante a especificação que de fato são comuns. Essa especificação é complexa de ser “revisada” e com isso, podem-se surgir inconsistências na especificação de requisitos e assim podendo não ser identificadas. (PRESSMAN, 1995).

Na Figura 16 a seguir, percebe-se a grande diferença encontrada na maioria dos projetos de softwares, em relação aos erros encontrados nos mesmos. Segundo uma pesquisa realizada em 2009 na Universidade Federal do Vale do São Francisco (Univast), 40% do percentual de erros detectados nos sistemas, deve-se às especificações mal feitas.

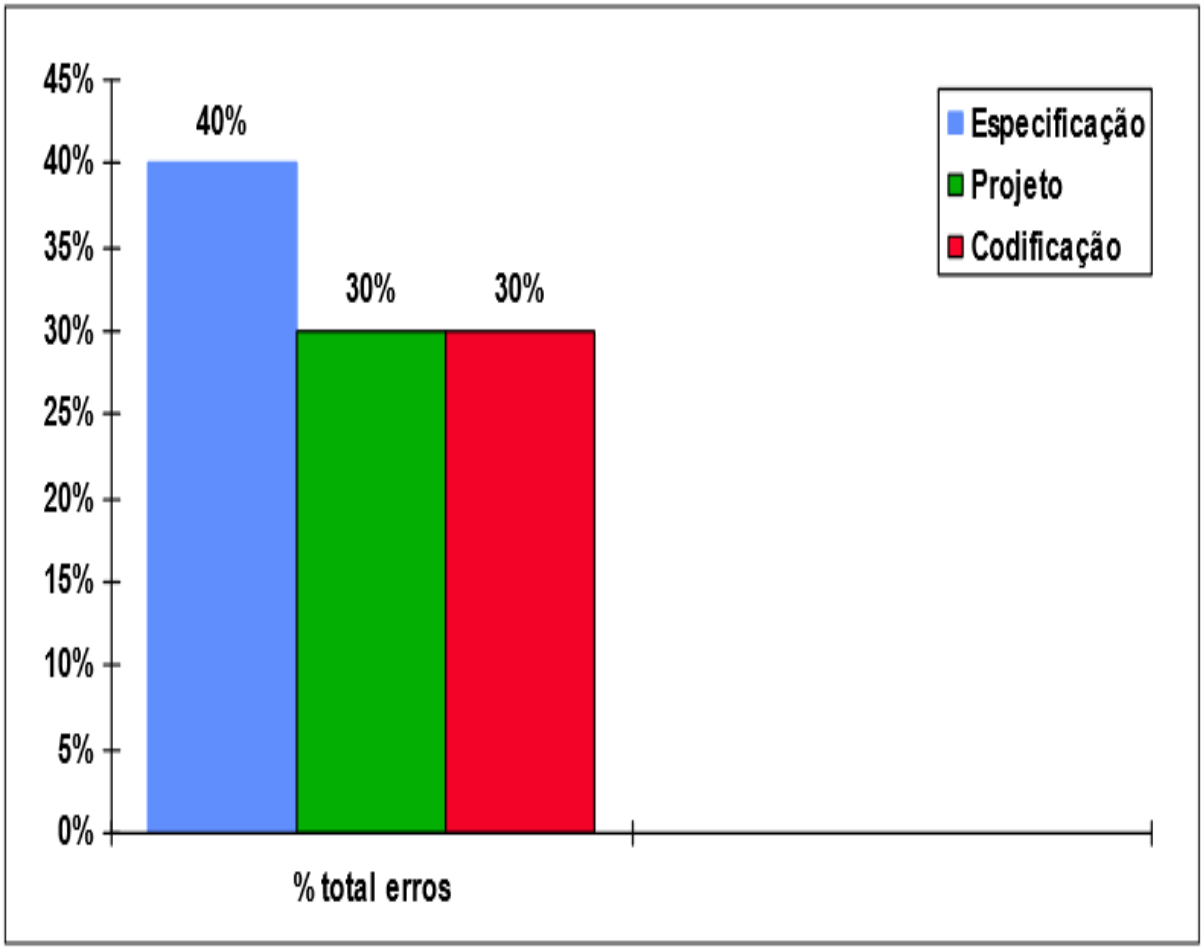

Figura 16 - Estatística dos erros encontrados nos sistemas de informação. Fonte: (Univast-2009).

5. ESTUDO DE CASO – A IMPORTÂNCIA DO DIAGRAMA DE CASOS DE USO NO PROCESSO DA ENGENHARIA DE REQUISITOS

Hoje em dia, as empresas estão cada vez mais preocupadas com as ferramentas, técnicas que são adotadas ao longo do processo de engenharia de requisitos, para que possam auxiliar na produção de qualidade e mais concisa dos requisitos de software. Antes da introdução ao estudo de caso que iremos abordar nesta seção, precisamos começar com uma definição do que é a UML, segundo Guedes (2009):

> A UML – Unified Modeling Language ou Linguagem de Modelagem Unificada – é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. Essa linguagem tornou-se, nos últimos anos, a linguagem padrão de modelagem adotada internacionalemente pela indústria de software. (GUEDES, 2009).

5.1. Introdução a UML

Segundo Guedes (2009), a UML não é uma tecnologia de programação, mas sim uma linguagem de modelagem, é usada para ajudar os profissionais da área, como, analistas, engenheiros a especificarem as características do sistema, como requisitos, comportamentos, estruturas lógicas, entre outros, para suas reais necessidades com relação a um determinado objetivo do sistema.

5.2. Diagrama de Casos de Uso

A seguir será mostrado basicamente um exemplo de um diagrama de casos de uso oferecido pela UML, e no qual, posteriormente será mostrado um estudo de caso.

Segundo Guedes (2009), o diagrama de casos de uso é um dos principais diagramas da UML, sendo também o mais informal. Esse diagrama é normalmente usado nas etapas de análise e levantamento dos requisitos de software, como também pode ser usado em todo o processo da engenharia de requisitos. Ele é bastante simples, utiliza uma representação de uma linguagem fácil de ser interpretada entre os usuários do sistema, por isso a grande importância do mesmo na fase da engenharia de requisitos.

Os casos de uso possuem atores, que pode ser usuários, outros sistemas, um hardware, tais características utilizam de qualquer maneira o sistema e também existem os serviços, as funcionalidades que o sistema fornecerá aos atores.

Segundo Guedes (2009), o diagrama de casos de uso é essencial para demostrar às funcionalidades do sistema, sem se importar como as funcionalidades que serão programadas. Ele ajuda na identificação e análise de requisitos, bem como, nas etapas de especificação, documentação e visualização de características e funcionalidades do sistema.

A seguir na Figura 17, será mostrado um exemplo para esclarecer como são os casos de uso de um diagrama de casos de uso. Para Guedes (2009), os casos de uso são usados para obtenção dos requisitos de sistema, eles mostram as funcionalidades, serviços que o projeto ou sistema necessita para interagir com os atores.

Esse último será mostrado posteriormente, pois, é também um dos componentes do diagrama de casos de uso.

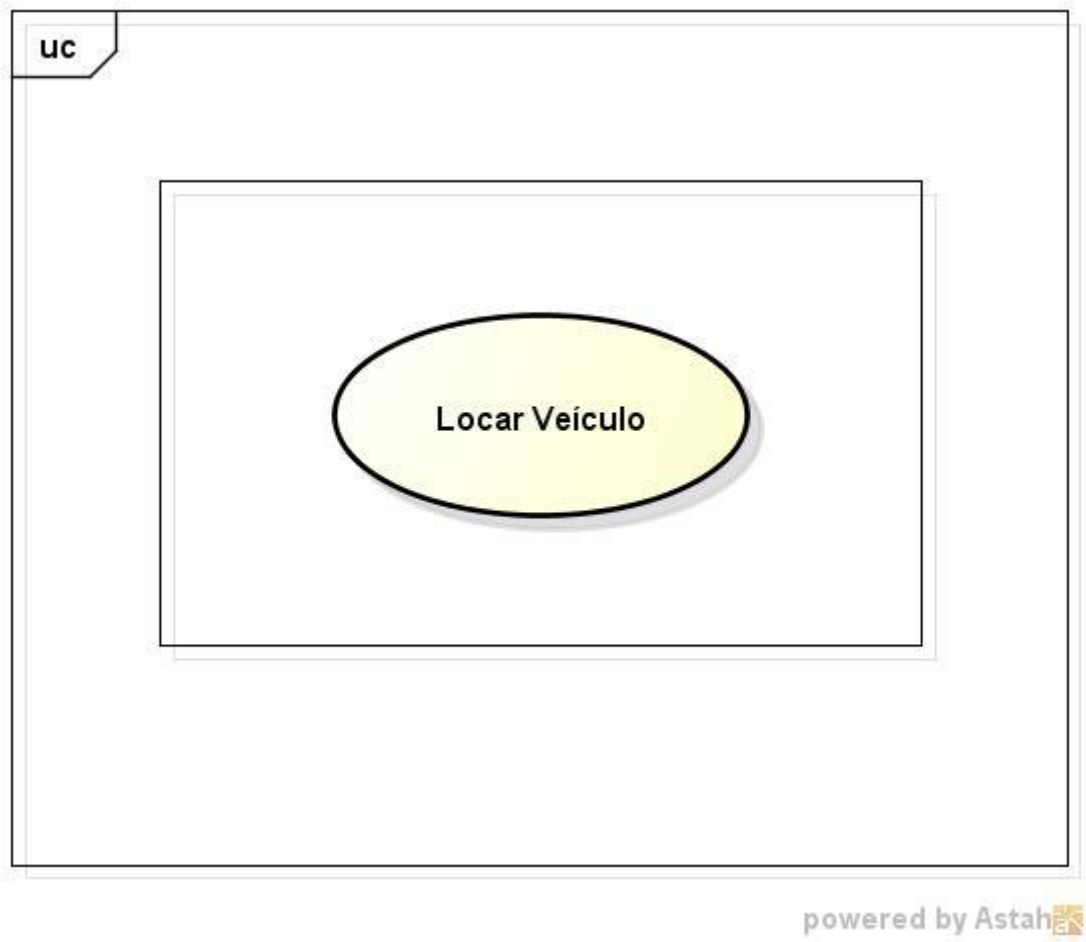

Figura 17 - Exemplo de Caso de Uso. (Autoria própria).

Segundo Guedes (2009), atores demonstram a função agregada pelos possíveis usuários que interagem com o sistema. Os atores também podem representar um sistema especifico um hardware, ou software, algo que possui interação através do sistema.

Na Figura 18, pode-se observar o exemplo de atores do diagrama de casos de uso:

Figura 18 - Exemplo de atores. (Autoria própria).

5.2.1. Objetivos e vantagens do diagrama de casos de uso

É de suma importância ressaltar alguns objetivos que o diagrama de casos de uso oferece, para com isso, ficar mais clara a ideia sobre os casos de uso, que são essenciais no processo de engenharia de requisitos.

Segundo Leite (2007), alguns dos principais objetivos do diagrama de casos de uso, são: delimitação do contexto do sistema, documento e compreendimento dos requisitos, demonstração dos requisitos finais do sistema, primordial na atividade de analise dos requisitos, ajuda no dialogo com os *stakeholders* e por fim ajuda no desenvolvimento dos casos de teste finais.

Dentre as vantagens do diagrama de casos de uso, encontram-se as seguintes:

- Modelando um caso de uso, é normalmente visto como uma boa técnica para pegar os requisitos funcionais;
- Podem ser reutilizáveis dentro do contexto do sistema;
- Fornece mais facilidade na interpretação de requisitos, pois mostra em forma de cenários ou histórias os requisitos envolvidos no projeto.

A seguir será mostrado na Figura 19 um breve exemplo de diagrama de casos de uso sobre um sistema de controle bancário:

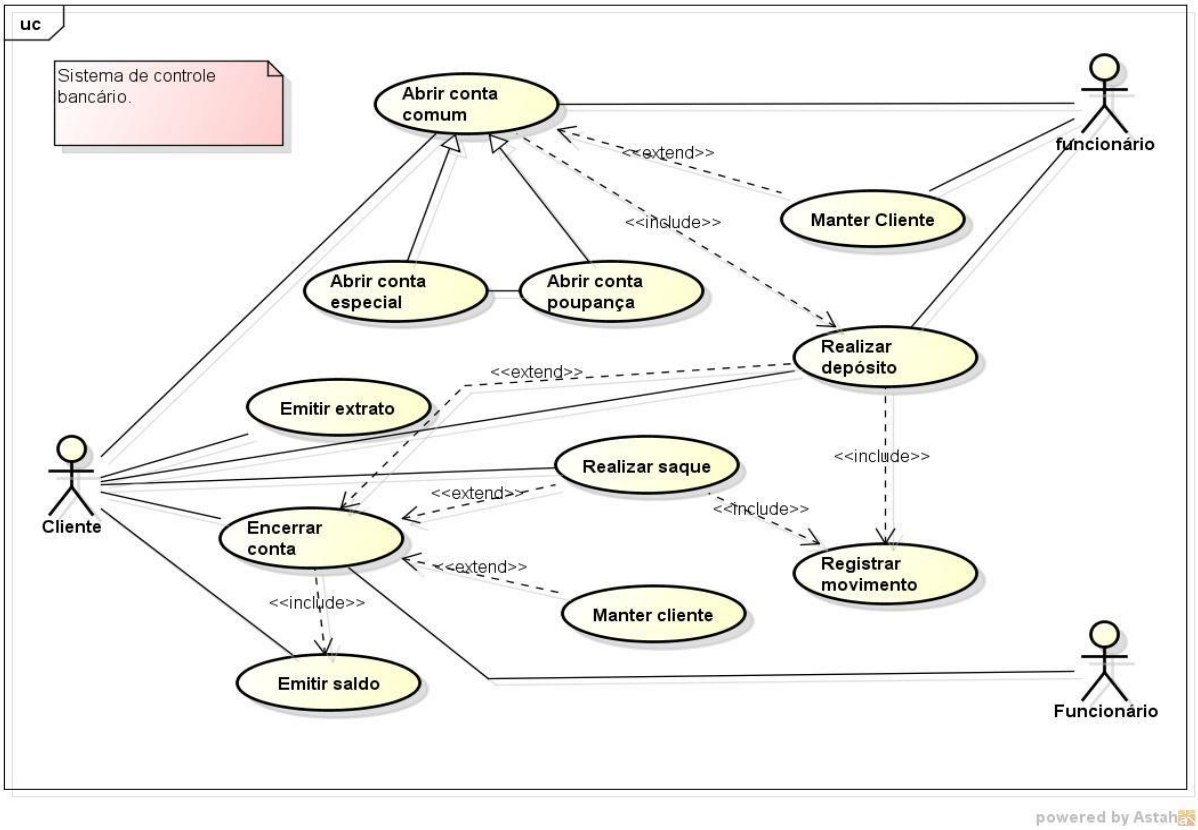

Figura 19 - Exemplo de um diagrama de casos de uso. (Autoria própria).

5.3. Enunciado do ambiente a ser modelado

Nesta parte será descrita de forma básica o estudo de caso a ser modelado, e também será apresentada a necessidade do cliente em questão de acordo com o problema.

O empresário Airton precisa de um sistema para controlar sua empresa de automóveis, ele possui uma empresa de locação de veículos e percebeu que seu sistema estava defasado diante de seus concorrentes, sem muitas funcionalidades e opções para o aluguel dos veículos. Para isso, o próprio contatou Amaury, um analista de sistemas já com bastante experiência no mercado e com conhecimentos sólidos de modelagem, usando diversos diagramas da UML. Para automatizar o sistema da empresa de Airton, o analista deve levar em conta os seguintes requisitos de sistema.

A empresa possui uma frota enorme de carros de passeio, sendo que esses carros apresentam diferentes marcas e modelos. As vezes um carro pode ser

retirado da frota devido a algum acidente grave ou pelo fato de ter sido considerado velho demais para o padrão da empresa e tenha sido vendido. Da mesma forma, a empresa de Airton às vezes renova a frota, mas sempre mantendo o cadastro de veículos da empresa.

Os clientes vão até a empresa e solicitam o aluguel de carros. Mas, primeiramente deve-se cadastra-los, caso ainda não tenham cadastro ou seus dados tenham sido modificados.

Depois de ter cadastrado, o cliente pode escolher o carro que deseja alugar, mas com isso, deve-se levar em conta os seguintes pontos: o valor da locação varia de acordo com o ano, marca e o modelo do veiculo. Durante o processo de locação, o cliente deve informar por quanto tempo vai utilizar o carro, para qual finalidade e por onde deseja se locomover, pois essas informações influenciam no preço da locação do mesmo. Antes de liberar o veículo, a empresa exige que o cliente forneça um valor superior ao estudado durante a locação, a titulo de caução. Caso o cliente não utilize todo o valor da caução planejado até o momento da devolução do veiculo, o valor restante será devolvido ao mesmo.

No momento que o cliente devolve o carro é necessário definir o veículo como devolvido, assim registrando a data, hora da devolução e a quilometragem em que se encontra, e, contudo verificar se o automóvel se encontra nas mesmas condições em que foi alugado. Caso o cliente tenha usado o carro por mais tempo que o planejado, deverá pagar o aluguel referente ao tempo extra em que permaneceu com o veículo. Da mesma forma, o cliente terá que pagar por qualquer dano sofrido pelo veiculo quando este encontrava locado. No entanto, o cliente pode ser ressarcido de parte do valor que pagou, caso o custo do tempo em que esteve de com o veículo seja menor que o valor fornecido no momento da locação.

5.4. Identificando os atores e seus casos de uso no sistema

Os atores identificados no estudo de caso foram:

Cliente: este ator representa os clientes que desejam locar veículos da empresa. Esse ator interage com todos os casos de uso do sistema, visto que o funcionário necessita de informações do cliente para utilizá-los, assim sendo essa a única exceção do caso de uso manter veículos, manipulado exclusivamente pelo funcionário.

Funcionário: este ator representa os funcionários que atendem os clientes da empresa.

Já para os casos de uso, foram identificados os seguintes:

Manter veículos: este é um caso de uso a parte que representa o processo de manutenção do cadastro de veículos da empresa de Airton.

Manter clientes: este também é um caso de uso a parte que representa a manutenção do cadastro de clientes. Toda vez que um novo cliente solicitar a locação de um veículo e se este ainda não estiver registrado ou seus dados tiverem sido modificados, o funcionário deverá planejar esse caso de uso específico.

Locar veículo: este caso de uso identifica as etapas necessárias para que um cliente consiga locar um veículo. É necessário que o mesmo selecione o veículo que deseja locar e informar por quanto tempo deseja utilizar, também para qual finalidade e por onde deseja locomover, e ainda terá de fornecer um valor de caução para poder alugar o veículo.

Devolver locação: este caso de uso identifica as etapas que serão executadas quando o usuário devolver o veículo, onde será registrada a data, hora da devolução do veículo, sua quilometragem e se este se encontra nas mesmas condições de quando foi alugado. Nesse processo o cliente pode ter que pagar o aluguel referente ao período extra que ficou com o veiculo ou qualquer dano ou multa enquanto utilizava o mesmo. No entanto, ele pode vir a ser ressarcido de parte do valor que pagou se tiver ocupado por menos tempo.

5.5. Modelando o sistema de acordo com o estudo de caso proposto

Nesta parte será mostrado a seguir na Figura 20 o sistema modelado, utilizando o diagrama de casos de uso para identificação dos requisitos do sistema, e logo depois a documentação do mesmo para todos os casos de espuso especifico:

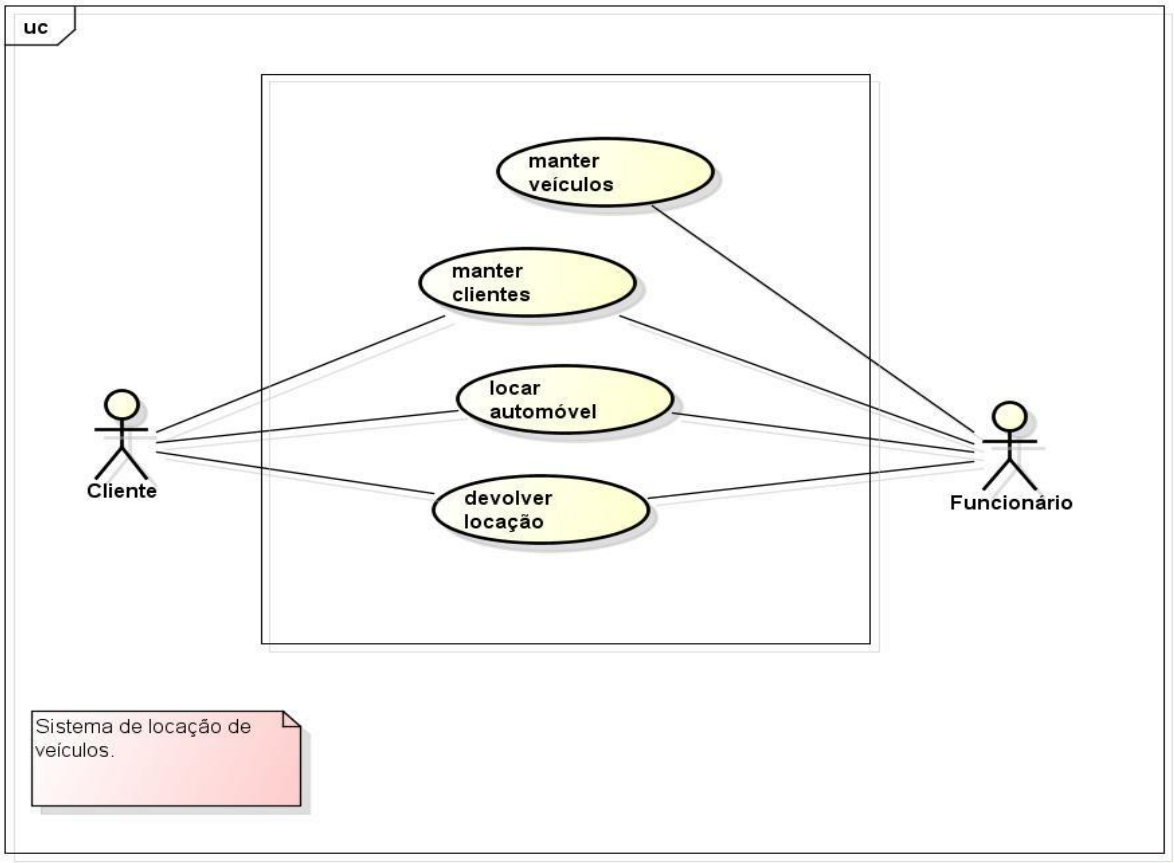

Figura 20 - Diagrama de casos de uso referente ao estudo de caso. (Autoria própria)

.

Outra parte importante para facilitar a visualização dos requisitos é a documentação de cada processo usado no diagrama de casos de uso. A documentação é importante na etapa de análise e levantamento de requisitos, pois mostram de maneira clara os requisitos envolvidos no contexto do sistema bem como o seu entendimento.

A seguir, conforme a Tabela 1 mostra a documentação do caso de uso **Locar automóvel:**

Sistema de Locação de Veículos **Especificação de Caso de Uso: Locar Automóvel**	
Nome do caso de uso	**Locar Automóvel**
Caso de Uso Geral	
Ator Principal	**Funcionário**
Atores secundários	**Cliente**
	Este caso de uso descreve as etapas percorridas por um funcionário para realizar a locação de um automóvel
Pré-condições	**O cliente precisa estar cadastrado**
Pós-condições	**Receber valor de caução**
Fluxo Principal	
Ações do ator	**Ações do sistema**
1. **Selecionar opção de locação de veículos**	
	2. Carregar clientes
	3. Carregar veículos disponíveis
4. **Selecionar cliente**	
5. **Selecionar veículo**	
	6. Apresentar detalhes do automóvel
7. **Informar dados locação**	
	8. Calcular valor da locação e valor da caução
9. **Fornecer valor da caução**	
	10. Registrar locação
Restrições/Validações	

Tabela 1 – Documentação do Caso de Uso Locar automóvel. (Autoria própria).

Na Tabela 2 a seguir será mostrada a documentação do Caso de Uso **Manter Clientes**, que faz parte também do estudo de caso:

Sistema de Locação de Veículos **Especificação de Caso de Uso: Manter Cliente**	
Nome do Caso de Uso	**Manter Cliente**
Caso de Uso Geral	
Ator Principal	**Funcionário**
Atores Secundários	**Cliente**
pResumo	**Este caso de uso descreve as possíveis atividades de manutenção do cadastro de clientes, ou seja, permite incluir, alterar ou consultar clientes. Um cliente não pode ser excluído apenas tornado inativo**
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Informar o CPF ou CNPJ do cliente	
	2. Consultar o cliente por seu CPF ou CNPJ
	3. Se já houver um cliente cadastrado com o CPF ou CNPJ informados, apresentar seus dados.
4. Se necessário, alterar ou inserir os dados do cliente.	

	5. Se necessário, gravar as atualizações.
Restrições/Validações	**- o CPF ou CNPJ precisam ser validados** **- os campos nome, endereço e data de nascimento são obrigatórios.**

Tabela 2 – Documentação do Caso de Uso Manter Clientes. (Autoria própria).

Na Tabela 3 abaixo, é exibida a documentação do Caso de Uso **Manter Veículos**:

Sistema de Locação de Veículos **Especificação de Caso de Uso: Manter Veículos**	
Nome do Caso de Uso	**Manter Veículos**
Caso de Uso Geral	
Ator Principal	**Funcionário**
Atores Secundários	**Cliente**
Resumo	**Este caso de uso descreve as etapas percorridas por um funcionário para manutenção do cadastro de veículos da empresa, ou seja, permite a alteração de dados do mesmo, tais como, estado, fotos, opcionais.**
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Manter dados dos veículos atualizados	

	2. Registrar veículos
3. Informar o cliente sobre os planos de cada locação	
	4. Mostrar veículos disponíveis para locação
5. Atualizar sistema para após cada locação	
Restrições/Validações	

Tabela 3 – Documentação do Caso de Uso Manter Veículos. (Autoria própria).

E por fim, a seguir na Tabela 4, é mostrada a documentação do Caso de Uso **Devolver Locação:**

Sistema de Locação de Veículos **Especificação de Caso de Uso: Devolver Locação**	
Nome do Caso de Uso	**Devolver Locação**
Caso de Uso Geral	
Ator Principal	**Funcionário**
Atores Secundários	**Cliente**
Resumo	**Este caso de uso descreve as etapas necessárias para quando o cliente devolver o veículo.**
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
1. Verificar se o veículo encontra-se em perfeito estado	
	2. Registrar data e hora da

	devolução e a quilometragem
3. Verificar se o cliente permaneceu mais tempo com o veículo	
4. Verificar se o cliente permaneceu menos tempo com o veículo	
	5. Calcular valor se o cliente permaneceu mais tempo com o veículo
	6. Calcular valor se o cliente permaneceu menos tempo com o veículo
	7. Registrar locação
Restrições/Validações	

Tabela 4 – Documentação do Caso de Uso Devolver Locação. (Autoria própria).

6. CONCLUSÃO

Os profissionais da tecnologia da informação, tais como, engenheiros de softwares, programadores entre outros, quando vão construir um sistema, realizam de fato todas as etapas da engenharia de software, mas acabam pecando na fase da engenharia de requisitos, que é importante para a construção de um projeto de software. A grande maioria dos profissionais acaba não realizando corretamente a fase dos requisitos e com isso, acarretando em consequências como, custos mais altos do projeto e o prazo de entrega comprometido.

A Figura 1, mostrada na justificativa, ilustra bem algumas etapas do desenvolvimento de software segundo (SPÍNOLA, 2007). Pode se observar que devido a cada parte não realizada adequadamente, ou seja, não seguindo as atividades da engenharia de requisitos, faz com que o projeto fracasse, prejudicando todas as partes envolvidas. É de suma importância, conforme mostrado nos capítulos 3 e 4 a realização sistemática das atividades da engenharia de requisitos, bem como algumas técnicas e abordagens para que o projeto alcance o “sucesso”.

Na engenharia de requisitos pode se analisar vários fatores de riscos que podem comprometer o desenvolvimento do software, ela não faz apenas documentos que para muitos profissionais seriam uma perda de tempo, ela faz uma politica de segurança, de tratamento para diminuição dos riscos e custos. A engenharia de requisitos é o alicerce para o desenvolvimento de software, é considerado um fator essencial para o sucesso do produto software.

Com base no estudo de caso realizado, posso concluir que a utilização de diagramas de casos de uso para o processo da engenharia de requisitos é importante, visto que, facilita o entendimento dos requisitos da parte do cliente e auxilia o analista a projetar e gerenciar o sistema em que está inserido. Sobre as atividades do processo da engenharia de requisitos, pode-se concluir que é essencial na implementação de um sistema, o profissional da área seguir corretamente todas as etapas, pois, a fase de requisitos é uma das principais e mais complexas de se fazer.

7. REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS: **Citação:** NBR-10520/ago-2002. Rio de Janeiro: ABNT, 2002.

_____**. Referências:** NBR-6023/ago. 2002. Rio de Janeiro: ABNT, 2002.

AURUM, A, WOHLIN, C. **Engineering and Managing Software Requirements**, Springer, 2005.

AVÍLA, Ana Luiza. **Introdução a Engenharia de Software Magazine**, Disponivel em: <http://www.devmedia.com.br/revista-engenharia-de-software/8028>, 2007 Acesso em: 09 Fev. 2013.

GUEDES, Gilleanes T. A. **UML 2: Uma abordagem prática**. 1ª ed. São Paulo: Novatec Editora, 2009.

KOTONYA, Gerald. **Requirements Engineering: Process and Techniques**. 1ª ed. Person Addison Wesley, 1998.

PRESSMAN, Roger S. **Engenharia de Software**. 1° ed. São Paulo: Pearson Makron Books, 1995.

SPíNOLA, Rodrigo Oliveira. **Introdução a Engenharia de Software Magazine**, Disponivel em: <http://www.devmedia.com.br/revista-engenharia-de-software/8028> 2007. Acesso em: 09 Fev. 2013.

SOARES, Michel. **Qualidade de software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. 2ª ed. São Paulo, SP: Novatec, 2006.

SOMMERVILLE, Ian. **Engenharia de Software**. Traduzido por André Maurício de Andrade. 6ª ed. São Paulo: Person Addison-Wesley, 2003.

ALMEIDA, Ricardo. Engenharia de Requisitos: notas de aula: Departamento de Informática da Universidade Federal do Espírito Santo, nd. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Engenharia_Requisitos.pdf>, 2012. Acesso em: 03/03/2013.

CRISTEL, M.G e KANG, K. Issues in Requirements Elicitation**,** nd. Disponivel em: <http://www.sei.cmu.edu>. Acesso em: 03/04/2013.

Imagem retirada do link: <http://catarinagarrido.wordpress.com/2011/02/04/costumer-relationship-management/relacionamento-com-cliente/>. Acesso em: 26 Mar. 2013.

LEITE, J.C.S.P. Engenharia de Requisitos, Rio de Janeiro. Disponível em: <http://www.livrodeengenhariaderequisitos.blogspot.com>. 2007. Acesso em: 14/03/2013.

RICHA, Rafael. Estudo comparativo entre ferramentas de gerência de requisitos: Centro de Informática da Universidade Federal do Paraná, nd. Disponível em: <http://www.cin.ufpe.br/~tg/2009-2/rrta.pdf>, 2009. Acesso em: 25/02/2013.