# PRIORIZAÇÃO DE REQUISITOS E AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A PERCEPÇÃO DOS USUÁRIOS

**ALINE GOMES CORDEIRO**

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CAMPOS DOS GOYTACAZES - RJ

ABRIL – 2010

# PRIORIZAÇÃO DE REQUISITOS E AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A PERCEPÇÃO DOS USUÁRIOS

**ALINE GOMES CORDEIRO**

Dissertação apresentada ao Centro de Ciências e Tecnologia da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como parte das exigências para obtenção do título de Mestre em Engenharia de Produção.

Orientador: André Luís Policani Freitas, D. Sc.

CAMPOS DOS GOYTACAZES - RJ

ABRIL – 2010

# PRIORIZAÇÃO DE REQUISITOS E AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A PERCEPÇÃO DOS USUÁRIOS

**ALINE GOMES CORDEIRO**

Dissertação apresentada ao Centro de Ciências e Tecnologia da Universidade Estadual do Norte Fluminense Darcy Ribeiro, como parte das exigências para obtenção do título de Mestre em Engenharia de Produção.

Aprovada em 09 de abril de 2010

Comissão Examinadora:

_____

Prof.: Manuel Antonio Molina Palma, D. Sc. – UENF

_____

Prof.: Rogério Atem de Carvalho, D. Sc. – IFF

_____

Prof.: Aline Pires Vieira de Vasconcelos, D. Sc. – IFF

_____

Prof: André Luís Policani Freitas, D. Sc. – UENF

Orientador

## AGRADECIMENTOS

A Deus, por permitir que esse momento aconteça.

Ao meu orientador, o professor André Luís Policani Freitas, por me mostrar o melhor caminho em cada etapa do mestrado e por estar tão presente nesses últimos anos da minha vida.

Aos meus pais e às minhas irmãs por compreenderem os momentos de ausência que foram necessários. Em especial, à minha mãe por acreditar que sempre posso ir além e por se sentir realizada com as minhas vitórias.

A Adelson por ser mais que um companheiro e por sempre me incentivar.

Aos meus amigos do HGG que acompanharam o início de minha trajetória no mestrado.

Aos meus amigos da FJBM que estiveram comigo nos momentos finais do mestrado e compreenderam a importância desse meu objetivo. Em especial, à Direção da FJBM que permitiu que a pesquisa fosse realizada e a Petrúcio Pessanha de Oliveira que contribuiu diretamente através de sua participação na análise dos resultados.

Aos professores do Leprod que, entre outras contribuições, me ajudaram a amadurecer academicamente.

Aos funcionários do Leprod que sempre se dispuseram a me ajudar no que foi preciso.

## RESUMO

### PRIORIZAÇÃO DE REQUISITOS E AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A PERCEPÇÃO DOS USUÁRIOS

Aline Gomes Cordeiro

Atualmente o software tem sido reconhecido como uma importante ferramenta de apoio às diversas atividades e à tomada de decisões. No entanto a atividade de desenvolvimento de software tem enfrentado algumas dificuldades, existem diversos relatos a respeito de projetos de desenvolvimento de software fracassados. Nesse cenário, a fase de elicitação de requisitos, aliada à satisfação dos usuários, tem sido destacada como preponderante para a melhoria do processo de desenvolvimento de software. Neste sentido a questão problema apresentada por esta dissertação é a seguinte: Como é possível realizar a avaliação da qualidade do produto de software desde as etapas iniciais do projeto, de forma que seja possível realizar as melhorias com menor esforço? Buscando responder a esta questão, esta dissertação propõe uma metodologia que através da execução de algumas etapas, possibilita a priorização dos requisitos de software e a avaliação da qualidade do produto de software, segundo a percepção dos usuários. Em especial, a metodologia propõe para a etapa de priorização o emprego da Análise Importância-Desempenho (IPA) e método dos 100 pontos e para a etapa de avaliação de desempenho, o emprego da IPA e da escala contínua. Por meio de um estudo de caso, a metodologia foi aplicada a um projeto de desenvolvimento de um software para gestão de recursos humanos. A partir desta aplicação, os desenvolvedores conseguiram, no início do desenvolvimento, ter como referência uma lista de requisitos priorizados para direcionar as atividades do desenvolvimento. Além disso, ao final do processo, puderam identificar os requisitos mais críticos para a melhoria do software de acordo com a percepção dos usuários.



Palavras-chave: *Priorização de requisitos; Qualidade de software; Produto de software.*

***ABSTRACT***

***REQUIREMENTS PRIORITIZATION AND SOFTWARE QUALITY EVALUATION ACCORDING TO USERS' PERCEPTION***

Aline Gomes Cordeiro

*Nowadays software has been recognized as an important tool to support a diverse set of activities and decision making. However software development has faced some difficulties, there are several reports about failed software development projects. In this scenario, the step of requirements elicitation, together with users' satisfaction, has been highlighted as preponderant to the software development process improvement. In this sense, the problematic presented by this work is: How is it possible to perform quality evaluation of software product since project initial stages, in a way that it becomes possible to make improvements with less effort? In order to answer this question, this work proposes an approach that following some steps, allows software requirements prioritization and quality evaluation of software product, according to users' perception. Accordingly to the proposed approach, Importance-Performance Analysis (IPA) and 100 points method are used for requirements prioritization purposes (prioritization step), and the continuous scale and IPA are used for evaluating the software performance (performance evaluation step). By conducting a case study, the proposed approach was applied to a software development project for human resource management. By doing so, the developers were able to get a list of prioritized requirements as reference to guide development activities, at the beginning of development. In addition, in the end of the process, they were able to identify the most critical requirements for software improving according to users' perception.*

*Keywords: Requirements Prioritization, Software Quality, Software Product.*

**SUMÁRIO**

## LISTA DE SIGLAS

| | |
|---|---|
| **ABNT** | Associação Brasileira de Normas Técnicas |
| **AHP** | *Analytic Hierarchy Process* (Método de Análise Hierárquica) |
| **DP** | Departamento de Pessoal |
| **ERP** | *Enterprise Resources Planning* |
| **IPA** | *Importance-Performance Analysis* (Análise importância-desempenho) |
| **ISO** | *International Organisation for Standardisation* |
| **RH** | Recursos Humanos |
| **SI** | Sistema de Informação |
| **SOFTEX** | Associação para Promoção da Excelência do Software Brasileiro |
| **SQuaRE** | Requisitos e Avaliação da Qualidade de Produto de Software |
| **SUS** | Sistema Único de Saúde |
| **TDD** | *Test-Driven Development* (Desenvolvimento Guiado por testes) |
| **TI** | Tecnologia da Informação |
| **UML** | *Unified Modeling Language* |

## LISTA DE QUADROS

## LISTA DE FIGURAS

## LISTA DE TABELAS

# CAPÍTULO I

# INTRODUÇÃO

Atualmente, nota-se que o software tem sido utilizado nas organizações como instrumento de apoio às diversas atividades e à tomada de decisões. É possível observar a presença de software como ferramenta em diferentes ramos de negócios como saúde, educação e indústrias. As melhorias obtidas a partir de sua utilização podem ser notadas e geralmente justificam os investimentos necessários.

## 1.1 O Problema de pesquisa

Apesar da importância do software para as organizações, um dos grandes desafios refere-se ao fato de que em alguns casos, o investimento feito em sistemas informatizados não fornece o retorno esperado pelos investidores, ou seja, os sistemas não se adequam à realidade das empresas onde são implantados. Os processos de negócio nesses casos não são condizentes com os processos definidos pelo sistema informatizado.

Ao longo do tempo diversos casos de projetos de desenvolvimento de software que fracassaram têm sido descritos. Os casos de fracasso geralmente estão associados a falhas de segurança, custos que excedem as estimativas e atrasos nas entregas (LINBERG, 1999).

Karlsson e Ryan (1996) fornecem uma visão complementar ao afirmar que um dos maiores riscos enfrentado por organizações que desenvolvem software comercial está associado ao não atendimento das necessidades e expectativas dos usuários. Para esses autores, esse risco pode ocasionar danos na reputação, perda de pedidos e redução dos lucros da empresa.

Pesquisas que visam à identificação das causas para o problema acima citado apontam a fase de elicitação de requisitos, também conhecida como levantamento de requisitos, como preponderante para a melhoria do processo de desenvolvimento de software.

A elicitação de requisitos é uma das atividades que ocorre no início do desenvolvimento de software. O objetivo principal da engenharia de requisitos é permitir o desenvolvimento de sistemas de software que satisfaçam a todos os *stakeholders* (Karlsson *et al*., 1997).

Erros gerados nesta atividade, se não são corrigidos, se estendem até o final do desenvolvimento e após a verificação de cada erro, todas as fases anteriores precisam ser refeitas. Para Sadraei *et al*. (2007) os requisitos de software são determinantes críticos da qualidade de software.

Neste sentido, a avaliação da qualidade de software vem sendo identificada como uma possibilidade de minimização do problema acima descrito.

A qualidade de software, como tratada atualmente na literatura científica, preocupa-se com a qualidade do processo de desenvolvimento ou com o produto final gerado desse processo. O foco de estudo desta dissertação está relacionado ao produto. Este tipo de abordagem, de uma forma geral, busca analisar a qualidade do produto gerado do processo de desenvolvimento de software.

A abordagem baseada no produto tem contribuído para elevar a qualidade de software. No entanto, esta abordagem possui algumas limitações, a principal delas está relacionada ao fato da avaliação da qualidade ser realizada apenas quando o produto de software já está implementado. Após a avaliação são identificadas correções e melhorias, porém nesta etapa do projeto o esforço necessário para alterar características do software é grande.

Desta forma caracteriza-se o problema de pesquisa tratado por este estudo: como é possível realizar a avaliação da qualidade do produto de software desde as etapas iniciais do projeto, de forma que seja possível realizar as melhorias com menor esforço?

## 1.2 Objetivo

Esta dissertação visa propor uma metodologia para avaliar a qualidade de software desde a elicitação de requisitos. A metodologia proposta deve permitir a priorização dos requisitos e a avaliação de desempenho do produto de software, segundo a percepção dos usuários. Os usuários nesse contexto podem ser

compreendidos como as pessoas que de alguma forma fazem ou farão uso do software. Os requisitos priorizados relacionados ao resultado de desempenho devem fornecer diretrizes para melhorias na qualidade do software.

Como objetivos secundários desta dissertação é possível destacar:

- Direcionar as atividades do processo de desenvolvimento para as funções mais relevantes, por meio da priorização dos requisitos;
- Realizar a avaliação de desempenho dos produtos de software tendo como critérios os requisitos;
- Identificar uma forma de sinalizar para os desenvolvedores as funções do software que precisam ser melhoradas com o objetivo de elevar a qualidade.

## 1.3 Justificativa

Alguns trabalhos buscam a qualidade de software por meio da melhoria do processo de desenvolvimento identificando melhorias possíveis em cada etapa do processo. Gomes *et al*. (2001), em seu trabalho têm como foco a melhoria do processo de desenvolvimento de software através da seleção adequada de métricas.

Jung (2007) verifica a relação entre as características de qualidade externa e a satisfação dos usuários. O estudo foi realizado com usuários de um pacote de software já desenvolvido e em uso.

As duas referências anteriores retratam as abordagens que são usualmente encontradas na literatura: as avaliações de qualidade têm como objetivo o processo, definindo atividades a serem executadas em cada etapa ou têm como objetivo o produto, identificando melhorias possíveis ao final do processo de desenvolvimento.

O diferencial desta dissertação, que justifica a sua elaboração, refere-se à busca pela melhoria da qualidade final do produto com base na percepção dos usuários do software desde o início do desenvolvimento.

Acredita-se que a metodologia proposta pode permitir dois benefícios. O primeiro é aproximar o processo de desenvolvimento das necessidades dos clientes, elevando a probabilidade de alcançar a satisfação dos usuários. E o segundo consiste em reduzir a necessidade de retrabalho resultante da elicitação de requisitos inadequada.

Além disso, um estudo realizado por Cordeiro e Freitas (2008) mostra que, de acordo com os dados utilizados na pesquisa, existe uma lacuna relativa à aplicação de metodologias que visam à avaliação da qualidade de software. Neste estudo foi constatado que, embora novas metodologias sejam criadas e publicadas em diversos periódicos como exposto por Xenos (2001); Belgamo *et al.* (2005) e Herrmann e Paech, (2008), profissionais da área de Tecnologia da Informação (TI) não conhecem questões básicas relativas ao assunto.

A lacuna identificada justifica a necessidade de desenvolvimento de uma metodologia que possa ser aplicada sem a necessidade de inclusão de muitos passos adicionais no processo de desenvolvimento e que possibilite a busca pela qualidade de software por empresas e profissionais da área de desenvolvimento.

## 1.4 Organização

Esta dissertação está estruturada em 5 capítulos. O capítulo I introduz o tema da dissertação, apresentando uma descrição do problema tratado, os objetivos pretendidos e a justificativa para a sua elaboração.

O capítulo II fornece um referencial teórico a respeito dos aspectos centrais tratados na dissertação. Neste capítulo alguns conceitos são destacados com o objetivo de facilitar o entendimento a respeito de como metodologia está estruturada.

O capítulo III descreve a metodologia proposta, detalhando as etapas necessárias para a priorização e avaliação da qualidade de software segundo a percepção dos usuários.

O capítulo IV apresenta o estudo de caso realizado tendo como base a metodologia proposta. Neste capítulo a execução de cada etapa é descrita em detalhes, permitindo que se compreenda como o estudo de caso foi realizado e os resultados obtidos a partir de sua execução.

O capítulo V refere-se enfim as considerações finais que podem ser feitas a partir do estudo realizado. Este capítulo também apresenta as limitações identificadas e as sugestões para trabalhos futuros.

Esta dissertação é composta ainda por dois Apêndices e um Anexo. Os Apêndices A e B contêm os questionários utilizados nas etapas de priorização e avaliação de desempenho, respectivamente. Enquanto o Anexo A fornece uma descrição a respeito do software desenvolvido através do projeto onde o estudo de caso foi realizado.

# CAPÍTULO II

# REFERENCIAL TEÓRICO

Neste capítulo são apresentados alguns aspectos teóricos relativos ao tema tratado por esta dissertação.

## 2.1 O Software e suas características

O software é usualmente conceituado comparativamente ao hardware. No entanto, por volta da década de 60 o software deixou de ser visto como um complemento do hardware e passou a ser considerado um produto independente, apesar de necessitar do hardware para funcionar. Os processos produtivos do hardware e do software passaram a ser identificados de formas distintas. E apesar da proximidade existente entre hardware e software, este último possui características bastante específicas que são apresentadas nessa seção.

Sommerville (2007) torna claras algumas especificidades relativas ao software ao afirmar que o software não é limitado por materiais, controlado por leis da física ou por processos de manufatura. Para o autor, a falta dessas restrições torna o software complexo e difícil de ser compreendido.

O software é caracterizado pela sua natureza não-material e sua função de produção não envolve o emprego de matérias-primas consumíveis ao longo de seu ciclo produtivo (ROSELINO, 2006).

Pressman (2002) apresenta algumas características do software estabelecendo um paralelo entre hardware e software. O autor afirma que como o software é um elemento de um sistema lógico e não físico possui características diferentes das encontradas no hardware, dentre as quais se destacam:

i. o software é desenvolvido mas não é manufaturado, fazendo com que os custos estejam concentrados na fase de engenharia;

ii. o software não se desgasta com o passar do tempo, as falhas que podem ocorrer têm origem no projeto ou processo de desenvolvimento, tornando a manutenção do software mais complexa do que a do hardware;

iii. na produção do hardware parte da produção está baseada na montagem de componentes, enquanto no desenvolvimento de software a reutilização de componentes ainda não está amplamente difundida e soluções específicas precisam ser encontradas.

As características apresentadas deixam evidente que o software possui características bastante específicas que não são normalmente encontradas em outros tipos de bens.

Os autores acima citados destacam características relativas ao processo de produção do software. Associada a este aspecto situa-se uma das dificuldades relativas à classificação do software. Alguns autores (PRESSMAN, 2002; SOMMERVILLE, 2007) percebem o software como um produto. No entanto, segundo Roselino (2006), o software é usualmente classificado como um serviço, devido à intangibilidade característica desta atividade e ao emprego direto de força de trabalho.

No contexto desta dissertação, o software é tratado como um produto devido ao foco na percepção do usuário, ou seja, mesmo participando do processo de produção, no início do desenvolvimento o que o usuário espera é um produto que o auxilie na execução de suas atividades. Nas etapas posteriores do desenvolvimento, embora possa continuar sendo adaptado, para o usuário, o software é um produto que ele recebe para ser utilizado.

Uma outra justificativa para a classificação adotada por esta dissertação refere-se à característica “simultaneidade” apresentada pelos serviços. Segundo Parasuraman *et al.* (1988), os serviços são consumidos quase no mesmo momento em que são produzidos. Essa característica não é observada no software.

Além da classificação acima, o software pode ainda ser classificado quanto a sua forma de chegada ao mercado e quanto ao tipo de mercado de destino. O quadro 2.1 apresenta de forma resumida essa duas classificações.

| CLASSIFICAÇÃO DO SOFTWARE | | | |
|---|---|---|---|
| ***Quanto a forma de chegada ao mercado*** | | ***Quanto ao tipo de mercado de destino*** | |
| Pacote | Vendido pronto e geralmente em larga escala. A produção não é caracterizada pela interação entre desenvolvedores e usuários. | Segmento Vertical | O software é desenvolvido para determinada atividade econômica. |
| Por encomenda | O software é proveniente de um projeto solicitado pelos usuários. Geralmente a venda ocorre antes do produto estar pronto e os custos são mais elevados. | Segmento Horizontal | O conteúdo é proveniente da área de informática com pouco conteúdo específico de outras áreas. É vendido na forma de pacotes e ajuda a resolver problemas de diversas áreas. |
| Embarcado | O software chega ao mercado embutido em algum equipamento. | | |

Quadro 2.1- Classificação do Software

Fonte: Adaptado de Freire (2002).

As soluções em software podem ser usadas para satisfazer as necessidades dos indivíduos diretamente, como bens finais de consumo ou indiretamente, indiretamente, como “meio de produção” (ROSELINO, 2006). Quando utilizado indiretamente, o software funciona como uma ferramenta de apoio a outras atividades.

Neste sentido o software é também reconhecido como importante insumo tecnológico que ocupa papel central na etapa de desenvolvimento das forças produtivas capitalistas (ROSELINO, 2006).

Apesar de não haver consenso entre as classificações possíveis para o software, os tipos de classificação descritos nesta seção devem servir de base para o entendimento de algumas questões relativas a esta dissertação.

## 2.2 O usuário

No contexto desta dissertação o usuário constitui-se como um aspecto importante, visto que a avaliação de qualidade tem como foco a percepção dos usuários.

Para Tor (2009) para que um Sistema de Informação (SI) seja considerado de sucesso é necessário verificar a satisfação dos usuários em relação ao sistema. O autor argumenta que os usuários são as pessoas que conhecem como o sistema funciona.

Berander (2004) afirma que é necessário distinguir os clientes dos usuários finais. O cliente pode ser, por exemplo, a pessoa que solicitou o desenvolvimento do software. No entanto, pode ocorrer desse cliente não utilizar o software de fato e sim outras pessoas que estão sob sua gerência. Neste caso, o cliente não é usuário.

Por isso é importante destacar que no contexto desta dissertação, os usuários podem ser compreendidos como as pessoas que de alguma forma fazem ou farão uso do software. A percepção desses usuários é o foco para avaliação do software e priorização dos requisitos.

Acredita-se que os usuários são os maiores conhecedores do domínio de aplicação do software, ou seja são eles que conhecem os detalhes dos problemas presentes na área de aplicação do software. Além disso, quando o software está em uso são os usuários que acessam com maior freqüência as suas funções, portanto eles são considerados pessoas aptas a realizar as avaliações.

A opinião destes usuários pode ser determinante para a continuidade do projeto de desenvolvimento, o que justifica o foco desta dissertação na percepção do usuário.

## 2.3 Qualidade de Software

Considerando a grande importância que o software assume no dia-a-dia das organizações e a crescente busca dessas organizações por produtos de software de qualidade, faz-se necessário neste momento definir algumas questões relacionadas à qualidade e à qualidade de software especificamente.

A qualidade em produtos e serviços pode ser entendida como a combinação de características de produtos e serviços referentes a marketing, engenharia, produção e manutenção, através das quais produtos e serviços em uso corresponderão às expectativas do cliente (FEIGENBAUM, 1994).

Para Campos (1994), um produto ou serviço de qualidade é aquele que atende perfeitamente, de forma confiável, de forma acessível, de forma segura e no tempo certo as necessidades do cliente.

Observa-se nos conceitos citados a preocupação com o cliente. Portanto, para que se possa avaliar a qualidade de um produto, é essencial levar em consideração a opinião dos clientes em relação ao mesmo.

Após a análise de alguns conceitos de qualidade, é possível descrever os conceitos mais diretamente relacionados à qualidade de software.

É comum encontrar definições de qualidade de software que apontam que este conceito está relacionado ao fato do software atender ou não aos requisitos, ou seja, às necessidades e expectativas dos usuários. No entanto, essas necessidades variam de usuário para usuário, o que torna ainda mais complexa a obtenção de uma definição.

Segundo o IEEE (1990), qualidade pode ser entendida como o grau no qual um sistema, componente ou processo satisfaz os requisitos especificados e as necessidades e expectativas do cliente ou usuário.

Denning (1992) trata a qualidade como uma característica que pode ser construída dentro de um sistema, seguindo certas regras e procedimentos. No entanto, para o autor, a satisfação dos usuários é uma referência para a qualidade do software. Os usuários não se preocupam com detalhes estruturais do software, mas sim com o quanto o software facilita o trabalho.

Prado (1999) lista algumas características necessárias para a qualidade de software, como a necessidade de manuais do usuário, por exemplo. Mas segundo o autor, a principal característica é que o produto deve agradar ao cliente.

Xenos (2001) destaca a usabilidade como característica principal para determinação da qualidade de software. Segundo ele, os principais modelos de garantia de qualidade de software qualificam o atributo usabilidade como essencial.

O processo de desenvolvimento também tem recebido destaque na definição da qualidade de software. Segundo Gomes *et al*. (2001), a abordagem atual da garantia da qualidade possui como foco principal o processo de produção do software, visto que este tem se tornado fator determinante para o alcance da qualidade do produto final. Neste sentido, medições de software têm desempenhado um importante papel para o aumento da qualidade do processo.

Pressman (2002) destaca a necessidade de conformidade aos requisitos funcionais, aos padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido. O autor ressalta a importância dos requisitos ao afirmar que os requisitos de software são a fundação a partir da qual a qualidade é medida. A falta de conformidade com os requisitos é falta de qualidade.

Além dos conceitos acima, uma referência importante é a do *Swebok Guide* (2004), que afirma que existe uma relação entre o valor e o custo da qualidade que deve ser observada. Um projeto de desenvolvimento de software objetiva criar um software que tenha valor e este valor pode ou não ser quantificado como custo. Por outro lado, o cliente tem um custo máximo em mente e espera que com esse custo as características básicas que espera do software sejam atendidas. Nem sempre a relação entre custo e benefício está clara. Essas características já mostram parte das complexidades relativas ao assunto.

Belgamo *et al*. (2005) destacam a qualidade dos produtos intermediários do processo de produção do software como essencial para que a qualidade do produto final possa ser alcançada. Os autores ainda ressaltam a importância do documento de requisitos para que ao final do processo ou de uma iteração do mesmo, o produto gerado atenda aos requisitos do usuário.

Para Magalhães (2006), a qualidade de software não pode ser confundida com a atividade de testes. Para que um software seja considerado de qualidade é necessário que esteja em conformidade com os seus requisitos, atenda aos requisitos e expectativas do cliente e seja bem aceito por seus usuários.

As referências acima tornam evidente a importância do papel do usuário para a obtenção da qualidade. De uma forma geral, para um produto ser considerado de qualidade, as necessidades dos usuários em relação ao software devem ser atendidas.

Após a definição a respeito da qualidade de software, é necessário ressaltar a distinção entre a qualidade voltada para o produto e a qualidade voltada para o processo de desenvolvimento.

A qualidade de software voltada para o produto refere-se, tradicionalmente, à avaliação do software após o seu desenvolvimento. O quadro 2.2 apresenta de forma sucinta alguns exemplos de trabalhos que realizam avaliações de qualidade tendo como referência produtos de software.

| | |
|---|---|
| Xenos e Christodoulakis (1995) | Os autores buscam através de questionários obter a opinião de usuários a respeito de software com o objetivo de mensurar a qualidade do produto de software. A avaliação é feita considerando as características externas do produto, ou seja, as que são diretamente percebidas pelos usuários. |
| Oliveira e Belchior (2002) | Os autores apresentam uma ferramenta *Fuzzy* (*AdeQuaS)*, para apoiar o processo de avaliação da qualidade de software, para isso se baseiam no modelo de Avaliação de Qualidade de Software. A avaliação é feita considerando o julgamento de um grupo de especialistas. |
| Silva (2003) | A autora ressalta a usabilidade como fundamental para a satisfação dos usuários e propõe uma metodologia baseada nas normas ISO/IEC 9126-1 e ISO/IEC 14598. O objetivo da metodologia proposta é avaliar o grau de satisfação dos usuários com relação a produtos de software, tendo como foco a usabilidade. |
| Punter *et al*. (2004) | Os autores propõem um novo processo para avaliação de produtos de software baseado na ISO/IEC 14598. Após a apresentação do processo, os autores descrevem três estudos de caso nos quais a avaliação foi feita por meio do processo proposto. Os resultados para as avaliações são apresentados. |
| Jung (2007) | O autor realiza uma pesquisa com usuários para verificar se a alta qualidade externa de produtos de software implica em alta qualidade em uso, ou seja, na satisfação dos usuários. |
| Boente e Moré (2009) | Um modelo *fuzzy* é proposto para avaliar a qualidade de produtos de software de acordo com a percepção dos gerentes de projetos de desenvolvimento. Os autores buscam identificar o grau de satisfação dos gerentes em relação a 18 critérios. |
| Celestino e Abe (2009) | Os autores utilizam a lógica paraconsistente anotada para a avaliação da qualidade de software, especificamente para a avaliação de um ERP. Um questionário foi elaborado tendo com referência os critérios definidos pela ISO 9126-1 para avaliar a satisfação dos usuários em relação ao ERP. |
| Larsen (2009) | O autor avalia a satisfação dos usuários finais em relação a um ERP (*Enterprise Resources Planning*), neste caso, o autor trata da avaliação de um produto de software. |

Quadro 2.2 – Exemplos de avaliações de qualidade de software baseadas no produto.

A qualidade de software voltada para o processo vislumbra a qualidade do produto final por meio do alcance da qualidade nas etapas intermediárias do processo de desenvolvimento. Em geral, metodologias que visam à qualidade do processo, inserem tarefas e métodos que devem ser realizados durante cada etapa do processo, um exemplo, são os métodos de teste de software. O quadro 2.3 apresenta alguns trabalhos que realizam avaliações de qualidade de software utilizando a abordagem baseada no processo de desenvolvimento.

| | |
|---|---|
| Patenaude, *et al*. (1999) | Neste artigo, os autores tratam especificamente de avaliação de qualidade em projetos que utilizam Java como linguagem de programação. A avaliação é feita utilizando métricas específicas aplicadas ao código fonte, desta forma busca-se identificar características indesejadas no software desenvolvido. Segundo os autores, a ferramenta de avaliação de qualidade proposta pode ser utilizada em sistemas de médio e grande porte. |
| Carvalho *et al.* (2001) | Os autores propõem uma estratégia para a implantação dos processos de desenvolvimento e gerenciamento de requisitos. A estratégia define atividades e documentos que devem ser criados durante o desenvolvimento do software. Segundo os autores, para que a qualidade seja alcançada os requisitos precisam ser bem definidos e controlados e, para isto, é necessário estabelecer e implantar processos. |
| Crespo *et al.* (2004) | Neste trabalho propõe-se uma metodologia para a melhoria da atividade de teste de software. Após a apresentação da metodologia, os autores descrevem aplicação em micro empresas de software. Segundo os mesmos, a metodologia possibilita a melhoria da qualidade de software por meio da melhoria do processo. |
| Rosqvist *et al.* (2003) | Um *framework* é apresentado para avaliação de software, baseado no julgamento de especialistas, neste caso os autores buscam o consenso entre os especialistas. O *framework* deve apoiar os desenvolvedores no controle de qualidade do processo de desenvolvimento de software e como conseqüência do produto de software, para isso os autores sugerem que periodicamente sejam realizadas verificações e validações. |
| Belgamo *et al*. (2005) | O artigo tem como objetivo testar a viabilidade de aplicação de uma técnica denominada GUCCRA para a identificação de defeitos na especificação de casos de uso. Desta forma esta técnica apóia a atividade de inspeção presente no desenvolvimento de software. |
| Gousios *et al.* (2007) | Os autores desenvolvem uma plataforma com o objetivo de automatizar a avaliação da qualidade de software. A plataforma é voltada especificamente para projetos *open source* e a avaliação realizada tem como referência requisitos não funcionais. |
| Janzen e Saiedian (2008) | Os autores apresentam um estudo relacionado ao TDD (Test-Driven Development – Desenvolvimento Orientado a testes) enfatizando a importância da técnica. A técnica citada, tem como objetivo antecipar a identificação e a correção de falhas durante o desenvolvimento de software. Segundo os autores, com base no estudo realizado, não é possível afirmar que o TDD melhora a coesão e diminui o acoplamento. Estas são características inerentes ao software e que precisam ser consideradas. |
| Bertrán (2009) | A autora afirma que muitos trabalhos focam no código fonte para a identificação de métricas que auxiliem a redução dos erros do desenvolvimento de software. No entanto, segundo a autora, para evitar o retrabalho é importante utilizar métricas em etapas anteriores do desenvolvimento. Por isso um conjunto de estratégias é proposto com o objetivo de identificar problemas de *design* específicos em modelos UML (*Unified Modeling Language*). |

Quadro 2.3– Exemplos de avaliações de qualidade de software baseadas no processo.

Os trabalhos listados no quadro 2.3 permitem observar que abordagens baseadas no processo de desenvolvimento, de uma forma geral, propõem e utilizam

técnicas associadas a etapas ou atividades específicas do processo de desenvolvimento.

### 2.3.1 Normas e Padrões relativos à Qualidade de Software

Normas e padrões têm sido aplicados na tentativa de elevar a qualidade de software, alguns são mais voltados para o processo de desenvolvimento e outros para o produto derivado do processo. Esta seção descreve de forma resumida algumas normas e padrões que têm sido utilizados.

Uma das normas mais utilizadas é a ISO/IEC 9126-1. O modelo de qualidade ISO/IEC 9126-1 é composto por duas partes. A primeira parte é referente à qualidade interna e à qualidade externa. Esta especifica seis características para qualidade de software: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Cada uma das seis características é subdividida em subcaracterísticas (NBR ISO/IEC 9126-1, 2003).

A segunda parte do modelo se refere à qualidade em uso e consiste na visão do usuário a respeito da qualidade. A qualidade em uso segundo a ISO/IEC 9126-1 é composta pelos critérios: eficácia, produtividade, segurança e satisfação (NBR ISO/IEC 9126-1, 2003).

Uma segunda norma utilizada é a ISO/IEC 15504 que define a avaliação do processo de desenvolvimento. Esta ISO descreve os processos relevantes de acordo com as boas práticas da engenharia de software (ISO/IEC 15504-1, 2004).

ISO/IEC 14598 fornece uma estrutura para avaliar a qualidade de quaisquer produtos de software e estabelece os requisitos para métodos de medição e avaliação de produtos de software. Além disso, contém requisitos, recomendações e orientações para a medição, julgamento e avaliação sistemática da qualidade de produto de software durante a aquisição de produtos de software (ABNT, 2009).

ISO/IEC 12207 estabelece uma estrutura comum para os processos de ciclo de vida de software que pode ser referenciada pela indústria de software. A estrutura contém processos, atividades e tarefas que servem para serem aplicadas durante a aquisição de um sistema (ABNT, 2009).

O quadro 2.4 classifica as normas descritas anteriormente em relação ao foco principal: produto ou processo.

| Norma/Padrão | Produto | Processo |
|---|---|---|
| ISO/IEC 9126-1 | X | |
| ISO/IEC 15504 | | X |
| ISO/IEC 14598 | X | |
| ISO/IEC 12207 | | X |

Quadro 2.4 – Normas relativas a qualidade de software, classificação em relação ao foco principal.

As normas descritas podem ser utilizadas de forma conjunta para alcançar determinados objetivos, é o caso, por exemplo, das normas ISO/IEC 14598 e ISO/IEC 9126. Segundo ABNT (2009), existe a previsão de substituição dessas duas normas por uma nova série de normas a ISO/IEC 25000, denominada “Requisitos e Avaliação da Qualidade de Produto de Software” (SQuaRE).

Além das normas descritas, alguns modelos de referências são bastante utilizados, é o caso do CMMI (Capability Maturity Model Integration). Segundo Morgado *et al*. (2007), trata-se de um modelo de maturidade para o desenvolvimento e manutenção de software. Este modelo fornece um conjunto de boas práticas que são agrupadas de acordo com 5 níveis de maturidade: otimizado, gerenciado, definido, repetitivo e inicial.

Já o MPS.Br é o programa para Melhoria de Processo do Software Brasileiro, trata-se de um modelo baseado no CMMI e em normas ISO. Em desenvolvimento desde dezembro de 2003, o programa é coordenado pela Associação para Promoção da Excelência do Software Brasileiro - SOFTEX (RENAPI, 2010).

O MPS.Br é composto por sete níveis de maturidade: Em Otimização (A); Gerenciado Quantitativamente (B); Definido (C); Largamente Definido (D); Parcialmente Definido (E); Gerenciado (F); Parcialmente Gerenciado(G). A implantação do modelo em empresas e em projetos de software inicia-se no Nível G, chegando gradativamente ao Nível A. O objetivo é permitir que as organizações possam evoluir e amadurecer seus processos de desenvolvimento de software de forma incremental (RENAPI, 2010).

Além das normas e padrões, a conformidade com os requisitos também é uma questão relevante para o alcance da qualidade de software. Por este motivo a seção seguinte descreve o processo de desenvolvimento de software e as fases de elicitação e priorização de requisitos que fazem parte desse processo.

## 2.4 O processo de desenvolvimento de software

O aspecto não repetitivo do desenvolvimento de software torna essa atividade difícil e até mesmo imprevisível. Apenas uma pequena parcela da construção de software corresponde a atividades que correspondem à “montagem”. Desta forma a atividade de desenvolvimento de software caracteriza-se como uma atividade complexa (KOSCIANSKI E SOARES, 2007).

O desenvolvimento de software é constituído por várias etapas e atividades que devem ser realizadas com o objetivo de obter um produto de software. Ao longo do tempo diversos modelos de processo de desenvolvimento têm sido criados e testados. Segundo Pressman (2002), o modelo mais adequado para um determinado projeto deve ser escolhido com base na natureza do projeto e da aplicação, nos métodos e ferramentas a serem usados e nos produtos intermediários e finais que são requeridos.

De uma forma geral, um projeto de desenvolvimento de software tem início quando se identifica a necessidade de desenvolvimento e decide-se investir no mesmo. A partir daí são realizadas diversas atividades em diferentes etapas, dentre as quais é possível destacar (BOOCH *et al*., 2000):

- Análise de Requisitos
- Análise e Projeto
- Implementação
- Teste

O desenvolvimento iterativo incremental é atualmente uma das metodologias mais usadas para a implementação de software. Uma pesquisa realizada com

profissionais de diferentes tipos de indústrias revelou que, de acordo com os dados obtidos, em projetos com mais de dois anos de duração o desenvolvimento incremental é a metodologia de desenvolvimento mais utilizada (NEILL E LAPLANTE, 2003).

Essa metodologia possibilita a redução dos riscos e a obtenção de um produto funcionando mais cedo (BOEHM, 1983). Por isso tem sido considerada por alguns autores como uma metodologia adequada para o desenvolvimento de software.

Segundo esta abordagem, o desenvolvimento é constituído de uma série de miniprojetos, chamados de iterações. Em cada iteração são realizadas as atividades anteriormente listadas. O objetivo de cada iteração é a obtenção de um produto com qualidade superior ao produto derivado da iteração anterior. Várias iterações acontecem para que no final seja obtido um software pronto para ser utilizado (LARMAN, 2004).

Sommerville (2007) destaca que o desenvolvimento iterativo possibilita o refinamento das informações com os usuários para que seja possível produzir um software que os satisfaçam.

A figura 2.1, elaborada por Cordeiro e Freitas (2008), ilustra do processo de desenvolvimento de software de forma simplificada.

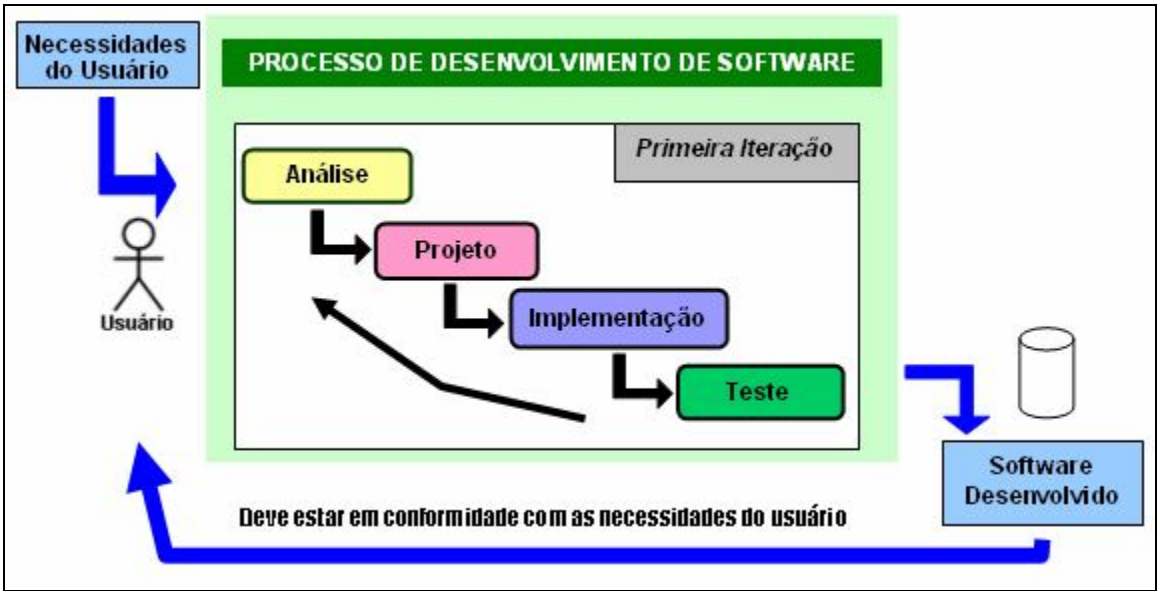


Figura 2.1– Visão simplificada de um processo de desenvolvimento

Fonte: Cordeiro e Freitas (2008)

Na figura 2.1 tem-se a representação das necessidades do usuário como ponto de partida para o desenvolvimento do software. A partir dessas necessidades o processo de desenvolvimento ocorre através da execução de suas etapas. No desenvolvimento iterativo, a cada iteração as etapas são executadas e uma entrega é feita aos usuários.

Após várias iterações tem-se o produto desenvolvido, mesmo sendo comum o software sofrer várias modificações para adaptação à realidade dos usuários ou para correção de erros. Este produto, resultante do processo, deve estar de acordo com as necessidades dos usuários.

Em geral metodologias que buscam a qualidade do produto de software, definem atividades que são desempenhadas ao final do processo de desenvolvimento, com o software já desenvolvido. Este tipo de abordagem por ser observado em Xenos e Christodoulakis (1995); Silva (2003) e Celestino e Abe (2009).

Por outro lado, metodologias que visam à qualidade do processo de desenvolvimento definem atividades realizadas durante o processo, por meio da melhoria das etapas intermediárias: análise de requisitos; análise e projeto; implementação e teste. Tais metodologias interferem diretamente na forma como os desenvolvedores executam as atividades, como pode ser notado em Carvalho et al. (2001); Rosqvist et al. (2003) e Janzen e Saiedian (2008).

A etapa de análise dos requisitos é um dos aspectos centrais para o entendimento da metodologia proposta por esta dissertação. Por isso as seções seguintes descrevem algumas atividades executadas nesta etapa do processo de desenvolvimento.

### 2.4.1 Elicitação de Requisitos

O software deve possuir características que contribuam para a solução de problemas e melhoria da qualidade de trabalho dos usuários, tendo como conseqüência a melhoria da qualidade do serviço ou produto da empresa à qual se destina o software. Portanto é preciso utilizar uma forma adequada de identificar tais características, que constituem os requisitos.

A Engenharia de Requisitos está relacionada à descoberta das características desejadas e sua transformação em uma forma que sirva de base para o desenvolvimento de um produto que deve possuir essas características (SAWYER *et al*., 1998).

A elicitação de requisitos e a priorização de requisitos são atividades situadas no contexto da engenharia de requisitos. Essas atividades são apresentadas a seguir.

A elicitação de requisitos tem sido reconhecida como uma das mais determinantes para a qualidade do software. Para Larman (2004), requisitos são capacidades e condições às quais o sistema – e de forma mais ampla, o projeto – deve atender.

O *Swebok Guide* (2004) define requisitos como uma propriedade que um software deve ter para resolver um problema do mundo real.

Pinheiro *et al*. (2003) destacam também as limitações tecnológicas ao fornecerem sua definição de requisitos de software. Segundo os autores, a engenharia de requisitos funciona como uma ponte entre as reais necessidades dos usuários, clientes e outras partes influenciadas pelo sistema de software e as capacidades e oportunidades permitidas pelas tecnologias de software.

Robertson e Robertson (2006) conceituam requisitos como sendo algo que um produto deve fazer ou uma característica que o produto de ter, e que é necessário ou desejado pelos *stakeholders.*

Young (2004) complementa afirmando que requisitos são atributos necessários em um sistema para que ele tenha valor e utilidade para os clientes e usuários. Para o autor, os requisitos do sistema são importantes porque fornecem a base para todo o trabalho de desenvolvimento de software subseqüente.

Os requisitos, dentre outras classificações possíveis, podem ser classificados como funcionais e não funcionais.

Os funcionais são as necessidades específicas dos usuários em relação ao sistema de acordo com o tipo de software. Descrevem o que o sistema deve fazer e dependem do tipo de software que está sendo desenvolvido e dos usuários a que se destina o software (SOMMERVILLE, 2007).

Os requisitos não funcionais estão relacionados às características que o sistema deve ter para atender às expectativas dos usuários. Por exemplo, o usuário pode especificar que o sistema deve possuir um bom tempo de resposta e este se

classifica como um requisito não funcional. Logo, os requisitos não funcionais estão associados às restrições identificadas para o software (*SWEBOK GUIDE*, 2004; SOMMERVILLE, 2007).

De uma forma geral, para que no final do projeto se obtenha de fato um produto adequado, é necessário identificar corretamente os requisitos, sejam eles funcionais ou não funcionais, já que é com base neles que o software é implementado. Para Karatzas *et al.* (2003) é muito importante definir cuidadosamente os requisitos de um sistema para esclarecer as características, funções e atributos necessários para os usuários.

Considerando que são as necessidades dos usuários que justificam o investimento em um projeto de software, não faz sentido que o foco principal do projeto seja outro senão a solução para essas necessidades.

Embora essa seja uma afirmativa lógica, a realidade de muitos projetos de software mostra que é comum os objetivos do projeto se distanciarem das necessidades e desejos de seus usuários.

Neste sentido, a fase de elicitação de requisitos pode ser compreendida como base para todas as outras atividades relativas ao desenvolvimento de software.

Segundo Leffingwell e Widrig (1999) erros na elicitação de requisitos representam a classe mais significativa de problemas relacionados ao desenvolvimento de software. Os mesmos autores afirmam ainda que erros causados por uma elicitação de requisitos inadequada são os mais caros de serem corrigidos.

Uma elicitação ineficaz dos requisitos traz conseqüências que podem levar ao fracasso do projeto. Isso pode ser explicado pelo fato desta etapa constituir a base para as atividades subseqüentes do desenvolvimento. Se a base é mal construída, os erros decorrentes da mesma podem continuar acontecendo e até mesmo se tornarem mais complexos durante todas as fases do projeto. Segundo Larman (2004), a indústria de software está repleta de projetos fracassados que não forneceram o que as pessoas realmente necessitavam.

De acordo com Boehm (1983) quando os problemas são detectados em fases mais avançadas do processo de desenvolvimento a correção se torna mais difícil e as etapas do processo precisam ser refeitas, o que ocasiona retrabalho. O retrabalho tem conseqüências negativas para o projeto como o aumento dos custos e atrasos no cronograma.

Segundo Boehm e Papaccio (1988) o retrabalho tende a seguir a distribuição de Pareto: 80 por cento do custo de retrabalho é resultante de 20 por cento dos problemas. Dessa forma ressalta-se a importância da identificação dos problemas nas fases iniciais do projeto como forma de reduzir grande parte dos custos.

A compreensão dos usuários, das suas necessidades e de como eles trabalham no contexto do sistema proposto, pode aumentar de forma significativa a possibilidade de sucesso do desenvolvimento de software. Contudo, o grau de compreensão depende da forma como os requisitos são levantados (COUGHLAN E MACREDIE, 2002).

A fase de elicitação de requisitos do processo de desenvolvimento de software é caracterizada por atividades de comunicação e envolve um grupo de pessoas que são distintas em termos de nível de conhecimento, habilidades e *status*. O objetivo desta atividade é o entendimento do problema e do contexto onde o software será inserido (COUGHLAN E MACREDIE, 2002).

Hickey e Davis (2002) destacam as atividades relativas ao processo de requisitos:

- Elicitação: aprendizado e descoberta das necessidades dos clientes, usuários e outros *stakeholders;*
- Modelagem: criação e análise dos modelos de requisitos com o objetivo de aumentar o entendimento e buscar inconsistências;
- Triagem: determinação de qual subconjunto de requisitos deve ser direcionado a cada versão do software;
- Especificação: documentação do funcionamento externo desejado do sistema;
- Verificação: determinação da consistência, integridade e ausência de defeitos de um conjunto de requisitos.

As técnicas utilizadas são de grande importância para a elicitação de requisitos porque elas facilitam a comunicação entre as pessoas envolvidas. Neste sentido uma técnica de elicitação de requisitos pode ser entendida como um método de mediação da comunicação.

O quadro 2.5 resume as características das técnicas identificadas por Maiden e Rugg (1996).

| Técnica | Descrição | Forças | Fraquezas |
|---|---|---|---|
| Entrevistas Não-Estruturadas | Pergunta-se a um *stakeholder* a respeito do assunto sem ter preparado uma lista de perguntas. | Simplicidade, permite saber do *stakeholder* o que é relevante. | Pode gastar muito tempo em assuntos secundários. A informação obtida pode ser difícil de analisar. |
| Entrevistas Estruturadas | Prepara-se uma lista de perguntas para fazer ao *stakeholder*. | Sistemático. | Impõe um padrão ao usuário e assuntos importantes podem faltar. |
| Análise de Protocolo | Alguém responsável por uma tarefa e explica durante a sua execução. | Fornece muitos dados relativos a tarefas específicas. | Protocolos são difíceis de entender. |
| Brainstorming | Solicita-se a um grupo de *stakeholders* para gerarem quantas idéias forem possíveis, com maior ênfase na geração do que na avaliação das idéias. | Bom para elicitar entidades do domínio de alto nível. | Suscetível a processos em grupo; não sistemático; |
| Prototipagem rápida | Solicita-se ao *stakeholder* que comente sobre um modelo em protótipo do sistema desejado. | Bom para capturar questões tomadas como verdade. | Leva tempo e esforço para produzir. Protótipo evolutivo pode se tornar um perigo para o projeto do sistema, se mal utilizado. |
| Análise de cenário | Um cenário é uma descrição de uma seqüência de ações e eventos para um caso específico de alguma tarefa genérica que o sistema deve realizar. Cenários incluem casos de uso. | Integração com métodos estruturados. | Gough *et al.* (1995) relataram que o tempo gasto durante a geração foi um fator desmotivador. |
| Workshops RAD | Workshop no qual de 8 a 20 indivíduos tomam decisões buscando consenso. Deve haver um facilitador que não seja um *stakeholder* do sistema. | Melhora a qualidade e a velocidade do projeto do sistema. Integrado com os métodos estruturados atuais. | Workshops interrompem as tarefas dos *stakeholders*, algumas vezes por dias. É efetivo apenas para sistemas menores. |
| Métodos etnográficos | Gastam-se longos períodos de tempo observando os usuários no seu ambiente de trabalho. | Bom para capturar questões tomadas como verdade. | Requer muito tempo. Pode falhar na detecção de eventos que não são freqüentes. |

Quadro 2.5- Técnicas de elicitação de requisitos.

Adaptado de Maiden e Rugg (1996).

Ao observar o quadro 2.5 é possível notar que várias técnicas de elicitação podem ser usadas, no entanto cada uma possui características que as tornam adequadas ou não para um determinado projeto de desenvolvimento de software. Além disso, é possível que uma técnica seja mais adequada que outra em determinada fase do projeto.

As técnicas de elicitação devem ser definidas de acordo com as características dos requisitos e do negócio onde o software será inserido (COUGHLAN E MACREDIE, 2002).

Para Koscianski e Soares (2007) provavelmente a entrevista é a técnica mais comum. As entrevistas podem ser substituídas por questionários, caso, por exemplo,

os usuários estejam situados geograficamente distantes dos desenvolvedores. Neste caso, o primeiro contato para elicitação poderia ser feito através de questionários.

### 2.4.2 Priorização de Requisitos para o desenvolvimento de software

Como exposto anteriormente os requisitos possuem grande relevância para a qualidade do produto gerado pelo processo de desenvolvimento de software. No entanto, uma das dificuldades encontradas por equipes de desenvolvimento está relacionada à impossibilidade de desenvolver todos os requisitos de uma só vez. De uma forma geral, as equipes definem um conjunto de funções ou processos do software para iniciar a atividade de desenvolvimento.

Cada vez mais as empresas estão sentindo a dificuldade de alcançar equilíbrio entre requisitos e recursos. Orientadas por um mercado competitivo, elas tentam adicionar características e reduzir os cronogramas para a entrega de cada produto. O resultado pode levar ao desacordo entre recursos e requisitos, gerando produtos que não conseguem atender as necessidades dos clientes (DAVIS, 2003).

Gilb e Maier (2005) conceituam prioridade como o direito relativo que um requisito tem de utilizar recursos limitados ou escassos. Neste contexto recurso pode ser compreendido como tempo, esforço humano, recursos financeiros, espaço ou qualquer outro tipo de recurso.

Os requisitos são itens que os usuários querem porque possuem valor para eles. As prioridades são como eles esperam obtê-las considerando os diversos níveis de requisitos diante dos recursos limitados. Se os recursos fossem ilimitados não haveria a necessidade de priorização (GILB E MAIER, 2005).

No entanto, os recursos necessários ao desenvolvimento de um software são limitados. Devido a restrições de tempo e orçamento, pode ser difícil implementar todos os requisitos identificados para um sistema. Os requisitos geralmente são implementados em etapas e a priorização ajuda a definir quais devem ser implementados primeiro (ALLEN *et al*., 2008).

Karlsson *et al.* (1998) afirmam que os requisitos precisam ser alocados em diferentes versões do software. Neste processo é necessário considerar a quantidade de requisitos e a classificação dos requisitos como funcionais e não

funcionais. Não é aconselhável a inclusão de requisitos funcionais e não funcionais em um mesmo processo de avaliação (KHAN, 2006).

Karlsson e Ryan (1996) e Berander (2004) apresentam visões semelhantes ao afirmarem que seleção do conjunto de requisitos é determinante para a satisfação dos usuários.

Para Berander (2004) a seleção de forma “correta” dos requisitos que farão parte de cada versão é a etapa principal em direção ao sucesso de um projeto ou produto.

Por isso, de acordo com Duan *et al.* (2009), os desenvolvedores precisam ser cuidadosos na seleção do subconjunto dos requisitos que serão desenvolvidos. É necessário distinguir os requisitos que terão maior impacto para a satisfação dos usuários.

De acordo Karatzas *et al.* (2003) é essencial que os *stakeholders* concordem com qual subconjunto de requisitos será inicialmente implementado. A priorização auxilia os gerentes do sistema a resolver conflitos e planejar as entregas.

No entanto, para Karlsson *et al.* (1998) não basta apenas priorizar, é necessário buscar formas para comunicar o resultado da priorização aos membros do projeto. Os autores definem três etapas necessárias em um processo de priorização de requisitos:

i) Preparação: etapa na qual uma pessoa estrutura os requisitos de acordo com o método de priorização utilizado. Uma equipe é escolhida e recebe as informações necessárias;

ii) Execução: nesta etapa os decisores realizam a priorização utilizando para isso as informações recebidas na etapa anterior;

iii) Apresentação: consiste na apresentação dos resultados aos envolvidos. Alguns métodos de priorização utilizam diferentes formas para os cálculos das prioridades, mas estes devem ser realizados antes da apresentação dos resultados.

Os métodos de priorização de requisitos são classificados em duas categorias: métodos baseados em valores atribuídos aos requisitos e métodos de negociação (LEHTOLA E KAUPPINEN, 2004).

Os métodos classificados na primeira categoria permitem a priorização de forma absoluta ou relativa. Quando a atribuição é feita de forma absoluta, os valores são atribuídos a cada requisito sem levar em consideração os demais. Na

priorização de forma relativa os valores são atribuídos aos requisitos comparativamente aos demais (LEHTOLA E KAUPPINEN, 2004).

Os métodos de negociação baseiam-se na idéia de que a prioridade de um requisito pode ser determinada através do consenso entre os diferentes *stakeholders* (LEHTOLA E KAUPPINEN, 2004).

Esta dissertação tem como foco os métodos de priorização que se baseiam na atribuição de valores. Alguns desses métodos listados por Allen *et al*. (2008) são apresentados no quadro 2.6:

| MÉTODO | DESCRICAO | TIPO DE PRIORIZAÇÃO |
|---|---|---|
| Árvore de busca binária | Algoritmo utilizado para a busca de informações. Consiste em selecionar um dos requisitos de uma lista e colocá-lo no nó raiz, a partir daí cada requisito restante na lista deve ser comparado com cada nó da árvore em relação a importância. Ao final obtém-se a árvore de requisitos priorizados. | Relativa |
| Atribuição numérica | Utiliza uma escala de 1 a 5, os *stakeholders* devem identificar a qual nível da escala cada requisito corresponde. | Absoluta |
| Método dos 100 pontos | A cada *stakeholder* são fornecidos 100 pontos que devem ser utilizados em favor dos requisitos mais importantes com base na percepção de cada um. | Relativa |
| Triagem de requisitos | Consiste em um processo de várias etapas que inclui estabelecer prioridades relativas para os requisitos, estimar recursos necessários para cada requisito e selecionar um subconjunto de requisitos para otimizar a probabilidade de sucesso do projeto. | Relativa |
| AHP (Analytic Hierarchy Process - Método de Análise Hierárquica) | Método de apoio à decisão utilizado em situações nas quais múltiplos objetivos estão presentes. O método utiliza a comparação par a par para calcular o valor relativo de cada item, neste caso requisito. | Relativa |

Quadro 2.6- Métodos de priorização de requisitos.

Adaptado de Allen *et al*. (2008)

No método dos 100 pontos os avaliadores devem distribuir os 100 pontos entre os atributos que estão sendo avaliados. A soma dos pontos atribuídos a todos os itens deve totalizar 100 (ALLEN *et al*., 2008).

Malhotra (2006) refere-se a este método denominando-o como “escala de soma constante”, mas afirma que os entrevistados devem atribuir uma soma constante de unidades. Para o autor a unidade pode ser ponto ou outro tipo de unidade, além disso, a quantidade de unidades não precisa ser 100.

Segundo Malhotra (2006) a escala de soma constante é uma escala comparativa na qual os atributos são escalonados a partir do cálculo dos pontos atribuídos a cada um deles por todos os participantes e dividindo-se pela quantidade de respondentes.

Lee *et al*. (2008) utilizam AHP (*Analytic Hierarchy Process*) para definir a prioridade relativa dos requisitos necessários em sistemas voltados para a web. Para os autores embora o AHP possa ser utilizado em outros contextos, o método se mostra satisfatório para a obtenção das prioridades relativas de requisitos.

Por outro lado, Lehtola e Kauppinen (2004) fornecem uma visão contrária ao uso do AHP ao destacarem alguns desafios encontrados durante uma aplicação do método:

i) Embora a definição do requisito mais importante na comparação par a par seja relativamente simples, os usuários acharam difícil estimar quanto um requisito é mais importante em relação a outro, mesmo utilizando uma escala de 5 pontos;
ii) Comparações par a par com mais de 20 requisitos tornaram-se difíceis na prática, os usuários tiveram dificuldades em se concentrar nas comparações;
iii) Requisitos de diferentes níveis de abstração causaram problemas;
iv) Alguns usuários não aceitaram a comparação par a par. Eles consideraram que seria mais fácil selecionar os requisitos mais importantes ou colocar os requisitos em ordem descendente.

Os métodos listados anteriormente têm sido bastante utilizados para priorização de requisitos. No entanto, para Duan *et al.* (2009) é necessário buscar métodos capazes de realizar a priorização de requisitos de projetos mais complexos, de grande porte. Os autores propõem um novo método para automatizar parte do processo de priorização.

Além disso, uma das dificuldades associadas à priorização dos requisitos refere-se à possibilidade de mudança nos requisitos. Segundo Berander (2004) quando a escala utilizada para priorização é ordinal ou absoluta basta alterar o valor atribuído ao requisito que sofreu a mudança ou realizar a avaliação caso um novo requisito seja inserido. Por outro lado, quando uma escala relativa é utilizada, como

através do AHP, o problema torna-se mais complexo, todos os requisitos precisam ser reavaliados.

Uma forma de contornar esse problema, para o caso do método dos 100 pontos, por exemplo, consiste em identificar dois requisitos: um mais prioritário e outro menos prioritário. Em seguida o novo requisito deve ser avaliado relativamente aos dois requisitos de referência selecionados, por fim os valores obtidos são normalizados para o conjunto completo de requisitos. No entanto, deve-se avaliar a relação custo benefício desta solução visto que o resultado pode divergir do processo completo de priorização (BERANDER, 2004).

Outra dificuldade associada à priorização refere-se à dependência entre requisitos. Como exemplo, um requisito prioritário A pode depender de um requisito menos importante B. Em situações desse tipo, geralmente opta-se por desenvolver o requisito B antes do requisito A, o que faz com que as prioridades dos usuários não sejam seguidas. No entanto, para resolver esse problema é possível utilizar um *test double*.

Segundo Meszaros (2010) quando é necessário testar uma parte do sistema que depende de outros componentes que não podem ser utilizados no ambiente de testes, é possível utilizar um *test double* para substituir os componentes não disponíveis. Desta forma é possível contornar o problema da dependência entre componentes.

Um tipo de *test double* é o *mock*, segundo Freeman *et al*. (2004), objetos *Mock* permitem que os testes sejam mais bem estruturados, preserva o encapsulamento e reduz as dependências. Para os autores, objetos *Mock* permitem escrever o código a ser testado como se ele tivesse tudo o que precisa do seu ambiente.

## 2.5 Análise Importância-Desempenho (Performance)

A análise importância-desempenho (IPA – Importance-Performance Analysis) é um instrumento que começou a ser utilizado em 1977. Inicialmente o método foi aplicado em um departamento de serviços de um distribuidor de automóveis (MARTILLA E JAMES, 1977).

Atualmente está sendo amplamente usado para mensuração de satisfação de clientes em diferentes ramos de negócios, como exemplo, é possível citar a indústria hoteleira e os serviços bancários. Alguns autores usam o termo SERVIMPERF para se referir a esta análise, no entanto, nesta dissertação optou-se por utilizar o termo IPA, pois é o mais utilizado na literatura sobre o assunto.

Slack (1994), desenvolveu uma forma de apresentar os dados que reflete a percepção dos gerentes na relação entre importância, desempenho e prioridade para melhoria, com base na forma como os dados são apresentados nesta técnica.

Skok *et al.* (2001) realizaram a avaliação do sucesso de investimentos em sistemas de informação em uma organização da área de saúde através do uso da análise importância-desempenho.

Magal e Levenburg (2005) utilizaram a IPA para avaliar estratégias de *e-business* em organizações de pequeno porte. Com base nos resultados obtidos os autores sugerem recomendações para alocação de recursos, indicando quais ações são mais urgentes com base na priorização obtida através da IPA.

Cordeiro e Moll (2006) aplicaram a IPA para avaliação da qualidade de produto de software utilizando como critérios requisitos não funcionais. As avaliações de importância e desempenho foram realizadas em um mesmo momento para um produto de software já desenvolvido e em uso. A aplicação do método mostrou-se satisfatória para a identificação dos requisitos que necessitavam de melhorias, ou seja, que possuíam grande importância para os usuários, mas que não apresentaram desempenho adequado de acordo com a percepção destes.

Segundo Hudson *et al*. (2004) a IPA mostra a importância relativa a vários atributos e o desempenho do produto no que se refere ao estudo e análise desses atributos. Na aplicação deste método, cada entrevistado deve julgar a importância que um atributo tem de acordo com a sua opinião, bem como a sua percepção de desempenho para cada atributo.

Na pesquisa aplicada por Hudson *et al*. (2004) os clientes foram entrevistados a respeito de importância e desempenho em momentos distintos. A pesquisa foi aplicada a clientes de agências de turismo para medir a satisfação dos mesmos com os serviços prestados. Antes de fazer a viagem, os clientes responderam sobre a importância de cada item e no retorno informaram sua percepção de desempenho.

Duke e Mount (1996) afirmam que, tendo como base um conjunto de critérios, a avaliação de importância permite a identificação das prioridades de cada um dos

critérios para os clientes. Para os autores, a IPA pode ser usada para compreender a percepção dos usuários a respeito de elementos críticos do produto. A relação entre a avaliação de cada elemento crítico e sua importância permite que os gerentes captem o sucesso do produto em cada característica e sua prioridade conforme classificado pelo cliente.

O uso da IPA pode trazer benefícios gerenciais consideráveis, um deles é a possibilidade de identificar exatamente as áreas que necessitam de melhorias de qualidade. Para Skok *et al.* (2001) a análise importância-desempenho tem se revelado com uma ferramenta de gestão simples e efetiva. Contudo, segundo os autores existem poucos relatos a respeito da utilização desta ferramenta na área de sistemas de informação.

Uma das vantagens relatadas na literatura a respeito da IPA refere-se à forma de apresentação dos resultados característica desse método.

Leeworthy e Wiley (1996) citam que a análise importância-desempenho oferece uma forma de visualização simples, na qual a apresentação dos dados é feita por meio de quatro quadrantes que formam a matriz da análise importância-desempenho.

Segundo Skok *et al.* (2001) a IPA apresenta a percepção de importância e desempenho dos clientes em um formato que facilita a interpretação dos resultados. A figura 2.2 é um exemplo desse tipo de matriz.

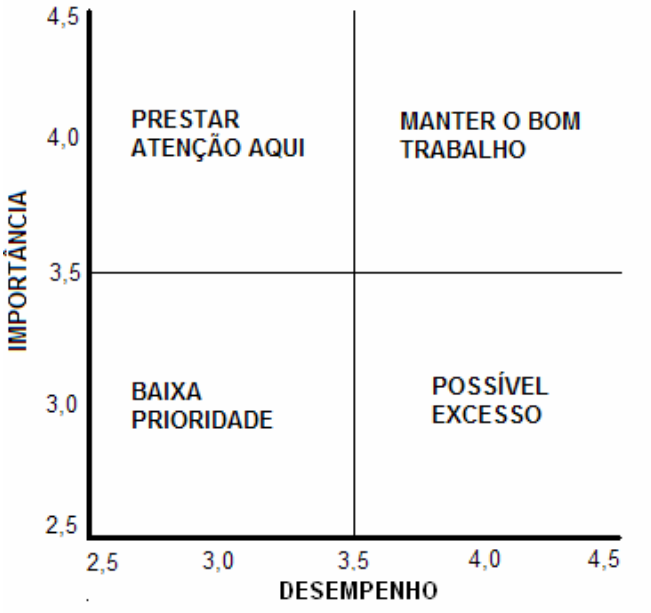


Figura 2.2- Exemplo de matriz de Análise Importância-Desempenho.
Adaptado de Leeworthy e Wiley (1996)

Conforme mostrado na figura 2.2, o eixo vertical da matriz é referente à percepção de importância e o horizontal à percepção de desempenho.

De acordo com Skok *et al.* (2001) o início da avaliação através deste instrumento ocorre com a identificação de uma lista de elementos ou atributos sobre os quais a avaliação deve ser conduzida. A partir da lista, os instrumentos de pesquisa devem ser desenvolvidos para a coleta dos graus de importância e desempenho, para isso são utilizadas escalas numéricas. Em seguida estes questionários são aplicados ao grupo de respondentes.

Depois de realizada a mensuração, os atributos utilizados durante a avaliação são dispostos na matriz e de acordo com a posição onde ficam situados, obtém-se o resultado relativo à satisfação dos clientes em relação aos mesmos.

Quando um atributo é classificado no quadrante “Manter o bom trabalho”, isto significa que o atributo possui uma grande importância para o cliente e que o mesmo está satisfeito com o desempenho. Segundo Wade e Eagles (2003), importância e desempenho se enquadram nos padrões de qualidade, logo esse é um ótimo resultado.

O quadrante “Possível excesso” concentra atributos com os quais os clientes estão satisfeitos com o desempenho, mas que possuem pouca importância. No entanto essa interpretação deve ser tomada com a devida cautela, visto que o fato do atributo apresentar um bom desempenho pode tornar confuso para o usuário o conceito de importância, é possível até mesmo que o cliente não tenha conhecimento a respeito da importância do atributo.

No quadrante “Baixa prioridade” são dispostos os atributos que possuem baixo desempenho, mas que também não possuem muita importância para os clientes, ou seja, embora de acordo com clientes necessitem receber melhorias, estas não são prioritárias.

E por fim, a classificação no quadrante “Prestar atenção aqui”, de acordo com Blose *et al*. (2005), indicam que itens considerados importantes pelos clientes estão precisando receber melhorias. Os atributos concentrados neste quadrante devem ser considerados como prioritários na busca pela melhoria de qualidade.

Sendo assim, a matriz de análise importância-desempenho permite que sejam analisados visualmente os resultados dos atributos presentes na pesquisa de satisfação.

Para Skok *et al.* (2001) a simplicidade da análise importância-desempenho e o seu baixo custo de aplicação tornam viável a aplicação de outras avaliações para acompanhamento das melhorias realizadas. Para os autores, a representação visual auxilia no processo de tomada de decisão, possibilitando conclusões na hora certa e facilitando a recomendação de ações.

Levando em consideração as características da IPA apresentadas nesta seção, acredita-se que esse método pode ser de grande utilidade para a avaliação da qualidade de software. Durante a elicitação de requisitos a análise importância-desempenho pode ser usada para identificar quais atributos os usuários consideram mais importantes e após o desenvolvimento o método pode ser utilizado para mensuração do desempenho.

Logo, a exemplo da pesquisa realizada por Hudson *et al*. (2004) também é possível inquirir os clientes a respeito de importância e desempenho em momentos distintos no caso de produtos de software.

# CAPÍTULO III

# METODOLOGIA PROPOSTA

Neste capítulo apresenta-se inicialmente um modelo conceitual dos aspectos envolvidos nesta dissertação e em seguida descreve-se uma proposta de metodologia que possibilite a priorização de requisitos para o desenvolvimento de software e posterior avaliação da satisfação dos usuários em relação a esses requisitos.

## 3.1 Modelo Conceitual

A seguir descreve-se o modelo conceitual da dissertação com o objetivo de relacionar os conceitos tratados no referencial teórico e que são relevantes para o entendimento da metodologia proposta. A figura 3.1 representa esse modelo.

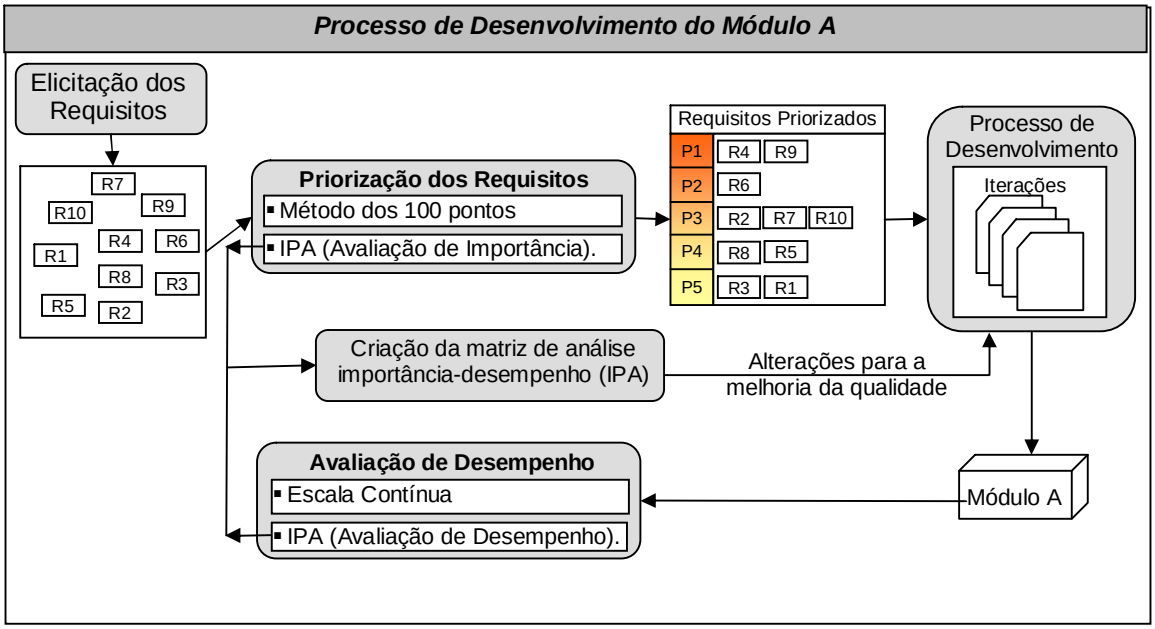


Figura 3.1- Modelo Conceitual

Conforme descrito, na literatura são identificados dois tipos de abordagens para a qualidade de software: a abordagem baseada no processo de desenvolvimento e a abordagem baseada no produto derivado deste processo.

Pressman (2002) afirma que alguns desenvolvedores de software continuam a achar que qualidade de software deve ser tratada após a etapa de programação do processo de desenvolvimento, no entanto essa deve ser uma preocupação presente no processo de desenvolvimento desde o seu início. Neste caso, a avaliação da qualidade e as melhorias no processo podem levar à qualidade do produto.

A metodologia proposta por esta dissertação sugere a avaliação da qualidade do produto de software, de forma que seja possível buscar a qualidade desde o início do desenvolvimento.

Apesar de existir a preocupação com a qualidade desde o início, não seria correto afirmar, neste caso, que a abordagem utilizada é baseada no processo porque o foco do estudo não está nas atividades executadas em cada etapa do desenvolvimento. O foco concentra-se na satisfação do usuário em relação ao produto de software resultante do processo de desenvolvimento.

Acredita-se, como destacado no referencial teórico, que a conformidade do software aos requisitos seja essencial para a satisfação dos usuários e que a satisfação dos usuários por sua vez seja determinante para a qualidade de software. Por isso, uma das etapas iniciais da metodologia proposta tem como foco os requisitos, conforme pode ser observado na figura 3.1.

Neste sentido, para que seja possível melhorar a qualidade de software propõe-se a priorização dos requisitos no início do desenvolvimento e a avaliação de desempenho quando o produto já está em uso. Estas duas etapas podem ser observadas na figura 3.1.

Após a elicitação dos requisitos obtém-se uma lista de requisitos que precisam ser desenvolvidos, contudo, neste momento, os desenvolvedores não sabem quais requisitos são mais relevantes para os usuários, por isso é necessário priorizá-los. Para a priorização dos requisitos, de acordo com a percepção dos usuários, utiliza-se o método dos 100 pontos em conjunto com a avaliação de importância da análise importância-desempenho.

Propõe-se a utilização destes dois métodos, dentre os citados no referencial teórico, pois possibilitam a avaliação dos requisitos de forma relativa e absoluta, respectivamente. Além disso, são considerados métodos de simples aplicação, ou seja, possuem baixa complexidade para os usuários facilitando o processo de avaliação. Segundo Leffingwell e Widrig (1999), o método dos 100 pontos geralmente traz resultados satisfatórios para priorização.

A priorização é relevante porque os usuários apresentam expectativas em relação ao software que está sendo desenvolvido. Além disso, a entrega dos requisitos, de acordo com a percepção dos usuários, além de contribuir para a satisfação dos usuários, pode ser essencial para a continuidade do projeto de desenvolvimento. No exemplo exposto pela figura 3.1, após a priorização, cinco níveis de prioridades foram definidos.

Caso o software seja desenvolvido de forma iterativa, várias iterações podem ocorrer até que o primeiro módulo seja considerado “pronto”. Nesta circunstância, entregas intermediárias do software devem ser feitas para que os usuários comecem a utilizar o software.

Quando todos os requisitos ou a maior parte deles já tiverem sido desenvolvidos e o módulo estiver em uso, é possível realizar a avaliação de desempenho do produto. Nessa avaliação utiliza-se uma escala contínua e a avaliação de desempenho da análise importância-desempenho como é possível notar no quadro referente à avaliação de desempenho da figura 3.1.

A escala contínua permite a classificação dos objetos através do uso de uma reta que vai de um extremo a outro do critério que está sendo avaliado. O entrevistado deve marcar na reta a posição relativa a sua avaliação para cada atributo e o pesquisador deve identificar a pontuação relativa à marcação (MALHOTRA, 2006).

Segundo Malhotra (2006), uma vantagem das escalas contínuas é a facilidade de construção e uma desvantagem é a dificuldade de identificação dos valores atribuídos na reta.

Os resultados obtidos através da avaliação de importância e da avaliação de desempenho da IPA devem ser usados para a construção da matriz de análise importância-desempenho. Conforme descrito do referencial teórico, a matriz permite a identificação dos atributos, neste caso requisitos, que necessitam de melhorias. As

alterações necessárias podem ser realizadas no módulo para a melhoria da qualidade.

Após as atividades de melhoria novas avaliações de desempenho podem ser realizadas com o objetivo de obter um produto mais adequado de acordo com a percepção dos usuários.

As etapas que constituem a metodologia proposta são as seguintes:

i) Identificação de uma equipe de desenvolvimento que esteja iniciando um projeto;

ii) Elicitação de requisitos;

iii) Priorização dos requisitos;

iv) Tabulação dos dados para priorização;

v) Desenvolvimento do software;

vi) Verificação do desenvolvimento dos requisitos prioritários;

vii) Implantação do software;

viii) Avaliação de desempenho do software;

ix) Tabulação dos dados para avaliação de desempenho;

x) Definição e execução das melhorias para os requisitos críticos;

xi) Realização de nova avaliação de desempenho (se desejado).

## 3.2 Metodologia Proposta

A seguir apresentam-se as etapas que compõem a metodologia proposta de forma detalhada.

### 3.2.1 Identificação de uma equipe de desenvolvimento que esteja iniciando um projeto

Como esta dissertação busca a qualidade de software desde o início do desenvolvimento, é preciso acompanhar o trabalho de uma equipe que tenha um

projeto sendo iniciado, desta forma é possível realizar a priorização dos requisitos e a avaliação de qualidade do software.

Além disso, a aplicação desta metodologia deve ser feita em um software cuja forma de chegada no mercado seja por encomenda porque softwares classificados dessa forma possibilitam o acompanhamento do processo de desenvolvimento. Softwares vendidos em pacote ou embarcados inviabilizam a aplicação da metodologia.

### 3.2.2 Elicitação de requisitos

No referencial teórico são apresentadas técnicas utilizadas para a elicitação de requisitos. A escolha das técnicas que serão utilizadas em cada projeto depende da equipe e das características do projeto. Em projetos que exigem curto período de desenvolvimento, por exemplo, não é adequado utilizar técnicas que exigem muito tempo para aplicação.

Além disso, como a metodologia busca a avaliação do software segundo a percepção dos usuários, os requisitos considerados são, preferencialmente, os requisitos funcionais. Acredita-se que os usuários conseguem perceber de forma mais significativa esse tipo de requisito. Embora os requisitos não funcionais também sejam desejáveis, representam restrições que o software deve possuir e não funções que os usuários esperam do software.

### 3.2.3 Priorização dos requisitos

Tendo em vista que nem sempre os requisitos definidos como prioritários pela equipe de desenvolvimento são os mais relevantes para os usuários, nesta dissertação objetiva-se realizar a priorização de requisitos para que seja possível iniciar o desenvolvimento pelos requisitos prioritários, de acordo com a percepção dos usuários.

Neste caso, considerando as categorias de métodos de priorização descritos no referencial teórico, utilizam-se os métodos de priorização baseados em valores

atribuídos aos requisitos. Como o objetivo é permitir que cada usuário faça a sua avaliação para a partir das avaliações individuais obter a avaliação do software, métodos de negociação não são utilizados.

**a. Desenvolvimento do questionário de priorização**

Após a definição da lista de requisitos na etapa anterior, é necessário preparar e aplicar o questionário de priorização. Esse questionário deve conter a lista de requisitos para que os usuários avaliem.

No referencial teórico o termo “atributos” é utilizado para descrever objetos ou características avaliados através dos métodos de priorização. No contexto de aplicação desta metodologia, tais atributos são os requisitos identificados como necessários na etapa de elicitação de requisitos.

Com o objetivo de facilitar a interpretação, sugere-se o agrupamento dos requisitos em relação ao contexto de aplicação das funcionalidades, ou seja, requisitos que estejam associados a um mesmo item do domínio de aplicação podem pertencer a um mesmo grupo.

O questionário deve ser elaborado de acordo com os métodos definidos para priorização: a análise importância-desempenho (avaliação de importância) e o método dos 100 pontos.

A IPA permite identificar a importância de cada item da lista de requisitos. Para isso, os usuários devem julgar cada requisito utilizando uma escala de Likert de 5 pontos. O valor 1 indica um requisito de menor importância e o valor 5, um requisito de maior importância. O quadro 3.1 apresenta a escala sugerida.

| **Grau de Importância** | | | | |
|---|---|---|---|---|
| **Nada Importante** | **Pouco Importante** | **Neutro** | **Importante** | **Muito Importante** |
| **1** | **2** | **3** | **4** | **5** |

Quadro 3.1- Escala de julgamento de importância.

O método dos 100 pontos funciona como um complemento à IPA. Neste método, cada usuário tem que distribuir 100 pontos pela lista de requisitos, atribuindo mais pontos aos requisitos considerados mais importantes, e menos pontos para os menos importantes. A soma dos pontos atribuídos aos requisitos deve totalizar 100 pontos.

Nesta etapa, a avaliação é feita de forma relativa, ou seja, um avaliador atribui seu julgamento para um requisito estabelecendo uma comparação com os demais requisitos.

**b. Aplicação do questionário de priorização**

Após a preparação, o questionário de priorização deve ser aplicado a futuros usuários que atuam no domínio onde o software será implantado. É preciso que os usuários tenham alguma experiência em relação à área de negócio do software para que estejam aptos a julgar a importância relativa dos requisitos.

Visando obter dados confiáveis é preciso observar a abordagem utilizada. Sugere-se aplicar o questionário individualmente, para que os usuários não interfiram nas opiniões uns dos outros.

### 3.2.4 Tabulação dos dados para priorização

Os dados contidos nos questionários devem ser tabulados para que seja definida a ordem de prioridade dos requisitos. A prioridade relativa de cada requisito é calculada com base na média da avaliação dos usuários que responderam o questionário.

**a. Resultados da Análise Importância-desempenho**

Para a análise de importância da IPA é necessário calcular a importância de cada requisito. A equação 3.1 representa o cálculo da média de importância para o requisito *r*.

$$\mu(GI)_r = \frac{\sum_{j=1}^{n}(GI)_{r,j}}{n} \quad (3.1)$$

Onde:

$\mu(GI)_r$ : Grau de Importância médio para o requisito r.

$(GI)_{r,j}$ : Grau de importância para o requisito r, segundo o avaliador j.

**b. Resultados do método dos 100 pontos**

A média de cada requisito também precisa ser calculada para o método dos 100 pontos. Neste caso deve-se calcular a média das pontuações atribuídas a cada requisito, equação 3.2 representa o cálculo dessa média.

$$\mu(P_r) = \frac{\sum_{j=1}^{n}(P)_{r,j}}{n} \quad (3.2)$$

As médias de importância e de pontuação de cada requisito devem ser calculadas através das equações 3.1 e 3.2. A partir das médias é possível obter a lista de requisitos priorizados para os dois métodos, valores mais altos para as médias indicam requisitos mais importantes (prioritários).

As listas de requisitos devem ser fornecidas à equipe de desenvolvimento para auxiliar os desenvolvedores na identificação dos requisitos mais relevantes de acordo com a percepção dos usuários.

### 3.2.5 Desenvolvimento do software

O software é desenvolvido seguindo a prioridade de cada requisito. No entanto, é importante ressaltar que a lista de requisitos priorizados constitui-se como uma base que deve servir de referência para o trabalho subseqüente dos desenvolvedores. Características específicas do projeto de desenvolvimento ou da

área de negócio podem impossibilitar que o desenvolvimento ocorra exatamente conforme definido pela lista de requisitos priorizados.

Caso seja possível, versões do software podem ser entregues aos usuários à medida que o desenvolvimento prossegue. O desenvolvimento iterativo descrito no referencial teórico permite essas entregas.

De acordo com as características do projeto, mais de um requisito pode ser desenvolvido concorrentemente, por exemplo, se a equipe for composta de muitos desenvolvedores. Mas em equipes menores é provável que em cada iteração apenas um requisito seja desenvolvido.

Como o desenvolvimento tem início pelos requisitos prioritários, nas versões iniciais, os usuários podem iniciar o uso do software pelos requisitos considerados mais importantes.

### 3.2.6 Verificação do desenvolvimento dos requisitos prioritários

Esta atividade consiste em verificar, através de um *check-list*, se os requisitos prioritários de fato foram desenvolvidos. A ausência de requisitos prioritários no software pode causar insatisfação nos usuários, o que é prejudicial para o projeto como um todo.

Berander (2004) afirma que não importa se o produto é considerado completo pelos desenvolvedores, se um conjunto “errado” de requisitos foi implementado e os usuários resistem a sua utilização.

### 3.2.7 Implantação do software

Caso nenhuma versão tenha sido entregue anteriormente, nesta etapa a equipe responsável realiza a implantação do software e desta forma inicia-se a utilização.

### 3.2.8 Avaliação de desempenho do software

Depois que o software já está em uso, é possível realizar a avaliação de desempenho em relação aos requisitos. Nesta etapa, os usuários fornecem seus julgamentos para cada requisito do software em relação ao desempenho do software em uso.

**a. Desenvolvimento do questionário de avaliação de desempenho**

Para a avaliação de desempenho recomenda-se a utilização da escala contínua e a análise importância-desempenho (avaliação de desempenho). Desta forma, assim como no estudo realizado por Hudson *et al*. (2004), as avaliações de importância e desempenho da IPA são feitas em momentos distintos.

Os questionários são compostos pela lista de requisitos utilizada para priorização. Para a avaliação de desempenho da IPA, os usuários fornecem seus julgamentos utilizando uma escala de Likert de 5 pontos. Neste caso, o valor 1 refere-se a um requisito com desempenho “Muito Ruim” e o valor 5, a um requisito com desempenho “Muito Bom”. O quadro 3.2 apresenta a escala de julgamento de desempenho.

| **Grau de Desempenho** | | | | |
|---|---|---|---|---|
| **Muito Ruim** | **Ruim** | **Neutro** | **Bom** | **Muito Bom** |
| **1** | **2** | **3** | **4** | **5** |

Quadro 3.2 - Escala de julgamento de desempenho.

A escala contínua é usada em conjunto com a avaliação de desempenho da IPA. Sugere-se a utilização da escala contínua e não o método dos 100 pontos porque nesta etapa da metodologia a avaliação não deve ser relativa. Além disso, optou-se pela escala contínua devido a sua facilidade de construção e a simplicidade de aplicação. Em relação à dificuldade de identificação dos valores atribuídos pelos avaliadores, como descrito por Malhotra (2006), acredita-se que dependendo da quantidade de avaliadores essa característica não se constitui como um impedimento à utilização da escala.

A figura 3.2 apresenta um exemplo de escala contínua para um requisito. Neste caso tem-se uma escala de 0 a 100. O avaliador deve marcar uma posição da

escala de acordo com a sua percepção de desempenho do requisito no software ou módulo que está sendo avaliado.

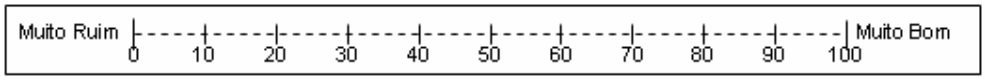


Figura 3.2- Exemplo de escala contínua.

**b. Aplicação do questionário de avaliação de desempenho**

O questionário de avaliação de desempenho deve ser aplicado aos usuários do software. Os usuários podem ser ou não os mesmos que participaram da etapa de priorização, mas é desejável que tenham experiência no uso do software para que sejam capazes de avaliar o desempenho deste à luz dos requisitos.

Nesta etapa é necessário buscar uma forma de identificar a experiência dos usuários em relação ao software. Caso dados de acessos dos usuários ao software estejam disponíveis, sugere-se a utilização da *Análise dos Quartis* para classificar os usuários em relação à quantidade de acessos ou ações realizadas no software.

Segundo Freitas *et al.* (2006) o *Quartil* é uma medida de tendência central. Os referidos autores utilizaram o *Quartil* para definir prioridades para os itens durante uma avaliação de qualidade de serviços de TI.

Assim como na etapa de priorização, o questionário deve ser aplicado individualmente.

### 3.2.9 Tabulação dos dados para avaliação de desempenho

Após a aplicação dos questionários de desempenho é necessário fazer a tabulação dos dados. A partir da tabulação os requisitos que possuem baixo desempenho são identificados.

**a. Resultados da Análise Importância-desempenho**

Para a análise de desempenho da IPA é necessário calcular as médias de desempenho para os requisitos. A equação 3.3 representa o cálculo da média de desempenho para o requisito *r*.

$$\mu(GD)_r = \frac{\sum_{j=1}^{n} (GD)_{r,j}}{n} \quad (3.3)$$

Onde:

$\mu(GD)_r$: Grau de Desempenho médio para o requisito r.

$(GD)_{r,j}$: Grau de Desempenho para o requisito r, segundo o avaliador j.

Para o caso da IPA, os resultados provenientes das avaliações de importância e desempenho devem ser analisados de forma conjunta para a identificação dos requisitos críticos. É importante realizar essa análise de forma conjunta porque os requisitos de baixo desempenho e pouca importância não são considerados tão graves quanto os de baixo desempenho e grande importância.

Por esse motivo, é preciso criar a matriz de análise importância-desempenho que possibilita a visualização dos itens onde as melhorias são mais urgentes de acordo com o quadrante onde esses itens são situados.

Para a criação da matriz, é necessário identificar o ponto onde eixos de importância e desempenho se interceptam. De acordo com Magal e Levenburg (2005) e Jesus (2005), esse ponto pode ser definido com base nas médias globais de importância e desempenho. As equações para cálculo das médias globais de importância e desempenho são representadas nas equações 3.4 e 3.5, respectivamente.

$$\mu(GI) = \frac{\sum_{r=1}^{m} \mu(GI)_r}{m} \quad (3.4)$$

$$\mu(GD) = \frac{\sum_{r=1}^{m} \mu(GD)_r}{m} \quad (3.5)$$

Após a definição dos eixos é possível inserir na matriz todos os requisitos que fazem parte da avaliação. Para identificar em qual posição deve ser inserido cada

requisito, é necessário utilizar as médias de importância e desempenho de cada requisito conforme representado nas equações 3.1 e 3.3.

**b. Resultados da escala contínua**

Os resultados obtidos através da escala contínua também devem ser utilizados para a avaliação de desempenho. Para isso, inicialmente o pesquisador identifica nos questionários respondidos, a pontuação atribuída por cada usuário a cada requisito. Em seguida, calcula-se a média dos valores identificados na escala contínua, equação 3.6 representa o cálculo dessa média.

$$\mu(V_r) = \frac{\sum_{i=1}^{n} (V)_{ij}}{n} \qquad (3.6)$$

Onde:

$\mu(V_r)$: valor médio de desempenho para o requisito *r*.

$n$: quantidade de usuários que avaliaram o desempenho do requisito por meio da escala contínua.

As médias de desempenho devem ser calculadas para todos os requisitos por meio das equações 3.3 e 3.6.

Como a metodologia propõe a utilização da escala contínua e a avaliação de desempenho da IPA, são obtidos dois resultados de desempenho. Deve-se observar se os resultados são coincidentes ou divergentes. Caso exista divergência dos resultados, deve-se aprofundar o estudo para os requisitos que tiveram pior desempenho para os dois métodos.

Sendo assim, para a melhoria da qualidade do software e satisfação dos usuários é necessário buscar elevar o desempenho iniciando pelos requisitos mais críticos. Os requisitos críticos para os resultados obtidos por meio da escala contínua são os que tiveram baixo valor médio de desempenho (equação 3.6) e para os resultados da IPA são os que ficaram situados no quadrante referente à grande importância e baixo desempenho.

### 3.2.10 Definição e execução das melhorias para os requisitos críticos

Nesta etapa recomenda-se a utilização da ferramenta ~~o~~ 5W1H para a identificação das possibilidades de melhoria para os requisitos críticos. Segundo Dantas *et al*. (2005) o 5W1H inclui as seguintes informações: *Who* (quem), *When* (quando), *Where* (onde), *Why* (por quê), *What* (o que), *How* (como). Para os autores, o 5W1H pode ser usado para dar suporte ao desenvolvimento de software possibilitando o controle de informações relativas ao processo desenvolvimento, principalmente a respeito das alterações realizadas no software.

Para a identificação das informações que compõem o 5W1H Dantas *et al.* (2005) utilizaram como fonte de dados um sistema de controle de versões. Esse tipo de sistema funciona como uma ferramenta para o desenvolvimento de software permitindo que os desenvolvedores tenham controle sobre as alterações realizadas no software e suas versões.

Caso não seja possível utilizar um sistema de controle de versões, deve-se realizar um estudo mais aprofundado juntos aos usuários. Neste caso o objetivo é investigar o motivo dos requisitos terem sido considerados críticos e o que pode ser feito para melhorá-los.

### 3.2.11 Realização de nova avaliação de desempenho

Caso seja desejado, novas avaliações de desempenho podem ser feitas após a melhoria dos requisitos críticos. Essa etapa da metodologia permite que se busque a melhoria contínua do software tendo como base os requisitos. A cada avaliação de desempenho, caso novos erros sejam identificados no software, é necessário voltar a etapa de desenvolvimento de software para fazer os ajustes necessários.

# CAPÍTULO IV

# ESTUDO DE CASO

Neste capítulo é apresentado um estudo de caso realizado para verificar a viabilidade de aplicação da metodologia proposta e os resultados possíveis a partir de sua utilização.

## 4.1 Execução das etapas da metodologia

Para especificar como o estudo de caso foi realizado, descreve-se como as etapas da metodologia apresentadas no capítulo III foram executadas.

### 4.1.1 Identificação de uma equipe de desenvolvimento que esteja iniciando um projeto

A metodologia foi aplicada em um projeto de desenvolvimento de software de uma fundação situada no município de Campos dos Goytacazes, no Estado do Rio de Janeiro. Esta fundação é mantenedora de algumas unidades de saúde vinculadas ao Sistema Único de Saúde (SUS). O objetivo deste projeto é desenvolver um software que apóie as atividades relativas à gestão de Recursos Humanos (RH) da referida fundação.

Os setores envolvidos no projeto de desenvolvimento possuem necessidades específicas em relação ao software e após algumas tentativas de aquisição de pacotes fechados de software, chegou-se à conclusão que essas necessidades específicas justificam o investimento em desenvolvimento.

Desta forma o software desenvolvido pelo projeto em questão é classificado como software por encomenda em relação à forma de chegada no mercado.

Conforme descrito no capítulo III essa é uma característica que deve ser considerada para a aplicação da metodologia.

O Anexo A desta dissertação apresenta uma breve descrição do software desenvolvido pelo projeto. O software será composto por alguns módulos, entre eles o de Departamento de Pessoal (DP). O estudo de caso foi realizado no desenvolvimento deste módulo.

### 4.1.2 Elicitação de requisitos

Os requisitos do software para atender ao setor de Departamento de Pessoal foram identificados por meio de quatro técnicas descritas no referencial teórico desta dissertação:

- entrevista não estruturada;
- entrevista estruturada;
- prototipagem rápida;
- análise de cenário.

As quatro técnicas listadas acima foram consideradas adequadas pela equipe de desenvolvimento, pois permitiram o entendimento do domínio onde o software se insere.

A elicitação de requisitos teve início em 2008 e vem sendo atualizada desde então. Foram realizadas entrevistas não estruturadas com os clientes do software. Através dessas entrevistas, os analistas puderam identificar algumas das funções básicas que o software deve possuir. Em seguida, entrevistas estruturadas foram realizadas com o objetivo de refinar as análises anteriores e resolver pequenas inconsistências.

Ao longo do tempo algumas trocas de funcionários aconteceram, tanto no setor desenvolvedor do software quanto nos setores clientes. Essas mudanças geraram a necessidades de novas entrevistas.

As técnicas de prototipagem e análise de cenário também foram aplicadas. A técnica de prototipagem permitiu que os clientes tivessem uma percepção mais aproximada de como o software estava sendo desenvolvido e este protótipo possibilitou a identificação da necessidade de alguns ajustes. A análise de cenário foi considerada interessante para facilitar a compreensão das necessidades por parte de analistas e programadores do software em casos específicos.

A partir da aplicação das técnicas de elicitação utilizadas, a lista inicial de requisitos foi identificada. A lista pode ser observada no quadro 4.1.

| **REQUISITOS** |
|---|
| Controle de Férias |
| Registro de informações de escalas e horários de trabalho |
| Emissão de relatório de freqüência |
| Controle das informações pessoais dos funcionários. |
| Registro de freqüência |
| Controle de Afastamentos como Licenças Médicas e INSS |
| Registro de deficiências apresentadas pelos funcionários |
| Emissão de crachás |
| Controle de Alocação de funcionários em unidade e setores |
| Solicitação e entrega de declarações |
| Manutenção de informações relativas à convocação |
| Entrada e saída de carteiras de trabalho e previdência social |

Quadro 4.1 - Lista de requisitos identificados para o software.

### 4.1.3 Priorização dos requisitos

Nesta etapa a equipe de desenvolvimento realizou a priorização da lista de requisitos obtida. Para que isso fosse possível, foi necessário desenvolver e aplicar o questionário de priorização.

**a. Desenvolvimento do questionário de priorização**

Inicialmente os requisitos foram agrupados em relação ao contexto de aplicação. Como exemplo, requisitos relativos ao controle de freqüência dos funcionários foram inseridos em um mesmo grupo. O quadro 4.2 apresenta a lista de requisitos agrupada.

| | REQUISITOS |
|---|---|
| **1** | **Controle de Freqüência dos Funcionários** |
| 1.1 | Registro de freqüência |
| 1.2 | Registro de informações de escalas e horários de trabalho |
| 1.3 | Emissão de relatório de freqüência |
| **2** | **Manutenção dos dados pessoais** |
| 2.1 | Controle das informações pessoais dos funcionários. |
| 2.2 | Registro de deficiências apresentadas pelos funcionários |
| **3** | **Manutenção dos dados funcionais** |
| 3.1 | Emissão de crachás |
| 3.2 | Manutenção de informações relativas à convocação |
| 3.3 | Controle de Alocação de funcionários em unidade e setores |
| **4** | **Manutenção dos Afastamentos** |
| 4.1 | Controle de Férias |
| 4.2 | Controle de Afastamentos como Licenças Médicas e INSS |
| **5** | **Movimentação de Documentos** |
| 5.1 | Entrada e saída de carteiras de trabalho e previdência social |
| 5.2 | Solicitação e entrega de declarações |

Quadro 4.2 - Lista de requisitos agrupados.

A partir da lista de requisitos agrupados, o questionário de priorização foi criado utilizando os métodos e as escalas descritos no capítulo anterior. O questionário desenvolvido encontra-se no Apêndice A desta dissertação.

**b. Aplicação do questionário de priorização**

Aplicação do questionário foi feita em outubro de 2008 em horários diferenciados para que fosse possível obter as respostas dos usuários que atuam em turnos distintos.

Optou-se por aplicar o questionário a todos os usuários presentes nos dias e turnos de aplicação. A informação a respeito da experiência de cada usuário foi obtida através de um item inserido no questionário.

O grupo de participantes da pesquisa se constituiu de dez futuros usuários do software. O questionário foi aplicado no setor de trabalho dos usuários com acompanhamento de um responsável pela pesquisa. Assim, buscou-se evitar trocas de opiniões entre os participantes.

### 4.1.4 Tabulação dos dados para Priorização

Os dados dos questionários obtidos foram tabulados para que as análises pudessem ser realizadas. Como o questionário possibilita a avaliação por meio de dois métodos, são obtidos resultados para esses dois métodos.

**a. Resultados da Análise Importância-desempenho**

A tabela 4.1 apresenta a freqüência dos itens da escala de julgamentos da avaliação de importância da IPA.

Tabela 4.1- Tabela de freqüências dos itens da escala de julgamentos, avaliação de importância (IPA)

| | **Requisitos** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | | | **2** | | **3** | | | **4** | | **5** | |
| **Grau de importância** | **1.1** | **1.2** | **1.3** | **2.1** | **2.2** | **3.1** | **3.2** | **3.3** | **4.1** | **4.2** | **5.1** | **5.2** |
| 1- Nada Importante | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2- Pouco Importante | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3- Neutro | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4- Importante | 2 | 6 | 4 | 3 | 3 | 5 | 5 | 5 | 6 | 5 | 7 | 8 |
| 5- Muito Importante | 8 | 4 | 6 | 7 | 5 | 5 | 5 | 5 | 4 | 5 | 3 | 2 |
| **TOTAL:** | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

Os dados apresentados na tabela 4.1 mostram que 10 usuários avaliaram os 12 requisitos presentes na etapa de priorização dos requisitos. Para a análise de importância (priorização) foi necessário calcular as médias de importância para cada requisito (equação 3.1). A tabela 4.2 fornece os dados relativos a essas médias.

Tabela 4.2- Tabela de médias de importância dos requisitos

| | Requisitos | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | | | 2 | | 3 | | | 4 | | 5 | |
| **Grau de Importância** | **1.1** | **1.2** | **1.3** | **2.1** | **2.2** | **3.1** | **3.2** | **3.3** | **4.1** | **4.2** | **5.1** | **5.2** |
| 1- Nada Importante | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2- Pouco Importante | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3- Neutro | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4- Importante | 8 | 24 | 16 | 12 | 12 | 20 | 20 | 20 | 24 | 20 | 28 | 32 |
| 5- Muito Importante | 40 | 20 | 30 | 35 | 25 | 25 | 25 | 25 | 20 | 25 | 15 | 10 |
| **TOTAL** | 48 | 44 | 46 | 47 | 43 | 45 | 45 | 45 | 44 | 45 | 43 | 42 |
| **MÉDIA** | 4,80 | 4,40 | 4,60 | 4,70 | 4,30 | 4,50 | 4,50 | 4,50 | 4,40 | 4,50 | 4,30 | 4,20 |
| **Ordem** | **1** | **8** | **3** | **2** | **10** | **4** | **4** | **4** | **8** | **4** | **10** | **12** |

Para melhor compreensão, apresentam-se os cálculos relacionados ao requisito 1.1: dois usuários o avaliaram com grau de importância 4, obtendo o valor 8 como resultado do produto; oito pessoas avaliaram este requisito com grau de importância 5, obtendo o valor 40 como resultado do produto. A soma total para este requisito é 48, correspondendo à soma dos produtos obtidos (8 + 40). Após dividir este valor pela quantidade de respondentes (10), obteve-se a importância média deste requisito.

A partir das médias de importância obteve-se a ordem de importância (prioridade) dos requisitos. O requisito 1.1 foi o que obteve a maior média de importância por isso a ordem obtida foi 1, de acordo com a IPA este requisito destacou-se como o mais importante.

Os dados apresentados nas tabelas 4.1 e 4.2 mostram que existe uma tendência relacionada à aplicação da IPA: os avaliadores tendem a atribuir altos valores de importância para os itens avaliados. Esta tendência também foi notada em outros trabalhos como pode ser observado em Ainin e Hisham (2008); Duke (2002); e em Jesus (2005).

**b. Resultados do método dos 100 pontos**

A tabela 4.3 apresenta os resultados obtidos por meio do método dos 100 pontos para cada requisito identificado como necessário. Trata-se do cálculo da média das pontuações para cada requisito conforme representado pela equação 3.2.

Tabela 4.3- Tabela de resultados da distribuição dos 100 pontos.

| | Requisitos | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | | | 2 | | 3 | | | 4 | | 5 | |
| | 1.1 | 1.2 | 1.3 | 2.1 | 2.2 | 3.1 | 3.2 | 3.3 | 4.1 | 4.2 | 5.1 | 5.2 |
| **Soma** | 146,3 | 89,3 | 73,3 | 89,3 | 60,3 | 67,3 | 73,3 | 77,3 | 90,3 | 100,3 | 68,8 | 63,8 |
| **Média** | 14,6 | 8,9 | 7,3 | 8,9 | 6,0 | 6,7 | 7,3 | 7,7 | 9,0 | 10,0 | 6,9 | 6,4 |
| **Ordem** | 1 | 4 | 7 | 4 | 12 | 10 | 7 | 6 | 3 | 2 | 9 | 11 |

O requisito com a maior média é o 1.1, desta forma assim como o resultado obtido por meio da IPA, esse foi o requisito com maior prioridade.

O quadro 4.3 lista os requisitos priorizados de acordo com os resultados apresentados nas tabelas anteriores.

| | **Avaliação de Importância (IPA)** | **Método dos 100 pontos** |
|---|---|---|
| **1** | 1.1 Freqüência | 1.1 Freqüência |
| **2** | 2.1 Informações pessoais | 4.2 Controle de Afastamentos |
| **3** | 1.3 Relatório de freqüência | 4.1 Controle de Férias |
| **4** | 3.1 Crachás | 2.1 Informações pessoais |
| | 3.2 Informações sobre convocação | |
| | 3.3 Alocação de funcionários | 1.2 Escalas e horários de trabalho |
| | 4.2 Controle de Afastamentos | |
| **5** | 4.1 Controle de Férias | 3.3 Alocação de funcionários |
| | 1.2 Escalas e horários de trabalho | |
| **6** | 2.2 Registro de deficiências | 3.2 Informações sobre convocação |
| | 5.1 Entrada e saída de carteiras de trabalho | 1.3 Relatório de freqüência |
| **7** | 5.2 Solicitação e entrega de declarações | 5.1 Entrada e saída de carteiras de trabalho |
| **8** | | 3.1 Crachás |
| **9** | | 5.2 Solicitação e entrega de declarações |
| **10** | | 2.2 Registro de deficiências |

Quadro 4.3- Lista de requisitos priorizados

A utilização de dois métodos para priorização resultou em duas listas distintas de requisitos priorizados, conforme pode ser observado no quadro 4.3. A partir desse resultado é possível afirmar que o requisito mais relevante de acordo com a percepção dos usuários está relacionado à necessidade de controle de freqüência dos funcionários da instituição.

Para os demais requisitos houve variação em relação à prioridade de cada requisito para os dois métodos aplicados. Isso pode ser explicado pelo fato de alguns usuários terem julgado muitos itens como “Muito Importante” na escala de importância. No entanto, no método dos 100 pontos esses mesmos usuários forneceram julgamentos diferenciados para os atributos.

Uma observação que deve ser feita refere-se ao esforço cognitivo necessário para o julgamento através da utilização dos dois métodos. O esforço necessário para o julgamento através do método dos 100 pontos pode ser considerado maior do que o necessário para a escolha de um item da escala de graus de importância. Neste caso, onde há divergência, aconselha-se a utilização do resultado do método dos 100 pontos.

A priorização pode ser considerada um grande benefício da metodologia empregada porque os desenvolvedores puderam obter um direcionamento para a realização das etapas subseqüentes do processo de desenvolvimento de software.

### 4.1.5 Desenvolvimento do software

O desenvolvimento foi realizado tendo como referência a lista de requisitos priorizados. A evolução do processo de desenvolvimento foi acompanhada pelos usuários que validaram as funcionalidades, sugeriram melhorias e identificaram possíveis erros.

A metodologia de desenvolvimento utilizada pelo projeto em questão é o desenvolvimento iterativo incremental descrito no referencial teórico. As características desse tipo de metodologia de desenvolvimento possibilitam que entregas do software sejam feitas à medida que o desenvolvimento prossegue. Desta forma os usuários tiveram acesso aos requisitos desenvolvidos gradativamente.

Mesmo antes da finalização do processo de desenvolvimento, o software despertou o interesse de outras organizações que possuem características semelhantes à fundação para a qual ele está sendo desenvolvido. Por esse motivo, pretende-se distribuí-lo como software livre, o que possibilitará que outras pessoas, além da equipe inicialmente definida, participem do desenvolvimento.

### 4.1.6 Verificação do desenvolvimento dos requisitos prioritários

De acordo com a metodologia proposta após o desenvolvimento do software ou após a realização de várias iterações, é necessário verificar através de um *checklist* se os requisitos prioritários foram desenvolvidos. Apenas após essa verificação deve-se implantar e avaliar o desempenho do software.

Para a construção do *checklist* utilizou-se como referência a lista de requisitos priorizados resultante do método dos 100 pontos. O *checklist* é apresentado no quadro 4.4.

| | **Requisitos Priorizados** | **Desenvolvido** | | |
|---|---|---|---|---|
| | | **SIM** | **NAO** | **MOTIVO (CASO NÃO)** |
| **1** | 1.1 Freqüência | X | | |
| **2** | 4.2 Controle de Afastamentos | X | | |
| **3** | 4.1 Controle de Férias | X | | |
| **4** | 2.1 Informações pessoais | X | | |
| | 1.2 Escalas e horários de trabalho | X | | |
| **5** | 3.3 Alocação de funcionários | X | | |
| **6** | 3.2 Informações sobre convocação | X | | |
| | 1.3 Relatório de freqüência | X | | |
| **7** | 5.1 Entrada e saída de carteiras de trabalho | | X | Decisão gerencial em conjunto com usuários |
| **8** | 3.1 Crachás | X | | |
| **9** | 5.2 Solicitação e entrega de declarações | | X | Decisão gerencial em conjunto com usuários |
| **10** | 2.2 Registro de deficiências | X | | |

Quadro 4.4 - *Checklist* dos requisitos.

Ao observar o quadro 4.4 verifica-se que a maioria dos requisitos foi desenvolvida, com exceção dos requisitos 5.1 (*Entrada e saída de carteiras de trabalho*) e 5.2 (*Solicitação e entrega de declarações*). Esses requisitos não foram desenvolvidos devido a uma decisão gerencial. Os gestores perceberam, através de reuniões com os usuários, que as necessidades relativas a esses requisitos não está situada no contexto de aplicação do software. Trata-se de necessidades gerais relativas a vários setores e não só dos setores relacionados aos recursos humanos.

Por isso os requisitos 5.1 e 5.2 não passaram pelas etapas seguintes da metodologia. É importante destacar que esses requisitos não haviam sido avaliados como prioritários pelos usuários na etapa de priorização.

### 4.1.7 Implantação do software

As atividades iniciais necessárias para a implantação foram executadas, como a preparação do banco de dados, por exemplo. Uma dificuldade encontrada nesta etapa refere-se ao esforço inicial necessário para colocar o software em funcionamento. De uma forma geral, é necessário realizar a inserção de um grande volume de dados, o que requer pessoas aptas a executarem esta função.

Conforme já descrito, o software foi desenvolvido tendo como referência o processo iterativo incremental. Por isso a implantação não ocorreu em um único momento, mas sim gradativamente à medida que as entregas foram feitas.

A primeira versão do software entregue para os usuários pronta para ser utilizada foi disponibilizada no dia 23 de outubro de 2008. No entanto, os usuários começaram a utilizar o sistema no dia 12 de janeiro de 2009, a justificativa para esse atraso está associada à limitação de funcionários no setor usuário.

No período de outubro de 2008 e outubro de 2009 ocorreram aproximadamente 150 atualizações no software, destas, cerca de 64 atualizações foram externas.

### 4.1.8 Avaliação de desempenho do software

Após as etapas de implantação do software os usuários passaram a utilizá-lo. A partir da experiência que foram adquirindo em relação ao uso do software, tornaram-se aptos a avaliar seu desempenho.

A avaliação de desempenho, assim como a avaliação de importância, é composta por duas etapas: o desenvolvimento do questionário de avaliação de desempenho e a aplicação desse questionário.

**a. Desenvolvimento do questionário de avaliação de desempenho**

De acordo com a metodologia proposta para a avaliação de desempenho, é necessário desenvolver o questionário para o uso da IPA (avaliação de desempenho) e da escala contínua.

A lista de requisitos da etapa de avaliação de desempenho é semelhante à utilizada na etapa de priorização, a diferença encontra-se na exclusão dos requisitos 5.1 e 5.2. O questionário utilizado para a avaliação de desempenho encontra-se no Apêndice B desta dissertação.

**b. Aplicação do questionário de avaliação de desempenho**

O questionário de avaliação de desempenho foi aplicado no mês de outubro de 2009, ou seja, um ano após a aplicação dos questionários de priorização. Durante o preenchimento os usuários avaliaram cada requisito utilizando a escala de Likert de 5 pontos da avaliação de desempenho da IPA e a escala contínua, cujos valores variam de 0 a 100.

Participaram da etapa de avaliação de desempenho 14 usuários do software. Desses 14 usuários, 4 participaram também da etapa de priorização. Essa diferença se deve ao grande número de transferências de funcionários pelos setores da instituição.

O tempo médio de experiência dos usuários que responderam o questionário de avaliação de desempenho é de 2 anos e 4 meses. Essa é uma característica importante de ser considerada, pois para avaliar o desempenho é importante que os usuários tenham experiência na utilização do software.

Além do tempo médio de experiência, considera-se relevante verificar a quantidade de ações realizadas no software por cada usuário. Para obter essa informação foram considerados os dados relativos aos acessos dos usuários. Nos dados de acesso ficam registradas ações de inserção, exclusão e alteração, data e hora de ocorrência, qual usuário executou e de qual computador a ação foi realizada.

Tendo como referência os dados de acesso, optou-se por utilizar a *Análise dos Quartis* para classificar os usuários em relação à quantidade de ações realizadas. Neste caso os usuários que realizaram uma quantidade de ações menor do que o valor referente ao primeiro *Quartil* são os que executaram menos ações e possivelmente possuem menos experiência em relação ao software. As respostas

desses usuários não foram consideradas nas análises, por isso dos 14 questionários respondidos, 10 foram considerados.

A tabela 4.4 apresenta a classificação dos usuários por meio da *Análise dos Quartis.*

Tabela 4.4 Classificação dos usuários segundo quantidade de ações realizadas utilizando *Análise dos Quartis.*

| **Classificação dos usuários segundo quantidade de ações realizadas** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Freqüência Baixa | | | | Freqüência Moderada | | | Freqüência Alta | | | Freqüência Muito Alta | | | |
| U12 | U14 | U13 | U15 | U6 | U4 | U7 | U8 | U9 | U3 | U10 | U1 | U2 | U11 |
| 1 | 3 | 9 | 28 | 33 | 162 | 392 | 450 | 466 | 921 | 939 | 1919 | 2774 | 4084 |
| 1º Q = 29,25 | | | | 2º Q = 421 | | | 3º Q = 934,5 | | | | | | |

A figura 4.1 apresenta a quantidade de ações realizadas por cada usuário considerado na análise e por quinzena no período de 8 meses aproximadamente.

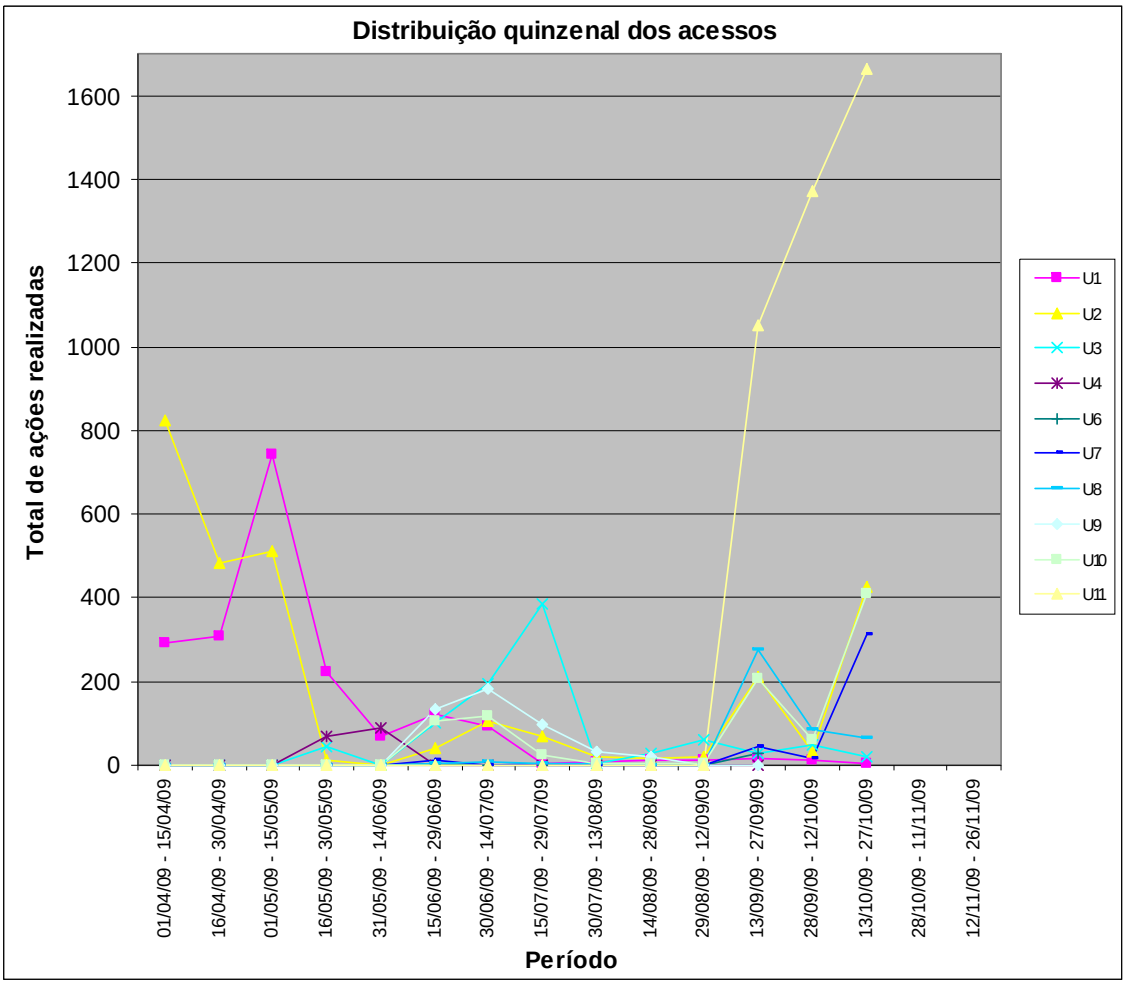


Figura 4.1– Gráfico de distribuição quinzenal de ações por usuário

Ao observar a figura 4.1 nota-se que o usuário 11 (U11), por exemplo, começou a utilizar o sistema no final do mês de agosto e a quantidade de ações desse usuário continuou crescendo.

Durante a aplicação do questionário alguns usuários relataram a dificuldade que encontram para obter os dados necessários para inserção no software. Estes usuários sabem que o software só funciona bem se os dados corretos forem inseridos. No entanto, em alguns casos as informações não estão disponíveis. Para obtê-las é necessário contar com a participação das unidades de saúde e setores envolvidos.

### 4.1.9 Tabulação dos dados para avaliação de desempenho

Os dados obtidos através dos questionários de desempenho foram tabulados e os resultados obtidos por meio da aplicação da IPA e da escala contínua são apresentados a seguir.

#### a. Resultados da Análise Importância-desempenho

A tabela 4.5 apresenta a freqüência dos itens da escala de julgamentos da avaliação de desempenho da IPA.

Tabela 4.5- Tabela de freqüências dos itens da escala de julgamentos, avaliação de desempenho (IPA)

| | **Requisitos** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | | | **2** | | **3** | | | **4** | |
| **Grau de desempenho** | **1.1** | **1.2** | **1.3** | **2.1** | **2.2** | **3.1** | **3.2** | **3.3** | **4.1** | **4.2** |
| 1- Muito Ruim | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2- Ruim | 0 | 2 | 0 | 0 | 0 | 3 | 2 | 0 | 1 | 1 |
| 3- Neutro | 1 | 2 | 3 | 1 | 4 | 2 | 3 | 1 | 2 | 4 |
| 4- Bom | 6 | 5 | 4 | 5 | 4 | 2 | 4 | 6 | 3 | 2 |
| 5- Muito Bom | 3 | 0 | 2 | 4 | 2 | 3 | 1 | 3 | 4 | 2 |
| **TOTAL:** | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

Conforme apresentado na tabela 4.5 os 10 usuários avaliaram os 10 requisitos presentes nesta etapa da avaliação. De acordo com o que foi descrito na metodologia para a análise de desempenho, é necessário calcular as médias de desempenho para cada requisito (equação 3.3). A tabela 4.6 fornece os dados relativos a essas médias.

Tabela 4.6– Médias de desempenho dos requisitos

| | **Requisitos** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | | | **2** | | **3** | | | **4** | |
| **Grau de desempenho** | **1.1** | **1.2** | **1.3** | **2.1** | **2.2** | **3.1** | **3.2** | **3.3** | **4.1** | **4.2** |
| 1- Muito Ruim | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2- Ruim | 0 | 4 | 0 | 0 | 0 | 6 | 4 | 0 | 2 | 2 |
| 3- Neutro | 3 | 6 | 9 | 3 | 12 | 6 | 9 | 3 | 6 | 12 |
| 4- Bom | 24 | 20 | 16 | 20 | 16 | 8 | 16 | 24 | 12 | 8 |
| 5- Muito Bom | 15 | 0 | 10 | 20 | 10 | 15 | 5 | 15 | 20 | 10 |
| **TOTAL:** | 42 | 31 | 36 | 43 | 38 | 35 | 34 | 42 | 40 | 33 |
| **MÉDIA** | 4,20 | 3,10 | 3,60 | 4,30 | 3,80 | 3,50 | 3,40 | 4,20 | 4,00 | 3,30 |
| **Ordem** | 2 | 10 | 6 | 1 | 5 | 7 | 8 | 2 | 4 | 9 |

Cada item da escala foi multiplicado pela freqüência de ocorrências desse item para um dado requisito. Em seguida calculou-se a soma das pontuações de cada requisito para a partir da soma calcular a média. Dessa forma, para o requisito 1.1: uma pessoa o avaliou com grau de desempenho 3, obtendo o valor 3 como resultado do produto; 6 usuários o avaliaram com grau de desempenho 4, obtendo o valor 24 como resultado do produto e 3 usuários avaliaram este requisito com grau de desempenho 5, obtendo o valor 15 como resultado do produto. A soma total para este requisito é 42, correspondendo à soma dos produtos obtidos (3 + 24 + 15). Após dividir este valor pela quantidade de respondentes (10), obteve-se o desempenho médio deste requisito.

A partir das médias de desempenho foi possível identificar a ordem de desempenho dos requisitos. O requisito 2.1 foi o que obteve a maior média de desempenho por isso a ordem obtida é 1. Os requisitos com as menores médias de desempenho foram os requisitos 1.2, 3.2 e 4.2, estes requisitos estão destacados na tabela 4.6.

No entanto, nesse momento ainda não é possível afirmar que esses sejam os requisitos mais críticos porque a IPA sugere que para identificar os itens críticos em uma avaliação, as análises de importância e desempenho devem ser feitas de forma conjunta.

Por isso, após realizar as avaliações de importância e desempenho, a matriz de análise importância-desempenho foi criada. Para a criação da matriz foi preciso calcular as médias globais de importância e desempenho (equações 3.4 e 3.5). Na pesquisa em questão o valor obtido para a média global de importância foi 4,52 e para a média global de desempenho foi 3,74.

Além das médias globais, as médias obtidas para cada requisito também são necessárias. A tabela 4.7 resume os valores obtidos, apresentando as médias de importância e desempenho para cada requisito.

Tabela 4.7– Médias de importância e desempenho para os requisitos

| **Requisitos** | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | | | | | | **2** | | | | **3** | | | | | | **4** | | | |
| **1.1** | | **1.2** | | **1.3** | | **2.1** | | **2.2** | | **3.1** | | **3.2** | | **3.3** | | **4.1** | | **4.2** | |
| **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** | **D** | **I** |
| 4,20 | 4,80 | 3,10 | 4,40 | 3,60 | 4,60 | 4,30 | 4,70 | 3,80 | 4,30 | 3,50 | 4,50 | 3,40 | 4,50 | 4,20 | 4,50 | 4,00 | 4,40 | 3,30 | 4,50 |

A figura 4.2 apresenta a matriz de análise importância e desempenho.

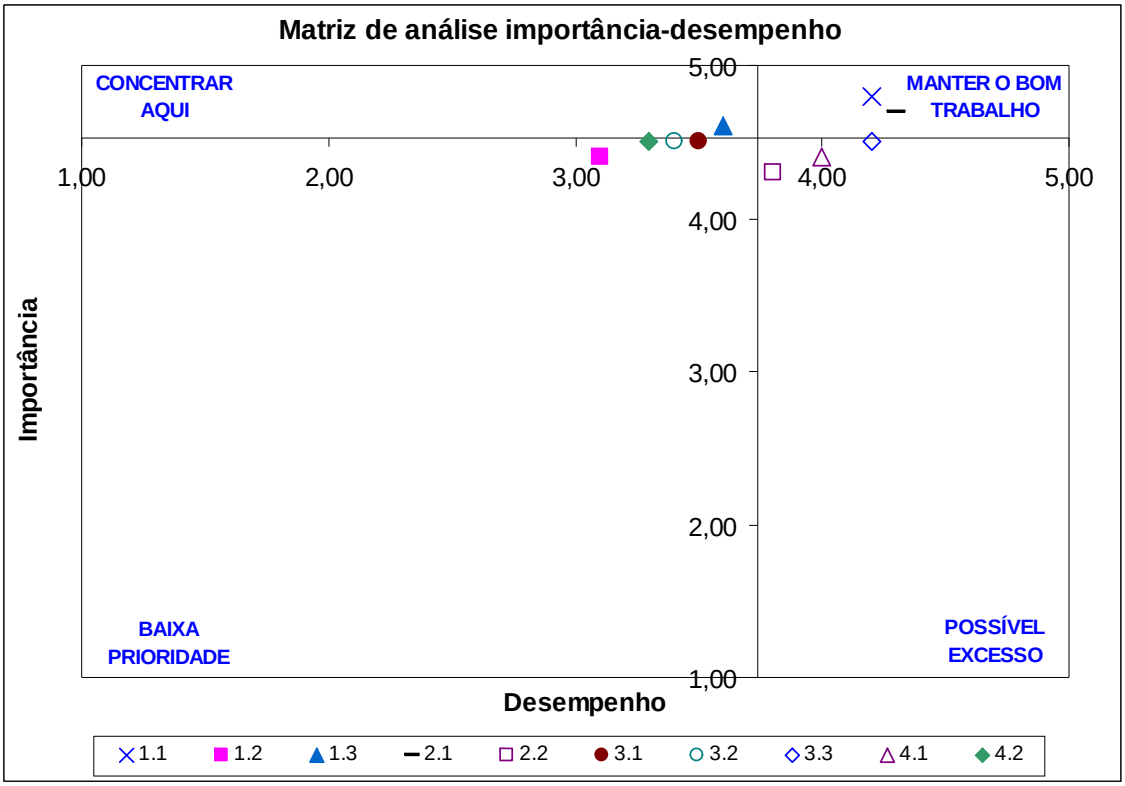


Figura 4.2- Matriz de análise importância-desempenho.

O ponto de interseção do gráfico apresentado na figura 4.2 foi definido com base nas médias globais de importância e desempenho anteriormente calculadas.

Para identificar o ponto onde cada requisito se situa, utilizaram-se as médias de importância e desempenho obtidas. Para o requisito 1.3, por exemplo, as médias obtidas foram, 4,6 e 3,6 para importância e desempenho, respectivamente, conforme pode ser verificado na tabela 4.7.

A observação da matriz permitiu que a análise de alguns requisitos fosse realizada mais facilmente enquanto que para outros foi necessário verificar também as médias obtidas (tabela 4.7). Esse é o caso dos requisitos que estão posicionados sobre os eixos da matriz.

Os requisitos 1.1 e 2.1 estão posicionados no quadrante “Manter o bom trabalho”. Trata-se de requisitos prioritários mas que estão tendo bom desempenho de acordo com a percepção dos usuários, por isso ações de melhoria nesse conjunto de requisitos não são prioritárias.

Os requisitos 2.2, 3.3 e 4.1 estão situados no quadrante “Possível Excesso”, relativo aos requisitos menos prioritários na avaliação de importância, mas que

tiveram bom desempenho. A localização nesse quadrante indica que recursos podem estar sendo alocados aos itens de forma indevida, ou seja, os mesmos recursos poderiam ser usados para melhorar o desempenho de itens prioritários. No entanto, nesse estudo de caso, os requisitos já foram desenvolvidos e os recursos já foram empregados. Uma observação que pode ser feita em relação a esses requisitos refere-se ao fato de que as solicitações feitas pelos usuários, associadas a esses requisitos não são prioritárias.

As posições dos requisitos 1.2, 3.1, 3.2 e 4.2 na matriz sugerem que esses requisitos possuem baixa prioridade porque quando comparados com os demais possuem menor importância e baixo desempenho.

Apenas o requisito 1.3 está posicionado no quadrante “Concentrar Aqui”. Esse é o quadrante onde as necessidades de melhoria são prioritárias. No entanto, é importante observar que outros requisitos estão situados bem próximos a esse quadrante. Este é o caso dos requisitos 1.2, 3.1, 3.2 e 4.2, para estes requisitos um pequeno aumento na importância também os deslocaria para o quadrante de ações prioritárias.

Neste caso, devido a esta proximidade, aconselha-se que esses requisitos sejam observados cuidadosamente, assim como o requisito 1.3 e que de acordo com os resultados obtidos por meio da escala contínua ações de melhoria sejam identificadas para esses requisitos.

**b. Resultados da escala contínua**

Após a análise de desempenho da IPA calculou-se a média dos valores obtidos por meio da escala contínua para os requisitos, conforme representado pela equação 3.6. A tabela 4.8 apresenta os resultados obtidos.

Tabela 4.8– Médias de desempenho obtidas pela escala contínua.

| | **Requisitos** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | | | **2** | | **3** | | | **4** | |
| | **1.1** | **1.2** | **1.3** | **2.1** | **2.2** | **3.1** | **3.2** | **3.3** | **4.1** | **4.2** |
| **Soma** | 688 | 581 | 691 | 737 | 706 | 680 | 626 | 769 | 741 | 566 |
| **Média** | 68,8 | 58,1 | 69,1 | 73,7 | 70,6 | 68 | 62,6 | 76,9 | 74,1 | 56,6 |
| **Ordem** | 6 | 9 | 5 | 3 | 4 | 7 | 8 | 1 | 2 | 10 |

Como pode ser observado na tabela 4.8, quanto menor a média pior o desempenho do requisito. Retomando os resultados obtidos por meio da matriz de análise importância-desempenho observa-se que os requisitos 1.2, 3.1, 3.2 e 4.2, situados próximos ao quadrante “Concentrar Aqui” também tiveram as piores médias por meio da escala contínua.

No entanto, o requisito 1.3 que de acordo com a matriz, necessita de ações urgentes, na escala contínua é o quinto em ordem de prioridade. Apesar desse resultado, para planejamento das ações de melhoria é aconselhável considerar os 5 requisitos como críticos para a melhoria do software.

### 4.1.10 Definição e execução das melhorias para os requisitos críticos

Nesta etapa utilizou-se o 5W1H para identificar as ações de melhoria possíveis para os requisitos críticos. A exemplo do método utilizado por Dantas *et al.* (2005), utilizou-se um sistema de controle de versões para a coleta das informações necessárias para compor o 5W1H.

No projeto em questão usuários e desenvolvedores podem identificar possibilidades de melhoria e correção de erros para o software. No entanto, apenas os desenvolvedores inserem os dados no sistema de controle de versões. Quando um usuário identifica uma necessidade de alteração informa essa necessidade a um desenvolvedor que insere no sistema de controle de versões.

Os quadros a seguir apresentam as informações relativas ao 5W1H para cada um dos requisitos críticos. A coluna “o que” representa possibilidades de melhoria ou correção de erros identificados por desenvolvedores ou usuários. A coluna “por que” justifica a necessidade de implementação da alteração identificada na coluna “o que”. A terceira coluna, “quando”, identifica a data de identificação da necessidade de alteração. A coluna “quem” indica quem identificou a ação a ser realizada, neste caso identifica-se os desenvolvedores pela letra “D” e os usuários por “U”. A última coluna, “como”, descreve de forma resumida como a alteração deve ser feita, classificando-a.

| Requisito 1.2 | | | | |
|---|---|---|---|---|
| **O QUE (*What*)?** | **POR QUE (*Why*)?** | **QUANDO (*When*)?** | **QUEM (*Who*)? (IDENTIFICOU)** | **COMO (*How*)?** |
| Implementar Escala para os funcionários plantonistas. | Porque a escala é o único jeito de saber quando um funcionário plantonista deve trabalhar. | 08/09/09 | D3 | Implementação de uma função no sistema |
| Colocar no sistema uma possibilidade de abonar a redução de carga horária. | Devido a inconsistência entre cargas horárias. | 23/09/09 | U16 | Implementação de uma melhoria no sistema |

Quadro 4.5– 5W1H para o requisito 1.2.

De acordo com o quadro 4.5, para o requisito 1.2 foram identificadas apenas duas possibilidades de melhorias, uma foi identificada por um desenvolvedor e outra por um usuário.

| Requisito 1.3 | | | | |
|---|---|---|---|---|
| **O QUE (*What*)?** | **POR QUE (*Why*)?** | **QUANDO (*When*)?** | **QUEM (*Who*)? (IDENTIFICOU)** | **COMO (*How*)?** |
| Os relatórios de freqüência devem possuir uma linha para assinatura do funcionário. | O funcionário deve assinar para confirmar as informações apresentadas. | 28/04/09 | U6 | Implementação de uma melhoria no sistema |
| Relatório de freqüência, permitir impressão sem informar setor. | Para facilitar a impressão pelos usuários do DP. | 22/05/09 | D2 | Implementação de uma melhoria no sistema |
| Implementar Tela consolidada de freqüência. | Possibilitar a visualização do cálculo de horas. | 26/05/09 | U16 | Implementação de uma função no sistema |
| Restrição de acesso aos relatórios de freqüência de acordo com o setor. | Um usuário não pode visualizar dados de funcionários que não estão sob sua responsabilidade. | 05/06/09 | U16 | Implementação de uma melhoria no sistema |
| Aumentar campo data para impressão dos relatórios de freqüência. | Usuários não conseguiam informar os dados corretamente. | 18/06/09 | D3 | Correção de um erro de implementação do sistema |
| Relatório de freqüência mostrar dia da semana. | Facilitar a visualização dos dados. | 24/06/09 | D4 | Implementação de uma melhoria no sistema |
| Colocar soma de horas trabalhadas nos relatórios de freqüência. | Para verificação da carga horária cumprida. | 19/08/09 | D2 | Implementação de uma função no sistema |

Quadro 4.6 - 5W1H para o requisito 1.3.

O quadro 4.6 mostra que para o requisito 1.3 necessidades de alterações foram identificadas ao longo de um período de aproximadamente 4 meses. Isso se justifica por novas necessidades que foram surgindo ao longo do tempo, das 7 ações identificadas apenas uma representa um erro. Este é um exemplo de como um software precisa ser adaptar ao domínio de aplicação dos usuários.

| Requisito 3.1 | | | | |
|---|---|---|---|---|
| **O QUE (*What*)?** | **POR QUE (*Why*)?** | **QUANDO (*When*)?** | **QUEM (*Who*)? (IDENTIFICOU)** | **COMO (*How*)?** |
| Criar crachás distintos por unidade. | Para que seja possível identificar de forma mais rápida de qual unidade é o funcionário. | 30/01/09 | U2 | Implementação de uma melhoria no sistema |
| Alterar crachá, mostrar local de trabalho. | Antes estava mostrando o setor, mas é importante que o crachá identifique onde o funcionário atua. | 03/04/09 | U1 | Implementação de uma melhoria no sistema |
| Crachá - aumentar altura do código de barras e área em branco. | Para facilitar a leitura pelo relógio de ponto. | 30/06/09 | U2 | Correção de um erro de implementação do sistema |
| Alterar crachá possibilitando impressão mesmo quando o funcionário é de unidade diferente da unidade principal. | Só estava sendo possível imprimir crachás de funcionários lotados na unidade principal. | 16/10/09 | U2 | Correção de um erro de implementação do sistema |
| Permitir a impressão de uma lista de crachás impressos. | Para facilitar o controle de impressão e entrega de crachás. | 19/10/09 | U2 | Implementação de uma função no sistema |

Quadro 4.7- 5W1H para o requisito 3.1.

As alterações necessárias no requisito 3.1 (quadro 4.7) caracterizam-se por intervalos de tempo relativamente longos (2 meses) para a identificação das ações. Esta pode ser considerada uma característica negativa porque indica que os usuários permaneceram durantes esses intervalos sem utilizar o requisito.

| Requisito 3.2 | | | | |
|---|---|---|---|---|
| **O QUE (*What*)?** | **POR QUE (*Why*)?** | **QUANDO (*When*)?** | **QUEM (*Who*)? (IDENTIFICOU)** | **COMO (*How*)?** |
| Alterar tela dos dados de funcionários. | O select de Regime de Trabalho não está mostrando as opções. | 30/10/08 | U15 | Correção de um erro de implementação do sistema |
| Acrescentar campo salário base no cadastro de função. | O sistema deve permitir o controle dos salários referentes a cada função. | 07/11/08 | D2 | Implementação de uma função no sistema |
| Acrescentar campo nivel no cadastro da função. | Para possibilitar a impressão de relatórios por nível de instrução da função do funcionário. | 04/12/08 | D1 | Implementação de uma melhoria no sistema |
| Alterar consulta por portaria. | Consulta de Portaria por número não está buscando. | 13/01/09 | U15 | Correção de um erro de implementação do sistema |
| Alterar inserção de função. | Na inserção de função, o primeiro caractere de cada palavra está sendo omitido. | 14/01/09 | U15 | Correção de um erro de implementação do sistema |
| Alterar campo salário no cadastro da função. | Campo salário quando não preenchido aparece no banco como NaN. | 21/01/09 | D2 | Correção de um erro de implementação do sistema |
| Permitir cadastro de funcionário quando não tem portaria. | Porque atualmente aparece um erro na tela. | 04/02/09 | D2 | Correção de um erro de implementação do sistema |
| Alterar cadastro de portaria. | Sistema não está permitindo inserção. | 02/04/09 | D3 | Correção de um erro de implementação do sistema |
| Incluir verificação de duplicidade no cadastro de portaria. | Para evitar dados duplicados no banco. | 02/04/09 | D3 | Implementação de uma melhoria no sistema |
| Colocar no sistema uma opção para registrar funcionários cedidos, rescindidos, a disposição etc. | É necessário ter o controle de funcionários que vieram de outros lugares, que estão atuando em outros lugares ou que não estão mais no quadro de funcionários. | 18/09/09 | U11 | Implementação de uma função no sistema |
| Não está sendo possível atualizar cadastros portarias. | Existe um erro na verificação de duplicidade. | 19/10/09 | D2 | Correção de um erro de implementação do sistema |

Quadro 4.8 - 5W1H para o requisito 3.2.

Considerando os requisitos críticos, o requisito 3.2 (quadro 4.8) é o que possui o maior número de ações necessárias, envolvendo 3 usuários e 2 desenvolvedores. Além disso, essas ações foram identificadas ao longo de um período de 1 ano e das 11 ações 7 representam correções de erros.

| Requisito 4.2 | | | | |
|---|---|---|---|---|
| **O QUE (*What*)?** | **POR QUE (*Why*)?** | **QUANDO (*When*)?** | **QUEM (*Who*)? (IDENTIFICOU)** | **COMO (*How*)?** |
| Alterar abono, colocar por dia e não período. | Essa mudança facilita a identificação das horas trabalhadas por dia. | 20/08/09 | D3 | Implementação de uma melhoria no sistema |
| Mostrar os abonos nos relatórios de freqüência. | Para que o responsável pelo funcionário tome ciência e dessa forma comunique ao DP. | 02/10/09 | D2 | Implementação de uma função no sistema |

Quadro 4.9 - 5W1H para o requisito 4.2

De acordo com o quadro 4.9 apenas 2 ações foram identificadas para o requisito 4.2. Essas ações foram identificadas por desenvolvedores, o que indica que é necessário um retorno maior dos usuários a respeito do requisito para as melhorias possam ser realizadas.

### 4.1.11 Realização de nova avaliação de desempenho

No caso do software em questão uma nova avaliação de desempenho pode ser feita após a execução de todas as ações de melhoria identificadas.

## 4.2 Análise dos resultados e da metodologia – Gerente do Projeto

Com o intuito de observar a opinião do gerente do projeto em questão a respeito dos resultados obtidos e da metodologia proposta uma reunião foi realizada, os pontos mais importantes encontram-se listados a seguir.

Em relação aos resultados obtidos para os requisitos, o gerente demonstrou discordância em relação a 2 requisitos. O primeiro, o requisito 3.1 (Emissão de crachás), segundo o gerente, tem um desempenho muito bom, atende às necessidades dos usuários, não apresenta falhas e o sistema agiliza o processo de emissão de crachás. Para ele o processo de negócio relativo a esse requisito apresenta algumas dificuldades. Para que essa funcionalidade do sistema funcione corretamente é preciso encontrar a foto de cada funcionário e digitalizá-la para inserir no sistema. Trata-se de um processo demorado, mas a dificuldade não está no sistema e sim no processo. Além disso, existe uma dificuldade em relação à

obtenção do material necessário para a emissão dos crachás (plástico, papel para foto, etc., cuja compra é autorizada pela Prefeitura) e, para o gerente, os usuários de certa forma “confundem” isso e associam essa dificuldade ao sistema.

De acordo com a percepção do gerente, o requisito 4.1 (Controle de Férias) é mais importante do que os usuários avaliaram, visto que se trata de um processo bastante complexo e importante para o funcionamento da instituição. Além disso, ainda há o que melhorar nesse requisito, por isso, segundo ele o desempenho do requisito seria pior do que o que foi informado pelos usuários.

Ao analisar especificamente as etapas da metodologia proposta, o gerente destaca alguns pontos:

**a. Dificuldade em seguir as prioridades**

Para o gerente, a priorização dos requisitos é importante para o desenvolvimento de software porque mesmo que os desenvolvedores tenham uma percepção do que é mais importante, são os usuários as pessoas capazes de definir quais funcionalidades são mais urgentes para agilizar os processos internos. Após conhecer as prioridades, os desenvolvedores podem focar nos requisitos mais importantes.

Contudo, na instituição onde a metodologia foi aplicada, existem usuários de diferentes níveis gerenciais. O diretor da fundação, por exemplo, pode determinar que um requisito seja desenvolvido primeiro e solicitações desse tipo precisam ser atendidas. Por esse motivo nem sempre é possível seguir exatamente a ordem de prioridades definida pelos usuários.

Para resolver esse tipo de conflito é necessário definir formalmente um responsável pelo projeto com autonomia para tomar as decisões em situações deste tipo.

**b. Relação de dependência entre usuários e sistema**

A priorização dos requisitos de acordo com a percepção dos usuários permite que os mesmos logo tenham contato com requisitos importantes, o que de certa forma possibilita que os usuários estabeleçam uma relação de dependência com o sistema. Essa característica pode ser considerada relevante para a continuidade do projeto de desenvolvimento.

**c. Dependência entre requisitos**

Segundo o gerente, a dependência entre os requisitos do sistema pode ser ressaltada como um fator que dificulta a aplicação da metodologia. Ou seja, como desenvolver o requisito A prioritário se para ele funcionar o requisito B deve estar pronto? Neste caso, os desenvolvedores tendem a desenvolver logo o requisito B. Para seguir as prioridades em situações desse tipo é necessário desenvolver de forma básica os requisitos menos prioritários, inserindo posteriormente regras de negócio, por exemplo.

# CAPÍTULO V

# CONSIDERAÇÕES FINAIS

Neste capítulo são feitas algumas conclusões a respeito do estudo realizado e descrito nos capítulos anteriores. Além disso, as limitações identificadas e as sugestões para trabalhos futuros também são apresentadas

## 5.1 Conclusões

O software assume um papel de grande importância nas organizações, no entanto, o desenvolvimento de software é uma atividade complexa que agrega os interesses de várias pessoas. Por esse motivo, são comuns relatos de projetos de desenvolvimento de software fracassados.

Um dos maiores problemas relatados na literatura a respeito dos projetos de desenvolvimento de software refere-se à não conformidade do software aos requisitos desejados pelos usuários. Considerando essa dificuldade, esta dissertação teve como objetivo propor uma metodologia que possibilite a avaliação da qualidade de produtos de software desde as fases iniciais dos projetos de desenvolvimento, tendo sempre como foco a percepção dos usuários que, neste caso são os maiores conhecedores do domínio tratado pelo software.

Acredita-se que a satisfação dos usuários seja preponderante para o sucesso de um projeto de desenvolvimento e para a continuidade de utilização de um software. Por isso, a metodologia proposta buscou o alinhamento entre as necessidades dos usuários e o software desenvolvido.

Desta forma, embora não seja possível garantir, espera-se que através da aplicação da metodologia apresentada seja possível aumentar as chances de sucesso dos projetos de desenvolvimento e elevar a qualidade dos produtos de software.

O estudo de caso apresentado e o seu desdobramento objetivaram verificar a viabilidade de aplicação da metodologia proposta. A partir dessa aplicação conclui-

se que é possível aplicar a metodologia em projetos de desenvolvimento de pequeno porte. No estudo, as respostas fornecidas pelos usuários do software foram obtidas para as avaliações de importância e desempenho de um conjunto de requisitos previamente identificados.

Após as avaliações de importância e desempenho o gerente responsável pelo projeto de desenvolvimento do software fez algumas considerações a respeito da aplicação da metodologia. Segundo ele, um aspecto negativo da metodologia refere-se ao fato de que nem sempre é possível seguir as prioridades definidas pelos usuários. Além disso, para ele é necessário inserir na metodologia uma forma de mapear as interdependências entre requisitos. Por outro lado a priorização torna os usuários dependentes do software, o que pode ser positivo para o projeto de desenvolvimento.

Os resultados obtidos no estudo de caso permitiram a obtenção de dois benefícios principais provenientes da aplicação da metodologia.

O primeiro deles é a possibilidade de priorização dos requisitos com base na percepção dos usuários do software. Essa priorização permitiu que os desenvolvedores tivessem uma base para conduzir o trabalho de desenvolvimento, tendo como referência os requisitos mais relevantes de acordo com a percepção dos usuários.

Um segundo benefício é obtenção de um método para avaliar o desempenho do software, tendo como critérios os requisitos funcionais. Esses requisitos são importantes para percepção dos usuários em relação ao software. Em geral os usuários criam expectativas relacionadas a esse tipo de requisito. A avaliação de desempenho permitiu que ações de melhorias do software fossem sugeridas através da busca da melhoria contínua.

Além dos benefícios, algumas dificuldades foram encontradas. A principal delas refere-se ao fato de que nem sempre são os desenvolvedores que decidem a ordem de desenvolvimento dos requisitos. Por isso, mesmo que os desenvolvedores desejem seguir exatamente a lista de requisitos priorizada derivada da etapa de priorização, influências externas podem impedir que isso seja feito. Um exemplo pode ser a decisão gerencial tomada por gerentes da instituição onde o software será implantado.

Diante dos benefícios e dificuldades, esta dissertação possui como diferencial a apresentação de uma proposta que busca a melhoria da qualidade do produto de

software desde as etapas iniciais do desenvolvimento, tendo como referência as necessidades dos usuários.

Além disso, foi identificada uma lacuna referente às novas metodologias que são propostas publicadas, mas que nem sempre são aplicadas pelos profissionais de TI. O estudo de caso mostrou que é possível aplicar metodologias desse tipo até mesmo em projetos de pequeno porte.

## 5.2 Limitações

Após o estudo realizado algumas limitações fora identificadas. Por tratar-se de um estudo de caso os resultados obtidos são específicos para o contexto onde a aplicação foi feita, ou seja, não é possível afirmar que em um projeto de grande porte, por exemplo, os resultados obtidos serão satisfatórios.

Em relação às características dos métodos utilizados, segundo Duke e Mount (1996) a matriz de análise importância-desempenho possui como limitação a falta de testes de significância estatística. Esta também pode ser considerada uma limitação da metodologia proposta.

Além disso, as equipes envolvidas no estudo, tanto a de desenvolvedores, quanto a de usuários são pequenas e por isso as amostras utilizadas são pequenas, o que dificulta a realização de análises estatísticas. No entanto, buscou-se envolver na pesquisa todos os usuários do setor cliente.

Por fim uma das características do desenvolvimento de software é a capacidade dos requisitos mudarem, seja por fatores externos ou internos. Essas mudanças podem dificultar a etapa de priorização dos requisitos, caracterizando-se como mais uma limitação da metodologia proposta.

## 5.3 Trabalhos futuros

Como trabalho futuro, sugere-se a aplicação da metodologia em projetos de grande porte para testar a sua viabilidade.

Uma outra sugestão seria o desenvolvimento de um método capaz de mapear as interdependências entre requisitos. Requisitos podem depender uns dos outros e essa dependência pode influenciar a etapa de priorização, no entanto os métodos apresentados e estudados para priorização não permitem esse mapeamento.

Além disso, acredita-se que um sistema informatizado para apoiar a aplicação da metodologia proposta seria de grande utilidade, pois facilitaria a tabulação e análise dos dados referentes às avaliações de importância e desempenho. Por meio do uso do sistema seria possível realizar novas avaliações de forma mais fácil.

## REFERÊNCIAS BIBLIOGRÁFICAS


ABNT. Associação Brasileira de Normas Técnicas, 2009. Disponível em: <http://www.abnt.org.br>. Acesso em: 08/12/2009.

Ainin, S.; Hisham, N.H. (2008) Applying Importance-Performance Analysis to Information Systems: An Exploratory Case Study. *Journal of Information, Information Technology, and Organizations,* 3:95-103.

Allen, J.H.; *et al.* (2008) *Software Security Engineering: A Guide for Project Managers.* Addison Wesley Professional. 368p.

Belgamo, A.; Fabbri, S.; Maldonado, J. C. (2005) Avaliando a qualidade da técnica GUCCRA com técnica de inspeção. *Proceedings of the VII Workshop on Requirements Engineering*, Porto.

Berander, P. (2004) *Prioritization of Stakeholder Needs in Software Engineering Understanding and Evaluation*. Thesis (degree of Licentiate of Technology in Software Engineering), Sweden, Department of Systems and Software Engineering - Blekinge Institute of Technology, 172p.

Bertrán, I.M. (2009) *Avaliação da qualidade de software com base em modelos UML*. Dissertação de Mestrado. Rio de Janeiro, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Blose, J.E.; Tankersley, W.B.; Flynn, L.R. (2005) Managing Service Quality Using Data Envelopment Analysis. *American Society for Quality, Quality Management Journal*. 12(2):7-24.

Boehm, B.W. (1983) Seven Basic Principles of Software Engineering. *The Journal of Systems and Software.* 3:3-24.

Boehm, B.W.; Papaccio, P.N. (1988) Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineerlng*. 14(10):1462-1477.

Boente, A.N.P.; Moré, J.D. (2009) Um Modelo *Fuzzy* para Avaliação da Satisfação dos gerentes de Projetos de Produtos de Software de uma Fundação Pública Estadual. XXIX Encontro Nacional de Engenharia de Produção, Salvador – BA.

Booch, G.; Rumbaugh, J.; Jacobson, I. (2000) *UML, Guia do Usuário.* Rio de Janeiro: Campus. 472p.

Campos, V.F. (1994) *TQC: Gerenciamento da rotina do trabalho do dia-a-dia.* Rio de Janeiro: Bloch. 274p.

Carvalho, A.E.S.; Tavares, H.C.; Castro, J.B. (2001) Uma Estratégia para Implantação de uma Gerência de Requisitos visando a Melhoria dos Processos de Software. *WER2001 - IV Workshop on Requirements Engineering*, Buenos Aires.

Celestino, U.; Abe, J.M. (2009) Avaliação da qualidade de produto de software utilizando Lógica Paraconsistente Anotada: estudo de caso com software ERP. *INGEPRO - Inovação, Gestão e Produção*. 1(4):105-111.

Cordeiro, A.G.; Moll, R.N. (2006) Pesquisa de Satisfação de Usuários de Software de Gestão Hospitalar Utilizando os Critérios da ISO 9126*. Anais do X Congresso Brasileiro de Informática em Saúde*, Florianópolis - SC.

Cordeiro, A.G.; Freitas, A.L.P. (2008) O cenário atual da qualidade de software. *Anais do XV Simpósio de Engenharia de Produção*, Bauru – SP.

Coughlan, J.; Macredie, R.D. (2002) Effective Communication in Requirements Elicitation: A Comparison of Methodologies*. Requirements Engineering.* 7:47–60.

Crespo, A.N.; *et al.* (2004) Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo. *III Simpósio Brasileiro de Qualidade de Software (SBQS)*.

Dantas, C.R.; Murta, L.G.P.; Werner, C.M.L. (2005) Consistent Evolution of UML Models by Automatic Detection of Change Traces. *Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*, p.14-147.

Davis, A.M. (2003) The Art of Requirements Triage. *IEEE Computer Society*.

Denning, P.J. (1992) What is software quality. *Communications of ACM.* 35(1).

Duan, C.; *et al.* (2009) Towards automated requirements prioritization and triage. *Requirements Engineering* 14:73–89.

Duke, C.R.; Mount, A.S. (1996) Rediscovering performance-importance analysis of products. *Journal of product & brand management.* 5(2):43-54.

Duke, C.R. (2002) Learning Outcomes: Comparing Student Perceptions of Skill Level and Importance. *Journal of Marketing Education*. 24:203-217.

Feigenbaum, A.V. (1994) *Controle da qualidade total*. São Paulo: Makron Books. 205p.

Freeman, S.; *et al*. (2004) Mock Roles, Not Objects. *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. Vancouver, Canadá.

Freire, E. (2002) *Inovação e competitividade: o desafio a ser enfrentado pela indústria de software.* Dissertação de mestrado (Pós-graduação em política científica e tecnológica) – Campinas - SP, Universidade Estadual de Campinas - Instituto de Geociências, 94p.

Freitas, A.L.P.; Manhães, N.R.C.; Cozendey, M.I. (2006) Emprego do SERVQUAL na avaliação da qualidade de serviços de Tecnologia da Informação: uma análise experimental. *Anais do XXVI ENEGEP*, Fortaleza - CE.

Gilb, T.; Maier; M.W. (2005) Managing Priorities: A Key to Systematic Decision-Making. *Proceedings of INCOSE Conference, Rochester NY USA*.

Gomes, A.; Oliveira, K.; Rocha, A.R. (2001) Avaliação de Processos de Software Baseada em Medições. *XV Simpósio Brasileiro de Engenharia de Software*, Rio de Janeiro. p. 84-99.

Gough, P.A.; Fodemski, F.T.; Higgins, S.A.; Ray, S.J. (1995) Scenarios - an industrial case study and hypermedia enhancements. *Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE '95)*, York, UK: IEEE Computer Society Press, p. 10-17.

Gousios, G.; *et.al*. (2007) Software quality assessment of open source software. *11th Panhellenic Conference on Informatics, PCI 2007*. A:303–315, Athens.

Herrmann, A.; Paech, B. (2008) MOQARE: misuse-oriented quality requirements engineering. *Requirements Engineering*. 13:73-86.

Hickey, A.M.; Davis, A.M. (2002) Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. *Proceedings of the 36th Hawaii International Conference on System Sciences*, IEEE Computer Society.

Hudson, S.; Hudson, P.; Miller, G.A. (2004) The Measurement of Service Quality in the Tour Operating Sector: A Methodological Comparison. *Journal of Travel Research*. 42:305-312.

IEEE – Institute of Electrical and Electronics Engineers (1990) Standard Glossary of Software Engineering Terminology, Document Number: IEEE 610.12.

ISO/IEC 15504-1 (2004) Information technology — Software process assessment — Part 1: Concepts and introductory guide. International Organization for Standardization.

Janzen, D.S.; Saiedian, H. (2008) Does Test-Driven Development Really Improve Software Design Quality? *IEEE Software*. 25(2):77-84.

Jesus, A.A. (2005) *Satisfação de clientes de serviços de restaurantes: um estudo na cidade de Salvador/BA*. Dissertação de mestrado – Balneário Camboriú, Universidade do vale do Itajaí, 180p.

Jung, H. (2007) Validating the external quality subcharacteristics of software products according to ISO/IEC 9126. *Computer Standards & Interfaces*. 29:653-661.

Karatzas, K.; Dioudi, E.; Moussiopoulos, N. (2003) Identification of major components for integrated urban air quality management and information systems via user requirements prioritization. *Environmental Modelling & Software.* 18:173-178.

Karlsson, J.; Ryan, K. (1996) Supporting the selection of Software Requirements. *Proceedings of the 8th International Workshop on Software Specification and Design (IWSSD '96)*, p. 146-149.

Karlsson, J.; Oisson, S.; Ryan, K. (1997) Improved Practical Support for Large-scale Requirements Prioritising. *Requirements Engineering.* 2:51-60.

Karlsson, J.; Wohlin, C.; Regnell, B. (1998) An evaluation of methods for prioritizing software requirements. *Information and Software Technology*. 39:939-947.

Khan, K.A. (2006) *A Systematic Review of Software Requirements Prioritization*. (Master Thesis of Science in Software Engineering) – Sweden, Blekinge Institute of Technology, 93p.

Koscianski, A.; Soares, M.S. (2007) *Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. 2. ed., São Paulo: Novatec Editora. 395p.

Larman, C. (2004) *Utilizando UML e padrões: uma introdução à análise e ao projeto orientado a objetos e ao Processo Unificado.* 2. ed., Porto Alegre: Bookman. 607p.

Larsen, T.J. (2009) A multilevel explanation of end-user computing satisfaction with an enterprise resource planning system within an international manufacturing organization. *Computers in Industry*. 60(9):657-668.

Lee, K.; Joshi, K.; Bae, M. (2008) Using analytical hierarchy Process (AHP) to identify the relative importance of the features needed for web-based systems development. *Information Resources Management Journal*. 21(3):88-100.

Leeworthy, V.R.; Wiley, P.C. (1996) *Importance and Satisfaction ratings by recreating visitors to the Florida Keys/ Key West.* The University of Georgia. 27p.

Leffingwell, D.; Widrig, D. (1999) *Managing Software Requirements.* Addison Wesley. 528p.

Lehtola, L.; Kauppinen, M. (2004) Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects. *Proceedings of the European Software Process Improvement Conference (EuroSPI 2004)*, p. 161-170.

Linberg, K.R. (1999) Software developer perceptions about software project failure: a case study. *The Journal of Systems and Software*. 49:177-192.

Magal, S.R.; Levenburg, N.M. (2005) Using Importance-Performance Analysis to Evaluate E-Business Strategies among Small Businesses. *Proceedings of the 38th Hawaii International Conference on System Sciences*.

Magalhães, A.L.C. (2006) A Garantia da Qualidade e o SQA: Sujeito que ajuda e sujeito que atrapalha. *ProQualiti – Qualidade na produção de software*. 2(2):9-14.

Maiden, N.A.M.; Rugg, G. (1996) ACRE: selecting methods for requirements aicquisition*. Software Engineering Journal*. 11(3):183-192.

Malhotra, N. (2006) *Pesquisa de Marketing: uma orientação aplicada*. 4. ed., Porto Alegre: Bookman. 720p.

Martilla, J.A.; James, J.C. (1977) Importance–Performance Analysis. *Journal of Marketing*. 41:13-17.

Meszaros, G. Test Double, 2010. Disponível em: <http://xunitpatterns.com/Test%20Double.html>. Acesso em: 11/05/2010.

Morgado, G.P.; *et al*. (2007) Práticas do CMMI® como regras de negócio. *Produção*. 17(2):383-394.

NBR ISO/IEC 9126-1 (2003) Engenharia de Software – Qualidade de Produto parte 1: modelo de qualidade. ABNT – Associação brasileira de normas técnicas.

NBR ISO/IEC 12207. (1998) Tecnologia de informação – Processos de ciclo de vida de software. ABNT – Associação brasileira de normas técnicas.

Neill, C.J.; Laplante, P.A. (2003) Requirements Engineering: The State of the Practice. *IEEE Software*. 20(6):40-45.

Oliveira, K.R.; Belchior, A.D. (2002) *AdeQuaS*: Ferramenta *Fuzzy* para Avaliação da Qualidade de Software. *I Simpósio Brasileiro de Qualidade de Software – SBQS'2002,* Gramado - RS. p. 56-70.

Parasuraman, A., Zeithaml, V., Berry, L. (1988) A conceptual model of service quality and its implications for future research. *Journal of Marketing.* 49: 41-50.

Patenaude, J.-F.; *et al*. (1999) Extending Software Quality Assessment Techniques to Java Systems. *Proceedings Seventh International Workshop on on Program Comprehension*. p.49-56.

Pinheiro, F.A.C.; Leite, J.C.S.P.; Castro, J.F.B. (2003) Requirements Engineering Technology Transfer: An Experience Report*. Journal of Technology Transfer*. 28:159–165.

Prado, D.S. (1999) *Gerência de projetos em tecnologia da informação.* Belo Horizonte: Editora de Desenvolvimento Gerencial. 39p.

Pressman, R.S. (2002) *Engenharia de Software*. 5. ed., Rio de Janeiro: McGraw-Hill. 843p.

Punter, T.; *et al*. (2004) TheW-Process for Software Product Evaluation: A Method for Goal-Oriented Implementation of the ISO 14598 Standard. *Software Quality Journal*. 12:137–158.

RENAPI. Rede Nacional de Pesquisa e Inovação em Tecnologias Digitais, 2010. Disponível em: <http://www.renapi.org/qualidade/metodologia1/copy_of_finalidade>. Acesso em: 07/05/2010.

Robertson, S.; Robertson, J. (2006) *Mastering the requirements process*. 2. ed., Addison Wesley Professional. 592p.

Roselino, J.E. (2006) *A Indústria de Software: o "modelo brasileiro" em perspectiva comparada.* (Tese de doutorado) – Campinas – SP, Universidade Estadual de Campinas - Instituto de economia, 222p.

Rosqvist, T.; Koskela, M.; Harju, H. (2003) Software Quality Evaluation Based on Expert Judgement. *Software Quality Journal*. 11:39–55.

Sadraei, E.; Aurum, A.; Beydoun, G.; Paech, B. (2007) A field study of the requirements engineering practice in Australian software industry. *Requirements Engineering*. 12:145-162.

Sawyer, P.; Sommerville, I.; Viller, S. (1998) Improving the Requirements Process. *Proceedings of the Fourth International Workshop on Requirements Engineering*: Foundation for Software Quality, p. 8-9.

Silva, S.V. (2003) *Qualidade de software – uma abordagem baseada na satisfação do usuário*. Dissertação de Mestrado. Campos dos Goytacazes, Universidade Estadual do Norte Fluminense – UENF, 170p.

Skok, W.; Kophamel, A.; Richardson, I. (2001) Diagnosing information systems success: importance–performance maps in the health club industry. *Information & Management.* 38:409-419.

Slack, N. (1994) The Importance-Performance Matrix as a Determinant of Improvement Priority. *International Journal of Operations and Production Management*. 4(5):59-75.

Sommerville, I. (2007) *Engenharia de Software*. 8. ed., São Paulo: Pearson Addison Wesley.

Swebok Guide. (2004) *Guide to the Software Engineering Body of Knowledge.* Los Alamitos: IEEE Computer Society. 204p.

Tor J.L. (2009) A multilevel explanation of end-user computing satisfaction with an enterprise resource planning system within an international manufacturing organization. *Computers in Industry*. 60(9):657-668.

Wade, D.J.; Eagles, P.F.J. (2003) The Use of Importance–Performance Analysis and Market Segmentation for Tourism Management in Parks and Protected Areas: An Application to Tanzania's National Parks. *Journal of Ecotourism*. 2(3):196-212.

Xenos, M.; Christodoulakis, D. (1995) Evaluating Software Quality by the Use of User Satisfaction Measurements. *Proceedings of the 4th Software Quality Conference*, University of Abertay Dundee, p. 181-188, 1995.

Xenos, M. (2001) Usability Perspective in Software Quality*. Usability Engineering Workshop, Proceedings of the 8th Panhellenic Conference on Informatics with international participation*, Cyprus. v.2, p. 523-529.

Young, R.R. (2004) *The Requirements Engineering Handbook*. Boston: Artech House. 251p.

# APÊNDICE A – QUESTIONÁRIO DE PRIORIZAÇÃO DE REQUISITOS

UENF Mestrado PPGEP-UENF
Universidade Estadual do Norte Fluminense Darcy Ribeiro

FUNDAÇÃO Dr. João Barcellos Martins

HOSPITAL FERREIRA MACHADO

Esta pesquisa tem como objetivo a priorização dos requisitos relativos ao sistema que está sendo desenvolvido para os setores de Departamento Pessoal, Recursos Humanos e Segurança e Medicina do trabalho. Através dessa pesquisa pretendemos desenvolver o software para atender da melhor forma possível às necessidades dos setores envolvidos. **Por isso, esteja atento às questões para definir suas respostas**.

Função: _____ Tempo de Experiência no setor: _____

**1) Na tabela a seguir observe os requisitos, características que o sistema deve possuir. Marque na tabela a importância relativa a cada requisito.**

| | REQUISITOS | Grau de Importância | | | | |
|---|---|---|---|---|---|---|
| | | Nada Importante | Pouco Importante | Neutro | Importante | Muito Importante |
| 1 | **Controle de Freqüência dos Funcionários** | | | | | |
| 1.1 | Registro de freqüência | 1 | 2 | 3 | 4 | 5 |
| 1.2 | Registro de informações de escalas e horários de trabalho | 1 | 2 | 3 | 4 | 5 |
| 1.3 | Emissão de relatório de freqüência | 1 | 2 | 3 | 4 | 5 |
| 2 | **Manutenção dos dados pessoais** | | | | | |
| 2.1 | Controle das informações pessoais dos funcionários | 1 | 2 | 3 | 4 | 5 |
| 2.2 | Registro de deficiências apresentadas pelos funcionários | 1 | 2 | 3 | 4 | 5 |
| 3 | **Manutenção dos dados funcionais** | | | | | |
| 3.1 | Emissão de crachás | 1 | 2 | 3 | 4 | 5 |
| 3.2 | Manutenção de informações relativas à convocação | 1 | 2 | 3 | 4 | 5 |
| 3.3 | Controle de Alocação de funcionários em unidade e setores | 1 | 2 | 3 | 4 | 5 |
| 4 | **Manutenção dos Afastamentos** | | | | | |
| 4.1 | Controle de Férias | 1 | 2 | 3 | 4 | 5 |
| 4.2 | Controle de Afastamentos como Licenças Médicas e INSS | 1 | 2 | 3 | 4 | 5 |
| 5 | **Movimentação de Documentos** | | | | | |
| 5.1 | Entrada e saída de carteiras de trabalho e previdência social | 1 | 2 | 3 | 4 | 5 |
| 5.2 | Solicitação e entrega de declarações | 1 | 2 | 3 | 4 | 5 |

**2) Considere que você tem 100 pontos, distribua-os pelos requisitos apresentados de acordo com a necessidade referente a cada requisito. A soma dos pontos distribuídos deve totalizar 100 pontos**.

| | REQUISITOS | PONTOS |
|---|---|---|
| 1 | **Controle de Freqüência dos Funcionários** | |
| 1.1 | Registro de freqüência | |
| 1.2 | Registro de informações de escalas e horários de trabalho | |
| 1.3 | Emissão de relatório de freqüência | |
| 2 | **Manutenção dos dados pessoais** | |
| 2.1 | Controle das informações pessoais dos funcionários | |
| 2.2 | Registro de deficiências apresentadas pelos funcionários | |
| 3 | **Manutenção dos dados funcionais** | |
| 3.1 | Emissão de crachás | |
| 3.2 | Manutenção de informações relativas à convocação | |
| 3.3 | Controle de Alocação de funcionários em unidade e setores | |
| 4 | **Manutenção dos Afastamentos** | |
| 4.1 | Controle de Férias | |
| 4.2 | Controle de Afastamentos como Licenças Médicas e INSS | |
| 5 | **Movimentação de Documentos** | |
| 5.1 | Entrada e saída de carteiras de trabalho e previdência social | |
| 5.2 | Solicitação e entrega de declarações | |
| | **TOTAL:** | 100 |

**3) Informe outros requisitos que você julgue relevantes e que não foram encontrados nas listas acima:**

_____

_____

# APÊNDICE B – QUESTIONÁRIO DE AVALIAÇÃO DE DESEMPENHO COM BASE NOS REQUISITOS

UENF Mestrado PPGEP-UENF
Universidade Estadual do Norte Fluminense Darcy Ribeiro

FUNDAÇÃO Dr. João Barcellos Martins

HOSPITAL FERREIRA MACHADO

Esta pesquisa tem como objetivo a avaliação do sistema que está sendo desenvolvido para os setores de Departamento Pessoal, Recursos Humanos e Segurança e Medicina do trabalho. Através dessa pesquisa pretendemos verificar os requisitos do software que precisam ser alterados para atender da melhor forma possível às necessidades dos setores envolvidos. **Por isso, esteja atento às questões para definir suas respostas.**

Função: _____ Tempo de Experiência no setor: _____

**1) Na tabela a seguir observe os requisitos, características que foram consideradas necessárias no sistema. Marque na tabela o grau de desempenho relativo a cada requisito.**

| | | Grau de Desempenho | | | | |
|---|---|---|---|---|---|---|
| | REQUISITOS | Muito Ruim | Ruim | Neutro | Bom | Muito Bom |
| 1 | **Controle de Freqüência dos Funcionários** | | | | | |
| 1.1 | Registro de freqüência | 1 | 2 | 3 | 4 | 5 |
| 1.2 | Registro de informações de escalas e horários de trabalho | 1 | 2 | 3 | 4 | 5 |
| 1.3 | Emissão de relatório de freqüência | 1 | 2 | 3 | 4 | 5 |
| 2 | **Manutenção dos dados pessoais** | | | | | |
| 2.1 | Controle das informações pessoais dos funcionários | 1 | 2 | 3 | 4 | 5 |
| 2.2 | Registro de deficiências apresentadas pelos funcionários | 1 | 2 | 3 | 4 | 5 |
| 3 | **Manutenção dos dados funcionais** | | | | | |
| 3.1 | Emissão de crachás | 1 | 2 | 3 | 4 | 5 |
| 3.2 | Manutenção de informações relativas à convocação | 1 | 2 | 3 | 4 | 5 |
| 3.3 | Controle de Alocação de funcionários em unidade e setores | 1 | 2 | 3 | 4 | 5 |
| 4 | **Manutenção dos Afastamentos** | | | | | |
| 4.1 | Controle de Férias | 1 | 2 | 3 | 4 | 5 |
| 4.2 | Controle de Afastamentos como Licenças Médicas e INSS | 1 | 2 | 3 | 4 | 5 |

**2) Forneça um pontuação de 0 a 100 de acordo com o funcionamento de cada requisito do sistema. Os requisitos que você considera que estão funcionando melhor devem receber uma pontuação maior.**

| | |
|---|---|
| *EXEMPLO: Requisito A* | Muito Ruim 0 10 20 30 40 **X** 50 60 70 80 90 100 Muito Bom |
| *Registro de Freqüência* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Registro de informações de escalas e horários de trabalho* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Emissão de relatório de freqüência* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Controle das informações pessoais dos funcionários* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Registro de deficiências apresentadas pelos funcionários* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Emissão de crachás* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Manutenção de informações relativas à convocação* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Controle de Alocação de funcionários em unidade e setores* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Controle de Férias* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |
| *Controle de Afastamentos como Licenças Médicas e INSS* | Muito Ruim 0 10 20 30 40 50 60 70 80 90 100 Muito Bom |

**3) Comentários e observações (*use o verso da folha, se for preciso*):**

_____

_____

# ANEXO A – DESCRIÇÃO INICIAL DO SOFTWARE DO DEPARTAMENTO DE PESSOAL

**AGOSTO / 2008 – Projeto do DP / RH**

A Fundação é responsável por várias unidades de saúde, a admissão e controle dos funcionários lotados nessas unidades são de responsabilidade da fundação. O sistema deve possibilitar a manutenção das informações relativas aos funcionários referentes aos dados pessoais e funcionais.

Antes da admissão, uma série de procedimentos deve ser realizada para que admissão possa acontecer, esses procedimentos incluem realização de exames e consulta médica, entrevista e obtenção de documentação comprobatória. Nesta fase inicial, a pessoa que está sendo admitida é atendida pelos seguintes setores da fundação: DP, RH e SESMT.

Após os procedimentos iniciais e caso não haja nenhum impedimento, a pessoa é contratada e lotada em um setor de uma das unidades abrangidas pela fundação. Durante o seu tempo de atuação, o funcionário pode ser transferido para outro setor ou unidade e o sistema deve controlar esse tipo de movimentação.

Usualmente, o funcionário é identificado na fundação através de sua matrícula, no entanto, em alguns casos uma mesma pessoa pode possuir mais de um vínculo funcional e por isso podem possuir mais de uma matrícula. Funcionários podem ser plantonistas ou diaristas, o sistema deve possibilitar o registro desse tipo de informação associada ao horário de trabalho. A função desempenhada está associada ao Cadastro Brasileiro de Ocupações (CBO).

Uma dificuldade dos gestores do DP, diz respeito ao controle de férias, o sistema deve auxiliar na identificação de quais funcionários têm direito a férias e quais estão com o período de férias próximo do vencimento.