AUTARQUIA EDUCACIONAL DO VALE DO SÃO FRANCISCO – AEVSF
FACULDADE DE CIÊNCIAS APLICADAS E SOCIAIS DE PETROLINA – FACAPE
CURSO DE ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

A Importância Do Uso De Uma Ferramenta No Gerenciamento de Requisitos

Ubirajara Santos Nogueira

Petrolina
2010

Ubirajara Santos Nogueira

A Importância Do Uso De Uma Ferramenta No Gerenciamento de Requisitos

Monografia apresentada no Curso de Especialização em Engenharia de Software, da Faculdade de Ciências Aplicadas e Sociais de Petrolina, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof° Msc. Jocélio de Oliveira Dantas Passos

Petrolina
2010

Ubirajara Santos Nogueira

A Importância Do Uso De Uma Ferramenta No Gerenciamento de Requisitos

Monografia apresentada ao curso de Especialização em Engenharia de Software da Faculdade de Ciências Aplicadas e Sociais de Petrolina - FACAPE.

Petrolina, 2010

_____

Prof° Msc. Jocélio de Oliveira Dantas Passos (orientador) – FACAPE

_____

Prof° Dr. Ricardo José Rocha Amorim

_____

Prof° Dra. Dinani Gomes Amorim

À minha esposa pelo apoio constante e à toda minha família por acreditar em mim.

Agradecimentos

Agradeço a Deus, pelo dom da vida, da inteligência e da sabedoria que me permitiram chegar até aqui, bem como por permitir mais esta evolução pessoal e profissional.

Ao meu orientador Prof. Ms. Jocélio de Oliveira Dantas Passos, pela orientação e por acreditar que eu conseguiria.

A todos que direta ou indiretamente colaboraram na execução deste trabalho.

E por fim, à minha querida esposa Gorete e aos sempre amados filhos Victor e Victoria, pelo amor e dedicação. Obrigado por tudo!

Resumo

A atividade de especificação de requisitos exige esforços consideráveis por parte dos profissionais que executá-la. Realizar a atividade, não é suficiente para obter os requisitos com os clientes. É necessário ter uma compreensão clara e completa das necessidades do cliente.

Entre as principais preocupações no desenvolvimento de *software*, destaca-se a necessidade de se entender e rastrear requisitos adequadamente. Essas atividades são realizadas em conjunto pelos engenheiros de requisitos, desenvolvedores, e clientes e/ou usuários que requerem o *software*. A engenharia de requisitos desenvolveu algumas técnicas, modelos de rastreamento de requisitos, e ferramentas para ajudar os engenheiros de requisitos.

Considerando as questões associadas à exigência de registro e de gestão, e a necessidade de otimização do processo, durante a concepção do produto, foram desenvolvidas ferramentas de apoio à atividade de gerência dos requisitos.

Demonstra-se, neste trabalho, como o uso de uma ferramenta de gerenciamento de requisitos pode ajudar uma equipe de engenharia de requisitos a gerenciá-los da melhor forma, desde a definição e todas as modificações sofridas pelos mesmos até a implementação do sistema.

Palavras-chave: Engenharia de *Software*, Requisitos, Ferramenta, CaliberRM.

ABSTRACT

The activity of requirements specification requires considerable effort on the part of professionals who run it. Perform the activity is not sufficient for the requirements with customers. You must have a clear and complete understanding of customer needs.

Among the main concerns in software development, there is a need to understand and track requirements adequately. These activities are undertaken jointly by the requirements engineers, developers, customers and / or users requesting the software. The requirements engineering has developed some techniques, models of requirements traceability, and tools to help requirements engineers.

Considering the issues related to the requirement of registration and management, and the need to optimize the process for product design tools have been developed to support activity management requirements.

It is demonstrated in this work, as the use of a requirements management tool can help a team of engineering requirements to manage them the best way, from the definition and all modifications made by them to implement the system.

Key words: Engineering, Software Requirements, tool, CaliberRM.

Lista de Figuras

Lista de tabelas

Lista de siglas

CPU	Central Processing Unit
GB	Gigabyte
HCI	Human Computer Interaction
HTML	Hyper Text Markup language
OOA	Object Oriented Analysis
PC	Personal Computer
QFD	Quality Function Deployment
RAM	Random Access Memory
RH	Recursos Humanus
SSA	Structured Systems Analysis
SSM	Soft Systems Methodology

SUMÁRIO

1 - Introdução

De acordo com as metodologias clássicas de desenvolvimento, o ciclo de desenvolvimento de um produto de software, tem início pelo processo de requisitos. Esse processo constitui-se, basicamente, de um conjunto de atividades divididas em elicitação, modelagem e análise de requisitos (Leite, 2001).

Segundo Sommerville (Sommerville, 2007), elicitação e análise de requisitos é o processo de derivação de requisitos de sistema através da observação de sistemas existentes, discussões com usuários potenciais e compradores, análise de tarefa, etc. Isso pode envolver o desenvolvimento de um ou mais modelos de sistema e protótipos. Eles ajudam o analista a compreender o sistema a ser especificado.

Estes modelos são criados a partir de um processo chamado modelagem e nada mais são do que representações simplificadas do mundo real. Essa simplificação facilita a compreensão da realidade e ajuda na busca de soluções para os problemas na forma de sistemas computacionais (Quadros, 2002).

A complexidade apresentada para a execução dessas tarefas determina que estas sejam algumas das atividades mais difíceis de um projeto de software. Erros cometidos nesta etapa podem implicar em alterações não apenas nos requisitos, mas também em muitos outros artefatos gerados no processo de desenvolvimento (Cardoso, 2003).

A correção de erros originados em requisitos implica em retrabalho, cujos custos são da ordem de 10 a 100 vezes o custo da correção desses mesmos erros, caso houvessem sido detectados nas etapas iniciais do processo de desenvolvimento (Sayão et al., 2003).

Desenvolver software é uma atividade complexa por natureza. Uma das razões para esta afirmação é que não existe uma única solução para cada cenário de desenvolvimento. Além disso, lidamos o tempo todo com pessoas (analistas, programadores e usuários), o que torna o sucesso do projeto bastante relacionado à competência da equipe e à forma como trabalham, e, para dificultar ainda mais, muitas vezes não fazemos uso de um processo bem definido para apoiar as atividades do projeto (Spínola, 2009).

Entende-se por processo, neste contexto, como sendo um conjunto de atividades bem definidas com os respectivos responsáveis por execução, ferramentas de apoio e artefatos produzidos. Ou seja, define-se como a equipe deverá trabalhar para alcançar o objetivo: desenvolver software com qualidade dentro de prazos, custos e requisitos definidos (Spínola, 2009).

Segundo Sommerville (2007), um processo de construção de um programa de computador é um conjunto de atividades e resultados associados que produz um *software*. Como é possível realizar todas essas atividades, com a qualidade necessária para alcançar os resultados esperados, sem uma ferramenta que dê todo o devido suporte? A quantidade de requisitos pode ser muito grande diante do problema a ser resolvido e a dificuldade em gerenciar tais requisitos aumenta proporcionalmente à quantidade deles. Introduzir uma ferramenta que automatize o processo de gerência dos requisitos é algo fundamental, pois ajudará a documentá-los, possibilitando uma melhor análise dos impactos de possíveis alterações e manterá a equipe totalmente atualizada com relação ao escopo e prazo do projeto, além de facilitar a comunicação entre as pessoas envolvidas no mesmo.

1.1 – Objetivos

1.1.1 – Objetivo Geral

Este trabalho tem como objetivo potencializar a importância do uso de uma ferramenta no gerenciamento de requisitos.

1.1.2 – Objetivos Específicos

- Entender os processos da engenharia de requisitos
- Apresentar a Ferramenta CaliberRM
- Avaliar a Ferramenta CaliberRM

1.2 – Motivação

Há um grande número de sistemas atualmente desenvolvidos e em utilização em todo o mundo, criados nas mais diversas linguagens de programação, utilizando as mais diversas técnicas e nos mais diversos segmentos de negócios aplicados (Araújo, 2009).

A utilização de técnicas para alcançar o objetivo final do sistema, pode parecer, em um primeiro momento, algo difícil de se conseguir, em função de mitos existentes, como por exemplo o de que "O estabelecimento geral de objetivos é suficiente para iniciar a escrita de programas – podemos fornecer os detalhes posteriormente" (Pressman, 2006). Quando na realidade embora uma declaração abrangente e estável dos requisitos nem sempre seja possível, uma definição inicial mal feita é a principal causa de esforços mal sucedidos de *software*. Uma descrição formal e detalhada do domínio da informação, da função, do comportamento, do desempenho, das interfaces, das restrições de projeto e dos critérios de validação é essencial. Essas características podem ser determinadas somente depois de intensa comunicação entre o cliente e o desenvolvedor (Caiado, 2009a).

Toda empresa desenvolvedora de *Software* vive o dilema: Qualidade X Tempo. Na avaliação das vantagens e desvantagens de se adotar um processo formal de desenvolvimento de *Software* não há dúvidas de que o tempo despendido na elaboração dos documentos é muito inferior ao tempo gasto na manutenção de sistemas sem documentação (Filho, 2010).

O registro e o gerenciamento das solicitações do cliente e as suas mudanças ao longo ciclo de vida do projeto exigem, da empresa desenvolvedora de *Software*, organização e padronização, para que o produto final apresente qualidade, proporcionando uma manutenção mais segura e a satisfação do cliente (Filho, 2010).

Além do surgimento de erros, não é incomum que os requisitos já definidos sofram alterações ao longo do processo de desenvolvimento. Mudanças no contexto onde o *software* está inserido, novas expectativas por parte dos clientes e usuários, negociação entre clientes e desenvolvedores, são apenas alguns exemplos de motivos que levam um requisito a ser modificado. Desta forma, mesmo que se faça

um bom trabalho de elicitação e gerência de requisitos, à medida que aumenta o número de requisitos que sofrem modificações, o esforço necessário para manter e gerenciar esses artefatos tende a crescer.

Assim, introduzir uma ferramenta que automatize o processo de gerência dos requisitos é algo fundamental, pois ajudará a documentá-los, possibilitando uma melhor análise dos impactos de possíveis alterações e manterá a equipe totalmente atualizada com relação ao escopo e prazo do projeto, além de facilitar a comunicação entre as pessoas envolvidas no mesmo. Grandes quantidades de requisitos não são gerenciáveis sem a utilização de alguma ferramenta computacional de apoio. Como exemplos de ferramentas existentes atualmente no mercado usadas para este fim, pode-se citar RequisitePro (RequisitePro, 2010), Doors (Doors, 2010) e Caliber-RM (CaliberRM, 2009). Nesse trabalho será analisado o uso da ferramenta CaliberRM.

1.3 – Metodologia

Os passos a serem seguidos para a aplicação da metodologia utilizada, encontram apresentados na Figura 1, que estão detalhados nas subsessões de 1.3.1 a 1.3.6.

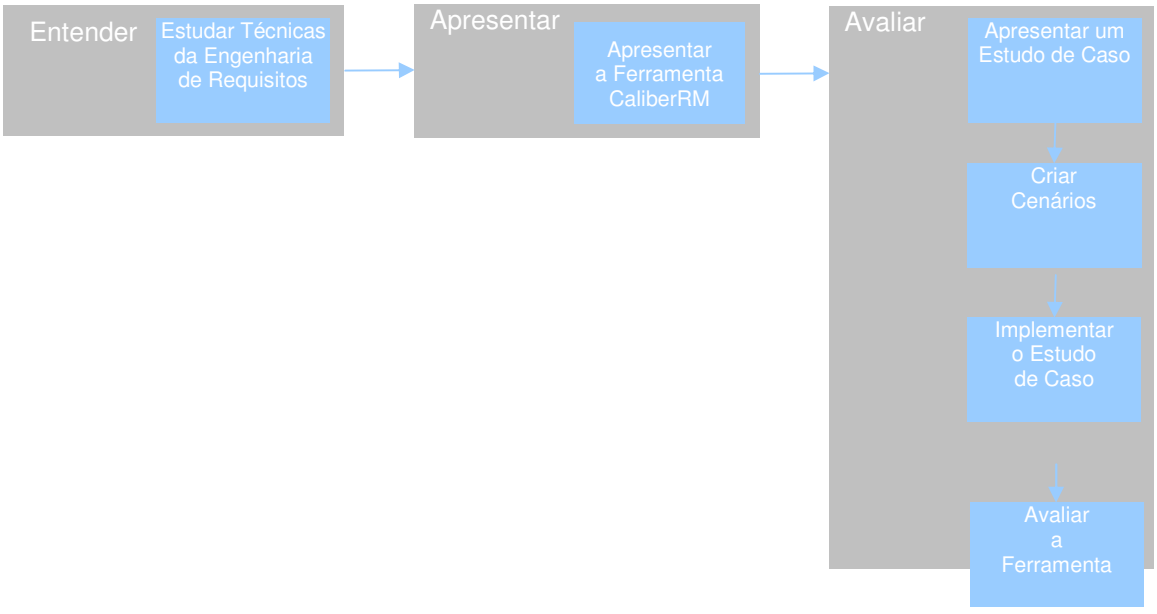

Figura 1 – Etapas da metodologia
Fonte: (Autor,2010)

1.3.1 – Estudar técnicas da Engenharia de Requisitos

Este passo é necessário para compreender os recursos que serão abordados na ferramenta utilizada bem como os critérios para sua avaliação.

1.3.2 – Apresentar a ferramenta CaliberRM

Será dada um visão geral sobre a ferramenta utilizada, sob o ponto de vista da empresa proprietária da referida ferramenta, com o objetivo de identificar se o propósito final da mesma é alcançado. Esta apresentação se dará através do estudo de telas e/ou manual disponível.

1.3.3 – Apresentar um Estudo de Caso

Será apresentado um estudo de caso visando guiar os testes da ferramenta. Este estudo de caso é simples mais o suficientemente bastante para demonstrar a eficácia da ferramenta no gerenciamento de requisitos. Chegou-se a este estudo de caso a partir das técnicas de pontos de vista (Sommerville, 2007) e etnografia (Sommerville, 2007). Será descrito também um cenário de gerenciamento de requisitos sem o suporte de nenhuma ferramenta. O estudo de caso está descrito na sessão 3.1 deste material.

1.3.4 – Criar Cenários

Segundo Sommerville (2007), cenários são exemplos de sessões onde os usuários simulam suas interações com o sistema. A partir dessas simulações da vida real, diferentes tipos de informações e níveis de detalhes sobre o sistema são extraídos, bem como o que os usuários estão fazendo e que informações eles necessitam do sistema para realizar a tarefa descrita no cenário. Esta técnica pode ser realizada informalmente com *stakeholders* para identifiar cenários e captar detalhes desses cenários.

Será descrito pelo engenheiro de requisitos, com a participação do auxiliar de

pessoal, cenários com exemplos de situações reais que ocorrem no desenvolvimento de suas atividades atualmente. Com isso poderá ser observada a modificação de requisitos sem e com a utilização da ferramenta. Será percebido como o uso de uma ferramenta de gerenciamento de requisitos poderá evitar atrasos nos prazos de entrega do projeto provocados por retrabalho.

1.3.5 – Implementar o Estudo de Caso

Após criado o estudo de caso, o mesmo terá seus requisitos implementados na ferramenta como pré-requisito para a etapa seguinte da metodologia. Serão digitados os requisitos do sistema, atores, requisitos do usuário, requisitos funcionais, requisitos não funcionais e restrições. Nos requisitos do usuário serão digitados os casos de uso. Nos requisitos funcionais serão informadas as alterações no sistema identificadas através dos cenários conforme descrito na sessão 3.2.

1.3.6 – Avaliar a ferramenta

Para a avaliação da ferramenta serão observadas restrições sobre os serviços ou as funções oferecidas pela ferramenta. Entre elas destacam-se restrições sobre o processo de desenvolvimento, confiabilidade e tempo de resposta. A seguir serão listadas um conjunto de características relevantes que serão usadas para a avaliação da ferramenta (Pressman, 2006):

1. Funcionalidade: a ferramenta faz com precisão o que se propõe a fazer?
2. Usabilidade : foi fácil usar a ferramenta?
3. Eficiência: a ferramenta realiza de forma rápida o que se propõe?
4. Confiabilidade: é segura com relação aos dados registrados?
5. Portabilidade: pode ser instalada em qualquer sistema operacional?
6. Compreensibilidade: disponibiliza ajuda para o usuário? É de fácil compreensão?

1.4 – Estrutura do Trabalho

Esta monografia está organizada basicamente em seis capítulos, sendo este

o primeiro. Tem-se no segundo capítulo a fundamentação teórica sobre a engenharia de requisitos, enquanto que no capítulo 3 descreve-se a apresentação do estudo de caso e dos cenários. No capítulo 4 apresenta-se a aplicação do estudo de caso na ferramenta e a análise enquanto no capítulo 5 apresenta-se a análise e a discussão dos resultados. E por último, o capítulo 6 que apresenta as considerações finais e os trabalhos futuros que podem estender esta pesquisa.

2 – Fundamentação Teórica

O objetivo deste capítulo é apresentar os conceitos de engenharia de requisitos, tais como: definição de engenharia de requisitos, gerência de requisitos, técnicas da engenharia de requisitos e elicitação.

2.1 – Engenharia de Requisitos

Segundo Loucopoulos (1995), a engenharia de requisitos pode ser dita como a área do conhecimento preocupada na comunicação com agentes organizacionais, com respeito a suas visões, intenções e atividades relativas às suas necessidades de suporte de computadores, desenvolvimento e manutenção de uma especificação de requisitos adequada a um sistema. Isto sugere que a engenharia de requisitos também inclua problemas e aspectos gerenciais, organizacionais, econômicos, técnicos, sociais e ambientais. Ainda neste contexto, Bubenko (2001), salienta que a própria definição da engenharia de requisitos evoluiu ao longo dos anos.

A engenharia de requisitos ajuda os engenheiros de *software* a compreender melhor o problema que eles vão trabalhar para resolver. Ela inclui o conjunto de tarefas que levam a um entendimento de qual será o impacto do *software* sobre o negócio, do que o cliente quer e de como os usuários finais vão interagir com o *software* (Pressman, 2006).

Em um processo de *software*, a engenharia de requisitos é a primeira atividade importante, após a conclusão de um relatório de necessidades resultante de um processo de pré-desenvolvimento. Ela é definida em função de suas atividades principais: entendimento dos problemas (descritos em um relatório de necessidades), determinação de soluções e especificação de uma solução que é testável, compreensível, manutenível e que satisfaça às diretrizes de qualidade do projeto (Peters, 2001).

A engenharia de requisitos tem sido reconhecida como uma das mais importantes fases do processo de engenharia de *software*. Este reconhecimento decorre da descoberta que a maior parte dos problemas e geralmente os mais dispendiosos e de maior impacto negativo no desenvolvimento de *software*, são

originados nas etapas iniciais do desenvolvimento. Estas etapas constituem o processo de engenharia de requisitos, no qual, as principais atividades podem ser definidas como: elicitação, análise, negociação, especificação, gerenciamento e validação de requisitos (kotonya et al., 1997). Normalmente, falhas na realização destas atividades, resultam em documentos de requisitos inconsistentes, incompletos e consequentemente produtos de *software* de baixa qualidade.

2.2 – Gerência de Requisitos

O gerenciamento de requisitos é um processo para compreender e controlar as mudanças dos requisitos do sistema. Precisa-se manter o acompanhamento dos requisitos e manter as ligações entre os requisitos dependentes, de modo que seja possível avaliar o impacto das mudanças de requisitos. Faz-se necessário estabelecer um processo formal para sugerir-se propostas de mudanças e ligá-las aos requisitos de sistema. O processo de gerenciamento de requisitos deve se iniciar assim que uma visão inicial do documento de requisitos esteja disponível, mas deve-se iniciar o planejamento das mudanças de requisitos durante o processo de elicitação de requisitos (Sommerville, 2007).

A gerência de requisitos está associada ao processo de engenharia de requisitos, uma sub área da Engenharia de *Software*, que engloba todas as atividades relacionadas com a produção de requisitos de *software* (Grings, 2009).

Grings (2009) define as principais atividades da gerência de requisitos como sendo:

- Controle de Mudanças: pressupõe um processo de mudança controlado e bem definido. Deve existir um mecanismo formal para solicitação de mudanças nos requisitos do *software*.
- Gerência de Configuração: apresenta duas atividades principais: controle de versões e controle de configuração. O controle de versões é a atividade relacionada à evolução dos artefatos, dando atenção aos artefatos gerados no processo de desenvolvimento, e às diferentes revisões e versões geradas por modificações. Já o controle de configuração, está relacionado às *releases* do *software* e o seu foco está voltado a diferentes configurações do *software*,

considerando possíveis combinações de artefatos.

- Rastreabilidade: Está no centro da atividade de gerenciamento de requisitos. É definida como a habilidade de se acompanhar a vida de um requisito durante todo o processo de construção.
- Gerência da qualidade de requisitos: a qualidade de um artefato de requisitos pode ser avaliada através da utilização de métricas, indicadores ou testes. Uma análise crítica desses artefatos pode ser realizada através de inspeções.

Uma das principais justificativas para empregar o esforço necessário para a realização dessas atividades é garantir que mesmo executando um bom processo de análise, construção e verificação de qualidade, não se obtenha, como resultado final, um *software* que não atenda às necessidades do cliente.

2.3 – Técnicas da Engenharia de Requisitos

Segundo (Zanlorenci, 2010), técnica é o conjunto de conceitos aplicáveis ao desenvolvimento dos processos.

As técnicas de engenharia de requisitos referem-se ao conjunto de conceitos aplicáveis às atividades para extrair, validar e manter um documento de requisitos. O papel dessas técnicas pode ser sumarizado como necessário para suportar as diferentes fases do processo de engenharia de requisitos. Tem-se vários tipos de técnicas, como: (Zanlorenci, 2010)

- Processos ER;
- Comunicação humana;
- Desenvolvimento do conhecimento;
- Documentação de requisitos;
- Gerenciamento de requisitos.

Para definir quais técnicas do conteúdo do portfólio são necessárias e quando usá-las, é necessário identificar os cenários envolvidos sob o enfoque de dois componentes básicos: (Zanlorenci, 2010)

1. O relacionamento entre o fornecedor de *software* e o cliente;
2. O modelo do processo de engenharia de requisitos.

Existe uma série de abordagens para o entendimento do problema de

requisitos (Zanlorenci, 2010):

- Marketing – interessada no relacionamento entre requisitos e o sucesso de um produto no mercado (suporte ao usuário, pesquisas, grupos de usuários, apresentações produto, grupos de foco, análise de mercado,...);
- Psicologia e Sociologia – interessada no relacionamento entre requisitos e necessidades de pessoas como seres inteligentes e sociais (entrevistas com usuário, observações, gravação de comportamento do usuário em vídeo, experimentos pensando em voz alta, jogos de cartas, estudos etnográficos,...);
- Análise Orientada a Objetos (OOA) – interessada no relacionamento entre requisitos e o processo de desenvolvimento do *software*, iniciando de uma perspectiva de objetos do mundo real (modelo de objetos, comportamentos,...);
- Análise Estruturada (SSA) – interessada no relacionamento entre requisitos e o processo de desenvolvimento de *software*, iniciando de uma perspectiva de processos e dados;
- Projeto Participativo (*Participative Design*) – interessada em requisitos como parte de um processo que permite o envolvimento ativo do usuário no projeto de sistemas que afeta seu próprio trabalho (reuniões conjuntas e participativas);
- Interação Computador-Humanos e Fatores Humanos (HCI) – interessada na aceitabilidade de sistemas para pessoas, a usabilidade dos sistemas e o relacionamento entre requisitos e avaliação do sistema em uso;
- Método Sistemas Soft (SSM) – interessada no relacionamento entre requisitos e como as pessoas trabalham como parte de um sistema organizacional;
- Qualidade – interessada no relacionamento entre requisitos e a qualidade de um produto, em relação ao processo de aperfeiçoamento que conduz à satisfação do cliente (QFD - *quality function deployment*);
- Representação Formal Ciência da Computação – interessada no

relacionamento entre requisitos e a necessidade de precisão da engenharia de *software*.

2.4 – Classificação de Requisitos

Durante o processo de especificação dos requisitos, surge também a necessidade de estabelecer o tipo de requisito de que se está tratando, a fim de melhorar a compreensão das necessidades do cliente, bem como modelar melhor esta necessidade (Wazlawick, 2004).

De forma geral, podemos categorizar os requisitos, em três classes básicas distintas, mas que podem estar relacionadas: funcionais, não-funcionais e de domínio (Loucopoulos et al., 1995). Alguns autores (Sommerville, 2007; Chung et al., 2000) preferem classificar os requisitos somente segundo os dois primeiros tipos, funcionais e não funcionais.

Os requisitos funcionais dizem respeito à definição das funções que um sistema ou um componente de sistema deve fazer. Eles descrevem as transformações a serem realizadas nas entradas de um sistema ou em um de seus componentes, a fim de que se produzam saídas (Sommerville, 2007).

Os requisitos não-funcionais dizem respeito a restrições, aspectos de desempenho, interfaces com o usuário, confiabilidade, segurança, manutenibilidade, portabilidade, padrões, e outras propriedades que o sistema deve possuir, bem como aspectos sociais e políticos. Alguns desses requisitos são provavelmente traduzidos em funções (operacionalizados), ao longo do processo de desenvolvimento de *software* (Chung et al., 2000).

Os requisitos de domínio são provenientes do ambiente de onde o sistema será utilizado e que refletem as características e as restrições desse ambiente. Podem ser funcionais ou não funcionais.

De forma geral, a diferença entre funcionais e não-funcionais está no fato dos primeiros descreverem o que o sistema deve fazer, enquanto que os outros fixam restrições sobre como os funcionais serão implementados (Sommerville, 2007).

Os requisitos podem ser vistos como sendo "o que" o sistema deve fazer, associado com "o porque" deve fazer, em lugar do "como". Procura-se ampliar a

visão tradicional dos requisitos que trata apenas de aspectos funcionais e não-funcionais, com informações organizacionais que abordam a intencionalidade dos fatos, ou seja, os requisitos organizacionais (Loucopoulos et al., 1995).

2.5 – Elicitação de Requisitos

Dentro da engenharia de requisitos cabe à elicitação a tarefa de identificar os fatos que compõem os requisitos do sistema, de forma a prover o mais correto e mais completo entendimento do que é demandado daquele *software*. Captura de informações é um processo de descoberta que objetiva aumentar o conhecimento do objeto em questão, buscando o máximo de informações possível. Para isso é necessária uma habilidade para trabalhar com especialistas humanos e com o conhecimento tácito, que é trivial para quem tem o conhecimento, mas não é para quem o procura, sendo dificilmente lembrado e, portanto, não transmitido (Goguen, 1994).

2.6 – Ferramenta CaliberRM

O Borland CaliberRM (CaliberRM, 2009), automatiza o gerenciamento dos requisitos através de um repositório único e centralizado, permitindo que as equipes de projetos entreguem produtos com um melhor nível de qualidade e, principalmente, atendendo às necessidades do cliente (Caiado, 2009b).

O CaliberRM é um sistema de automação de gerenciamento de requisitos cliente servidor desenvolvido pela Borland. A ferramenta possui objetivos similares ao RequisitePRO (RequisitePro, 2010) e DOORS(Doors, 2010), porém, ao invés de tratar os requisitos como documentos, o CaliberRM os trata como um grupo de objetos integrados e reutilizáveis que são armazenados em repositório central, permitindo que um mesmo requisito possa ser utilizado em vários projetos gerenciados pela ferramenta.

O CaliberRM considera o papel do administrador para definir projeto, fazer backup e controlar o acesso dos usuários. Inicialmente, o usuário deve definir um projeto para definir requisitos, visões, matrizes de rastreamento, etc. O

suporte do Caliber para o rastreamento de requisito permite ao usuário visualizar relacionamentos entre requisitos e outras informações, tais como: outros requisitos, conjunto de teste, caso de uso e classes. O Caliber possui interface para vários aplicativos como Excel (Microsoft, 2010), arquivos gráficos, HTML (W3C, 2010), áudio e vídeo, fornece funcionalidades para apoiar e registrar informações relacionadas com problemas com os requisitos.

O CaliberRM se propõe a: criar tipos de requisitos e seus atributos, cadastrar e editar requisitos, controlar e manter histórico de alterações, criar o *Baseline*[1] (Peters, 2001) e controle de versões, manter fórum de discussões, integrar-se com outras ferramentas, dar suporte à segurança, controlar o acesso concorrente e o acesso Web. Dentre esses itens não serão analisados neste trabalho: integração com outras ferramentas, *baseline*, suporte à segurança e controle de acesso concorrente e de acesso Web.

Para iniciar-se o uso da ferramenta, após a instalação da mesma, deve-se escolher a opção Borland CaliberRM / CaliberRM através do menu iniciar. A Figura 2 é a tela inicial da ferramenta. Para efetuar-se o *login* na ferramenta, faz-se uso do *user* ADMIN e *password* admin, onde em seguida tem-se a tela que possibilita o acesso aos recursos da ferramenta (conforme Figura 3).

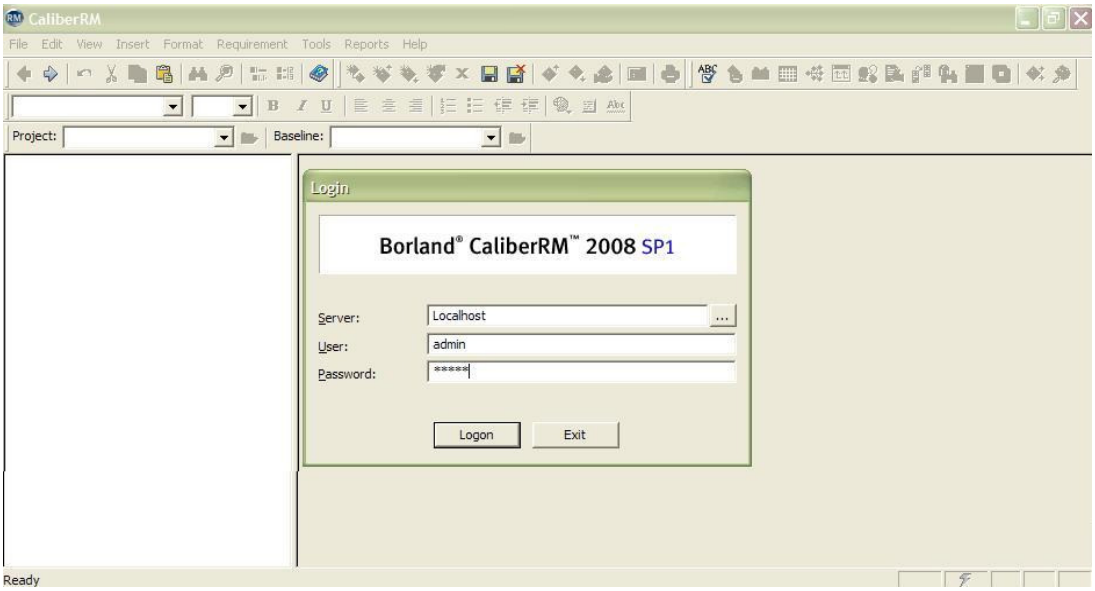

Figura 2 – Tela de logon do CaliberRM
Fonte: (CaliberRM, 2009)

[1] Uma baseline é um artefato de *hardware* ou de *software* que tenha sido revisto formalmente e consentido, e que serve como base para posterior desenvolvimento (Peters,2001).

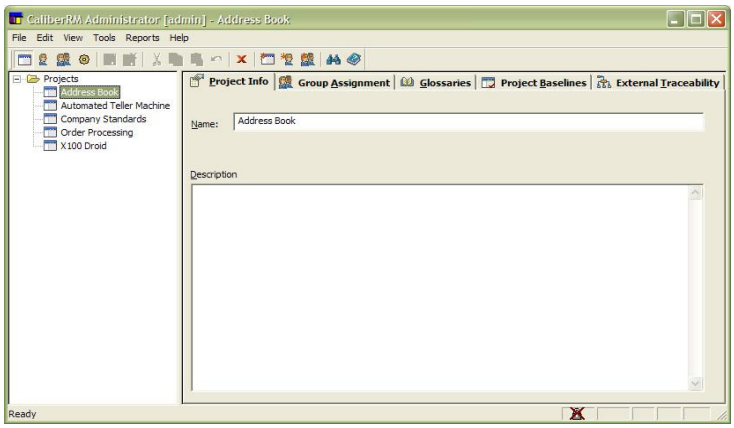

Figura 3 - Tela inicial do CaliberRM
Fonte: (CaliberRM, 2009)

2.6.1 – Especificações Técnica

Para instalação e utilização da ferramenta, é necessário o *Windows Server* 2003 nas versões *standard edition* ou *enterprise edition* com SP1 ou SP2, com um *hardware* mínimo com 4 GB de RAM, processador Dual CPU e espaço mínimo em disco de 1 GB. A Tabela 1 identifica os requisitos mínimos de sistema para instalar e executar o CaliberRM.

Plataformas	Windows Server 2003 Standard Edition R2 SP1, 32-bit Windows Server 2003 Standard Edition R2 SP2, 32-bit Windows Server 2003 Enterprise Edition R2 SP1, 32-bit Windows Server 2003 Enterprise Edition R2 SP2, 32-bit Windows Server 2003 Standard Edition SP2, 32-bit Windows Server 2003 Enterprise Edition SP1, 32-bit Windows Server 2003 Enterprise Edition SP2, 32-bit
Hardware	4 GB RAM Dual CPU
Espaço em disco rígido	1 GB (mínimo), 30 GB ou superior (recomendado).

Tabela 1: Requisitos mínimos para instalação do CaliberRM
Fonte: (CaliberRM, 2009)

A versão do CaliberRM que foi utilizada para avaliação, é a 2008 SP1 *release*

10.0.3.45.000, instalada no windows XP e obtida da *internet* [2].

[2] https://borland.subscribenet.com/control/borl/download?element=1099303

3 – Apresentação do Estudo de Caso e Cenários

Este capítulo trata da criação do cenário, iniciando-se com sua definição e registrando-se a evolução de alguns requisitos, evolução essa sugerida pelo usuário durante as interações com a equipe responsável pelo cenário.

3.1 – Apresentação do Estudo de Caso

A partir da aplicação das técnicas de pontos de vista e entrevista, chegou-se a conclusão de que há a necessidade em agilizar o processo de verificação de débitos quando os colaboradores forem desligados da empresa. Essa verificação se dá de maneira sequencial, ou seja, não há dia exclusivo para a demissão de colaboradores. Portanto o sistema deve, permitir que cada um dos setores possam informar o valor do débito do colaborador a ser desligado. O sistema também deverá permitir que cada desconto informado seja descontado na rescisão de contrato de trabalho do colaborador.

3.1.1 – Definição de Requisitos do Usuário

Necessita-se de um sistema para Quitação de Débitos que possibilite efetuar os descontos dos débitos diretamente na rescisão do colaborador. O usuário necessita que o sistema permita: cadastrar setor, manter quitação e efetuar o acompanhamento da quitação.

Este sistema deve interagir com os sistemas de controle de usuários onde será feita a validação do usuário e com o sistema de folha de pagamento onde será feito o desconto dos débitos.

Há dois grupos de usuários do sistema: auxiliar de pessoal e atendente do setor.

A Figura 4 representa os requisitos do usuário.

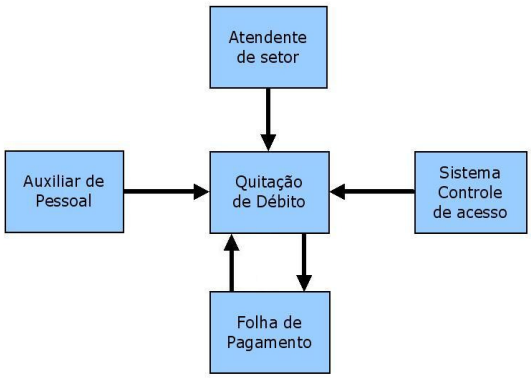

Figura 4 – Definição de requisitos do usuário
Fonte: (Autor, 2010)

3.1.2 Especificação de requisitos do sistema

Nesta subsessão, especifica-se os requisitos do sistema sob a visão do usuário e da equipe responsável pelo projeto e estão classificados em funcionais e não funcionais.

3.1.2.1 Requisitos Funcionais

Os requisitos funcionais foram identificados através da aplicação da técnica de levantamento de requisitos chamada cenários descritos na subsessão 3.2. Os requisitos identificados foram:

1) Cadastrar Setores:
   a) Fornecer, registrar e informar um número (código seqüencial);
   b) Solicitar dados cadastrais: nome, usuário, código da folha.

2) Gerar Quitação de Débitos:
   a) Fornecer, registrar e informar um número (código seqüencial);
   b) Solicitar dados cadastrais: data e colaborador;
   c) Clicar no botão <Salvar>;
   d) O sistema deverá registrar e informar o valor '1 – Aberta' para o campo status da quitação;

e) Para cada setor cadastrado, gerar o registro recuperando e registrando o código da quitação, recuperar o código de cada setor cadastrado, registrando data nula e valor nulo;
f) O sistema deve enviar e-Mail para todos os usuários dos setores informando o número da quitação criada e as informações de data e colaborador.

3) Quitar Débito:
a) Acessar o sistema, informando o número da quitação;
b) Clicar no botão <Localizar> e confirmar os dados da quitação;
c) Clicar no botão <Quitar>;
d) O sistema deve associar o setor ao usuário, não permitindo um usuário ver o setor do outro;
e) Informar o valor a descontar;
f) Clicar no botão <Salvar>;
g) O sistema deverá registrar a data e hora da quitação;
h) O sistema deverá atualizar o campo status da quitação para '2 - Pendente'. Quando não houver mais nenhum setor faltando valores, deverá ser '3 – Quitada'.

4) Acompanhar Quitação;
a) Acessar o sistema, informando o número da quitação;
b) Clicar no botão <Localizar> e confirmar os dados da quitação;
c) Clicar no botão <Acompanhar>;
d) O sistema deverá exibir para cada setor da quitação as informações nome do setor, data e valor.

5) Atualizar Desconto:
a) Acessar o sistema, informando o número da quitação;
b) Clicar no botão <Localizar> e confirmar os dados da quitação;
c) Clicar no botão <Atualizar>;
d) Neste momento o sistema se comunicará com a folha de pagamento;
e) Para cada setor da quitação, o sistema deverá recuperar do cadastro de

setores o campo código folha e o campo valor do setor da quitação;

f) O sistema deverá habilitar o botão <Atualizar> somente se todos os setores possuírem registro de data e valor;
g) O sistema deverá exportar o valor para a Folha de Pagamentos informando o valor a descontar no campo Código da Folha no cadastro de setor;
h) O sistema deverá alterar o status da quitação para '4 – Lançada'.

3.1.2.2 Requisitos não Funcionais

Os requisitos não funcionais foram identificados através da aplicação da técnica de levantamento de requisitos chamada cenários e que estão descritos na subsessão 3.2. Os requisitos identificados foram:

1) Confiabilidade

- O sistema não deve permitir que usuários visualizem ou alterem valores que não sejam do(s) setor(es) sob sua responsabilidade;
- Não será permitido que o usuário auxiliar de pessoal faça alterações nos valores.

2) Usabilidade

- A interface deve ser fácil para o usuário;
- O atendente do setor não deverá levar mais que 1 (um) minuto para:
  - Consultar um código de quitação;
  - Informar o valor do débito.

3) Transportabilidade

- Disponibilizar, num painel de controle, configuração de setores, responsáveis pela verificação dos débitos e código para desconto em folha**.**

4) Portabilidade

- O sistema deve ser portável o suficiente para rodar em equipamentos PC's com diferentes configurações de hardware.

5) Compreensibilidade

- Todas as funções do sistema, bem como suas possibilidades, devem ser explicadas para o usuário em forma de *Help Online*, acessados via tecla F1, sensível ao contexto.

6) Implantabilidade

- O sistema deve ser implantado em no máximo 1 (um) dia;
- Deve haver rotinas que permitam integração com o sistema de folha de pagamento, sendo que os valores informados nele possam ser transportados automaticamente.

7) Integridade

- O usuário do sistema deve ser identificado com um *login* e uma senha de 6 dígitos no mínimo;
- Realizar controle de acesso por usuário, onde será disponibilizado para o usuário do sistema apenas as opções disponíveis para ele;
- Solicitar *login* e senha para acesso ao sistema;
- Registrar *log* de entrada e saída para usuários.

Ainda na identificação dos requisitos foram identificadas as seguintes características que o sistema não deve possuir:

1) Permitir que em uma quitação seja informado um colaborador que possua situação diferente de Demitido;
2) Exibir dados de setores para usuários que não são responsáveis por ele ou para o usuário auxiliar de pessoal;
3) Permitir manutenção em uma quitação já lançada no sistema de folha de pagamento.

3.1.3 Modelos do Sistema

Após definidos todos os requisitos, apresenta-se os modelos gráficos que representam as funcionalidades que o sistema deverá apresentar ao usuário, que

são:

1) Análise da tarefa

Após identificar-se os objetivos do usuário, suas tarefas, quais estratégias utiliza para alcançar seus objetivos, como o usuário lida com emergências, que ferramentas utiliza e quais problemas ele encontra, chegou-se a uma visão da aplicação sob a perspectiva do usuário, ou seja, um modelo das tarefas do usuário executando a aplicação (conforme Figura 5).

As tarefas identificadas são:

- Criar quitação: o usuário informa matrícula, data da quitação e nome do colaborador.
- Quitar débito: o usuário informa um número de quitação e o valor a descontar.
- Acompanhar quitação: o usuário informa o número da quitação a consultar.
- Atualizar quitação: o usuário informa um número de quitação e clica no botão Atualiza.

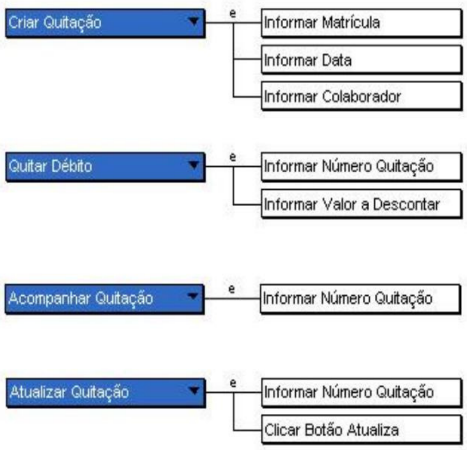

Figura 5 – Análise da tarefa
Fonte: (Autor, 2010)

2) Diagrama de casos de uso

Para descrever as interações típicas entre os usuários do sistema e o próprio sistema, gerou-se o diagrama de casos de uso, após identificação dos atores e das relações destes com as diversas funções do sistema (ver Figura 6).

Os atores identificados foram: auxiliar de pessoal, colaborador do setor e sistema de folha de pagamento.

As funções do sistema são: criar “Quitação de Débitos”, verificar “Quitação de Débitos”, informar valor, verificar situação, processar desconto, retornar dados e armazenar informações.

O ator auxiliar de pessoal se relaciona com as seguintes funções do sistema: criar “Quitação de Débitos”, verificar “Quitação de Débitos”, verificar situação e processar desconto.

O ator colaborador do setor mantém relações com as seguintes funções do sistema: verificar “Quitação de Débitos” e informar valor.

O ator sistema de folha de pagamento relaciona-se com as seguintes funções do sistema: retornar dados e armazenar informações.

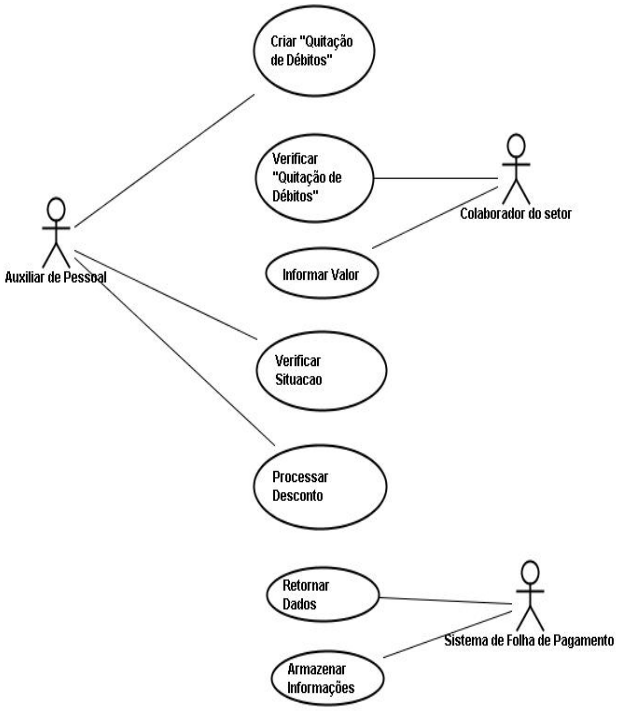

Figura 6 – Diagrama de caso de uso

Fonte: (Autor, 2010)

3) Diagrama de Atividades

Através do diagrama de atividades, procurou-se descrever a lógica de procedimento, o processo de negócio e o fluxo de trabalho, além de demonstrar a dependência entre as atividades (conforme Figura 7).

O auxiliar de pessoal cria a quitação de débitos, atividade que recebe dados do colaborador vindos do sistema de folha de pagamento e gera a atividade de verificação da quitação de débitos por parte do colaborador do setor que após informar o valor a descontar gera a atividade de verificação do status da quitação de débitos para o auxiliar de pessoal. Na sequência, o auxiliar de pessoal processa o desconto da quitação no sistema de folha de pagamento.

O colaborador do setor verifica quitação de débito e informa o valor, informação que será visualizada pelo auxiliar de pessoal.

O sistema de folha de pagamento retorna dados do colaborador e armazena dados da quitação. Também é responsável pelo desconto do valor da quitação na rescisão de contrato de trabalho do colaborador.

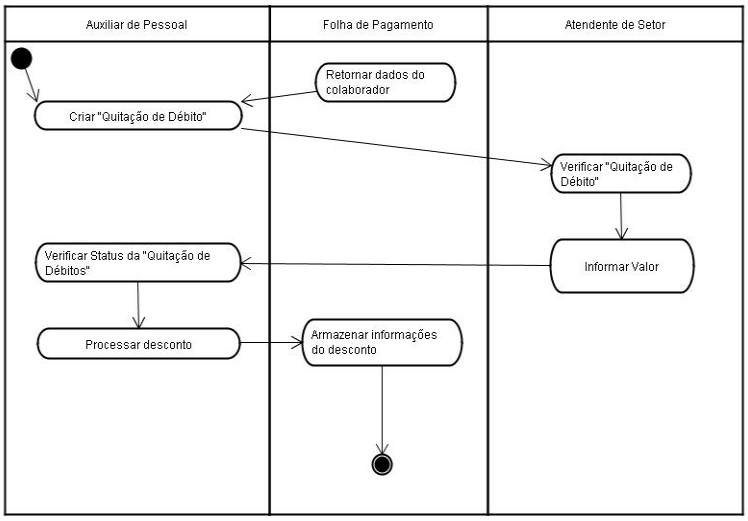

Figura 7 – Diagrama de atividades
Fonte: (Autor, 2010)

4) Diagrama de sequência

Através do diagrama de sequência, procurou-se demonstrar a sequência dos processos e a interação entre os mesmos, onde objetos e mensagens são passadas entre esses objetos (ver Figura 8).

O auxiliar de pessoal gera a quitação de débitos e recebe do usuário do setor o valor informado na quitação. Ele também verifica o *status* da quitação recebendo dados sobre a mesma, exporta valores ao processar o desconto da quitação e recebe a mensagem de fechamento da quitação.

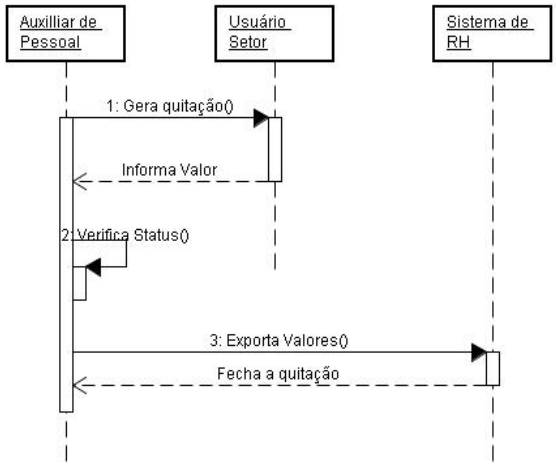

Figura 8 – Diagrama de sequência
Fonte: (Autor, 2010)

5) Diagrama de Classes

Através do diagrama de classes, procurou-se descrever os tipos de objeto presentes no sistema e os vários tipos de relacionamentos estáticos existentes entre eles. Identificou-se os seguinte tipos de objetos: setor, usuário, quitação, valor desconto, acompanhamento, atualização do desconto. Os tipos de relacionamento entre estes objetos são (conforme Figura 9):

- Um setor possui somente um usuário e um usuário possui apenas um setor;
- Um usuário realiza uma quitação e uma quitação é realizada por apenas um usuário;
- Um usuário informa um valor de desconto e um valor de desconto é informada por apenas um usuário;
- Um usuário faz um acompanhamento e um acompanhamento é feito

por apenas um usuário;

- Um usuário atualiza um desconto e um desconto é atualizado por apenas um usuário.

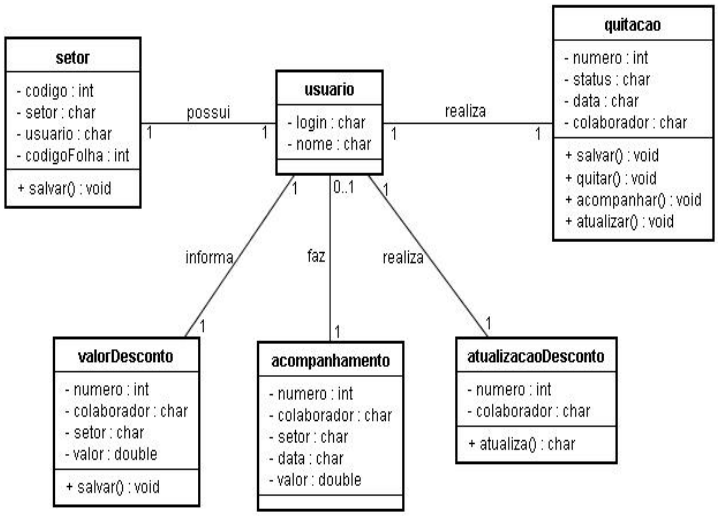

Figura 9 – Diagrama de classes
Fonte: (Autor, 2010)

6) Diagrama de contexto

Com o diagrama de contexto, ilustra-se em um nível alto de abstração, quais elementos externos interagem com que funcionalidades do sistema, ou seja, apresenta-se os elementos externos do sistema e as maneiras segundo as quais eles as utilizam (ver Figura 10).

A quitação de débito recebe nome e e-mail do usuário enviados pelo sistema de controle de acesso e recebe matrícula e nome do colaborador vindos do sistema de folha de pagamento. Ela envia a matrícula, o código da rubrica e o valor a descontar para a folha de pagamento.

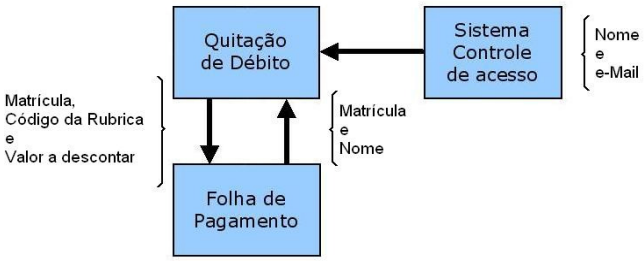

Figura 10 – Diagrama de contexto
Fonte: (Autor, 2010)

3.2 – Apresentação dos Cenários

Descreve-se dois cenários com exemplo de situação real que relata a utilização do formulário da quitação de débitos pelo auxiliar de pessoal quando da demissão de um colaborador. O cenário 1 (um) representa a primeira interação entre o usuário e o engenheiro de software. Enquanto que o cenário 2 (dois) apresenta a necessidade de modificação em requisitos previamente identificados no cenário 1 (um), após nova interação entre o analista de requisitos e o usuário auxiliar de pessoal.

1) Cenário 1

Atores: auxiliar de pessoal (ator principal), contínuo, colaborador do setor, sistema de folha de pagamento e analista de requisitos.

Ambiente: levantamento de valores monetários a serem descontados de um colaborador demitido quando da geração de sua rescisão pelo sistema de folha de pagamento.

Roteiro: o auxiliar de pessoal recebe um documento informando sobre a demissão de um determinado colaborador e preenche o formulário quitação de débitos. No preenchimento da quitação de débitos, o atendente anota a matrícula e o nome do colaborador na quitação, que já contém os nomes dos setores por onde a mesma deverá passar. Depois de preenchida a quitação, a mesma é entregue ao contínuo que a leva ao primeiro setor da lista de setores da empresa onde o colaborador demitido poderá ter valores a ser descontado. Cada setor tem um atendente responsável por receber a quitação e efetuar anotações na mesma. O atendente do setor verifica se o colaborador demitido tem algum valor a ser descontado e anota o valor na quitação no espaço reservado ao seu setor. Em seguida assina a quitação e a devolve ao contínuo que vai passar pelos demais setores da lista. O contínuo poderá não esperar e retornar depois para a apanhar a quitação. Após a quitação ter sido preenchida por todos os setores listados, o contínuo leva a quitação para o auxiliar de pessoal. Se a quitação demora muito a

retornar, o atendente de pessoal efetua a cobrança aos setores para descobrir onde a mesma está e pede para que seja passada adiante. Quando a quitação retorna, o auxiliar de pessoal informa cada valor a ser descontado, em uma determinada rubrica, no sistema de folha de pagamento que efetuará os referidos descontos quando a rescisão de contrato do colaborador for gerada.

2) Cenário 2

Atores: auxiliar de pessoal e analista de requisitos.

Ambiente: levantamento de valores monetários a serem descontados de um colaborador demitido quando da geração de sua rescisão pelo sistema de folha de pagamento.

Roteiro: ao preencher o formulário quitação de débitos, o auxiliar de pessoal informa também a data de criação da quitação. O atendente do setor ao anotar o valor a ser descontado preenche também a data da anotação. Esta falha na definição do cenário identificada pelo usuário e conhecido apenas pelo analista de requisitos, dificulta que os demais membros da equipe de desenvolvimento conheçam as alterações exigidas após a modificação do cenário. O sistema encontra-se na etapa de implementação e o projetista do sistema terá que efetuar alterações. O DBA deverá efetuar alterações no banco de dados. O programador será notificado para proceder as alterações no código fonte do sistema. Em decorrência desta falha não identificada na definição do cenário 1 (um), gerou atraso no cronograma de desenvolvimento do projeto.

Na sessão 4 será demonstrado como o uso de uma ferramenta de gerenciamento de requisitos poderá evitar prejuízos diante de situações descritas nos cenários 1 e 2.

4 – Utilização da ferramenta

Esta sessão apresenta a utilização da ferramenta CaliberRM no gerenciamento dos requisitos, que foi realizada com a aplicação do estudo de caso apresentado na subsessão 3.1.

Para criação do projeto denominado quitação de débitos, foi utilizado o módulo administrador do CaliberRM, iniciado através do botão iniciar / todos os programas / Borland CaliberRM / *Administration* / *Administrator*. A tela inicial do sistema é a tela de *login* onde fez-se uso do *user* ADMIN e *password* admin. A tela seguinte possibilita criar o projeto através do menu *File* / *New project*. O primeiro passo foi dar um nome ao projeto, que neste caso foi identificado como quitação de débitos (ver Figura 11).

Figura 11 – Passo 1 da criação do projeto: definição do nome
Fonte: (CaliberRM, 2009)

Depois de definido o nome do projeto, obteve-se a tela que permitiu entrar com uma descrição para o mesmo. Neste momento entrou-se com o requisito do usuário sendo a justificativa para o desenvolvimento do sistema (conforme Figura 12). Este requisito encontra-se identificado na subsessão 3.1.1.

Figura 12 – Passo 2 da criação do projeto: descrição
Fonte: (CaliberRM, 2009)

Após informada a descrição do projeto definiu-se quais grupos de usuários, disponíveis no CaliberRM, participariam do mesmo. Neste caso definiu-se somente o grupo de administradores (ver figura 13).

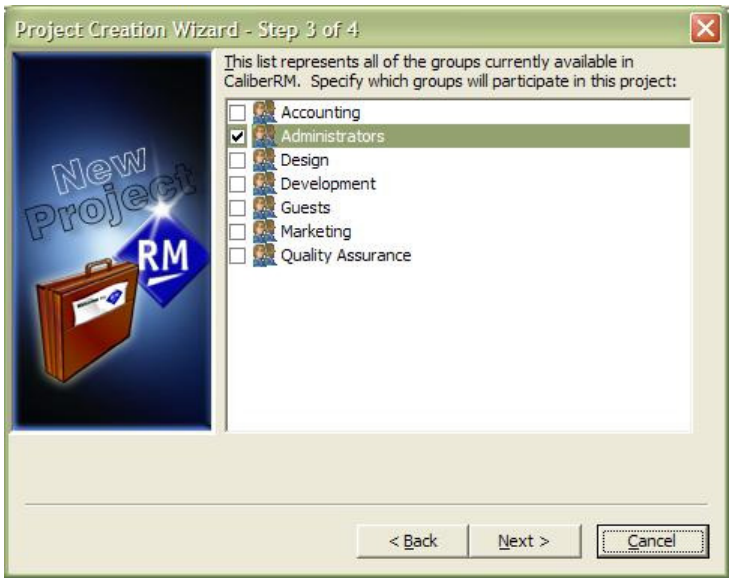

Figura 13 – Passo 3 da criação do projeto: definição dos grupos participantes
Fonte: (CaliberRM, 2009)

Depois de criado o projeto, iniciou-se o gerenciamento dos requisitos. Para

isso a ferramenta CaliberRM deve foi iniciada através do botão iniciar / todos os programas / Borland CaliberRM / CaliberRM. Uma vez efetuado o logon escolheu-se o projeto quitação de débitos através do *drop down Project* localizado na barra de ferramentas (conforme Figura 14).

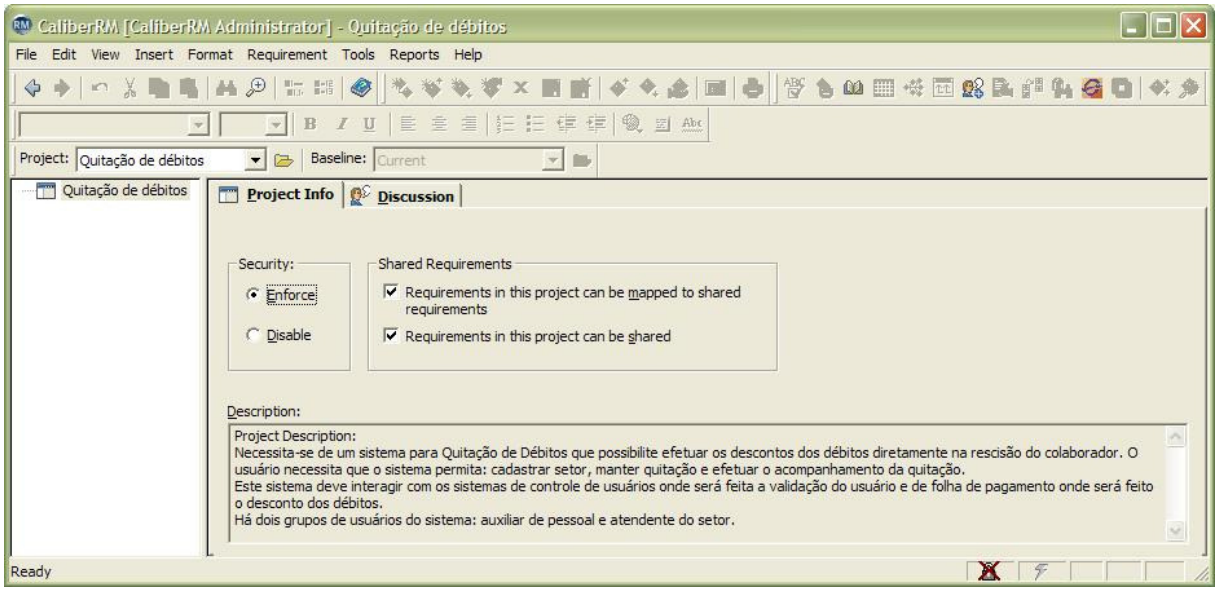

Figura 14 – Tela do CaliberRM com o projeto selecionado
Fonte: (CaliberRM, 2009)

A próxima etapa foi entrar os requisitos na mesma ordem estabelecida na subsessão 3.1. Para isso clicou-se com o botão auxiliar no nome do projeto e escolheu-se a opção *New* / *Requirement Type*. Primeiro entrou-se com os requisitos de sistema, que são os requisitos especificados segundo a visão do usuário e da equipe responsável pelo projeto. Para isto criou-se um tipo de requisito chamado requisito de sistema com o objetivo de apresentar o sistema e informar uma descrição geral sobre o mesmo. Depois de criado o tipo de requirimento, criou-se os requisitos apresentação e descrição geral (ver figura 15). Para cadastrar cada um dos requisitos clicou-se com o botão auxiliar no tipo de requisito e escolheu-se a opção *New* / *Child*.

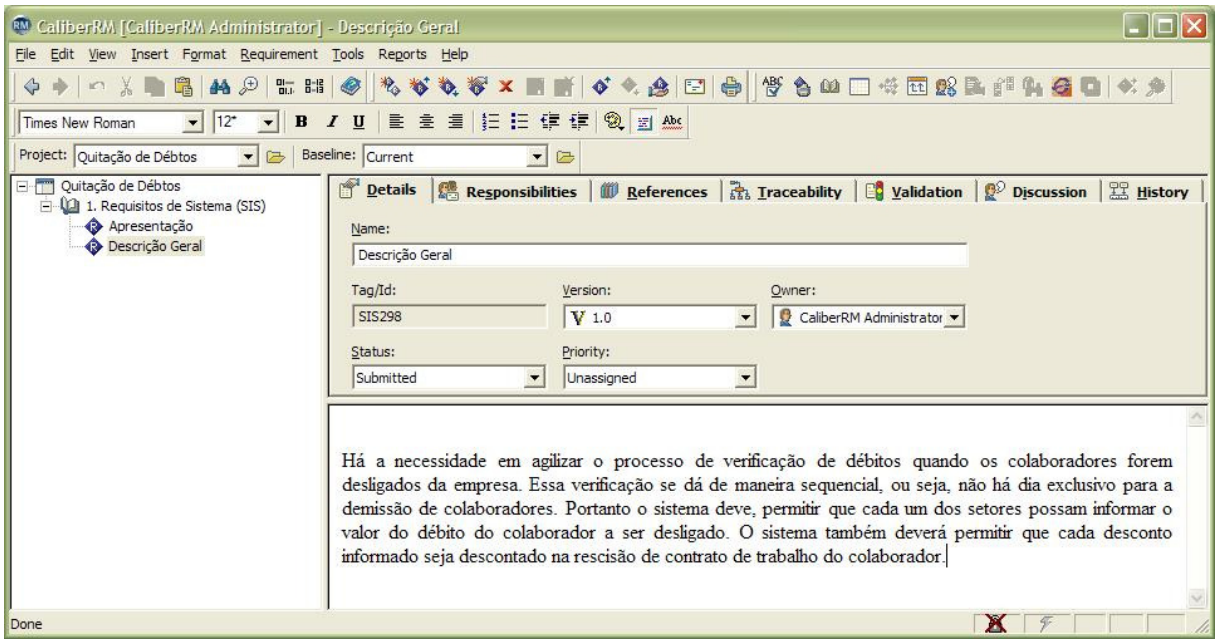

Figura 15 – Tela com os requisitos de sistema
Fonte: (CaliberRM, 2009)

Durante o processo de cadastramento de um requisito, é importante destacar três etapas: definição de responsabilidades, acompanhamento das discussões e histórico.

A ferramenta permite que informe-se os responsáveis pelo requisito através da aba *Responsabities* onde pode-se escolher os membros responsáveis pelo requisito (conforme Figura 16).

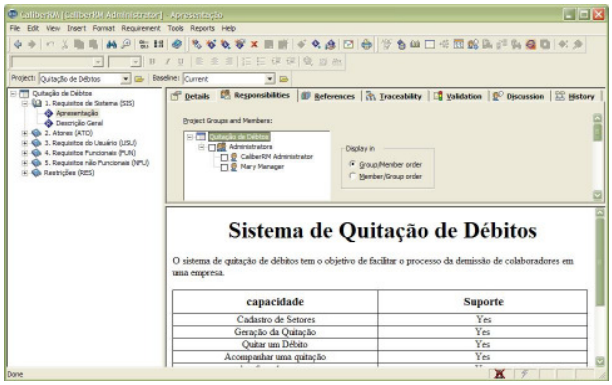

Figura 16 – Aba *Responsabilities:* definição dos responsáveis pelo requisito
Fonte: (CaliberRM, 2009)

Através da aba *Discussion* a ferramenta exibe o registro das mensagens postadas pelos responsáveis pelo requisito, facilitando assim a comunicação entre os membros da equipe responsável pelo projeto (ver Figura 17). Clicou-se no botão

Post New onde informou-se um título para a discussão e um texto para a mesma.

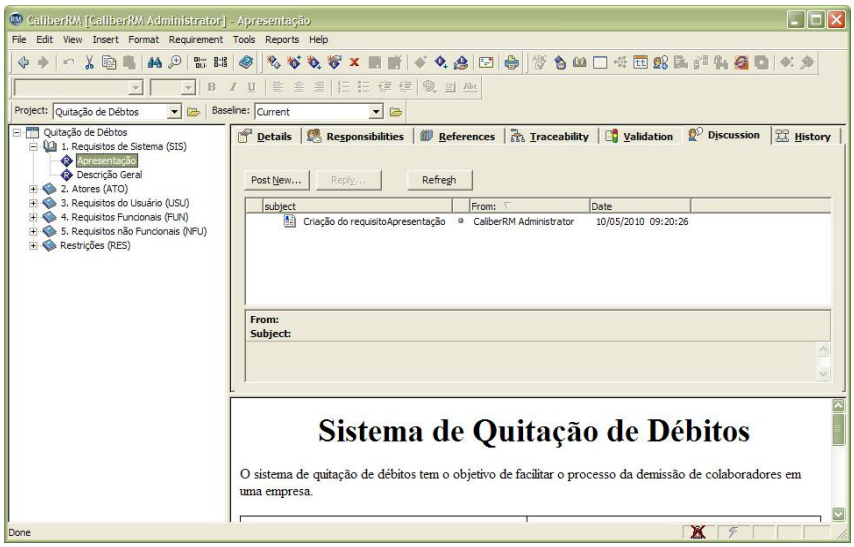

Figura 17 – Aba *Discussions:* visualização das discussões de um requisito
Fonte: (CaliberRM, 2009)

A ferramenta permitiu acesso ao histórico das versões do requisito, iniciando com sua criação e sendo atualizado a cada alteração sofrida pelo requisito, através da aba *History* (conforme Figura 18). As alterações foram definidas como versão do requisito, onde cada alteração recebeu um número. Esta forma de manter o histórico das alterações permite a visualização das mesmas por toda a equipe envolvida no projeto.

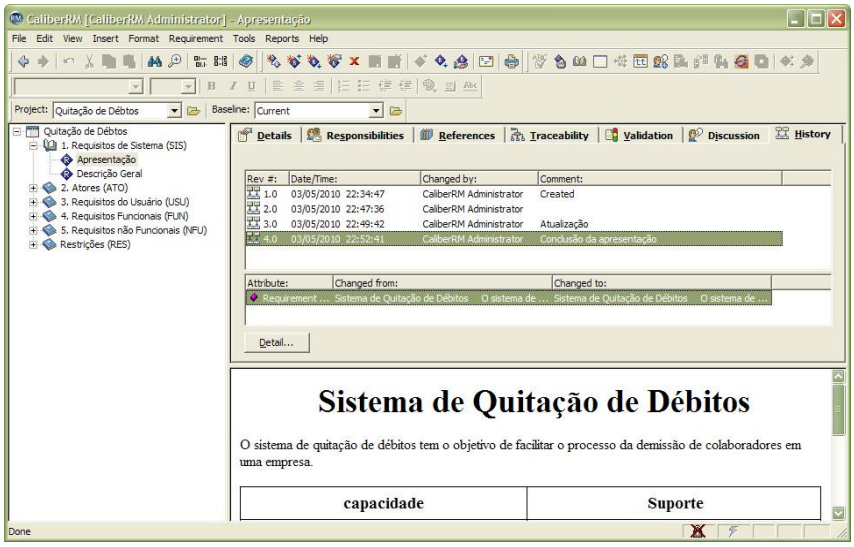

Figura 18 – Aba *History*: exibição do histórico do requisito
Fonte: (CaliberRM, 2009)

Para informar os atores do sistema, conforme apresentado na subsessão 3.1.3 (Modelos do Sistema / Casos de Uso), criou-se um requisito chamado atores onde foram informados os atores e suas ações no sistema. Os passos seguidos foram: criar um requisito denominado atores e em seguida informar cada um dos atores do sistema.

Aqui pode-se destacar uma grande flexibilidade da ferramenta que permite criar tipos de requisitos conforme a conveniência do engenheiro de requisitos (ver Figura 19).

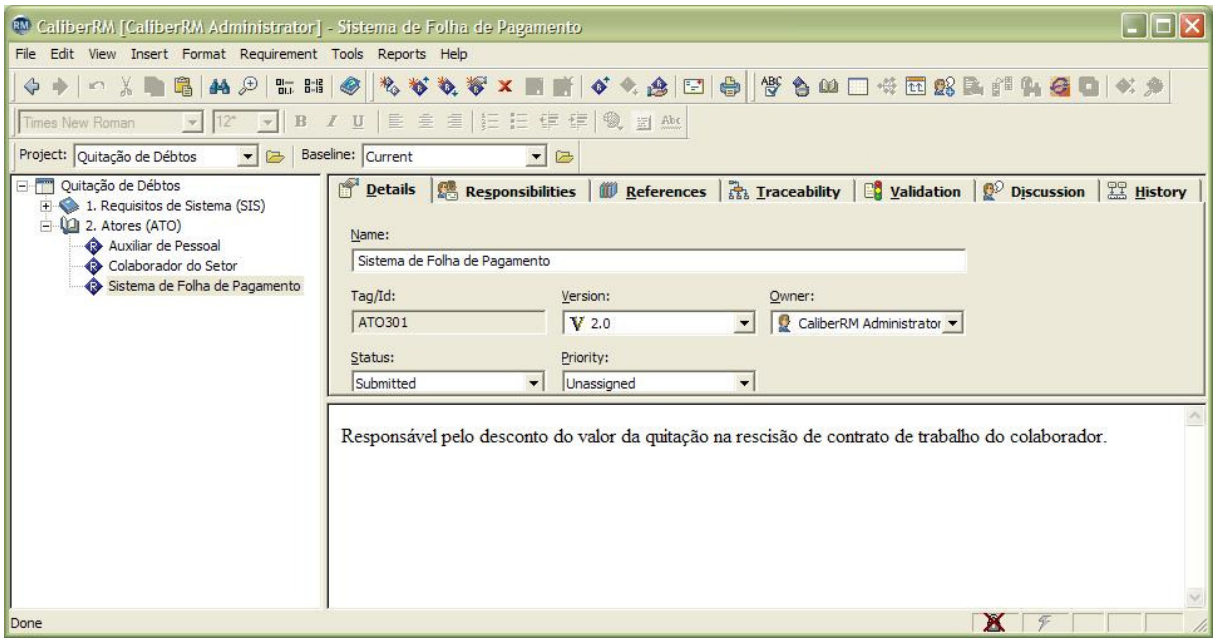

Figura 19 – tela da ferramenta após criação dos atores
Fonte: (CaliberRM, 2009)

O passo seguinte foi entrar com os requisitos do usuário (conforme descrito na subsessão 3.1.3 – Modelos do Sistema / Casos de Uso). A ferramenta permitiu entrar os dados gerais como no cadastro de qualquer requisito. Aqui é importante destacar a facilidade com que a ferramenta permite que sejam identificados os casos de uso. Pode-se observar isto através da aba *Use Case Data* (ver Figura 20).

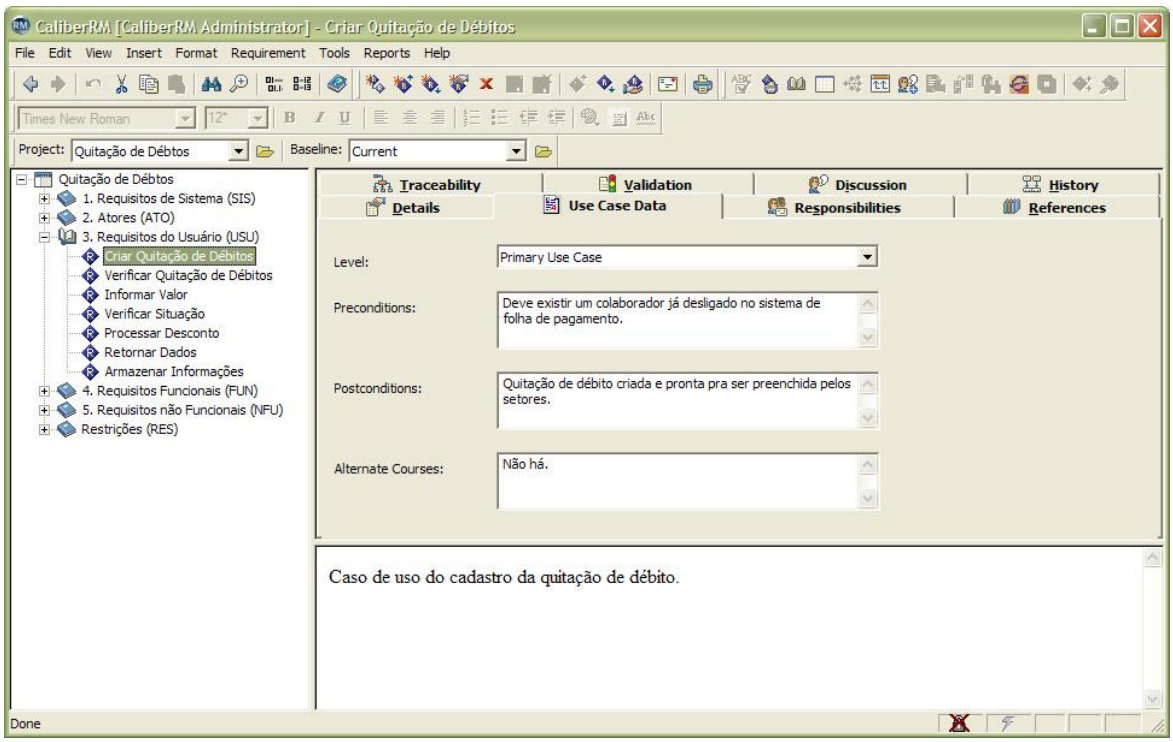

Figura 20 – *Use Case Data*: tela de entrada dos dados de um caso de uso
Fonte: (CaliberRM, 2009)

A seguir foram informados os requisitos funcionais conforme descrito na subsessão 3.1.2.1. Por ser uma repetição de ações dentro do sistema, somente foram informados os requisitos relacionados com o cadastro da quitação (ver Figura 21).

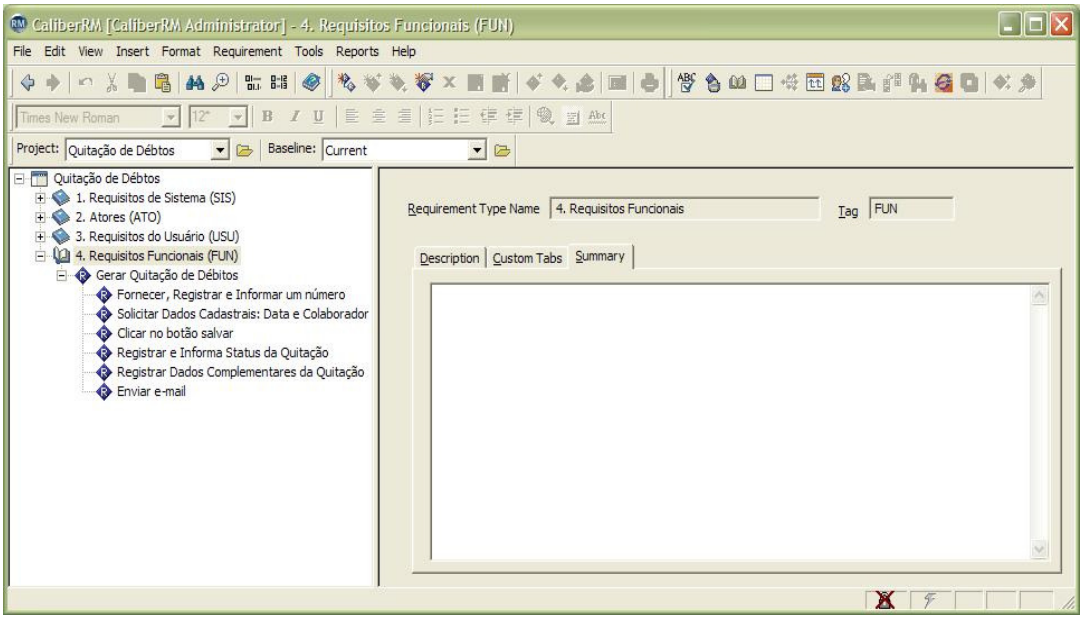

Figura 21 - Tela da ferramenta após criação dos requisitos funcionais
Fonte: (CaliberRM, 2009)

Faz-se importante destacar a facilidade com que a ferramenta permitiu informar as alterações nos requisitos identificadas através do cenário 2 (dois). Através da aba *Discussion* avisou-se a todos os participantes do projeto que o requisito sofreu alterações (ver figura 22) e acionando-se a aba *History* verificou-se quais foram as alterações sofridas (conforme figura 23).

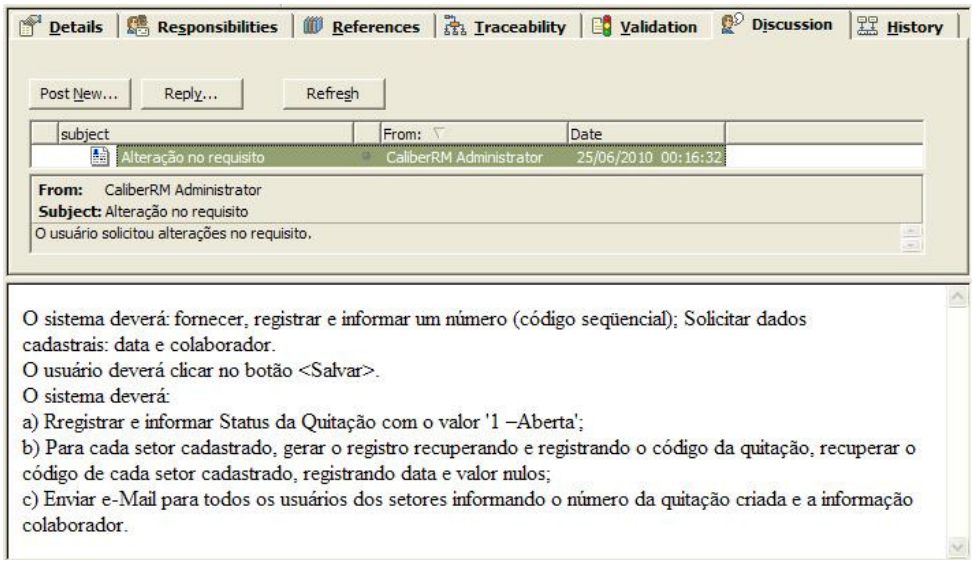

Figura 22 - Aba *Discussions* na tela de requisitos funcionais

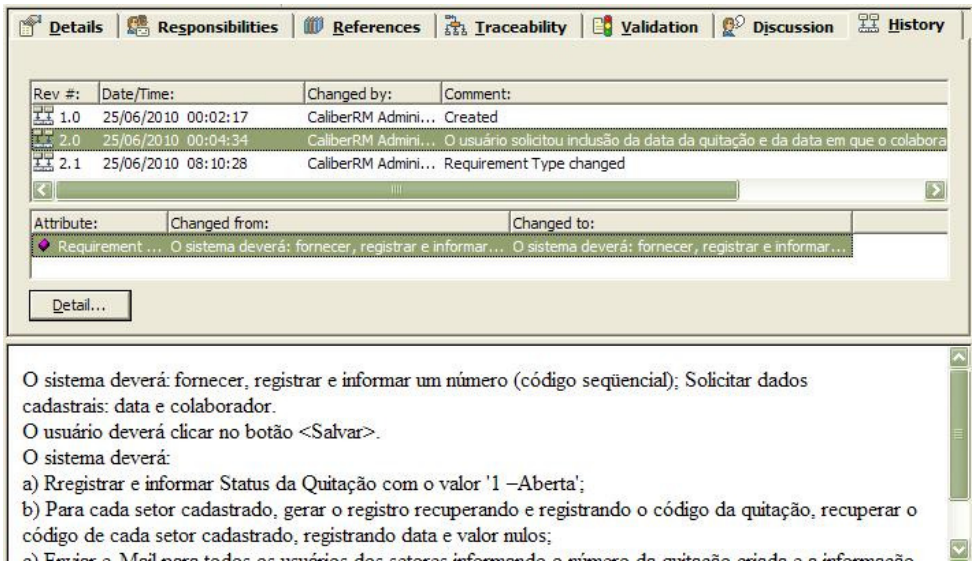

Figura 23 - Aba *History* na tela de requisitos funcionais

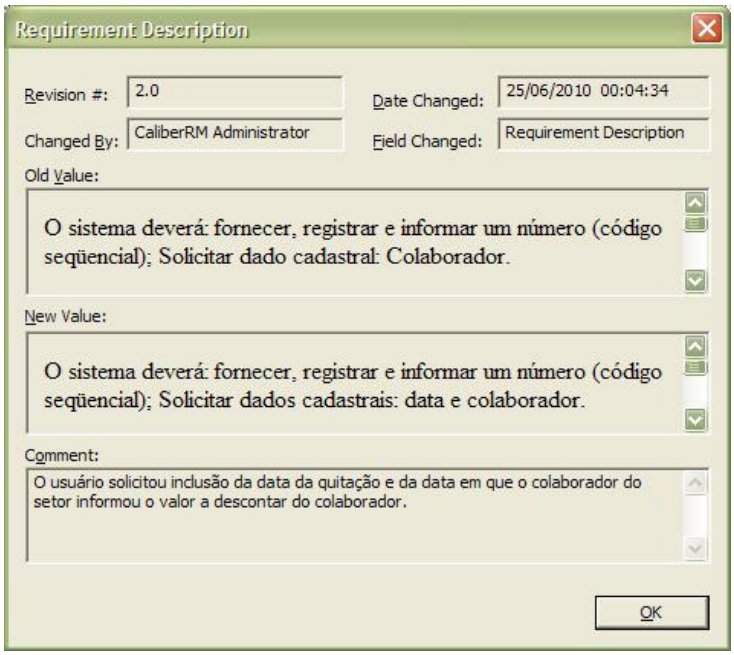

Figura 24 – tela com detalhes da alteração sofrida pelo requisito

O próximo passo foi informar os requisitos não funcionais conforme descritos na subsessão 3.1.2.2 (ver Figura 25). Observou-se também que para este tipo de requisito a ferramenta permitiu efetuar discussões assim como verificar o histórico das alterações sofridas pelo requisito.

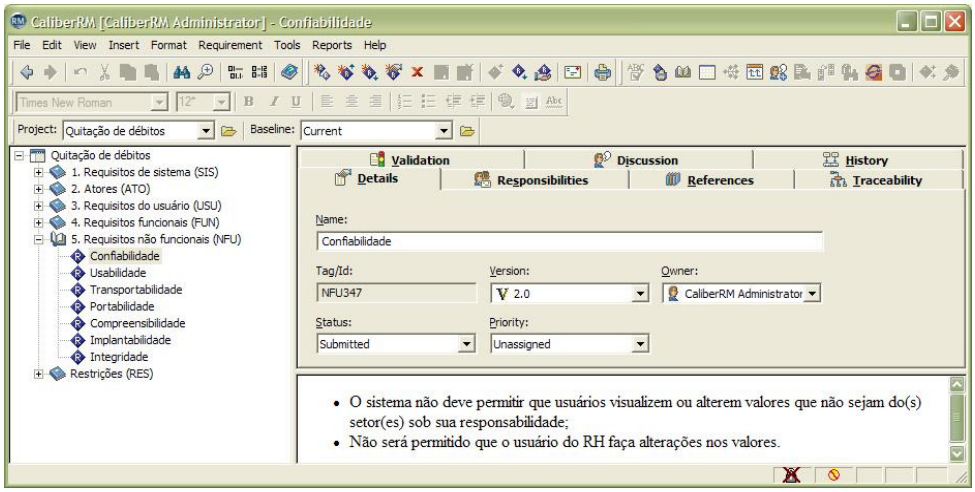

Figura 25 - Tela da ferramenta após criação dos requisitos não funcionais
Fonte: (CaliberRM, 2009)

Para concluir a entrada de requisitos, informou-se as restrições do sistema conforme descrito na subsessão 3.1.2.3 – Características que o sistema não deve exibir.

5 – Análise e Discussão dos Resultados

Apresenta-se nesta sessão uma análise da ferramenta sob dois aspectos: a facilidade no gerenciamento dos requisitos e a avaliação de alguns requisitos não funcionais da ferramenta.

Observou-se que a ferramenta CaliberRM permite uma administração precisa dos requisitos através do envolvimento de todos os membros do projeto, desde a criação dos requisitos até as alterações sofridas pelas quais os requisitos possam ter. Esta observação ressalta a importância de uma ferramenta no gerenciamento dos requisitos, que facilite a comunicação entre os diversos membros da equipe e consequentemente o acompanhamento das alterações nos requisitos do sistema em desenvolvimento.

Verificou-se que o uso de uma ferramenta de gerenciamento de requisitos durante o desenvolvimento de projetos de *software*, permite o compartilhamento dos requisitos entre todos os envolvidos no projeto, bem como discutir as alterações feitas por todos através do histórico de alterações sofridas pelos requisitos, atendendo às necessidades do processo da análise de requisitos, além de permitir uma gerência efetiva dos requisitos.

Efetuou-se a avaliação de alguns requisitos não funcionais da ferramenta CaliberRM realizada sobre os aspectos citados na metodologia, conforme se segue:

1) Funcionalidade: a ferramenta permitiu realizar-se com precisão: criação do projeto, cadastro de item (grupo de requisitos), cadastro detalhado das especificações do requisito, cadastro detalhado das informações dos casos de uso, cadastro dos requisitos funcionais e não funcionais.

   Foi possível definir responsáveis pelo requisito.

   Permitiu inserir e acompanhar discussões.

   Foi possível acompanhar o histórico das alterações nos requisitos. Utiliza números com casa decimal para representar a versão, que é incrementada quando o texto do requisito é alterado; se apenas atributos são modificados, apenas a parte decimal do número é incrementada.

2) Usabilidade: a ferramenta foi de fácil utilização, permitindo criar

intuitivamente os itens acima bem como alterá-los. A árvore exibida à esquerda da tela, facilitou o acesso aos itens e seus requisitos. A disposição das informações em abas facilitou o acesso a várias funcionalidades. A disposição dos menus e a presença da barra de ferramentas está conforme o padrão das aplicações para ambientes gráficos, permitindo assim o usuário encontrar facilmente opções como: abrir projeto, salvar projeto, ferramentas e ajuda.

3) Eficiência: a ferramenta “respondeu” de forma rápida as ações de escolha do usuário.

4) Confiabilidade: os dados são armazenados em banco de dados próprio da ferramenta e demonstrou-se confiável porque nenhuma informação cadastrada foi perdida.

5) Portabilidade: A ferramente não é portável, sendo disponível apenas para o sistema operacional Windows (Microsoft, 2010).

6) Compreensibilidade: a ferramenta disponibiliza ajuda em formato html (W3C, 2010) permitindo ao usuário encontrar textos que o ajudem a solucionar sua dúvida (textos em inglês).

Cabe destacar que a avaliação efetuada não teve o objetivo de realizar uma avaliação detalhada da ferramenta, mas sim identificar as principais funcionalidades oferecidas pela mesma no gerenciamento de requisitos.

Como pode ser observado, optou-se em dar prioridade às funcionalidades diretamente relacionadas à atividade de registro dos requisitos, além das funcionalidades básicas para a utilização da ferramenta. Contudo, a realização deste estudo, ofereceu subsídios para apoiar o grande diferencial desse trabalho, que é a utilização de uma ferramenta no gerenciamento de requisitos.

6 – Conclusão e Trabalhos Futuros

A ferramenta CaliberRM é especialmente preparada para atender às necessidades de rastreabilidade presentes em processos de gerência de requisitos, fazendo com que as equipes consigam cadastrar os requisitos, compreendê-los melhor, tomar conhecimento das alterações e analisar o seu impacto no projeto.

Realizou-se o estudo das técnicas da engenharia de requisitos para melhor compreender os recursos disponibilizados na ferramenta. A criação de um estudo de caso facilitou o uso da ferramenta, pois possibilitou o cadastro de requisitos, atores e casos de uso, além do registrar as alterações sofridas pelos requisitos. Todos esses passos foram importantes para a avaliação realizada.

O uso de uma ferramenta de gerenciamento de requisitos durante o desenvolvimento de projetos de *software*, possibilita compartilhamento automático dos requisitos com outros usuários, bem como discutir as alterações feitas por todos através do histórico de alterações sofridas pelos requisitos. O CaliberRM demonstra ser uma ferramenta poderosa e pronta para atender às necessidades do processo da análise de requisitos, permitindo uma gerência efetiva dos requisitos e auxiliando na garantia da qualidade.

Alcançou-se de forma satisfatória o objetivo proposto neste trabalho.

Sugere-se para trabalhos futuros:

1) Testar outras funcionalidades da ferramenta tais como: *export to Access, import from word (menu file); map/unmaped requirement, compare requirement (menu requirement*); *document factory, requirement grid, traceability diagram, traceability matrix, unread discussions, compare baselines, baseline signature, mapped diagram, estimation, teamdefine (menu tools)*. Alguns desses itens não estavam disponíveis na ferramenta pelo fato da versão utilizada ser de demonstração.
2) Analisar também as funcionalidades: suporte à segurança e controle de acesso concorrente e de acesso Web.
3) Fazer o comparativo entre a ferramenta CaliberRM com outras que se propõem a desenvolver as mesmas tarefas.

Referências

(Araújo, 2009) Araújo, Marco Antônio Pereira. **Análise de Requisitos com CaliberRM**. Clube Delphi, Rio de Janeiro, Edição nº 84.

(Blaschek, 2002) Blaschek, José Roberto. **Gerência de Requisitos: O principal problema dos projetos de Software**. Rio de Janeiro, Brasil, 4p.

(Bubenko et al., 2001) J.J., PERSON, A., STIRNA, J. (2001), **D3 Appendix B: EKD User Guide, Royal Institute of Technology (KTH) and Stockholm University, Stockholm, Sweden**. Disponível em <ftp://ftp.dsv.su.se/users/js/ekd_user_guide_2001.pdf>. Acessado em 21 de fevereiro de 2010.

(Caiado, 2009a) Caiado, Robinson. **CaliberRM: Automatizando a Gerência de Requisitos**. Clube Delphi, Rio de Janeiro, Edição nº 58.

(Caiado, 2009b) Caiado, Robinson. **CaliberRM: O poder da rastreabilidade**. Clube Delphi, Rio de Janeiro, Edição nº 59.

(CaliberRM, 2009) Borland CaliberRM, Borland Software Corp. Disponível em <http://www.borland.com/us/products/caliber/rm.html>. Acessado em 19 de setembro de 2009.

(Cardoso, 2003) Cardoso, Caíque – **UML na prática: do problema ao sistema**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2003.

(Chung, P. et al., 2000) Chung, L. Nixon, B. Yu, E. e MULOPOULOS, J., **Non-Functional Requirements in Software Engineering**. Kluwer Academic Publisher, 2000.

(Doors, 2010) Rational Doors, IBM. <http://www-01.ibm.com/software/awdtools/doors/> Acessado em 28 de abril de 2010.

(Filho, 2010) Filho, Clayton Vieira Fraga – **CONTROLA: Ferramenta de apoio ao processo de desenvolvimento de software em pequenas empresas**. Disponível em <http://www.unibratec.com.br/anaisdecongresso/diretorio/FDV-CVFF%20j%C3%A1%20revisado.pdf>. Acessado em 31 de março de 2010.

(Goguen, 1994) Goguen, J. **Requirements Engineering as the Reconciliation of Social and Technical Issues. Requirements Engineering: Social and Technical Issues**, Academic Press Inc., London, England, 1994.

(Grings, 2009) Grings, Claiton Luiz; Sayão, Miriam. **OpenReq: uma Ferramenta para Auxílio à Gerência de Requisitos**. Rio Grande do Sul, 2009. Disponível em <http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER09/grings.pdf>. Acessado em 4 de Outubro de 2009.

(Kotonya et al., 1997) Kotonya, G., Sommerville, I.. **Requirements Engineering: Processes and Techniques**, John Wiley & Sons, 1997.

(Leite, 2001) Leite, Julio C. S. P. - **Gerenciando a Qualidade de Software com Base em Requisitos**. In: A. R. C. Rocha, J. C. Maldonado e K. C. Weber. Qualidade de Software Teoria e Prática. São Paulo: Prentice - Hall, 2001, pp. 238-246.

(Loucopoulos, P. et al., 1998). **Using the EKD approach: The modeling component**. Manchester, UMIST.

(Microsoft, 2010) Microsoft Corporation. <http://www.microsoft.com/en/us/default.aspx>. Acessado em 29 de abril de 2010.

(Peters, 2001) Peters, James F; Pedrycz, Witold. **Engenharia de Software.** Rio de Janeiro: Campus, 2001.

(Pressman, 2006) Pressman. Roger S. **Engenharia de Software**. 6. ed. São Paulo: McGraw-Hill, 2006.

(Quadros, 2002) Quadros, Moacir – **Gerência de Projetos de Software – Técnicas e Ferramentas**. Florianópolis: Bookstore Livraria Ltda., 2002.

(RequisitePro, 2010) Rational RequisitePro, IBM. <http://www-01.ibm.com/software/rational/?pgel=ibmhzn&cm_re=masthead-_-products-_-sw-rational>. Acessado em 28 de abril de 2010.

(Sayão et al., 2003) Sayão, M., A. V. Staa e J. C. S. P. Leite, **"Qualidade em Requisitos"**, Monografia em Ciência da Computação, DI/PUC-Rio, Rio de Janeiro, 2003.

(Sommerville, 2007) Sommerville, Ian – **Engenharia de Software, 8ª edição**. São Paulo: Pearson Addison-Wesley, 2007.

(Spínola, 2009) Spínola, Rodrigo Oliveira. **Introdução à Engenharia de Requisitos**. Engenharia de Software Magazine, Rio de Janeiro, Edição Especial.

(Wazlawick, 2004) Wazlawick, Raul Sidnei. **Análise e projeto de sistemas de informação orientada a objetos**. Rio de janeiro: Elsevier, 2004.

(Wiegers, 2003) Wiegers, Karl E., **Software Requirements**, Microsoft Press, Redmond, 2003.

(W3C, 2010) World Wide Web Consortium Escritório Brasi – **Guia de Referência HTML**. Disponível em <http://www.w3c.br/divulgacao/guiasreferencia/xhtml1/>. Acessado em 29 de abril de 2010.

(Zanlorenci, 2010) Zanlorenci, Edna Pacheco e Burnett, Robert Carlise, **Engenharia de requisitos: processos e técnicas no contexto organizacional**. PUC/PR, 2010. Disponível em <http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=601>. Acessado em 30 de março de 2010.