UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DIRETRIZES PARA ELABORAÇÃO DE DOCUMENTO DE REQUISITOS COM ÊNFASE NOS REQUISITOS FUNCIONAIS

KARINA KIYOMI KAWAI

São Carlos-SP
Setembro/2005

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Diretrizes para elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais"

KARINA KIYOMI KAWAI

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

Membros da Banca:

Profa. Dra. Sandra Camargo P. Ferraz Fabbri
(Orientadora – DC/UFSCar)

Profa. Dra. Rosely Sanches
(ICMC/USP)

Profa. Dra. Silvia Regina Vergílio
(INF/UFPR)

São Carlos
Setembro/2005

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

K22de

Kawai, Karina Kiyomi.
Diretrizes para elaboração de documento de requisitos com ênfase nos requisitos funcionais / Karina Kiyomi Kawai. -- São Carlos : UFSCar, 2007.
172 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Engenharia de software. 2. Engenharia de requisitos. 3. Técnicas de inspeção. 4. Inspeção de documento de requisitos. 5. Requisitos funcionais. I. Título.

CDD: 005.1 (20ª)

Dedico este trabalho aos meus queridos pais ...

Agradecimentos

Muito agradeço,

a Deus pelo conforto nas horas difíceis e pela luz que ilumina sempre o meu caminho.

à minha orientadora, Prof[a]. Dr[a]. Sandra Fabbri, pela oportunidade de realização deste trabalho e que durante este percurso me ajudou e orientou no desenvolvimento deste.

em especial, aos meus pais, Kohatiro e Kinue, pelo exemplo de perseverança e desvelo em aprimorar a minha formação.

aos meus irmãos, Rogério e Daniel, pelo exemplo de caráter e de força em busca de realização profissional.

ao amigo Glauco, que me ajudou ao realizar a minha inscrição na prova de seleção ao curso de mestrado.

aos amigos da turma de 2003 do PPGCC pelas amizades conquistadas durante esse período e horas de estudo em que passávamos juntos preparando-se para as provas.

aos professores Hélio, Júnia, Prado, Rosângela, Saito, Sandra e Wanderley pelo ensinamento das disciplinas cursadas durante o curso de mestrado.

ao colega Anderson pelo apoio dado quando dúvidas relacionadas a este trabalho eram prontamente respondidas.

às minhas amigas de república, Ana Paula, Daniela e Mirela, pela amizade, companhia e momentos de descontração.

às meninas Clara, Débora, Talita e Thaís pela bondade e receptividade durante minha estadia em sua república.

a todos que acreditaram neste trabalho e que de certa forma contribuíram para conclusão deste.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho apresenta Diretrizes para a Elaboração de Documento de Requisitos (DR) com ênfase nos Requisitos Funcionais, compostas por três itens: i) Formato para Especificação de Requisitos Funcionais, que determina um conjunto de informações básicas que deve compor a descrição do requisito; ii) Recomendação de Escrita, que oferece sugestões para que determinados defeitos sejam evitados durante a escrita do requisito e iii) *Checklist* Pré-Inspeção, que apóia uma sucinta avaliação dos requisitos de forma a ajudar na decisão se o DR deve ser submetido a uma inspeção. A definição dessas Diretrizes teve como base a análise de alguns padrões de especificação de requisitos, algumas recomendações de escrita sugeridas por alguns autores e, principalmente, uma avaliação da aplicação das técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA (*Technique for Use Case Construction and construction-based requirements Analysis*) em três DRs diferentes. O objetivo principal de se definir essas Diretrizes foi apoiar a aplicação da TUCCA, a qual dá suporte à construção de Modelos de Casos de Uso (MCU) e à inspeção do DR, para a qual está sendo desenvolvida uma ferramenta que possui como entrada um DR. O estudo de caso para a avaliação das Diretrizes propostas consistiu da aplicação da PBR-Usuário, *Checklist* e TUCCA em três DRs e posterior aplicação dessas três técnicas de inspeção nesses mesmos DRs alterados de acordo com as Diretrizes propostas. O resultado obtido mostrou que a aplicação da TUCCA é bastante facilitada quando o DR segue as Diretrizes e que houve uma redução de defeitos no DR, o que contribui para a qualidade desse documento.

ABSTRACT

This work presents Guidelines to elaborate the Requirements Document (RD) based on Functional Requirements made up of three items: i) a Template to specify the Functional Requirements, which determines a set of basic information that should compose the requirement description; ii) Writing Recommendations that offer suggestions to avoid certain defects during requirement writing and iii) a Pre-Inspection Checklist that supports a brief evaluation of the requirements to help in deciding if the RD should be submitted to an inspection. Definition of such Guidelines were based on the analysis of some requirement specification standards, some writing recommendations suggested by some authors and mainly, on the evaluation of applying PBR-User, Checklist and TUCCA (*Technique for Use Case Construction and construction-based requirements Analysis*) inspection techniques in three different RDs. The main objective to define these Guidelines was to facilitate the application of TUCCA, which supports the elaboration of Use Case Model (UCM) and the inspection of RD, for which a tool that has the RD as input is being developed. The case study to evaluate the proposed Guidelines consisted of the application of PBR-User, Checklist and TUCCA in three RDs and posterior application of the three techniques in these same RDs after they were modified according to the proposed Guidelines. The obtained results showed that TUCCA application was made easier when RD followed the Guidelines and there was a defect reduction in RD thus increasing the quality of this document.

Capítulo 1

1. Introdução

1.1 Contexto

Há muito tempo tem-se visto o relato dos problemas no desenvolvimento de sistemas baseados em computador. Freqüentemente esses sistemas são entregues com atraso e com seu orçamento elevado, isso quando são entregues. Muitas vezes, o sistema desenvolvido não corresponde às reais necessidades do usuário e, por conseqüência, não é usado com total eficiência. Atribuem-se várias razões a esses problemas, mas o fator principal que contribui para isso é o manuseio inadequado dos requisitos, o qual tem sido repetidamente reconhecido como um problema real [Kotonya & Sommerville, 1998]. A Engenharia de Requisitos ainda é o princípio do processo de desenvolvimento de software para desenvolver com êxito um sistema que cumpre as necessidades e expectativas dos usuários e clientes.

Estudada no contexto da Engenharia de Software, a Engenharia de Requisitos é a etapa inicial da construção de um sistema baseado em computador, cuja finalidade é definir o objetivo do sistema proposto. Também pode ser definida como um processo sistemático e repetitivo de desenvolvimento de requisitos de software, abrangendo todas as atividades envolvidas na descoberta, documentação e manutenção de requisitos [Kotonya & Sommerville, 1998].

O produto final do Processo de Engenharia de Requisitos é um Documento de Especificação de Requisitos de Software, neste trabalho denominado apenas DR, que é basicamente o entendimento que uma organização de desenvolvimento de software tem dos requisitos de um sistema de um cliente e cujo presente trabalho pretendeu explorar.

Sendo freqüentemente referenciado em muitos dos documentos elaborados e gerados posteriormente, o DR serve de base para eles e assim exerce uma significante influência nos estágios seguintes do ciclo de vida de um sistema, ou seja, nos diferentes estágios do tempo de vida de um produto de software, que são tipicamente: análise, projeto, implementação, verificação e validação, integração e manutenção. Considerando isso e o fato de esse documento poder ocasionar altos impactos nos recursos do projeto, no seu cronograma e no

resultado final do software, há a necessidade de se elaborar um DR que seja bem definido e de alta qualidade.

Com uma boa especificação pretende-se levar à construção de um software que satisfaça o cliente e que resolva suas reais necessidades, tenta-se prevenir erros ao invés de consertá-los, evita-se perda de tempo para reparar defeitos quando os requisitos errôneos são implementados, propicia-se menos re-trabalho, evitam-se atrasos na entrega do produto, problemas de orçamento, gastos com recursos extras e até mesmo perda de vidas no caso de sistemas de segurança crítica. Portanto, é necessário que os requisitos sejam especificados corretamente para gerar uma especificação clara e precisa.

Embora haja consciência da importância do DR e vários trabalhos na literatura [Davis et al., 1993b; Firesmith, 2003; Hooks, 1993; Jacobs, 1999; Kar & Bailey, 1996; Kauppinen et al., 2002; Wiegers, 1999a] que discutam os problemas associados à sua elaboração, seus aspectos de qualidade, etc., não se encontrou uma proposta mais abrangente, que agrupe todas essas características e que proponha um formato que auxilie sua elaboração. Entende-se que essa não é uma tarefa trivial, mesmo porque o próprio domínio da aplicação tem uma grande influência sobre a forma de elaborá-lo. A questão é que os problemas da falta de qualidade do DR são levantados, mas uma solução ampla para que se consiga obter um DR ideal ou que ao menos auxilie a sua elaboração não foi encontrada.

De qualquer forma, uma vez elaborado o DR, para se dar prosseguimento ao desenvolvimento do software, esse documento deve passar por um processo de inspeção para que os defeitos sejam detectados e corrigidos e então, os requisitos declarados nesse documento devem ser modelados, de forma a colocá-los em um formato que seja mais apropriado para as subseqüentes transformações a que eles irão passar, até chegar no produto final. Essa modelagem, embora existam várias formas de fazê-la, tem sido feita, com mais freqüência atualmente, por meio de Modelo de Casos de Uso (MCU) [Jacobson et al., 1992] que, embora seja bastante utilizado no contexto de orientação a objetos, na verdade, independe do paradigma de desenvolvimento.

Para essa modelagem, foi realizado um outro trabalho de mestrado, já concluído, que propôs a técnica de leitura TUCCA – [Belgamo, 2004], que fornece diretrizes para a elaboração de MCU e, simultaneamente, conduz a uma avaliação (inspeção) do DR à medida que o MCU é construído. Ou seja, essa inspeção tem por objetivo detectar defeito no DR, mas com vistas à construção do MCU. Sendo assim, esse procedimento pode ajudar a detectar defeitos que tenham persistido após uma atividade de inspeção no DR ou, eventualmente, agir

como uma atividade de inspeção, caso o DR não tenha passado por uma inspeção prévia e esteja, naquele momento, sendo modelado com base em MCU.

Por meio da realização de alguns estudos experimentais [Belgamo & Fabbri, 2004a, 2004b, 2004c, 2005], [Belgamo et al., 2005a, 2005b], a TUCCA tem se mostrado uma técnica bastante efetiva tanto na elaboração de MCU como na detecção de defeitos no DR. Muitos passos dessa técnica são bastante procedimentais e como o DR é a entrada para a sua aplicação, é natural que a qualidade desse documento tenha interferência na sua aplicação. A forma apresentada por um DR pode dificultar os passos da TUCCA, pois ela pode ter muitos defeitos ou então não estar num modelo que facilite a identificação de certas informações durante os passos. Para automatizar essa técnica, uma ferramenta está sendo desenvolvida, cuja entrada é o DR. Mas para isso, o DR deve apresentar-se de tal forma que facilite a aplicação da TUCCA. Sendo assim, o tema deste trabalho foi definido, principalmente, com o objetivo de contribuir para a implementação dessa ferramenta, estabelecendo algumas Diretrizes que ajudem na elaboração de um DR. Com isso, essas Diretrizes fornecem instruções em como especificar um DR com ênfase nos Requisitos Funcionais. Como dito anteriormente, uma proposta para elaboração de DR não foi encontrada, assim tais Diretrizes iniciam esse interesse, contribuindo também para a qualidade do DR.

1.2 Motivação e objetivo

Com base no contexto apresentado anteriormente, os principais pontos que motivaram este trabalho foram os seguintes:

i) No grupo de pesquisa em que este trabalho está inserido, foi definida a técnica de leitura TUCCA que ajuda na construção de MCU e na detecção de defeitos no DR. Atualmente está sendo desenvolvida uma ferramenta que apóia a aplicação dessa técnica e definir um DR que facilite a entrada para essa ferramenta é um ponto bastante importante.

ii) Embora se encontrem na literatura algumas contribuições para elaboração do DR, percebe-se que estas são contribuições que atacam aspectos particulares.

iii) Não se encontrou na literatura nenhuma diretriz que ajude a determinar se um DR está apto a sofrer uma inspeção ou não, ou seja, se o DR está num estado de evolução e de detalhe tal que não fosse fazer da atividade de inspeção uma atividade pouco efetiva, pois o DR ainda não estava pronto para ser inspecionado.

Assim, este trabalho tem por objetivo investigar e agregar todas as recomendações encontradas na literatura que forem adequadas e pertinentes para compor diretrizes de elaboração do DR que auxiliem a aplicação da TUCCA por meio da ferramenta que está sendo desenvolvida para automatizar alguns passos dessa técnica de leitura, além dar início ao estabelecimento de uma forma de avaliação que possa ser usada para tomar a decisão sobre a condução de uma atividade de inspeção no DR.

1.3 Organização do trabalho

Este trabalho está estruturado em sete capítulos, além das Referências Bibliográficas e Anexos, a saber:

Neste capítulo de Introdução foram apresentados o contexto no qual este trabalho se insere, com uma breve explicação da importância da qualidade de um DR, os motivos que levaram ao seu desenvolvimento e o seu objetivo.

No Capítulo 2, apresentam-se os conceitos de Engenharia de Requisitos. É descrito todo o seu processo, suas fases são especificadas e definidas e dá-se ênfase ao DR.

No Capítulo 3, descrevem-se alguns tópicos relacionados ao DR, como padrão para sua especificação, as propriedades de qualidade que um DR deve conter e recomendações sugeridas na literatura para a elaboração desse documento.

No Capítulo 4, fala-se sobre Inspeção de DR, no qual apresentam-se algumas Técnicas de Leitura, que auxiliam essa atividade.

No Capítulo 5, são descritas as Diretrizes propostas para Elaboração de DR com Ênfase em Requisitos Funcionais, que são compostas de um Formato para Definição de Requisitos Funcionais, Recomendações de Escrita de sentenças de Requisitos e um *Checklist* Pré-Inspeção.

No Capítulo 6, é descrito o estudo de caso realizado para avaliar as Diretrizes propostas.

No Capítulo 7, apresenta-se a conclusão deste trabalho.

Capítulo 2

2. Engenharia de Requisitos

2.1 Considerações Iniciais

Sabe-se que muitas das maiores causas de falhas em projetos de software são as deficiências em definir requisitos. Estima-se que 85% dos defeitos originam-se nos requisitos [Young, 2001 apud Young, 2002][1], sendo que os tipos mais comuns de erros incluem: suposições incorretas (49%), requisitos omitidos (29%), requisitos inconsistentes (13%) e ambigüidades (5%) [Hooks & Farry, 2001 apud Young, 2002][2]. Enquanto esses problemas podem ser mais toleráveis em sistemas de segurança não crítica, pois embora causem enorme prejuízo não envolvem risco de vida a seres humanos, em sistemas críticos de segurança os erros não podem ser aceitáveis, eles são inadmissíveis.

Uma análise em 8000 projetos empreendida por 350 companhias dos Estados Unidos confirmou que problemas na definição de requisitos consistiam a principal causa de falhas em projetos, sendo especificamente, falta de envolvimento do usuário (13%), requisitos incompletos (12%), mudança nos requisitos (11%), expectativas irreais (6%) e objetivos não claros (5%) [*The Standish Group*, 1995 apud Lamsweerde, 2000][3]. Na Europa, pesquisa feita em 3800 organizações em 17 países também concluiu que a maioria dos problemas percebida nos software recaía sobre a especificação de requisitos (>50%) e gerenciamento de requisitos (50%) [*European Software Institute*, 1996 apud Lamsweerde, 2000][4].

Inúmeras razões são associadas à incapacidade de se definir bons requisitos [Scharer, 1990]: falta de uma atitude adequada para lidar com usuários (e vice-versa); a articulação dos requisitos é difícil, as funções e processos não são facilmente descritas; há mudanças nos requisitos dos sistemas; o uso inadequado de técnicas e ferramentas; as expectativas de usuários e engenheiros de requisitos são diferentes; o processo de definição pode tornar-se altamente político; uma solução sofisticada para o problema pode ser necessária para produzir

[1] Young, 2001.
[2] Hooks, 2001.
[3] The Standish Group, "Software Chaos", 1995.
[4] European Software Institute, "European User Survey Analysis", 1996.

boas declarações de requisitos; falta de critério/padrão de definição de requisitos; dificuldade para motivar o usuário até que se chegue ao estágio de implementação.

Outras causas de falhas em projetos de software estão associadas a diversos fatores, como às restritas informações de usuários, uma vez que eles assumem que os engenheiros de requisitos entendem todo o trabalho a ser desenvolvido; os conflitos de interesses, prioridades e falta de consenso entre as pessoas envolvidas com o sistema que será desenvolvido, sendo os projetos cancelados por falta de compatibilidade de negócios ou então retardados; uma precária estimativa de orçamento e cronograma, tornando a execução do projeto irreal; a não habilidade técnica da autoridade que decide o que deve ser realizado, pelo fato de ela não entender as implicações de riscos técnicos específicos; um inadequado planejamento, o que pode provocar descontrole no projeto; à falha de comunicação entre a equipe de projeto, pois muitas vezes as pessoas envolvidas não têm uma visão completa do projeto e não sabem como seu trabalho se encaixa no todo [May, 1998].

Observando-se os altos índices de dificuldade em se expor com maior precisão e clareza os requisitos de um sistema de software e considerando-se os problemas decorrentes disso, tem-se que a preocupação dos analistas de sistemas deve voltar-se justamente para a melhoria dessa fase inicial que é a definição de requisitos. Leite [2001] salienta que a qualidade do produto final é resultado dos métodos de produção escolhidos e seguidos e, portanto, a qualidade deve estar presente não só nos produtos gerados, mas nos processos utilizados.

O início do processo de concepção dos requisitos, denominado Engenharia de Requisitos, começa a partir do momento em que se propõe a elaboração de um sistema de software, o qual deveria ser criteriosamente seguido pelos analistas de sistemas.

Na seção seguinte é abordado o termo fundamental Requisito e, posteriormente, na Seção 2.3 o que se compreende por Processo de Engenharia de Requisitos, pois ambos estão diretamente relacionados ao DR, o qual é o artefato de pesquisa deste presente trabalho. Na Seção 2.4 são apresentados alguns trabalhos relacionados ao Processo de Engenharia de Requisitos, mostrando as diversas pesquisas que têm sido realizadas e, na Seção 2.5, apresentam-se as Considerações Finais.

2.2 O que são Requisitos

A definição ou descrição do termo Requisito pode ser encontrada em muitos livros de Engenharia de Software, sendo que em termos gerais pode-se dizer que Requisito é algo que é

solicitado ou desejado pelo cliente e/ou usuários finais, ou até mesmo indispensável para o que se pretende realizar. Pode ser uma função, característica ou restrição que o sistema deveria incluir ou exibir para satisfazer os objetivos da organização, apoiando suas tarefas e resolvendo um conjunto de problemas. Um requisito não deveria especificar características de projeto do sistema, apenas o seu comportamento externo.

Sommerville & Sawyer [1997] definem que Requisitos são descrições de como o sistema deveria se comportar, ou descrições da propriedade ou do atributo de um sistema. Eles podem ser uma restrição no processo de desenvolvimento de um sistema e também podem incluir informações sobre o domínio da aplicação.

Uma definição comum mencionada em várias fontes literárias e científicas é a do IEEE [1990], o qual define Requisito como: (1) uma condição ou capacidade necessitada por um usuário para resolver um problema ou alcançar um objetivo; (2) uma condição ou capacidade que deve ser cumprida ou possuída por um sistema ou componente do sistema para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos; (3) uma representação documentada de uma condição ou capacidade, como em (1) ou (2).

Embora o foco da definição de Requisitos de um sistema incida sobre as necessidades dos usuários finais, ao defini-los também se deve levar em conta as necessidades de todos os *stakeholders*, ou seja, de todas aquelas pessoas que são afetadas pelo resultado desse sistema e que têm uma ligação direta ou indireta com os seus requisitos. Eles incluem: clientes, usuários finais, gerentes, analistas de sistemas, mantenedores, treinadores, autoridades reguladoras, etc. [Kotonya & Sommerville, 1998].

Karlsson [1996 apud Dahlstedt, 2003][5] concluiu que cada Requisito tem três elementos que o caracteriza: a origem, o motivo e o objeto de realização, os quais estão ilustrados na Figura 2.1.

181vii

[5] Karlsson, 1996.

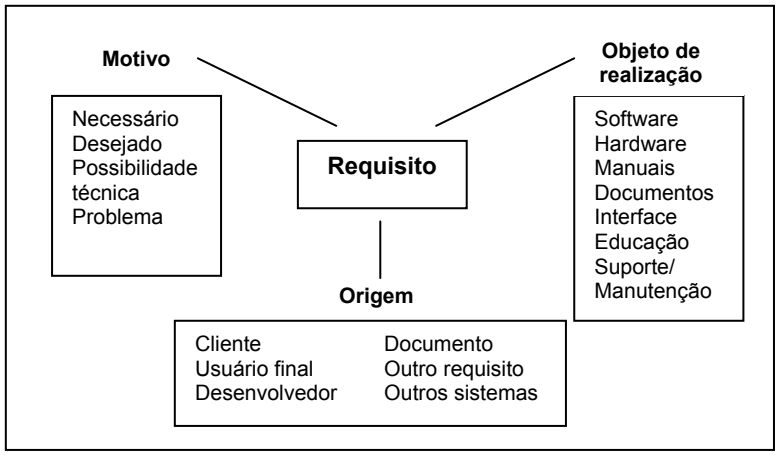


Figura 2.1 Os elementos de um requisito [Adaptado de Karlsson, 1996 apud Dahlstedt, 2003].

A origem de um Requisito significa a fonte de onde ele surgiu, podendo ser de um usuário ou de mais *stakeholders*, podendo ser adquirido de um documento de negócio, derivado de requisitos especificados anteriormente ou levantados devido à necessidade de interação com sistemas ou equipamentos existentes. O motivo pelo qual um Requisito existe geralmente é para realizar ou satisfazer algo de especial para um *stakeholder*, por exemplo: uma funcionalidade que o usuário precisa para desempenhar seu trabalho, uma possibilidade técnica que fará uma certa atividade de negócio mais eficiente, uma necessidade para apoiar as estratégias de negócio da organização. Por fim, um Requisito tem um ou mais objetos de realização, isto é, algo que implementa ou realiza a propriedade desejável do sistema. O objeto de realização mais usual é o software, mas também pode ser um hardware, manuais, interface, documentação do sistema, etc.

Os requisitos podem ser classificados em várias dimensões, sendo que a forma tradicional de classificá-los é:

- Requisitos Funcionais: descrevem os serviços ou funcionalidades que devem ser oferecidos pelo sistema, como ele deveria reagir a uma entrada particular e como deveria comportar-se em situações particulares. Ou seja, o que o sistema deve ser capaz de fazer [Sommerville, 1995].
- Requisitos Não-Funcionais: descrevem as limitações e restrições que precisam ser respeitadas pelo sistema, pelo seu ambiente e pelo processo de desenvolvimento, isto é, eles são restrições externas que o sistema deve cumprir [Kotonya & Sommerville, 1998]. Assim, esses Requisitos Não-Funcionais, também conhecidos como requisitos de qualidade, podem estar relacionados à política da organização, ao desempenho do sistema, ao processo de

engenharia que foi adotado, a um sistema operacional, à necessidade de se interoperar com outros sistemas ou de possuir certo grau de segurança, ao orçamento do projeto, etc. Exemplo de diferentes tipos de Requisitos Não-Funcionais: manutenibilidade, usabilidade, confiabilidade, eficiência, desempenho e capacidade do sistema, restrições legais e econômicas e requisitos de interoperabilidade.

2.3 Processo de Engenharia de Requisitos

A Engenharia de Requisitos é a fase inicial do processo do ciclo de vida de um software e seu objetivo é, basicamente, entender as necessidades dos *stakeholders* e, baseando-se nelas, especificá-las em requisitos do sistema. Ela pode ser considerada como um processo sistemático e disciplinado de desenvolver requisitos através de um processo que cobre todas as atividades requeridas na descoberta de requisitos, criação e manutenção do DR. O conjunto dessas sucessivas atividades estruturadas é o que se chama de Processo de Engenharia de Requisitos, cujas etapas incluem estudo da viabilidade do sistema, elicitação de requisitos e análise, especificação, validação e gerenciamento de requisitos [Sommerville, 1995 e 2001]. O documento gerado por esse processo é o artefato de interesse deste presente trabalho.

Vale ressaltar que utilizando-se de processos bem definidos para a definição dos requisitos, os problemas associados a eles, descritos na Seção 2.1, podem ser amplamente diminuídos ou evitados. E lembrando, para atender às exigências de mercado, à crescente demanda do uso de software e por inúmeras outras razões, diversas normas e padrões vêm sendo criados para regular e orientar a produção de software, como:

Norma ISO/IEC 12.207 [ISO/IEC 12.207, 1995]: descreve em detalhes os processos, atividades e tarefas relacionados ao desenvolvimento de software para auxiliar os envolvidos no fornecimento, desenvolvimento, operação e manutenção de produtos de software. Estabelece uma estrutura comum de processos para ser utilizada como referência em negócios relacionados a produtos de software. Esta norma agrupa os processos do ciclo de vida do software em três classes: Fundamentais, de Apoio e Organizacionais, que representam a sua natureza.

Norma ISO/IEC 15.504 [ISO/IEC 15.504]: conhecido como SPICE (*Software Process Improvement and Capability dEtermination*), oferece um "*framework"* para avaliação de processos de software cujos objetivos são a melhoria dos processos e a determinação de capacidade de processos de uma organização. Para a realização de uma avaliação é necessário

que o modelo de processo utilizado seja compatível com o modelo de referência de processo da norma ISO/IEC 12.207, já que este é o modelo utilizado pela norma ISO/IEC 15.504.

2.3.1 Estudo da viabilidade

Partindo de um esboço da descrição do sistema, de seus benefícios e de sua utilização dentro de uma organização, é que se realiza o estudo da viabilidade. Uma estimativa é feita para verificar se as necessidades do usuário podem ser satisfeitas usando as tecnologias de software e hardware corrente. É decidido se o sistema proposto terá um custo-efetivo do ponto de vista do negócio e se ele poderá ser desenvolvido conforme as restrições de orçamento [Sommerville, 1995].

Destacam-se aqui as questões relevantes que esse estudo deveria comprometer-se a responder [Sommerville, 2001]:

1) O sistema contribui para o completo objetivo da organização?
2) O sistema pode ser implementado usando uma tecnologia corrente e dentro das restrições de custos e cronograma?
3) O sistema pode ser integrado com outro que já está em funcionamento?

Ao final dessa primeira etapa tem-se um relatório decisivo que sugere quanto à continuidade ou não do processo de desenvolvimento do sistema proposto.

2.3.2 Elicitação de Requisitos e Análise

O objetivo da Elicitação é descobrir e entender precisamente quais problemas precisam ser resolvidos e identificar os requisitos do sistema. Portanto é uma atividade envolvida com a descoberta do domínio da aplicação, de quais serviços o sistema deveria prover, do desempenho requerido pelo sistema, das restrições de hardware e assim por diante [Sommerville, 2001].

No momento da exploração e aquisição dos requisitos é importante considerar suas inúmeras fontes de informação e identificar e avaliar todas aquelas de grande potencial. Os principais pontos levantados por Sawyer & Kotonya [2001] são:

- *Metas*: o termo meta refere-se aos objetivos globais do sistema. Muitas vezes as metas são vagamente formuladas e precisam ser cuidadosamente analisadas pelos analistas quanto à sua prioridade e custo.

- *Conhecimento do domínio*: os analistas de sistema precisam adquirir ou ter grande conhecimento do domínio da aplicação para que possam deduzir sobre informações mal articuladas, avaliar sobre os problemas de requisitos conflitantes e outros eventuais casos.
- *Stakeholders do sistema*: os analistas de sistema precisam identificar e gerenciar o "ponto de vista" de diferentes tipos de *stakeholders*.
- *O ambiente operacional*: os requisitos são derivados do ambiente em que o novo sistema será executado, assim este deve ser altamente considerado, pois pode afetar a viabilidade do sistema, o seu custo ou restringir as escolhas do projeto.
- *O ambiente organizacional*: ao construir sistemas de apoio ao negócio, normalmente condicionado pela estrutura, cultura e política da organização, os analistas de sistemas devem levar em conta esses aspectos para que o novo sistema não force uma mudança não planejada no processo de negócio.

A necessidade de grande troca de informações durante a Elicitação de Requisitos faz com esta fase seja caracterizada por uma estreita interação com os clientes, usuários finais e outros envolvidos com o sistema, sendo que aqui são identificados os seus diferentes *stakeholders*.

Essa intensa comunicação envolvida no processo de Elicitação de Requisitos não é uma tarefa simples na qual os engenheiros de requisitos coletam informações/requisitos e os documentam. Geralmente problemas técnicos e sociais estão envolvidos com esse processo devido a inúmeras razões, os quais podem ser representados pela dificuldade dos *stakeholders* em expressar conhecimentos implícitos, pela falta de entendimento do negócio, pelas limitações tecnológicas nos requisitos solicitados, pelos requisitos conflitantes e assim por diante.

Os problemas encontrados na Elicitação de Requisitos são agrupados em três categorias por Christel & Kang [1992]:

- *Problemas de escopo*: acontecem quando o limite do sistema é mal definido ou informações desnecessárias são dadas.
- *Problemas de entendimento*: podem levar a requisitos que são ambíguos, incompletos, inconsistentes e até mesmo incorretos, porque eles não endereçam as verdadeiras necessidades dos *stakeholders*.
- *Problemas de volatilidade*: ocorre porque os requisitos evoluem o tempo todo. Durante o desenvolvimento do sistema as necessidades do usuário podem tornar-se mais claras e amadurecidas porque um maior conhecimento é obtido pelas atividades de

desenvolvimento, ou porque elas podem ser substituídas por um novo conjunto de necessidades causadas por uma dificuldade ambiental ou organizacional inesperada.

Algumas técnicas usadas no processo de Elicitação para facilitar o trabalho dos analistas que coletam os requisitos incluem: Entrevistas, Questionários, *Focus Groups*, Cenários, Observação e Análise Social, Uso de Protótipo, Análise de Documento, *Brainstorming*, JAD, etc [Goguen & Linde, 1993; Kotonya & Sommerville, 1998].

Uma vez coletados os requisitos eles precisam ser analisados para verificar se há problemas, pois os conflitos são inevitáveis quando as necessidades de múltiplos *stakeholders* são consideradas e suas diferentes percepções sobre o sistema são analisadas. Assim, tem-se que a atividade de Análise de Requisitos está intimamente relacionada com a atividade de Elicitação, sendo que elas são freqüentemente intercaladas.

A Análise de Requisitos é feita com o objetivo de checar o conjunto de requisitos elicitados a fim de detectar e resolver conflitos entre eles, descobrir os limites do sistema e como ele deve interagir com seu ambiente. Também são elaborados requisitos do sistema para os requisitos do software e então é feita a negociação das mudanças necessárias para satisfazer todos os *stakeholders* do sistema [Kotonya & Sommerville, 1998; Sawyer & Kotonya, 2001]. Durante a análise, requisitos faltantes, conflitantes, ambíguos e que se sobrepõem são, normalmente, descobertos. Os problemas são reconhecidos e os *stakeholders* devem negociá-los e concordar em modificar ou simplificar os requisitos. Também pode ocorrer que novos requisitos sejam encontrados.

A Análise de Requisitos também tem por finalidade desenvolver informações adicionais tais como custo dos requisitos, os seus benefícios, relação com outros requisitos, etc. Suas atividades comuns incluem o uso de *checklists* para analisar os requisitos, a priorização e classificação dos requisitos, o uso de matrizes de interação para encontrar conflitos e sobreposições e uma avaliação dos riscos dos requisitos [Sommerville & Sawyer, 1997].

Algumas questões básicas que acontecem durante o processo de elicitação e análise são levantadas por Faulk [1997]:

- Como coletar efetivamente um conjunto completo de requisitos do cliente ou outras fontes?
- Como decompor o problema em partes intelectualmente gerenciáveis?
- Como organizar as informações para que possam ser entendíveis?
- Como comunicar sobre o problema com todas as partes envolvidas?

- Como resolver necessidades conflitantes?
- Como saber quando parar?

2.3.3 Especificação de Requisitos

Terminada a etapa de Elicitação e Análise, previamente descrita, um DR deve ser gerado. Esse é o artefato final produzido no Processo de Engenharia de Requisitos e o seu objetivo principal. Nele são especificados, de maneira completa, precisa e verificável, os requisitos, o comportamento, ou outras características de um sistema ou componente e, freqüentemente, os procedimentos para determinar se estas prescrições foram satisfeitas [IEEE, 1990]. O DR deve conter uma descrição de alta qualidade do sistema que está sendo considerado, expondo o que deve ser feito, porém sem descrever como fazê-lo ou como alcançá-lo [Davis, 1993a].

Em termos gerais, o DR representa uma ponte de comunicação entre os requisitos do cliente e a comunidade técnica, descreve o que o cliente espera que o sistema faça, o ambiente esperado do sistema, o perfil de uso, os parâmetros de desempenho e a qualidade e efetividade esperada [IEEE, 1998a]. Portanto, esse documento deve atender as necessidades dos usuários, ser compreensível para os usuários, clientes, analistas de sistemas e testadores e capaz de servir como base para as fases seguintes do desenvolvimento de software.

Com descrição precisa e detalhada das características esperadas de um sistema de software e de todo seu comportamento externamente observáveis, esse documento pode desempenhar uma variedade de funções [Faulk, 1997; Sawyer & Kotonya, 2001]:

- Para clientes, os requisitos tipicamente documentam o que deveria ser entregue e pode ser utilizado para estabelecer um contrato oficial entre o cliente e os analistas de sistemas.
- Para gerentes, ele provê uma base realista para estimar o custo do produto, os riscos e o cronograma e oferece um critério para medir o progresso de desenvolvimento e averiguar o que resta a ser feito.
- Para projetistas, ele pode servir de base para a especificação de projeto.
- Para codificadores, ele define o conjunto de implementações aceitáveis e é a autoridade final sobre as saídas que devem ser produzidas.
- Para o pessoal de garantia de qualidade, ele proporciona a base para a validação, planos de teste e verificação. É o artefato principal do testador para determinar o comportamento aceitável do software.

- Para os mantenedores, ele oferece a definição padrão do comportamento esperado do sistema e é usado para registrar as mudanças de engenharia.
- Esse documento força uma avaliação rigorosa dos requisitos antes de se começar o projeto, reduzindo possivelmente um re-projeto.
- Serve de base para uma posterior expansão do software, uma vez que ele discute o produto e não o projeto que o desenvolveu.
- Providencia uma base informativa que facilita a transferência de um produto de software para novos usuários ou novas máquinas.
- Serve de base para determinar quais requisitos estão completos e quais necessitam de uma análise adicional.
- O DR define quais propriedades o sistema deve ter e as restrições em seu projeto e implementação. Define os pontos em que há e não há liberdade de projeto. Ajuda a garantir que as decisões sobre os requisitos sejam feitas explicitamente durante a fase de requisitos e não implicitamente durante a programação.
- Também pode ser usado por diversos grupos tais como reguladores governamentais, marketing, etc.

Pelas inúmeras finalidades para o qual o DR serve, fica evidente a sua importância em todos os aspectos relacionados ao desenvolvimento de software. Assim, tem-se que a descrição desse documento deve ser de mais alta qualidade.

Os problemas na documentação podem induzir não só a riscos no projeto como também afetar a qualidade global de seu produto, não restando dúvida da necessidade de se melhorar a forma com que uma especificação de requisitos é documentada.

O Capítulo 3 descreve com mais detalhes aspectos relacionados ao DR, artefato principal de pesquisa deste trabalho.

2.3.4 Validação de Requisitos

Após a elaboração do DR, ele é avaliado para ser checado quanto à sua consistência, completitude e precisão e para detecção de omissões, erros ou outros problemas que possam ser corrigidos antes que seja firmado um compromisso em relação aos requisitos definidos para o projeto. Assim, os requisitos são validados para garantir que eles não tenham sido declarados de modo ambíguo, inconsistente, incompleto, incorreto e mostrar que eles realmente definem o que o cliente solicitou e, portanto, para certificar-se de que representam uma descrição aceitável do sistema que será implementado [Sommerville, 2001].

O ideal seria que o DR apresentasse somente os requisitos pretendidos pelos *stakehoders*, porém problemas inevitavelmente são descobertos durante a validação de requisitos e eles devem ser corrigidos. Exemplo desses problemas de requisitos identificados na fase de validação podem ser [Kotonya & Sommerville, 1998]:

- Falta de conformidade com o padrão de qualidade;
- Requisitos pobremente formulados gerando ambigüidade;
- Erros nos modelos do sistema;
- Requisitos conflitantes que não foram detectados durante o processo de análise.

A principal dificuldade dessa etapa é que não existe algum outro documento para que os requisitos possam ser contrapostos, ou seja, não existe outro documento que possa ser uma base para a validação, somente as intenções na mente das pessoas. Mas a validação ainda é o meio de garantir que o DR defina o sistema correto (o sistema que o usuário espera) e apresente uma descrição clara.

Algumas técnicas de validação incluem [Sommerville, 2001]: revisão de requisitos, uso de protótipo, geração de casos de teste, análise automatizada de consistência (usando ferramenta CASE). No Capítulo 4 será comentada a técnica de inspeção de DR, abordando principalmente as Técnicas de Leituras que auxiliam a inspeção deste documento.

2.3.5 Gerenciamento de Requisitos

Esta etapa da Engenharia de Requisitos refere-se ao conjunto de procedimentos que visa gerenciar as mudanças que ocorrem nos requisitos do sistema durante o processo de desenvolvimento, uma vez que os requisitos podem sofrer alterações e outros novos podem surgir. A falta de mudanças documentadas e controladas pode ocasionar sérios problemas para o desenvolvedor, principalmente em relação à manutenção de requisitos. Para tentar minimizar estas dificuldades é que se faz necessário o gerenciamento deles.

Algumas das razões pelas quais novos requisitos surgem ou se modificam são [Kotonya & Sommerville, 1998]:

- Um sistema de grande porte geralmente tem diversos usuários com diferentes requisitos e prioridades, os quais podem ser contraditórios ou conflitantes. Para manter o compromisso com todos eles, alguns requisitos finais do sistema precisam ser mudados.

- Devido às restrições financeiras e organizacionais, o cliente do sistema impõe determinados requisitos que são conflitantes com os requisitos do usuário final.

- O ambiente técnico e de negócio do sistema mudam e estes devem ser refletidos no próprio sistema. Novos hardware podem ser necessários, prioridades de negócio podem alterar, novas legislações e regulamentos podem ser introduzidos.
- Conforme o cliente adquire maior compreensão de suas reais necessidades no decorrer do desenvolvimento do sistema, ele pode requerer outras funcionalidades ou modificações.
- Mudanças nos requisitos também podem ocorrer devido a erros e mal-entendimento no Processo de Engenharia de Requisitos, projeto ou problemas na implementação de um requisito.

Essas mudanças procedem desde a etapa de elicitação, passando pela análise, validação e até mesmo depois que o sistema esteja operando. Sob a ótica da evolução de requisitos, esses podem ser divididos em duas classes [Sommerville, 2001]:

Requisitos permanentes: são requisitos relativamente estáveis derivados das atividades centrais da organização e relacionados diretamente com o domínio do sistema.

Requisitos voláteis: são requisitos que têm uma alta probabilidade de mudar durante o desenvolvimento do sistema ou depois que o sistema tenha sido colocado em operação. São específicos à instanciação do sistema em um ambiente particular e para um cliente particular.

O gerenciamento dos requisitos requer que informações relevantes sobre o requisito sejam armazenadas e que a rastreabilidade entre requisitos relacionados seja garantida, assim é possível rastrear a "vida" de um requisito em ambas as direções, tanto para frente quanto para trás. Para isso é fundamental que eles sejam unicamente identificados.

Gerenciar e controlar mudanças nos requisitos são outras importantes partes desta etapa. Elas dizem respeito aos procedimentos, processos e padrões que são usados para gerenciar os requisitos do sistema. Seu processo constitui de um conjunto de atividades que avaliam o impacto e o custo das mudanças.

É muito importante lembrar que o gerenciamento de requisitos é a primeira área-chave de processo que deve ser cumprida para uma organização atingir o nível 2 do CMM *(Capability Maturity Model)*. O CMM é um modelo que define a maturidade que uma organização possui para desenvolver software [Paulk, 1995]. Enfatizando a importância do gerenciamento de requisitos para alcançar a qualidade nos processos de software é que [Carvalho et al., 2001] definiu uma estratégia para sua implantação. Tal estratégia consistiu na definição dos processos para gerência de requisitos que foram divididos em: Definição, Gerenciamento & Métricas, Validação e Verificação. As atividades de cada processo e suas entradas e saídas foram descritas. Para a implantação desse processo de melhoria foram

estabelecidas 9 fases: Fase 1 – Conscientização, Fase 2 – Preparação, Fase 3 – Treinamento, Fase 4 – Levantamento de Processos de Engenharia de Requisitos existentes, Fase 5 – Elaboração do Plano de Melhoria, Fase 6 – Definição dos Processos, Fase 7 – Definição de Papéis e Responsabilidades, Fase 8 – Implantação, Fase 9 – Acompanhamento e Ajustes. Um relacionamento entre essa estratégia e o CMM foi descrito, mostrando as práticas-chaves para gerência de requisitos do CMM e em qual parte delas os processos e as fases da estratégia são executadas.

Compreendida as etapas do Processo de Engenharia de Requisitos, fica fácil entender a sua importância na definição de sistemas. Alguns relatos sobre esse processo são mostrados na seção seguinte.

2.4 Trabalhos Relacionados ao Processo de Engenharia de Requisitos

Diversos estudos têm sido realizados abrangendo a Engenharia de Requisitos. Nesta seção serão comentados alguns deles, mostrando as pesquisas mais recentes desta área. Para facilitar a leitura, cada trabalho será citado pelo seu título, autor e ano de publicação.

Capturing the Benefits of Requirements Engineering [Sawyer et al., 1999]

Este artigo mostra um guia de boas práticas de Engenharia de Requisitos em forma de diretrizes, o REGPG (*Requirements Engineering Good Practice Guide*). Este guia foi desenvolvido com base num modelo SPI (*Software Process Improvement*) existente a fim de definir um *framework* de melhoria de processo. Assim como o CMM, que apresenta níveis de maturidade de processo, o REGPG também apresenta, porém apenas três níveis: Inicial (a organização tem um processo de requisitos *ad hoc*), Repetível (a organização tem um padrão definido para documentar requisitos e tem uma política e procedimentos para gerenciamento de requisitos) e Definido (a organização tem um modelo de processo definido baseado em boas práticas e métodos definidos). As diretrizes apresentam um *checklist* de questões que permitem classificar um processo de requisitos num determinado nível. O REGPG abrange 66 boas práticas em forma de diretrizes, e são classificadas em práticas básicas, intermediárias e avançadas.

Requirements Engineering as a Success Factor in Software Projects [Hofmann & Lehner, 2001]

Um estudo feito com quinze equipes de Engenharia de Requisitos apoiando o desenvolvimento de projetos de software personalizados na área de telecomunicações e banco, e soluções de prateleira (soluções prontas), identificou as práticas da Engenharia de Requisitos que contribuem para o sucesso de um projeto. As melhores práticas foram apresentadas em termos de conhecimento do domínio da aplicação, da alocação de recursos e do processo utilizado. A Tabela 2.1 resume o estudo realizado:

Tabela 2-1 Melhores práticas utilizadas pela maioria das equipes de Engenharia de Requisitos de sucesso [Adaptado de Hofmann & Lehner, 2001].

Área focada	Melhores práticas
Conhecimento	Envolver clientes e usuários durante a Engenharia de Requisitos.
Conhecimento	Identificar e consultar todas as possíveis fontes de requisitos.
Conhecimento	Designar gerentes de projeto com habilidade na área e membros da equipe para as atividades da Engenharia de Requisitos.
Recursos	Alocar de 15 a 30% dos investimentos totais do projeto para as atividades da Engenharia de Requisitos.
Recursos	Prover modelos de especificação e exemplos.
Recursos	Manter um bom relacionamento entre os *stakeholders*.
Processo	Priorizar requisitos.
Processo	Desenvolver modelos complementares junto com protótipos.
Processo	Manter uma matriz de rastreabilidade.
Processo	Usar *peer review*, cenários e *walkthroughs* para validar e verificar requisitos.

Requirements Problems in Twelve Software Companies: An Empirical Analysis [Hall et al., 2002]

Com a finalidade de entender o Processo de Engenharia de Requisitos e enquadrar os problemas relacionados a requisitos, as dificuldades enfrentadas por doze companhias de software em seus Processos de Engenharia de Requisitos foram coletados. Compreendendo-se tais problemas, seria possível oferecer aos gerentes de projeto um plano que pudesse delinear um processo de requisitos mais efetivo. Três questões foram alvo de pesquisa, sendo elas: Que padrões de problemas relacionados a processo de requisitos as companhias estão experimentando?, O aumento no processo de maturidade reduz os problemas relacionados ao processo de requisitos?, Diferentes equipes de trabalho relatam diferentes problemas de requisitos?

O estudo concluiu que a maioria dos problemas era organizacional e não técnico, e que há uma relação entre a maturidade de uma companhia e padrões de problemas de requisitos.

Alguns tipos de problemas de requisitos eram mais freqüentes naquelas companhias que apresentavam menor maturidade no seu processo de desenvolvimento de software.

Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice [Kauppinen et al., 2002]

Colocar em prática o Processo de Engenharia de Requisitos não é tão simples. Isso porque para implantar um novo programa é necessário que haja comprometimento de todos, ou seja, a sua introdução requer não somente uma mudança de processo e de tecnologia, mas também mudança de comportamento e de cultura de uma organização e um massivo treinamento no processo adotado [Jacobs, 1999]. Wiegers [1994] diz que a cultura de uma organização é um fator crítico de sucesso em seus esforços de melhoria de processo. É sobre os fatores que apóiam essa mudança cultural e os obstáculos que o impedem que este artigo relata. Em um estudo realizado em quatro organizações finlandesas que estavam iniciando as práticas da Engenharia de Requisitos no desenvolvimento de seus produtos foram identificados cinco fatores que apóiam a mudança cultural na prática:

- Ligar os objetivos do negócio com requisitos técnicos através das necessidades dos usuários e requisitos dos usuários;
- Integrar um modelo de Processo de Engenharia de Requisitos simples com o processo de desenvolvimento de produto;
- Elicitar necessidades do usuário ativamente, com mais intensidade;
- Representar requisitos do usuário sistematicamente na forma de casos de uso;
- Aumentar a consciência das pessoas sobre Engenharia de Requisitos;

Quanto aos obstáculos que impedem a mudança cultural de ocorrer na prática, estes estão listados na Tabela 2.2. Junto são mostradas as crenças, que são exemplos de obstáculos que impedem os desenvolvedores de mudarem sua maneira de pensar e comportar, e as respostas a eles.

Tabela 2-2 Crenças relacionadas aos obstáculos da mudança cultural e as respostas para esses obstáculos [Kauppinen et al., 2002].

Obstáculos	Crenças	Respostas
Não vale a pena descobrir necessidades diretamente dos usuários. Ou seja, por que descobrir necessidades diretamente de usuários reais?	- Desenvolvedores dizem que desenvolvem certos tipos de produto a longo tempo e já sabem as necessidades dos usuários. - Desenvolvedores também usam seus próprios produtos, portanto sabem agir como usuário. - Desenvolvedores estão desenvolvendo um novo produto, portanto usuários não podem ter alguma necessidade para esse produto.	1) Quanto mais profundamente as necessidades dos usuários são entendidas, mais útil e usável pode ser o sistema desenvolvido. 2) Um sistema útil e usável apóia os objetivos e tarefas dos usuários. 3) Usuários são peritos em suas tarefas, portanto eles são as fontes primárias das suas necessidades reais.
É difícil e arriscado descobrir as necessidades diretamente de usuários. Ou seja, como descobrir necessidades diretamente de usuários?	- Desenvolvedores acreditam que os usuários são incapazes de dizer o que eles necessitam e querem. - Há tantos usuários que os desenvolvedores não podem entrevistar todos eles. - Desenvolvedores dizem que não podem mostrar a seus clientes que eles realmente não sabem o básico de seus negócios. - Gerentes dizem que os desenvolvedores podem prejudicar relações com o cliente ao perguntar questões sem sentido/bobas.	1) Há técnicas de elicitação, como entrevista e observação, que desenvolvedores podem aprender e usar facilmente. 2) Ao combinar técnicas de entrevista e observação é possível conseguir um retrato compreensível das necessidades dos usuários sem requerer que eles articulem explicitamente suas necessidades. 3) Uma visita bem preparada ao usuário pode melhorar a imagem da companhia entre seus clientes e pode criar uma ponte de competitividade.
Não vale a pena documentar requisitos de usuários sistematicamente. Ou seja, por que documentar requisitos de usuário sistematicamente?	- Gerentes e desenvolvedores falam que os clientes estão interessados em requisitos técnicos e em detalhes da interface do usuário. - Desenvolvedores dizem não ter tempo para documentar requisitos do usuário.	1) Documentação sistemática de requisitos de usuários garante que um produto correto será desenvolvido. 2) Requisitos de usuários bem documentados são informações úteis para muitos grupos: projetistas, testadores, escritores de manuais de usuários, gerentes e pessoal de marketing. 3) Documentação sistemática de requisitos de usuários no começo do desenvolvimento do produto poupa tempo e diminui o re-trabalho em fases posteriores de um projeto.

Enquanto estes obstáculos perduram, tem-se observado até hoje a construção de produtos de software com seus cronogramas atrasados, com custo maior do que o esperado e que apresentam defeitos, resultando em insatisfação e inconveniências para os usuários.

Requirements Engineering Process Models in Practice [Martin et al., 2002]

A pesquisa apresentada neste artigo teve como objetivo analisar os modelos de Processo de Engenharia de Requisitos utilizados na prática e compará-los com os modelos

existentes na literatura. O propósito dessa análise era entender as práticas correntes a fim de melhorar os Processos de Engenharia de Requisitos. Para isso, três modelos foram selecionados da literatura, os quais se diferenciaram pela sua estrutura: linear, linear com iterações entre atividades e iterativo. O estudo foi realizado em duas companhias australianas, o qual revelou que os modelos de Processos de Engenharia de Requisitos comumente aceitos na literatura diferem dos modelos aplicados na prática. Esta constatação foi feita ao se construir modelos de alto nível do processo de engenharia praticados em seis projetos, três de cada companhia, e compará-los com os três modelos selecionados. Foi percebido que o Processo de Engenharia de Requisitos se beneficiaria de uma combinação entre uma estrutura linear e iterativa. Nem um modelo puramente linear seria tão bem sucedido quanto um que mostrasse iterações de atividades. E ainda, nem mesmo numa mesma companhia seria possível utilizar um único modelo representando o Processo de Engenharia de Requisitos, visto que o processo é dependente de determinada situação.

An Industrial Case Study of the Impact of Requirements Engineering on Downstream Development [Damian et al., 2003]

Os benefícios em longo prazo das práticas da Engenharia de Requisitos foram os resultados obtidos de uma avaliação para investigar o Processo de Engenharia de Requisitos na prática industrial realizada numa organização de desenvolvimento de software chamada *Australian Center for Unisys Software* (ACUS), a qual estava se submetendo a uma melhoria em seu processo com foco centrado no Processo de Engenharia de Requisitos [Damian et al., 2003]. Nesse estudo de caso, no qual a investigação durou um período de dezoito meses (durante todo o ciclo de vida do projeto), foram revelados uma forte relação entre um processo de requisitos bem definido e o aumento da produtividade do desenvolvedor, uma melhora no planejamento de projeto através de melhores estimativas e uma elevada habilidade dos *stakeholders* em negociar o escopo do projeto.

Effective Requirement Specification as a Precondition for Successful Software Development Project [Pozgaj et al., 2003]

Neste artigo os autores citam alguns problemas relacionados à definição de requisitos de software: definição de má qualidade, seleção errada de requisitos e as mudanças de requisitos que ocorrem durante o processo de desenvolvimento. De forma a resolver esses

problemas são propostas metodologias para o seu gerenciamento, focando o ciclo de vida completo do requisito, desde a descoberta até o seu efetivo uso. Três metodologias foram definidas com as respectivas atividades a serem executadas para um eficiente gerenciamento de requisitos. São elas: Metodologia para descoberta e descrição de requisitos, na qual cada membro do projeto é responsável pela descoberta e definição dos requisitos de acordo com sua área de competência, pois se subentende que os requisitos serão entendidos e coletados melhores; Metodologia para organização dos requisitos, cujas atividades devem ser desempenhadas para se criar uma infraestrutura de requisito adequada para todas as fases do desenvolvimento do projeto e Metodologia para gerenciamento de mudança de requisito, que melhora a análise de requisitos e foca no impacto de mudança dos requisitos para completar a arquitetura do sistema.

An industrial case study of immediate benefits of requirements engineering process improvement at the Australian Center for Unisys Software [Damian et al., 2004]

Uma avaliação da melhoria do Processo de Engenharia de Requisitos em um dos laboratórios de desenvolvimento da ACUS (*Australian Center for Unisys Software*) é descrita neste artigo. A pesquisa foi conduzida durante um período de oito meses, sendo que os métodos utilizados incluíram dois questionários, inspeção de documento de requisitos, observações da análise de requisitos e sessões de negociação, além de entrevista com os engenheiros e equipe de gerenciamento da ACUS.

O foco principal da melhoria estava no processo de gerenciamento de requisitos, uma vez que este era um dos grandes desafios enfrentados pela organização. Mais especificamente, as dificuldades para satisfazer as necessidades do mercado e os requisitos de estratégia de produto estavam relacionados a: requisitos não claramente definidos e documentados; requisitos não completamente compreensíveis pelos desenvolvedores; dificuldade em entender claramente a direção do mercado/negócio e alinhá-la à estratégia de tecnologia e em prover estimativa de esforço de desenvolvimento, uma vez que as características de alto nível do sistema não forneciam informações suficientes; prática inadequada de gerenciamento de requisitos.

A estratégia para a melhoria foi adequar a prática da Engenharia de Requisitos utilizada para atender às necessidades da ACUS. Esta estratégia incluiu cinco passos: 1) Definir uma política de alto nível e estabelecer um acordo com gerentes e *stakeholders*; 2) Revisar e refinar o Processo de Engenharia de Requisitos para contornar os problemas

enfrentados; 3) Auxiliar o Processo com métodos e ferramentas apropriadas; 4) Prover treinamento e capacidade na utilização do Processo revisado, e nos métodos e ferramentas associados; 5) Praticar a revisão e avaliação para o contínuo melhoramento.

A avaliação da melhoria no Processo de Engenharia de Requisitos revelou benefícios logo nos primeiros estágios de desenvolvimento, sendo que os requisitos foram definidos mais claramente, entendidos e especificados melhores, e a ACUS mostrou uma elevada habilidade em endereçar as necessidades do mercado e requisitos de estratégia do produto.

Requirements Engineering Practice in Pharmaceutical and Healthcare Manufacturing [Prakash et al., 2004]

Vista a importância dos sistemas de manufatura médica e farmacêutica, uma vez que esses são considerados sistemas de segurança crítica que podem ameaçar a vida humana, as práticas do Processo de Engenharia de Requisitos foram examinadas em três companhias multinacionais australianas dessa área. O estudo analisou seis projetos, dois de cada companhia, e mostrou inconsistência no Processo de Engenharia de Requisitos usado, sendo que cada companhia seguia um processo diferente embora produzissem produtos similares. Determinadas atividades da Engenharia de Requisitos eram executadas de forma implícita, sendo o conhecimento centralizado nas pessoas, elevando os riscos envolvidos quando não se tem um processo explícito e definido, principalmente em sistemas como este que são cruciais e devem prover produtos medicinais precisos e exatos. O esforço dedicado às atividades do Processo de Engenharia de Requisitos também se apresentou inconsistente, sendo eles diferentes em cada companhia. Com essa pesquisa, ficou clara a necessidade de se adotar as melhores práticas de um modelo de Processo de Engenharia de Requisitos, deixando explícitas as atividades desse processo, o que é de grande importância para esse domínio de aplicação.

Combining Requirements Engineering Techniques - Theory and Case Study [Jiang et al., 2005]

Como forma de melhorar o Processo de Engenharia de Requisitos é proposta uma combinação de suas técnicas. O artigo menciona que grande parte dos insucessos dos projetos de software está relacionada a uma péssima prática da Engenharia de Requisitos e ainda à seleção inadequada de técnicas para elicitação, modelagem, documentação, verificação e

validação de requisitos. Porém, a escolha de uma única técnica nem sempre é a mais apropriada, havendo necessidade de combinação delas. Sendo assim, o artigo mostra a metodologia MRETS (*Methodology for Requirements Engineering Techniques Selection*), a qual é um processo sistemático para a seleção de uma combinação de técnicas de Engenharia de Requisitos baseada nos atributos do projeto e características das técnicas de Engenharia de Requisitos. Para isso, MRETS é auxiliada por uma biblioteca de técnicas de requisitos. Um dos casos de uso conduzidos para examinar o mérito da combinação das técnicas de Engenharia de Requisitos é relatado. O resultado foi positivo, demonstrando os benefícios dessa combinação, como: as mudanças nos requisitos foram menores quando comparados a projetos similares previamente realizados pela companhia; nenhum requisito de grande impacto foi adicionado ou removido; ajudou a descobrir e corrigir mais requisitos antecipadamente. A vantagem do uso das técnicas recomendadas reduziu o re-trabalho, pesando mais que o custo associado com o treinamento e o tempo gasto na sua aplicação.

Como se pode perceber, pelos diversos trabalhos comentados, a prática da Engenharia de Requisitos não é uma tarefa fácil, principalmente porque requer a aplicação de um processo sistemático, com uma série de atividades entrelaçadas; há o envolvimento de múltiplas partes, tendo cada uma diferentes conhecimentos, habilidades, experiências e percepções, o que pode gerar entendimento errôneo do negócio e dos requisitos do sistema; há uma variedade de preocupações a serem consideradas, além da funcional, tais como segurança, usabilidade, desempenho, interoperabilidade, manutenibilidade. Com isso, a especificação de requisitos pode apresentar uma variedade de deficiências [Lamsweerde, 2000; Damian, 2000].

Porém, os trabalhos brevemente descritos mostram o crescente reconhecimento da Engenharia de Requisitos como uma importante atividade no processo de desenvolvimento de software e sua grande influência para o sucesso de um projeto de software. Como se pode observar, pesquisas e estudos de casos relacionados ao Processo de Engenharia de Requisitos têm sido freqüentemente realizados com o objetivo de melhorá-lo.

2.5 Considerações Finais

Neste capítulo foram apresentados os princípios da Engenharia de Requisitos, a qual tem como artefato final o DR, cujo presente trabalho irá explorar.

Para melhor entender e caracterizar o que é requisito de software, a sua definição foi descrita. Em seguida, detalhes de cada etapa do Processo de Engenharia de Requisitos foram explicados, que englobam: estudo da viabilidade do sistema, elicitação de requisitos e análise, especificação, validação e gerenciamento de requisitos.

Todo o processo que leva à concretização do DR foi mencionado com o intuito de mostrar que a sua elaboração requer fases a serem seguidas e, portanto, não é um simples documento no qual as informações são expressas.

Também foram comentados diversos estudos e pesquisas que envolvem o Processo de Engenharia de Requisitos, o que ressalta a importância dessa fase inicial de especificação de requisitos. Nesses trabalhos ficam claros que um sólido Processo de Engenharia de Requisitos ainda é a melhor solução existente para garantir uma especificação de software conforme as necessidades do cliente e que alcance as suas expectativas. É fato que quanto maior investimento nesta primeira fase do ciclo de vida do software maior é a probabilidade de se construir um sistema adequado às exigências solicitadas e menor é o risco de se enfrentar os problemas típicos quando se apressa em começar logo a implementação. Os trabalhos pesquisados contribuíram para mostrar a necessidade de se ter instruções de auxílio para melhor especificar requisitos de um sistema e definir o quão bom um DR está descrito. Como este presente trabalho está focado no DR, o capítulo seguinte apresentará diversos aspectos relacionado a ele.

Capítulo 3

3. Documento de Requisitos

3.1 Considerações Iniciais

Como visto no capítulo anterior, o DR é um documento importante que agrega a maior parte das informações necessárias para gerar subseqüentes artefatos. O impacto de sua definição errônea é o princípio pelo qual um esforço maior deve ser dado a ele. De forma a facilitar elaboração do DR, alguns padrões para sua especificação foram desenvolvidos. Esses padrões procuram definir os principais tópicos ou seções necessárias para o entendimento e posterior implementação do sistema a que ele é designado. Normalmente, a finalidade de cada seção contida no padrão é explicada.

O uso de um padrão para especificar requisitos contribui para a organização de suas informações e facilita a identificação dos dados principais que devem ser descritos no DR, evitando que eles sejam esquecidos. Porém, apenas a utilização do padrão não garante que a qualidade do DR seja boa. Sendo assim, vários estudos surgiram para identificar quais características um DR deve possuir a fim de que este se apresente com as mais completas, consistentes e precisas informações.

Assim, este capítulo apresenta essas questões relacionadas ao DR, estando ele organizado da seguinte maneira: na Seção 3.2 é dada uma visão geral do que seriam padrões de especificação usados como esqueleto para documentar os requisitos, na Seção 3.3 são apresentadas as definições de cada propriedade de qualidade que um DR deve apresentar, de forma que este se torne ideal, na Seção 3.4 comentam-se algumas Diretrizes recomendadas por diversos autores para facilitar a elaboração do DR e, finalmente, na Seção 3.5 apresentam-se as Considerações Finais.

3.2 Padrões de Especificação de Documento de Requisitos

Vários padrões foram propostos para organizar o conteúdo do DR, cada um com uma estrutura de organização, servindo como modelo para se documentar requisitos e assim guiar

na sua elaboração. Não há um padrão para todos os projetos em todas as indústrias, pois os requisitos individuais são únicos não somente de companhia para companhia, mas também de projeto para projeto dentro de uma companhia. O ideal é escolher e ajustar o modelo que melhor se adapte às necessidades de um projeto em particular.

A maioria destes padrões é militar em oposição aos padrões "comerciais". Alguns exemplos de padrões são listados abaixo [Tran, 1999]:

- Volere [2004]
- IEEE Recommended Practice for Software Requirements Specifications [IEEE, 1998b]
- Software Requirements Specification Document Template [CSDL, 1996]
- British Standard Guide to Specifying User Requirements for a Computer-Based Standard (BS6719 - 1986)
- Canadian Standard, Basic Guidelines for the Structure of Documentation of System Design Information (Z242.15.4-1979)
- Military Standard Specification Practices. MIL-STD-490A. U.S. Department of Defense, 4 June 1985.
- System/Segment Specification (DI-CMAN-80008A, 2/29/1988)
- Software Requirements Specification (DI-MCCR-80025A, 2/29/1988)
- Interface Requirements Specification (DI-MCCR-80026A, 2/29/1998)

Esses padrões exibem os principais tópicos que um DR deve conter, explicando o que significa cada seção presente, porém não mostram como eles devem ser escritos. O objetivo é apresentar como deve ser a estrutura do DR a fim de caracterizá-lo e facilitar a busca de informações através de suas seções. Como os padrões comumente referenciados em estudos são o IEEE [1998b] e Volere [2004], estes serão comentados especificamente.

O padrão IEEE [1998b] apresenta uma estrutura que permite organizar os requisitos de forma que sejam mais adequados para o seu entendimento, uma vez que classes de sistemas diferentes levam a organizá-los de formas diferentes. Assim, a organização da seção Requisitos Específicos pode ser:

- por modo de operação do sistema. Por exemplo, treinamento, normal ou emergencial;
- por classes de usuários diferentes, pois alguns sistemas podem oferecer um conjunto de funções para classes de usuários diferentes;

- por objetos, que são entidades do mundo real que têm uma ligação com o sistema. Por exemplo, em sistema de controle de estacionamento os objetos incluem sensor de passagem, máquina de emissão de ingresso, leitor de ingresso, motorista, funcionário, etc.;
- por característica, que é um serviço externamente desejado pelo sistema, o qual pode requerer uma seqüência de entradas para realizar o resultado desejado. Por exemplo, em um sistema de telefonia, as características incluem chamada local, direcionamento de chamada, chamada por conferência;
- por estímulo recebido. Por exemplo, sistema de aterrissagem automática por ser organizados em seções para perda de velocidade, ventania, excesso de velocidade, etc.;
- por resposta, assim todas as funções que dão suporte à geração de uma determinada resposta são agrupadas. Por exemplo, os requisitos de uma sistema de departamento pessoal podem ser organizados em seções correspondentes a todas os requisitos associados com a geração de folha de pagamento, todos os requisitos associados com a geração de uma lista atual de funcionários, etc.;
- por hierarquia de funções organizada por entradas comuns ou saídas comuns ou acesso a dados internos comuns.

As partes essenciais de uma boa especificação de requisitos de software, segundo o modelo do padrão IEEE [1998b], deveria incluir as seções que são ilustradas pela Figura 3.1

1. Introdução
   - 1.1 Proposta
   - 1.2 Escopo
   - 1.3 Definições, acrônimos e abreviaturas
   - 1.4 Referência
   - 1.5 Visão geral
2. Descrições gerais
   - 2.1 Perspectiva do produto
   - 2.2 Funções do produto
   - 2.3 Características do usuário
   - 2.4 Restrições
   - 2.5 Suposições e dependências
3. Requisitos específicos

Apêndices

Índice

Figura 3.1 Modelo de Especificação de Requisitos de Software da IEEE [Adaptado de IEEE, 1998b].

Uma breve descrição de cada item desse modelo é apresentada a seguir.

1. Introdução

Deve apresentar uma visão geral de todo o DR.

1.1 Proposta

Descreve a proposta do DR e especifica a sua pretensão.

1.2 Escopo

Identifica o software a ser produzido, explica o que ele deve e não deve fazer, descreve a sua aplicação.

1.3 Definições, acrônimos e abreviaturas

Providencia as definições de todos os termos, acrônimos e abreviaturas requeridas para interpretar corretamente o DR.

1.4 Referência

Lista todos os documentos referenciados no DR, identifica cada um e especifica as fontes das quais as referências podem ser obtidas.

1.5 Visão geral

Descreve o que o resto do DR contém e explica como ele está organizado.

2. Descrições gerais

Descreve os fatores gerais que afetam o produto e seus requisitos.

2.1 Perspectiva do produto

Coloca o produto em perspectiva com outros produtos relatados. Se o DR define um produto que faz parte de um grande sistema, é preciso identificar as suas interfaces. Esta seção também deveria descrever como o software opera dentro de várias restrições.

2.2 Funções do produto

Apresenta um resumo das principais funções que o software desempenhará.

2.3 Características do usuário

Descreve as características gerais de um pretendido usuário do produto, incluindo nível educacional, experiência e habilidade técnica.

2.4 Restrições

Provê as descrições gerais de quaisquer outros itens que limitarão as opções do desenvolvedor. Alguns desses itens incluem: limitações de hardware, considerações de segurança, requisitos de confiabilidade, operação paralela, funções de controle e outros.

2.5 Suposições e dependências

Lista cada um dos fatores que afetam os requisitos declarados no DR. Esses fatores não são restrições de projeto do software, mas são quaisquer mudanças nele que podem afetar os requisitos no DR.

3. Requisitos específicos

Contém todos os requisitos de software em detalhes suficientes para permitir ao desenvolvedor projetar um sistema que satisfaz esses requisitos e ao testador testá-los. A Seção 3 é a maior e mais importante do DR, compreendendo vários itens.

3.1. Interfaces externas: é uma descrição detalhada de todas as entradas e saídas do sistema de software.

3.2. Funcionais: definem as ações fundamentais que devem ocorrer no software ao aceitar e processar a entrada e ao processar e gerar a saída.

3.3. Requisitos de Desempenho: especifica os requisitos numéricos estáticos e dinâmicos colocados sobre o software ou sobre a interação humana com o software como um todo.

3.4. Requisitos de banco de dados lógico: especifica os requisitos lógicos para qualquer informação que é colocada no banco de dados.

3.5. Restrições de projeto: especifica as restrições de projeto que podem ser impostas por outros padrões, limitações de hardware, etc.

3.6. Atributos do sistema de software: Confiabilidade, Disponibilidade, Segurança, Manutenibilidade, Portabilidade.

3.7. Organização dos requisitos específicos: de acordo com o tipo de sistema a ser desenvolvido, seus requisitos específicos são organizados de diferentes formas: por modo do sistema, classes de usuário, objetos, características do usuário, estímulo, resposta, hierarquia funcional.

Por fim, o DR pode conter informações de apoio, tais como apêndices e índices. Assim, informações extras do sistema podem ser acrescentadas no apêndice.

Quanto ao modelo de especificação de requisitos do padrão Volere [2004], este é ilustrado pela Figura 3.2.

Especificação de requisitos do sistema.....
Versão.....

Diretivas de Projeto
1. A proposta do produto
2. Cliente, *Customer, Stakeholders*
3. Usuários do produto

Restrições de Projeto
4. Restrições obrigatórias
5. Definições e convenções de nomes
6. Fatos relevantes e suposições

Requisitos Funcionais
7. O escopo do trabalho
8. O escopo do produto
9. Requisitos Funcionas e de dados

Requisitos Não Funcionais
10. Requisitos de interface
11. Requisitos de usabilidade e humanidade
12. Requisitos de desempenho
13. Requisitos operacionais
14. Requisitos de manutenibilidade e portabilidade
15. Requisitos de segurança
16. Requisitos culturais e políticos
17. Requisitos legais

Questões de projeto
18. Questões em aberto
19. Soluções de prateleira
20. Novos problemas
21. Tarefas
22. *Cutover*
23. Riscos
24. Custos
25. Documentação do usuário e treinamento
26. Sala Limpa
27. Idéias para soluções

Especificação preparada por..... Data: .....

Figura 3.2 Modelo de Especificação de Requisitos de Software do Volere [Adaptado de Volere, 2004].

Neste modelo, cada requisito estabelecido utiliza-se de um *shell* que o descreve. Este *shell* é como se fosse uma ficha com vários campos a serem preenchidos, os quais indicam as características de determinado requisito. São eles: número do requisito, tipo do requisito, evento/caso de uso, descrição, razão, fonte, critério de adequação, satisfação do cliente, insatisfação do cliente, dependências, conflitos, materiais de apoio, histórico. Assim, no próprio DR são obtidas algumas informações que seriam identificadas na fase de análise do desenvolvimento de software, pois dados de casos de uso são informados.

Uma breve descrição da estrutura do modelo Volere é apresentada a seguir.

- Diretivas de Projeto

Nesta seção, uma pequena descrição do contexto do trabalho é feita. A proposta do produto e a razão pelo qual ele está sendo desenvolvido são declaradas.

É identificada cada pessoa envolvida no projeto, dentre elas tem-se: *client* - aquelas que pagam para desenvolver o produto e que são proprietárias do sistema entregue, *customer* – aqueles que comprarão o produto do cliente e os *stakeholders*. Usuários do sistema também são identificados. Algumas de suas informações são providenciadas, como: nome do usuário, sua função, experiência na organização, experiência técnica e outras características pessoais. Com essa informações é possível definir requisitos de usabilidade do sistema. Para cada categoria de usuário é associada uma prioridade, ou seja, é dada um grau de importância, uma vez que o sucesso do sistema também depende dos requisitos de cada usuário.

- Restrições de Projeto

As restrições que devem ser consideradas pelo sistema são descritas nesta seção. Algumas delas são: datas críticas que tem efeito nos requisitos do sistema, aspectos orçamentários, o ambiente físico e tecnológico na qual o sistema será instalado, características do ambiente de trabalho que poderia afetar o projeto do sistema, aplicações que devem ser usadas para implementar alguns dos requisitos do sistema.

Todos os termos usados pelo sistema para evitar um entendimento errôneo de seu significado são definidos.

São especificados os fatores externos que podem causar efeito no produto a ser desenvolvido e uma lista de suposições são relatadas pelos desenvolvedores.

- Requisitos Funcionais

O escopo do trabalho é definido, ou seja, todo o trabalho necessário para que se possa construir o sistema é descrito.

O escopo do produto é mostrado em forma de diagrama de casos de uso, identificando as fronteiras entre os usuários e o produto. Assim, uma lista de todos os casos de uso relevante para o produto é especificada.

Ainda nesta seção, cada requisito funcional que deve ser apoiado pelo sistema é declarado. Assim como todo requisito, este também utiliza o *shell*. Os requisitos de dados permitem esclarecer o conteúdo do sistema e descobrir os requisitos que ainda não foram pensados. Para a modelagem de dados do negócio pode-se utilizar a notação UML (*Unified Modeling Language*), modelo Entidade-Relacionamento ou alguma outra.

- Requisitos Não Funcionais

São as propriedades comportamentais que uma especificada função deve ter. Neste modelo de especificação do Volere, os requisitos não funcionais estabelecidos são: requisitos de interface, usabilidade e humanidade, desempenho, operacionais, manutenibilidade e portabilidade, segurança, culturais e políticos, legais.

- Questões de projeto

Define as condições sob o qual o projeto será feito, como:

- Questões que ainda são incertos e que poderiam fazer uma significante diferença no produto é relatada nessa seção.

- Uma lista de produtos existentes no mercado e que podem servir de solução para a organização, chamado de solução de prateleira, é referenciada nesta parte da especificação.

- São levantados os problemas que um novo sistema pode causar numa determinada implementação atual. O objetivo é descobrir, o quanto antes, algum conflito que poderia ser percebido apenas durante a implementação do novo sistema.

- São descritos os passos que devem ser tomados para a entrega do sistema, ou seja, detalhes do ciclo de vida e a abordagem que será usada, assim como as fases de desenvolvimento.

- Uma lista de riscos inerentes a todo projeto é especificada nessa seção.

- O custo é avaliado por algum método de estimativa.

- Um plano que descreve como o treinamento estará disponível e como será feita a documentação para o usuário é descrito.

- Requisitos não fazem parte do produto atual, mas poderia fazê-lo numa versão futura são documentados para garantir que nenhuma idéia boa seja descartada. Da mesma forma, algumas idéias de soluções são consideradas.

A principal diferença notada entre os dois modelos de padrões de especificação de software apresentados é quanto à apresentação dos requisitos. No modelo da IEEE [1998b], os requisitos são organizados conforme as diferentes classes de sistemas que se pretende desenvolver, porém não é definido o que deve conter na descrição de cada requisito. Já no modelo Volere [2004], não se comenta quanto à disposição dos requisitos, mas é oferecido um formato (*shell*) de quais informações eles devem apresentar. Além disso, no Volere [2004] parte da fase de análise é descrita no próprio DR.

Independente do modelo escolhido, o ideal é que cada organização de desenvolvimento de software adote um padrão de DR para seus projetos, vista a importância

que tal documento desempenha (descrita na Seção 2.3.3). No estudo de Huang & Tilley [2003] ressalta-se a importância da documentação para um bom entendimento do programa que se deseja construir. O seu trabalho é o início de um estudo que pretende elaborar o Modelo de Maturidade para Documentação de Sistema (DMM), que engloba como componente tanto o processo usado para elaborar a documentação quanto o próprio produto/artefato (documento). É dada uma visão geral dos níveis de maturidade do processo DMM, que é exatamente uma cópia dos níveis do CMM: inicial, repetível, definido, gerenciado, otimizado. O foco é dado nos aspectos de qualidade do componente produto, mostrando uma estrutura única para avaliar a qualidade da documentação de um sistema de software. O componente produto do DMM é baseado na noção de atributos chaves do produto (KPA). Os níveis de maturidade do produto é invertido: cada KPA tem 5 níveis de maturidade. Por enquanto foram definidos três KPA's: Formato (Texto/Gráfico), Eficiência e Granularidade. A Figura 3.3 ilustra a estrutura do modelo de avaliação do DR.

KPA		**Nível de maturidade**				
		1	**2**	**3**	**4**	**5**
Formato	**Texto**	*Inline* & Informal	*Inline* & Padronizado	Hipertexto	Contextual	Personalizado
	Gráfico	Estático & Informal	Estático & Padronizado	Animado	Interativo	Editável
Eficiência		Manual	Semi-automático & Estático	Semi-automático & Dinâmico	Automático & Estático	Automático & Dinâmico
Granularidade		Código Fonte	Padrões de Projeto	Arquitetura de Software	Requisitos	Linhas de Produto

Figura 3.3 KPAs e Nível de Maturidade do DMM [Adaptado de Huang & Tilley [2003].

Embora haja padrões de especificação de DR, a dificuldade está na forma escolhida para expressar os requisitos. O dilema enfrentado pelos engenheiros de requisitos é o nível de formalidade contida no DR. Se tal documento é para ser usado na comunicação com clientes e usuários do sistema, representações informais e linguagem natural são mais prováveis de serem bem sucedidas, já que clientes e usuários geralmente não são familiarizados com notações computacionais. Por outro lado, se esse documento é para ser usado como uma especificação do que um sistema de computador é solicitado a fazer, os defensores de representações formais argumentam que tais formatos trazem benefícios na comunicação com os desenvolvedores de sistema, em termos de entendimento da aplicação, da ausência de ambigüidade e da identificação de abstrações úteis [Damian, 2000].

A questão é que, conforme dito por Schach [2002], treinamento na preparação da documentação freqüentemente é deixado de lado, o que pode ser observada/evidenciada pela baixa qualidade de muitos DRs. Da mesma forma que um treinamento adicional é requerido quando um novo hardware é utilizado ou quando uma nova ferramenta de software é executada, um treinamento para documentar requisitos de um software se faz necessário.

O problema normalmente citado quanto ao uso da linguagem natural é que ela conduz ao mal-entendimento devido à inerente ambigüidade de suas palavras e por ela ser bastante flexível, pois se pode dizer as mesmas coisas de maneiras completamente diferentes. Essa dificuldade pôde ser comprovada pelo resultado de um estudo realizado pelo SATC (*Software Assurance Technology Center*) em mais de 50 DRs da NASA, escritos em linguagem natural, o qual identificou que a maioria deles apresentava problemas estruturais e lingüísticos, principalmente termos mal-definidos, inconsistências, ambigüidades e a tendência de se misturar requisitos com decisões de projeto [Wilson, 1997]. Para esse estudo foi utilizada a ferramenta ARM (*Automated Requirements Measurement*), a qual oferece métricas para que gerentes de projeto da NASA possam usar para avaliar a qualidade de seus DRs e ajudar a identificar os riscos que podem ser introduzidos em seus projetos através de requisitos precariamente especificados. A ferramenta ARM avalia a estrutura do DR, as sentenças individuais de especificações e o vocabulário usado para estabelecer os requisitos. Este estudo destacou as deficiências mais comuns encontradas na documentação e as suas causas. Também foram recomendadas algumas práticas para elaborar um DR que evita os tipos de problemas encontrados no estudo de DRs feito por SATC.

Outra ferramenta que auxilia na análise e avaliação da qualidade de DR escritos em linguagem natural é a QuARS (*Quality Analyser of Requirements Specifications*) [Fabbrini et al., 2000 e 2001]. Tal ferramenta foi definida com base numa estrutura chamada *Quality Model*, a qual avalia dois aspectos de qualidade do DR: qualidade das sentenças de requisitos e qualidade do DR. A Figura 3.4 mostra o *Quality Model* com as suas propriedades relacionadas e os indicadores de qualidade que ela avalia.

Objetivo das propriedades	Propriedades	Indicadores de Qualidade
Qualidade das Sentenças de Requisitos	Não-ambíguo	Sentenças Subjetivas Implícitas Sentenças Opcionais Sentenças Subjetivas Sentenças Vagas Sentenças Insignificantes
	Completo	Sentenças especificadas insuficientemente
	Compreensível	Sentenças Múltiplas
Qualidade do Documento de Requisitos	Consistente	Sentenças referenciadas insuficientemente
	Compreensível	Freqüência de Comentário Índice de Leitura Sentenças explicadas insuficientemente

Figura 3.4 Quality Model [Adaptado de Fabbrini et al., 2000].

No estudo de Denger [2003] é apresentada uma abordagem para reduzir a imprecisão inerente às especificações de requisitos escritos em linguagem natural dentro do domínio de sistemas embutidos. O objetivo dessa abordagem é evitar a introdução de ambigüidade, não completitude e inexatidão durante a elaboração do documento que está sendo escrito. Para isso foi elaborado um padrão de linguagem natural que permite formular sentenças de requisitos de forma menos ambígua, mais completa e precisa. Esse padrão é baseado num metamodelo para especificações de requisitos desenvolvido especialmente para sistemas embutidos. Ele descreve os elementos que precisam conter tais especificações de requisitos, sendo o foco da abordagem um padrão para requisitos funcionais de sistemas embutidos.

Neste presente trabalho, para compor as Diretrizes para elaboração de DR, foi elaborado um formato para especificar os requisitos de um sistema escritos em linguagem natural, que é ao mesmo tempo não técnico, para que o cliente possa entender, e ainda preciso o suficiente para obter um sistema livre de defeitos.

Na seção seguinte são apresentadas as características que definem um DR de boa qualidade.

3.3 Propriedades de Qualidade do Documento de Requisitos

As características que definem uma boa especificação de requisitos, ou seja, as propriedades de qualidade que esta deve exibir foram exploradas no trabalho de Davis et al. [1993b] onde foi definido um total de 24 atributos de qualidade. Também foi dada uma idéia de como medi-los; um peso relativo de um atributo em relação a outros atributos foi

recomendado e foram descritos os tipos de atividades que podem ser usadas para otimizar a presença de cada atributo.

As propriedades de qualidade de um DR ou de sentenças de requisitos também são definidas por alguns autores e pelo padrão de especificação. Nem todos consideram o mesmo conjunto de propriedades [Davis et al., 1993b; Kar & Bailey, 1996; Wiegers, 1999a; IEEE, 1998b; Hooks, 1993; Firesmith, 2003].

Abaixo são listadas e descritas as propriedades de qualidade de um DR.

1) Não ambíguo: para que uma especificação de requisitos seja não ambígua, cada requisito apresentado nela deve gerar uma e somente uma interpretação. A sentença que expressa o requisito não deve ocasionar dúvida. Evitar palavras subjetivas como amigável, fácil, simples, rápido, eficiente, vários, maximizar, minimizar, etc. [Firesmith, 2003; Hooks, 1993; IEEE, 1998b; Kar & Bailey, 1996; Wiegers, 1999a].

2) Completo: a especificação torna-se completa se tudo o que o software é suposto a fazer está incluído nela e então nenhum requisito ou informação necessária deve estar faltando. O conjunto de requisitos está completo e não precisa de maiores detalhes. O requisito apresentado deve oferecer capacidade suficiente e ser capaz de permanecer sozinho quando separado de outros requisitos. Não deve haver nenhuma seção marcada "Para ser determinada"; todas as páginas são numeradas; todas as figuras e tabelas são numeradas, nomeadas e referenciadas; todos os termos são definidos; todas as unidades de medida são oferecidas; e todos os materiais referenciados são apresentados. Respostas do software para todas as classes realizáveis de entrada de dados em todas as classes realizáveis de situações é incluída. [Kar & Bailey, 1996; Davis et al., 1993b; Wiegers, 1999a; IEEE, 1998b; Firesmith, 2003].

3) Correto: uma especificação é correta se e somente se cada requisito estabelecido nela representa algo solicitado do sistema para ser construído, ou seja, cada requisito na especificação deve descrever exatamente a funcionalidade requerida e contribuir para satisfazer alguma necessidade. Ela deve ser semântica e sintaticamente correta [Davis et al., 1993b, Firesmith, 2003; IEEE, 1998b; Wiegers, 1999a].

4) Compreensível: uma especificação é compreensível se todas as suas classes de leitores (clientes, usuários, gerentes de projeto, desenvolvedores de software e testadores) podem facilmente compreender o significado de todos os requisitos com um mínimo de explicação [Davis et al., 1993b; Firesmith, 2003].

5) Verificável: uma especificação é verificável se existe uma técnica finita e de custo eficaz que pode ser usada para verificar se cada requisito apresentado é satisfeito pelo sistema

quando construído. É preciso determinar um critério de aceitação. Os requisitos não devem ser vagos ou genéricos, mas sim específicos, não-ambíguos e quantificados de tal modo que possam ser verificados [Davis et al., 1993b; Hooks, 1993; IEEE, 1998b; Kar & Bailey, 1996; Wiegers, 1999a]. Gilb [1997] mostra que requisitos qualitativos ou não-funcionais (por exemplo, fácil de manter, adaptável, portável, amigável, seguro, etc), normalmente difíceis de mensurar, podem ser especificados quantitativamente de forma clara. É dada uma idéia de como alcançar essa propriedade através de definição de uma escala de medida.

6) Consistente internamente: uma especificação é internamente consistente se os requisitos apresentados não contrariam outros requisitos, ou seja, nenhum subconjunto de requisitos individuais estabelecidos se conflitam. Eles não são uma duplicação de algum outro requisito. O mesmo termo é usado para o mesmo item em todos os requisitos [Firesmith, 2003; IEEE, 1998b; Kar & Bailey, 1996].

7) Consistente externamente: uma especificação é externamente consistente se e somente se nenhum requisito apresentado se conflita com alguma documentação de projeto já concluída [Davis et al., 1993b; Wiegers, 1999a].

8) Praticável: uma especificação é praticável se e somente se existir pelo menos um projeto de sistema e implementação que implemente corretamente todos os requisitos apresentados nela, dentro das capacidades e limitações conhecidas do sistema e do seu ambiente e num custo definível. Isso implica que pelo menos um projeto conceitual de alto nível foi completado e estudos de custos foram conduzidos [Davis et al., 1993b; Firesmith, 2003; Hooks, 1993; Kar & Bailey, 1996; Wiegers, 1999a].

9) Conciso: uma especificação é concisa se ela é o mais sucinta possível sem adversamente afetar qualquer outra qualidade dela. Cada sentença que expressa o requisito inclui somente um requisito estabelecendo apenas o que deve ser feito, de forma simples e clara, que seja fácil de ler e entender [Davis et al., 1993b; Hooks, 1993; Kar & Bailey, 1996].

10) Independente de projeto: uma especificação é independente de projeto se e somente se existir mais que um projeto de sistema e implementação que implemente corretamente todos os requisitos apresentados nela [Davis et al., 1993b].

11) Rastreável para frente: uma especificação é rastreável se e somente se ela é escrita de uma maneira que facilita a referência de cada um de seus requisitos individuais num desenvolvimento futuro ou numa outra documentação [Davis et al., 1993b; IEEE, 1998b; Wiegers, 1999a].

12) Modificável: uma especificação é modificável se sua estrutura e estilo são de tais formas que permitem que qualquer mudança nos requisitos possa ser feita facilmente,

completamente e consistentemente. Deve-se poder revisar a especificação sempre que necessário e manter o histórico da mudança feita em cada requisito. Isso significa que cada requisito seja unicamente identificado e expresso separadamente de outros requisitos [IEEE, 1998b; Wiegers, 1999a].

13) Armazenado eletronicamente: uma especificação é armazenada eletronicamente se e somente se ela toda está num processador de texto, se foi gerada de um banco de dados de requisitos ou foi sintetizada de alguma outra forma [IEEE, 1998b].

14) Executável/Interpretável/Protótipo: uma especificação é executável, interpretável ou está num formato de protótipo se e somente se existe uma ferramenta de software capaz de tê-la como entrada e que ofereça um modelo dinâmico do seu comportamento [IEEE, 1998b].

15) Anotado por importância relativa: uma especificação é anotada por importância relativa se cada requisito é designado com grau de prioridade de implementação. Assim torna-se fácil determinar quais deles são mais importantes para o cliente e indica o quão essencial é incluí-los numa versão particular do produto [Davis et al., 1993b; IEEE, 1998b; Wiegers, 1999a]. Estabelecer escala de prioridade para requisitos é uma excelente característica, pois quando as expectativas do cliente são altas, quando não é possível entregar todos os requisitos de software dentro de um determinado cronograma e os recursos são limitados, a equipe de desenvolvimento ainda tem a opção de implementar aqueles de maior prioridade e entregar um produto que contém a maioria das principais funcionalidades. Priorizar ajuda o gerente de projeto a resolver conflitos, planejar os estágios de entrega do produto e fazer as necessárias decisões de negócio [Wiegers, 1999b].

16) Anotado por estabilidade relativa: uma especificação é anotada por estabilidade relativa se cada requisito tem um identificador que indica sua estabilidade, sendo fácil determinar quais deles são mais prováveis de mudar, quais são os próximos mais prováveis de mudar e assim por diante [Davis et al., 1993b; IEEE, 1998b].

17) Anotado por versões: uma especificação é anotada por versão se é fácil determinar quais requisitos serão satisfeitos em quais versões do produto [Davis et al., 1993b].

18) Não redundante: uma especificação é não redundante se nenhum requisito é apresentado mais do que uma vez [Davis et al., 1993b].

19) No nível certo de detalhe/de abstração: para que a especificação apresente-se num nível certo de abstração, todos os requisitos precisam estar neste nível.

20) Preciso: uma especificação é precisa/exata se e somente se quantidades numéricas são usadas sempre que possível e níveis apropriados de precisão são usados para todas quantidades numéricas [Davis et al., 1993b].

21) Reusável: uma especificação é reusável se e somente se suas sentenças, parágrafos e cenários possam ser facilmente adotados ou adaptados para uso numa subseqüente especificação [Davis et al., 1993b].

22) Rastreável para trás: uma especificação permite rastreio se for possível ligar cada requisito de software com sua fonte/origem. Isso implica que todo requisito que tem uma base é referenciado para aquela base [Davis et al., 1993b; IEEE, 1998b; Wiegers, 1999a].

23) Organizado: uma especificação é organizada se e somente se seu conteúdo está arrumado/arranjado de tal forma que leitores podem facilmente localizar informações e relações lógicas entre seções adjacentes é aparente [Davis et al., 1993b].

24) Possui Referência-Cruzada: uma especificação possui referência-cruzada se e somente se referências-cruzadas são usadas para relacionar seções que possuem requisitos para outras seções contendo: requisitos idênticos; descrições mais abstratas ou mais detalhadas do mesmo requisito; requisitos que dependem deles ou que eles dependem [Davis et al., 1993b].

25) Necessário: uma especificação pode ser considerada necessária quando cada um de seus requisitos documenta algo que o cliente realmente necessita ou algo que é requerido para conformar-se com um requisito externo, uma interface externa ou um padrão. O requisito declarado é uma capacidade essencial, característica física ou fator de qualidade do produto ou processo. Se o requisito é removido uma deficiência existirá, o qual não pode ser realizado por outras capacidades do produto ou processo [Firesmith, 2003; Kar & Bailey, 1996; Wiegers, 1999a].

26) Livre de implementação: uma especificação é livre de implementação quando cada um de seus requisitos estabelece o que é requerido e não como ele deveria ser satisfeito. Uma sentença de requisito não deveria refletir um projeto ou implementação e nem descrever uma operação [Firesmith, 2003; Kar & Bailey, 1996].

27) Atualizado: uma especificação é atualizada quando cada um de seus requisitos representa as necessidades atuais ou antecipadas de seus clientes e usuários. Nenhum requisito é obsoleto [Firesmith, 2003].

28) Orientado à cliente/usuário: uma especificação é orientada a cliente ou usuário quando seus requisitos são expressos numa linguagem compreensível para eles, sem jargões técnicos usados pelos desenvolvedores [Firesmith, 2003].

29) Metadados: requisitos individuais devem ter metadados, isto é, atributos ou anotações, que os caracterizam. Por exemplo: critério de aceitação, alocação, suposição, identificação, priorização, razões, status, informações de rastreios e outros [Firesmith, 2003].

Para melhor visualizar o que cada autor ou padrão considera como atributo de qualidade de boa especificação, a Tabela 3.1 apresenta essa informação. Uma adaptação foi feita a essa tabela encontrada em Davis et al. [1993b], na qual foram adicionadas as seis primeiras colunas.

Tabela 3-1 Propriedades de qualidade de um DR [Adaptado de Davis et al., 1993b].

Propriedades de qualidade do DR	Referências																						
	IEEE [1998b]	DAVIS et al. [1993b]	KAR & BAILEY [1996]	WIEGERS [1999a]	HOOKS [1993]	FIRESMITH [2003]	BOE74	ALF76	BEL76	DAV79	BAS81	ZAV81	CEL83	IEE84	NCC87	ESA87	DOD88	JPL88	CAR90	DAV90	ROM90	DAV93
1) Não-ambíguo	x	x	x	X	x	x		x	x	x	x	x	x	x	x	x		x		x	x	x
2) Completo	x	x	x	X		x	x	x	x	x	x	x	x	x	x	x		x		x	x	x
3) Correto	x	x		X		x		x	x		x							x		x	x	x
4) Compreensível		x				x		x		x		x	x	x	x	x	X			x	x	x
5) Verificável	x	x	x	X	x	x		x	x			x		x		x	X	x		x	x	x
6) Consistente internamente	x	x	x	X		x	x	x	x	x	x	x	x	x	x	x	X	x		x	x	x
7) Consistente externamente		x							x						x	x	X	x			x	x
8) Praticável		x	x	X	x	x		x	x							x					x	
9) Conciso		x	x		x														x			x
10) Independente de projeto		x						x	x	x		x						x				x
11) Rastreável para frente	x	x		X			x	x						x		x		x		x	x	x
12) Modifcável	x	x		x				x			x	x		x						x	x	x
13) Armazenado eletronicamente		x													x							
14) Executável/Interpretável/Protótipo		x										x										x
15) Anotado por importância relativa	x	x		x												x		x		x		x
16) Anotado por estabilidade relativa	x	x														x				x		x
17) Anotado por versões		x																				
18) Não redundante		x						x	x	x				x		x				x		x
19) No nível certo de detalhe		x																			x	
20) Preciso		x										x			x	x					x	
21) Reusável		x																				
22) Rastreável para trás	x	x		x				x					x	x			x	x		x		x
23) Organizado		x						x							x						x	x
24) Possui referência-cruzada		x																				
25) Necessário			x	x	x	x																
26) Livre de implementação			x			x																
27) Atualizado						x																
28) Orientado à cliente/usuário						x																
29) Metadados						x																

Entendido o significado de qualidade de uma especificação, fica-se mais preparado e melhor equipado para detectar erros na especificação e então evitar que eles permaneçam e custe mais para detectá-los e repará-los [Davis et al., 1993b].

Embora haja muitos estudos que mostrem as características que uma especificação bem escrita deve conter, a maioria das especificações de requisitos de software nas indústrias apresenta-se mal definida, mal elaborada e pobremente escrita. Algumas das razões para isso é que as pessoas não têm treinamento ou experiência em como escrever bons requisitos e o insuficiente esforço e tempo dedicado à definição dos mesmos, principalmente porque a Engenharia de Requisitos é vista como um processo demorado, burocrático e contratual [Nuseibeh & Easterbrook, 2000].

Na seção seguinte será descrito o quê alguns autores recomendam para a elaboração do DR a fim de tê-lo correto, consistente, sem ambigüidade, claro, enfim recomendações que tentam minimizar os problemas decorrentes de DR mal elaborado.

3.4 Recomendações Gerais para Elaboração de Requisitos

As exigências quanto a um bom DR são cada vez maiores, pois é a partir do entendimento dele que a arquitetura do sistema será construída. Além dos padrões para sua especificação e das propriedades que o qualificam, descritas nas seções anteriores, algumas recomendações específicas para descrever requisitos, sugeridas por diversos autores, são retratadas nesta seção. O fato dos requisitos serem o alvo de atenção é que eles são tidos como o principal causador de falhas em projeto de software, como foi dito no Capítulo 2.

Wilson [1997] aproveita para destacar algumas diretrizes para elaboração de um DR, ao identificar diversas deficiências da análise de vários DRs através da ferramenta ARM (*Automated Requirements Measurement*). Suas diretrizes são agrupadas da seguinte forma:

- Palavras e Frases
  - Usar as palavras mais simples e apropriadas para o contexto da sentença.
  - Usar imperativos corretamente e ser consistente, analogamente, como no seguinte exemplo da língua inglesa: *shall* indica “prescrição”, *will* indica “descrição”, *must* e *must not* indica “restrição” e *should* indica “sugestão”.
  - Evitar palavras de pouca expressão, tais como “como um mínimo”, “ser capaz de”, “capacidade de”, “não limitado a”.
  - Não usar palavras ou termos que sugiram opção quando o requisito realmente deve ser satisfeito, tais como “pode”, “se requerido”, “como apropriado”, “se praticável”.
  - Não usar generalidades em que números são realmente requeridos, tais como “grande”, “rápido”, “muitos”, “periodicamente”, “a maior parte”, “próximo”.

  - Evitar palavras indistintas que têm significados relativos, tais como “fácil”, “normal”, “adequado”, “efetivo”.

- Prover exemplos
  - Ofereça imediatamente o que é para ser ilustrado com o exemplo.
  - Repita um exemplo se ele não estiver localizado na mesma página.
  - Destaque o exemplo (escrevendo em itálico, usando aspas ou sendo explícito, como “Este é um exemplo”) para garantir que ele não esteja errado como parte da especificação.
- Citar referências
  - Identifique todos os documentos externos na seção do DR designado a este propósito através de uma estrutura única que contenha todas as informações sobre esses documentos.
  - Identifique cada referência citada com um número único ou identificador.
  - Cite as referências por um título curto ou comum, título completo, versão ou indicação de publicação, data, editor ou fonte, e número do documento ou outro identificador único de documento.
- Usar tabelas e quadros
  - Intitule e identifique cada tabela e quadro por um identificador único.
  - Liste cada tabela e quadro na tabela de conteúdo do DR através de um título, identificador único e número da página.
  - Identifique o propósito da tabela ou quadro no texto imediatamente precedente a ele.
  - Explique cada aspecto ou elemento da tabela ou quadro (colunas, linhas, símbolos, em brancos, etc.).
- Resumo e Conclusão

  Quando usar linguagem natural para especificar requisitos os seguintes aspectos devem ser sempre mantidos em mente:
  - O DR é um meio para expressar requisitos e não um esboço ou linhas gerais para uma metodologia para derivar requisitos.
  - O DR é um item do software e como tal deveria ser um produto trabalhado.
  - Use estruturas de sentença apropriadas e selecione palavras e frases baseadas nas suas definições formais e não o que a cultura popular supõe.

Outro trabalho que apresenta sugestões para melhorar a escrita de requisitos é o de Hooks [1993], que lista os problemas mais comuns que são encontrados ao se escrever requisitos e descreve como evitá-los. Para isso inclui alguns exemplos desses problemas e como corrigi-los. Os principais problemas e soluções levantadas foram:

- *Problema*: fazer más suposições porque o autor do requisito não tem acesso às informações suficientes ou porque a informação não existe. *Solução*: para evitá-los deve-se manter uma lista de todos os documentos relevantes e torná-los acessíveis a cada autor. No caso da informação não existir, o autor do requisito deveria documentar todas as suposições sobre o requisito para posterior revisão;
- *Problema*: escrever implementações (como) ao invés de requisitos (o que). *Solução*: as especificações deveriam expressar *O que* é necessário, e não *Como* o requisito deve ser providenciado ou realizado;
- *Problema*: usar termos incorretos. *Solução*: há termos que precisam ser evitados para não causar confusões (por exemplo: apoiar, etc, e/ou, não limitado a). Existem alguns termos que precisam ser usados de maneira específica. Na especificação americana estabeleceu-se um padrão de uso dos termos *shall*, *will* e *should* nas agências governamentais e na indústria. Requisitos usam *shall*, sentenças de fatos usam *will* e objetivos usam *should*;
- *Problema*: usar estrutura de sentença incorreta ou péssima gramática. Muitas vezes autores escrevem tudo o que sabem numa única sentença, tornando-as complexas, longas e usando muitas cláusulas. *Solução*: o bom seria construir sentenças seguidas de simples predicados e não por uma lista. O uso de péssima gramática pode causar más interpretações. Para evitar esses problemas é sugerido que cada requisito seja escrito usando o seguinte formato: O Sistema deve prover..., O Sistema deve ser capaz de..., O Subsistema #1 deve fazer interface com...;
- *Problema*: escrever requisitos inverificáveis através de termos ambíguos (maximizar, minimizar, rápido, amigável, fácil, suficiente, adequado, ágil). *Solução*: como esses termos são subjetivos, eles devem ser evitados;
- *Problema*: deixar de escrever alguns requisitos. *Solução*: os itens faltantes podem ser evitados usando uma especificação padrão;
- *Problema*: especificar em excesso, o qual ocorre quando há requisitos que são descritos desnecessariamente ou que são excessivamente limitadas/severas, ou seja, não consideram tolerâncias. *Solução*: no primeiro caso deve haver uma cuidadosa revisão e controle de gerenciamento. A solução para o segundo caso seria uma discussão das

tolerâncias permitidas para cada valor e então uma posterior escrita do requisito considerando tais tolerâncias.

Para combater esses e outros incidentes na escrita de requisitos, Firesmith [2003] apresenta um questionário para auxiliar na especificação e avaliação dos requisitos. Firesmith considera algumas características que um requisito de boa qualidade deveria exibir e para cada uma delas ele fornece um conjunto de questões para garantir requisitos de qualidade. Este questionário pode ser considerado um *checklist* que auxilia no momento da escrita do DR, contribuindo para a sua elaboração.

Uma fórmula para escrever requisitos excelentes não existe. Essa é a opinião de Wiegers [1999a], que considera isso uma questão de experiência e aprendizado de problemas de requisitos encontrados no passado. Algumas diretrizes para documentar requisitos de software são oferecidas por Wiegers [1999a]:

- Manter sentenças e parágrafos curtos. Usar voz ativa. Usar gramática, ortografia e pontuação apropriada. Usar termos consistentes e defini-los em glossário ou dicionário de dados.
- Ler uma sentença de requisito sob a perspectiva do desenvolvedor para ver se ela está suficientemente bem definida.
- Evitar parágrafos narrativos longos que contenham múltiplos requisitos. Escrever requisitos num certo nível de granularidade; uma diretriz útil para isso é escrever requisitos testáveis individualmente.
- Atentar-se para requisitos múltiplos que tenham sido agregados numa única sentença. Conjunções como “e” e “ou” sugere que vários requisitos foram combinados. Nunca usar “e/ou” numa sentença de requisitos.
- Escrever requisitos num nível consistente de detalhe em todo o documento.
- Evitar declarar requisitos de forma redundante numa especificação de requisitos de software, pois fica mais difícil mantê-lo.

Um manual para especificar requisitos é oferecido pelo guia de especificação norte-americano [Oriel, 1999], que é uma coleção de artigos curtos escritos por engenheiros experientes do governo americano. O objetivo é ajudar os recém-chegados nessa área e aqueles que preparam especificações informais; porém, esse manual não os prepara suficientemente para manter o alto nível de conhecimento e as habilidades que eles necessitam para fazer o trabalho correto. Esse guia é para ser usado como uma introdução geral aos princípios que se aplicam para escrever especificação.

Um outro recurso que também foi identificado na literatura para ajudar na elaboração de DR é uma linguagem de especificação de requisitos chamada PLanguage ou *"Planning Language"* de autoria de Gilb [1997]. Ela permite que requisitos não-funcionais ou de qualidade, que descrevem quão bem uma funcionalidade definida deve desempenhar, sejam quantificados e testados através de definição de uma escala de medida, evitando a falta de clareza e de detalhamento. Gilb diz que para ser preciso, os requisitos de qualidade devem ser mensuráveis. Basicamente eles devem ser divididos em sub-requisitos para cobrir os diferentes aspectos dos termos globais. Por exemplo, o requisito de qualidade usabilidade poderia ser dividido em requisitos fácil de usar, fácil de aprender, rápido de usar, etc. E esses requisitos podem então ser divididos em outros. Enquanto requisitos funcionais podem ser considerados binários, isto é, eles são implementados ou não, os requisitos não-funcionais são mais difíceis de quantificar e verificar. Para evitar requisitos vagos é que Gilb propôs a PLanguage.

Com base nessas recomendações para elaboração de requisitos é que as Recomendações contidas nas Diretrizes propostas por este trabalho foram estabelecidas.

3.5 Considerações Finais

Neste capítulo viu-se que estudos relacionados ao DR são diversos. Mais uma vez pôde-se perceber que não se trata de um simples documento. A importância de sua boa definição para um correto desenvolvimento de software levou à elaboração de modelos de especificação que padronizam a sua documentação; outros estudos mostram um conjunto de propriedades que esses documentos deveriam apresentar como forma de qualificá-los e recomendações sugeridas para melhor descrevê-los e evitar que informações relevantes deixem de ser expressos ou que haja interpretação incorreta foram mostradas.

Foi com base nessa pesquisa que as Recomendações de Escrita de requisitos, contida nas Diretrizes propostas por este trabalho, foram definidas.

No capítulo seguinte será comentada a inspeção de DR, mostrando algumas técnicas específicas para sua validação.

Capítulo 4

4. Inspeção de Documento de Requisitos

4.1 Considerações Iniciais

Para assegurar que o DR esteja consistente com as pretensões dos *stakeholders*, é fundamental que sejam aplicadas técnicas apropriadas para avaliar a documentação gerada, sendo que a Inspeção é um importante meio de verificar e alcançar a qualidade suficiente em muitos projetos de software.

A avaliação feita pela Inspeção permite que defeitos sejam detectados, reduzindo posteriores re-trabalhos. De forma auxiliar a atividade de revisão, algumas técnicas surgiram, sendo uma delas a Técnica de Leitura, cujo propósito é tornar essa atividade mais sistemática e eficiente.

Como o motivo principal de realização desse trabalho é facilitar particularmente a Técnica de Leitura TUCCA (*Technique for Use Case Construction and construction-based requirements Analysis*) fez-se necessário contextualizar o que seria Inspeção e as Técnicas de Leitura cujo artefato a ser examinado é o DR.

Sendo assim, este capítulo está organizado da seguinte forma: na Seção 4.2 apresenta-se a atividade de Inspeção, na Seção 4.3 é apresentada a abordagem *Checklist,* nas Seções 4.4, 4.5, 4.6 e 4.7, diferentes Técnicas de Leitura são apresentadas e na Seção 4.8, apresentam-se as Considerações Finais.

4.2 Atividade de Inspeção

A técnica de inspeção foi originalmente desenvolvida na IBM em meados de 1970 por Michael Fagan e, inicialmente, proposta para avaliação do código-fonte de programa. O objetivo específico é a identificação e remoção de erros de forma antecipada, ou seja, antes de o produto ser entregue, sem, contudo, envolver-se na procura de soluções para esses erros [Fagan, 1976 e 1986; Laitenberger, 1995]. Para isso é feita uma investigação minuciosa do produto a ser inspecionado através de sua submissão à equipe de revisores, os quais coletam o

maior número de erros possível que são discutidos numa reunião posterior e enviados ao autor para corrigir o produto.

No processo de "Inspeção de Fagan" [Fagan, 1976 e 1986] cada participante envolvido tem responsabilidades específicas e desempenha funções bem-definidas de acordo com os papéis a eles atribuídos. De modo geral, os papéis podem ser descritos como mostra a Tabela 4.1.

Tabela 4-1 Papel dos participantes de uma inspeção e sua respectiva descrição.

Papel	***Descrição do Papel***
Organizador	Planeja todas as atividades de inspeção dentro do projeto.
Moderador	Como líder da equipe esse conduz as atividades de inspeção e controla as reuniões.
Autor	Cria o produto de trabalho a ser inspecionado e é responsável pela correção dos defeitos identificados.
Apresentador	Descreve as seções do produto de trabalho para a equipe de inspeção, percorrendo-o e então o explicando e interpretando, ou seja, relatando o que esse se supõe a fazer.
Redator	Classifica e registra todos os defeitos e questões levantadas durante a reunião de inspeção.
Coletor	Coleta os defeitos encontrados pelos inspetores caso não haja reunião de inspeção.
Inspetor	Responsável por encontrar erros no produto. Todos os participantes podem atuar como inspetores em adição a quaisquer outras responsabilidades.

Esses participantes realizam suas tarefas seguindo um processo de inspeção que constitui de uma série de atividades, cada uma com um objetivo específico. A estrutura desse processo proposto por Fagan [1976; 1986] é apresentado a seguir:

- *Planejamento*: a equipe de inspeção é selecionada; o material a ser utilizado é obtido do autor e distribuído à equipe com antecedência; é verificada a disponibilidade dos participantes para a realização da inspeção e então, um local e horário para o encontro é agendado.
- *Reunião de Apresentação*: são descritas as características importantes do produto a ser inspecionado para a equipe de inspeção e os papéis são atribuídos a cada participante.
- *Preparação*: os participantes são responsáveis por examinar criteriosamente o material antes da reunião de inspeção, anotando qualquer defeito encontrado ou questão a ser levantada e preparando-se para executar seu devido papel.
- *Reunião de Inspeção*: a equipe é convocada para discutir o material examinado. Durante a reunião, todos os inspetores podem reportar os defeitos ou levantar outras questões, que são documentadas num formulário pelo redator. Ressalta-se que nenhuma solução ao erro encontrado deve ser dada uma vez que a intenção é apenas encontrar erros.

- *Re-trabalho*: o autor é responsável por resolver todos os defeitos e questões expostas na reunião de inspeção.
- *Acompanhamento*: é a parte final através do qual o material deve passar a fim de que a inspeção seja completada. O moderador é responsável por acompanhar o autor para assegurar que o re-trabalho necessário foi desempenhado adequadamente e que nenhum outro defeito foi introduzido.

A vantagem da inspeção é que ela encurta o tempo de entrega do produto uma vez que reduz o tempo gasto na fase de integração e teste do sistema, pois um produto mais qualificado é passado nos estágios posteriores ao exame de inspeção. E tendo um produto final de qualidade, além de aumentar a satisfação do cliente, poupa-se tempo de manutenção e assim aproveita-se o esforço para outros trabalhos. Reduzindo o tempo que se teria com o re-trabalho, aumenta-se a produtividade das atividades seguintes e a melhora no processo como um todo. Ainda, os dados de relatórios de inspeção podem ser usados para identificar os tipos de defeitos mais comuns e os mais custosos, determinar a causa de sua origem e mudar a maneira como o trabalho é feito a fim de evitá-los [Wiegers, 1995].

Visto os benefícios da inspeção e suas contribuições para a qualidade do sistema como um todo, sua importância ficou reconhecida entre muitos desenvolvedores de software e organizações. Desde então, várias outras propostas foram desenvolvidas com a finalidade de melhorar a inspeção de Fagan [1976; 1986] e com variações nos artefatos utilizados nesse processo. Aurum et al. [2002] sintetizam outros tipos de modelo de processo de inspeção de software que emergiram nos últimos 25 anos, mostrando a evolução da inspeção em duas principais áreas, o qual inclui mudança na estrutura do processo e o surgimento de vários métodos, ferramentas e modelos de apoio a essa estrutura do processo de inspeção. Para citar alguns exemplos, têm-se os trabalhos dos seguintes autores: Schneider et al. [1992], Porter et al. [1995], Basili et al. [1996a e 1996b], Shull et al. [2000], Travassos et al. [1999a, 1999b, 2000, 2002].

Dentre as diversas abordagens e técnicas que auxiliam no processo de inspeção, destacam-se as Técnicas de Leitura, que têm por objetivo conduzir os revisores na leitura de produtos de software enquanto descobrem defeitos nesses produtos. Elas oferecem um conjunto de instruções que explica como ler um documento de software e o que o revisor deverá procurar nele durante o processo de inspeção [Basili et al., 1996b]. Portanto, são classificadas como técnicas sistemáticas que fornecem diretrizes para serem usadas durante a fase de preparação de inspeção para examinar um dado artefato e identificar seus defeitos.

Sendo assim, elas tornam o trabalho dos revisores mais metódico e ordenado propiciando uma significativa melhora no processo de inspeção.

As Técnicas de Leitura podem ser definidas como uma série de passos ou procedimentos preparados para a análise individual de artefatos de software cujo propósito é guiar um inspetor na aquisição de um profundo entendimento de um produto de software, o qual é necessário para realizar uma determinada tarefa [Basili et al., 1996a; Laitenberger, 2002]. Devido à importância de algumas destas técnicas para o contexto deste trabalho, estas técnicas serão comentadas.

4.3 Checklist

Checklist é uma abordagem de inspeção que guia na identificação de defeitos através de um formulário de questões que devem ser respondidas pelos revisores enquanto eles examinam determinado produto de trabalho. Foi adotada no processo de inspeção de Fagan para a verificação de código fonte de programas de software e, a partir de então, diversos tipos de *checklists* foram elaborados. O questionário provido por esta abordagem oferece aos revisores um suporte para saber o que examinar.

As vantagens do uso do *checklist* é que ela é simples de se aplicar, rápida, de baixo custo e implementação, ajuda a treinar pessoas sem experiência em validação e fornece uma estrutura que permite aos autores do produto a ser inspecionado de se lembrarem de checar determinados aspectos deste produto [Sawyer et al., 1999]. A desvantagem é que não há nenhum procedimento bem definido para procurar uma variedade de defeitos. Uma forma concreta de como usar um *checklist* normalmente não é oferecido, ou seja, não é claro quando e baseado em quais informações um inspetor tem que responder uma pergunta particular. Além de que as questões do *checklist* freqüentemente são limitadas para a detecção de tipos particulares de defeito. Essas e outras debilidades no uso de *checklist* são comentadas por Laitenberger [2002].

Como este trabalho tem como foco o DR, *checklists* para a revisão deste documento foram pesquisados, como os apresentados por Pressman & Associates, [2001], Construx [2002], FoxPro Wiki [2000], Firesmith [2005], Mosley [1997]. Alguns apresentaram mais detalhes que outros, mas em geral analisam a qualidade do DR, em termos de consistência de informação, a não ambigüidade, a clareza dos requisitos, a ausência de funcionalidades, contradições, entrada e saída de dados e restrições.

Alguns princípios a serem considerados ao elaborar um *checklist* são listados por Laitenberger [2002] e Brykczynski [1999]:

- O *checklist* não deve exceder uma página, ou seja, não deve ser extenso;

- As questões do *checklist* não devem ser muito genéricas, e sim tão precisas quanto possível;

- O *checklist* deve ser estruturado de tal forma que o atributo de qualidade investigado seja claro para o revisor e as suas questões dêem dicas de como garantir o atributo de qualidade;

- O *checklist* deve ser regularmente atualizado baseando-se na análise dos defeitos;

- Os itens do *checklist* devem ser formulados em forma de questões.

No estudo apresentado por Brykczynski [1999], 117 *checklists* foram analisados a partir de 24 fontes. Nele encontra-se um breve resumo de cada *checklist*, incluindo o número de itens ou questões que eles contêm e para qual produto de trabalho ele é designado. Também são mostradas as diferentes categorias de itens de *checklists* e exemplos de bons itens de *checklists* assim como aqueles que devem ser evitados.

Para o estudo de caso realizado por este trabalho, foi utilizado o *checklist* que se encontra no Anexo 5.

Ao pesquisar os *checklists* para DR, tentou-se encontrar algum tipo de verificação que pudesse ser utilizado como critério de entrada para o processo de inspeção. O intuito era achar algum critério que limitasse o prosseguimento do DR para a fase de inspeção sem antes obedecer a esses critérios mínimos. Porém, não foi encontrado algo que informasse quando o DR estaria num estágio bom que pudesse ser revisado. Isso, de certa forma, também motivou o desenvolvimento desse trabalho.

4.4 Técnica de Leitura Baseada em Perspectiva (PBR – *Perspective Based Reading*)

A PBR foi criada especificamente para revisar os requisitos de software representados em linguagem natural a fim de encontrar defeitos [Shull et al., 2000]. Ela verifica a qualidade das especificações de requisitos ao solicitar que cada membro da equipe de revisores PBR leia o DR sob um determinado ponto de vista ou necessidades de um usuário específico desse documento, ou seja, sob uma perspectiva particular, tais como projetistas, testadores, usuários finais [Shull et al., 2000]. A identificação desses usuários varia conforme a organização e as necessidades do projeto, pois dependendo do ambiente em que se aplicará a PBR, diferente

conjunto de perspectivas pode ser mais apropriado e assim será preciso ajustar a PBR para o uso mais adequado ao ambiente em questão.

Uma vez que os inspetores assumem diferentes perspectivas para revisar o DR, cada um deles encontrará diferentes aspectos desse documento como importante para realizar sua tarefa. Sendo assim, a inspeção PBR oferece um conjunto de revisões individuais, que é definido com base em cenários operacionais, cada um focado num ponto de vista particular de um usuário desse documento. Um cenário operacional requer que o leitor primeiro crie um modelo e então responda as questões baseando-se na análise do modelo com uma ênfase particular [Basili et al., 1996b]. Cada cenário consiste de atividades que um inspetor tem que realizar para ajudá-lo no entendimento do produto de software e elaboração do modelo, e um conjunto de questões para ajudá-lo a identificar os defeitos [Laitenberger, 2002].

Assim, durante o processo de inspeção, cada revisor que avaliar o DR produzirá uma versão de alto nível do produto de trabalho típico da perspectiva que foi assumida, para que possa melhor compreendê-lo. Por exemplo, para um projetista, testador e usuário final, o artefato elaborado seria respectivamente um projeto de sistema de alto nível, um plano de teste do sistema e uma enumeração das funcionalidades e restrições descritas [Shull et al., 2000]. Após a criação dessas representações (modelos), os revisores identificam os defeitos existentes analisando essas representações e respondendo as questões fornecidas pelo cenário. Shull et al. [2000] comentam que quando os requisitos não fornecem informações suficientes para responder às questões, isso significa que também não fornecem informações suficientes para apoiar seus usuários em suas tarefas. Uma listagem contendo os defeitos encontrados deveria ser elaborada para que se pudesse realizar uma posterior correção. Os tipos de defeitos de requisitos que a PBR ajuda a detectar estão listados na Tabela 4.2.

Tabela 4-2 Defeitos de requisitos que a PBR ajuda a detectar [Adaptado de Shull et al, 2000].

Tipo de defeito	Descrição do defeito
Omissão	Qualquer requisito significante relacionado à funcionalidade, desempenho, restrições de projeto, atributos, ou interface externa que tenha sido omitido do Documento de Requisitos.
Informação ambígua	Múltiplas interpretações causadas pelo uso de múltiplos termos para a mesma característica ou múltiplos significados de um termo num contexto particular.
Informação inconsistente	Dois ou mais requisitos que se contradizem.
Fato incorreto	Um requisito declarando um fato que não pode ser verdade diante das condições especificadas para o sistema.
Informação Extra	Informação desnecessária ou não usada (pode confundir os requisitos dos usuários).
Defeitos diversos	Outros defeitos, tais como incluir um requisito na seção errada.

O uso da PBR oferece um número de qualidades benéficas, as quais são enfatizadas por Shull et al. [2000]. São elas:

Focalizada: os revisores focalizam diferentes aspectos do documento, concentrando-se em certos tipos de defeitos do que em todos os possíveis tipos.

Sistemática: os passos a serem seguidos no processo de revisão individual são bem definidos.

Orientada à meta e adaptável: os revisores podem adaptar a técnica para um projeto específico ou uma organização. As perspectivas podem mudar para refletir como uma organização usa o DR.

Transferível via treinamento: como a técnica PBR possui procedimentos bem definidos, os revisores podem receber treinamento em como usá-las.

Essas vantagens da PBR puderam ser comprovadas na prática pelo experimento realizado pela empresa CPqD, que fez uma comparação da técnica *ad-hoc* (sem nenhuma diretriz em como realizar a inspeção, sendo dependente do conhecimento dos revisores) e PBR. Como resultado, a empresa optou por desenvolver o método GVR, Guia de Validação de Requisitos, uma combinação da PBR juntamente com a sua experiência em validação de documentos utilizando-se a técnica *ad-hoc*. O objetivo da empresa CPqD era melhorar o seu processo de elaboração e validação de requisitos, e desenvolver uma técnica que fosse padrão para servir de apoio aos revisores, tornando a inspeção menos dependente do conhecimento e domínio de seus revisores [Pagliuso et al., 2003].

Sendo a PBR uma técnica de inspeção cujo objetivo é apoiar os revisores em descobrir defeitos em requisitos de software, ela está diretamente ligada ao tópico de Engenharia de Requisitos, que tem como produto final o DR, alvo deste trabalho.

4.5 Técnicas de Leitura para Orientação a Objetos (OORTs - Object-Oriented Reading Techniques)

A família de técnicas de leitura OORTs foi desenvolvida por Travassos et al. [1999a, 1999b, 2000, 2002] para apoiar a inspeção de artefatos de projetos de alto nível produzidos segundo o paradigma OO *(Object-Oriented)* e representados em modelos UML *(Unified Modeling Language)*. Elas foram definidas apenas para um subconjunto da notação UML (sendo eles: diagrama de casos de uso, diagrama de classes acompanhado de sua descrição, diagramas de estados e diagramas de seqüência) e para descrição de requisitos, e considera um processo de desenvolvimento genérico e simplificado.

As OORTs consistem de sete técnicas diferentes de leitura, cada uma formada por um conjunto de diretrizes procedimentais, que direcionam a leitura de diferentes diagramas quando os revisores as examinam a fim de encontrar defeitos.

Diferentemente da PBR, o processo de leitura usando OORTs ocorre em duas vias, sendo suas técnicas classificadas como Horizontal ou Vertical. Na Leitura Horizontal, ocorre a comparação entre os documentos que pertencem à mesma fase de projeto para verificar se eles descrevem o mesmo sistema. Já na Leitura Vertical, ocorre comparação de documentos entre fases para verificar se os artefatos do projeto representam o sistema correto, ou seja, se os requisitos foram mantidos, através de sua comparação com o DR e os Casos de Uso. Na Figura 4.1 é ilustrado o conjunto completo das técnicas leitura que compõem as OORTs.

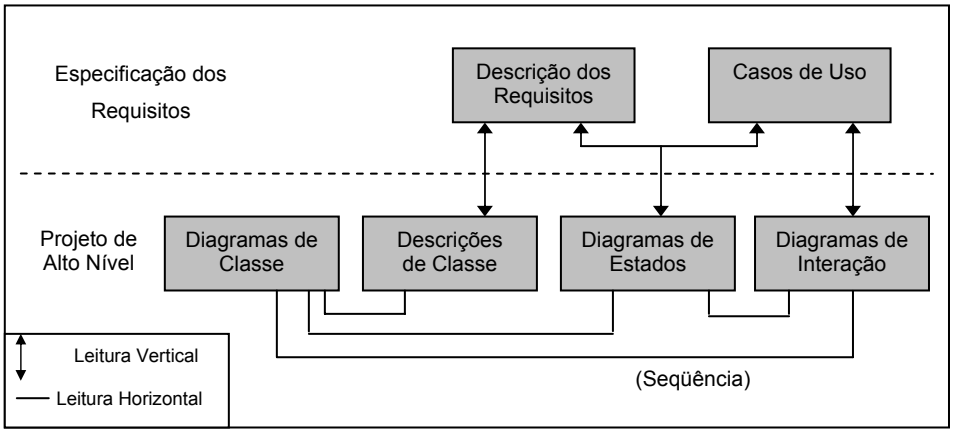


Figura 4.1 Conjunto de Técnicas de Leitura OO [Adaptado de Travassos et al., 2000].

4.6 Técnicas de Leitura para Orientação a Objetos baseada em um Processo específico de Desenvolvimento de Software(OORTs/ProDeS)

Uma outra abordagem de inspeção baseada em técnicas de leitura que inclui o DR como um dos artefatos a serem revisados é a OORTs/ProDeS, definida por Marucci [2002]. Essa técnica é composta de um conjunto de técnicas de leitura para apoiar as atividades de inspeção do processo de desenvolvimento de software OO, denominado ProDeS/UML (Processo de Desenvolvimento de Software para UML) proposta por Colanzi [1999].

Este trabalho de Marucci [2002] foi inspirado nas técnicas OORTs e considerou um processo de desenvolvimento bem definido que utilizasse os modelos UML, pois a UML não possui um processo associado a ela, sendo apenas uma linguagem de modelagem.

O ProDeS/UML usa a notação UML para a confecção de seus artefatos e inclui atividades de teste ao longo de todas as suas fases de desenvolvimento. Esse processo é baseado nas fases do método Fusion [Coleman et al., 1994] para desenvolvimento de software que são: Engenharia de Requisitos, Análise, Projeto e Implementação.

O conjunto das técnicas de leitura proposto para o ProDeS/UML é: duas técnicas (ER1 e ER2) para a Fase de Engenharia de Requisitos, quatro técnicas (A1, A2 A3, A4) para a Fase de Análise e duas outras para a Fase de Projeto (P8 e P9). Todas essas integradas às sete técnicas da OORTs (P1 a P7), compõem o conjunto total de leituras, denominado OORTs/ProDeS, mostradas na Figura 4.2. Portanto, esse apóia a inspeção dos artefatos de todas as fases do processo ProDeS/UML.

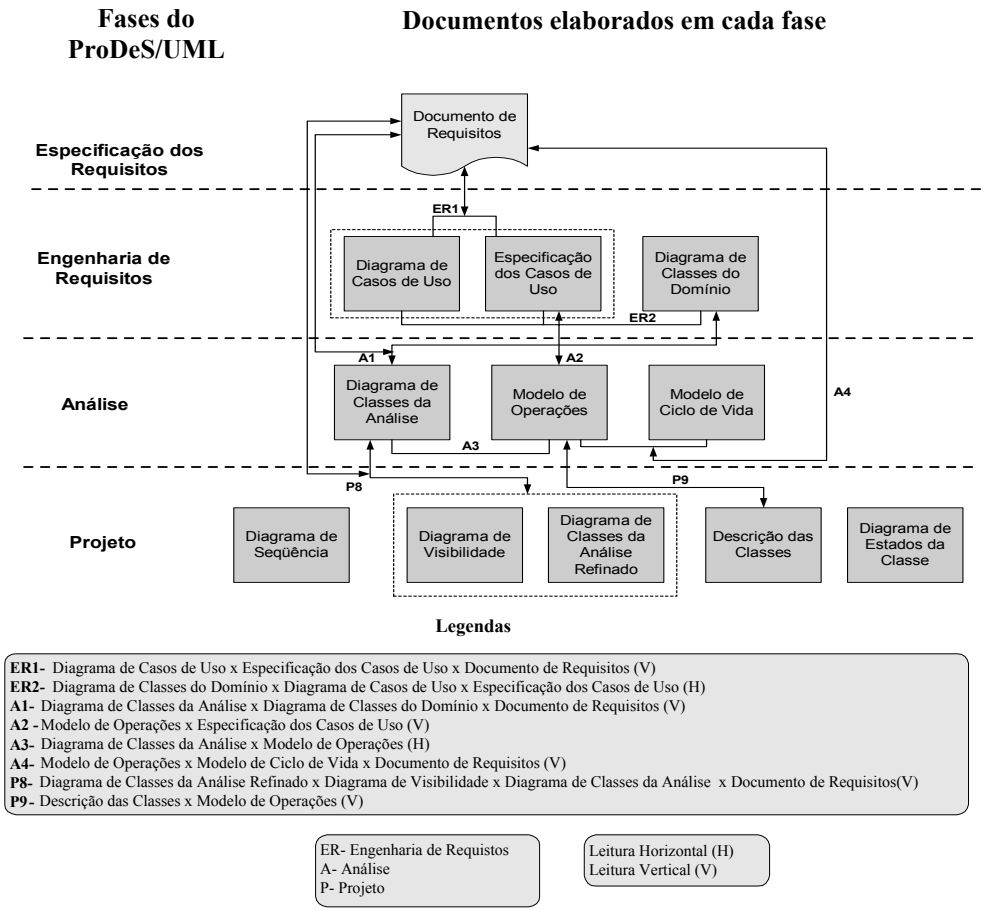


Figura 4.2 Técnicas de Leitura definidas para o ProDeS/UML [Marucci, 2002].

Como este trabalho está no contexto da Engenharia de Requisitos a ênfase para este trabalho se dá na primeira fase do ProDeS/UML. Um dos primeiros artefatos elaborados nessa Fase de Engenharia de Requisitos são o Diagrama de Casos de Uso e suas Especificações, que correspondem à compreensão dos requisitos do sistema com base no DR. Foi considerando

esses aspectos que Marucci [2002] definiu a Técnica de Leitura ER1 (Diagrama de Casos de Uso x Especificação dos Casos de Uso x Documento de Requisitos), que tem como objetivo verificar se os conceitos e funcionalidades que estão descritos no DR foram capturados apropriadamente pelo Diagrama de Casos de Uso e se as Especificações dos Casos de Uso estão descritas de forma precisa. Assim, a técnica ER1 ajuda na detecção de defeitos de requisitos através da validação da consistência entre Diagrama de Casos de Uso e DR.

4.7 Técnica TUCCA (*Technique for Use Case Construction and construction-based requirements Analysis*)[6]

Uma outra técnica de leitura que também inspeciona DR é a TUCCA, que foi proposta por Belgamo [2004]. O objetivo dessa técnica foi definir diretrizes para a construção de Modelos de Casos de Uso (MCU) (Diagrama e Especificação de Casos de Uso) e ao mesmo tempo auxiliar na inspeção tanto desse Modelo quanto do DR. A detecção de defeitos é realizada à medida que a técnica é aplicada na construção do MCU, pois suas diretrizes de construção possuem instruções para inspecionar o DR. Assim, as técnicas TUCCA agregam atividades de construção do MCU e análise do DR. Uma comparação feita entre as técnicas TUCCA e PBR-Usuário quanto à identificação de defeitos em DR pode ser vista em Belgamo & Fabbri [2004b].

Essas técnicas são compostas de duas leituras: AGRT *(Actor Goal Reading Techniques)* e UCRT *(Use Case Reading Techniques)*, as quais basearam-se nas técnicas de leitura PBR com perspectiva do usuário, que utiliza o MCU como apoio para fazer a inspeção do DR, e ER1, da família de técnicas de leitura OORTs/ProDeS descrita anteriormente, que foi definida para fazer a validação do MCU em comparação com o DR. Tanto os aspectos da ER1 quanto da PBR-Usuário foram utilizados e transformados em diretrizes para as técnicas TUCCA, sendo a sua apresentação e formato similares ao da ER1.

O propósito da leitura AGRT é entender o DR para que se possa identificar os possíveis atores e seus objetivos dentro do sistema, além de identificar defeitos no DR. Já a leitura UCRT tem por objetivo construir o MCU (Diagrama de Casos de Uso e suas Especificações) além de também identificar defeitos no DR. A Figura 4.3 mostra como essas técnicas devem ser aplicadas.

181lvii

[6] Técnica anteriormente referenciada como GUCCRA *(Guidelines for Use Case Construction and Requirements Analysis)*

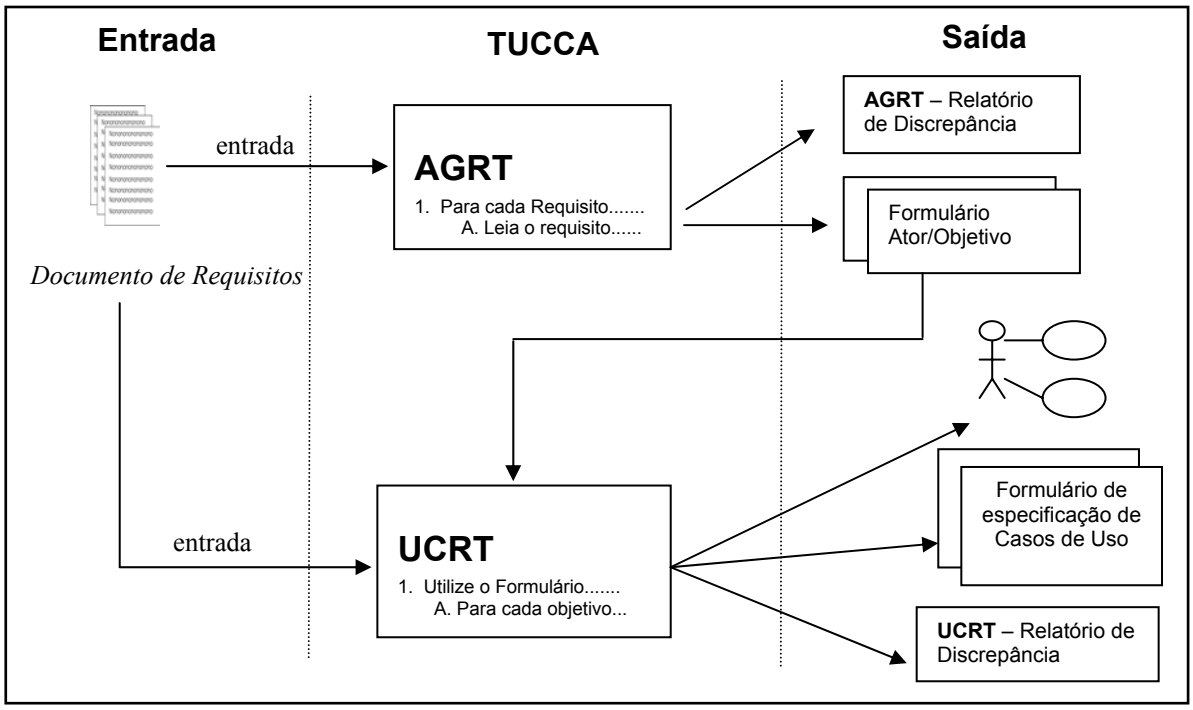


Figura 4.3 Ordem de aplicação das técnicas TUCCA [Belgamo, 2004]

O primeiro passo da técnica TUCCA começa pela aplicação da leitura AGRT, recebendo como entrada o DR, o qual deve estar no padrão IEEE [1998b] ou possuir, ao menos, seções equivalentes às seções Definições, Funções do Produto, Características do Usuário, Requisitos Funcionais e Requisitos Não-Funcionais desse padrão. Essa leitura consiste de várias etapas para se obter o Formulário Ator x Objetivo (FAO), no qual são identificados os atores do sistema, as funcionalidades relacionadas a cada ator e a referência onde estas funcionalidades são encontradas no DR. Para isso, são feitos as marcações dos substantivos com traço para identificar os atores, dos verbos com círculos para descobrir a funcionalidade e das restrições com retângulo. A saída gerada por essa técnica é o FAO e o Relatório de Discrepância, que contém os defeitos encontrados. O segundo passo é a leitura UCRT que recebe como entrada o FAO, gerado anteriormente, e o DR. Nessa leitura, o Formulário de Casos de Uso Preliminares (FCUP), é elaborado a partir do FAO para construir o MCU. Assim, os casos de usos que devem existir são determinados e os relacionamentos entre ele são identificados para que possam ser agrupados, gerando um único caso de uso. As especificações dos casos de uso são descritas e as relações de *include* e *extend* são identificadas. A leitura UCRT gera como saída o MCU e outro Relatório de Discrepâncias contendo as discrepâncias decorrentes da mesma.

Seguindo as diretrizes desses passos tem-se então que a TUCCA, além de facilitar a construção de MCU através de um procedimento mais sistemático, também incorpora em sua

técnica a inspeção do DR. O estudo de viabilidade dessa técnica de inspeção mostrou que a TUCCA torna mais fácil a identificação dos casos de uso que compõem o sistema, aumenta a efetividade em identificar casos de uso e ajuda na tomada de decisão quanto ao agrupamento ou separação de funcionalidade, e os casos de uso identificados fornecem uma representação mais precisa do DR [Belgamo & Fabbri, 2004c].

Como puderam ser observadas pelas abordagens de inspeção mostradas, várias propostas têm surgido para auxiliar essas atividades, sendo que muitas delas apóiam a revisão do DR a fim de que este se apresente de forma mais adequada para dar continuidade ao processo de desenvolvimento de software. O DR é o artefato que dá início a todo esse processo e que, portanto deve receber maior atenção por parte dos desenvolvedores quanto à sua clareza, consistência, completitude, corretitude, enfim, quanto à sua qualidade como um todo.

Ressalta-se que na técnica TUCCA, o DR é de grande importância, visto que os MCU´s gerados baseiam-se nesse documento e a sua elaboração pode ser dificultada pelos inúmeros defeitos encontrados no tal documento. Assim, um DR melhor escrito, que apresente um determinado formato pode facilitar a aplicação e automatização das técnicas TUCCA.

4.8 Considerações Finais

Neste capítulo foram apresentadas as principais técnicas de inspeção voltadas para a validação do DR.

Uma breve descrição do que seria inspeção foi descrita além de cinco técnicas que auxiliam essa atividade, sendo elas: Ckecklist, PBR, OORTs, OORTs/ProDeS e TUCCA.

Exceto a abordagem *Checklist*, as demais técnicas são consideradas Técnicas de Leitura, cujos procedimentos específicos e bem definidos para se realizar a leitura de um documento as classificam como sistemática.

É com relação à técnica TUCCA que a definição deste trabalho teve início. Como esta técnica baseia-se no DR como entrada para a sua aplicação, a forma apresentada por esse documento exerce forte influência para a aplicação de seus passos. Assim, as Diretrizes propostas por este trabalho, as quais são apresentadas no capítulo seguinte, fornecem instruções que tornam mais prática a aplicação da TUCCA e provê um padrão que facilita a implementação da ferramenta que está sendo desenvolvida para a automatização da mesma.

Capítulo

5

5. Diretrizes para elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais

5.1 Considerações Iniciais

Embora haja padrões para elaboração de DR, como pode ser visto no Capítulo 3, Seção 3.2, nenhum deles explica o que realmente deve ser escrito na seção que especifica um requisito funcional, mas apenas cita que o DR deve conter tal seção. Visto que esta é geralmente considerada uma das partes mais importantes de um DR, pois descreve como o sistema de software deve funcionar [Pozgaj et al., 2003], um mínimo de detalhe deve ser exigido para que seja possível entender melhor o que está sendo solicitado para o sistema a ser desenvolvido. Muitas vezes, cada requisito é exposto de uma forma diferente num mesmo DR, alguns abordando mais detalhes que outros ou com as informações agrupadas em um único parágrafo. Desta forma, partes do requisito funcional ficam subentendidas, tendo o desenvolvedor que fazer suposições sobre o que está sendo requisitado.

Outra questão é que o fato de os Requisitos Funcionais estarem descritos de maneiras diferentes dificulta a aplicação da técnica TUCCA [Belgamo, 2004], tornando-a mais demorada, pois seus passos exigem que sejam localizadas e destacadas determinadas características dos requisitos funcionais.

Sendo assim, este capítulo apresenta as Diretrizes para elaboração de DR com enfoque em Requisitos Funcionais, as quais procuram contribuir para a qualidade do DR, evitando muitos dos problemas decorrentes de especificações de software mal escritas, e reduzir o esforço dedicado ao trabalho de inspeção. Além disso, o objetivo principal dessas Diretrizes é facilitar a aplicação da técnica TUCCA [Belgamo, 2004], tornando seus passos mais fáceis de serem seguidos. Essas Diretrizes são compostas de um Formato para especificação de Requisitos Funcionais, de algumas Recomendações de Escrita e de um *Checklist* Pré-Inspeção que antecede o processo de inspeção do DR.

O capítulo está organizado da seguinte maneira: na Seção 5.2, apresentam-se as Diretrizes propostas, sendo esta subdividida nas Seções: 5.2.1, na qual se apresenta o Formato para especificação de um Requisito Funcional; 5.2.2, Seção em que as Recomendações de Escrita são apresentadas e 5.2.3, na qual o *Checklist* Pré-Inspeção está descrito. Por fim, na Seção 5.3 são apresentadas as Considerações Finais.

5.2 Diretrizes propostas para escrita de Requisitos Funcionais

Considerando-se a relevância do Requisito Funcional para um correto entendimento e implementação do sistema, foram elaboradas algumas Diretrizes para a sua especificação. O propósito disso é estabelecer regras que sirvam de apoio para designar o DR num estado bom para servir de base às etapas seguintes à sua documentação.

Primeiramente, antes de começar a descrever os requisitos de um sistema, um padrão de DR deve ser adotado para que se tenha um DR cujo conteúdo apresente uma estrutura de organização bem definida. Como este trabalho pretende facilitar a aplicação da técnica TUCCA e esta recomenda o uso do formato do padrão IEEE [1998b], o mesmo será adotado neste trabalho. Assim, conforme exigido pela TUCCA, o DR deve possuir pelo menos as seções "Propósito", "Escopo", "Definições", "Funções do Produto", "Características do Usuário", "Requisitos Funcionais" e "Requisitos Não Funcionais". O enfoque deste trabalho será dado à seção "Requisitos Funcionais", a qual descreve os serviços ou funcionalidades que devem ser oferecidos pelo sistema. Assim, as Diretrizes propostas incluem:

i) Formato para especificar Requisitos Funcionais, o qual é composto de vários itens que detalham a sua descrição, deixando mais completa e visível a sua função;
ii) Recomendações de como escrever os Requisitos Funcionais são sugeridas;
iii) *Checklist* que deve ser aplicado pelo próprio autor do DR para certificar-se de que determinadas informações não estejam ausentes.

5.2.1 Formato para especificação de um Requisito Funcional

Para a definição do formato proposto para documentar um Requisito Funcional alguns *templates* de DR existentes na literatura, como Volere [2004], IEEE [1998b], CSDL [1996] e Construx [2001] foram examinados para verificar como eles definiam o que seria Requisito Funcional. A maioria explicava-o em forma textual, não apresentando qualquer estrutura para defini-lo melhor. Também foi observada a estrutura dos DR do PG, Hotel e ATM,

encontradas no Anexo 1, 2 e 3, respectivamente. A definição concreta do Formato proposto foi obtida após a aplicação das técnicas PBR-Usuário, *Checklist* e principalmente da TUCCA, nos Anexos 4, 5 e 6, respectivamente, nos DRs citados. Os passos da TUCCA foram determinantes para elaborar um Formato composto por um conjunto de itens separados, pois seus passos mostraram que seria muito mais fácil compreender um requisito caso suas informações estivessem descritas separadamente sob um determinado rótulo, ou seja, discriminadas com determinadas informações.

Com base nos estudos dos DRs citados, verificou-se quais itens seriam necessários para compor um Requisito Funcional, de tal forma que fossem básicos e imprescindíveis para o entendimento dele e não ficassem sobrecarregados de informações para não torná-los uma especificação de caso de uso. Alguns *Checklists* [Pressman & Associates, 2001; Construx, 2002; FoxPro Wiki, 2000; Firesmith, 2005; Mosley, 1997] também foram consultados a fim de avaliar o formato que estava sendo definido.

O formato definido é apresentado na Figura 5.1, o qual apresenta vários itens que melhor caracterizam e definem um Requisito Funcional. A apresentação do requisito funcional desta forma, composta por itens separados, tem como objetivo detalhar as funcionalidades requeridas de um sistema, facilitando a compreensão e utilização desse por vários *stakeholders* que fazem uso do DR.

Requisito Funcional x.
Descrição:
Agente Fornecedor/Receptor:
Agente Executor:
Entrada:
Processamento:
Condição/Restrição:
Saída:

Figura 5.1 Formato de um Requisito Funcional.

Observa-se que, para destacar, cada Requisito Funcional é escrito em itálico e negrito e os itens são sublinhados. Todos os itens são obrigatórios para descrever um Requisito Funcional, pois fornecem os dados mínimos para a sua compreensão e aplicação da técnica TUCCA, exceto o item "Condição/Restrição", que pode não existir. Neste caso, deve-se escrever "Nenhum" ou "Nenhuma" para esse item que não será especificado, ficando claro que este não foi esquecido no momento da elaboração do DR.

Também vale ressaltar que caso se tenha um DR elaborado sem as instruções aqui propostas, e queira-se transformá-lo no formato definido neste trabalho, pode ocorrer de não haver informações suficientes para preencher todos esses itens. Neste caso, deve-se contactar o cliente do sistema para o esclarecimento dos dados faltantes. Ou isso pode indicar que o requisito não seja um Requisito Funcional e esteja situado em local errado do DR.

A seguir, é explicado cada um dos itens especificados no Formato definido, apresentando o objetivo do item, o motivo pelo qual ele foi definido e possível(s) forma(s) sugerida(s) de como deve ser descrito cada item.

I. Identificação do Requisito Funcional

Requisito Funcional x. **:** Este cabeçalho permite que cada requisito funcional seja identificado unicamente através de um número denotado por x. Esse identificador único permite localizar o Requisito Funcional mais facilmente e facilita a sua referência em um outro documento. Também contribui para que se tenha a propriedade 'rastreável para frente', uma das propriedades de qualidade do DR, descrita no Capítulo 3, Seção 3.3.

II. Item "Descrição"

a) Objetivo: Este item serve para explicar, de forma sucinta, a idéia principal da funcionalidade de um requisito, descrevendo "o que" fazer e não "como" fazer. O item "Descrição" deve sempre começar com um verbo, o que facilita identificar a ação ou função principal do requisito funcional, tornando mais clara a tarefa que deve ser realizada pelo sistema.

b) Motivo de definição: Através deste item, obtém-se de forma resumida a função desempenhada pelo Requisito Funcional. A sugestão de começar com um verbo surgiu depois da aplicação da técnica TUCCA no DR do PG, pois se constatou que seria muito mais fácil e prática a sua aplicação caso os verbos que representassem a funcionalidade do sistema estivessem explícitos e diretos já no item "Descrição". A composição da coluna "Objetivo" do FAO, na leitura AGRT da TUCCA, tornar-se-ia mais prática. Além de que um simples verbo principal resume o que de fato um requisito realizará no sistema.

c) Formato sugerido 1:

Descrição: <[*verbo*] [*objeto*]>

[*verbo*]: indica o verbo principal de ação que representa a funcionalidade do sistema.

[*objeto*]: indica o objeto sobre o qual o [*verbo*] tem ação.

Exemplos:

i) Para o DR do Hotel, onde um dos requisitos funcionais é a inclusão do cadastro de hóspedes, tem-se:

Descrição: Cadastrar hóspedes.

ii) Para o DR do PG, que gerencia a entrada e a saída de veículos necessita-se controlar o sensor de passagem para a abertura e o fechamento do portão, então se tem:

Descrição: Controlar o sensor de passagem.

iii) Para o DR do ATM, que automatiza o sistema bancário, é preciso analisar o cartão do banco para verificar se este é válido. Então se tem:

Descrição: Verificar se o cartão é válido.

d) Formato sugerido 2:

Descrição: <[*verbo*] [*objeto*] [*condição*].>

[*condição*]: indica que o sistema irá realizar uma ação conforme a condição especificada.

Este segundo formato é para o caso de haver uma sentença condicional no item "Descrição". A parte que representa a condição deve ser especificada após a sentença que expressa o que o sistema deve fazer.

Exemplo:

i) Para o DR do ATM, um dos requisitos funcionais é a exibição de uma tela inicial no monitor caso nenhum cartão bancário esteja inserido na máquina. Então se tem:

Descrição: "Mostrar tela inicial se nenhum cartão estiver inserido na máquina".

Assim, primeiro descreve-se "Mostrar tela inicial", depois a sentença condicional "se nenhum cartão estiver inserido na máquina".

Observação: No caso em que se queira transformar uma sentença usando o formato aqui proposto, e este contiver em um único parágrafo várias ações, ou seja, vários verbos principais que representam as funções a serem realizadas pelo sistema, este deve ser desmembrado para que cada função seja especificada em um determinado requisito funcional (ver a recomendação Requisitos Múltiplos na Seção 5.2.2). Assim, um único verbo principal de ação é mantido no item "Descrição".

Exemplo:

i) No DR do Hotel tem-se o seguinte requisito funcional:

"1. O sistema deve permitir a inclusão, alteração e remoção de hóspedes do hotel."

Este requisito expressa 3 funções, que seriam incluir, alterar e remover. Todas sendo especificadas em um único requisito. Cada uma destas funções deveria ser especificada separadamente, pois são funcionalidades diferentes. Desta forma, os itens "Descrição" dos Requisitos Funcionais gerados seriam:

Descrição: Incluir cadastro de hóspedes no hotel.

Descrição: Alterar cadastro de hóspedes do hotel.

Descrição: Remover cadastro de hóspedes do hotel.

III. Item "Agente Fornecedor/Receptor"

a) Objetivo: Representar qualquer entidade que interage com o sistema de forma indireta, ou seja, que fornece as informações ou dados de entrada para o sistema e conseqüentemente recebe a resposta, mas depende do item "Agente Executor" para que essas informações sejam transmitidas para o sistema.

b) Motivo de definição: Nos DRs que foram analisados percebeu-se que não havia, de forma explícita, os participantes que interagiam com o sistema. Não ficava claro quem era o ator responsável por determinada funcionalidade requisitada. Assim, para cada requisito funcional foi definido o item "Agente Fornecedor/Receptor", de tal forma que ficasse explícito e fácil de identificar os participantes de um determinado Requisito Funcional. Além de que facilita no uso da notação UML, pois seus diagramas fazem referência a atores do sistema e este item "Agente Fornecedor/Receptor" seria o tal correspondente. Sendo a modelagem realizada a partir do DR, a exibição do Agente Fornecedor/Receptor facilita a sua

elaboração, evitando que participantes diferentes sejam associados a um requisito erroneamente.

Outra questão é que a aplicação da técnica PBR-Usuário torna-se mais prática, assim como a TUCCA, pois ambos exigem que sejam mostrados os participantes do sistema. Na TUCCA, este item facilita o preenchimento do FAO, onde os atores são identificados. Após ter verificado que tais técnicas de inspeção seriam beneficiadas se já no DR cada requisito fosse discriminado com o participante do sistema, optou-se por incluir no formato do Requisito Funcional o item "Agente Fornecedor/Receptor".

c) Formato:

Agente Fornecedor/Receptor: <[*agente n*]>

[*agente n*]: indica o agente que interage de forma indireta, podendo haver um ou mais agentes.

Exemplos:

i) Para o DR do Hotel, considerando-se a inclusão de hóspedes, tem-se:

Agente Fornecedor/Receptor: Hóspede

ii) Para o DR do PG, considerando-se o pedido de ingresso para a entrada de motorista no estacionamento, tem-se:

Agente Fornecedor/Receptor: Motorista

IV. Item "Agente Executor"

a) Objetivo: Representar qualquer entidade que interage com o sistema de forma direta, ou seja, informa ao sistema as informações obtidas do "Agente Fornecedor/Receptor". É um agente que opera, que executa o sistema. Obviamente, os agentes Fornecedor/Receptor e Executor podem ser os mesmos.

b) Motivo de definição: O item "Agente Executor" foi definido para evitar que participantes que intermediam com o sistema sejam considerados sempre os atores solicitantes de um requisito, pois muitas vezes se confundem o participante que interage fornecendo dados para a realização de um requisito daquele que apenas faz a intermediação com o sistema. Essa é uma das grandes confusões que se cometem quando se elaboram MCU. Desta

forma, a separação dos itens "Agente Fornecedor/Receptor" e "Agente Executor" foi sugerida.

c) Formato:

Agente Executor: <[*agente n*]>

[*agente n*]: indica o agente que interage de forma direta, podendo haver um ou mais agentes.

Exemplos:

i) Para o DR do Hotel, considerando-se a inclusão de hóspedes, tem-se:

Agente Executor: Funcionário

ii) Para o DR do PG, considerando-se o pedido de ingresso para a entrada de motorista no estacionamento, tem-se:

Agente Executor: Motorista

V. Item "Entrada"

a) Objetivo: Neste item são descritas as informações de entrada, ou seja, o(s) dado(s) que o sistema recebe e processa para a execução de uma determinada funcionalidade. A Tabela 5.1 foi especificada para descrever melhor cada atributo de entrada.

b) Motivo de definição: Para que um determinado requisito funcional seja executado é necessário indicar o que o faz iniciar a sua execução, pois o sistema não opera nada de forma automática, como é comum observar em muitos DRs, com o uso do termo "automaticamente". Esse termo é ambíguo e vago, pois é preciso saber qual a entrada, o estímulo, o sinal que provoca a ação de um requisito funcional. No padrão IEEE [1998] é dito que os requisitos deveriam incluir no mínimo uma descrição de cada entrada (estímulo) no sistema.

Outra razão para se ter o item "Entrada" é que a maioria das técnicas de inspeção de DR, como a TUCCA, verifica a completitude das informações de entrada do requisito funcional para a sua realização. Também, o DR torna-se mais proveitoso para os Desenvolvedores, pois como pode ser observado na técnica PBR-Desenvolvedor, que consiste em checar o DR para verificar se ele contempla todas as necessidades do Desenvolvedor, os

dados de entrada devem ser detalhados, ou seja, o seu tipo deve ser definido (exemplo: a precisão e a unidade requerida).

Modelo da tabela de entrada:

Campo	Tipo	Tamanho	Descrição

Tabela 5-1 Tabela de dados de entrada.

Onde:

- *Campo*: refere-se ao nome do campo ou atributo.
- *Tipo*: refere-se aos seguintes tipos:
  - Numérico: somente números são permitidos para o determinado campo.
  - Alfabético: somente caracteres alfabéticos são permitidos para o determinado campo.
  - Alfanumérico: tanto números, caracteres alfabéticos e símbolos são permitidos para o determinado campo.
- *Tamanho*: refere-se à quantidade de caracteres que um campo tipo alfabético ou alfanumérico possui ou ao tamanho ocupado por um tipo numérico.
- *Descrição*: refere-se a alguma descrição complementar do campo, não sendo obrigatória. Ex: o formato da data deve ser dd/mm/aaaa; o estado deve ser apenas as abreviaturas.

c) Formato sugerido 1:

Entrada: <[*descrição*] [*Tabela 5.1*].>

[*descrição*]: é uma breve descrição de uma lista de atributos, normalmente no caso de cadastro, relatório, consulta; é opcional.

Exemplos:

i) Entrada:

Campo	Tipo	Tamanho	Descrição
Número do ingresso	Numérico	8 inteiros	

ii) Entrada:

Campo	Tipo	Tamanho	Descrição
Sinal de que um cartão foi inserido na máquina	Numérico	1 inteiro	0 = cartão não inserido 1 = cartão inserido

iii) Entrada: O cadastro de funcionários deve conter os seguintes atributos:

Campo	Tipo	Tamanho	Descrição
Nome	Alfabético	40 caracteres	Nome completo do funcionário
Endereço	Alfanumérico	50 caracteres	
Cidade	Alfabético	30 caracteres	
Data de Nascimento	Alfanumérico	8 caracteres	Formato: dd/mm/aaaa
Telefone	Numérico	10 inteiros	Formato: (xx) xxxx-xxxx.

d) Formato sugerido 2:

Entrada: <[*descrição*] [*Tabela 5.1 rotulada*].>

[*Tabela 5.1 rotulada*]: é a Tabela 5.1 com um rótulo para separar os diferentes tipos de entrada. Exemplos de rótulos ou títulos: Campos a Consultar, Campos a Imprimir, Campos a Analisar, Campos a Alterar.

Exemplo:

No caso de alteração de cadastro, onde é preciso informar o campo de entrada para localizar o cadastro e mais os campos a serem alterados, o item "Entrada" deve ser da seguinte forma:

i) Entrada:

Campo	Tipo	Tamanho	Descrição
Número da acomodação	Numérico	4 inteiros	
Campos a Alterar			
Andar	Numérico	2 números	Andar no qual se encontra a acomodação
Código do tipo de acomodação	Alfanumérico	6 caracteres	

A única diferença nesta tabela é a separação do campo de entrada usado para localizar um determinado cadastro dos demais campos, neste caso os campos a serem alterados, que são colocados sob o rótulo Campos a Alterar.

VI. Item "Processamento"

a) Objetivo: Ao contrário do item "Descrição", que é bem geral, neste item é descrito com mais detalhes o fluxo de atividades ou ações que o sistema deve realizar para alcançar o que foi escrito no item "Descrição" e assim completar e caracterizar o requisito funcional. Dessa forma, indica as ações que devem ser desempenhados pelo sistema a fim de concretizar a funcionalidade requerida. Para facilitar a identificação do fluxo de atividades desse item, pode-se fazer a seguinte pergunta: 'Para realizar este Requisito Funcional, o que deve ser validado, comparado, verificado, realizado ou executado, como parte para obtenção desse requisito?'.

b) Motivo de definição: Incluindo este item no formato proposto, fica mais fácil entender o fluxo de atividades necessário para executar um requisito funcional. Facilita principalmente no momento da especificação do curso normal e alternativo de casos de uso, portanto auxilia na aplicação da leitura UCRT da TUCCA.

c) Formato sugerido:

Processamento: <[*verbo*] [*objeto*]. {[Em caso afirmativo,] [*ação positiva*]. [Senão,] [*ação negativa*]}.>

[*verbo*]: indica o verbo de ação que representa o processamento do sistema.

[*objeto*]: indica o(s) objeto(s) sobre o qual o [*verbo*] tem ação.

{ }: indica que ocorre somente no caso de haver *ação positiva* e *negativa*.

[*ação positiva*]: indica a ação a ser executada se a resposta do processamento for positiva.

[*ação negativa*]: indica a ação a ser executada se a resposta do processamento for negativa.

Exemplos:

i) Processamento: Iniciar o processo de autorização.

ii) Processamento: Verificar se há vagas livres. Em caso afirmativo, emitir ingresso. Senão, mostrar mensagem de erro.

iii) Processamento: Verificar se a quantidade de dinheiro é menos do que **n**. Em caso afirmativo, mostrar uma mensagem de erro e devolver o cartão. Senão, seguir o processamento.

Observações: Ao escrever a sentença do "Processamento", deve ser verificado que para expressar uma seqüência de ações repetidas em dois objetos diferentes, deve-se expressar as duas ações explicitamente e evitar a fatoração de objetos [Ben Achour, 1998].

Exemplo:

Forma incorreta.

Processamento: O sistema deve checar o nome. O sistema deve checar a senha.

Forma correta:

Processamento: O sistema deve checar o nome e a senha.

Similarmente, se um objeto participa de duas ações consecutivas, evitar sua fatoração [Ben Achour, 1998].

Exemplo:

Forma incorreta:

Processamento: A senha é checada. A senha é modificada.

Forma correta:

Processamento: A senha é checada e modificada.

VII. Item "Condição/Restrição"

a) Objetivo: Como o próprio nome diz, este item indica a condição ou restrição necessária para que seja realizada a funcionalidade requerida do sistema. Descreve as circunstâncias das quais a funcionalidade depende para desempenhar uma função.

b) Motivo de definição: Foi percebido nos DRs analisados que apenas os dados do item "Entrada" não eram suficientes para que alguns Requisitos Funcionais fossem executados, pois muitas vezes estes estavam associados a alguma condição ou restrição que impedia ou que deveria ser satisfeito para que eles fossem efetuados. Para isso, o item "Condição/Restrição" foi estabelecido como um dos itens que descrevem um Requisito Funcional. Também facilita a marcação que é feita na aplicação da leitura AGRT da técnica TUCCA, a qual requer um retângulo sobre a condição ou restrição atrelado a um requisito. Além de que este item também ajuda na especificação de casos de uso.

c) Formato sugerido 1:

Condição/Restrição: <[*condição/restrição*].>

[*condição/restrição*]: indica que o sistema irá realizar uma ação conforme a condição/restrição especificada. Normalmente, inicia-se pelos termos "quando", "para isso", "no caso", "se", "desde que", "ao [verbo]".

Se algum desses termos, que dão indício de que uma condição/restrição está associada ao requisito, aparecer na descrição do item "Descrição", é preciso verificar se realmente é uma condição/restrição. Caso isso seja subentendido, deve-se repetir a parte que representa a condição também neste item "Condição/Restrição". Nem sempre uma condição ou restrição aparece explícita no item "Descrição".

Exemplos:

i) Para a seguinte "Descrição" de um Requisito Funcional do DR do Hotel:

Descrição: Recuperar os dados da reserva de uma acomodação durante a entrada do hóspede se uma reserva prévia foi feita.

Tem-se a "Condição/Restrição":

Condição/Restrição: Se uma reserva prévia da acomodação foi feita.

Assim, a recuperação dos dados da reserva de uma acomodação durante a entrada do hóspede será realizada se e somente se uma reserva prévia da acomodação foi feita. Portanto, este Requisito Funcional está vinculado a uma condição a ser satisfeita.

ii) Para a seguinte "Descrição" de um Requisito Funcional do DR do Hotel:

Descrição: Imprimir o histórico de estadias do hóspede no hotel.

Tem-se a "Condição/Restrição":

Condição/Restrição: Para isso, o hóspede deve ter sido previamente cadastrado e deve portar de um código de identificação e de uma senha.

Para realizar este Requisito Funcional, uma determinada restrição foi identificada.

d) Formato sugerido 2:

Condição/Restrição: <([Requisito Funcional y. concluído] *[com sucesso ou sem sucesso]).>*

[*com sucesso*]: implica que um determinado Requisito Funcional x depende do Requisito Funcional y e ainda, esse Requisito Funcional y deve ter finalizado com sucesso. O

uso do termo "com sucesso" significa que a ação desejada do Requisito Funcional y, escrita no seu item "Descrição", foi alcançada, ou seja, teve uma resposta positiva.

[*sem sucesso*]: implica que um determinado Requisito Funcional x depende do Requisito Funcional y e ainda, esse Requisito Funcional y deve ter finalizado sem sucesso. O uso do termo "sem sucesso" significa que a ação desejada do Requisito Funcional y, escrita no seu item "Descrição", não foi alcançada, ou seja, teve uma resposta negativa.

Este formato é usado no caso em que um Requisito Funcional depende da finalização de algum outro Requisito Funcional, ou seja, quando necessita que um ou mais Requisitos Funcionais sejam executados anteriormente. Assim, colocar entre parênteses no item "Condição/Restrição" a identificação do requisito do qual ele depende.

Exemplo:

i) Tendo o seguinte Requisito Funcional y do DR do Hotel:

Requisito Funcional y.
Descrição: Processar a saída do hóspede do hotel.

O item "Condição/Restrição" do Requisito Funcional x seria:

Requisito Funcional x.
Descrição: Imprimir um comprovante de saída do hóspede.
...
Condição/Restrição: (Requisito Funcional y. concluído com sucesso).

Portanto, no exemplo mostrado, o Requisito Funcional x apenas irá imprimir o comprovante de saída do hóspede se houver o processamento da saída do determinado hóspede, ou seja, a resposta do Requisito Funcional y for positiva.

VIII. Item "Saída"

a) Objetivo: Este item indica a resposta do sistema dada uma determinada funcionalidade. Descreve a saída gerada pelo sistema de acordo com o Requisito Funcional especificado. A "Saída" pode ser uma alteração do estado ou das variáveis do sistema (modificação interna) ou pode ser algo externo, visível, como uma mensagem de erro, impressão de relatório, espera por uma resposta.

b) Motivo de definição: A definição do item "Saída" foi considerada uma vez que toda função desempenhada pelo sistema gera uma saída como resposta. Após a execução de um requisito uma resposta é sempre esperada. Conforme o padrão IEEE [1998b], cada saída (resposta) do sistema deve ser descrita. Além de que tal item facilita no momento da checagem feita pelos casos de testes, ou seja, é possível verificar se a saída gerada está conforme definido na especificação. Esse item contribui para satisfazer a propriedade 'verificável', uma das propriedades de qualidade do DR.

c) Formato sugerido 1:

Saída: <[*verbo*] [*objeto*].>

[*verbo*]: indica o verbo de ação esperado na saída.

[*objeto*]: algum complemento da sentença de saída.

Nos casos em que a "Saída" tem respostas positivas e negativas, devem ser indicadas as duas para o mesmo requisito. Assim, evita que haja requisitos separados apenas para mostrar uma com saída positiva e outra com saída negativa, quando elas têm as mesmas "Entradas".

Exemplos:

i) Saída: Armazenar os dados do hóspede.

ii) Saída: Apresentar novo valor para a variável **r**.

iii) Saída: Abrir portão ou nada acontece.

iv) Saída: Fechar ou abrir o portão, dependendo do sinal recebido.

d) Formato sugerido 2:

Saída: <Um(a) [mensagem|aviso|...] de [erro|alerta|êxito|...] deve ser [exibida|mostrada|emitida|apresentada|...] [indicando|alertando|apontando|...] que [*razão da mensagem*].>

[*razão da mensagem*]: explica o motivo pelo qual a mensagem está sendo mostrada. Normalmente explica o sucesso ou insucesso de determinado evento.

Quando a "Saída" for uma mensagem que deve ser exibida na tela para o usuário do sistema, essa deve ser escrita de tal forma que explique qual o conteúdo da mensagem a ser transmitida.

Exemplos:

i) Saída: Uma mensagem de erro deve ser exibida indicando que o número digitado é incorreto.

ii) Saída: Uma mensagem de erro deve ser exibida alertando que os dados estão incompletos.

iii) Saída: Uma mensagem deve ser exibida indicando que não foi possível localizar o cliente X.

Desta forma, fica mais clara a compreensão da "Saída" desejada.

Os itens que foram apresentados formam um conjunto de informações que caracterizam cada requisito funcional. Como pode ser observado, são informações básicas que deveriam fazer parte do próprio requisito, mas que muitas vezes acabam por ser esquecidas ou consideradas irrelevantes para o momento. O uso deste formato obriga que tais dados sejam coletados durante a elicitação de requisitos e documentados no DR.

Para complementar estas Diretrizes, algumas Recomendações para escrever um Requisito Funcional foram estudadas e compiladas para apoiar no entendimento e clareza de um requisito. Na seção seguinte, são mostradas tais Recomendações.

5.2.2 Recomendações de Escrita

Além do Formato proposto para descrever um Requisito Funcional, aqui são dadas algumas Recomendações para especificá-lo, as quais fazem parte das Diretrizes propostas. Dessa forma, tenta-se direcionar/orientar a redação dos Requisitos Funcionais para que apresente os critérios de qualidade desejável para ser inspecionada. São mostrados quais pontos devem ser observados.

Requisitos múltiplos

Ao escrever uma funcionalidade do sistema a ser elaborado, deve-se especificá-la em sentenças curtas e simples, atentando-se para evitar que requisitos múltiplos sejam agregados numa única sentença ou frase. Para evitar esses requisitos múltiplos, uma sentença de especificação de requisitos não deve:

- ter mais que um sujeito ou mais que um verbo principal
- ter mais que um complemento direto ou indireto que especifica seu sujeito

- ter conjunções como “e” e “ou”, pois isto sugere que vários requisitos tenham sido combinados.

Exemplo de requisito múltiplo: O sistema deve incluir e alterar dados de cliente.

Tal recomendação foi observada por diversos autores, como Fabbrini et al. [2000], Oriel [1999], Wiegers [1999a].

Acrônimos

Ao usar acrônimos numa sentença de requisitos, explicá-los explicitamente e completamente dentro do DR, para tornar claro e legível a sentença. Deve-se defini-los em glossário ou dicionário de dados. Por exemplo: UML deve ser definido no glossário. De acordo com Fabbrini et al. [2000], uma sentença de um Requisito Funcional torna-se inexplicável se este contém acrônimos que não estejam explicados dentro do próprio DR.

Termos consistentes

Usar termos consistentes, ou seja, não usar mais que um termo ou palavra para referir-se a mesma coisa. Por exemplo, ao usar a palavra “cadastrar” para uma funcionalidade, deve-se usar essa mesma denominação sempre que se referir a ela e não “inserir”, “incluir” ou qualquer outra palavra semanticamente correspondente. Desta forma previne-se a confusão de colocar no DR uma nova funcionalidade.

Essa recomendação teve como base ao se aplicar a técnica TUCCA [Belgamo, 2004] nos DRs estudados. Como diversos termos inconsistentes foram verificados, achou-se melhor apontá-lo nessas recomendações.

Requisitos redundantes

Evitar declarar requisitos de forma redundante num DR, pois o torna mais difícil de manter. Muitas vezes, com o intuito de explicar melhor ou enfatizar um Requisito Funcional, encontra-se ele repetido em várias seções do DR, mas isso pode atrapalhar a atualização do DR.

As propriedades de qualidade do DR apresentadas no Capítulo 3, Seção 3.3, abrangem a não redundância como uma das características que contribui para uma boa especificação de requisitos. O autor Wiegers [1999a], em seu guia para escrever requisitos com qualidade,

também menciona que não se deve ter requisitos redundantes, pois embora o mesmo requisito em diversos lugares possa facilitar a leitura do documento, isso o torna difícil de manter. A atualização das diversas instâncias de um requisito pode gerar inconsistência.

Frases imperativas

Não se deve usar a forma imperativa ao escrever uma sentença de requisitos, pois não é uma ordem direcionada à pessoa que está lendo o DR, ou seja, não é uma ordem dada ao desenvolvedor que irá implementar tal requisito e sim um requisito sendo solicitado para o sistema a ser desenvolvido. Por exemplo, as sentenças: "Armazene os parâmetros", "Mostre uma mensagem de erro", "Atualize o arquivo" devem ser evitadas. O correto seria, "O sistema deve armazenar os parâmetros", "Uma mensagem de erro deve ser mostrada", "O sistema deve atualizar o arquivo".

Essa recomendação foi definida quando o DR do ATM foi estudado. Diversas sentenças que descreviam os Requisitos Funcionais estavam expressas na forma imperativa. Para tornar uma sentença mais clara e objetiva tal recomendação foi sugerida.

Marcador "/"

Não se deve usar o marcador "/" numa especificação de requisitos. Quando se escreve A/B, essa notação pode gerar confusão, pois vários significados podem ficar subentendidos, a saber: "A ou B"; "A ou B ou ambos"; "A e B"; "números de A dividido por números de B".

Pode-se dizer que o marcador "/" gera ambigüidade, pois diversas interpretações podem ser entendidas. Assim, essa recomendação de Oriel [1999] foi incluída nas Recomendações de Escrita desse trabalho.

Composição de lista

Ao compor uma lista no texto da especificação de requisitos, deve-se atentar para que ela seja o mais completa possível e fácil de ler. Todos os seus elementos devem ser explicitamente especificados (evitar usar "e outros", "assim por diante") e listados verticalmente usando rótulos que os identificam ("a.", "b.", "c.", ou "1.", "2.", "3."). E ainda, todos os seus elementos devem consistir da mesma classe de palavras. Exemplo:

a. vassoura,
b. lixeira,
c. pá, e
d. limpar.

O item "d. limpar" é um verbo e todos os outros elementos são substantivos. Esta lista deveria ser toda constituída somente de substantivos.

Dada a importância da completitude da descrição do Requisito Funcional, essa recomendação citada por Oriel [1999] foi aqui considerada.

Termo "e/ou"

Nunca usar "e/ou" numa sentença de requisitos, pois gera confusão. Seu uso permite ao leitor a liberdade de escolher se a sentença deve ser lida "e" ou então "ou", qualquer que for a leitura que seja mais fácil de realizar ou satisfazer.

Segundo Oriel [1999], devido ao custo que pode resultar do uso desse termo, deve-se evitá-lo na especificação.

Palavra "qualquer"

Evitar usar a palavra "qualquer" numa sentença de especificação de requisitos para não causar mal-entendimento. Quando "qualquer" é usado para descrever a seleção dos itens de um conjunto, é o leitor quem seleciona determinados itens e não quem escreveu o requisito. Qual(is) e quantos itens o leitor seleciona depende de seu ponto de vista. Além de que quem escreveu o requisito pode usar a palavra "qualquer" com a intenção de denotar "pluraridade" e o leitor pode interpretá-lo para denotar "singularidade", apenas "um dos". Por exemplo: A opção para localizar o cadastro de cliente pode ser efetuada através da indicação do nome do cliente, do seu RG, do código único de cada cliente, qualquer que seja a mais conveniente.

Sendo assim, seguindo as orientações citadas por Oriel [1999] quanto ao uso da palavra "qualquer", tal recomendação foi considerada.

Expressão "assim como"

Não usar a expressão "assim como" em sentenças de especificação de requisitos, pois gera ambigüidade. Por exemplo: Fazer a tarefa A assim como a tarefa B. Isso gera duplo sentido: uns podem entender que a tarefa A e B devem ser feitas e outros podem entender que ambas as tarefas sejam igualmente bem feitas.

Essa recomendação também teve como base às orientações quanto ao uso de palavras e frases dada por Oriel [1999].

Palavras ambíguas

Ao especificar um requisito deve-se evitar o uso de palavras inerentemente ou potencialmente ambíguas. O uso de termos ambíguos em requisitos funcionais faz com que as interpretações do requisito também sejam ambíguas.

Diversos autores mencionam quais seriam as palavras ambíguas, como Fabbrini et al. [2000], Firesmith [2003], Kar & Bailey [1996], Oriel [1999]. Pode-se considerar a classificação das palavras ambíguas da seguinte forma:

- Sujeitos vagos que podem referir-se às coisas múltiplas:
  - Pronomes, tais como: "eles", "elas".
  - Adjetivos demonstrativos, tais como: "este(s)", "esta(s)", "aquele(s)", "aquela(s)".
- Adjetivos vagos que podem significar coisas diferentes para leitores diferentes:
  - Características intrínsecas, tais como: macio, flexível, suave, rígido, rápido, lento, demorado, quente, frio, forte, resistente, potente, robusto, fraco, bem, forte, ruim, baixo, devagar.
  - Características passíveis de julgamento ou opinião, tais como: fácil, difícil, simples, claro, eficiente, aceitável, adequado, bom, mal, razoável, suficiente, útil, significante, amigável.
  - Características de localização, tais como: perto, longe, término, próximo.
  - Adjetivos de ordem, tais como: primeiro, último, prévio, próximo, seguinte, anterior.
  - Características temporais, tais como: novo, velho, recente, futuro, passado, logo, hoje.
- Preposições vagas:

- Preposições, tais como: acima, abaixo, em frente de, atrás de, sobre, por cima de, superior a, sob, ao alto, baixo.

- Verbos vagos que são mais qualitativos do que quantitativos:
  - Verbos, tais como: maximizar, minimizar, aumentar, diminuir, otimizar.
- Termos subjetivos, que são aqueles que se referem a uma opinião pessoal ou sentimento, tais como:
  - tendo em mente, levando em conta, levando em consideração...
  - similar, semelhante, melhor, pior...
  - tão [adjetivo] quanto possível.
  - se possível, quando custo-efetivo, onde apropriado...
- Termos opcionais, que são aquelas que contém uma opção. Normalmente aparecem palavras tais como:
  - possivelmente, eventualmente, se apropriado, se necessário, se praticável...
- Os termos:
  - não menos que, não mais que, tolerante a falha, ser capaz de, suporta, capacidade de, freqüentemente, normal, usual, amplo, intuitivo, melhorado, flexível, "e/ou", "etc", não limitado a, automaticamente, imediatamente, periodicamente, preciso, possível, seguro, outros, vários.

Sentenças pouco expressivas

Sentenças pouco expressivas, que são aquelas que contêm verbos fracos (por exemplo: pode/poderia/deveria), devem ser evitadas numa sentença de especificação de requisitos, pois elas transmitem insegurança a respeito do que está sendo solicitado pelo requisito. Por exemplo: O sistema poderia permitir a emissão de comprovante de saída. O uso de verbos fracos em Requisitos Funcionais dá a impressão de que o requisito não é necessário, ou seja, a escolha de implementá-lo ficaria a cargo do desenvolvedor.

Essa recomendação foi sugerida após a observação do indicador de qualidade do Modelo de Qualidade para especificação de DR descrita por Fabbrini et al. [2000]. Wilson [1997], em suas diretrizes para elaboração de DR, descreve algo semelhante ao recomendar o não uso de palavras ou termos, tais como "pode", "se requerido", "como apropriado", "se praticável", que sugerem opção quando, na verdade, o requisito realmente deve ser satisfeito.

Referências explícitas

Quando o DR fizer alguma referência a um determinado documento, sentença ou entidade, ou seja, referir-se explicitamente a algo, deve-se atentar para que a referência esteja devidamente indicada no DR, pois uma sentença de uma especificação de requisitos pode referir-se a:

- Sentenças não numeradas do próprio DR.
- Documentos não referenciados no próprio DR.
- Entidades não definidas nem descritas dentro do próprio DR.

Os seguintes termos são considerados referências explícitas: de acordo com, com base em, relativo a, conforme.

Cada referência citada deve ser unicamente identificada.

O uso de referências explícitas foi considerado por Fabbrini et al. (2000) no Modelo de Qualidade para avaliar a qualidade de uma especificação escrita em linguagem natural. Wilson [1997] também recomenda que todas as referências sejam identificadas na seção do DR que tem este propósito.

Seguindo estas recomendações de escrita tenta-se adequar o DR para que este não esteja inconsistente, ambíguo e incompleto, e que contribua para apresentar as propriedades de qualidade do DR.

5.2.3 Checklist Pré-Inspeção

Como parte das Diretrizes para elaboração do DR, um *Checklist* Pré-Inspeção foi elaborado como forma de verificar os requisitos funcionais descritos. Mesmo usando o Formato proposto e as Recomendações de Escrita, o escritor do DR deve certificar-se de que seu produto esteja numa forma que poucos ou nenhum defeito seja encontrado. O objetivo é impactar o menos possível o processo de inspeção, ou seja, exigir menos esforço e re-trabalho.

Terminada a elaboração de DR, a sua inspeção é iniciada como forma de validar aquilo que foi documentado. Porém, a atividade de inspeção com a aplicação das técnicas de leitura está associada com um alto custo, demandando tempo para a sua execução, dinheiro e emprego de pessoas com conhecimento adequado. Não valeria a pena submeter um DR para validação sem que este apresentasse um mínimo de qualidade.

Outra questão é que mesmo usando determinados *checklist* para inspeção, cuja aplicação requer menos esforço, também não faria sentido. Isso porque segundo Brykczynski [1999], alguns itens do *checklist* seriam melhores usados como critério de entrada ou saída anterior à inspeção. Seria mais eficaz caso uma única pessoa verificasse determinadas propriedades ou ações ao invés de se ter uma equipe completa de inspeção para realizar tal checagem.

Por esta razão, um *Checklist* Pré-Inspeção foi proposto como critério de entrada para o DR poder iniciar um processo de inspeção. O objetivo é fornecer dicas que previne certos erros freqüentes de serem cometidos e que determine quando um DR está pronto para ser submetido a um processo de inspeção. Embora sejam amplamente comentadas e diversos estudos divulguem as dificuldades de se ter um DR de boa qualidade, não foi encontrado, até o estudo desse presente trabalho, assuntos que abordem os requisitos de quando ingressar a inspeção de um DR, de que forma um DR deve se apresentar para que qualquer um não seja examinado, ou seja, de como selecionar um bom DR.

Assim, o *Checklist* Pré-Inspeção deve ser usado pelo próprio autor do DR, assim que este for concluído, como forma de verificar a qualidade do DR antes de prosseguir direto com a inspeção.

Além da conformidade do DR com o formato do padrão IEEE [1998], do uso do Formato para especificação de Requisitos Funcionais e Recomendações de Escrita propostas, o *Checklist* Pré-Inspeção exposto abaixo deve ser procedido.

1. Mensagens que devem ser exibidas ao usuário do sistema foram definidas e para qual finalidade (alerta, erro, sucesso, status, aviso,...)?
2. Variáveis definidas que estão relacionadas a outras são devidamente atualizadas quando a variável associada a elas se altera?
3. A partir da descrição dos requisitos funcionais é possível especificar curso normal e alternativo de um MCU?
4. Caso um requisito funcional apresente mais de uma possibilidade de saída como resposta de sua execução, tais saídas foram especificadas?
5. Os requisitos funcionais que solicitam alteração de dados especificam quais deles podem ser alterados?
6. Os requisitos funcionais que solicitam remoção, alteração e consulta a dados armazenados, os quais necessitam localizar dados, informam como será feita a busca, ou seja, a partir de qual dado as informações serão localizadas?
7. No caso em que um requisito funcional interage com outro através da solicitação de serviço, o qual é executado por este outro requisito, os dados enviados e recebidos de um requisito e outro são coerentes?
8. Requisitos funcionais que realizam comparação (<,>,=) apresentam informações do que deve ser feito em todos os casos?
9. Requisitos funcionais que realizam verificação / checagem, por exemplo, se é válido ou inválido, se correto ou incorreto, normalmente os que permitem apenas duas possibilidades, apresentam informações do que deve ser feito em ambos os casos?

Figura 5.2 *Checklist* Pré-Inspeção.

O *checklist* apresentado tenta ser menos genérico em relação aos *checklist*s usados para inspecionar documento, pois são questionadas principalmente as ausências de certas informações que deveriam estar descritas nos Requisitos Funcionais. Tais perguntas foram baseadas nos estudos realizados usando os DRs que se encontram nos Anexos 1, 2 e 3. Esse *Checklist* Pré-Inspeção dá suporte ao autor do DR ao verificar se a descrição dos Requisitos Funcionais estão completas.

Nos *checklist*s de inspeção de DR são muito comuns as perguntas que verificam se todas as entradas do sistema foram especificadas, se as limitações, condições e restrições

foram identificadas, se os requisitos estão não ambíguos, claros, etc. Tais checagens já foram consideradas no Formato de especificação de Requisitos Funcionais proposto e nas Recomendações de Escrita.

Através do *Checklist* Pré-Inspeção tenta-se minimizar a quantidade de vezes que um DR tem de retornar ao autor que o elaborou devido a erros e falta de informações.

5.3 Considerações Finais

Neste capítulo foram apresentadas as Diretrizes para elaboração de DR com ênfase em Requisitos Funcionais, as quais são compostas de um Formato para Especificação de Requisitos Funcionais, Recomendações de Escrita e *Checklist* Pré-Inspeção.

O conjunto de instruções que compõem as Diretrizes foi baseado nos estudos feitos em alguns DRs aplicando-se neles as técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA e observando-se os defeitos encontrados. Dessa forma, pôde-se definir essas Diretrizes para que um DR não apresentasse esses tipos de defeitos encontrados durante a inspeção.

Com essas Diretrizes pretende-se auxiliar na definição de um DR de forma a melhorar a sua qualidade, contribuindo indiretamente para que uma possível atividade de inspeção seja mais efetiva e de forma a facilitar a aplicação da técnica TUCCA, no que diz respeito à ferramenta que está sendo construída para apoiar a aplicação dessa técnica. Tais instruções proporcionam ao autor de um DR um subsídio no momento da sua preparação, visto que não há nenhum outro documento que possa servir de apoio, pois o DR é o primeiro documento elaborado num processo de desenvolvimento de software.

Considerando-se as propriedades de qualidade que um DR deve apresentar, descrito no Capítulo 3, Seção 3.3, tem-se que as Diretrizes propostas proporciona o alcance de determinadas delas, pois seu conjunto de instruções leva à especificação de Requisitos Funcionais usando um formato mais definido, possui recomendações que auxilia na escrita de sentenças de requisitos de forma a evitar, principalmente, a ambiguidade e, por fim, um *checklist* que faz uma breve verificação destes requisitos. A Tabela 5.2 mostra as propriedades de qualidade que cada item das Diretrizes permite que sejam alcançadas.

Tabela 5-2 Relação entre as Diretrizes propostas e as Propriedades de qualidade do DR.

Diretrizes propostas	Propriedades de Qualidade alcançada
1) Formato de um Requisito Funcional	Completo; Compreensível; Modificável; Organizado; Livre de implementação; Orientado à cliente/usuário
2) Recomendações de Escrita	
Requisitos Múltiplos	Conciso
Acrônimos	Completo
Termos consistentes	-
Requisitos Redundantes	Não redundante
Frases imperativas	-
Marcador "/"	Não ambíguo
Composição de lista	Completo
Termos "e/ou"	Não ambíguo
Palavra "qualquer"	Não ambíguo
Expressão "assim como"	Não ambíguo
Palavras ambíguas	Não ambíguo
Sentenças pouco expressivas	-
Referências explícitas	Completo
3) Checklist Pré-Inspeção	Completo; Correto

Seguindo as Diretrizes propostas, primeiro analisando como deve ser definido um Requisito Funcional por meio do formato, depois observando as Recomendações de Escrita e por fim conferindo os itens do *Checklist* Pré-Inspeção, pretende-se que o DR esteja em um formato mais apropriado para ser revisado. Um dos pontos de interesse desta linha de pesquisa é justamente criar subsídios para essa decisão.

No capítulo seguinte é relatado o estudo realizado para avaliação das Diretrizes propostas, sendo detalhada a aplicação das técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA nos DRs antes e depois do uso das Diretrizes.

Capítulo

6

6. Estudo de Caso

6.1 Considerações Iniciais

As Diretrizes propostas para a elaboração de DR, no que dizem respeito à especificação de Requisitos Funcionais, apresentadas no capítulo anterior, fornecem um Formato que facilita a identificação das informações que devem estar presentes na descrição de um Requisito Funcional, algumas Recomendações a serem consideradas no momento da escrita destas informações e um *Checklist* que pode auxiliar a avaliar se o DR está pronto para uma inspeção.

Para a definição destas Diretrizes, alguns DR foram estudados a fim de se perceber quais as principais deficiências encontradas nelas, ou seja, quais dados relevantes para um bom DR estavam faltando. Assim, as técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA foram aplicadas nestes DRs e, a partir de então, um formato genérico foi estabelecido de tal forma que contivesse as principais informações necessárias e suficientes para o entendimento de um requisito e para a continuação das etapas seguintes à sua elaboração. Conforme essas técnicas iam sendo aplicadas nos DR, as Diretrizes eram consolidadas e alguns ajustes eram feitos.

Terminada a definição das Diretrizes, elas foram utilizadas nos mesmos DRs estudados para a sua composição, resultando em novos DRs. As mesmas técnicas de inspeção foram aplicadas nestes DRs submetidos às instruções das Diretrizes propostas.

A execução das atividades realizadas para concluir este estudo foram feitas pela própria autora deste trabalho, a qual exerceu de forma imparcial para que o resultado não influenciasse na pesquisa.

O estudo de caso realizado com objetivo de avaliar as Diretrizes propostas para elaboração de DR com ênfase em Requisitos Funcionais é apresentado neste capítulo, cuja organização está da seguinte forma: na Seção 6.2 mostra-se a caracterização das atividades de avaliação desenvolvidas; na Seção 6.3 apresenta-se a aplicação das técnicas citadas nos DRs originais, ou seja, do jeito em que elas foram elaboradas inicialmente; na Seção 6.4

apresenta-se a aplicação das mesmas técnicas, porém nos DRs modificados pelas Diretrizes propostas a fim de avaliação; por fim, na Seção 6.5, apresentam-se as Considerações Finais.

6.2 Caracterização do Estudo de Caso

Para o estudo de caso realizado foram utilizados três DRs, os quais se encontram nos Anexos 1, 2 e 3, respectivamente. A escolha destes DRs ocorreu pelo fato de eles representarem diferentes sistemas, cada qual num contexto que aborda diferentes aspectos a serem implementados, com diversos tipos de requisitos solicitados. Isso permitiu que as Diretrizes fossem avaliadas segundo esses três diferentes contextos. Os DRs estudados foram:

- DR para um sistema de gerência de um Hotel: consiste, basicamente, do controle de entrada e saída de hóspedes, reserva de acomodação, impressão de relatórios e diversas consultas.
- DR para um sistema de controle de estacionamento, referenciado neste trabalho como PG (*Parking Garage*): um sistema embutido responsável por controlar a entrada e saída de veículos de um estacionamento, conforme o número de vagas disponíveis.
- DR para um sistema de Rede de Caixa Eletrônico, referenciado neste trabalho como ATM (*Automatic Teller Machine*): também um sistema embutido com a finalidade de automatizar uma rede bancária oferecendo ao cliente do banco acesso à sua conta e permitindo a transação de saque.

No decorrer deste texto utiliza-se a seguinte nomenclatura para fazer referência aos DRs utilizados:

- DRO (DR Original): DR que foi selecionado da literatura, sem qualquer alteração. Os DROs usados no estudo de caso estão apresentados nos Anexos 1, 2 e 3.
- DRT (DR Transformado): DR que foi transformado para o padrão IEEE [1998b] incorporando apenas a estrutura do Formato para especificação de Requisito Funcional proposto nas Diretrizes, de forma a evidenciar as informações faltantes.
- DRM (DR Modificado): após ter sido transformado, o DR foi modificado de forma a suprir informações propostas nas Diretrizes.

Para a obtenção dos DRMs, a etapa de transformação dos DROs foi realizada. Primeiro, os DROs foram inicialmente reestruturados, ou seja, transformados para se adequar ao padrão IEEE [1998b] e se ajustar às Diretrizes propostas, obtendo-se os DRTs. Como vários itens do Formato de um Requisito Funcional proposto ficaram sem serem preenchidas nos DRTs porque nos DROs não havia tais informações, alguns dos itens foram completados,

modificando os DRTs. Assim, os DRMs foram obtidos ao completar certos itens do Formato, mas que não alteravam o fluxo de execução do sistema a ser desenvolvido. As modificações ocorridas foram as designações explícitas dos atores nos itens Agente Fornecedor/Receptor e Executor do Formato, alguns dados de entrada e/ou saída que estavam subentendidas e acréscimo de sentenças ou palavras não claramente explícitas.

Para avaliar a contribuição desta proposta no que diz respeito à qualidade de um DR, as técnicas PBR-Usuário, *Checklist* e TUCCA foram aplicadas primeiro nos DROs e depois nos DRMs, de acordo com a ordem especificada nas Tabelas 6.1 e 6.2, respectivamente. Tais técnicas não foram aplicadas nos DRTs porque os mesmos foram apenas emoldurados, possuindo as mesmas informações que nos DROs. Com isso, o resultado que se obteria nestes dois tipos de documentos seriam os mesmos.

Tabela 6-1 Ordem de aplicação das Técnicas PBR-Usuário, *Checklist* e TUCCA nos DROs.

Técnica aplicada	**DRO**	**Ordem de aplicação**
I) PBR-Usuário	PG	1º.
II) *Checklist*	PG	2º.
III) TUCCA	PG	3º.
I) PBR-Usuário	Hotel	4º.
II) *Checklist*	Hotel	5º.
III) TUCCA	Hotel	6º.
I) PBR-Usuário	ATM	7º.
II) *Checklist*	ATM	8º.
III) TUCCA	ATM	9º.

Tabela 6-2 Ordem de aplicação das Técnicas PBR-Usuário, *Checklist* e TUCCA nos DRMs.

Técnica aplicada	DRM	Ordem de aplicação
I) PBR-Usuário	PG	10º.
II) *Checklist*	PG	11º.
III) TUCCA	PG	12º.
I) PBR-Usuário	Hotel	13º.
II) *Checklist*	Hotel	14º.
III) TUCCA	Hotel	15º.
I) PBR-Usuário	ATM	16º.
II) *Checklist*	ATM	17º.
III) TUCCA	ATM	18º.

A seguir são mostrados com mais detalhes os passos realizados no estudo de caso e as comparações da aplicação das técnicas de inspeção nos DROs e DRMs.

Assim, os tipos de defeitos encontrados e a quantidade deles, tanto nos DROs quanto nos DRMs, após a aplicação de cada técnica de inspeção utilizada neste estudo foram levantados. Os tipos de defeitos abordados referem-se àqueles em que as técnicas de inspeção abrangem, que são: omissão, ambiguidade, informação inconsistente, fato incorreto, informação estranha, miscelânea, questões gerais do DR (maiores detalhes podem ser verificados nos Anexos 4, 5 e 6).

6.3 Aplicação de Técnicas de inspeção nos DROs

Com o objetivo de analisar os principais defeitos encontrados nos DROs e tê-los como subsídio para definir as Diretrizes aqui propostas, aplicaram-se as técnicas PBR-Usuário, *Checklist* e TUCCA nesses documentos.

Para cada DRO que foi estudado, os tipos de defeitos constatados por cada técnica de inspeção que foi aplicada são descritos sob a seguinte estrutura:

I. Aplicação da PBR-Usuário (seria a aplicação da técnica de inspeção PBR-Usuário no DRO; os tipos de defeitos encontrados são especificados).

II. Aplicação do *Checklist* (seria a aplicação do *checklist* no DRO; os tipos de defeitos identificados são relatados).

III. Aplicação da TUCCA (seria a aplicação da técnica TUCCA no DRO; os tipos de defeitos encontrados também são apresentados).

6.3.1 Inspeção no DRO do PG

I. <u>Aplicação da PBR-Usuário no DRO do PG</u>

Os defeitos encontrados com essa técnica são listados abaixo.

- <u>Omissão</u>:

1) Alguns Requisitos Funcionais não continham informações suficientes para especificar o caso de uso, tornando-os incompletos. Os defeitos assinalados foram:

- No Requisito Funcional 14 não foi especificado o que fazer quando na entrada do estacionamento o ingresso mensal é inválido.
- Nos Requisitos Funcionais 20 e 21 não foram especificados o que fazer quando na saída do estacionamento os ingressos são inválidos.
- No Requisito Funcional 22 não havia informações sobre os eventos que aconteciam quando vários carros deixavam o estacionamento ao mesmo tempo.

2) Averiguou-se que havia Requisitos Funcionais que alteravam os valores de algumas variáveis, mas não atualizavam as que estavam relacionadas. Os defeitos assinalados foram:

- No Requisito Funcional 8, no item Saída, não é descrito qual o novo valor da variável **a**.
- No Requisito Funcional 10 faltou atualizar o valor da variável **a** ao alterar o valor da variável **r**.
- No Requisito Funcional 11 faltou atualizar o valor da variável **a** ao alterar o valor da variável **k**.
- No Requisito Funcional 13, no item Saída, não é descrito qual o novo valor da variável **a**.
- No Requisito Funcional 16 faltou decrementar ‘a’ no item Processamento e incrementar ‘o’ no item Descrição.

3) O documento apresenta Requisitos Funcionais com o item Processamento incompleto. Os defeitos assinalados foram:

- No Requisito Funcional 10 faltou verificar se a variável **r** <= 0.4***k** ao fixar o seu novo valor.

- No Requisito Funcional 11 faltou verificar se a variável **k** <= 1000 ao fixar o seu novo valor.

4) No Documento não fica claro como o sistema de controle de estacionamento (PGCS) interage com dispositivos de hardware. O defeito assinalado foi:

- Interação com a máquina de ingresso, os leitores de cartão, os portões, os sensores de passagem e a unidade de controle. Tem-se apenas que o PGCS receberá e enviará sinais para estes dispositivos.

- Ambigüidade:

1) Havia Requisito Funcional com termo ambíguo. O defeito assinalado foi:

- Requisito Funcional 17: - Descrição: "será dado no máximo um ingresso de entrada a cada motorista".

  O termo "máximo" causa ambigüidade, pois significa que pode não dar nenhum ingresso ao motorista.

- Informação Inconsistente:

1) No documento havia várias palavras que semanticamente apresentavam o mesmo significado, mas estavam designadas por termos diferentes, gerando defeitos de inconsistência de informação. O defeito assinalado foi:

- Botão e tecla; sensor de passagem, loop de indução e sensor de indução; leitor de cartão e leitor de ingresso; painel de controle e unidade de controle;

2) Havia contradições entre o que estava descrito na Descrição e no Processamento. Os defeitos assinalados foram:

- No Requisito Funcional 16, no item Descrição era descrito para diminuir o número de vagas disponíveis, que seria a variável **a**, mas no item Processamento era descrito para incrementar o valor de **o** por 1.
- No Requisito Funcional 8, no item Descrição era descrito para aumentar de um o valor da variável **r**, mas no item Processamento era descrito para decrementar o valor de **r** por 1.

- Fato Incorreto:

1) Havia Requisito de Desempenho considerado como tal, mas que não era um Requisito de Desempenho. Os defeitos assinalados foram:

- Requisito de Desempenho 4: - Descrição: "só um carro deve atravessar o portão a cada vez".
- Requisito de Desempenho 7: - Descrição: "para cada carro que entra no estacionamento há uma vaga disponível".

2) Havia Requisito Funcional considerado como tal, mas que não era um Requisito Funcional. Os defeitos assinalados foram:

- Requisito Funcional 3: - Descrição: "O valor default para **k** é 1000".
- Requisito Funcional 4: - Descrição: "**k** é dividido em **r** vagas reservadas e a vagas não reservadas".
- Requisito Funcional 7: - Descrição: "ingressos mensais são válidos durante 30 dias".

- Miscelânea:

1) Não estava claro os atores que interagiam com o sistema, pois não estavam explícitos. Além disso, por se tratar de um documento para um sistema embutido, o qual não se tem experiência, a identificação dos atores foi dificultada.

2) Havia Requisitos Funcionais que eram melhores explicados por outros requisitos e, portanto deveriam ser desconsiderados. Os defeitos assinalados foram:

- Requisito Funcional 1: - Descrição: "O PGCS deve controlar as entradas e saídas de um estacionamento".
- Requisito Funcional 5: - Descrição: "O PGCS deve apoiar **n** entradas e **m** saídas, eventualmente simultâneas".

II. Aplicação do *Checklist* no DRO do PG

- Questões Gerais:

1) Por causa do fato de o *Checklist* consistir de perguntas que verificam o documento como um todo, onde são checados alguns pontos, não foram percebidas a falta de determinadas informações que seriam necessárias para o MCU. Suas questões foram sendo respondidas com base no entendimento do DR, não prendendo a atenção a detalhes importantes para a modelagem.

2) Verificou-se que a forma apresentada por alguns Requisitos Funcionais não era clara. Alguns eram descritos apenas pelo item Descrição, com grande quantidade de

informações, confundindo ou dificultando o entendimento do requisito. Outros apresentavam os itens Descrição, Entrada, Processamento e Saída.

3) Verificou-se que o documento não especificava o software e o hardware necessários.

- Omissão:

1) No quesito desempenho, não foi especificado o volume de entrada e saída de carros simultaneamente.

2) Nem todos os defeitos do tipo omissão que foram encontrados na inspeção PBR-Usuário foram descobertos pelo *Checklist*, pois alguns são mais percebíveis durante a modelagem.

- Ambigüidade, Informação Inconsistente e Fato Incorreto:

1) Os mesmos defeitos do tipo ambigüidade, fato incorreto e informação inconsistente que foram encontrados na inspeção PBR-Usuário também foram identificados pelo *Checklist*.

III. Aplicação da TUCCA no DRO do PG

Leitura AGRT

- Omissão:

1) Alguns Requisitos Funcionais descritos na seção "2.3 Funções do Produto" não foram especificados na seção "3.1 Requisitos Funcionais". Os defeitos assinalados foram:

  - Controlar painel de status.
  - Controlar os portões.
  - Controlar as máquinas de ingresso.
  - Controlar os leitores de ingresso.

2) Havia funcionalidades na Lista de Objetivos Não Associados que foram desconsideradas pela técnica TUCCA, pois a mesma não trata os casos em que o ator é o próprio Sistema. Porém eram funcionalidades que deveriam ser executadas. Os defeitos assinalados foram:

  - A funcionalidade "Garantir que não mais que k carros estejam no estacionamento" tem associado o ator PGCS.

- A funcionalidade "Liberar vagas reservadas quando ingressos mensais expiram" tem associado o ator Sistema.

- Ambigüidade:

1) Os mesmos defeitos do tipo ambigüidade que foram encontrados na inspeção PBR-Usuário também foram identificados aqui.

- Informação Inconsistente:

1) Os mesmos defeitos do tipo informação inconsistente que foram encontrados na inspeção PBR-Usuário também foram identificados aqui.

2) Funcionalidades idênticas, porém com sintaxes diferentes foram encontradas na coluna "Objetivo" do FAO. Os defeitos assinalados foram:

- "Fixar um valor novo de **r**" (Requisito Funcional 10) e "Entrar o número de vagas reservadas" (Seção 2.3 Funções do Produto).
- "Escrever o número total de vagas" (Requisito Funcional 11) e "Entrar o número total de vagas" (Seção 2.3 Funções do Produto).

- Fato Incorreto:

1) O defeito do tipo fato incorreto 2 que foi encontrado na inspeção PBR-Usuário também foi identificado aqui.

- Miscelânea:

1) O mesmo defeito do tipo miscelânea 1 que foi encontrado na inspeção PBR-Usuário também foi identificado aqui.

2) O item 2 das Questões Gerais do *Checklist* também foi verificado aqui.

3) Percebeu-se que havia Requisito Funcional que apresentava determinada descrição que confundia a associação do ator com a funcionalidade. Os defeitos assinalados foram:

- Requisito Funcional 10: - Descrição: "a unidade de controle pode fixar um valor novo de **r**".

  Da forma que foi expressa a sentença dá a entender que o ator é a unidade de controle, quando o correto seria ator funcionário.

Leitura UCRT

- Omissão:

1) Todos os defeitos do tipo omissão que foram encontrados na inspeção PBR-Usuário também foram identificados aqui.

- Fato Incorreto:

1) O mesmo defeito do tipo fato incorreto 1 que foi encontrado na inspeção PBR-Usuário também foi identificado aqui.

- Miscelânea:

1) O mesmo defeito do tipo miscelânea 2 que foi encontrado na inspeção PBR-Usuário também foi identificado aqui.

6.3.2 Inspeção no DRO do Hotel

I. Aplicação da PBR-Usuário no DRO do Hotel

- Omissão:

1) Diversas funcionalidades descritas no documento não apresentaram informações suficientes que permitissem uma adequada especificação de caso de uso. Os defeitos assinalados foram:

  - Os requisitos que solicitam a inclusão de dados não especificam o que fazer quando um determinado dado já existe ou quando certa informação está incompleta.
  - Os requisitos que solicitam a alteração de dados não especificam o que fazer quando um determinado dado não foi encontrado para ser alterado.
  - Os requisitos que solicitam a remoção de dados não especificam o que fazer quando um determinado dado não foi encontrado para ser removido.
  - O Requisito Funcional 10, que descreve as opções de pagamento, apresenta-se vago, sem maiores detalhes da funcionalidade.
  - O Requisito Funcional 11, que permite a quitação de uma fatura, não apresenta detalhes de quantas parcelas o pagamento pode ser efetuado.

2) Verificou-se a ausência de funcionalidades. Os defeitos assinalados foram:

  - Os requisitos para consulta de dados não foram especificados, o que é comum no caso de sistemas como este, que possui diversos cadastros.

- Não foi definido que tipo de acesso cada usuário do sistema teria.

3) Verificou-se a ausência de informações de entrada de dados. O defeito assinalado foi:

- Os Requisitos Funcionais de alteração e remoção não informavam a partir de qual informação de entrada o dado seria localizado para possibilitar a alteração e remoção, respectivamente.
- Os Requisitos Funcionais de alteração não indicavam quais dados poderiam ser alterados.

4) No caso de inclusão de dados, os atributos informados a serem armazenados não especificavam qual era o seu tipo do dado e o limite de seu tamanho/comprimento (Por exemplo, o atributo Nome poderia ser do tipo apenas caracter e de tamanho 40).

5) Também não foi especificada nenhuma mensagem de alerta, erro ou sucesso de operação ao usuário do sistema.

- Miscelânea:

1) Verificou-se a não clareza de funcionalidades ou sentenças. Os defeitos assinalados foram:

- No documento não ficou clara a interação do hóspede no caso da requisição de sua saída do hotel a partir da televisão de seu apartamento.
- Não ficou claro o que seriam quiosques especiais.

II. Aplicação do *Checklist* no DRO do Hotel

- Questões gerais:

1) Nem todos os requisitos foram especificados de forma clara que não gerasse dúvida.

2) Os requisitos não foram especificados em termos de entrada, processamento e saída, sendo suas sentenças descritas num único parágrafo.

3) No documento não foi fornecida uma visão geral das formas de operação do sistema quando diferentes usuários utilizam o sistema.

- Omissão:

1) Os defeitos do tipo omissão 2 e 4 que foram encontrados pela PBR-Usuário também foram identificados pelo *Checklist*. Os restantes não o foram, pois seriam mais visíveis no caso de modelagem.

- Ambigüidade:

1) Verificou-se a ocorrência do termo automaticamente, o que gera ambigüidade. Os defeitos assinalados foram:

  - No Requisito Funcional 7 é descrito que o sistema recupera automaticamente os dados da reserva.
  - No Requisito Funcional 9 é descrito que o sistema deve automaticamente totalizar os gastos de consumo do hóspede.

- Miscelânea:

1) Todos os defeitos do tipo miscelânea encontrados pela PBR-Usuário foram identificados aqui.

III. Aplicação da TUCCA no DRO do Hotel

Leitura AGRT

- Omissão:

1) Somente o defeito do tipo omissão 2 encontrado pela PBR-Usuário foi detectado nesta Leitura.

- Ambigüidade:

1) A forma com que os requisitos foram descritos pode induzir ao erro ou causar confusão no momento de associar o ator à funcionalidade, uma vez que todos eles iniciaram-se pela expressão “O sistema deve”. É nesta Leitura que o ator é identificado através de marcação.

- Miscelânea:

1) Foi observado que o documento não apresentava a seção "Funções do Produto" ou, a qual é necessária para a aplicação da técnica TUCCA, pois esta exige que determinadas seções do padrão IEEE [1998b] estejam presentes no documento.

2) Verificou-se que funcionalidades diferentes foram descritas sob uma única referência. O defeito assinalado foi:

- "1. O sistema deve permitir a inclusão, alteração e remoção de hóspedes do hotel."

  Este Requisito Funcional apresenta três requisitos diferentes, que seriam incluir, alterar e remover referenciados pelo número 1.

Leitura UCRT

- Omissão:

1) Exceto o defeito do tipo omissão 2 encontrado pela PBR-Usuário, anteriomente detectada na Leitura AGRT, todos os restantes foram encontrados aqui, pois são pertinentes a especificação de caso de uso.

- Miscelânea:

1) Todos os defeitos do tipo miscelânea encontrados pela PBR-Usuário foram identificados aqui.

6.3.3 Inspeção no DRO do ATM

O DRO do ATM encontrava-se na língua inglesa e para facilitar o seu uso neste trabalho ele foi traduzido para o português. Feito isso, as técnicas foram aplicadas.

I. Aplicação da PBR-Usuário no DRO do ATM

- Omissão:

1) O documento apresenta Requisito Funcional com o item Processamento incompleto. O defeito assinalado foi:

- No Requisito Funcional 11, Seção 3.1.1, faltou verificar a variável **n**, que corresponde ao dinheiro mínimo disponível no ATM para permitir uma transação.

- Informação Inconsistente:

1) Foram identificados termos diferentes para designar-se à mesma coisa. O defeito assinalado foi:

  - Caixas Eletrônicos e ATM, número do cartão e número serial, cliente e usuário, banco e computador do banco, ejetar cartão e devolver cartão.

2) Verificou-se que havia contradições nas descrições dos requisitos especificados. Os defeitos assinalados foram:

  - No Requisito Funcional 5, Seção 3.1.1, no item Descrição foi descrito para ler o número serial e o código do banco, mas no item Processamento apenas o número serial foi lido. E ainda, o código do banco é verificado no Requisito Funcional 7, Seção 3.1.1.
  - No Requisito Funcional 7, Seção 3.1.1, no item Descrição foi descrito que o ATM verifica o código do banco, mas no item Processamento foi descrito número serial.

3) Verificou-se que os dados trocados entre dois requisitos eram contraditórios. Os defeitos assinalados foram:

  - O Requisito Funcional 5, Seção 3.1.2, recebe como entrada um cartão bancário válido e uma senha válida. Porém, o Requisito Funcional 7, Seção 3.1.1, verifica código do banco e senha, mas envia número serial e senha.
  - No Requisito Funcional 7, Seção 3.1.2, em seu item Saída foi descrito que o computador do banco envia a mensagem "transação bem sucedida" ou "transação falhou". Mas, de acordo com os Requisitos Funcionais 14 e 16, ambos da Seção 3.1.1, foi descrito que o ATM recebe a mensagem "transação com êxito" ou "transação sem êxito".
  - No Requisito Funcional 1, Seção 3.1.2, no item Descrição foi descrito que o computador do banco checa o código do banco, mas no item Entrada foi solicitado número serial e senha.

- Fato Incorreto:

1) Verificou-se que a descrição da funcionalidade estava incorreta. O defeito assinalado foi:

- No Requisito Funcional 5, Seção 3.1.1, no item Saída foi descrito para iniciar diálogo de autorização, mas apenas depois de conectado o número serial este diálogo deveria ser iniciado.

2) Verificou-se que variáveis utilizadas no documento foram designadas incorretamente. Os defeitos assinalados foram:

- O Requisito Funcional 3, Seção 3.1.1, fez referência à variável **t** quando o correto seria **n**.
- O Requisito Funcional 7, Seção 3.1.2, referiu-se à variável **m** como sendo a quantidade a sacar, mas a variável **m** serve de comparação para a quantidade a ser sacada.

3) Havia Requisito Funcional considerado como tal, mas que não era um Requisito Funcional. O defeito assinalado foi:

- Requisito Funcional 10 (Seção 3.1.2): - Descrição: "O banco oferece segurança somente para seus próprios computadores e seus próprios softwares".

4) Havia Requisito de Desempenho considerado como tal, mas que não era um Requisito de Desempenho. Os defeitos assinalados foram:

- Requisito de Desempenho 3: - Descrição: "O ATM libera dinheiro se e somente se o saque de uma conta é processado e aceito pelo banco".
- Requisito de Desempenho 4: - Descrição: "Cada banco pode processar transações de vários ATM´s ao mesmo tempo".

- <u>Miscelânea</u>:

1) No documento não ficou claro alguns requisitos especificados. O defeito assinalado foi:

- No Requisito Funcional 7, Seção 3.1.2, em seu item Processamento não ficou claro a atualização da variável **k**.
- No Requisito Funcional 6, Seção 3.1.1, não foi possível compreender o item Saída de acordo com sua descrição.
- No Requisito Funcional 13, Seção 3.1.1, no item Entrada não foi especificada claramente o que seria "seqüência de saque inicial com sucesso". Somente lendo os requisitos anteriores deduz-se que a palavra "com sucesso" significa quantidade a sacar menor que a variável **m**.

II. Aplicação do *Checklist* no DRO do ATM

- Questões Gerais:

1) A maioria dos Requisitos Funcionais foi declarada em termos de entrada, processamento e saída. Apenas o Requisito Funcional 2, Seção 3.1.1, foi descrito por uma descrição.

- Ambigüidade:

1) Identificou-se o uso de termo ambíguo. O defeito assinalado foi:

  - Na Seção 3.4.2 do documento é descrito que a rede ATM deveria oferecer o máximo de segurança.
    A palavra "máximo" gera ambigüidade.

- Informação Inconsistente:

1) Os mesmos defeitos do tipo informação inconsistente encontrados pela PBR-Usuário foram encontrados pelo *Checklist*.

- Fato Incorreto:

1) Os defeitos do tipo fato incorreto 2, 3 e 4 encontrados pela PBR-Usuário foram encontrados pelo *Checklist*.

- Miscelânea:

1) Os mesmos defeitos do tipo miscelânea encontrados pela PBR-Usuário foram encontrados pelo *Checklist*.

III. Aplicação da TUCCA no DRO do ATM

Leitura AGRT

- Omissão:

1) Alguns Requisitos Funcionais descritos na seção "2.2 Funções do Produto" não foram especificados na seção "3.1 Requisitos Funcionais". Os defeitos assinalados foram:

  - Fornecer segurança.

- Manusear corretamente acessos concorrentes.

2) Havia funcionalidade na Lista de Objetivos Não Associados que foi desconsiderada pela técnica TUCCA, pois a mesma não trata os casos em que o ator é o próprio Sistema. Porém era funcionalidade que deveria ser executada. O defeito assinalado foi:

- A funcionalidade "Mostrar tela inicial se nenhum cartão bancário estiver inserido no ATM" tem associado o ator Sistema.

3) Identificou-se requisito que não havia na sua descrição verbo de ação para ser marcado e colocado na coluna Objetivo do FAO. O defeito assinalado foi:

- Requisito Funcional 8 (Seção 3.1.1): - Descrição: "Diferentes respostas negativas do computador do banco para diálogo de autorização".

- Informação Inconsistente:

1) Os mesmos defeitos do tipo informação inconsistente 1 e 2 encontrados pela PBR-Usuário foram identificados aqui.

- Fato Incorreto:

1) O defeito do tipo informação inconsistente 3 encontrado pela PBR-Usuário foi identificado aqui.

- Miscelânea:

1) Percebeu-se que havia Requisito Funcional que apresentava determinada descrição que confundia a associação do ator com a funcionalidade. O defeito assinalado foi:

- Requisito Funcional 4 (Seção 3.1.1): - Descrição: "O ATM tem que checar se o cartão inserido é um cartão bancário válido".

  A funcionalidade descrita é checar validade do cartão bancário, então o ator correspondente seria cliente. Da forma que foi expressa a sentença dá a entender que o ator é o ATM.

2) Ao preencher a coluna "Referência" do FAO, onde é designada a localização em que se encontra o requisito ou o número que o identifica, notou-se que os requisitos não tinham uma identificação única, o que confundia ao referenciá-los.

3) Havia requisitos que agrupavam diversas informações no item Descrição, sendo que eles deveriam ser colocados nos itens apropriados, como Entrada, Processamento ou Saída. Os defeitos assinalados foram:

- Requisito Funcional 3 (Seção 3.1.1): - Descrição: "Se o ATM não tem mais dinheiro, nenhum cartão deveria ser aceito. Uma mensagem de erro é mostrada".
  A mensagem deveria ter sido descrita no item Saída.
- Requisito Funcional 10 (Seção 3.1.1): - Descrição: "Se o cartão foi inserido mais do que três vezes numa linha de qualquer ATM e a senha era errada a cada vez, o cartão é retido pelo ATM. Uma mensagem será mostrada que o cliente deveria chamar o banco".
  A mensagem deveria ter sido descrita no item Saída.

Leitura UCRT

- Omissão:

1) Os mesmos defeitos do tipo omissão encontrados pela PBR-Usuário foram identificados aqui.

- Informação Inconsistente:

1) O mesmo defeito do tipo informação inconsistente 3 encontrado pela PBR-Usuário foi identificado aqui.

- Fato Incorreto:

1) Os defeitos do tipo informação inconsistente 1, 2 e 4 encontrados pela PBR-Usuário foram identificados aqui.

- Miscelânea:

1) Os mesmos defeitos do tipo miscelânea encontrados pela PBR-Usuário foram encontrados aqui.

Terminado o estudo inicial feito nos DROs, relatado acima, através de inspeção com PBR-Usuário, *Checklist* e TUCCA, diversos defeitos foram coletados e anotados. Percebendo as suas falhas, as Diretrizes para elaboração de DR foram preparadas e depois de estabelecidas elas foram usadas nos DRs do PG, Hotel e ATM. A explicação detalhada das

mudanças que ocorreram nos DRs após o uso das Diretrizes propostas e como isso influenciou na aplicação das mesmas técnicas de inspeção estão relatadas na seção seguinte.

6.4 Modificações dos DROs do PG, do Hotel e do ATM

Após estudados os DROs, os mesmos foram transformados e modificados através das Diretrizes propostas para então serem novamente inspecionados através das mesmas técnicas aplicadas anteriormente. O propósito disso seria a avaliação da eficiência das Diretrizes para elaboração de DR. Assim, o Formato para especificação de um Requisito Funcional, as Recomendações de Escrita e o *Checklist* Pré-Inspeção foram aplicados nos DROs.

As mudanças ocorridas nos DROs, o que posteriormente foram designados como DRMs, são apresentadas nesta seção.

6.4.1 Modificações no DRO do PG

Para que os Requisitos Funcionais do DRO do PG se apresentassem no Formato de especificação de Requisitos Funcionais proposto, diversos ajustes tiveram de ser realizados. A adequação a este Formato foi a principal modificação realizada, pois a apresentação dos Requisitos Funcionais teve de ser reformulada.

Para facilitar a explicação das mudanças que ocorreram nos requisitos após o uso do Formato proposto alguns exemplos serão dados. Para cada exemplo, primeiro será mostrado como o Requisito Funcional estava apresentado no DRO e depois como o mesmo Requisito Funcional ficou após a sua transformação com o uso do Formato.

A identificação do Requisito Funcional no DRO pode não corresponder com a identificação do mesmo requisito no DRM, pois ao tentar transformar determinados requisitos usando as Diretrizes propostas verificou-se que os mesmos não eram Requisitos Funcionais, sendo eles desconsiderados.

Exemplo 1:

Requisito Funcional 13

- Descrição: o painel de status deve representar o estado de ocupação das vagas não reservada. Deve exibir ‘vagas’ se houver uma vaga não reservada disponível naquele momento. Deve exibir ‘lotado’, se não há nenhuma vaga não reservada disponível naquele momento.

Figura 6.1 Requisito Funcional 13 do DRO do PG.

Requisito Funcional 8.
Descrição: Mostrar o estado atual de ocupação das vagas não reservadas.
Agente Fornecedor/Receptor: Painel de status
Agente Executor: Sistema
Entrada:

Campo	Tipo	Tamanho	Descrição
Sinal de que houve mudança de estado "há vagas" para "lotado", ou "lotado" para "há vagas"	Numérico	1 inteiro	0 = estado "há vagas" para "lotado" 1 = estado "lotado" para "há vagas"

Processamento: Enviar uma notificação para o painel de status para que este exiba determinada mensagem se houver uma vaga não reservada disponível naquele momento ou se não há nenhuma vaga não reservada disponível naquele momento.
Condição/Restrição: Nenhuma.
Saída: Exibir 'Há vagas'. Senão, exibir 'Lotado'.

Figura 6.2 Requisito Funcional 8 do DRM do PG.

Como pode ser visto, o requisito da Figura 6.1 apresenta-se apenas com o item Descrição. Ao usar o Formato, percebeu-se que determinados itens ficariam sem serem preenchidos, pois o requisito original não deixa claro qual a informação de entrada necessária para a sua realização e nem o que o sistema deve fazer para permitir que o painel de status mostre o estado de ocupação das vagas não reservadas. Sendo assim, algumas informações foram acrescentadas e outras colocadas no item adequado.

É visível que o Formato proposto detalha o Requisito Funcional de forma a incluir informações essenciais que melhoram o seu entendimento. Cada item foi especificado da seguinte forma: a Descrição foi alterada para começar por um verbo que indique 'o quê' o requisito deve fazer. Conforme é recomendado no formato, uma descrição sem detalhes é oferecida por tal item, de forma que fique mais clara a funcionalidade requerida. Foi aproveitada parte essencial da descrição original. Os itens Agente Fornecedor/Receptor e Agente Executor foram completados após a compreensão da interação que ocorre no sistema, pois não estavam explícitos. O item Entrada foi preenchido com a informação que diz qual a entrada recebida para que o requisito possa ser concluído. Nesse caso, por ser sistema embutido, percebe-se que apenas quando o painel de status receber um sinal de mudança de estado é que o requisito será executado. Uma tabela que especifica os dados de entrada é usada. O Processamento iniciou-se por um verbo e detalha os passos necessários para a execução do requisito. Como é o painel de status quem mostra a mensagem se há vagas ou

não, é para ele que o sistema deve notificar a mudança de estado. Para este requisito não houve alguma condição ou restrição que delimitava a sua realização. E no item Saída foi colocada parte da informação contida na descrição do requisito original, pois seria a resposta dada pelo requisito.

Exemplo 2:

Requisito Funcional 14

- Descrição: todo motorista com um ingresso mensal deverá inserir o ingresso no leitor de ingresso. Se este for válido, o portão abrirá. Um ingresso reservado válido sempre permitirá entrada.
- Entrada: motorista insere ingresso mensal.
- Processamento: o número de ingresso será analisado pelo sistema para determinar se a data de sua compra foi há 30 dias ou menos antes da data atual. Em caso afirmativo, o portão abrirá.
- Saídas: o portão é aberto.

Figura 6.3 Requisito Funcional 14 do DRO do PG.

Requisito Funcional 9
Descrição: Permitir a entrada de motoristas com ingresso mensal se este for válido.
Agente Fornecedor/Receptor: Motorista
Agente Executor: Motorista
Entrada:

Campo	Tipo	Tamanho	Descrição
Número do ingresso mensal	Numérico	8 inteiros	

Processamento: Analisar o número de ingresso para verificar se a data da sua compra foi há 30 dias ou menos da data atual. Em caso afirmativo, o ingresso é válido, então o portão deve ser aberto. Senão, nada acontece.
Condição/Restrição: Se o ingresso mensal for válido.
Saída: Abrir portão. Senão, nada acontece.

Figura 6.4 Requisito Funcional 9 do DRM do PG.

O requisito funcional 14, mostrado na Figura 6.3, é descrito com mais detalhes através dos termos Descrição, Entrada, Processamento e Saída, embora no mesmo documento que o Requisito Funcional 13 do Exemplo 1, apresentado pela Figura 6.1. Ainda assim, modificações foram feitas de acordo com as orientações definidas no Formato para especificação de Requisitos Funcionais.

Ao ler a Descrição do requisito original pode-se notar a dificuldade de compreendê-la devido às informações misturadas, ou seja, dados de entrada e sua explicação, condições e afirmações. Não está claro qual a finalidade do requisito. Para compor o item Descrição do requisito modificado de forma simples, direta e começando por verbo, teve que se ler e entender o que estava sendo solicitado por aquele requisito original. Verificou-se que o objetivo principal do requisito era permitir que motoristas com ingresso mensal entrassem no estacionamento se o ingresso fosse válido. Assim, a funcionalidade foi resumidamente explícita numa única frase. A parte em que diz "todo motorista com um ingresso mensal deverá inserir o ingresso no leitor de ingresso" foi desconsiderado na nova Descrição, pois trata-se de uma explicação de como o motorista deve agir. A sentença "Se este for válido, o portão abrirá" também foi retirada da Descrição, pois expressa uma condição e processamento. E a última sentença "Um ingresso reservado válido sempre permitirá entrada", expressando uma afirmação, foi então reformulada para começar por um verbo e exprimir o que realmente o requisito deve fazer.

Para a realização dessa funcionalidade, identificou-se que o motorista era quem interagia com o sistema ao inserir o número do ingresso, sendo ele o Agente Fornecedor/Receptor e Agente Executor, pois além de informar o número do ingresso também operava diretamente com o sistema. A informação de entrada, ao invés de se apresentar em forma de sentença, foi especificada pela tabela que caracteriza cada dado de entrada necessário para a conclusão do requisito. No item Processamento foi descrito o que o sistema deve fazer para permitir a realização do requisito, ou seja, daquilo que foi mencionado na Descrição. Também foram mostradas tanto as ações positivas quanto negativas após a verificação da data da compra do ingresso. Como a ação negativa não estava definida no requisito original, foi colocada apenas a sentença 'Senão, nada acontece', pois assim não modificaria as funções definidas para o sistema. Caso fosse exibida uma mensagem ao motorista indicando que o ingresso é inválido, o documento deveria informar que o leitor de ingresso apresentava um visor e isso não foi mencionado em lugar algum. Apenas para conformar com o formato proposto e explicar de forma mais didática é que tal ação negativa foi descrita, mas o correto seria verificar com o cliente do sistema sendo desenvolvido. Para o item Condição/Restrição, a presença do termo 'se' na Descrição facilitou a sua identificação. E para o item Saída foi descrita a resposta esperada com a realização do requisito.

Terminada a transformação nos Requisitos Funcionais, o DR foi analisado para verificar a sua adequação às Recomendações de Escrita. Diversos termos inconsistentes foram eliminados.

E para finalizar, o *Checklist* Pré-Inspeção foi checado no documento, o que possibilita uma verificação rápida de determinadas informações.

Aplicando-se as Diretrizes propostas no DRM do PG pôde-se perceber que diversos defeitos foram sendo percebidos antes mesmo da aplicação de algum tipo de inspeção ser iniciado. Isso porque as Diretrizes são compostas de itens que permitem que as sentenças usadas para especificar um requisito sejam descritas de forma mais completa, objetiva e padronizada.

6.4.2 Modificações no DRO do Hotel

O DRO do Hotel, dentre os estudados, foi o documento que menos continha a maioria das informações solicitadas para preencher os itens do Formato proposto. As requisições a serem desenvolvidas pelo sistema estavam sucintamente descritas. Embora as informações estivessem organizadas, o documento não apresentava seções recomendadas pela técnica TUCCA, portanto o mesmo foi adequado ao padrão de especificação IEEE [1998b] para depois serem aplicadas as Diretrizes propostas.

Ao transformar a apresentação dos Requisitos Funcionais para o Formato proposto, muito dos itens ficaram em branco, mostrando que o DR estava incompleto de dados básicos. Tais ausências de dados demonstraram que os requisitos não tinham todas as informações essenciais para apoiar as próximas etapas do processo de desenvolvimento de software. Sendo assim, os itens não preenchidos foram completados para que o Requisito Funcional apresentasse informações fundamentais.

A seguir são mostrados exemplos de como ficaram os Requisitos Funcionais após a modificação proporcionada pelas Diretrizes.

Exemplo 1:

1. O sistema deve permitir a inclusão, alteração e remoção de *hóspedes do hotel*, contendo os seguintes atributos: nome, endereço, cidade onde mora, estado, país, telefone, email, documento de identificação (RG ou CPF para brasileiros e passaporte para estrangeiros), data de nascimento e nome dos pais.

Figura 6.5 Requisito Funcional 1 do DRO do Hotel.

Requisito Funcional 1.
Descrição: Incluir o cadastro de hóspedes.
Agente Fornecedor/Receptor: Hóspede
Agente Executor: Funcionário
Entrada: O cadastro de hóspedes deve conter os seguintes atributos:

Campo	Tipo	Tamanho	Descrição
Nome	Alfabético	40 caracteres	
Endereço	Alfanumérico	50 caracteres	
Cidade	Alfabético	20 caracteres	
Estado	Alfabético	2 caracteres	Apresentar para selecionar Estado.
País	Alfabético	20 caracteres	
Telefone	Alfanumérico	10 caracteres	Formato: (xx) xxxx-xxxx.
Email	Alfanumérico	30 caracteres	
Documento de identificação	Alfanumérico	RG: 9 caracteres CPF: 11 caracteres Passaporte: 15 caracteres	RG ou CPF para brasileiros e passaporte para estrangeiros
Data de Nascimento	Alfanumérico	10 caracteres	Formato: dd/mm/aaaa
Nome dos pais	Alfabético	40 caracteres	

Processamento: Verificar se o cadastro de hóspedes existe. Em caso afirmativo, o cadastro não deve ser incluído e uma mensagem de aviso deve ser exibida indicando que o hóspede já existe. Senão, incluir o cadastro e exibir uma mensagem de sucesso indicando que a inclusão foi efetuada.
Condição/Restrição: Nenhuma.
Saída: Armazenar dados do hóspede e exibir mensagem de sucesso. Senão, exibir mensagem de aviso.

Figura 6.6 Requisito Funcional 1 do DRM do Hotel.

De acordo com as Recomendações de Escrita, requisitos múltiplos devem ser evitados. Sendo assim, as três funcionalidades apresentadas na Figura 6.5 (inclusão, alteração e remoção) foram desmembradas e identificadas unicamente. Assim, a Figura 6.6 mostrou apenas o Requisito Funcional de inclusão.

Adequando-se à exigência do item Descrição, que é começar sempre por um verbo principal que expresse a idéia do que o requisito deve fazer, a sua sentença no Requisito Funcional modificado foi descrita. Assim, a partir da descrição anterior, "O sistema deve permitir a inclusão de hóspedes do hotel", verificou-se que o verbo de ação principal seria 'inclusão'. A palavra 'cadastro', que estava subentendida, foi expressa. Desta forma, gerou-se a sentença 'Incluir o cadastro de hóspedes'.

O Agente Fornecedor/Receptor associado a este requisito foi o hóspede, uma vez que é ele quem informa os dados para a realização da funcionalidade. Já o Agente Executor foi o funcionário, pois é ele quem operacionaliza a execução do requisito.

Os atributos do cadastro de hóspedes foram melhores definidos através da tabela que caracteriza cada informação de entrada. As colunas Tipo, Tamanho e Descrição foram completadas a fim de exemplo, uma vez que no requisito original não havia tal informação.

Com o objetivo de mostrar a nova forma com que os requisitos deveriam se apresentar, as informações faltantes foram preenchidas, mas o correto seria a confirmação com o cliente do sistema sendo desenvolvido.

No item Processamento foi definido aquilo que o sistema deve processar para incluir o cadastro de hóspede. Esta informação foi incluída com base na experiência e estudos em sistemas de informação. Para este requisito não foi estabelecida nenhuma condição ou restrição que afetasse a sua realização. E a saída retornada por este requisito foi descrita de acordo com aquilo que se espera ao incluir dados, que é a concretização do que foi solicitado (dados armazenados) e uma mensagem para quem realizou a execução do requisito, como forma de esclarecer se houve ou não a inclusão.

Exemplo 2:

> 6. O sistema deve permitir a *reserva de acomodação*. Cada reserva possui os seguintes atributos: data e hora de chegada do hóspede, data e hora de saída do hóspede, identificação do hóspede principal (previamente cadastrado), tipo de acomodação desejada, nomes e idades dos *acompanhantes*, valor da *diária*, taxa de multa a ser cobrada em caso de desistência de última hora (a menos de 12 horas do início previsto de entrada), os dados do cartão de crédito do hóspede e desconto concedido (opcional). A reserva somente deve ser concretizada se houver vagas suficientes para atendê-la. Caso contrário deverá ser mostrada uma mensagem alertando que não há disponibilidade de acomodações para o período indicado. A remoção de reserva somente é permitida sem maiores encargos até 12 horas antes do início previsto para *estadia no hotel*. Após esse período, a remoção da reserva deve alertar o *funcionário* do hotel de que deve ser cobrada a taxa de multa estabelecida durante a reserva.

Figura 6.7 Requisito Funcional 6 do DRO do Hotel.

Requisito Funcional 25.
Descrição: Reservar acomodação.
Agente Fornecedor/Receptor: Hóspede
Agente Executor: Funcionário
Entrada: O cadastro de reserva deve conter as seguintes informações:

Campo	Tipo	Tamanho	Descrição
Data de chegada	Alfanumérico	10 caracteres	Data de chegada do hóspede. Formato: dd/mm/aaaa.
Hora de chegada	Alfanumérico	5 caracteres	Hora de chegada do hóspede. Formato: hh:mm.
Data de saída	Alfanumérico	10 caracteres	Data prevista para saída do hóspede. Formato: dd/mm/aaaa.
Hora de saída	Alfanumérico	5 caracteres	Hora prevista para saída do hóspede. Formato: hh:mm.
Identificação do hóspede	Alfanumérico	RG: 9 caracteres CPF: 11 caracteres Passaporte: 15 caracteres.	RG ou CPF para brasileiros e passaporte para estrangeiros
Tipo de acomodação	Alfanumérico	6 caracteres	Tipo de acomodação desejada. Oferecer forma de seleção.
Nome dos acompanhantes	Alfabético	40 caracteres	
Idade dos acompanhantes	Numérico	3 números	
Valor da diária	Numérico	10 números	
Taxa de multa	Numérico	10 números	Cobrada em caso de desistência de última hora (a menos de 12 horas do início previsto da hora de chegada)
Dados do cartão de crédito	Alfanumérico	30 caracteres	Dados do cartão de crédito do hóspede
Desconto concedido	Numérico	10 números	(Opcional)

Processamento: Verificar a disponibilidade de acomodação no período indicado. Em caso afirmativo, efetuar a reserva de acomodação e exibir uma mensagem de sucesso indicando que a reserva foi efetuada. Senão, a reserva não deve ser efetuada e uma mensagem de alerta deve ser exibida indicando que não há acomodações do tipo solicitadas disponíveis para aquele determinado período.
Condição/Restrição: Se houver acomodações suficientes para atender a reserva.
Saída: Armazenar dados da reserva e exibir mensagem de sucesso. Senão, exibir mensagem de alerta.

Figura 6.8 Requisito Funcional 25 do DRM do Hotel.

Requisito Funcional 27.
Descrição: Cancelar reserva de acomodação.
Agente Fornecedor/Receptor: Hóspede
Agente Executor: Funcionário
Entrada:

Campo	Tipo	Tamanho	Descrição
Identificação do hóspede	Alfanumérico	RG: 9 caracteres CPF: 11 caracteres Passaporte: 15 caracteres.	RG ou CPF para brasileiros e passaporte para estrangeiros

Processamento: Verificar se o pedido de cancelamento da reserva de acomodação foi feito até 12 horas antes do início previsto para a *estadia no hotel*. Em caso afirmativo, remover o cadastro de reserva de acomodação sem maiores encargos e exibir uma mensagem de sucesso indicando que o cancelamento foi efetuado. Senão, exibir uma mensagem de alerta indicando que uma taxa de multa, estabelecida durante a reserva, deve ser cobrada.
Condição/Restrição: Nenhuma.
Saída: Exibir mensagem de sucesso. Senão, exibir mensagem de alerta.

Figura 6.9 Requisito Funcional 27 do DRM do Hotel.

Como se pode perceber, no Requisito Funcional 6 há mais de uma funcionalidade numa única frase e sob uma única identificação, corrompendo as Recomendações de Escrita no que se refere a requisito múltiplos. Desta forma, o parágrafo corrido e extenso foi desmembrado gerando os Requisitos Funcionais 25 e 27 no DRM do Hotel, ilustrados pelas Figuras 6.8 e 6.9, respectivamente. Percebeu-se que seriam dois requisitos pelo fato de haver duas ações diferentes que o sistema deveria executar, sendo uma a reserva de acomodação e a outra a remoção da reserva. São solicitações diferentes, mas que foram apresentadas juntamente.

É notório como o Formato proposto reestrutura a apresentação do Requisito Funcional, facilitando a sua compreensão através de uma aparência mais discreta.

Quanto à Figura 6.8, cuja funcionalidade principal identificada foi "reservar acomodação", os itens do Formato foram compostos da seguinte maneira: o Agente Fornecedor/Receptor foi designado ao hóspede, pois é ele quem solicita a reserva informando os dados e o funcionário foi atribuído ao Agente Receptor, pois ele é a interface que manuseia o sistema; os dados da Entrada foram preenchidos com informações adequadas ao seu tipo; já o Processamento, explica em detalhes a atividade que o sistema deve desempenhar para reservar a acomodação. Como no item Condição/Restrição foi estabelecido que deve haver

acomodações suficientes para efetuar a reserva, isso se traduziu para o item Processamento que o sistema deveria verificar a disponibilidade de acomodação. E conforme o Formato, a ação positiva (Em caso afirmativo) e negativa (Senão) foram descritas. A parte do requisito original que diz "A reserva somente deve ser concretizada se houver vagas suficientes para atendê-la" foi colocada no item Condição/Restrição, pois ao verificar o termo "se" observou-se que realmente se tratava de algo a ser respeitado; na Saída, foi estabelecida resposta de quando a reserva era concretizada e quando não era possível realizá-la.

O segundo requisito gerado, ilustrado pela Figura 6.9, foi identificado ao ler o seguinte trecho do Requisito Funcional 6 mostrada na Figura 6.7: "A remoção de reserva somente é permitida sem maiores encargos até 12 horas antes do início previsto para *estadia no hotel*. Após esse período, a remoção da reserva deve alertar o *funcionário* do hotel de que deve ser cobrada a taxa de multa estabelecida durante a reserva". Foi percebida que a forma apresentada por esta sentença apenas explica a remoção de reserva, o que indiretamente solicita a sua realização. Assim, foi definida para a Descrição a frase "Cancelar reserva de acomodação". Por ser mais usual o termo 'cancelar' para este caso, este foi escolhido ao invés de 'remover'.

Os itens do Requisito Funcional 27 foram formatados da seguinte maneira: os Agentes foram estabelecidos como no Requisito Funcional 25; a Entrada foi percebida ao analisar a entrada do Requisito Funcional 25, cujo campo principal que melhor identifica cada reserva é o campo 'identificação do hóspede', sendo este escolhido para cancelar reserva de acomodação; no item Processamento, as ações positiva (Em caso afirmativo) e negativa (Senão) foram descritas, uma vez que ação anterior (verificar se o pedido de cancelamento da reserva de acomodação foi feita até 12 horas antes do início previsto para a *estadia no hotel*) tem essa possibilidade; não houve nenhuma condição/restrição que implicasse a realização do requisito; para o item Saída foram definidas mensagens para que o usuário do sistema saiba o que aconteceu ao solicitar o cancelamento da reserva.

Os exemplos apresentados mostram a forma com que os Requisitos Funcionais foram sendo identificados e transformados para o Formato proposto. Conforme as sentenças originais iam sendo reformuladas, as Recomendações de Escrita foram sendo observadas. Após a conclusão do DRM do Hotel, o *Checklist* Pré-Inspeção foi aplicado.

6.4.3 Modificações no DRO do ATM

A avaliação final das Diretrizes propostas deu-se com a sua aplicação no DRO do ATM.

Para exemplificar as modificações ocorridas nos Requisitos Funcionais originais após o uso do Formato proposto, serão mostrados exemplos de requisitos originais e os mesmos modificados. Como as mudanças ocorreram também é explicado.

Exemplo1:

Requisito Funcional 7

- Descrição: Diálogo de autorização: O usuário é solicitado a entrar sua senha. O ATM verifica o código do banco e senha com o computador do banco.
- Entrada: Senha do usuário, código do banco do cartão bancário.
- Processamento: Envie o número serial e senha para o computador do banco, receba resposta do banco.
- Saída: Aceite ou rejeite autorização do banco.

Figura 6.10 Requisito Funcional 7 do DRO do ATM.

Requisito Funcional 7.
Descrição: Iniciar o diálogo de autorização.
Agente Fornecedor/Receptor: Cliente
Agente Executor: Cliente
Entrada:

Campo	Tipo	Tamanho	Descrição
Senha	Numérico	10 inteiros	Senha do cliente
Código do banco	Numérico	10 inteiros	Código do banco do cartão bancário

Processamento: Solicitar ao cliente a entrada de sua senha. Enviar o código do banco e a senha para o Computador do Banco para que sejam verificados. Receber resposta do Computador do Banco.
Condição/Restrição: Nenhuma.
Saída: Aceitar ou rejeitar a autorização do Computador do Banco.

Figura 6.11 Requisito Funcional 7 do DRM do ATM.

Analisando o requisito da Figura 6.10, nota-se que sua Descrição está repleta de informações não condizentes com o definido no Formato. Este é o terceiro documento estudado em que a Descrição do Requisito Funcional aparece com informações mistas, ou

seja, informações que são entrada de dados ou algo para ser processado pelo sistema ou certa condição/restrição a ser obedecida ou até mesmo alguma resposta gerada pela execução do requisito. Isso acaba prejudicando o entendimento do que realmente deve ser feito pelo requisito. Eliminando-se tais informações, a nova Descrição começando por um verbo ficou "Iniciar o diálogo de autorização". Tal diálogo é requisitado e iniciado pelo cliente, o que o torna Agente Fornecedor/Receptor e Agente Executor, uma vez que ele informa os dados necessários para a sua realização e comunica-se independentemente com o sistema.

Os dados de entrada foram discriminados pela tabela recomendada no Formato proposto, permitindo que se verifique a compatibilidade dos dados enviados através do ATM com os recebidos pelo Computador do Banco.

Já no item Processamento, teve-se de proceder diversos ajustes. Foi excluída a forma imperativa das sentenças, obedecendo as Recomendações de Escrita descritas no Capítulo 5. Atentando-se para as perguntas do *Checklist* Pré-Inspeção, foi identificado que não eram coerentes os dados enviados e recebidos entre este requisito e outro, assim o número serial não deveria ser enviado ao Computador do Banco. Mesmo através do item Entrada, dava-se para perceber que o correto seria código do banco e não número serial.

Não houve nenhuma condição/restrição especificada para este requisito. E por último, o item Saída, também foi expresso de forma a eliminar a frase imperativa.

Exemplo 2:

Requisito Funcional 11

- Descrição: O tipo de transação que o ATM oferece é saque.
- Entrada: Autorização completada com sucesso. Entrar a quantidade a sacar.
- Processamento: Quantidade entrada é comparada com **m**.
- Saída: Quantidade de dinheiro a ser dispensada é mostrada. Inicia a seqüência de saque inicial.

Figura 6.12 Requisito Funcional 11 do DRO do ATM.

Requisito Funcional 12

- Descrição: Seqüência de saque inicial: se o saque é muito alto refaça a transação.
- Entrada: Cliente entrou a quantidade de dinheiro.
- Processamento: Erro se a quantidade é maior do que **m**.
- Saída: Inicie transação ou reinicie o diálogo de transação se a quantidade não está dentro da política de transação pré-definida.

Figura 6.13 Requisito Funcional 12 do DRO do ATM.

Requisito Funcional 13

- Descrição: Realizar transação.
- Entrada: Seqüência de saque inicial com sucesso.
- Processamento: Envie o pedido para o computador do banco.
- Saída: Espere por resposta do computador do banco.

Figura 6.14 Requisito Funcional 13 do DRO do ATM.

Requisito Funcional 11.
Descrição: Oferecer a transação de saque.
Agente Fornecedor/Receptor: Cliente
Agente Executor: Cliente
Entrada:

Campo	Tipo	Tamanho	Descrição
Quantidade	Numérico	Conforme o tamanho de **m** definido.	Quantidade de dinheiro a sacar

Processamento: Comparar a quantidade a sacar com **m**. Se a quantidade é maior do que **m**, refazer a transação ou reiniciar o diálogo de transação, pois a quantidade não está dentro da política de transação pré-definida. Senão, realizar a transação enviando o pedido para o Computador do Banco.
Condição/Restrição: (Requisito Funcional 9. concluído com sucesso)
Saída: Mostrar a quantidade de dinheiro a ser dispensada. Refazer a transação ou reiniciar o diálogo de transação. Senão, esperar por uma resposta do Computador do Banco.

Figura 6.15 Requisito Funcional 11 do DRM do ATM.

Este exemplo mostra três Requisitos Funcionais do DRO do ATM que foram agregados num único requisito. Como pode ser observado, o Requisito Funcional 11 da Figura 6.12, gera uma saída positiva e outra negativa em razão do seu processamento, que é comparar a quantidade de dinheiro a ser sacada. Porém, tais respostas estão separadas pelos Requisitos Funcionais 12 e 13, Figuras 6.13 e 6.14, respectivamente. Mas de acordo com o Formato proposto deve-se mantê-las juntas. O Requisito Funcional 12 representa a saída gerada quando quantidade de dinheiro a sacar é maior que **m**, e o Requisito Funcional 13, quando a quantidade de dinheiro entrada é menor ou igual a **m**, o qual está subentendida através da sentença 'Seqüência de saque inicial com sucesso'.

Juntando-se os três Requisitos Funcionais originais, o Requisito Funcional 11 da Figura 6.15 foi elaborada. Sua composição deu-se da seguinte forma: a Descrição do

Requisito Funcional 11 do DRO foi modificada para iniciar por um verbo principal de ação; os Agentes Fornecedor/Receptor e Executor foram designados ao cliente porque é ele quem interage com o sistema ao informar a quantidade de dinheiro desejado a ser sacado; a Entrada foi especificada através da tabela sugerida no Formato; as sentenças imperativas descritas nos requisitos originais foram eliminadas da nova especificação do Requisito Funcional; o item Processamento teve de ser complementado com as informações dos Requisitos Funcionais 12 e 13 do DRO. Assim, o item Descrição destes requisitos 12 e 13, que seriam refazer transação e realizar transação, tornaram-se as atividades do item Processamento no Requisito Funcional 11 do DRM. Com isso, eliminaram-se os termos ambíguos 'muito alto' e 'com sucesso'; a Condição/Restrição identificada foi que apenas quando o cliente for autorizado pelo Computador do Banco é que se poderá realizar a transação de saque. Desta forma, colocou-se "Requisito Funcional 9 concluído com sucesso" neste item, cuja Descrição é "Finalizar o processo de autorização"; no item Saída, as informações também foram agregadas.

Através dos exemplos apresentados nesta seção, foi mostrado o uso de cada item do Formato de especificação de Requisitos Funcionais, assim como a maneira de escrevê-los. As Recomendações de Escrita e o *Checklist* Pré-Inspeção foram procedidas. Com isso, pretendeu-se mostrar como as Diretrizes foram aplicadas nos DR estudados, cujo objetivo era definir uma forma de se escrever requisitos de tal maneira que contribuísse para a qualidade de um DR.

Na seção seguinte é apresentada a aplicação das técnicas de inspeção sobre tais DRs que foram previamente reformuladas.

6.5 Aplicação de Técnicas de inspeção nos DRMs

Com o objetivo de avaliar a eficácia das Diretrizes propostas, os DRMs também foram inspecionados. A aplicação das técnicas de inspeção em cada DRM considerado no estudo de caso será apresentada sob a seguinte estrutura:

I. Aplicação da PBR-Usuário (seria a aplicação da técnica de inspeção PBR-Usuário no DRM após a sua modificação; os tipos de defeitos encontrados são especificados).
II. Aplicação do *Checklist* (seria a aplicação do *checklist* no DRM após a sua modificação; os tipos de defeitos identificados são relatados).

III. Aplicação da TUCCA (seria a aplicação da técnica TUCCA, também após a modificação do DRM; os tipos de defeitos encontrados são apresentados).

6.5.1 Inspeção no DRM do PG

I. <u>Aplicação da PBR-Usuário no DRM do PG</u>

- <u>Omissão</u>:

1) Embora algumas informações tenham sido acrescentadas no DRM, alguns Requisitos Funcionais continuaram com informações insuficientes para especificar o caso de uso, pois o acréscimo de determinadas ações poderia modificar o sistema. Assim, os defeitos do tipo omissão 1 que foi identificado pela PBR-Usuário no DRO do PG também permaneceram aqui. Porém, o Formato de especificação de Requisito Funcional e o *Checklist* Pré-Inspeção, que fazem parte das Diretrizes propostas, dão subsídio para que esses defeitos de omissão não estejam presentes na especificação do Requisito Funcional.

2) O defeito do tipo omissão 3, identificado pela PBR-Usuário no DRO do PG, foi mantido no DRM correspondente, pois era falta de informações, não coberta pelas Diretrizes propostas.

3) O defeito do tipo omissão 4, identificado pela PBR-Usuário no DRO do PG, foi mantido no DRM correspondente, pois as Diretrizes não tratam de Requisitos de interface externas.

- <u>Informação Inconsistente</u>:

1) O defeito do tipo informação inconsistente 2, identificado pela PBR-Usuário no DRO do PG, também foi identificado no DRM correspondente, uma vez que é um defeito a ser encontrado durante a inspeção.

- <u>Fato Incorreto</u>:

1) O defeito do tipo fato incorreto 1, identificado pela PBR-Usuário no DRO do PG, também permaneceu no DRM correspondente, pois tratava-se de Requisitos Não-Funcionais de Desempenho.

II. Aplicação do *Checklist* no DRM do PG

- Informação Inconsistente e Fato Incorreto:

1) Os mesmos defeitos do tipo fato incorreto e informação inconsistente que foram encontrados na inspeção PBR-Usuário no DRM do PG também foram identificados aqui.

III. Aplicação da TUCCA no DRM do PG

Leitura AGRT

- Omissão:

1) Os mesmos defeitos do tipo omissão que foram encontrados pela Leitura AGRT da inspeção TUCCA, no DRO do PG, também foram identificados aqui.

Leitura UCRT

- Omissão:

1) Os mesmos defeitos do tipo omissão que foram encontrados pela PBR-Usuário, no DRM do PG, foram identificados aqui.

- Fato Incorreto:

1) O mesmo defeito do tipo fato incorreto que foi encontrado na inspeção PBR-Usuário, no DRM do PG, também foi identificado aqui.

6.5.2 Inspeção no DRM do Hotel

I. Aplicação da PBR-Usuário no DRM do Hotel

- Omissão:

1) Os defeitos do tipo omissão 1, 3, 4 e 5 encontrados pela PBR-Usuário no DRO do Hotel mantiveram-se no seu DRM correspondente, pois no DRO não estavam determinadas informações para que os itens do Formato proposto fossem preenchidos. Porém, o Formato de especificação de Requisito Funcional e o *Checklist* Pré-Inspeção permitem que tais defeitos

de omissão não sejam apresentados. Nos exemplos mostrados, os itens do Formato foram completados para mostrar como deveria ser o Requisito Funcional modificado.

2) O defeito do tipo omissão 2 encontrado pela PBR-Usuário no DRO do Hotel manteve-se no seu DRM correspondente, pois as Diretrizes não verificam ausência de funcionalidades, mas podem levar à sua descoberta.

- Miscelânea:

1) O defeito do tipo miscelânea identificado pela PBR-Usuário no DRO do Hotel também permaneceu no DRM correspondente, uma vez que não estavam determinadas informações no DRO. Mas caso o Formato fosse usado para a elaboração do DRO, tal defeito dificilmente ocorreria.

II. Aplicação do *Checklist* no DRM do Hotel

- Questões gerais:

1) O mesmo tipo de defeito relatado nas Questões Gerais 3, através da aplicação do *Checklist* no DRO do Hotel, foi detectado aqui.

- Omissão:

1) O defeito do tipo omissão 2 que foi encontrado pela PBR-Usuário no DRM do Hotel foi igualmente identificado pelo *Checklist*.

- Miscelânea:

1) O defeito do tipo miscelânea identificado pela PBR-Usuário no DRM do Hotel também foi encontrado pelo *Checklist*.

III. Aplicação da TUCCA no DRM do Hotel

Leitura AGRT

- Omissão:

1) O defeito do tipo omissão 2 que foi encontrado pela PBR-Usuário no DRM do Hotel foi identificado na Leitura AGRT.

Leitura UCRT

- Omissão:

1) O defeito do tipo omissão 1 que foi encontrado pela PBR-Usuário no DRM do Hotel também foi identificado aqui.

- Miscelânea:

1) O defeito do tipo miscelânea identificado pela PBR-Usuário no DRM do Hotel também foi detectado nesta Leitura UCRT.

6.5.3 Inspeção no DRM do ATM

I. Aplicação da PBR-Usuário no DRM do ATM

- Omissão:

1) O defeito de omissão detectado pela PBR-Usuário no DRO do ATM não foi eliminado no seu DRM correspondente. Era um defeito não percebido usando as Diretrizes. E mesmo porque as Diretrizes são instruções para melhor definir um Requisito Funcional e não é uma técnica de inspeção.

- Fato incorreto:

1) Os defeitos do tipo fato incorreto 1,2 e 4 encontrados pela PBR-Usuário no DRO do ATM foram mantidos aqui, pois não fazem parte das Diretrizes a correçãodos mesmos.

- Miscelânea:

1) Alguns requisitos não ficaram bem esclarecidos porque no DRO do ATM não estavam tais informações. Assim o defeito miscelânea encontrado pela PBR-Usuário no DRO do ATM não foi eliminado no seu DRM.

II. Aplicação do *Checklist* no DRM do ATM

- Fato incorreto:

1) Os defeitos do tipo fato incorreto 2 e 4 encontrados pela PBR-Usuário no DRO do ATM foram identificados pelo *Checklist*.

- Miscelânea:

1) O defeito do tipo miscelânea encontrado pela PBR-Usuário no DRM do ATM foi igualmente percebido aqui.

III. Aplicação da TUCCA no DRM do ATM

Leitura AGRT

- Omissão:

1) O mesmo defeito do tipo omissão 1 detectado pela Leitura AGRT no DRO do ATM permaneceu no DRM correspondente, assim as funcionalidades existentes na seção Funções do Produto não foram descritas na seção Requisitos Funcionais. O Formato proposto pode até levar à descoberta de demais funcionalidades, mas não é esse o seu objetivo.

2) Como a TUCCA não trata as funcionalidades requisitadas quando o ator é o próprio sistema, o defeito do tipo omissão 2 detectada pela Leitura AGRT no DRO do ATM permaneceu aqui.

Leitura UCRT

- Omissão:

1) O mesmo defeito do tipo omissão descrito pela PBR-Usuário no DRM do ATM foi observado por esta Leitura UCRT.

- Fato incorreto:

1) Os defeitos do tipo fato incorreto informados pela PBR-Usuário no DRM do ATM foram detectados nesta Leitura.

- Miscelânea:

1) O defeito do tipo miscelânea encontrado pela PBR-Usuário no DRM do ATM também foi identificado aqui.

Aplicadas as técnicas de inspeção tanto nos DROs quanto nos DRMs, a Tabela 6.3 mostra uma síntese do estudo de caso quanto à quantidade de defeitos que foram encontrados pelas inspeções PBR-Usuário, *Checklist* e TUCCA nos DROs e DRMs, ou seja, nos documentos analisados antes e após a aplicação das Diretrizes propostas. Para os três DRs considerados, PG, Hotel e ATM, diferentes tipos de defeitos foram checados. A marcação com 'X' significa que o tipo de defeito não é avaliado por determinada inspeção. Para a técnica TUCCA, a contagem de defeitos foi realizada considerando-se a soma dos defeitos identificados pela Leitura AGRT e UCRT.

Tabela 6-3 Quantidade de defeitos encontrados nos DROs e DRMs.

DR		Inspeção	Quantidade de defeitos encontrados para os diferentes tipos de defeitos						
			Questões Gerais	Omissão	Ambiguidade	Informação Inconsistente	Fato Incorreto	Informação Estranha	Miscelânea
PG	DRO	*PBR-Usuário*	X	11	1	6	5	0	3
		Checklist	3	8	1	6	5	X	0
		TUCCA	X	17	1	8	5	0	5
	DRM	*PBR-Usuário*	X	6	0	2	2	0	0
		Checklist	0	0	0	2	2	X	0
		TUCCA	X	12	0	0	2	0	0
Hotel	DRO	*PBR-Usuário*	X	11	0	0	0	0	2
		Checklist	3	3	2	0	0	X	2
		TUCCA	X	11	1	0	0	0	4
	DRM	*PBR-Usuário*	X	11	0	0	0	0	2
		Checklist	1	2	0	0	0	X	2
		TUCCA	X	11	0	0	0	0	2
ATM	DRO	*PBR-Usuário*	X	1	0	10	6	0	3
		Checklist	1	0	1	10	5	X	3
		TUCCA	X	5	0	10	6	0	7
	DRM	*PBR-Usuário*	X	1	0	0	5	0	3
		Checklist	0	0	0	0	4	X	3
		TUCCA	X	4	0	0	5	0	3

6.6 Considerações Finais

Neste capítulo foi apresentado o estudo de caso realizado com objetivo de validar as Diretrizes propostas para elaboração de DR com ênfase nos Requisitos Funcionais. Para tanto, três DRs com contextos diferentes foram estudados.

Para as atividades de avaliação conduzidas foram aplicadas técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA nesses DRs. Com isso, os defeitos encontrados foram coletados

e analisados, os quais serviram de suporte para a implementação das Diretrizes propostas. Após a adequação dos DRs à essas Diretrizes, a qual se traduziu na adequação da apresentação dos Requisitos Funcionais a um Formato de especificação de Requisitos Funcionais, às Recomendações de Escrita e a uma verificação prévia de principais pontos a serem checados, tais DRs passaram pelo mesmo processo de inspeção através das três técnicas citadas. O relato das aplicações dessas técnicas de inspeção nos DRs antes e depois de sua transformação pela Diretrizes foi descrito.

Como resultado do estudo realizado, pôde-se destacar os seguintes aspectos:

- Com o uso do Formato de especificação de Requisito Funcional, os requisitos ficaram mais claramente especificados e objetivos. Além de que se permite uma compreensão melhor de suas informações. Isso significa que compondo um requisito discriminando suas informações através dos itens do Formato, torna-se mais fácil assimilar o que está sendo solicitado, há maior clareza.
- O Formato conduz ao levantamento de determinadas informações já durante a fase de elicitação de requisitos, uma vez que seus itens precisam ser completados.
- Com as Recomendações de Escrita foram evitados principalmente os termos inconsistentes e a descrição de requisitos múltiplos. Também foi possível verificar o que deve ser evitado numa especificação de um requisito para não causar dúvida, erro ou até mesmo para melhor designar um requisito.
- As informações dos requisitos que antes ficavam misturadas foram separadas pelos itens do Formato.
- Os dados do item Entrada, caracterizados por uma tabela que discrimina os campos de entrada, evitaram que determinadas informações passassem desapercebidas, pois a tabela separa os campos por rótulos.
- Analisando-se os defeitos encontrados nos DROs e DRMs, descritos nas Seções 6.3 e 6.4, verificou-se que houve redução deles nos DRMs. O uso das Diretrizes implicou nessa diminuição, pois o Formato forneceu itens básicos e instruções para que eles fossem percebidos. Além disso, as Recomendações de Escrita e o *Checklist* Pré-Inspeção permitiram que alguns defeitos não fossem cometidos.
- O tempo de aplicação das técnicas de inspeção nos DRMs foi menor do que nos DROs, principalmente o da TUCCA, uma vez que o Formato facilita seus passos e os defeitos foram reduzidos. A Tabela 6.3 mostra o tempo total gasto ao se aplicar as técnicas de inspeção nos DROs e nos DRMs.

Tabela 6-4 Tempo de aplicação das técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA nos DROs e DRMs.

DR		Tempo		
		PBR-Usuário	**Checklist**	**TUCCA**
PG	DRO	6 horas e 30 min	1 hora	13 horas
	DRM	5 horas	45 min	9 horas e 40 min
Hotel	DRO	5 horas e 30 min	40 min	9 horas e 30 min
	DRM	5 horas	30 min	8 horas
ATM	DRO	6 horas e 20 min	1 hora	10 horas
	DRM	5 horas	45 min	8 horas

- Visto que as técnicas de inspeção PBR-Usuário e TUCCA são um tanto demoradas, fica-se comprovado que a aplicação delas em DRs mal elaborados não é vantajoso. Usando-se as Diretrizes, diversas deficiências nos DRs puderam ser eliminadas, como uso de termos inconsistentes, diversos requisitos agrupados, termos ambíguos, sentenças escritas de forma pouco compreensível, falta de dados de entrada para um requisito.
- Verificou-se que a descoberta de defeitos durante a modelagem, a qual é um dos passos realizados pela PBR-Usuário e TUCCA, é custosa, há desperdício de recursos e perde-se muito tempo, pois modelagem é um processo demorado. Certificar-se de que várias informações estão faltando, inconsistentes durante a modelagem, é um grande re-trabalho. Não é compensador ter que esperar pela modelagem para descobrir que o DR não oferece condições de uso para ser base de um processo de implementação. O DR deveria possuir o mínimo de informações que suportasse a construção do seu MCU.
- Foi notória a influência do Formato para especificar Requisito Funcional, principalmente na aplicação da técnica TUCCA nos DRMs, cujos passos foram bem facilitados. Isso se deve ao fato de que observando os defeitos e dificuldades encontradas durante a aplicação das técnicas nos DROs é que as Diretrizes foram propostas. As marcações realizadas pela TUCCA foram mais práticas.
- Pelo fato da própria técnica TUCCA ser sistemática e procedimental, a qual requer um certo rigor na sua aplicação, percebeu-se que não valeria a pena submeter um DR sem um mínimo de qualidade para este tipo de inspeção, pois muitos defeitos seriam encontrados e o MCU gerado não seria apropriado.

Esses resultados forneceram indícios de que as Diretrizes contribuem para uma melhor qualidade de documentação em geral. Além de fácil utilização, define um padrão que caracteriza igualmente os Requisitos Funcionais, ajudando na sua compreensão e eliminando defeitos.

Capítulo 7

7. Conclusões

É comum que se lembrem da qualidade de DR quando erros, incertezas ou falhas são detectadas em estágios avançados do processo de desenvolvimento de software. Porém, a atenção que deveria ter sido dado a esse documento é no primeiro estágio do ciclo de vida do software, pois é a partir dele que artefatos posteriores serão baseados. A importância da qualidade do DR é o fato dele ser a base principal para uma adequada implementação do sistema, além de que um DR mal definido pode resultar-se num software inesperado.

De forma a garantir um DR com mais qualidade, inspeções apropriadas para documentação devem ser aplicadas. A técnica de inspeção TUCCA [Belgamo, 2004], cujo presente trabalho pretendeu facilitar a sua aplicação, é uma das técnicas que tem como artefato de avaliação o DR. Essa técnica avalia o DR à medida que suas diretrizes são aplicadas para a elaboração do MCU. A modelagem proporcionada por esta técnica conduz à MCU semelhantes e padronizados, independente de quem a aplica, pois a sistematização de suas diretrizes torna menos influente a subjetividade e experiência do projetista.

Por ser criteriosa e bastante metódica, os passos da técnica TUCCA beneficiar-se-ia em muito caso DRs tivessem moldados de tal forma que facilitaria a sua aplicação. Não só esforço e tempo seriam poupados, mas o procedimento de seus passos tornar-se-ia mais fácil. Considera-se isso como uma das razões principais que este trabalho pretendeu servir-se.

Sendo assim, neste trabalho apresentaram-se as Diretrizes para elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais, as quais fornecem um conjunto de instruções que orientam a construção desse documento a fim de atender às propriedades de qualidade que ele deve possuir e, principalmente, de facilitar a aplicação da técnica TUCCA.

As Diretrizes propostas são compostas de três itens: um Formato para especificação de Requisitos Funcionais, Recomendações de Escrita e um *Checklist* Pré-Inspeção. O Formato definido permite padronizar e detalhar a descrição dos Requisitos Funcionais por meio de vários itens (Descrição, Agente Fornecedor/Receptor, Agente Executor, Entrada, Processamento, Condição/Restrição, Saída), facilitando seu entendimento e reduzindo a falta

de informações necessárias para a sua implementação. Para a composição de cada item, uma instrução de como descrevê-la foi dada. As Recomendações de Escrita foram sugeridas como forma de minimizar inconsistência, interpretações errôneas e a não completitude das sentenças que descrevem os Requisitos Funcionais. O *Checklist* Pré-Inspeção permite uma breve avaliação do DR, checando alguns pontos que devem ser satisfeitos, minimamente, para que um processo de inspeção possa ser iniciado.

Para a definição dessas Diretrizes, foram aplicadas as técnicas de inspeção PBR-Usuário, *Checklist* e TUCCA em três DRs: PG (*Parking Garage*), um sistema para controle de estacionamento; ATM (*Automatic Teller Machine*), um sistema para automatização de banco; e Hotel, um sistema para gerenciamento de um hotel. Essa atividade teve como objetivo analisar as deficiências encontradas nesses DRs e verificar o que poderia ser feito para melhorá-las. Conforme as técnicas foram sendo aplicadas, as Diretrizes foram sendo elaboradas. Para evitar certos tipos de defeitos, procurou-se agregar no Formato algumas instruções para deixar mais visível as informações essenciais de modo que o Requisito Funcional ficasse claro e para facilitar a aplicação da TUCCA.

De forma a validar as Diretrizes, foi realizado um estudo de caso utilizando os mesmos DRs que foram utilizados para ajudar na definição das mesmas. Os DRs originais foram modificados de acordo com as Diretrizes e as mesmas técnicas aplicadas inicialmente foram aplicadas novamente. Observou-se que nos resultados obtidos para todos os DRs e para todas as técnicas, a quantidade de defeitos identificados foi bastante reduzida. Além disso, o tempo de aplicação foi substancialmente menor, mesmo levando-se em conta que os DRs já eram conhecidos. Ou seja, as Diretrizes propostas realmente geram DRs mais completos e menos ambíguos.

Salienta-se que em relação à técnica TUCCA, para a qual tinha-se o objetivo de criar um modelo de DR mais apropriado de tal modo a facilitar alguns passos da ferramenta que está sendo desenvolvida nesse sentido, ressalta-se que uma atenção muito grande foi tomada em relação à técnica TUCCA e que os resultados obtidos indicam a contribuição da mesma nesse caso particular.

7.1 Contribuições do trabalho

Algumas contribuições deste trabalho são relacionadas abaixo:

- Definição das Diretrizes para elaboração de DR com ênfase em Requisitos Funcionais, a qual é composta por um Formato, Recomendação de Escrita e *Checklist* Pré-Inspeção, sendo que as contribuições de cada um desses 3 itens, separadamente, são:
  - Formato: permite que o Requisito Funcional se apresente num padrão de tal forma que suas informações fiquem mais claras e completas.
  - Recomendação de Escrita: fornece recomendações que se observadas no momento da escrita do Requisito Funcional evita determinados defeitos durante uma inspeção.
  - *Checklist* Pré-Inspeção: provê um sucinto *checklist* a ser aplicado antes do processo de inspeção, de forma a auxiliar na avaliação da qualidade do DR. Ressalta-se que não foi encontrado na literatura estudo que informasse quando iniciar uma inspeção, ou seja, quais critérios de entrada o DR deveria obedecer antes de ser inspecionado.
- As Diretrizes ajudam na aplicação da TUCCA, tornando o DR mais adequado para aplicação de seus passos.
- Redução da quantidade de defeitos detectados pela PBR-Usuário, *Checklist* e TUCCA quando os DRs utilizados por essas técnicas foram elaboradas segundo as Diretrizes propostas.
- Identificação de funcionalidades durante a aplicação da técnica TUCCA nos DRs estudados que eram desconsideradas pelo fato de serem realizadas pelo próprio Sistema, o qual a técnica TUCCA não tratava.

7.2 Trabalhos futuros

A seguir relacionam-se os aspectos que abrem perspectivas de continuidade deste trabalho:

- Ampliar as Diretrizes para que abranjam não somente os Requisitos Funcionais, mas também outros domínios de aplicação.
- Aplicar as Diretrizes em DRs reais, ou seja, utilizados nas indústrias.
- Aperfeiçoar o *Checklist* Pré-Inspeção para que juntamente com o Modelo de Maturidade para Documentação de Sistema (DMM), comentado no Capítulo 3, na Seção 3.2, seja definido um critério melhor para estabelecer quando iniciar um processo de inspeção no DR.

REFERÊNCIAS BIBLIOGRÁFICAS

[Aurum et al., 2002] Aurum, A.; Petersson, H.; Wohlin, C. State-of-the-Art: Software Inspections After 25 Years. **Software Testing, Verification and Reliability**, Chichester, v.12, n.3, p.133-154, Sept. 2002.

[Basili et al., 1996a] Basili, V. R.; Caldiera, G.; Lanubile, F.; Shull, F. Studies on Reading Techniques. In: **ANNUAL SOFTWARE ENGINEERING WORKSHOP**, 21., 1996, Greenbelt, Maryland. **Anais...** Greenbelt: NASA/Goddard Software Engineering Laboratory Series, 1996. p.59-65.

[Basili et al., 1996b] Basili, V. R.; Green, S.; Laitenberger, O.; Lanubile, F; Shull, F.; Sorumgard, S.; Zelkowitz, M. V. The Empirical Investigation of Perspective-Based Reading. **Empirical Software Engineering: An International Journal**, v.1, n.2, p.133-164, 1996.

[Belgamo, 2004] Belgamo, A. **GUCCRA: Técnicas de Leitura para Construção de Modelos de Casos de Uso e Análise do Documento de Requisitos**, 2004, 163 p. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, UFSCar, São Carlos.

[Belgamo & Fabbri, 2004a] Belgamo, A.; Fabbri, S. Um Estudo sobre a Influência da Sistematização da Construção de Modelos de Casos de Uso na Contagem dos Pontos de Casos de Uso. In: **SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE**, 3., 2004, Brasília, DF, Brasil. Anais... Brasília: Universidade Católica de Brasília , 2004.

[Belgamo & Fabbri, 2004b] Belgamo, A.; Fabbri, S. GUCCRA: Contribuindo para a Identificação de Defeitos em Documentos de Requisitos Durante a Construção de Modelos de Casos de Uso. In: **WORKSHOP EM ENGENHARIA DE REQUISITOS**, 4., 2004, Tandil, Argentina. Anais... Tandil: 2004, p. 100-111.

[Belgamo & Fabbri, 2004c] Belgamo, A.; Fabbri, S. Constructing Use Case Model by Using a Systematic Approach: Description of a Study. In: **WORKSHOP EM ENGENHARIA DE REQUISITOS**, 4., 2004, Tandil, Argentina. Anais... Tandil: 2004, p. 251-262.

[Belgamo & Fabbri, 2005] Belgamo, A.; Fabbri, S.TUCCA: Técnica de Leitura para apoiar a Construção Modelos de Casos de Uso e a Análise de Documentos de Requisitos. In: **SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE**, 19., 2005, Uberlândia, MG, Brasil. Anais... Uberlândia: Universidade Federal de Uberlândia, 2005.

[Belgamo et al., 2005a] Belgamo, A.; Fabbri, S.; Maldonado, J. C. Avaliando a Qualidade da Técnica GUCCRA com Técnica de Inspeção. In: **WORKSHOP EM ENGENHARIA DE REQUISITOS**, 8., 2005, Porto, Portugal. Anais… Porto: Faculty of Engineering of the University of Porto (FEUP), 2005.

[Belgamo et al., 2005b] Belgamo, A., Fabbri, S.; Maldonado, J. C. TUCCA: Improving the Effectiveness of Use Case Construction and Requirement Analysis. In: **IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING**, 4., 2005, Noosa Heads, Queensland, Austrália. Anais… Noosa Heads: Sheraton Noosa Resort & Spa, 2005.

[Ben Achour, 1998] Ben Achour, C. Guiding Scenario Authoring. In: **JAPANESE CONFERENCE ON INFORMATION MODELLING AND KNOWLEDGE BASES**, 8., Vamala, Finland, 1998, p.181-200.

[Brykczynski, 1999] Brykczynski, B. A survey of software inspection checklists. **ACM SIGSOFT Software Engineering Notes**. ACM Press New York, v. 24, n. 1, p. 82, Jan. 1999.

[Carvalho et al., 2001] Carvalho, A. E. S.; Tavares, C.; Castro, J. B. Uma Estratégia para Implantação de uma Gerência de Requisitos Visando a Melhoria dos Processos de Software. In: **WORKSHOP EM ENGENHARIA DE REQUISITOS**, 1, Buenos Aires, Argentina. **Anais**... Buenos Aires: 2001, p 32-54.

[Christel & Kang, 1992] Christel, M.G.; Kang, K.C. **Issues in Requirements Elicitation.** Software Engineering Institute, Technical Report CMU/SEI-92-TR- 012 ESC-TR-92-012, Sept. 1992.
Disponível em: <http://www.sei.cmu.edu/pub/documents/92.reports/pdf/tr12.92.pdf>. Acesso em: 24 maio 2003.

[Colanzi, 1999] Colanzi, T. E. **Uma Abordagem Integrada de Desenvolvimento e Teste de Software Baseada na UML**, 1999, 143 p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Matemáticas e Computação, USP, São Carlos.

[Coleman, 1994] Coleman, D.; Bodoff, S.; Arnold, P. **Object-Oriented Development: The Fusion Method.** Englewood Cliffs, New Jersey: Prentice Hall, 1994. 312p.

[Construx, 2001] Construx. Software Requirement Specification Template, 2001. Disponível em: <http://www.construx.com/docs/member/cxone/CxTemp_SoftwareRequirementsSpecification.doc> Acesso em: 16 maio 2005.

[Construx, 2002] Construx. Software Requirement Specification Checklist, 2002. Disponível em: <http://www.construx.com/docs/member/cxone/CxCheck_SoftwareRequirementsSpecification.pdf> Acesso em: 16 maio 2005.

[CSDL, 1996] Collaborative Software Development Laboratory, Department of Information and Computer Sciences. University of Hawaii. **Software Requirements Specification Document Template**, 1996. Disponível em: <http://www2.ics.hawaii.edu/~johnson/413/lectures/5.2.html>. Acesso em: 05 abril 2005.

[Dahlstedt, 2003] Dahlstedt, Å. Requirements Engineering - Chapter 11. Department of Computer Science, University of Skövde. 2003. Disponível em: <http://www.ida.his.se/ida/kurser/informationsystems_engineering/kursmaterial/forelasningar/Chapter11_2003.pdf>. Acesso em: 12 maio 2004.

[Damian, 2000] Damian, D. E. H. **Challenges in Requirements Engineering.** Computer Science Technical Reports. Department of Computer Science. The University of Calgary, Calgary, Canada, T2N 1N4.1999-645-08. 4 Jan., 2000. Disponível em: <http://pharos.cpsc.ucalgary.ca/Dienst/Repository/2.0/Body/ncstrl.ucalgary_cs/1999-645-08/pdf>. Acesso em: 08 mar. 2004.

[Damian et al., 2003] Damian, D; Chisan, J.; Vaidyanathasamy, L.; Pal, Y. An Industrial Case Study of the Impact of Requirements Engineering on Downstream Development. In: **INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING**, 2., 2003, Roman Castles, Rome, Italy. **Anais...** Roman Castles: University of Rome "Tor Vergata", IEEE Computer Society Press, 2003. p.40.

[Damian et al., 2004] Damian, D.; Zowghi, D.; Vaidyanathasamy, L.; Pal, Y. An industrial case study of immediate benefits of requirements engineering process improvement at the Australian Center for Unisys Software. **Journal of Empirical Software Engineering**, v.9, n.1 e 2, p.45-75, Mar. 2004.

[Davis, 1993a] Davis, A. M. **Software requirements: Objects, Functions and States.** 2nd ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1993.

[Davis et al., 1993b] Davis, A.; Overmyer, S.; Jordan, K.; Caruso, J.; Dandashi, F.; Dinh, A.; Kincaid, G.; Ledeboer, G.; Reynolds, P.; Sitaram, P.; Ta, A.; Theofanos, M. Identifying and Measuring Quality in a Software Requirements Specification. In: **INTERNATIONAL SOFTWARE METRICS SYMPOSIUM** (IEEE), 1., 1993, Baltimore. **Anais...** Baltimore, IEEE Xplore, 1993. p.141-152.

[Denger et al., 2003] Denger, C.; Berry, D.M.; Kamsties, E.. Higher Quality Requirements Specifications through Natural Language Patterns. In: **IEEE INTERNATIONAL CONFERENCE ON SOFTWARE-SCIENCE, TECHNOLOGY & ENGINEERING**, 3., Herzlia, Israel. **Anais**… IEEE Computer Society, Washington, USA, 2003, p.80-90.

[Fabbrini et al., 2000] Fabbrini, F.; Fusani, M.; Gnesi, S.; Lami, G. Quality Evaluation of Software Requirements Specifications. In: **INTERNATIONAL PROCEEDING OF SOFTWARE & INTERNET QUALITY WEEK CONFERENCE**, 13., 2000, San Francisco, California, USA.

[Fabbrini et al., 2001] Fabbrini, F.; Fusani, M.; Gnesi, S.; Lami, G. An Automatic Quality Evaluation for Natural Language Requirements. Istituto di Elaborazione dell'Informazione del C.N.R – Pisa, Italy, 2001.

[Fagan, 1976] Fagan, M. E. Design and Code Inspections to Reduce Errors in Program Development. **IBM Systems Journal**, Riverton, NJ, v.15, n.3, p.182-211, 1976.

[Fagan, 1986] Fagan, M. E. Advances in Software Inspections. **IEEE Transactions Software Engineering**, New York, v.12, n.7, p.744-751, 1986.

[Faulk, 1997] Faulk, S. R. Software Requirements: A Tutorial. In: Thayer, R. H.; Dorfman, M. **Software Requirements Engineering.** 2nd ed. Wiley-IEEE Computer Society Press, 1997, 549 p. cap3., p.128-149.

[Firesmith, 2003] Firesmith, D. Specifying Good Requirements. **Journal of Object Technology**, v. 2, n. 4, p. 77-87, July/Aug. 2003.

[Firesmith, 2005] Firesmith, D. Systems Requirements Specification (SRS) Inspection Checklist, 2005. Disponível em: <http://www.donald-firesmith.com/Components/WorkProducts/RequirementsSet/SystemRequirementsSpecification/SystemRequirementsSpecificationInspectionChecklist.html> Acesso em: 4 julho 2005.

[FoxPro Wiki, 2000] Visual FoxPro Wiki. Software Requirements Specification Checklist, 2000. Disponível em: <http://fox.wikis.com/wc.dll?Wiki~SoftwareRequirementsSpecificationChecklist~SoftwareEng> Acesso em: 16 maio 2005.

[Goguen & Linde, 1993] Goguen, J.A.; Linde, C. Techniques for Requirements Elicitation. In: **INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING**, 1993, San Diego, California, USA. **Anais...** San Diego, IEEE Xplore, 04-06 Jan. 1993. p. 152-164.

[Hall et al., 2002] Hall, T.; Beecham, S.; Rainer, A. Requirements problems in twelve software companies: an empirical analysis, **IEEE Proceedings Software**, v.149, n.5, p. 153-160. Oct. 2002.

[Hofmann & Lehner, 2001] Hofmann, H. F.; Lehner, F. Requirements Engineering as a Success Factor in Software Projects. **IEEE Software**, v.18, n.4, p. 58-66, July/Aug. 2001.

[Hooks, 1993] Hooks, I. Writing Good Requirements. In: **INTERNATIONAL SYMPOSIUM OF THE INCOSE**, 3., 1993. **Proceedings...** Prepared by the Requirements Working Group of the International Council on Systems Engineering, v.2, 1993. Disponível em: <http://www.incose .org/rwg/writinggoodreqs_hooks.htm>. Acesso em: 20 nov. 2003.

[Huang & Tilley, 2003] Huang, S.; Tilley, S. Towards a documentation maturity model. In: **ANNUAL INTERNATIONAL CONFERENCE ON DOCUMENTATION**, 21., San Francisco, CA, USA. **Anais**… New York : ACM Press, 2003, p. 93-99.

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

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

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

[ISO/IEC 12.207, 1995] Norma ISO 12.207. ”Information technology -Software life cycle processes”.

[ISO/IEC 15.504] Norma ISO 15.504 “Software Process Improvement andassurance standards Capability Determination (SPICE)”.

[Jacobs, 1999] Jacobs, S. Introducing Measurable Quality Requirements: A Case Study. In: **INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING**, 4., 1999, Limerick Ireland. **Anais…** Limerick: Universoty of Limerick, IEEE Computer Society Press, 07-11 June, 1999. p.172-179.

[Jacobson et al., 1992] Jacobson, I. **Object-Oriented Software Engineering - A Use Case Driven Approach**, Addison-Wesley Publish Company, 1992, 528p.

[Jiang et al., 2005] Jiang, L.; Eberlein, A; Far, B. H. Combining Requirements Engineering Techniques - Theory and Case Study. In: **IEEE INTERNATIONAL CONFERENCE AND WORKSHOPS ON THE ENGINEERING OF COMPUTER-BASED SYSTEMS**, 12., p. 105-112, 2005.

[Kar & Bailey, 1996] Kar, P.; Bailey, M. Characteristics of Good Requirements. In: **INTERNATIONAL SYMPOSIUM OF THE INCOSE**, 1996. **Proceddings...** Prepared by the Requirements Working Group of the International Council on Systems Engineering v. 2, 1996. Disponível em: <http://www.incose .org/rwg/goodreqs.html>. Acesso em: 20 nov. 2003.

[Kauppinen et al., 2002] Kauppinen, M. et al. Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice. In: **INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING**, 2002, Essen, Germany. **Anais...** Essen: University of Essen, IEEE Computer Society Press, 09-13 Sept. 2002. p.43.

[Kotonya & Sommerville, 1998] Kotonya, G.; Sommerville, I. **Requirements Engineering: Processes and Techniques.** Chichester, England: John Wiley & Sons Ltd, 1998.

[Laitenberger, 1995] Laitenberger, O. **Perspective-based Reading: Technique, Validation and Research in Future.** ISERN Technical Report ISERN-95-01, University of Kaiserslautern, Germany, 1995.

[Laitenberger, 2002] Laitenberder, O. A Survey of Software Inspections Technologies. **Handbook on Software Engineering and Knowledge Engineering**, v. 2, World Scientific Publishing, 2002. Disponível em: <http://citeseer.ist.psu.edu/cache/ papers/cs/25307/ftp:zSzzSzcs.pitt. eduzSzchangzSzhandbookzSz61b.pdf/a-survey-of-software. pdf>. Acesso em: 24 maio 2004.

[Lamsweerde, 2000] Lamsweerde, A. V. Requirements Engineering in the Year 00: A Research Perspective. The Future of Software Engineering. In: **INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING**, 22., 2000, Limerick, Ireland. **Anais...** Limerick: University of Limerick, ACM Press, 2000. p.5-19.

[Leite, 2001] Leite, J.C.S.P. **Qualidade de Software: Teoria e Prática**, Rocha, A. R. C., Maldonado, J.C. e Weber, K. C. Gerenciando a Qualidade de Software com Base em Requisitos. São Paulo: Prentice Hall, 2001, p. 238-246, cap. 17.

[Maldonado et al., 2001] Maldonado, J.C.; Martiniano, L. A. F.; Dória, E.S.; Fabbri, S.C.P.F.; Mendonça, M. "Readers Project: Replication of Experiments –A Case Study Using Requirements Document". In: ProTeM-CC-Project Evaluation Workshop – International Cooperation, CNPq, Rio de Janeiro, RJ, October, pp. 85-117, 2001.

[Martin et al., 2002] Martin, S.; Aurum, A.; Jeffery, R.; Paech, B. Requirements Engineering Process Models in Practice. In: **AUSTRALIAN WORKSHOP ON REQUIREMENTS ENGINEERING**, 7., 2002, Melbourne, Australia. **Anais...** Melbourne, 2002. p. 141-155. Disponíevel em: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-69/paper10.pdf>. Acesso em: 28 abr. 2004.

[Marucci, 2002] Marucci, R. A. **Definição de uma Estratégia de Inspeção para um processo de desenvolvimento de Software Orientado a Objetos**, 2002. Dissertação (Mestrado em Ciência da Computação) - Departamento de Computação, UFSCar, São Carlos.

[May, 1998] May, L.J. Major Causes of Software Project Failures. **CROSSTALK The Journal of Defense Software Engineering**, 1988. Disponível em: <http://www.stsc.hill.af.mil/crosstalk/1998/07/index.html> Acesso em: 16 maio 2005.

[Mosley, 1997] Mosley, D. J. Requirements Checklist. Client-Server Software Testing (CSST)Technologies, 1997. Disponível em: <http://www.csst-technologies.com/ reqchklL.htm> Acesso em: 30 maio 2005.

[Nuseibeh & Easterbrook, 2000] Nuseibeh, B.; Easterbrook, S. Requirements Engineering: A Roadmap. The Future of Software Engineering. In: **INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING**, 22., 2000, Limerick, Ireland. **Anais...** Limerick: University of Limerick, ACM Press, 2000. p.35-46.

[Oriel, 1999] Oriel, J. **Guide for Specification Writing for U.S. Government Engineers.** Naval Air Warfare Center Training Systems Division (NAWCTSD), Orlando, FL, 1999. Provided by Training Systems Division at NAVAIR, US. Disponível em: <http://www.ntsc.navy.mil/Resources/ Library/Acqguide/specgide.doc>. Acesso em: 28 jan. 2004.

[Pagliuso et al., 2003] Pagliuso, P. B. B.; Tambascia, C. A.; Villas-Boas, A.; Freitas, M. E.; Levantezi, M. F. V. GVR - Guia de Validação de Requisitos baseados nas técnicas PBR e ad hoc resultante de um estudo de caso no CPqD. In: **WORKSHOP EM ENGENHARIA DE REQUISITOS**, 3., Piracicaba, SP, Brasil. **Anais**... Piracicaba, p. 183-197, 2003.

[Paulk, 1995] Paulk, M.C.; Weber, C.V.; Curtis, B.; Chrissis, M.B. The **Capabilty Maturity Model: Guidelines for improving the software process.** 1st ed., Massachusetts: Addison-Wesley Publishing Company, 1995. 464 p.

[Porter et al.,1995] Porter, A.A.; Votta, L.G.; Basili, V.R. Comparing Detection Methods for Software Requirements Inspection: A Replicated Experiment. **IEEE Transactions on Software Engineering**, v.21, n.6, p.563-575, 1995.

[Pozgaj et al., 2003] Pozgaj, Z.; Sertic, H.; Boban, M. Effective requirement specification as a precondition for successful software development project. In: **INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY INTERFACES**, 25., p.669-674, 2003.

[Prakash et al., 2004] Prakash, S.; Aurum, A.; Cox, K. Requirements Engineering Practice in Pharmaceutical and Healthcare Manufacturing. In: **SOFTWARE ENGINEERING CONFERENCE**, 11., Busan, S. Korea, p. 402-409, 2004.

[Pressman & Associates, 2001] Pressman, R.S. & Associates. Software Requirements Specification Checklist, 2001. Disponível em: <http://www.rspa.com/checklists/reqspec.html> Acesso em: 30 maio 2005.

[Sawyer et al., 1999] Sawyer, P.; Sommerville, I.; Viller, S. Capturing the Benefits of Requirements Engineering. **IEEE Software**, v.16, n.2, p.78-85, Mar./Apr. 1999.

[Sawyer & Kotonya, 2001] Sawyer, P.; Kotonya, G. Chapter 2: Software Requirements. In: **Guide to the Software Engineering Body of Knowledge, SWEBOK.** A Project of the Software Engineering Coordinating Committee. May 2001. Disponível em: <http://www.swebok.org/stoneman/ trial_1_00.htm>. Acesso em: 16 maio 2004.

[Scharer , 1990] Scharer, L. Pinpointing requirements. In **SYSTEM AND SOFTWARE REQUIREMENTS ENGINEERING**. IEEE Computer Society Press, 1990, p. 17-22.

[Schneider et al., 1992] Schneider, G.M.; Martin, J. ; Tsai, W.T. An Experimental Study of Fault Detection in User Requirements Documents. **ACM Transactions on Software Engineering and Methodology**, v.1, n.2, p.188-204, Apr. 1992.

[Shull et al., 2000] Shull, F.; Rus, I.; Basili, V. R. How Perspective-Based Reading Can Improve Requirements Inspections. **IEEE Computer Society**, v.33, n.7, p.73-79, 2000.

[Sommerville, 1995] Sommerville, I. **Software Engineering.** 5th ed. Wokingham, England: Addison-Wesley, 1995.

[Sommerville & Sawyer, 1997] Sommerville, I.; Sawyer, P. **Requirements Engineering:** A good practice guide. 1st ed. New York: John Wiley & Sons, 1997.

[Sommerville, 2001] Sommerville, I. **Software Engineering.** 6th ed. Harlow, England: Addison-Wesley, 2001. 693p.

[Tran, 1999] Tran, E. **Requirements & Specifications.** Carnegie Mellon University. Electrical and Computer Engineering Department. 18-849b Dependable Embedded Systems. Spring 1999. Disponível em: <http://www.ece.cmu.edu/~koopman/des_s99/ requirements_specs/index.html>. Acesso em: 27 nov. 2003.

[Travassos et al., 1999a] Travassos, G. H.; Shull, F.; Carver, J.; Basili, V.R. Reading Techniques for OO Design Inspections. In: **ANNUAL SOFTWARE ENGINEERING WORKSHOP**, 24., 1999, GreenBelt, USA. **Anais...** GreenBelt: GSFC (Goddard Space Flight Center's), 1999.

[Travassos et al., 1999b] Travassos, G. H.; Shull, F.; Fredericks, M.; Basili, V. R. Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality. In: **CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, SYSTEMS, LANGUAGES, AND APLICATIONS**, 14., 1999, Denver, Colorado, USA. **Anais...** Colorado: Colorado Convention Center, ACM Press, 1999. p.47-56.

[Travassos et al., 2000] Travassos, G.H.; Shull, F.; Carver, J. A Family of Reading Techniques for OO Design Inspections. In: **WORKSHOP DE QUALIDADE DE SOFTWARE**, 7., 2000, João Pessoa, Brasil. **Anais...** João Pessoa: SBC, 2000, p. 225-237.

[Travassos et al., 2002] Travassos, G. H.; Shull, F.; Carver, J; Basili, V. R. **Reading Techniques for OO Design Inspections**, 2002, 56 p. Technical Report CS-TR-4353, UMIACS-TR-2002-33, University of Maryland, Maryland. Disponível em: <http://www.cs.umd.edu/Library/TRs/CS-TR-4353/CS-TR-4353.pdf>. Acesso em: 10 maio 2004.

[Volere, 2004] Volere Requirements Specification Template, 10th ed., 2004. Disponível em: <http://www.volere.co.uk/template.htm> Acesso em: 01 abril 2005.

[Wiegers, 1994] Wiegers, K. E. Creating a Software Engineering Culture. **Software Development magazine**, v.2, n.7, 1994.

[Wiegers, 1995] Wiegers, K. E. Improving Quality Through Software Inspections. **Software Development magazine**, v.3, n.4, 1995.

[Wiegers, 1999a] Wiegers, K. E. Writing Quality Requirements. **Software Development magazine**, v.7, n.5, p. 44-48, 1999.

[Wiegers, 1999b] Wiegers, K. E. First Things First: Prioritizing Requirements. **Software Development magazine**, v.7, n.9, p. 24-30, 1999.

[Wilson, 1997] Wilson, W. M. Writing Effective Requirements Specifications. In: **ANNUAL SOFTWARE TECHNOLOGY CONFERENCE**, 9., 1997, Salt Lake City, Utah. **Anais...** Disponível em: <http://satc.gsfc.
nasa.gov/support/STC_APR97/write/writert.PDF>. Acesso em 04 mar. 2004.

[Young, 2002] Young, R.R. Recommended Requirements Gathering Practices. **CROSSTALK The Journal of Defense Software Engineering**, Apr. 2002. Disponível em: <http://www.
stsc.hill.af.mil/crosstalk/2002/04/young.html>
Acesso em: 09 fev. 2004.

ANEXO 1

Neste anexo é apresentado o DR do *Parking Garage* (PG) [Maldonado et al., 2001]:

Documento de Requisitos para um Sistema de Controle de Estacionamento

1 Introdução

1.1 Propósito

Este documento descreve os requisitos de software para um sistema de controle de estacionamento (PGCS). Esta especificação está planejada para o projetista, para o desenvolvedor e para o responsável pela manutenção do PGCS.

1.2 Escopo

A função do PGCS é controlar e supervisionar as entradas e saídas de um estacionamento. O sistema permite ou rejeita entradas no estacionamento de acordo com o número de vagas disponíveis.

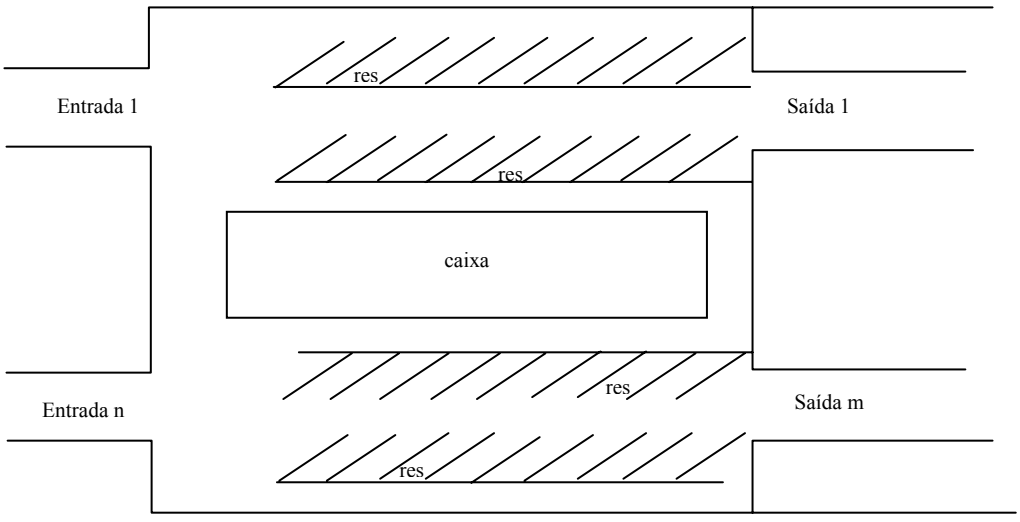
1.3 Organização do Documento

O resto deste documento é organizado como segue: haverá algumas definições importantes na próxima subseção. O capítulo 2 contém uma descrição geral do PGCS. O capítulo 3 identifica os requisitos funcionais específicos, as interfaces externas e os requisitos de desempenho do PGCS.

1.4 Definições

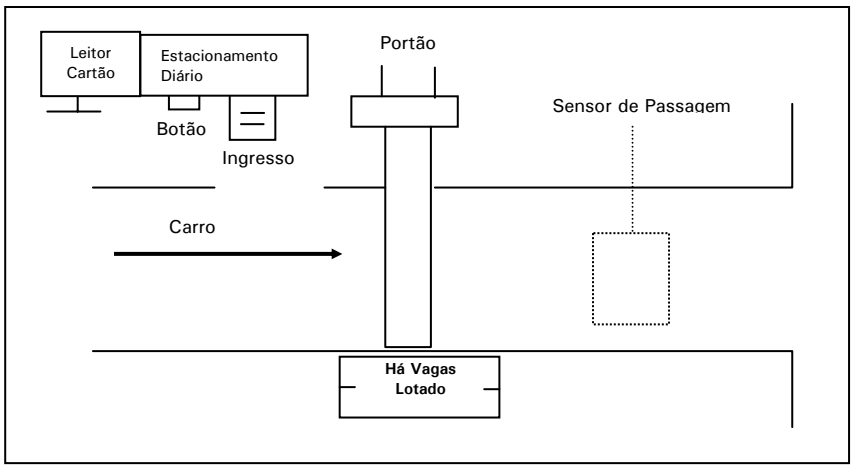
- Estacionamento

Consiste de **n** entradas e **m** saídas. Existem **k** vagas para estacionamento, das quais **r** são reservadas. O número máximo de vagas é 1000.



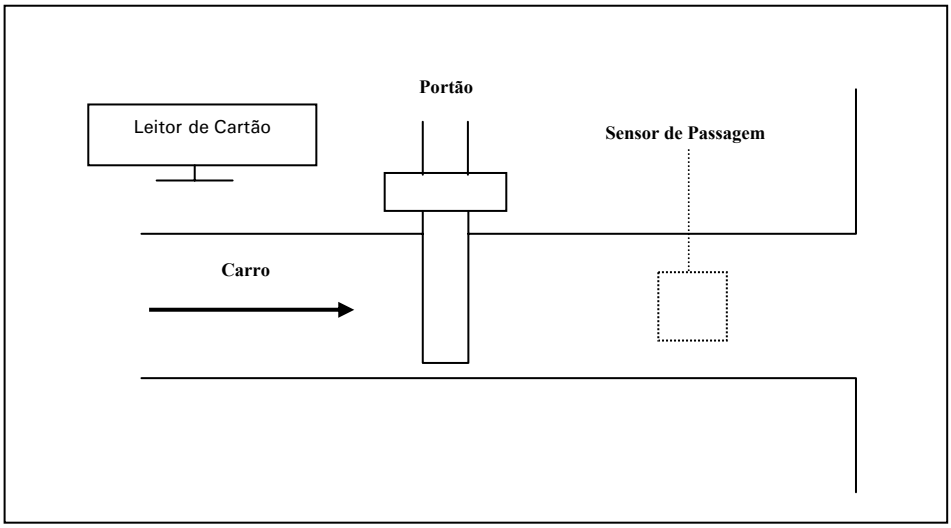
- Entrada

Uma entrada consiste em um portão, um painel de *status* que indica se existem vagas disponíveis, uma máquina de ingresso e um sensor de passagem. A máquina de ingresso consiste de um botão de pedido, uma unidade para a produção dos ingressos e um leitor de cartão.



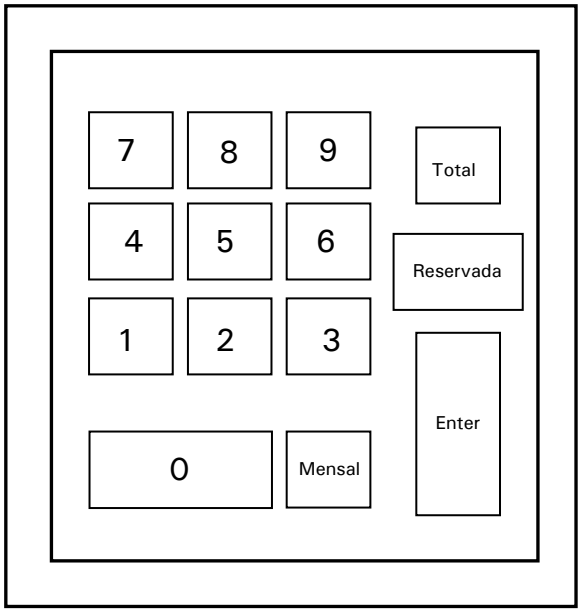
- Saída

A saída consiste em um portão, um leitor de cartão e um sensor de passagem que está atrás do portão.



- Unidade de Controle

A unidade de controle consiste de um teclado numérico.



Descrição geral

2.1 Descrição

Os seguintes cenários apresentam uma descrição sucinta da funcionalidade do PGCS:

- Entrada de motorista sem uma vaga de estacionamento reservada:
  1. Um motorista pressiona o botão da máquina de ingresso;
  2. Se o estacionamento estiver lotado, nada acontece (painel de status indica "lotado");
  3. Caso contrário, a máquina de ingresso escreve o horário no ingresso e o entrega para o motorista;
  4. O portão abre quando o motorista pegar o ingresso;
  5. O motorista entra no estacionamento;
  6. Depois da passagem do carro pelo loop de indução, o portão é fechado;
  7. O motorista estaciona o carro e deixa o estacionamento.

- Motorista com uma vaga reservada:
  1. O motorista insere o seu ingresso mensal no leitor de cartão da máquina de ingresso;
  2. A máquina de ingresso confere se é um ingresso mensal válido;
  3. Se for um ingresso mensal válido, o portão abre e o motorista entra no estacionamento;
  4. Depois da passagem do carro pelo sensor de passagem, o portão é fechado;
  5. O motorista estaciona o carro em uma vaga e deixa o estacionamento.

- Pagamento do estacionamento por motoristas sem uma vaga reservada:
  1. O motorista paga a taxa no caixa;
  2. O caixa notifica o sistema do número do ingresso que foi pago.

- Saída
  1. O motorista volta ao carro e vai até uma estação de saída;
  2. O motorista insere o ingresso no leitor de cartão;
  3. O leitor confere o ingresso através de seu número e verifica se a taxa foi paga nos últimos 15 minutos ou se foi inserido um ingresso mensal válido;

4. Em caso negativo, nada acontece. O motorista tem que retornar ao caixa;
5. Senão, o portão abre;
6. Depois da passagem de carro, o portão é fechado.

- Manutenção e teste do sistema

1. Para testar e manter o sistema é possível entrar o número total de vagas e o número de vagas reservadas com a ajuda da unidade de controle.

- Ingressos

1. Os ingressos mensais e diários serão identificados por números e um código que o leitor de ingresso possa interpretar para determinar se o ingresso é mensal ou diário. Esta funcionalidade está fora do contexto do sistema. O sistema receberá uma mensagem do leitor de ingresso que notifica se um ingresso mensal ou diário foi inserido e o número do ingresso.
2. Não é necessário que os números de ingresso sejam seqüenciais.

2.2 Perspectiva de produto

O software é um sistema embarcado. Serão descritas as características dos dispositivos. Em cada entrada, o sistema de software deve controlar:

- uma máquina de ingresso
- um portão
- um leitor de cartão
- um sensor de passagem
- um painel de status

Para cada saída:

- um leitor de ingresso
- um portão
- um sensor de passagem
- uma unidade de controle
- um teclado complementar numérico

A seguir, descrevemos o comportamento do sistema. O termo "automaticamente" descreve o comportamento de um dispositivo que é realizado sem controle do sistema de software. A máquina de ingresso imprime automaticamente o horário no ingresso se um ingresso é fornecido. O leitor de cartão lê automaticamente um cartão inserido. O leitor de ingresso identifica automaticamente os ingressos mensais.

O loop de indução pode estar em dois estados: um carro está presente no loop ou não. Após as mudanças de estado, um sinal será enviado ao PGCS. Os portões têm dois estados: aberto e fechado. A mudança de estado leva algum tempo. O painel de status mostra dois estados: lotado e com vagas, de acordo com o número de vagas disponíveis no estacionamento. O painel de status é irrelevante para os motoristas que possuem um ingresso mensal.

Com o teclado numérico da unidade de controle há a possibilidade de entrar o número de total de vagas reservadas. O caixa não é controlado pelo sistema de software, mas notifica o sistema do número de um ingresso pago.

2.3 Funções do produto

O sistema de software deve controlar o painel de status, os portões, as máquinas de ingresso e os leitores de ingresso. Se um ingresso mensal válido é inserido no leitor de cartão de uma máquina de ingresso, o portão deve abrir. Se houver vagas disponíveis e o botão de pedido for pressionado, o motorista deve adquirir um ingresso e o portão deve ser aberto.

A inserção de um ingresso não reservado em um leitor de cartão deve abrir o portão de saída se a taxa de estacionamento foi paga nos últimos 15 minutos. A inserção de um ingresso mensal válido no leitor de cartão deve abrir o portão de saída. Os portões devem ser fechados depois que o carro passar pelo sensor de indução.

O painel de status deve mostrar o estado atual de ocupação do estacionamento. Para testes e manutenção do sistema, há a possibilidade de entrar o número de total vagas e vagas reservadas com ajuda da unidade de controle. Ingressos mensais podem ser reservados no caixa. O número de vagas reservadas não deve ser superior a 40% de k. O caixa não é parte do sistema de software.

2.4 Características do usuário

Os usuários do sistema (os motoristas) não devem requerer treinamento especial.

2.5 Suposições sobre o estacionamento

- Toda vaga do estacionamento pode ser alcançada de qualquer entrada;
- Toda saída pode ser alcançada de cada vaga de estacionamento;
- Nenhuma entrada é conversível a saída e vice-versa;
- As vagas não são numeradas;
- Existem ingressos mensais para reservar vagas de estacionamento;
- Situações de emergência (por exemplo, fogo) não serão consideradas;
- Os ingressos mensais para as vagas reservadas estão disponíveis no caixa.
- O caixa controla o número de ingressos mensais para vagas reservadas.
- Quando um ingresso mensal for vendido, o caixa notifica o sistema do número do ingresso. O sistema reduzirá de um o número de vagas não reservadas e será responsável por aumentar o número de vagas não reservadas quando o ingresso expirar (em 30 dias).

3. Requisitos

3.1 Requisitos funcionais

Esta é uma lista de requisitos funcionais que o sistema deve satisfazer. Os requisitos funcionais são apresentados do seguinte modo:

- Descrição: uma descrição da exigência específica (opcional);
- Entradas: uma descrição das entradas que o sistema de software recebe;
- Processamento: uma descrição do que o sistema de software deve fazer com as entradas;
- Saídas: uma descrição da resposta ou novo estado do sistema de software.

Exigência Funcional 1: Objetos de Dados
No software, existem os seguintes objetos de dados:
k: número máximo de vagas disponíveis no estacionamento;
r: número de vagas reservadas no estacionamento;
a = k – r: número de vagas não reservadas e disponíveis;
o: número de vagas não reservadas e ocupadas.

3.1.1 Requisitos gerais

Requisito Funcional 1

- Descrição: O PGCS deve controlar as entradas e saídas de um estacionamento.

Requisito Funcional 2

- Descrição: O PGCS tem que garantir que não mais que **k** carros estão no estacionamento.

Requisito Funcional 3

- Descrição: O valor *default* para **k** é 1000.

Requisito Funcional 4
- Descrição: **k** é dividido em **r** vagas reservadas e **a** vagas não reservadas.

Requisito Funcional 5
- Descrição: o PGCS deve apoiar **n** entradas e **m** saídas, eventualmente simultâneas.

Requisito Funcional 6
- Descrição: Se o sensor de passagem é cruzado, o portão deve fechar.
- Entradas: sensor de passagem vai do estado "carro presente" para o estado "carro não presente"
- Saída: o portão fecha

Requisito Funcional 7
- Descrição: ingressos mensais são válidos durante 30 dias.

3.1.2 Requisitos de atualização

Requisito Funcional 8
- Descrição: a notificação do caixa sobre a compra de um novo ingresso mensal mantém histórico deste ingresso pelo seu número e aumenta de um o valor de **r**.
- Entrada: o caixa entra o número de ingresso mensal e pressiona o botão "mensal" no painel de controle.
- Processamento: Verifique se '**r** + 1 ≤ 0.4 *** k'** e se '**a - o** ≥ 1'. Nesse caso, decremente o valor de **r** por 1 e armazene o novo ingresso mensal. Caso contrário, produza uma mensagem de erro.
- Saídas: novo valor de **r** e de **a**, ou uma mensagem de erro

Requisito Funcional 9
- Descrição: a notificação de caixa para o sistema de que um ingresso foi pago.
- Entradas: caixa entra o número de ingresso e pressiona a tecla "Enter"
- Processamento: o sistema mantém histórico do número de ingresso e tempo pago, conferindo as saídas.

Requisito Funcional 10
- Descrição: a unidade de controle pode fixar um valor novo de **r**.
- Entrada: entrada de um número e pressão do botão "Reservado" e "Enter" na unidade de controle
- Processamento: fixe o valor novo de **r**
- Saída: novo valor de **r**

Requisito Funcional 11
- Descrição: o número total de vagas pode ser escrito na unidade de controle
- Entradas: entrada de um número e pressão do botão "Total" e "Enter" na unidade de controle
- Processamento: verifique se **k** é maior que '**r** + **o**' e aquele '**k** * 0.4 ≥ **r**'. Assume o novo valor de **k**.
- Saídas: novo valor de **k**

Requisito Funcional 12
- Descrição: sistema deve liberar as vagas reservadas quando os ingressos mensais expiram.
- Processamento: às 12:01 AM o sistema deve conferir quantos ingressos foram comprados a 31 dias atrás e reduzir o valor de **r** por este número.
- Saída: novos valores para **r** e **a**.

3.1.3 Requisitos de entrada

Estes requisitos caracterizam a exigência para uma entrada.

Requisito Funcional 13

- Descrição: o painel de status deve representar o estado de ocupação das vagas não reservadas. Deve exibir ‘vagas' se houver uma vaga não reservada disponível naquele momento. Deveria exibir ‘lotado’, se não há nenhuma vaga não reservada disponível naquele momento.

Requisito Funcional 14

- Descrição: todo motorista com um ingresso mensal deverá inserir o ingresso no leitor de ingresso. Se este for válido, o portão abrirá. Um ingresso reservado válido sempre permitirá entrada.
- Entrada: motorista insere ingresso mensal.
- Processamento: o número de ingresso será analisado pelo sistema para determinar se a data de sua compra foi há 30 dias ou menos antes da data atual. Em caso afirmativo, o portão abrirá.
- Saídas: o portão é aberto.

Requisito Funcional 15

- Descrição: todo motorista que usa uma vaga não reservada, só deve adquirir um ingresso se houver pelo menos uma vaga disponível.
- Entrada: motorista aperta o botão uma vez.
- Processamento: verifique se há uma vaga disponível, isto é, se ‘**a - o** > 0’.
- Saída: forneça um ingresso ao motorista se houver uma vaga disponível e abre o portão.

Requisito Funcional 16

- Descrição: diminua o número de vagas disponíveis se o motorista entrar no estacionamento
- Entradas: conclusão com sucesso da exigência 16 e carro atravessa o loop de indução
- Processamento: incrementa o valor de **o** por 1.

Requisito Funcional 17

- Descrição: será dado no máximo um ingresso de entrada a cada motorista
- Entradas: pressão do botão enquanto o portão estiver aberto.
- Processamento e Saídas: Pedidos de ingresso enquanto o portão estiver aberto serão ignorados.

Requisito Funcional 18

- Descrição: se botão é pressionado e não há nenhuma vaga não reservada, nada acontece.
- Entrada: motorista aperta o botão e ‘a–o = 0’.
- Processamento: nada acontece

Requisito Funcional 19

- Descrição: se mais de um carro quer entrar no estacionamento por entradas diferentes, o PGCS tem controlar todos os eventos na ordem eles acontecem.
- Entrada: vários motoristas apertam o botão de pedido de ingresso antes que qualquer ingresso de entrada seja emitido.
- Processamento: os eventos devem ser controlados na ordem em que eles acontecem e só os motoristas com vagas disponíveis terão entrada permitida.
- Saída: habilite entrada aos vários motoristas

3.1.4 Requisitos de saída

Estes requisitos caracterizam uma saída.

Requisito Funcional 20

- Descrição: um carro chega à saída e o motorista insere um ingresso mensal no leitor de cartão. Se o ingresso é válido, o portão abre.
- Entrada: motorista põe um ingresso mensal no leitor de cartão.
- Processamento: verifique o ingresso no sistema e determine se a data de compra foi 30 dias ou menos antes da data atual. Em caso afirmativo, o ingresso é válido: abra o portão.
- Saídas: se o ingresso for válido, portão abre. Senão, nada acontece.

Requisito Funcional 21

- Descrição: Motorista com um ingresso não reservado chega à saída e põe o ingresso no leitor de cartão. Se o ingresso for válido, o portão abre.
- Entrada: Motorista põe um ingresso no leitor de ingresso.
- Processamento: o sistema verifica o ingresso através de seu número. Se o ingresso foi pago nos últimos 15 minutos então é válido. Se o ingresso é válido, o portão abre. Uma vez o carro atravessou o sensor de passagem, o valor de **o** é reduzido de um. Se o ingresso é inválido, nada acontece.
- Saída: se o ingresso é válido, o portão abre. Senão, nada acontece.

Requisito Funcional 22

- Descrição: se vários carros deixam o estacionamento ao mesmo tempo, o PGCS tem que controlar os eventos em ordem.

3.2 Requisitos de interface externas

O PGCS tem que prover uma interface para obter mensagens do teclado numérico usado pelo caixa.

3.2.1 Interface com o usuário

Além da unidade de controle, não há nenhuma necessidade por uma interface de usuário.

3.2.2 Interfaces de hardware

Devem existir interfaces de hardware com as máquinas de ingresso, os leitores de cartão, os portões, os sensores de passagem e a unidade de controle. O PGCS receberá e enviará sinais para estes dispositivos.

3.3 Requisitos de desempenho

Requisito de Desempenho 1

- Descrição: depois que um carro passa o sensor de passagem, o portão tem que fechar dentro de 5 segundos.

Requisito de Desempenho 2

- Descrição: se um motorista pede um ingresso e existem vagas disponíveis, ele deve adquirir o ingresso dentro de 3 segundos.

Requisito de Desempenho 3

- Descrição: se um portão abrir, deve permanecer aberto no máximo 20 segundos, a menos que um carro esteja no sensor de passagem.

Requisito de Desempenho 4

- Descrição: só um carro deve atravessar o portão a cada vez.

Requisito de Desempenho 5

- Descrição: a compra de um ingresso mensal altera os valores de **a** e **r** dentro de 15 segundos.

Requisito de Desempenho 6

- Descrição: alterações em variáveis de estado (entradas ou saídas) devem acontecer dentro de 5 segundos.

Requisito de Desempenho 7

- Descrição: para cada carro que entra no estacionamento há uma vaga disponível.

3.4 Atributos

3.4.1 Disponibilidade

O sistema tem que estar disponível 24 h/dia. O estacionamento nunca estará fechado.

3.4.2 Segurança

Nenhum ingresso diferente dos ingressos deste estacionamento deve ser aceito pelo leitor de ingresso.

3.4.3 Manutenção

Não Aplicável

3.4.4 Conversões

Não Aplicável

3.4.5 Precaução

Não Aplicável

ANEXO 2

Neste anexo é apresentado o DR do Hotel [Maldonado et al., 2001]:

Sistema para Hotel

A – VISÃO GERAL DO SISTEMA

O sistema para o Hotel Uirapuru consiste basicamente do gerenciamento das estadias de *hóspedes* do hotel, controlando desde a reserva de *acomodações* até o acompanhamento do período de estadia, considerando os diversos tipos de consumo efetuados pelos hóspedes, tais como frigobar, restaurante, lavanderia e telefonemas. O hotel possui 70 apartamentos simples, 30 apartamentos para casal e 10 suítes de luxo, distribuídos pelos diversos andares do hotel. O sistema deve ainda emitir diversos tipos de relatórios e consultas, possibilitando um melhor gerenciamento das *acomodações* oferecidas.

B – REQUISITOS FUNCIONAIS

B1 – Lançamentos diversos

1. O sistema deve permitir a inclusão, alteração e remoção de *hóspedes do hotel*, contendo os seguintes atributos: nome, endereço, cidade onde mora, estado, país, telefone, email, documento de identificação (RG ou CPF para brasileiros e passaporte para estrangeiros), data de nascimento e nome dos pais.

2. O sistema deve permitir a inclusão, alteração e remoção de *itens de consumo*, classificados por diversas categorias (frigobar, restaurante e lavanderia), com os seguintes atributos: código do item, descrição e preço de venda.

3. O sistema deve permitir a inclusão, alteração e remoção de *funcionários* do hotel, com os seguintes atributos: nome, endereço, cidade, estado, telefone e data de nascimento.

4. O sistema deve permitir a inclusão, alteração e remoção dos *tipos de acomodação* oferecidos pelo hotel, com os seguintes atributos: código do tipo de acomodação, descrição do tipo de acomodação, quantidade de unidades total desse tipo de acomodação existente no hotel, preço da *diária*, número de pessoas adultas e número de crianças que esse tipo de acomodação comporta.

5. O sistema deve permitir a inclusão, alteração e remoção das *acomodações* existentes no hotel, com os seguintes atributos: número da acomodação, andar no qual se encontra e código do tipo de acomodação. Para cada tipo de acomodação, podem existir diversas acomodações com números diferentes e localizadas em diversos andares do hotel.

6. O sistema deve permitir a *reserva de acomodação*. Cada reserva possui os seguintes atributos: data e hora de chegada do hóspede, data e hora de saída do hóspede, identificação do hóspede

principal (previamente cadastrado), tipo de acomodação desejada, nomes e idades dos *acompanhantes*, valor da *diária*, taxa de multa a ser cobrada em caso de desistência de última hora (a menos de 12 horas do início previsto de entrada), os dados do cartão de crédito do hóspede e desconto concedido (opcional). A reserva somente deve ser concretizada se houver vagas suficientes para atendê-la. Caso contrário deverá ser mostrada uma mensagem alertando que não há disponibilidade de acomodações para o período indicado. A remoção de reserva somente é permitida sem maiores encargos até 12 horas antes do início previsto para *estadia no hotel*. Após esse período, a remoção da reserva deve alertar o *funcionário* do hotel de que deve ser cobrada a taxa de multa estabelecida durante a reserva.

7. O sistema deve permitir o processamento da *entrada de hóspede* no hotel, com os seguintes atributos: data e hora de chegada do hóspede, data e hora prevista para saída do hóspede, identificação do hóspede principal (previamente cadastrado), número da acomodação utilizada, nomes e idades dos *acompanhantes*, valor da *diária*, funcionário responsável pelo recebimento do hóspede e desconto concedido (opcional). Se tiver sido feita a reserva prévia da acomodação, então, durante a entrada do hóspede, informa-se seu nome e o sistema recupera automaticamente os dados da reserva, que podem ser alterados, se necessário.

8. O sistema deve permitir a inclusão, alteração e remoção de *consumo do hóspede*. Durante a estadia no hotel, diariamente um funcionário do hotel faz a coleta de informações no frigobar para informar ao sistema os itens consumidos. Além disso, pedidos feitos ao restaurante e serviços requisitados à lavanderia são instantaneamente informados ao sistema pelo funcionário responsável, que deve ter uma senha especial de acesso ao sistema. Cada consumo do hóspede possui os seguintes atributos: data do consumo, nome do funcionário responsável, número da acomodação, código dos itens consumidos, quantidades consumidas e valor unitário.

9. O sistema deve permitir o processamento da *saída de hóspede* do hotel, com os seguintes atributos: número da acomodação utilizada, data e hora de saída do hóspede, número de *diárias* cobradas, valor de cada diária, valor dos gastos com telefonemas, e desconto concedido (opcional). O sistema deve automaticamente totalizar os gastos de consumo do hóspede, que foram previamente cadastrados, mostrando os subtotais por categoria (frigobar, restaurante e lavanderia). O sistema deve também apresentar na tela o total a pagar, que é a soma das diárias, acrescentando-se os consumos e os telefonemas e subtraindo-se o desconto, se houver. Também é desejável que o sistema permita ao hóspede dar entrada ao seu processo de *saída do hotel* a partir da televisão de seu apartamento.

10. O sistema deve permitir as seguintes opções de *pagamento da estadia* no hotel: 1) à vista (em dinheiro, cheque ou cartão de crédito); 2) faturado em 30 dias.

11. O sistema deve permitir a quitação de uma fatura paga pelo hóspede, contendo as seguintes informações: número da fatura, data de vencimento, data de pagamento, valor total pago, juros e multa.

B2 – Impressão de diversos tipos de relatórios e consultas

12. O sistema deve permitir a impressão de uma listagem dos hóspedes que estão no hotel no momento, contendo o nome do hóspede principal, nome dos acompanhantes, data de entrada, data prevista para saída e número da acomodação.

13. O sistema deve permitir a impressão de uma listagem das reservas efetuadas para a data atual, contendo o nome do hóspede principal, telefone para contato, tipo de acomodação e data prevista para saída.

14. O sistema deve permitir a impressão de um comprovante de *saída do hóspede*, contendo o nome do hóspede, documento, datas e horários de entrada e saída, número total de diárias, valor total das

diárias, valor total do consumo do hóspede, valor total dos telefonemas, valor do desconto e total a pagar. Nesse mesmo comprovante deve ser mostrada uma lista com os produtos consumidos, contendo a data do consumo, descrição do item de consumo, quantidade consumida, preço unitário e preço total. Ainda nesse comprovante deve constar a forma de pagamento e deve ser reservado um espaço para assinatura do hóspede.

15. O sistema deve permitir ao hóspede imprimir um histórico de suas *estadias no hotel*. Para tal o hóspede deve ter sido previamente cadastrado e deve portar um código de identificação e uma senha. Esse histórico deve conter uma linha para cada estadia do hóspede, contendo as datas de entrada e saída e os totais pagos em cada ocasião.

16. O sistema deve permitir a consulta on-line da ocupação das *acomodações* num certo período. Uma acomodação está ocupada se existem hóspedes utilizando-a no momento. Uma acomodação está disponível se não está ocupada no período e o número de reservas para tal tipo de acomodação no período é inferior ao número total de acomodações existentes para tal tipo. Essa consulta deve mostrar uma linha para cada tipo de acomodação oferecida, constando, em cada uma dessas linhas, o código do tipo de acomodação, a descrição do tipo de acomodação, o número de acomodações existentes, o número de acomodações ocupadas, o número de acomodações reservadas e o número de acomodações disponíveis.

17. O sistema deve permitir a impressão de um relatório resumindo o faturamento do hotel no período (por exemplo, semanal ou quinzenal), contendo, para cada dia do período, um resumo das estadias pagas nesse dia, com sete colunas: diárias, frigobar, restaurante, lavanderia, telefonemas, descontos e total.

18. O sistema deve permitir a impressão diária das faturas a serem enviadas aos hóspedes que optaram pelo faturamento de suas contas. A fatura contém o nome e endereço completo do hóspede, o período de estadia, o total de diárias, o total com demais gastos, o valor do desconto, o total líquido a pagar e a data de vencimento.

19. O sistema deve permitir a impressão de um relatório contendo as faturas em atraso no período (por exemplo, semanal ou quinzenal), contendo, para cada dia do período, o nome do hóspede, a data de vencimento e o valor devido pelo hóspede

C – REQUISITOS NÃO FUNCIONAIS

C1. Confiabilidade

20. O sistema deve ter capacidade para recuperar os dados perdidos da última operação que realizou em caso de falha.

21. O sistema deve fornecer facilidades para a realização de *backups* dos arquivos do sistema.

22. O sistema deve possuir senhas de acesso e identificação para diferentes tipos de usuários: administrador do sistema, funcionários do hotel e clientes que têm acesso ao sistema no hotel (em quiosques especiais).

C2. Eficiência

23. O sistema deve responder a consultas on-line em menos de 5 segundos.

24. O sistema deve iniciar a impressão de relatórios solicitados dentro de no máximo 20 segundos após sua requisição.

C3. Portabilidade

25. O sistema deve ser executado em computadores Pentium 200mHz ou superior, com sistema operacional Windows 98 ou acima.

26. O sistema deve ser capaz de armazenar os dados em base de dados Oracle ou Sybase.

Glossário

Termo	Descrição
Acomodação	Quarto, apartamento, suíte ou outro tipo de cômodo do hotel que sirva para acomodar seus hóspedes durante sua *estadia no hotel*
Acompanhante	Pessoa que ficará hospedada na mesma *acomodação* do *hóspede* principal, porém cuja presença faz com que o valor da *diária* possa ser alterado.
Backup	Cópia de segurança ou cópia de salvaguarda
Consumo do hóspede	Refere-se a um ou mais *itens de consumo* que o hóspede utilizou durante sua *estadia no hotel* e que, portanto, deverá pagar pelos mesmos.
Diária	Valor a ser pago pelo *hóspede* referente a um dia completo de *estadia no hotel.*
Entrada do hóspede	Início de uma nova estadia de um *hóspede* no hotel. Significa que o hóspede chegou ao hotel e está ocupando uma acomodação
Estadia no hotel	Período no qual um hóspede permanece hospedado no hotel, desde sua *entrada* até sua *saída*.
Funcionário	Pessoa que trabalha no hotel e faz a verificação do consumo do hóspede
Hóspede	Pessoa que se hospeda no hotel por um determinado período de tempo
Item de consumo	Produto ou serviço oferecido do hotel para seus *hóspedes*, como por exemplo produtos do frigobar, refeições no restaurante, serviços de lavagem ou passagem de roupas, etc.
Pagamento da estadia	Pagamento que o hóspede deve fazer referente aos serviços prestados pelo hotel durante sua *estadia*. Além da cobrança pelo uso de uma *acomodação*, ele deve pagar pelo *consumo* de itens do frigobar, restaurante, serviços de lavanderia e telefonemas realizados.
Reserva de acomodação	Procedimento pelo qual um *tipo de acomodação* fica reservado para um *hóspede*, garantindo que haverá disponibilidade desse tipo de acomodação quando o hóspede fizer sua *entrada* no hotel.
Saída do hóspede	Fim de uma estadia de um *hóspede* no hotel. Significa que o hóspede deixou o hotel e portanto liberou sua acomodação para que outros hóspedes possam nela hospedar-se
Tipo de acomodação	Categorias nas quais as *acomodações* são divididas, para facilitar a gestão das vagas disponíveis e controlar os preços de diárias.

ANEXO 3

Neste anexo é apresentado o DR do *Automatic Teller Machine* (ATM) [Maldonado et al., 2001]:

Documento de Requisitos para um Sistema de Rede de Caixa Eletrônico

1 Introdução

1.1 Propósito
Este documento descreve os requisitos de software para um Sistema de Rede de Caixa Eletrônico (ATM). Esta especificação está planejada para o projetista, para o desenvolvedor e para o responsável pela manutenção do ATM.

1.2 Escopo
A função do ATM é apoiar uma rede bancária automatizada.

1.3 Organização do Documento
O resto deste documento é organizado como segue: haverá algumas definições de termos importantes. A seção 2 contém uma descrição geral do ATM. A seção 3 identifica os requisitos funcionais específicos, as interfaces externas e os requisitos de desempenho do ATM.

1.4 Definições

- Conta

Uma conta única num banco diante da qual transações podem ser aplicadas. Contas podem ser de diversos tipos com pelo menos conta corrente e conta poupança. Um cliente pode ter mais do que uma conta.

- ATM

Uma estação que permite aos clientes entrar suas próprias transações usando cartão bancário como identificação. O ATM interage com o cliente para obter informações de transação, para enviar as informações de transação para o computador central para validação e processamento e para dispensar o cartão para o cliente. Nós assumimos que um ATM não precisa operar independentemente da rede.

- Banco

Uma instituição financeira que mantém as contas para os clientes e que emite autorização de acesso por cartão bancário para contas através de rede ATM.

- Computador do Banco (Central)

Um computador pertencente a um banco que faz interface com a rede ATM e com as estações do caixa do próprio banco. Um banco pode ter sua própria rede interna de computadores para processar contas, mas nós estamos somente interessados com aqueles que interagem com a rede.

- Cartão Bancário

Um cartão designado para um cliente do banco que autoriza o acesso às contas usando uma máquina ATM. Cada cartão contém o código do banco e o número do cartão, codificado de acordo com o padrão nacional de cartões de crédito e cartão bancário. O código do banco identifica unicamente o banco dentro da associação. O número do cartão determina as contas que o cartão pode acessar. Um cartão não necessariamente acessa todas as contas de um cliente. Cada cartão bancário é pertencente a um único cliente, mas múltiplas cópias dele podem existir, assim a possibilidade do uso simultâneo do mesmo cartão de máquinas diferentes deve ser considerada.

- Cliente

O proprietário de uma ou mais contas num banco. Um cliente pode consistir de uma ou mais pessoas ou corporações; o correspondente não é relevante para este problema. Uma mesma pessoa mantendo uma conta em um banco diferente é considerada um cliente diferente.

- Transação

Um único pedido integral para operações sobre as contas de um único cliente. Nós somente especificamos que ATMs devem dispensar caixas, mas não deveríamos evitar a possibilidade de imprimir cheques ou aceitar dinheiro ou cheques. Nós também poderemos querer oferecer a flexibilidade para operar sobre contas de clientes diferentes, embora isso não é solicitado ainda. As operações diferentes devem se equilibrar adequadamente.

2 Descrição Geral

2.1 Perspectiva do Produto

A rede ATM não trabalha independentemente. Ele tem de trabalhar junto com os computadores/software pertencentes aos bancos. Existem interfaces claramente definidas para os diferentes sistemas.

2.2 Funções do Produto

O software deveria apoiar uma rede bancária automatizada. Cada banco oferece seus próprios computadores para manter suas próprias contas e processar transações diante eles. Caixas Eletrônicos comunicam com os computadores do banco. Um Caixa Eletrônico aceita um cartão bancário, interage com o usuário, comunica com o computador do banco para realizar a transação, libera cartão e imprime recibos. O sistema requer apropriada manutenção de registro e fornecimento de segurança. O sistema deve manusear corretamente os acessos concorrentes para a mesma conta. Os bancos oferecerão seus próprios softwares para seus próprios computadores. O custo do sistema compartilhado será dividido para os bancos de acordo com o número de clientes com cartão bancário.

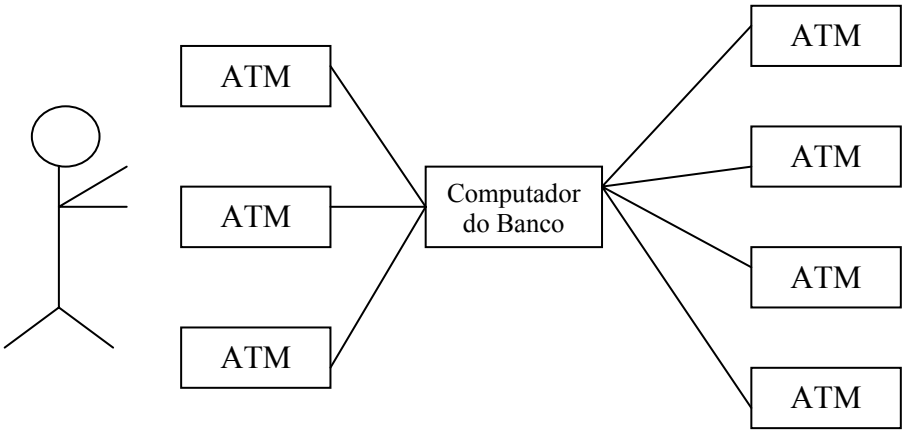


Figura: Rede ATM

2.3 Características do usuário

Existem vários usuários da rede ATM:

- Cliente

O cliente interage com a rede ATM através do ATM. Deve ser muito fácil para eles usar o ATM. Eles deveriam ser auxiliados pelo sistema em todas as formas possíveis.

- Mantenedor

Deveria ser fácil manter o sistema completo. O mantenedor deveria ser a única pessoa permitida a conectar um novo ATM para a rede.

2.4 Abreviações

Em todo esse documento as seguintes abreviações são usadas:

k é o saque máximo por dia e por conta
m é o saque máximo por transação
n é o dinheiro mínimo disponível no ATM para permitir uma transação
t é o capital total no ATM para começar o dia

3 Requisitos específicos

3.1 Requisitos Funcionais

Os requisitos funcionais estão organizados em duas seções: primeiro os requisitos do ATM e segundo os requisitos do banco.

3.1.1 Requisitos para o Caixa Eletrônico

Os requisitos do Caixa Eletrônico estão organizados da seguinte forma: requisitos gerais, requisitos para autorização e requisitos para uma transação.

Geral

Requisito Funcional 1
- Descrição: Inicie os parâmetros **t**, **k**, **m**, **n**.
- Entrada: ATM é iniciado com **t** dólares. **k**, **m**, **n**, são entrados.
- Processamento: Armazene os parâmetros.
- Saída: Parâmetros são estabelecidos.

Requisito Funcional 2
- Descrição: Se nenhum cartão bancário está inserido no ATM, o sistema deveria mostrar uma tela inicial.

Requisito Funcional 3
- Descrição: Se o ATM não tem mais dinheiro, nenhum cartão deveria ser aceito. Uma mensagem de erro é mostrada.
- Entrada: Um cartão é inserido.
- Processamento: A quantidade de dinheiro é menos do que **t**.
- Saída: Mostre uma mensagem de erro. Devolva o cartão bancário.

Autorização
A autorização começa depois de um cliente ter inserido seu cartão no ATM.

Requisito Funcional 4
- Descrição: O ATM tem que checar se o cartão inserido é um cartão bancário válido.
- Entrada: Cliente entra com o cartão bancário.
- Processamento: Cheque se ele é um cartão bancário válido. Ele será válido se:
    1. a informação sobre o cartão pode ser lido.
    2. ele não é expirado.
- Saída: Mostre uma mensagem de erro e devolva o cartão bancário se ele é inválido.

Requisito Funcional 5
- Descrição: Se o cartão bancário é válido, o ATM deveria ler o número serial e o código do banco.
- Entrada: Cartão bancário válido.
- Processamento: Leia o número serial.
- Saída: Iniciar diálogo de autorização.

Requisito Funcional 6
- Descrição: O número serial deveria ser conectado.
- Entrada: Número serial do cartão bancário.
- Processamento: Conecte o número.
- Saída: Atualize o arquivo de registro.

Requisito Funcional 7
- Descrição: Diálogo de autorização: O usuário é solicitado a entrar sua senha. O ATM verifica o código do banco e senha com o computador do banco.
- Entrada: Senha do usuário, código do banco do cartão bancário.
- Processamento: Envie o número serial e senha para o computador do banco, receba resposta do banco.
- Saída: Aceite ou rejeite autorização do banco.

Requisito Funcional 8
- Descrição: Diferentes respostas negativas do computador do banco para diálogo de autorização.
- Entrada: Resposta do banco ou diálogo de autorização:
    - "senha incorreta" se a senha era errada.
    - "código do banco incorreto" se o cartão bancário do banco não é suportado pelo ATM.
    - "conta incorreta" se existe problemas com a conta.
- Processamento: Se o ATM recebe qualquer uma dessas mensagens do computador do banco, o cartão será ejetado e o usuário receberá a mensagem de erro apropriada.
- Saída: Cartão é ejetado e uma mensagem de erro é mostrada.

Requisito Funcional 9
- Descrição: Se a senha e o número serial estão ok, o processo de autorização é finalizado.
- Entrada: O ATM recebe aprovação do computador do banco do processo de autorização.
- Processamento: Finalizar autorização.
- Saída: Começa diálogo de transação.

Requisito Funcional 10
- Descrição: Se o cartão foi inserido mais do que três vezes numa linha de qualquer ATM e a senha era errada a cada vez, o cartão é retido pelo ATM. Uma mensagem será mostrada que o cliente deveria chamar o banco.
- Entrada: Entrar uma senha errada por quatro vezes sucessivos.
- Processamento: Inicie processo de autorização. Resposta do computador do banco é reter o cartão.
- Saída: Mostre mensagem de erro que o cliente deveria chamar o banco.

Funções
Esses são os requisitos para as diferentes funções que o ATM deveria oferecer depois da autorização.

Requisito Funcional 11
- Descrição: O tipo de transação que o ATM oferece é saque.
- Entrada: Autorização completada com sucesso. Entrar a quantidade a sacar.
- Processamento: Quantidade entrada é comparada com **m**.
- Saída: Quantidade de dinheiro a ser dispensada é mostrada. Inicia a seqüência de saque inicial.

Requisito Funcional 12
- Descrição: Seqüência de saque inicial: se o saque é muito alto refaça a transação.
- Entrada: Cliente entrou a quantidade de dinheiro.
- Processamento: Erro se a quantidade é maior do que **m**.
- Saída: Inicie transação ou reinicie o diálogo de transação se a quantidade não está dentro da política de transação pré-definida.

Requisito Funcional 13
- Descrição: Realizar transação.
- Entrada: Seqüência de saque inicial com sucesso.
- Processamento: Envie o pedido para o computador do banco.
- Saída: Espere por resposta do computador do banco.

Requisito Funcional 14
- Descrição: Se a transação foi bem sucedida, o dinheiro é liberado.
- Entrada: ATM recebe a mensagem "transação com êxito" do computador do banco.
- Processamento: ATM imprime o recibo, atualiza **t** e ejeta o cartão. Diálogo: Cliente deveria retirar o cartão.
- Saída: Depois que o cliente retirou o cartão o dinheiro é liberado.

Requisito Funcional 15
- Descrição: Se o dinheiro é liberado, a quantidade é armazenada.
- Entrada: O número de notas de $20 requisitados é liberado para o cliente.
- Processamento: Registre a quantidade de dinheiro contra o número do serial do cartão.
- Saída: Quantidade registrada junto com o número serial. Resposta enviada ao banco para dinheiro liberado.

Requisito Funcional 16
- Descrição: Se a transação não foi bem sucedida, uma mensagem de erro deveria ser mostrada. O cartão deveria ser ejetado.
- Entrada: ATM recebe a mensagem "transação sem êxito" do computador do banco.
- Processamento: ATM mostra uma mensagem de erro. Diálogo: Cliente deveria retirar o cartão.
- Saída: Cartão é ejetado.

3.1.2 Requisitos do computador do banco para o ATM

Autorização
O computador do banco recebe um pedido do ATM para verificar uma conta.

Requisito Funcional 1
- Descrição: O computador do banco checa se o código do banco é válido. Um código do banco é válido se o cartão bancário foi emitido pelo banco.
- Entrada: Pedido do ATM para verificar cartão (Número de serial e senha).

- Processamento: Cheque se o cartão bancário foi emitido pelo banco.
- Saída: Código do banco válido ou inválido.

Requisito Funcional 2
- Descrição: Se não é um código de banco válido, o computador do banco enviará uma mensagem para o ATM.
- Entrada: Código de banco inválido.
- Processamento: Processar mensagem.
- Saída: O computador do banco envia a mensagem "código do banco incorreto" para o ATM.

Requisito Funcional 3
- Descrição: O computador do banco checa se a senha é válida para um cartão bancário válido.
- Entrada: Pedido do ATM para verificar senha.
- Processamento: Checar senha do cliente.
- Saída: Senha válida ou inválida.

Requisito Funcional 4
- Descrição: Se não é uma senha válida, o computador do banco enviará uma mensagem para o ATM.
- Entrada: Senha inválida.
- Processamento: Processar mensagem. Atualizar conta para senha inválida para a conta.
- Saída: O computador do banco envia a mensagem "senha incorreta" para o ATM.

Requisito Funcional 5
- Descrição: Se for um cartão bancário válido e uma senha válida, mas há problemas com a conta, o banco enviará para o ATM uma mensagem que há problemas.
- Entrada: Cartão bancário e senha válida.
- Processamento: Processar mensagem.
- Saída: O banco envia "conta incorreta" para o ATM.

Requisito Funcional 6
- Descrição: Se for um cartão bancário válido, uma senha válida e não há problemas com a conta, o computador do banco enviará para o ATM uma mensagem que tudo está ok.
- Entrada: Cartão bancário, senha e conta válida.
- Processamento: Processar mensagem.
- Saída: Enviar "conta ok" para o ATM.

Transação
O computador do banco recebe um pedido do ATM para processar uma transação.

Requisito Funcional 7
- Descrição: Depois de um pedido o computador do banco processa a transação.
- Entrada: Pedido para processar uma transação sobre uma conta e quantidade **m** para sacar.
- Processamento: Processar transação (junto com o software do banco). Atualizar **k** para quantidade.
- Saída: Se a transação foi bem sucedida, o computador do banco envia a mensagem "transação bem sucedida" para o ATM. Senão, ele envia "transação falhou".

Requisito Funcional 8
- Descrição: Atualizar conta depois que dinheiro é liberado.
- Entrada: Resposta do ATM sobre dinheiro liberado.
- Processamento: Atualizar conta.
- Saída: Novo registro de conta.

Requisito Funcional 9

- Descrição: Cada banco tem um limite **k** para cada conta sobre a quantidade de dinheiro que está disponível através de cartão bancário para cada dia/mês.
- Entrada: Pedido para processar transação.
- Processamento: Checar se a quantidade de dinheiro não excede **k**.
- Saída: Se a quantidade excede o limite, a transação falha.

Requisito Funcional 10

- Descrição: O banco oferece segurança somente para seus próprios computadores e seus próprios softwares.

3.2 Requisitos de interface externas

3.2.1 Interface com o usuário

A interface do ATM deve satisfazer requisitos ergonômicos. A seguir é apenas um exemplo de uma possível interface para o ATM.

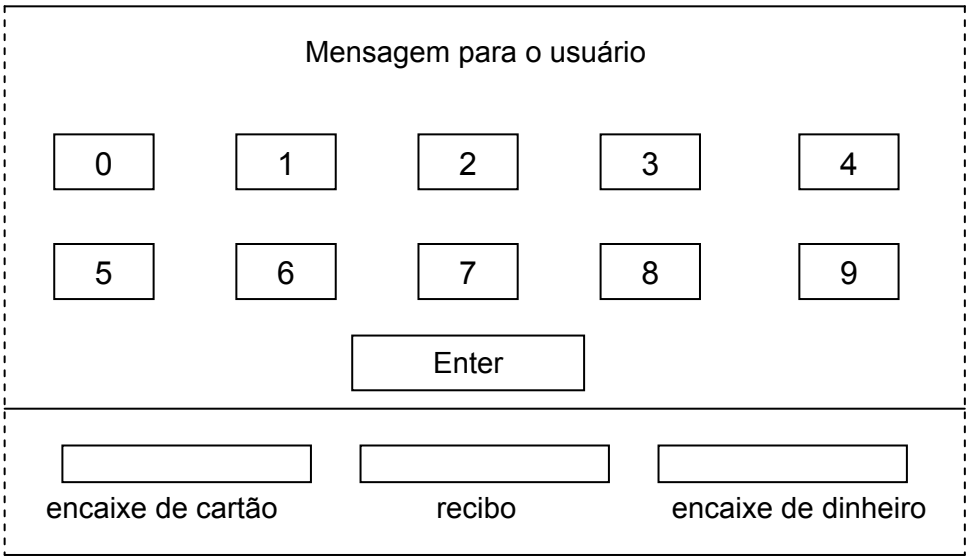


Figura: Formato da interface do ATM

3.2.2 Interfaces de hardware

A rede ATM tem que oferecer interfaces de hardware para:

- Diversas impressoras
- Diversas máquinas ATM (existem várias companhias produzindo máquinas ATM)
- Vários tipos de rede. A especificação exata das interfaces de hardware não faz parte deste documento.

3.2.3 Interfaces de software

A rede ATM tem que oferecer interfaces de software para:

- O software usado por diferentes bancos
- Diferentes softwares de rede

A especificação exata, detalhada das interfaces de software não faz parte deste documento.

3.2.4 Interfaces de comunicação

Não há restrição da rede ATM para um protocolo de rede específico visto que os requisitos de desempenho são satisfeitos.

3.3 Requisitos de desempenho

Requisito de Desempenho 1
- Descrição: Mensagem de erro deveria ser mostrada pelo menos em 30 segundos.

Requisito de Desempenho 2
- Descrição: Se não há resposta do computador do banco depois de um pedido dentro de 2 minutos o cartão é rejeitado com uma mensagem de erro.

Requisito de Desempenho 3
- Descrição: O ATM libera dinheiro se e somente se o saque de uma conta é processado e aceito pelo banco.

Requisito de Desempenho 4
- Descrição: Cada banco pode processar transações de vários ATMs ao mesmo tempo.

3.4 Atributos

3.4.1 Disponibilidade
A rede ATM tem que estar disponível 24 horas por dia.

3.4.2 Segurança
A rede ATM deveria oferecer o máximo de segurança. A fim de fazer isso muito mais transparente existem os seguintes requisitos:

1. Deve ser impossível conectar na rede.

3.4.3 Manutenibilidade
Somente os responsáveis pela manutenção são permitidos a conectar novos ATMs na rede.

3.4.4 Conversões
Não aplicável.

3.5 Outros requisitos

3.5.1 Base de Dados

O ATM deve ser capaz de usar vários formatos de dados de acordo com os formatos de dados que são oferecidos pela base de dados de diferentes bancos. Uma transação deveria ter todas as propriedades de uma transação de banco de dados (Atomicidade, Consistência, Proteção, Estabilidade).

ANEXO 4

Neste anexo encontra-se a descrição da técnica de inspeção PBR-Usuário [Basili et al., 1996a]:

Reading Technique for Use Case

Create a high-level description of the use cases for this system. This will require creating a use case diagram and a set of use case specifications. You will have to identify system participants (actors) and how they interact with the system. Remember to include all of the functionality of the system, including special/contingency conditions. Follow the procedure below to generate the use case diagram and specifications, using the questions provided to identify discrepancies in the requirements.

Inputs: **A set of new requirements**

Output: An use case diagram and a set of user case specifications for the system
A list of defects that should be fixed for the system

1) Read through the requirements once and identify all of the system participants (actor).

   a) Identify the system participants in the requirements. Participants are external entities (people, other systems, etc.) that interact with the system, either initiating functionality or receiving output. Make a list of those in the Form A (a). You may use the following questions to help you identify system participants:

- Which system participants use the system to perform a task?
- Which system participants are needed by the system in order to perform its functions? These functions could be its major functions, or secondary functions such as system maintenance and administration.
- Which system participants send information to the system?
- Which system participants receive information from the system?

Q1.1 Are multiple terms used to describe the same system user in the requirements?
Q1.2 Have necessary system participants been omitted? That is, does the system need to interact with another type of user that is not described?
Q1.3 Is the description of how the system interacts with a participant inconsistent with the description of the participant? Are the requirements unclear or inconsistent about this interaction? Is this situation omitting some important part of the whole functionality?
Q1.4 Is a system participant described in the requirements that do not actually interact with the system?

2) Read through the requirements a second time, identifying the use cases that will compose the system use case diagram. Make a list of those using Form A (c).

a) Identify how each system participant will interact with the system associating the actor with the number of the identified use cases, using Form A (b). Think about how the users will use the system to achieve some goal (e.g. adding information to an existing database, computing a payment, viewing the current status of an account). These user goals will be the use cases of the diagram that must be represented using Form B.

Q2.1 Are there use cases identified for each system participant?
Q2.2 Are the requirements unclear or inconsistent about this interaction? Is this situation omitting some important part of the whole functionality?
Q2.3 Are use cases described that are not necessary to accomplish a desired goal of a particular system participant?
Q2.4 Do the interactions between the system and the user make sense from what you know about the domain?
Q2.5 Do the requirements omit use cases that you think are necessary, based on your domain knowledge or the general description?

3) Create a use case specification to understand the functionality that must be present in the system to accomplish the desired user goals.
   a) For each use case identified in Step 2, create a textual description that describes the functionality represented by the use case, using Form C.

Q3.1 Is there enough information in the requirements to create these descriptions?
Q3.2 Have any necessary functionality information been left out of the requirements document?
Q3.3 Are functionality information described that are not necessary to accomplish the user goal?
Q.3.4 Does the functionality information make sense from what you know about the system?

ANEXO 5

Neste anexo apresenta-se o *checklist* usado na inspeção dos documentos utilizados neste trabalho [Maldonado et al., 2001]:

Checklist

1. **Questões Gerais**
   - Os objetivos do sistema foram definidos?
   - Os requisitos estão claros e não ambíguos?
   - Foi fornecida uma visão geral da funcionalidade do sistema?
   - Foi fornecida uma visão geral das formas de operação do sistema?
   - O software e o hardware necessários foram especificados?
   - Se existe alguma suposição que afete a implementação ela foi declarada?
   - Para cada função, os requisitos foram especificados em termos de entrada, processamento e saída?
   - Todas as funções, dispositivos e restrições estão relacionadas aos objetivos do sistema e vice-versa?

2. **Omissão**
   de Funcionalidade
   - As funções descritas são suficientes para alcançar os objetivos do sistema?
   - As entradas declaradas para as funções são suficientes para que elas sejam executadas?
   - Foram considerados os eventos indesejáveis e as repostas a eles foram especificadas?
   - Foram considerados o estado inicial e os estados especiais (por ex. inicialização do sistema, término anormal)?

   de Desempenho
   - O sistema pode ser testado, analisado ou inspecionado para mostrar que ele satisfaz seus requisitos?
   - Os tipos de dados, unidades, limites e resolução foram especificados?
   - A freqüência e o volume de entrada e saída foram especificados para cada função?

3. **Ambigüidade**
   - Cada requisito foi especificado de forma discreta, não ambígua e testável?
   - Todas as transições do sistema foram especificadas de forma determinística?

4. **Inconsistência**
   - Os requisitos estão consistentes entre si?

5. **Fato Incorreto**
   - As funções especificadas são coerentes com o sistema e com os objetivos a serem alcançados?

6. **Miscelânea**
   - Qualquer tipo de defeito que não as anteriores.

ANEXO 6

Neste anexo apresenta-se a técnica de inspeção TUCCA [Belgamo, 2004]:

Técnica de Leitura AGRT

Leitura para Construção Actor-Goal Reading Technique (A.G.R.T) – Documento de Requisitos

Objetivo: utilizar o documento de requisitos como apoio para criação de um Formulário contendo os atores e respectivos objetivos do sistema a ser desenvolvido.

Entradas para o processo:
1. Um Documento de Requisitos, preferencialmente no padrão IEEE.

I. Leia o Documento de Requisitos para entender a funcionalidade descrita.

ENTRADAS: Seção Funções do Produto do Documento de Requisitos, se existir
Um conjunto de Requisitos Funcionais (RFs).

SAÍDAS: Substantivos candidatos a atores (marcados em azul nos RFs);

Verbos ou descrições de ações candidatos a funcionalidades do sistema (marcados em verde nos RFs);
Restrições ou condições associadas aos substantivos ou verbos (marcadas em amarelo nos RFs).

Para cada Requisito Funcional do Documento de Requisitos faça:

A. Leia o Requisito para entender a funcionalidade que ele descreve.
B. Encontre os substantivos que constam desse Requisito. Sublinhe esses substantivos com uma caneta azul.
C. Encontre os verbos ou descrições de ações, os quais sejam candidatos a funcionalidades realizadas pelo sistema. Sublinhe esses verbos ou descrições de ações com uma caneta verde.
D. Procure descrições de restrições ou condições relacionadas aos substantivos e verbos que você identificou nos dois passos anteriores. Verifique se há limitações ou restrições explícitas na forma como as ações são executadas. Tente observar se foram especificadas quantidades bem definidas em qualquer ponto dos requisitos. Sublinhe essas condições e restrições com uma caneta amarela.
E. Repita os passos B, C, D, para a seção que descreve as principais funcionalidades do sistema caso o Documento de Requisitos possua essa seção.

II. Identifique, com base nas marcações realizadas na etapa I, os possíveis atores e seus respectivos objetivos na utilização do sistema.

ENTRADAS: Seção Funções do Produto marcada, se existir;
Requisitos Funcionais com substantivos, verbos ou descrições de ações, restrições ou condições marcados.
SAÍDAS: Formulário Ator X Objetivo com atores e respectivos objetivos identificados
Relatório de Discrepâncias

Identificação de atores e respectivos objetivos no Documento de Requisitos, mantendo referência ao Documento de Requisitos.

A. Para cada Requisito Funcional, de acordo com as marcações realizadas, identifique possíveis atores e seus respectivos objetivos de utilização do sistema a ser construído. Fique atento para o fato de um substantivo/ator ser seguido por verbos que indiquem uma funcionalidade do sistema que esse ator necessite.

A.1. Em caso de não existir um substantivo candidato a ator para o verbo/objetivo identificado verifique, na Seção Características do Usuário, se existe alguma informação de quem possa ser o ator para relacionar-se com o objetivo.

1. Caso não haja informações suficientes para que se possa identificar os possíveis atores para o objetivo, preencha o Relatório de Discrepâncias relatando o fato e uma nova comunicação com o cliente/usuário do sistema é necessária para que esse problema seja solucionado.

A.2. Caso ambos, ator e objetivo, já estejam identificados no Documento de Requisitos verifique em qual situação abaixo eles podem se encaixar.

A.1.1. Se o ator e o objetivo identificado já estiverem listados no Formulário Ator X Objetivo complete somente a coluna "Referência" com o número do requisito em questão. Marque, no Documento de Requisitos, os requisitos utilizados.

A.1.2. Caso contrário, preencha o Formulário Ator X Objetivo colocando, na coluna "Ator", os atores encontrados, na coluna "Objetivo", os objetivos respectivos a cada um desses atores e, na coluna "Referência", o número do requisito em questão. Marque, no Documento de Requisitos, os requisitos utilizados.

Identificar atores e objetivos da seção do Documento de Requisitos que descreve as principais funcionalidades.

B. Caso a seção que descreve as principais funcionalidades do sistema exista, identifique possíveis atores e seus respectivos objetivos de utilização do sistema a ser construído.

B.1. Se o ator e o objetivo identificado (tendo ele um nome idêntico ou semelhante, mas que corresponda à mesma funcionalidade) já estiver listado no Formulário Ator X Objetivo complete somente a coluna "Referência" com o número da seção em questão.

B.2. Caso contrário, preencha o Formulário Ator X Objetivo colocando, na coluna "Ator", os atores encontrados, na coluna "Objetivo", os objetivos respectivos a cada um desses atores e, na coluna "Referência", o número da seção em questão.

1. Caso o ator encontrado não esteja no Formulário Ator X Objetivo, possivelmente ou esse ator identificado não seja realmente um ator, ou não existe informações suficientes nos Requisitos Funcionais para que o ator esteja relacionado com algum objetivo, ou atores estão sendo identificados por nomes diferentes.

2. Caso o objetivo encontrado não esteja no Formulário Ator X Objetivo, possivelmente ou o objetivo identificado não seja realmente um objetivo ou não existe informações suficientes nos Requisitos Funcionais para que o objetivo seja realizado pelo ator identificado.

III. Utilize o Formulário Ator X Objetivo para identificar e eliminar as possíveis redundâncias ou identificações errôneas feitas anteriormente.

ENTRADAS: Formulário Ator X Objetivo com atores e respectivos objetivos identificados
SAÍDAS: Formulário Ator X Objetivo sem redundâncias
Lista de Objetivos não associados
Relatório de Discrepâncias

Verificar, nos objetivos associados ao mesmo ator, se existem objetivos com nomes diferentes porém sugerindo a mesma funcionalidade.

A. Para cada ator, verifique se existem objetivos associados a ele que estejam sugerindo, pelo seu nome, funcionalidades semelhantes. Como auxílio, releia os requisitos funcionais relacionados a esses objetivos que estão listados na coluna "Referência" do Formulário Ator X Objetivo.

A.1. Caso a funcionalidade seja a mesma, escolha o nome mais apropriado e elimine essa redundância, deixando na lista de objetivos desse ator apenas o nome escolhido e agrupando, na coluna "Referência", todas as referências que estavam associadas aos objetivos redundantes que foram unificados.

1. Preencha o Relatório de Discrepâncias relatando o fato de que objetivos idênticos estão sendo representados por palavras diferentes no Documento de Requisitos.

Eliminar dos atores "sistema" ou "nome do sistema" objetivos que já estejam identificados em outros atores.

B. Verifique, no Formulário Ator X Objetivo, se existe algum ator que esteja sendo referenciado pelo nome "sistema" ou pelo nome atribuído ao sistema (aplicação) em questão. Caso exista, para cada objetivo associado a esses atores que ainda não esteja marcado com uma letra ou sinal "*", faça os passos B1 e B2.

B.1. Selecione esse objetivo e considere-o como objetivo base. Considerando os demais atores do Formulário Ator X Objetivo, verifique em suas listas de objetivos se existe algum objetivo que sugere uma funcionalidade semelhante ao objetivo base, desde que o objetivo não possua uma letra ou um "*".

B.1.1. Caso exista, utilizando as referências que constam da coluna "Referência" do Formulário Ator X Objetivo, certifique-se, pelo Documento de Requisitos, que as funcionalidades sejam realmente as mesmas e, para esses casos, atribua uma letra diferente das já existentes, ao objetivo base e a todos os objetivos que possuam a mesma funcionalidade.

B.1.2. Caso não sejam encontrados nos demais atores objetivos semelhantes ao objetivo base, marque o objetivo base com um "*".

B.2. Considere o subconjunto de objetivos marcado com a mesma letra no passo anterior.

B.2.1. Caso os nomes desses objetivos sejam diferentes, escolha o nome mais apropriado para o objetivo em questão e elimine essa redundância deixando os objetivos desse subconjunto com o mesmo nome.

1. Preencha o Relatório de Discrepâncias relatando o fato de que objetivos idênticos estão sendo representados por palavras diferentes no Documento de Requisitos.

B.2.2. Agrupe as referências, contidas na coluna "Referência" do ator "sistema" e/ou "nome do sistema" com as demais referências de cada um dos objetivos pertencentes ao subconjunto em questão. Por fim, elimine do ator "sistema" ou "nome do sistema" o objetivo base identificado para o subconjunto em questão.

B.3. Verifique se existem objetivos marcados com o sinal "*" para esses dois tipos de atores. Em caso afirmativo, para cada um desses objetivos, verifique se o mesmo pode ser de algum outro ator relacionado no Formulário Ator X Objetivo, fazendo isso para todos atores, utilizando a seguinte pergunta para tomar essa decisão:

- *O objetivo é realmente necessário para esse? É ele quem faz com que a ação seja disparada ou quem efetivamente operacionaliza esse objetivo?*

B.3.1. Caso o objetivo possa ser de algum outro ator, acrescente-o na lista desse ator e transfira as referências associadas a ele para a coluna "Referência" do ator em questão.
B.3.2. Se o objetivo não foi atribuído a nenhum dos atores, coloque-o na Lista de Objetivos Não Associados e elimine-o da lista de objetivos do ator "sistema" e/ou "nome do sistema".

Verificar, nos objetivos associados a atores distintos, se existem objetivos com nomes diferentes porém sugerindo a mesma funcionalidade.

C. A partir de agora, considere somente os atores não nomeados "sistema" e/ou "nome do sistema". Para cada objetivo associado a eles que ainda não esteja marcado com um número ou sinal "*" faça:

C.1. Selecione esse objetivo e considere-o como objetivo base. Considerando os demais atores do Formulário Ator X Objetivo, verifique, em suas listas de objetivos se existe algum objetivo que sugere uma funcionalidade semelhante ao objetivo base, desde que não possua um número ou um "*".

C.1.1. Caso exista, utilizando as referências que constam da coluna "Referência" do Formulário Ator X Objetivo, certifique-se, pelo Documento de Requisitos, que as funcionalidades sejam realmente as mesmas e, para esses casos, atribua um número, diferente dos já existentes, ao objetivo base e a todos os objetivos que possuam a mesma funcionalidade.

C.1.2. Caso não sejam encontrados nos demais atores objetivos semelhantes ao objetivo base, marque o objetivo base com um "*".

C.2. Considere o subconjunto de objetivos marcado com o mesmo número no passo anterior.

C.2.1. Caso os nomes desses objetivos sejam diferentes, escolha o nome mais apropriado para o objetivo em questão e elimine essa redundância deixando os objetivos desse subconjunto com o mesmo nome.

1. Preencha o Relatório de Discrepâncias relatando o fato de que objetivos idênticos estão sendo representados por palavras diferentes no Documento de Requisitos.

IV. Utilize o Formulário Ator X Objetivo e o Documento de Requisitos com o objetivo de verificar se existem requisitos funcionais não utilizados.

ENTRADAS: Documento de Requisitos
SAÍDAS: Relatório de Discrepâncias

Verificar se todos os requisitos funcionais descritos no Documento de Requisitos foram utilizados para a identificação dos objetivos.

A. Verifique, no Documento de Requisitos, se todos os requisitos funcionais foram marcados.

1. Caso algum requisito não tenha sido marcado, preencha o Relatório de Discrepância relatando esse fato, pois alguma informação pode estar em local errado ou pode existir informações desnecessárias para o sistema (aplicação).

Formulário Ator X Objetivo (FAO)

Formulário Ator X Objetivo		
Ator	**Objetivo**	**Referência**

Relatório de Discrepâncias – AGRT

Formulário de Relato de Discrepância para AGRT

Nome do Projeto:
Equipe:
Nome do Integrante:

Início da Atividade: Data: _____ (data)

Tipo de Discrepância (Tipo Disc.):
(O) informação necessária omitida
(A) informação pode ser interpretada de várias maneiras
(II) informação contraditória a respeito do mesmo assunto
(FI) a informação não é correta considerando-se as condições especificadas pelo sistema
(IE) a informação é estranha, isto é, não é necessária no contexto da aplicação
(M) outros tipos de problemas

Preencha a tabela com as discrepâncias encontradas. Descreva a identificação dos requisitos, utilizando os números dos requisitos e números de página se possível:

Disc. #	Tipo Disc.	Referência	Comentários

Fim da Atividade: _____ (hora)

Técnica de Leitura – UCRT

Leitura para construção U.C.R.T – Documento de Requisitos e Formulário de Ator X Objetivo

Objetivo: utilizar o Documento de Requisitos e o Formulário de Ator X Objetivo como apoio para criação de Modelos de Caso de Uso (Diagrama de Casos de Uso e Especificações de Casos de Uso).

Entradas para o processo:

1. Um Documento de Requisitos com substantivos, verbos e restrições marcados.
2. Formulário de Ator X Objetivo preenchido.

I. Utilize o Formulário de Ator X Objetivo para entendimento de cada um dos objetivos dos atores identificados e para a criação dos casos de uso preliminares.

ENTRADAS: Documento de Requisitos marcado.
Formulário de Ator X Objetivo preenchido.
SAÍDAS: Formulário de Casos de Uso Preliminares.

A. Para cada objetivo do Formulário Ator X Objetivo que não esteja ticado com uma marcação (√), verifique se na coluna Referência desse objetivo existe uma referência que corresponde a seção "Funções do Produto" do Documento de Requisitos. Caso exista, preencha o Formulário de Casos de Uso Preliminares atribuindo um número seqüencial na coluna "Número", o objetivo na coluna "Caso de Uso", a(s) referência(s) que estão na coluna "Referência" e o ator que está na coluna "Ator" do Formulário Ator X Objetivo, nas colunas "Referência" e ator do Formulário de Casos de Uso Preliminares, respectivamente. Marque, com um tique, esse objetivo no Formulário Ator X Objetivo.

A.1. Caso o mesmo objetivo seja encontrado em algum outro ator associe ao caso de uso que acabou de ser criado as informações das colunas "Referência" e "Ator" do Formulário Ator X Objetivo nas colunas "Referência" e "Ator" do Formulário de Casos de Uso Preliminares respectivamente. Marque, com um tique, esse objetivo no Formulário Ator X Objetivo.

B. Para cada objetivo ainda não ticado, verifique se existem outros objetivos, independentemente do ator, que possuam o mesmo conjunto de referências na coluna "Referência" do Formulário Ator X Objetivo.

B.1. Caso existam, crie tantos casos de uso quantos forem necessários, avaliando se esses objetivos podem ser agrupados, pois constituem passos de um caso de uso ou dê origem a casos de uso distintos se eles não puderem ser agrupados. No caso de agrupamento, determine um nome apropriado para o caso de uso.

B.1.1. Para cada caso de uso criado, preencha o Formulário de Casos de Uso Preliminares atribuindo um número seqüencial na coluna "Número", o objetivo ou o nome criado na coluna "Caso de Uso", a(s) referência(s) que estão na coluna "Referência" e o(s) ator(es) do Formulário Ator X Objetivo na coluna "Referência" e coluna "Ator" do Formulário de Casos de Uso Preliminares, respectivamente. No caso de agrupamento, todos os atores associados aos objetivos agrupados devem ser associados ao caso de uso. Marque, com um tique no Formulário Ator X Objetivo, os objetivos usados para a criação dos casos de uso.

1. **Preencha o Relatório de Discrepâncias relatando o fato de que o mesmo requisito funcional está fazendo referência a muitas funcionalidades do sistema.**

B.2. Para cada um dos casos de uso criados no passo B.1 verifique se existem outros atores que possuem objetivos idênticos aos que foram usados na criação desses casos de uso, porém não estavam associados ao mesmo conjunto de referências. Em caso

afirmativo, transcreva para as colunas "Referência" e "Ator" do Formulário de Casos de Uso Preliminares as referências e atores do Formulário Ator X Objetivo, para cada ator em que essa situação ocorrer. Marque, com um tique no Formulário Ator X Objetivo, os objetivos usados para a criação dos casos de uso.

C. Para cada objetivo ainda não marcado no Formulário de Ator X Objetivo crie um caso de uso no Formulário de Casos de Uso Preliminares atribuindo um número seqüencial na coluna "Número", o objetivo na coluna "Caso de Uso", as informações que estão na coluna "Referência" e o(s) ator(es) que estão na coluna "Ator" do Formulário Ator X Objetivo, nas colunas "Referência" e ator do Formulário de Casos de Uso Preliminares, respectivamente. Marque, com um tique, esse objetivo no Formulário Ator X Objetivo.

C.1. Para cada um dos casos de uso criados no passo C verifique se existem outros atores que possuem objetivos idênticos. Em caso afirmativo, transcreva para as colunas "Referência" e "Ator" do Formulário de Casos de Uso Preliminares as referências e atores do Formulário Ator X Objetivo, para cada ator em que essa situação ocorrer. Marque, com um tique no Formulário Ator X Objetivo, os objetivos usados para a criação dos casos de uso.

II. Utilize os Casos de Uso Preliminares identificados na etapa I para a criação das especificações e possíveis relacionamentos entre os casos de uso.

ENTRADAS: Documento de Requisitos;
Formulário de Casos de Uso Preliminares identificados;
Template para Especificação de Casos de Uso.

SAÍDAS: Modelo de Caso de Uso (Diagrama de Caso de Uso e Especificação dos Casos de Uso).
Relatório de Discrepância.

A. Verifique, para cada caso de uso do Formulário de Casos de Uso Preliminares, se o mesmo possui como referência apenas o número de uma seção do Documento de Requisitos que contém as principais funcionalidades do sistema. Caso exista, selecione esse caso de uso e considere-o como base.

1. **Caso haja informações suficientes para a especificação do caso de uso base preencha o Relatório de Discrepâncias relatando o fato de que existe muita informação em um local incorreto do Documento de Requisitos.**
2. **Caso não haja informações suficientes para a especificação do caso de uso base preencha o Relatório de Discrepâncias relatando o fato de que essa funcionalidade não foi tratada na seção de requisitos funcionais do Documento de Requisitos.**

B. Seguindo a ordem numérica dos casos de uso preliminares do Formulário de Casos de Uso Preliminares, selecione o próximo caso de uso não especificado (veja marcação na coluna "Especificado") e que possua o menor número de referências associado a ele.

B.1. Para o caso de uso selecionado considere-o como base e verifique se o seu conjunto de referências contém todas referências de outros casos de uso já especificados e, em cada ocorrência desse fato, atribua o número seqüencial do caso de uso base à coluna "Relacionamento" do caso de uso já especificado e vice-versa.

B.2. Preencha o número e o nome do caso de uso, além de uma breve descrição do mesmo no *template* fornecido.

B.3. Identifique, entre os atores relacionados com o caso de uso base, qual(is) ator(es) que efetivamente operacionaliza(m) o caso de uso e aqueles que interagem com o caso de uso fornecendo ou recebendo informações do mesmo, colocando-os, respectivamente, como Ator Operador e Ator Participante no *template* de especificação de casos de uso.

B.3.1. Caso o ator existente seja somente o operador, utilize uma das perguntas abaixo para tomar a decisões especificadas no passo B.3.1.1 ou B.3.1.2.

- *Quem recebe informações do sistema?*

- *Quem envia informações para o sistema?*

B.3.1.1. Se, com base nas questões, descobre-se que o ator operador é o mesmo que o ator participante, coloque-o o nome do mesmo no campo "Ator Participante" do *template* de especificação de casos de uso.

B.3.1.2. Se, com base nas questões, descobre-se que o ator participante não é o mesmo que o ator operador, coloque-o o novo ator no campo "Ator Participante" do *template* de especificação de casos de uso.

B.4. Preencha o campo "Evento Disparador" do *template* com a condição de entrada para que o caso de uso base seja realizado.

B.5. Utilize os requisitos funcionais do Documento de Requisitos que estão referenciados na coluna "Referência" do Formulário de Casos de Uso Preliminares para a criação da especificação do caso de uso.

B.5.1. Verifique se existe mais de um ator participante relacionado ao caso de uso. Em caso afirmativo, verifique se eles podem ser considerados o mesmo ator.

B.5.1.1. Caso eles possam ser o mesmo ator, escolha o nome mais apropriado para o ator e elimine o(s) outro(s) nomes do campo "Ator Participante" do Template de Especificação de Casos de Uso.

1. Preencha o Relatório de Discrepâncias relatando o fato de que um mesmo ator está sendo representando por mais de um ator com nomes diferentes.

B.5.1.2. Caso eles não sejam o mesmo ator verifique se os atores possam ser generalizados e em caso afirmativo, crie o ator genérico e adicione ao *template* de especificação de casos de uso no campo "Ator genérico" o nome do ator genérico.

B.5.2. Determine o curso normal. Em geral, o curso normal está associado com os requisitos funcionais que não possuem restrições marcadas. No entanto, se estes não forem suficientes, verifique também os demais requisitos. Durante a especificação do curso normal, verifique na coluna "Relacionamento" do caso de uso base se a especificação dos casos de uso ali relacionados são utilizados integralmente na especificação do caso de uso base. Em caso afirmativo, essa especificação não precisa ser repetida no caso de uso base e deve-se colocar na coluna "Include" do caso de uso relacionado, no Formulário de Casos de Uso Preliminares, o número do caso de uso base e no caso de uso base, uma chamada ao caso de uso relacionado. Verifique se algum passo do curso normal pode ser considerado uma chamada a algum caso de uso já existente. Em caso positivo, coloque essa chamada com o estereótipo "Include" e coloque o número, no campo "Include" do Formulário de Casos de Uso Preliminares, do caso de uso base no caso de uso sendo chamado.

1. **Caso não haja informações suficientes para a elaboração do curso normal ou requisitos funcionais relacionados não foram utilizados preencha o Relatório de Discrepâncias relatando o problema. Possivelmente o documento de requisitos está incompleto ou existem informações estranhas. Marque a coluna "Especificado" do Formulário de Casos de Uso Preliminares e inicie novamente o passo B da etapa II.**

B.5.3. Determine os cursos alternativos. Faça isso analisando cada passo do curso normal e verificando, nos requisitos relacionados na coluna "Referência", se o passo admite uma alternativa de execução. Durante a especificação do curso alternativo, verifique na coluna "Relacionamento" do caso de uso base se a especificação dos casos de uso ali relacionados são utilizados integralmente na especificação do caso de uso base. Em caso afirmativo, essa especificação não precisa ser repetida no caso de uso base e deve-se colocar na coluna "Include" do caso de uso relacionado, o número do caso de uso base e no caso de uso base, uma chamada ao caso de uso relacionado. Verifique se algum passo do curso alternativo pode ser considerado uma chamada a algum caso de uso já existente. Em caso positivo, verifique se o estereótipo é do tipo "include" e em caso afirmativo coloque o número,

no campo "Include" do Formulário de Casos de Uso Preliminares, do caso de uso base no caso de uso sendo chamado. Em caso do estereótipo ser do tipo "extend" coloque a chamada ao caso de uso com o estereótipo <<extend>>. Marque no campo "Extend" do caso de uso base o número do caso de uso criado.

1. Caso não haja informações suficientes para elaborar os cursos alternativos ou requisitos funcionais relacionados não foram utilizados, preencha o Relatório de Discrepâncias relatando o problema. Possivelmente o documento de requisitos está incompleto ou existem informações estranhas. Marque a coluna "Especificado" do Formulário de Casos de Uso Preliminares e inicie novamente o passo B da etapa II.

B.5.4. Analise todos os cursos alternativos criados no caso de uso base e verifique, para cada nível de identação dos passos especificados, iniciando pelos níveis de identação mais internos, se o passo possui mais de 10 sub-passos. Em caso afirmativo, esse passo deve ser transformado em outro caso de uso, utilizando o *template* de casos de uso para sua descrição e substituindo o passo em questão por uma chamada a esse novo caso de uso, o qual estará agora relacionado ao caso de uso base pelo estereótipo <<extend>>. Marque no campo "Extend" do caso de uso base o número do caso de uso criado.

B.5.4.1. Ao preencher o *template* para o novo caso de uso criado, transfira do caso de uso base os campos "Ator" para o caso de uso criado, o primeiro passo do caso de uso base para o campo "Evento Disparador" e os passos retirados do caso de uso base para o campo "Curso Normal".

B.6. Transcreva para o campo "Requisitos Funcionais" do *template* do caso de uso base as referências aos requisitos funcionais contidas no Formulário de Casos de Uso Preliminares.

B.7. Marque o caso de uso base, na coluna "Especificado" do Formulário de Casos de Uso Preliminares, com a palavra "sim".

C. Percorra a coluna "Include" do Formulário de Casos de Uso Preliminares e, para cada caso de uso que possuir apenas um número de caso de uso nessa coluna, considere-o como caso de uso base e faça:

C.1. Verifique se a soma dos passos do caso de uso relacionado com os passos do caso de uso base é menor que 10. Em caso afirmativo, coloque os passos do caso de uso base no local da chamada a ele, no caso de uso relacionado. Exclua o template do caso de uso base.

C.2. Verifique se os passos substituídos do caso de uso relacionado são os passos iniciais. Caso afirmativo, considere o caso de uso base uma pré-condição e, preencha o *template* do caso de uso relacionado na linha "Pré-condição" com o nome do caso de uso base.

D. Percorra a coluna "Include" do Formulário de Casos de Uso Preliminares e, para cada caso de uso que possuir mais de um número de caso de uso nessa coluna, considere-o como caso de uso base e faça:

D.1. Transcreva os números dos casos de uso identificados na coluna "Include" do Formulário de Casos de Uso Preliminares para o campo "Include" do template do caso de uso base.

E. Para cada caso de uso especificado nos passos anteriores, verifique, na seção Requisitos Não-Funcionais do Documento de Requisitos, quais os requisitos não-funcionais que pertencem ao caso de uso em questão e marque-os na linha "Requisitos Não-Funcionais" do *template* de especificação de casos de uso.

1. Em caso de um requisito funcional ser encontrado na seção de requisitos não-funcionais preencha o Relatório de Discrepâncias relatando esse fato.

Formulário de Casos de Uso Preliminares – FCUP

Formulário de Casos de Uso Preliminares

Nome:
Grupo:

Nº	Ator	Caso de Uso	Referência	Relacionamento	*Include*	Especificado

Formulário de Especificação de Casos de Uso – FEsp

Especificação do Caso de Uso	**Número:**	**Número do Caso de Uso**
Nome do Caso de Uso	Nome do caso de uso criado	
Descrição ou Resumo	Pequena descrição do caso de uso	
Ator Participante	Proprietário da Informação	
Ator Operador	Manuseia o computador	
Ator Genérico	Ator genérico criado pela análise dos atores participantes.	
Pré-Condição	Condições que devem ser verdadeiras para que o caso de uso possa ser realizado	
Curso Normal	Corresponde a um fluxo de eventos	
Curso Alternativo	Corresponde a fluxos de eventos, porém mostrando os caminhos menos comuns de acontecer	
Evento Disparador	Descreve o critério de entrada para o caso de uso sendo especificado	
Include	Número dos casos de uso relacionados pelo estereótipo <<include>>	
Extend	Número dos casos de uso relacionados pelo estereótipo <<extend>>	
Requisitos Funcionais	Os números dos requisitos funcionais associados ao caso de uso criado	
Requisitos Não-Funcionais	Os números dos requisitos não funcionais associados ao caso de uso	
Autor	A pessoa que criou a especificação	
Data	A data da criação da especificação	
Versão	Código associado à versão da especificação	

Relatório de Discrepâncias – UCRT

Formulário de Relato de Discrepância para UCRT

Nome do Projeto:
Equipe:
Nome do Integrante:

Início da Atividade: Data: _____ (data)

Tipo de Discrepância (Tipo Disc.):
(O) informação necessária omitida
(A) informação pode ser interpretada de várias maneiras
(II) informação contraditória a respeito do mesmo assunto
(FI) a informação não é correta considerando-se as condições especificadas pelo sistema
(IE) a informação é estranha, isto é, não é necessária no contexto da aplicação
(M) outros tipos de problemas

Preencha a tabela com as discrepâncias encontradas. Descreva a identificação dos requisitos, utilizando os números dos requisitos e números de página se possível:

Disc. #	Tipo Disc.	Referência	Comentários

Fim da Atividade: _____ (hora)