UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INSPEÇÃO DE ESPECIFICAÇÕES DE REQUISITOS REPRESENTADAS EM UNIFIED MODELING LANGUAGE (UML)

ERIK RIBEIRO DA CRUZ

ORIENTADORA: PROF[A] DR[A] TEREZA GONÇALVES KIRNER

PIRACICABA, SP
2006

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INSPEÇÃO DE ESPECIFICAÇÕES DE REQUISITOS REPRESENTADAS EM UNIFIED MODELING LANGUAGE (UML)

ERIK RIBEIRO DA CRUZ

ORIENTADORA: PROF[A] DR[A] TEREZA GONÇALVES KIRNER

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP
2006

INSPEÇÃO DE ESPECIFICAÇÕES DE REQUISITOS REPRESENTADAS EM UNIFIED MODELING LANGUAGE (UML)

AUTOR: ERIK RIBEIRO DA CRUZ

ORIENTADORA: TEREZA GONÇALVES KIRNER

Dissertação de Mestrado defendida e aprovada em 03 de Março de 2006, pela Banca Examinadora constituída dos Professores:

_____

Profa. Dra. Tereza Gonçalves Kirner (Orientadora)

UNIMEP

_____

Profa. Dra. Rogéria Cristiane Gratão de Souza

UNESP – São José do Rio Preto

_____

Prof. Dr. Luiz Eduardo Galvão Martins

UNIMEP

À

Minha esposa Kelli e minha filha Maria Eduarda

Aos

Meus pais Eurico e Roseli

AGRADECIMENTOS

À professora Tereza Gonçalves Kirner a orientação, compreensão, incentivo e companheirismo dispensado ao desenvolvimento deste trabalho. Aos participantes e a professora Rogéria Cristiane Gratão de Souza que gentilmente contribuíram para a pesquisa.

RESUMO

O interesse na utilização e estudo da *Unified Modeling Language* (UML) vem crescendo tanto no ambiente empresarial quanto acadêmico. A inspeção, que é uma técnica que visa à detecção de defeitos em artefatos de software, tem sido utilizada para avaliar e corrigir os diagramas UML, diagramas estes que contém uma grande quantidade de informações. Esta dissertação de mestrado apresenta um estudo empírico sobre a avaliação das *Object-Oriented Reading Techniques* (OORTs) para inspeção de especificações representadas em UML, nele foram avaliados a eficiência das técnicas e fatores de sua utilização por profissionais da área, onde foi possível constatar que a técnica é capacitada e tem um grande potencial.

PALAVRAS-CHAVE: *Unified Modeling Language*, Técnica de Leitura Orientada a Objetos, Inspeção de Software

INSPECTION OF REQUIREMENTS SPECIFICATIONS REPRESENTED IN UNIFIED MODELING LANGUAGE (UML)

ABSTRACT

The interest in the use and study of Unified Modeling Language (UML) has been growing such in companies as well in academic atmosphere. The inspection, that is a technique that seeks the detection of defects in software artifacts, has been used to evaluate and to correct the diagrams UML, which presents a great amount of information. This mastership dissertation presents an empiric study about the evaluation of Object-Oriented Reading Techniques (OORTs) for inspection of specifications represented in UML, during it were evaluated the efficiency of the techniques and factors of its use for professionals of the area, where it was possible to verify that the technique is qualified and has a great potential.

KEYWORDS: Unified Modeling Language, Object-Oriented Reading Techniques, Software Inspection

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO DA PESQUISA

Sabe-se que a maioria da população mundial está envolvida por um ambiente computacional, onde a utilização do software se faz necessária para realização de diversas atividades diárias. Devido esta importância, muito se fala sobre qualidade e produtividade no desenvolvimento e manutenção de software.

Segundo Rocha (2001), na década de 1990 houve uma grande preocupação quanto à melhoria do processo de software. Uma contribuição de grande importância desta área de pesquisa foi a percepção de que o desenvolvimento de um software é um esforço coletivo, complexo e criativo, e que desta forma a qualidade deste depende de pessoas, da organização e dos procedimentos utilizados.

O presente trabalho insere-se nesse contexto de preocupação com a qualidade do software, inclusive para aqueles que utilizaram a UML (*Unified Modeling Language*) no seu desenvolvimento, onde a importância da inspeção, tanto do código como de outros artefatos utilizados no seu desenvolvimento, tem ganhado grande importância e valia para obtenção desta qualidade.

Segundo Travassos (1999a), a especificação de requisitos e o projeto de software são atividades realizadas em momentos distintos. Segundo o autor, quando as atividades de projeto de alto nível (*High-Level Design*) terminam, é possível inspecionar os documentos (artefatos) de projeto para verificar se os mesmos são consistentes entre si e se atendem corretamente os requisitos especificados.

Foram estudadas durante esta pesquisa as técnicas de inspeção OORT (*Object-Oriented Reading Techniques*), que constituem, atualmente, das poucas propostas conhecidas, senão a única, para inspecionar requisitos e projetos de alto nível modelados em UML. Desta forma, serão apresentados conceitos sobre as técnicas de leitura e principalmente sobre a OORT, sendo também discutidos conceitos relacionados a ela, como inspeção de software e UML. A partir deste aprendizado, um estudo foi conduzido com profissionais especialistas em UML trabalhando como inspetores e avaliadores da técnica, na busca de dados e informações de qualidade sobre sua utilização na inspeção de diagramas e documentos.

1.2. OBJETIVOS

Neste trabalho foram realizados o levantamento bibliográfico sobre a *Unified Modeling Language* (UML) e sobre a inspeção de software, abordando diversos e diferentes pontos relacionados ao assunto, com destaque para as técnicas de inspeção que surgiram ao longo do tempo, como a *Object-Oriented Reading Techniques* (OORTs), que é o objeto de estudo desta dissertação de mestrado.

Como objetivo principal deste trabalho, teve-se o desenvolvimento do material (formulários, diagramas, documentos e instruções) utilizado para o estudo avaliativo sobre as OORTs, realizado junto a inspetores especialistas em UML. Desta forma, foi possível a obtenção de dados e informações, que foram analisados e interpretados para conclusões a respeito da eficiência da técnica em detectar defeitos e da sua utilização, considerando também a sua auto-suficiência.

1.3. IMPORTÂNCIA E JUSTIFICATIVA DA PESQUISA

O crescimento da utilização da UML nos projetos de desenvolvimento de software, faz com que este trabalho seja considerado de grande importância,

pois ainda não existe uma aceitação formal de uma técnica de leitura para inspeção de modelos UML, o que garantiria uma maior qualidade no software final.

Além disso, como a OORT é ainda uma proposta pouco estudada, ou seja, existem poucos estudos empíricos que a tenham por foco, o trabalho proposto e realizado apresenta uma grande quantidade de informações importantes para sua lapidação e para checagem da sua auto-suficiência.

Dentre as informações obtidas se destaca a eficiência da técnica para detectar defeitos em diagramas UML e também as particularidades de sua aplicação por especialistas nesta linguagem de modelagem, que caracteriza a proposta do estudo como inédita.

A OORT que em si já é uma proposta interessante poderá ser melhorada a partir das informações obtidas do estudo realizado.

1.4. ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em capítulos, conforme destacado a seguir.

O capítulo 2, apresenta a revisão bibliográfica, abordando os temas de inspeção de software, *Unified Modeling Language* (UML) e os principais trabalhos relacionados à OORT. O capítulo 3 descreve a definição da pesquisa, incluindo aspectos de sua realização e descreve também a análise de dados efetuada, destacando os resultados identificados com algumas considerações. As conclusões são apresentadas no capítulo 4, onde foram fornecidas explicações e ponderações, com base no conhecimento obtido através da análise de resultados do estudo empírico, bem como contribuições e sugestões para trabalhos futuros.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

2.1. INSPEÇÃO DE SOFTWARE

2.1.1. CONCEITOS

A inspeção de software, em sua primeira versão, foi proposta em 1972, nas dependências da IBM Kingston, Nova York. Apesar deste fato, a maioria dos autores referencia a data de 1976 como sendo a de maior importância para a inspeção de software, por ser a data em que Michael E. Fagan, um engenheiro de controle de qualidade da IBM, introduziu a inspeção no processo de desenvolvimento de software, através da publicação de seu artigo, sendo então creditada a ele a sua criação (FAGAN, 1976).

Inspeção de software é uma técnica de detecção de defeitos utilizada para verificar se um software está de acordo com seus requisitos (FAGAN, 1976, 1986).

Completando esta definição, segundo Travassos (2001), inspeção de software tem como ideal assegurar que o produto produzido, seja ele um documento, artefato ou outro resultado a ser fornecido, possui qualidade suficiente para ser utilizado por seu usuário.

Ainda com relação à conceituação, de acordo com Tomayko (1994), a inspeção pode ser aplicada em diferentes estágios do ciclo de vida do desenvolvimento de um software e tem como benefício localizar defeitos no código ou em outro artefato de software.

Pode-se entender então que inspeção de software é uma técnica estruturada, como será visto na seção “2.1.3 Processo de Realização da Inspeção”, utilizada para identificar defeitos existentes, tanto no código como em outros documentos de um software, com a finalidade de garantir maior qualidade no produto final.

2.1.2. OBJETIVOS DA INSPEÇÃO

Segundo Travassos (2001), o conceito de inspeção de produtos não é exclusivo da engenharia de software, ou seja, existem outras áreas, cada qual com seus objetivos, que fazem uso deste método, antes de liberar o produto para o mercado ou para o usuário. Mesmo estas áreas tendo objetivos específicos, elas compartilham do objetivo comum da inspeção, que é garantir que o produto possua qualidade suficiente.

O objetivo, comentado anteriormente é um objetivo relacionado à inspeção de um modo geral. Sendo mais específico, portanto falando sobre inspeção de softwares, observa-se como principal objetivo a aplicação de avaliações não apenas no código, mas em todos os artefatos gerados ao longo do processo de desenvolvimento do software, de maneira a detectar defeitos, como comentado por Travassos (2001).

Completando os objetivos, tem-se que, segundo Biffl (2000), a inspeção de software tem por objetivo identificar e medir a qualidade do produto e do processo de inspeção. As medidas são obtidas através da observação do número de defeitos existentes no produto, depois do seu desenvolvimento e depois da inspeção. O objetivo da detecção de defeitos é encontrar o maior número de defeitos possível, dentro de um certo tempo e com um determinado número de inspetores.

Apesar de terem sido propostas algumas variações no processo de inspeção, elas focaram apenas o que diz respeito ao processo de inspeção e não o

objetivo da inspeção. Segundo Travassos (2001), utilizando as palavras de Fagan (1986), o objetivo da inspeção envolve a realização de uma avaliação criteriosa do artefato para identificar os possíveis defeitos. Isso dará a oportunidade de remover os defeitos identificados antes de liberar o produto.

O custo de retrabalhar defeitos em programas, segundo Fagan (1976, 1986), torna-se maior, quanto mais tarde eles forem retrabalhados no processo. Assim, todo esforço deveria ser feito para detectar e corrigir defeitos no processo, o mais cedo possível. A inspeção, tem como um de seus benefícios evitar um maior custo de retrabalho; em conseqüência, pode-se concluir que ela tem como um de seus objetivos também a diminuição deste custo.

2.1.3. PROCESSO DE REALIZAÇÃO DE INSPEÇÃO

Conforme Travassos (2001), a maioria dos trabalhos relativos à inspeção de software foi influenciada pelas idéias de Fagan (1986) e Gilb (1998), mas principalmente por Fagan (1976, 1986), que é o autor mais citado quando se fala em inspeção de software.

Ainda segundo Travassos (2001), a inspeção inclui as fases de planejamento, detecção, coleção e, adicionalmente, correção dos defeitos. Biffl (2000) por sua vez, define as seguintes fases da inspeção: iniciação e planejamento de inspeção, detecção individual de defeitos, reunião da equipe ou grupo para determinação do resultado da inspeção, e correção de defeitos.

Segundo Fagan (1976), o processo de inspeção é composto por cinco fases: Visão Geral, Preparação, Inspeção, Retrabalho e Acompanhamento. Já em um outro artigo posterior, de 1986, além destas cinco fases, Fagan (1986) incluiu mais uma, a de Planejamento, realizada antes de todas as outras. Isso fez com que a inspeção de software fosse composta por um processo composto por seis conjuntos de atividades, ou seis operações.

Para a maioria dos autores, é considerado muito mais vantajoso ou de menor custo (de 10 a 100 vezes menos) detectar e corrigir defeitos quanto mais perto do ponto de origem for possível. Este fato, segundo Fagan (1976), é possível através da inspeção do artefato resultante de cada operação.

A tabela a seguir resume o processo de inspeção de software.

Tabela 1 – Processo de Inspeção de Software

FASE	OBJETIVOS
Planejamento	Os artefatos a serem inspecionados devem atender aos critérios da inspeção; Organizar a disponibilidade dos participantes; Organizar local e prazos para realização da inspeção.
Revisão	Treinamento dos participantes a respeito do que deverá ser inspecionado; Definição das regras de inspeção.
Preparação	Os participantes estudam o material e preparam-se para cumprir as regras que lhes foram designadas.
Inspeção	Detectar defeitos, através de um determinado procedimento.
Retrabalho	O autor do artefato inspecionado corrige todos os defeitos.
Acompanhamento	Verificação, pelo moderador da inspeção ou por toda a equipe, para assegurar que todas as correções foram efetuadas e que nenhum defeito secundário foi introduzido.

Em suma cada autor tenta dar a sua visão do processo de inspeção de software, mas todos tendem a se apoiar na idéia proposta por Fagan (1976, 1986). Segundo Travassos (2001), desde a sua criação, a inspeção tem recebido variações na forma de propostas, com a finalidade de aprimorá-la. Em Porter (1996), pode-se averiguar esta verdade quando são observadas em seus estudos as variações, que incluem: mudanças no número de pessoas na equipe de inspeção; inclusão de um *brainstorming* logo após o encontro de

inspeção; enfatizar a fase de preparação para detectar defeitos; utilização de um questionário como guia para os inspetores; quebra da fase de inspeção em diversas mini-inspeções; realização da inspeção com múltiplas equipes; etc.

Resumidamente, segundo Travassos (2001), as principais variações poderiam ser no âmbito da modificação na estrutura da reunião de inspeção (ver VOTTA, 1993), no número de revisores que deveriam participar da reunião ou, ainda, na eliminação da reunião de inspeção (ver VOTTA, 1993) através do uso de ferramentas.

Ressalta-se, em relação ao processo de inspeção, que diversos autores (FAGAN, 1976; TOMAYKO, 1994; DOOLAN, 1992; dentre outros) consideram que as sessões de inspeção devem durar até duas horas, pois continuar a inspeção além deste tempo faz com que a habilidade da equipe em detectar defeitos tenda a diminuir. Comenta-se ainda (FAGAN, 1976) que, após um período de descanso de duas horas ou a realização de alguma outra atividade durante este período, é possível se restaurar a capacidade de detectar defeitos.

2.1.4. EQUIPE DE INSPEÇÃO

De acordo com Porter (1996), existem propostas de variações no processo de inspeção que atingem, inclusive, o formato da equipe de inspeção, basicamente variando o número de pessoas que participam do processo. Mas, como a estrutura de equipe mais utilizada ou comentada na maioria dos artigos lidos é a de Fagan (1976, 1986), e com a finalidade de esclarecer quantas pessoas participam do processo de inspeção e quais são suas funções dentro dele, tal estrutura será apresentada.

A equipe de inspeção inclui profissionais que desempenham diferentes papéis, tais como: moderador, projetista, codificador e testador (FAGAN, 1976):

- **Moderador.** É a pessoa chave em uma inspeção de sucesso. Ele deve ser uma pessoa que entenda muito bem de desenvolvimento de software e que servirá como gerente e líder da equipe de inspeção. É aconselhável que ele não tenha relação alguma com o artefato a ser inspecionado, preservando assim a objetividade, e também que ele receba um treinamento especial para exercer sua função.

- **Projetista.** É o profissional responsável por produzir o projeto do artefato a ser inspecionado.

- **Codificador.** É o profissional responsável por traduzir o projeto em código.

- **Testador.** É o profissional responsável por escrever ou executar casos de teste ou testar o produto dos dois membros anteriores (projetista e codificador).

Fagan (1986) destacou os seguintes participantes de uma equipe de inspeção: autor, leitor, testador e moderador, que é a nomenclatura mais próxima do que tem sido utilizada por muitos autores, nos dias de hoje. Observa-se a diferença, com relação à estrutura anterior, apenas com relação ao autor e ao leitor. Neste caso, o autor estaria referindo-se à pessoa que produziu o produto ou artefato a ser inspecionado, desta forma podendo ser o “projetista” ou o “codificador”. Já o “leitor”, estaria referindo-se à pessoa encarregada de parafrasear o código ou o projeto, ou seja, realizar um discurso para os inspetores identificarem os possíveis defeitos. Na maioria das vezes, pode-se encontrar também a nomenclatura “inspetores” como substituindo a nomenclatura “testadores”.

Como é possível perceber, as pessoas que compõem a equipe de inspeção são desenvolvedores de software, destacando que isto ocorre para o caso da inspeção de código fonte. A equipe geralmente fica em torno de quatro pessoas, mas podendo variar de acordo com a circunstância. Normalmente,

aconselha-se um número de pessoas que varia de 4 a 6, (FAGAN, 1976) de forma a facilitar o gerenciamento do processo.

Devido à equipe apresentada ter sido desenvolvida para inspeção de software, mas com um enfoque na inspeção do código do software (FAGAN, 1976), foi necessário encontrar uma formação de equipe mais generalizada, ou seja, que pudesse ser aplicada em outros contextos de inspeção. Sendo mais específico, era necessário encontrar uma formação de equipe de inspeção que fosse mais adequada ao contexto da inspeção de especificação de requisitos em UML, que é o foco deste estudo.

Em resposta a essa necessidade, tem-se a equipe proposta por Ebenau (1994). Essa equipe é composta por cinco papéis diferentes, o autor, o moderador, o leitor, o registrador e o inspetor. A definição destes profissionais é dada a seguir.

- **Autor.** É a pessoa responsável por efetivar mudanças no documento. Durante a inspeção, ele fica incumbido de responder a perguntas que o leitor não é capaz de responder e de detectar defeitos no seu modo de entender o produto em inspeção.

- **Moderador.** Ele vai assegurar que os procedimentos da inspeção sejam cumpridos e também que os inspetores cumpram seus papéis durante a inspeção.

- **Leitor.** O leitor, que não deve ser o autor do artefato, conduz a equipe através do trabalho durante a reunião de inspeção. Ele deve ser uma pessoa preparada para descrever as várias partes e funções do trabalho, parafraseando o material em detalhes e em um ritmo que permita um exame completo.

- **Registrador.** É a pessoa que registra e classifica todos os defeitos encontrados durante a inspeção e também que ajuda o moderador a preparar os relatórios da reunião de inspeção.

- **Inspetor.** O inspetor é responsável por detectar defeitos, e não oferecer soluções, conforme o material é apresentado pelo leitor durante a reunião de inspeção.

2.1.5. DIFICULDADES DA INSPEÇÃO

Acredita-se que a inspeção, mesmo tendo suas dificuldades, ainda seja de grande importância e valia para a construção de um software. Muitas das dificuldades podem ser evitadas, ou seja, podem ser tomadas medidas de precaução para que elas não ocorram.

Dentro do que pode ser identificado, encontram-se dificuldades apontadas por Doolan (1992) na utilização do método de inspeção de Fagan (1976, 1986) em suas experiências, assim como também as dificuldades de Porter (1996), Gilb (1998), Tomayko (1994), que puderam ser averiguadas em seus relatos.

Apesar da divisão que foi feita para facilitar a visualização, as dificuldades a seguir algumas vezes relacionam-se e afetam umas às outras em suas categorias.

a) Dificuldades relativas ao início da inspeção

Segundo Doolan (1992), é importante que o primeiro projeto seja visto por todos como sendo de sucesso, pois desta forma a equipe será incentivada a contribuir com o processo de inspeção, sempre que ele for necessário. A equipe deve, portanto ser escolhida com cautela e não deve ser muito grande. Começar é difícil também, pois, dos materiais de entrada para as seções de

inspeção, nenhum foi inspecionado e, conseqüentemente, é bem provável que apresentem defeitos.

Quanto às dificuldades do início da inspeção, Tomayko (1994), enfatiza que, sem o orçamento adequado e o tempo para “preparação” e “acompanhamento”, armadilhas podem ser geradas. A preparação insuficiente poderá reduzir o número de itens que podem ser inspecionados em uma reunião, pois os participantes terão de perder tempo tentando entender o que já poderia ter sido entendido em um momento mais adequado. O acompanhamento insuficiente pode gerar a permanência de defeitos que poderão não vir a serem encontrados nos testes.

b) Dificuldades relativas durante a condução da inspeção

Existe a pressão de se inspecionar cada vez mais, quando o potencial da inspeção começa a se tornar evidente (DOOLAN, 1992). Uma forma de evitar esta dificuldade seria deixar clara a regra de se inspecionar um documento de cada vez. Mesmo tendo isso claro, pode ser interessante treinar mais moderadores para auxiliar na inspeção, quando o volume de documentos for muito grande.

A inabilidade de se diferenciar o produto do profissional que o criou é, segundo Tomayko (1994) e Gilb (1998), uma das grandes dificuldades de se conduzir uma inspeção. Os resultados da inspeção não devem ser utilizados como avaliação de performance das pessoas envolvidas na criação do produto. Desta forma, o moderador deve ficar incumbido de manter a inspeção focada no produto, assim como também manter um tom cordial durante as reuniões.

Normalmente, em todo ambiente de trabalho existe uma enorme exigência dos profissionais da área em cumprir com suas obrigações e obter resultados. Desta forma, além de existir uma grande dificuldade em disponibilizar todos os inspetores na mesma sala ao mesmo tempo durante a reunião de inspeção,

segundo Gilb (1998), muitos não tem a disciplina de conduzir um trabalho de inspeção por inteiro no espaço pequeno de tempo.

c) Dificuldades relativas ao controle da inspeção

Segundo Doolan (1992), é necessário que se esteja constantemente à procura de formas para provar os benefícios que surgem da inspeção. Ao trabalhar com estatísticas, é possível tornar visível aos participantes da inspeção a importância dos investimentos realizados.

Ainda conforme Doolan (1992), é importante monitorar todo o processo e estar preparado para manter os elementos essenciais mesmo que sejam necessárias algumas atualizações. As mudanças podem ocorrer no âmbito da administração e organização do processo e também na documentação.

d) Dificuldades relativas às pessoas

A participação das pessoas adequadas no processo de inspeção é de grande importância. Segundo Doolan (1992), é importante que se invista em esforços para assegurar o apoio de chefes ou superiores a todo o processo de inspeção. Eles são as pessoas que podem facilitar a alocação dos profissionais mais indicados e também dar um maior incentivo para que a inspeção seja feita da melhor forma.

É importante também segundo Doolan (1992), que profissionais qualificados sejam responsáveis por escrever as especificações de requisitos do software, que é uma tarefa difícil de se realizar, pois desta forma estar-se-ia evitando gastar grande tempo na sua inspeção e na possível necessidade de reescrevê-las.

Pode-se interpretar que existe uma dificuldade quanto à cooperação, pois, segundo Fagan (1976), na opinião de programadores, ou de uma forma mais geral dos desenvolvedores do artefato, eles não vêem o valor na inspeção, pois

acreditam terem feito tudo correto, ou por que seu projeto é pequeno ou por que, de alguma forma, ele é diferente.

e) Dificuldades relativas ao custo de inspeção

Segundo Fagan (1976), as experiências têm mostrado que os profissionais necessitam ser treinados para achar defeitos efetivamente, ou seja, é importante condicioná-los a buscar defeitos de alta ocorrência e alto custo. O autor sugere, então, uma inspeção preliminar, que represente o artefato a ser inspecionado como solução para este problema. Mas esta solução irá requisitar tempo e aumentará o custo da inspeção.

Segundo Porter (1996) existem dois problemas que corroem o custo-eficiência do uso da inspeção de software. Primeiro, o custo-beneficio da inspeção de software ainda não foi definido adequadamente, portanto não foi medido. E segundo, os agentes que causam o aumento dos benefícios e/ou diminuição dos custos da inspeção não foram rigorosamente estudados, tornando difícil determinar como e quando melhor utilizar a inspeção.

2.1.6. CONTRIBUIÇÕES DA INSPEÇÃO PARA A QUALIDADE E PRODUTIVIDADE DE SOFTWARE

Dentro do que pôde ser identificado, encontram-se contribuições apontadas por Doolan (1992) na utilização do método de inspeção de Fagan (1976, 1986) em suas experiências, assim como também contribuições destacadas por Fagan (1976, 1986), Gilb (1998), Porter (1996), Tomayko (1994) e Pagliuso (2002).

Vale ressaltar que, para facilitar a visualização, foi feita uma divisão das contribuições em algumas categorias. Mas as contribuições também se relacionam e afetam umas às outras, em suas categorias.

a) Contribuições relativas à qualidade

Segundo Doolan (1992), devido à necessidade da contínua atualização de software, a inspeção pode ser considerada como um bom meio de se identificar e priorizar as áreas que necessitam ser melhoradas, no caso as que possuírem potencialmente maior quantidade de defeitos.

A inspeção, segundo Pagliuso (2002), elimina cerca de 50 a 90% dos defeitos do processo de desenvolvimento, antes mesmo de chegar na fase de testes. Já Fagan (1976) e Gilb (1998) afirmam que a inspeção tem o beneficio de localizar defeitos no código e em outros documentos, chegando a detectar cerca de 80% de todos os defeitos em um determinado produto.

Também de acordo com Fagan (1986), a qualidade é influenciada pela grande diminuição do esforço de manutenção que se obtém ao realizar a inspeção, ou seja, ao detectar defeitos se estará evitando que eles permaneçam no produto final. Além disso, os benefícios não quantificáveis da inspeção em direção ao melhoramento do serviço fornecido aos usuários e em direção a um ambiente de desenvolvimento mais profissional (mais disciplinado, conforme interpretação de Tomayko (1994)) justificam seu uso, independente do custo previsto.

b) Contribuições relativas à produtividade

Durante a inspeção, ao identificarem a origem de um defeito detectado, segundo Doolan (1992), os inspetores podem sugerir modificações no processo de desenvolvimento do software, o que irá prevenir que defeitos similares ocorram em futuros projetos.

A inspeção ao remover defeitos em fases anteriores (PAGLIUSO, 2002; FAGAN, 1976; GILB, 1998), permite um ganho de tempo pelo fato de reduzir o retrabalho, melhorando, desta forma, a produtividade. Possibilita também uma redução de custos relativos a correções em fases posteriores.

Existem evidências de que desenvolvedores que participam de inspeções diminuem a possibilidade de inserirem mais defeitos em projetos futuros. Além disso, a diminuição de defeitos durante o desenvolvimento permite que os programadores sejam remanejados para trabalharem em outras atividades do projeto (FAGAN, 1976, 1986).

c) Contribuições relativas a custo

Segundo Porter (1996), o custo da inspeção normalmente é justificado com base no argumento de que quanto mais tempo um defeito permanecer em um software mais caro será repará-lo, ou seja, o custo de detectar um defeito agora será menor do que o custo de repará-lo no futuro.

Segundo Pagliuso (2002), avaliando-se o tempo gasto para se detectar defeitos utilizando a inspeção, percebe-se que esta é uma técnica eficiente, que corresponde apenas de 10 a 15% do orçamento de desenvolvimento, em termos de custo.

A inspeção, comparada com teste de máquinas, encontra mais defeitos no produto a um custo mais baixo (FAGAN, 1986). Fortalecendo esta afirmação, Porter (1996), destaca o caso do Banco Standard da África do Sul, que obteve diminuição do custo de manutenção corretiva em cerca de 95%. Destaca também o exemplo da IBM em que ocorreu uma economia de esforço dos programadores de 85%, com o uso da inspeção, dentre outros casos analisados.

d) Contribuições relativas às pessoas

A inspeção faz com que os inspetores ganhem ou aumentem o conhecimento técnico quanto ao material que foi inspecionado. Ela proporciona também a participação de profissionais em equipes de trabalho o que, conseqüentemente, proporciona um aprendizado a cerca de métodos e

procedimentos eficientes de desenvolvimento de software (DOOLAN, 1992; FAGAN, 1976; GILB, 1998).

A participação em inspeções pode promover o espírito de equipe, contribuindo também para a transferência de conhecimento tecnológico e atuando como *back-up*, caso alguma pessoa seja removida repentinamente de um projeto e outra seja designada para seu lugar (DOOLAN, 1992).

Ao utilizar a inspeção em diferentes estágios do ciclo de desenvolvimento de software, os engenheiros de software ganham dados valiosos referentes à ocorrência de defeitos no produto e às possíveis soluções, melhorando também as suas habilidades individuais (TOMAYKO, 1994; FAGAN, 1976).

2.2. INSPEÇÃO DE ESPECIFICAÇÕES DE REQUISITOS EM *UNIFIED MODELING LANGUAGE* (UML)

2.2.1. A METODOLOGIA UML

Entre a década de 1970 e final da década de 1980, (BOOCH, 2000), surgiram as linguagens de modelagem orientadas a objetos. Durante o período de 1989 a 1994, foi identificado um grande crescimento dos métodos orientados a objetos, o que levou à chamada de “guerra de métodos”, pois os usuários tinham dificuldades em encontrar uma linguagem de modelagem que atendesse plenamente às suas necessidades. Novos métodos foram surgindo com a experiência, podendo-se destacar o Booch, o OOSE (*Object Oriented Software Engineering*) de Jacobson, o OMT (*Object Modeling Technique*) e outros, como Fusion, Shaler-Mellor e Coad-Yourdon.

Por volta da metade da década de 1990, Grandy Booch, Ivar Jacobson e James Rumbaugh deram início a coleta das melhores idéias de cada um destes métodos, iniciando-se , assim, os esforços para a definição da UML.

Em 1996, a comunidade de engenharia de software foi solicitada a contribuir com a criação da UML, sendo que várias empresas também manifestaram interesse. Diante deste fato, um consórcio de UML foi estabelecido com várias destas empresas, que desejavam envolver-se na definição da nova linguagem. Em janeiro de 1997, foi entregue a versão 1.0 para a OMG (*Object Management Group*), com a finalidade de padronização, a partir de uma solicitação da própria OMG, que buscava uma linguagem padrão para modelagem.

Segundo Almeida (2001), a UML representa uma tentativa de padronizar a modelagem orientada a objetos, com vistas a que qualquer sistema possa ser modelado corretamente, com consistência, com fácil comunicação com outras aplicações, simples de ser atualizado e compreensível.

A UML é uma linguagem padrão para a elaboração da estrutura de projetos de software, ou seja, é somente uma parte de um método para desenvolvimento de software. Desta forma, a UML é uma linguagem de modelagem, cujo vocabulário e regras têm seu foco voltado para a representação conceitual e física de um modelo de sistema (BOOCH, 2000).

Devido a UML possuir uma semântica bem definida, referente a cada símbolo utilizado, qualquer desenvolvedor será capaz de interpretar tais símbolos sem ambigüidade. Esses símbolos são utilizados para desenvolver diagramas, que podem ser considerados como a essência da UML. Ao todo, são utilizados nove diagramas: o diagrama de Casos de Uso, Classes, Objetos, Estados, Seqüência, Colaboração, Atividades, Componentes e Execução.

A seguir será apresentada uma breve explanação de cada diagrama (ALMEIDA, 2001; SILVA, 1999; BARROS, 2000; BOOCH, 2000; ERIKSSON, 1998).

a) Diagrama de Casos de Uso

A modelagem de um diagrama de casos de uso é usada para descrever e definir os requisitos funcionais de um sistema. Esses diagramas são escritos em termos de atores externos, casos de uso e o sistema modelado. Os atores representam o papel de uma entidade externa ao sistema, que iniciam a comunicação com ele. Os casos de uso representam uma seqüência de ações executadas pelo sistema.

Um ator é conectado a um ou mais casos de uso através de associações, e tanto atores quanto casos de uso podem possuir relacionamentos de generalização, que definem um comportamento comum de herança em superclasses especializadas em subclasses.

No modelo de casos de uso, o relacionamento entre um ator e um caso de uso representa a participação deste ator no caso de uso. Além deste relacionamento, existem dois outros tipos de relacionamentos entre casos de uso:

- O relacionamento estender, que é representado graficamente por uma seta com o estereótipo <<extends>>, mostrando que o caso de uso destino pode incluir o comportamento especificado pelo caso de uso origem.

- O relacionamento usar, é representado por uma seta com o estereótipo <<uses>>, mostrando que o caso de uso origem inclui o comportamento especificado pelo caso de uso destino.

A figura 1, mostra um exemplo de Diagrama de Casos de Uso.

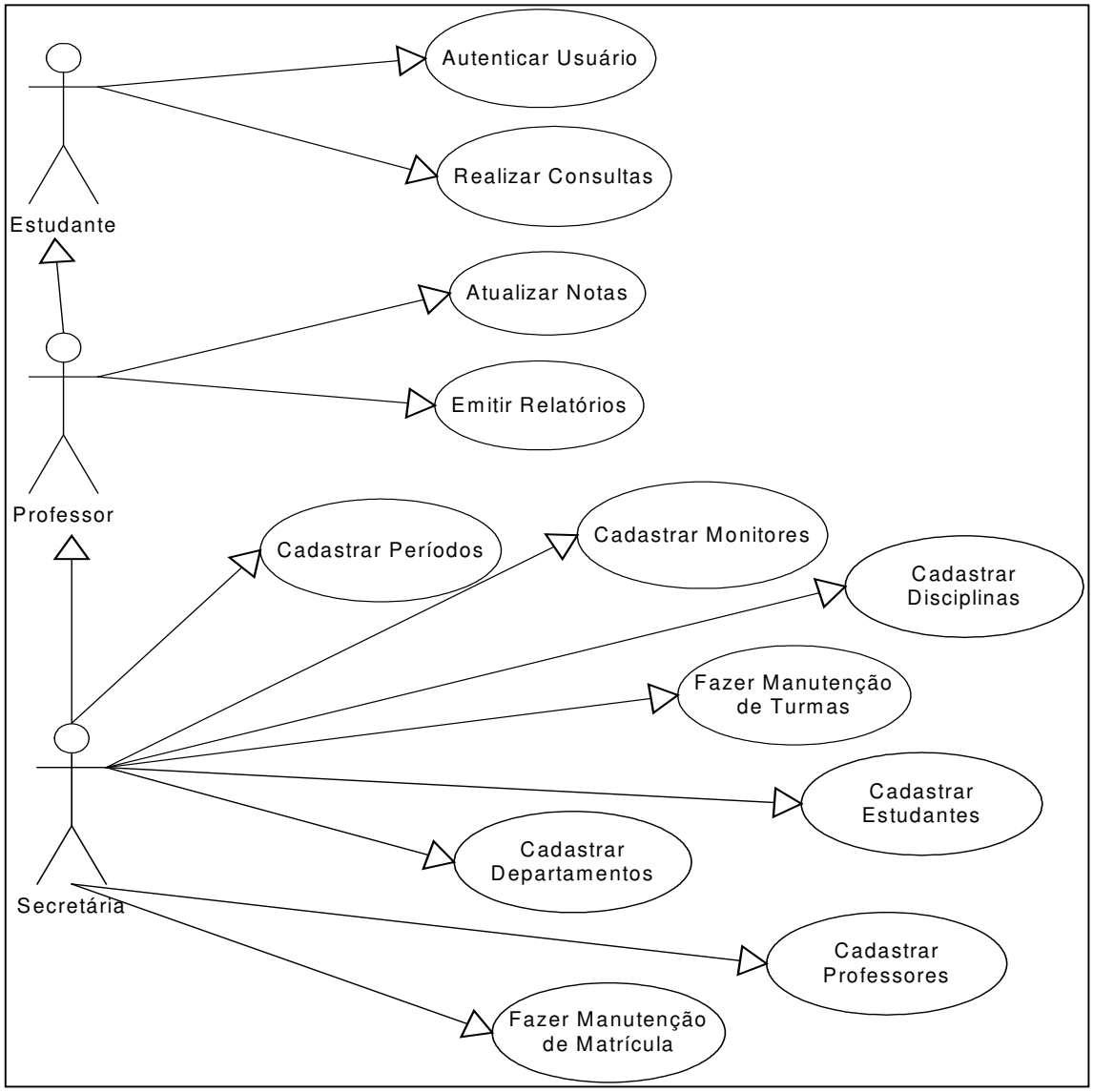


Figura 1 – Diagrama de Casos de Uso (SOUZA, 2003)

b) Diagrama de Classes

O diagrama de classes demonstra a estrutura estática (sempre válida em qualquer ponto do ciclo de vida do sistema) das classes de um sistema. Um sistema pode possuir mais de um diagrama de classes, pois nem todas as classes estão inseridas em um único diagrama. Além disso, uma classe pode participar de vários diagramas de classes.

Classes podem se relacionar com outras, através de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa

outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares).

Vale ressaltar que as associações representam relacionamentos estruturados entre objetos de diferentes classes, sendo representados graficamente através de uma linha conectando as classes.

A figura 2, mostra um Diagrama de Classes.

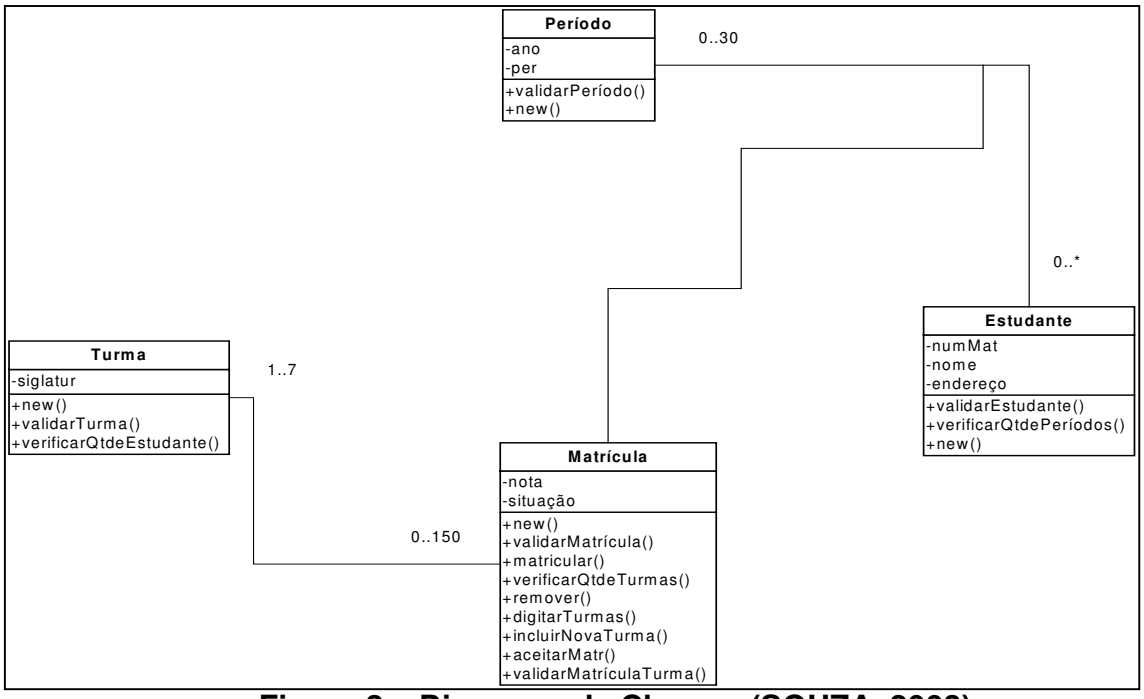


Figura 2 – Diagrama de Classes (SOUZA, 2003)

c) Diagrama de Objetos

O diagrama de objetos é uma variação do diagrama de classes e utiliza quase a mesma notação, com a diferença de que o diagrama de objetos mostra os objetos que foram instanciados das classes e seus nomes são escritos sublinhados. Este diagrama é como se fosse um momento congelado no tempo, ou seja, o perfil do sistema em um certo momento de sua execução.
Eles são muito úteis para exemplificar diagramas complexos de classes, ajudando muito em sua compreensão.

Frequentemente, o diagrama de objetos é criado para ilustrar como um diagrama de classes é instanciado em objetos, mostrando também, como objetos de classes podem ser combinados entre si em um determinado momento.

A figura 3 apresenta um Diagrama de Objetos.

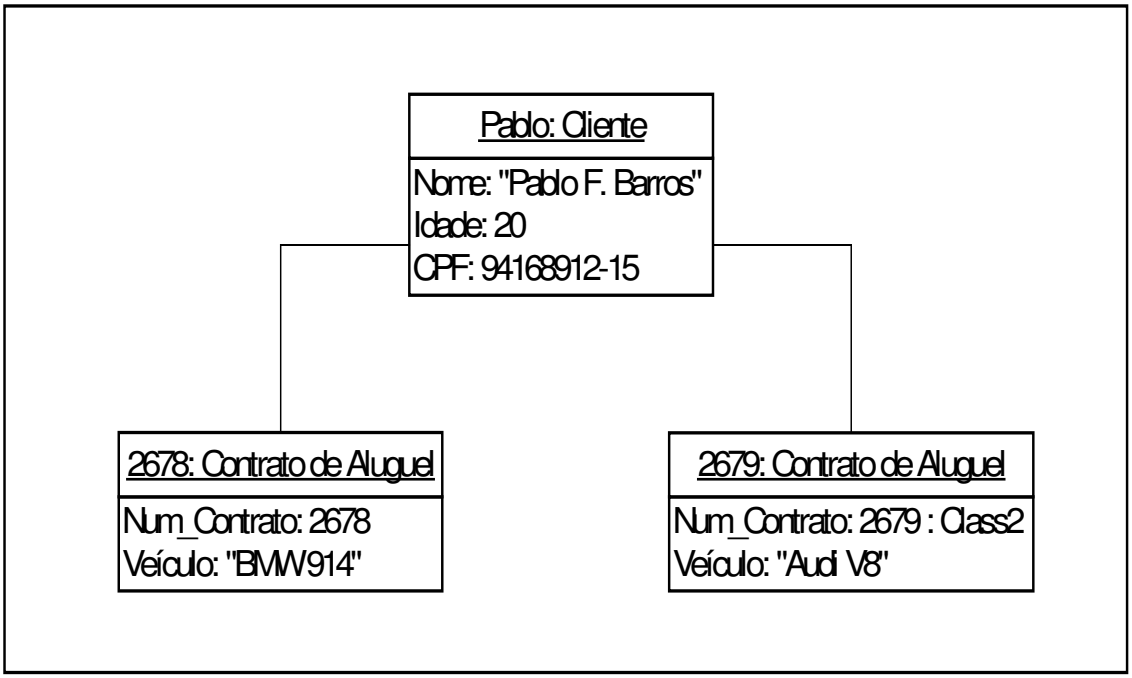


Figura 3 – Diagrama de Objetos (BARROS, 2000)

d) Diagrama de Estados

Este diagrama mostra todos os estados possíveis que objetos de uma determinada classe podem assumir. Mostra também os eventos do sistema que provocam tais mudanças, sendo então um complemento para a descrição das classes.

Eles mostram os estados que um objeto pode possuir e como os eventos (mensagens recebidas, timer, erros, e condições sendo satisfeitas) afetam estes estados, ao passar do tempo.

Diagramas de estado possuem um ponto de início e vários pontos de finalização. Um estado é mostrado como um retângulo com cantos arredondados. Entre os estados, estão as transições, mostradas como uma linha com uma seta no final de um dos estados. A transição pode ser nomeada com o seu evento causador. Quando o evento acontece, a transição de um estado para outro é executada ou disparada.

Um exemplo de Diagrama de Estados é mostrado na figura 4.

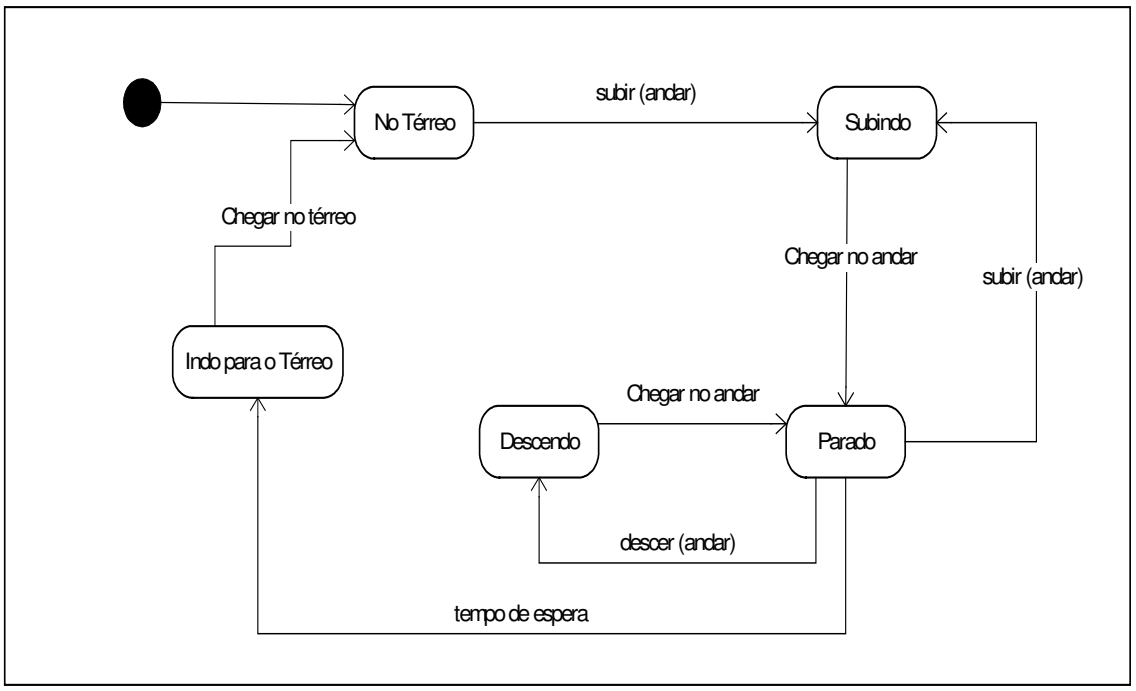


Figura 4 – Diagrama de Estados (ERIKSSON, 1998)

e) Diagrama de Seqüência

O mais importante aspecto deste diagrama é que, a partir dele, percebe-se a seqüência de mensagens enviadas entre os objetos. Ele possui dois eixos: o eixo vertical, que mostra o tempo, e o eixo horizontal, que mostra os objetos envolvidos na seqüência de uma determinada atividade. O decorrer do tempo é visualizado observando-se o diagrama no sentido vertical, de cima para baixo. As mensagens enviadas por cada objeto são simbolizadas por setas entre os objetos que se relacionam.

No eixo horizontal, cada objeto envolvido na seqüência é representado por um retângulo de objeto e uma linha vertical pontilhada, chamada de linha de vida do objeto.

O comportamento de um objeto é modelado em função de qual estado ele está inicialmente, e para qual estado ele vai passar quando um determinado evento ocorrer. Os eventos representam incidentes que causam a mudança de um estado para outro. As linhas de transição descrevem o movimento de um estado para o outro. Cada linha de transição é rotulada com o evento que causou a transição.

A figura 5 contém um Diagrama de Seqüência.

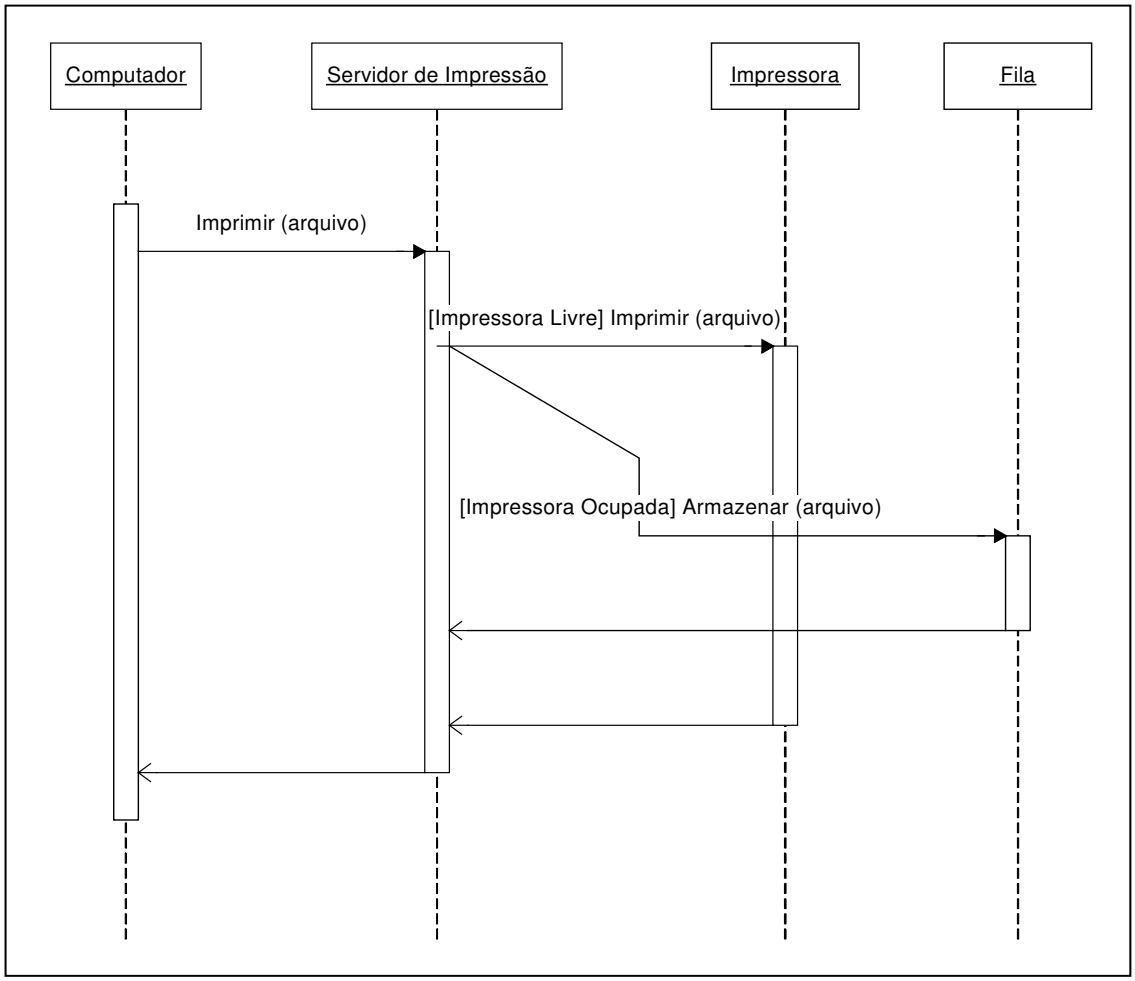


Figura 5 – Diagrama de Seqüência (ERIKSSON, 1998)

f) Diagrama de Colaboração

Um diagrama de colaboração mostra, de maneira semelhante ao diagrama de seqüência, a colaboração dinâmica entre os objetos. Mas, diferentemente do diagrama de seqüência, que focaliza a seqüência cronológica do cenário que está sendo modelado, o diagrama de colaboração enfoca o relacionamento entre os objetos e a compreensão dos efeitos sobre um objeto durante um cenário.

Se a ênfase do diagrama for o decorrer do tempo, é melhor escolher o diagrama de seqüência, mas se a ênfase for o contexto do sistema, é melhor dar prioridade ao diagrama de colaboração.

Os objetos do diagrama são conectados através de associações e cada associação representa uma instância de associação entre as respectivas classes envolvidas. As setas de mensagens são nomeadas e desenhadas entre os objetos, para mostrar o fluxo de mensagens entre eles e a ordem em que são enviadas, podendo também mostrar condições, interações, valores de resposta, etc.

Um exemplo de Diagrama de Colaboração é mostrado na figura 6.

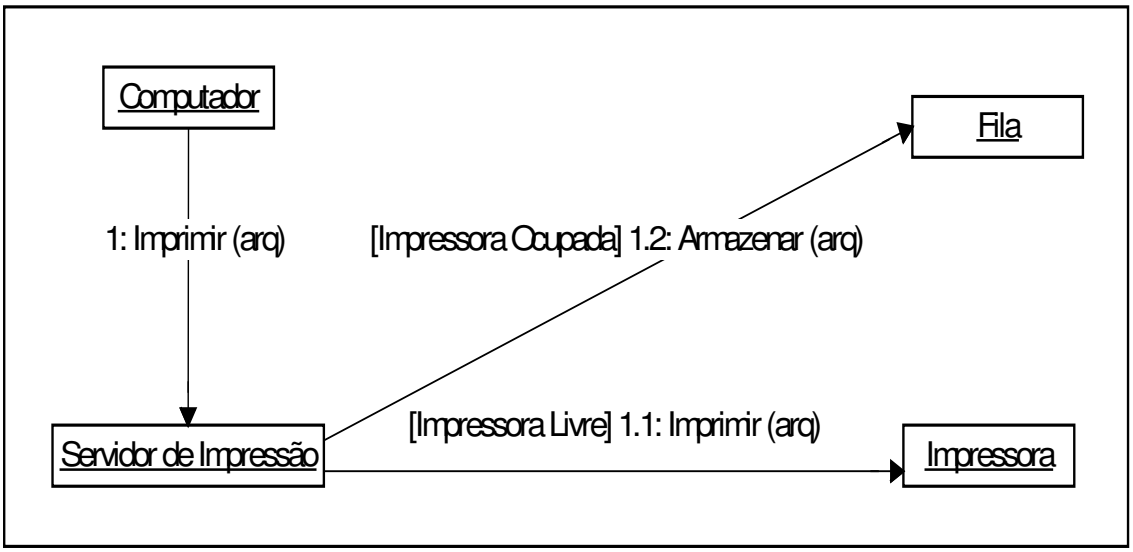


Figura 6 –Diagrama de Colaboração (ERIKSSON, 1998)

g) Diagrama de Atividades

Um diagrama de atividade é um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Ele captura ações e seus resultados, dando enfoque ao trabalho executado na implementação de uma operação (método), e suas atividades numa instância de um objeto. Possui o propósito de capturar ações (trabalho e atividades que serão executados) e seus resultados em termos das mudanças de estados dos objetos, focando nos fluxos dirigidos pelo processamento interno e descrevendo o comportamento de processamentos paralelos.

Ele é uma maneira alternativa de se mostrar interações, com a possibilidade de expressar como as ações são executadas, o que elas fazem (mudanças dos estados dos objetos), quando elas são executadas (seqüência das ações), e onde elas acontecem. São usados para detalhar classes, implementação de operações e casos de uso. Ele não diz qual classe é responsável por cada atividade.

Os diagramas de atividades são casos especiais de diagramas de estado, onde todos os estados têm uma ação interna e nenhuma transição tem um evento de entrada. Os estados no diagrama de atividade mudam para um próximo estágio quando uma ação é executada.

A figura 7 apresenta um Diagrama de Atividades.

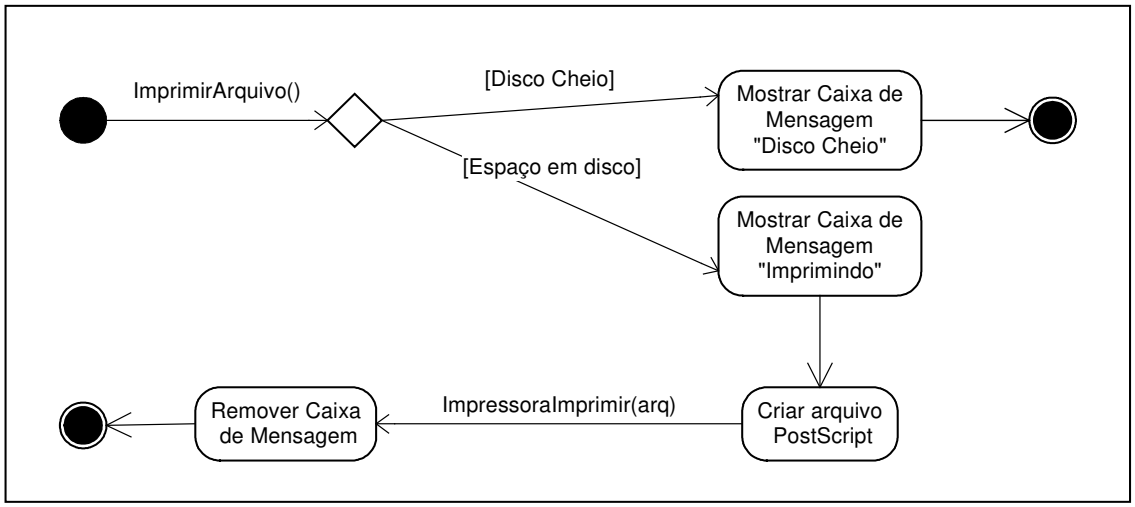


Figura 7 – Diagrama de Atividades (ERIKSSON, 1998)

h) Diagrama de Componentes

O diagrama de componentes mostra o sistema por um lado funcional. Ele descreve os componentes de software e suas dependências entre si, ou seja, representam, de forma estática, aspectos físicos do sistema sendo modelado.

Os componentes são a implementação, na arquitetura física do sistema, dos conceitos e da funcionalidade definidos na arquitetura lógica (classes, objetos e seus relacionamentos). Eles são tipicamente os arquivos implementados no ambiente de desenvolvimento.

Um diagrama de componentes mostra apenas componentes como tipos. Para mostrar instâncias de componentes, deve ser usado um diagrama de execução. A dependência entre componentes pode ser mostrada como uma linha tracejada com uma seta, simbolizando que um componente necessita do outro para possuir uma definição completa.

A figura 8 mostra um Diagrama de Componentes.

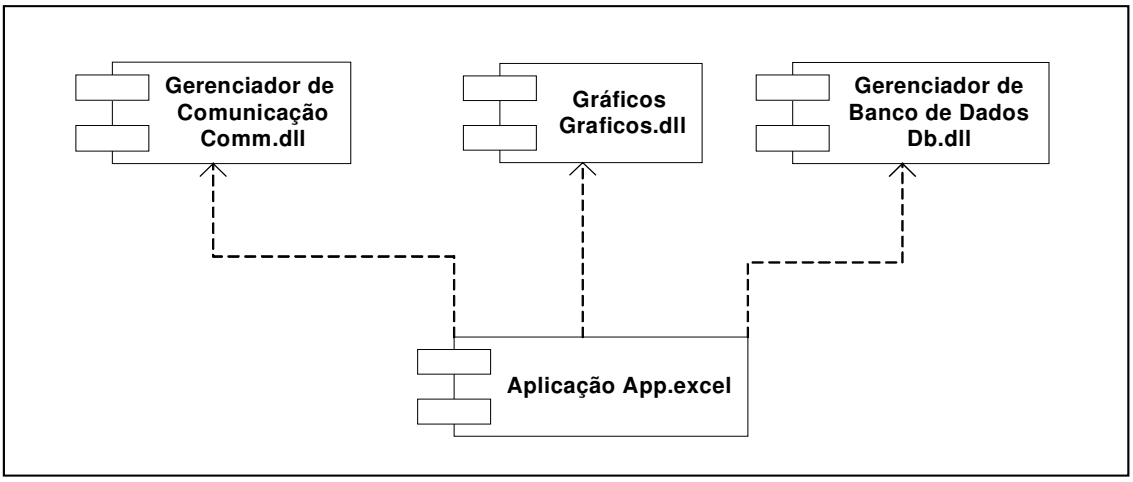


Figura 8 – Diagrama de Componentes (BARROS, 2000)

i) Diagrama de Execução ou Implantação

Os diagramas de execução são empregados para modelagem da visão estática de implantação de um sistema. Essa visão direciona primariamente a distribuição, entrega e instalação das partes que formam o sistema físico, ou seja, a arquitetura física do hardware e do software no sistema, mostrando os atuais computadores e periféricos, juntamente com as conexões e os tipos de conexões que eles estabelecem entre si. Especifica-se também os componentes executáveis e objetos que são alocados para mostrar quais unidades de software são executados e em quais destes computadores são executados.

O diagrama de execução é composto por componentes, que possuem a mesma simbologia dos componentes do diagrama de componentes, *nodes*, que significam objetos físicos que fazem parte do sistema, e conexões entre estes *nodes*.

Um exemplo de Diagrama de Execução é apresentado na figura 9.

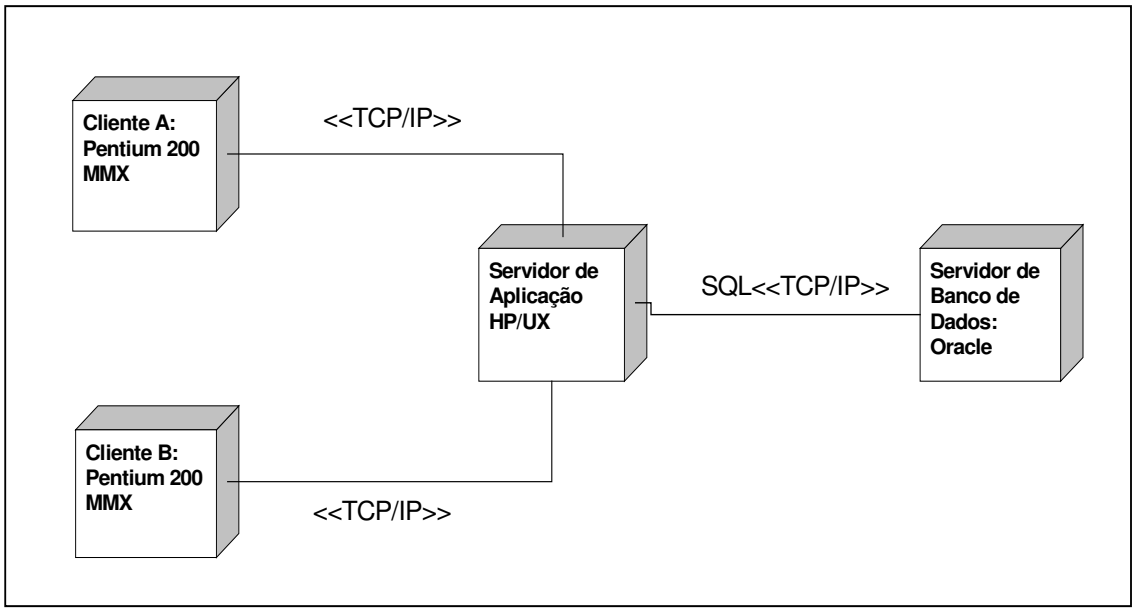


Figura 9 – Diagrama de Execução ou Implantação (BARROS, 2000)

Neste estudo, nem todos os diagramas apresentados anteriormente serão utilizados. Apesar disto, todos eles foram apresentados com o intuito de melhor esclarecer o que seria a UML e também a fim de exemplificar, para que o entendimento deste estudo fosse facilitado.

2.2.2. DEFEITOS EM ESPECIFICAÇÕES DE REQUISITOS

Ao analisar as variações de performance durante inspeções, Carver (2000), observou a existência de diferentes formas de classificação de defeitos. Segundo ele (CARVER, 2000), a maioria das propostas referia-se a defeitos nos documentos de requisitos e no código fonte, desta forma, sendo poucas as que consideravam defeitos em diagramas.

Em um estudo sobre inspeção de documentos UML, Bunde (2002) empregou um formulário de registro de defeitos para que os inspetores pudessem preenchê-lo, conforme os defeitos fossem encontrados durante a inspeção. Isso organizaria a identificação de defeitos e facilitaria tanto sua correção como a análise para conclusões. Bunde (2002) utilizou uma classificação de defeitos similar à utilizada por Travassos (1999a).

A Tabela 2 apresenta a classificação de defeitos, definida a partir de Bunde (2002) e Travassos (2001).

Tabela 2 – Classificação de Defeitos

TIPO DE DEFEITO	DESCRIÇÃO
Omissão	Um ou mais diagramas que deveriam conter algum conceito dos requisitos gerais ou do documento de requisitos, não contém uma representação para aquele conceito.
Fato Incorreto	O projeto de um diagrama possui uma interpretação errônea de um conceito descrito nos requisitos gerais ou no documento de requisitos.
Inconsistência	A representação de um conceito no projeto de um diagrama está em desacordo com a representação do mesmo conceito no mesmo diagrama ou em outro diagrama.
Ambigüidade	A representação de um conceito em um projeto não está clara, e poderia causar ao usuário (desenvolvedor, projetista de baixo nível, etc) do documento uma interpretação errônea ou um entendimento errôneo do significado do conceito.
Informação Estranha	O projeto inclui informações que, enquanto aparentemente verdadeiras, não se aplicam ao domínio e não deveriam ser incluídas no projeto.

Uma outra proposta de classificação de defeitos é feita por Ebenau (1994), na qual são incluídas as seguintes informações:

• **Localização do Defeito.** Em qual artefato (ou parte) está localizado o defeito.

• **Questão.** Identificação da questão do *checklist* que levou a detectar o defeito.

• **Descrição.** Envolve uma descrição detalhada do defeito.

- **Tipo.** O tipo do defeito irá conduzir a uma sub-localização, possibilitando uma melhor focalização de onde está o defeito.

- **Classificação.** Tendo encontrado o defeito, a classificação dará uma idéia do que causou aquele defeito.

- **Severidade.** Pode ser considerado como o grau de gravidade do defeito, que impacto ele tem no projeto.

A tabela 3 descreve níveis de severidade de um defeito.

Tabela 3 – Severidade do Defeito

SEVERIDADE DO DEFEITO	DESCRIÇÃO
Maior	Produz o abandono de um requisito ou especificação, ou produz uma falha na operação, ou impede um desenvolvimento de sucesso futuro do produto.
Menor	Defeitos de formato ou representação, ou aqueles que não causam falhas ou abandono de requisitos.

A tabela 4 apresenta uma classificação de defeitos, proposto por Ebenau (1994).

Tabela 4 – Tipos de Defeitos

TIPOS DE DEFEITOS	DESCRIÇÃO
Dado	Defeitos na especificação dos dados; declaração, inicialização, ou descrição imprópria do dado; incorreto uso do dado, conversão de tipos de dados, ou de limites de array.
Documentação	Descrição de componente inadequada ou incorreta (por exemplo: comentários faltando, incompleto ou incorreto).
Funcionalidade	Defeitos na especificação das funções de um componente.
Fator Humano	Operação de procedimentos incorreta ou inadequada, não especificação de qual operação de procedimento usar, envolvimento desnecessário do operador.
Interface	Defeitos na comunicação entre componentes de software (por exemplo: invocação incorreta do módulo, caminho incorreto do dado), excluindo defeitos no dispositivo ou nas interfaces humanas.
Entrada/Saída	Defeitos na comunicação ou especificação com dados externos ou dispositivos.
Lógico	Defeitos nos procedimentos ou na seqüência, seleção, interação de operações (por exemplo: incorreta condição de restrição em um *loop*, comparação incorreta); algoritmos ou matemática computacional incorreta.
Manutenibilidade	Expectativa de que o produto trabalhado seja difícil de dar manutenção (por exemplo: não é possível de se compreender ou possui um indesejável efeito colateral), e de excluir defeitos na documentação.
Performance	Uma expectativa de não encontrar a eficiência requerida na execução (por exemplo: a velocidade de execução está muito lenta, muita memória está sendo utilizada).
Sintaxe	Defeitos de gramática, pontuação, soletração, e especificação da linguagem usada.
Padrões	Um desvio dos padrões processuais ou representacionais.
Outros	Uma condição de defeito indefinida ou ambígua.

2.2.3. TÉCNICAS DE LEITURA PARA INSPEÇÃO DE SOFTWARE

As técnicas de leitura são essenciais à inspeção, pois, ao estruturarem o processo e fornecerem diretrizes para uma inspeção eficiente, elas determinam como os inspetores irão gerenciar a busca por defeitos nos artefatos avaliados (ARIF, 2001).

Diferentes técnicas baseadas em leitura foram propostas para apoiar a inspeção, sendo que estas técnicas podem diferir de inspeção para inspeção dependendo dos documentos inspecionados (ARIF, 2001; KYLVAG, 2001).

Segundo Kylvag (2001), uma técnica de leitura é um procedimento detalhado, indicado para guiar os inspetores a detectarem defeitos em um artefato de software. Com isto, os inspetores terão em suas mãos uma forma sistemática e bem definida para inspecionar um documento.

As principais técnicas de leitura para inspeção de software são: Leitura *Ad Hoc*, Leitura baseada em *Checklist* e Leitura Baseada em Cenários (ver ARIF, 2001; KYLVAG, 2001; CIOLKOWSKI, 1999). Apesar da técnica de leitura *Checklist* ser a mais recomendada na maioria dos métodos de inspeção (KYLVAG, 2001), todas as técnicas serão comentadas. Abaixo, seguem as explicações, conforme definições fornecidas por Arif (2001), Kylvag (2001) e Ciolkowski (1999).

a) Leitura *Ad Hoc*

A leitura Ad Hoc pode ser considerada como a técnica mais simples, sendo utilizada em vários estudos empíricos. Esta técnica consiste numa dependência de conhecimento, de experiência e de senso comum do próprio inspetor para detectar defeitos nos documentos. Desta forma, não existe um procedimento ou diretriz a ser seguido, não existe uma definição específica do que procurar durante a leitura e também não existe nenhum treinamento para a sua realização. Os resultados advindos da realização deste tipo de leitura são

totalmente dependentes do inspetor e de como ele determinar a procura por defeitos no artefato inspecionado.

b) Leitura baseada em *Checklist*

A leitura baseada em *Checklist* consiste numa descrição de regras que auxiliam o inspetor a achar defeitos. Para aplicar a técnica, o inspetor recebe uma lista de questões que abordam diferentes aspectos do documento sendo inspecionado. Estas questões são elaboradas com base no conhecimento de potenciais defeitos relacionados ao documento em inspeção e também a partir de lições previamente aprendidas durante outras inspeções. Além das questões, podem estar inclusas no *Checklist* as definições de responsabilidades e sugestões de como identificar defeitos nos documentos.

O inspetor, ao responder as questões do *Checklist*, será guiado a detectar defeitos no documento. Apesar da técnica não descrever como identificar a informação necessária para responder as questões e como realizar a checagem necessária, ainda assim, ela é bem recomendada e possui o benefício de fornecer o que procurar durante a inspeção, o que é uma vantagem sobre a leitura *Ad Hoc*.

c) Leitura baseada em Cenário

A leitura baseada em Cenário foi criada com a intenção de eliminar os problemas existentes na técnica de leitura baseada em *Checklist*.

Este tipo de leitura parte do princípio que uma inspeção seria mais eficiente se cada inspetor tivesse seu foco voltado para diferentes aspectos do artefato inspecionado. Desta forma, ela é uma técnica de leitura onde cada inspetor realiza a leitura com um propósito em especial. Ou seja, o documento é inspecionado por especialistas que teriam seus *checklists* ajustados para sua área de experiência, checando determinadas informações e ignorando as que não lhe fossem familiar.

A leitura baseada em Cenários força o inspetor a ter um papel ativo e um efeito adicional deste fato é que, ao seguir um cenário, o inspetor ganha um conhecimento mais profundo sobre o sistema que foi descrito no documento de requisitos.

2.3. A ABORDAGEM OORT PARA INSPEÇÃO DE ESPECIFICAÇÕES ORIENTADAS A OBJETOS

A Técnica de Leitura Orientada a Objetos (OORT – *Object Oriented Reading Technique*), foi criada visando apoiar inspeções de requisitos representados em UML (TRAVASSOS, 1999a).

A apresentação da técnica OORT, a seguir, baseia-se nos trabalhos de Travassos (2001, 1999a, 1999b, 2002)

A OORT apóia a inspeção do projeto de alto nível, que inclui artefatos preocupados com a representação dos conceitos do mundo real. Estes artefatos não pretendem representar os detalhes da implementação no qual o sistema existe. Eles iniciam-se após os requisitos terem sido capturados e representados textualmente e/ou através de Casos de Usos. Desta forma, os diagramas criados e que serão alvo da OORT são os diagramas de Classes, Seqüência e Estados.

Após ter sido finalizado o projeto de alto nível, os diagramas criados serão checados em busca de defeitos, averiguando se estão consistentes entre eles e se estão capturando corretamente os requisitos do software. Para isto, Travassos (2001) utilizou uma taxonomia de tipos de defeitos que podem ser encontrados em artefatos orientados a objetos. Esta taxonomia compreende defeitos de omissão, fato incorreto, inconsistência, ambigüidade e informação estranha (ver Tabela 2, pg. 30).

O primeiro estudo objetivando testar a viabilidade da OORT, foi realizado em 1998 (TRAVASSOS, 2002). Nele, estavam envolvidos 44 estudantes do curso de engenharia de software sendo que:

- 32% possuíam experiência industrial com projeto de software.
- 9% não possuíam experiência alguma.
- 59% possuíam experiência em sala de aula com projeto de software.

Neste primeiro estudo, foram aplicadas as técnicas de leitura horizontal e vertical. A leitura horizontal verifica a consistência entre os diagramas de Classes, de Seqüência e de Estados. A leitura vertical verifica a consistência entre os diagramas e os Requisitos do sistema, representados através do documento de Requisitos e o diagrama de Casos de Uso.

A figura 10 esquematiza as técnicas OORT, através das leituras horizontal e vertical.

No início, cada equipe recebeu os requisitos de um sistema de empréstimo e realizaram uma inspeção destes requisitos, melhorando seus conhecimentos sobre o sistema. A partir dos requisitos, eles criaram os diagramas e dois dos melhores resultados foram escolhidos para serem inspecionados, através das técnicas OORT. Durante o estudo, as técnicas foram divididas entre os estudantes, sendo que uma parte do grupo avaliou a partir da leitura vertical e a outra parte avaliou a partir da leitura horizontal.

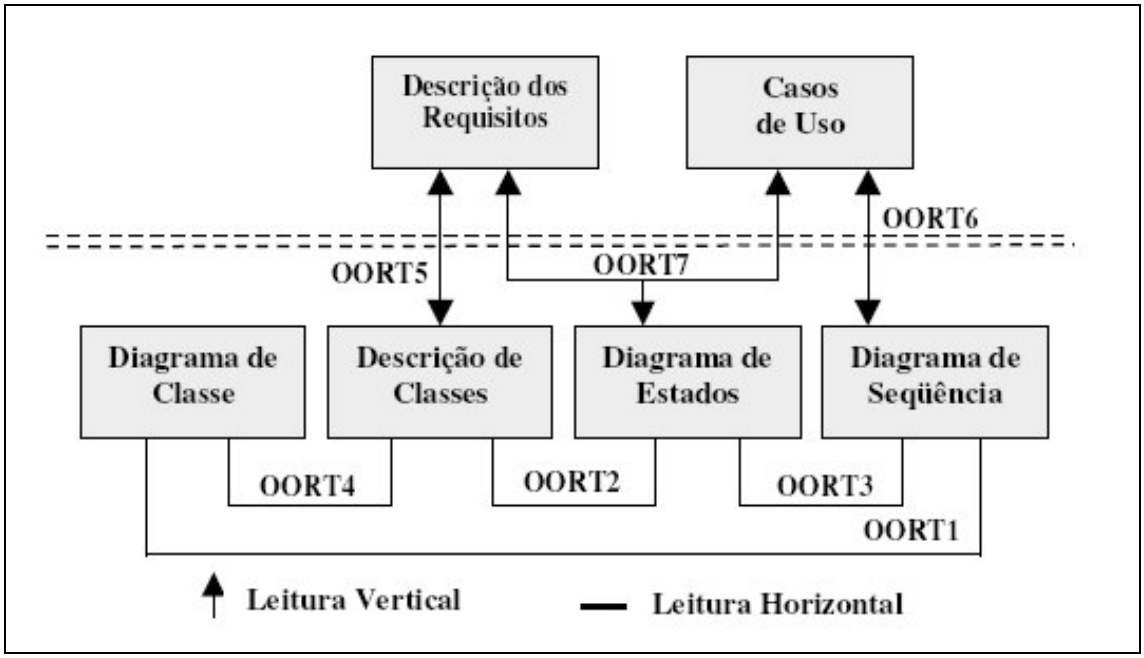


Figura 10 - Técnicas OORT

Desta forma as equipes realizaram as seguintes leituras:

Leitura Horizontal:

- Diagrama de Seqüência x Diagrama de Classes.
- Diagrama de Estados x Descrição de Classes.
- Diagrama de Seqüência x Diagrama de Estados.
- Diagrama de Classes x Descrição de Classes.

Leitura Vertical:

- Descrição de Classes x Descrição de Requisitos.
- Diagrama de Seqüência x Diagrama de Casos de Uso.
- Diagrama de Estados x (Descrição de Requisitos e Casos de Uso).

Durante a inspeção, diversas métricas foram coletadas, tanto qualitativas quanto quantitativas, sendo que as qualitativas foram obtidas através de questionários coletados juntamente com a lista de defeitos, questionários de retrospectiva no final do semestre e entrevistas pós-inspeção.

Após este primeiro estudo, outros foram desenvolvidos, destacando-se os seguintes:

a) Estudo Observacional

Realizado em 1999, na Universidade de Maryland, com 28 estudantes de engenharia de software (86% com experiência industrial com projetos orientados a objetos e 14% com experiência em sala de aula) separados em duplas (observador e executor). A técnica observacional é utilizada com o propósito de coletar dados de como uma tarefa em particular é concluída, ela consiste em observar uma pessoa ou pessoas ao executarem uma determinada tarefa. Este enfoque foi necessário para entender quais melhorias poderiam ser necessárias ao nível de passos individuais, por exemplo, onde ocorreram dificuldades ou interpretações errôneas quando aplicando a técnica, quais passos contribuíram para atingir o objetivo, e quais passos deveriam ser reordenados (TRAVASSOS, 2002).

b) Estudos de Caso

Os estudos de caso foram realizados em busca de evidências de que as técnicas poderiam ser utilizadas como parte do projeto de desenvolvimento de um software. Desta forma, foram averiguados se as técnicas necessitavam de um esforço proibitivo, se o que elas necessitam poderia ser encontrado no ambiente de desenvolvimento e se o efeito do seu uso foi útil para a continuação do desenvolvimento. Para isto dois estudos foram feitos em ambiente universitário e um terceiro em ambiente empresarial:

b.i) Estudo em Ambiente Universitário

O primeiro na Universidade de Maryland em 2000 e o segundo na Universidade do Sul da Califórnia em 2001. No primeiro a OORT foi utilizada no contexto de um modelo de ciclo de vida em cascata e no segundo em espiral, em ambos foram utilizados estudantes de engenharia de software, sendo que seus

conhecimentos e experiências (experiência industrial, experiência em sala de aula e nenhuma experiência) quanto ao projeto orientado a objetos variavam assim como nos estudos anteriores. A coleta de dados para análise foi feita através de questionários e da lista de defeitos utilizada para avaliar a eficiência das técnicas (TRAVASSOS, 2002).

b.ii) Estudo em Ambiente Empresarial

Em 2001, diferentemente dos estudos anteriores, este foi realizado em uma empresa tendo como inspetores da equipe pessoas pertencentes a determinados departamentos (TRAVASSOS, 2002). Assim como este, o estudo realizado por Bunde (2002), que será mais bem comentado adiante, é um dos poucos estudos sobre a OORT em ambiente empresarial que podem ser encontrados.

Travassos (2002) argumenta que os resultados dos estudos já realizados mostram que as técnicas já estão prontas para serem utilizadas em projetos reais. No entanto, novas experiências ainda são necessárias.

Bunde (2002) e Conradi (2003), relatam o experimento realizado junto à empresa Ericsson, no qual avaliou-se a técnica OORT aplicada à inspeção de artefatos de software.

Os resultados obtidos através da OORT foram analisados em comparação ao método R&I, já implantado na empresa.

Os autores adquiriram o conhecimento sobre o método existente na empresa, e aplicaram um questionário que apontava o nível de conhecimento dos participantes, para que desta forma fossem separados em grupos nivelados de inspetores. Os inspetores foram profissionais da empresa Ericsson, sendo que cada grupo inspecionou utilizando um dos métodos (OORT e R&I).

Com base nos trabalhos existentes (TRAVASSOS, 2001, 1999a, 1999b, 2002), os autores fizeram algumas alterações na estrutura das leituras da OORT e dos formulários de preenchimento de defeitos, utilizando-os na aplicação de seu experimento.

A principal alteração feita por Bunde (2002) ocorreu nos *checklists*. Como estes eram textuais, ou seja, textos corridos que sugeriam onde procurar defeitos nos artefatos, estes foram transformados em perguntas, para que fosse proporcionada uma leitura das OORTs mais objetiva e de fácil compreensão.

A partir do resultado desse experimento, os autores concluíram que, o método R&I, existente na empresa, e a técnica OORT auxiliavam na detecção de diferentes tipos de defeitos, o que os levou a entender que a técnica OORT poderia ser complementada com o método R&I.

A possibilidade de a técnica permitir uma complementação, levantou a hipótese de que ela poderia ainda não ser auto-suficiente. Assim, com a finalidade de averiguação desta hipótese, uma inspeção foi realizada, e seus resultados analisados. Tal inspeção foi cuidadosamente preparada, resultando no estudo empírico apresentado no próximo capítulo.

CAPÍTULO 3

UM ESTUDO EMPÍRICO SOBRE INSPEÇÃO DE ESPECIFICAÇÕES DE REQUISITOS REPRESENTADAS EM UML

3.1. PREPARAÇÃO DO ESTUDO

Ao iniciar um estudo, é importante que algumas atividades e definições sejam realizadas, ou seja, é importante que se tenha antes uma boa preparação. Desta forma, com a finalidade de obtenção de resultados consistentes, e de facilitar o entendimento do estudo, é necessário que estejam claros os procedimentos adotados (SINGER, 2004).

Optou-se, neste estudo, por seguir o processo de inspeção definido por Fagan (1976, 1986), composto pelas fases de planejamento, revisão, preparação, inspeção, retrabalho e acompanhamento. Este é o processo mais utilizado e comentado pela maioria dos autores.

Para satisfazer a necessidade de definição de uma técnica de leitura para inspeção em UML, foi adotada a Técnica de Leitura Orientada a Objetos (OORT – *Object-Oriented Reading Technique*). A técnica OORT foi inicialmente proposta por Travassos (1999a) como parte de estudos realizados na Universidade de Maryland. A OORT foi posteriormente, alvo de extensões, entre as quais se destacam as realizadas por Bunde (2002) e Conradi (2003), em um estudo realizado para a empresa Ericsson. Os trabalhos realizados por Bunde (2002) e Conradi (2003) forneceram a base principal para a definição da técnica empregada nesta dissertação de mestrado.

A técnica OORT utilizada neste estudo, enfoca o projeto de alto nível, que segundo Bunde (2002), envolve atividades realizadas após os documentos de requisitos terem sido finalizados. O projeto de alto nível captura os requisitos e emprega uma notação gráfica, que, neste caso, é a UML.

A leitura dos requisitos especificados em UML é executada verticalmente e horizontalmente. A leitura vertical compreende a leitura dos diagramas em diferentes fases do ciclo de vida e a horizontal compreende a leitura na mesma fase do ciclo de vida (BUNDE, 2002; TRAVASSOS, 2002).

A técnica OORT envolve a realização de sete tipos de leitura, classificadas em leitura vertical e leitura horizontal, descritas a seguir (BUNDE, 2002; TRAVASSOS, 2002).

a) Leitura Horizontal

- OORT – 1. Analisa o Diagrama de Seqüência em relação ao Diagrama de Classes. O objetivo desta leitura é verificar se o Diagrama de Classes do sistema descreve as classes e seus relacionamentos, de tal forma que o comportamento especificado no Diagrama de Seqüência seja corretamente capturado.

- OORT – 2. Analisa o Diagrama de Estados em relação à Descrição de Classes. O objetivo desta leitura é verificar se as classes estão definidas, para que elas possam capturar a funcionalidade especificada pelo Diagrama de Estado.

- OORT – 3. Analisa o Diagrama de Seqüência em relação ao Diagrama de Estados. O objetivo desta leitura é verificar se toda transição de estado para um objeto pode ser atingida pelas mensagens transmitidas e recebidas por aquele objeto.

- OORT – 4. Analisa o Diagrama de Classes em relação à Descrição de Classes. O objetivo desta leitura é verificar se a descrição detalhada das classes possui todas as informações necessárias, e se a descrição das classes tem senso semântico.

b) Leitura Vertical

- OORT – 5. Analisa a Descrição de Classes em relação à Descrição de Requisitos. O objetivo desta leitura é verificar se os conceitos e serviços que são descritos pelos requisitos funcionais são capturados pela descrição da classe.

- OORT – 6. Analisa o Diagrama de Seqüência em relação ao Diagrama de Caso de Uso. O objetivo desta leitura é verificar se os Diagramas de Seqüência descrevem uma combinação apropriada de objetos e mensagens que capturam a funcionalidade da especificação do Caso de Uso.

- OORT – 7. Analisa o Diagrama de Estados em relação à Descrição de Requisitos e ao Diagrama de Casos de Uso. O objetivo desta leitura é verificar se os Diagramas de Estado descrevem estados apropriados dos objetos e eventos que disparam mudanças de estados como descrito pela especificação do Caso de Uso.

As leituras são baseadas na rastreabilidade entre os diagramas, o que permite uma comparação entre dois ou três diagramas, com a finalidade de detectar defeitos. O Anexo 1 apresenta as OORTs.

Após ter definido e entendido a técnica de leitura utilizada, teve-se a preocupação de definir uma classificação de defeitos. Esta classificação foi obtida com base nos trabalhos estudados, principalmente nas propostas apresentadas em Bunde (2002), Ebenau (1994) e Travassos (2001).

Os tipos de defeitos compreendidos pela classificação (ver Anexo 4) adotada neste estudo são: omissão, fato incorreto, inconsistência, ambigüidade e informação estranha.

Tendo clara a Classificação do Defeito, foi elaborado o Formulário de Anotação de Defeitos para as leituras horizontal e vertical, a ser utilizado pelos

inspetores. A preparação deste formulário baseou-se nas propostas apresentadas por Travassos (2001, 1999a, 1999b, 2002) e Bunde (2002), com algumas adaptações (ver Anexo 2). De modo geral, o formulário contém a identificação sucinta do inspetor e dos documentos utilizados na inspeção (uma das leituras das OORTs) e tabelas informativas e de preenchimento para identificar e explicar o defeito.

Com a finalidade de colher as informações relativas às avaliações e opiniões dos especialistas em UML sobre a técnica OORT, um Formulário de Relatos Sobre Uso da Técnica OORT também foi preparado (ver Anexo 3). Este formulário contém a identificação do inspetor e um breve direcionamento para facilitar o preenchimento do restante. Optou-se por permitir uma livre escrita das opiniões, visando dar maior flexibilidade de expressão aos especialistas.

Os inspetores foram escolhidos de forma intencional, ou seja, a partir de um critério, que pode ser definido como especialistas em UML, mais especificamente, profissionais que trabalham ou trabalharam por um período mínimo de dois anos com a linguagem. Com isto, obteve-se um número de 5 profissionais que puderam participar e contribuir neste trabalho.

Dando continuidade à preparação do estudo, ocorreu a injeção de defeitos nos artefatos inspecionados, para assim verificar a eficiência da OORT na identificação destes. Os artefatos inspecionados neste estudo foram diagramas UML referentes à especificação de um Sistema de Controle de Matrícula Acadêmica (SOUZA, 2003).

Como penúltimo passo, foram preparados os *checklists* para cada uma das sete leituras que compõem a OORT utilizada no estudo empírico proposto. Estes *checklists* guiariam os inspetores a detectar defeitos, através de questões elaboradas de acordo com uma classificação de defeitos (KIRNER, 1997). A partir desta definição e tendo já determinado uma classificação de defeitos, foram analisados os modelos de *checklists* de Travassos (2002) e Bunde (2002), sendo decidido pelo modelo proposto por Bunde (2002), com

algumas modificações, pois este proporcionava uma leitura ágil, menos cansativa para o leitor, além de facilitar a identificação dos defeitos.

O passo final diz respeito à realização de um teste piloto, ou seja, foi feito um teste inicial referente ao material que foi produzido para a execução da inspeção. Este teste é descrito a seguir.

3.2. REALIZAÇÃO DO TESTE PILOTO

O teste piloto teve como objetivo avaliar o material preparado, para a realização da inspeção utilizando a técnica OORT. Esta avaliação se fez necessária e importante, pois ela averiguaria a auto-suficiência do que foi desenvolvido, em termos de compreensão e de utilização. O teste piloto permite a identificação de problemas, possibilitando também a inclusão de melhorias, antes do material de inspeção ser enviado para o restante dos participantes da pesquisa. Um outro objetivo deste teste piloto foi obter uma maior garantia de que a pesquisa seria bem sucedida.

Para o teste piloto, um participante, que possuía conhecimento em UML, foi escolhido e contatado.

Após o recebimento do material, o participante levou cerca de duas semanas para retornar os resultados. Destaca-se aqui, que este participante não solicitou qualquer ajuda, tendo executado a inspeção e respondido todos os formulários, de acordo com seu entendimento, o que era muito importante para a análise deste teste piloto.

O participante do teste piloto achou 12 dos 47 defeitos injetados nos artefatos, ou seja, aproximadamente 25,5% do que foi injetado. Apesar de não ter sido um excelente resultado, o teste piloto cumpriu seu objetivo apontando dificuldades que ocorreram e melhorias que foram consideradas, gerando,

desta forma, uma nova versão do material, enviado posteriormente aos especialistas em UML.

Os problemas identificados a partir da análise do resultado do teste piloto foram:

- Dificuldades de entendimento da OORT.
- Dificuldades sobre o entendimento de conceitos da UML.
- Dificuldades na interpretação da tabela de “Classificação de Defeitos”.
- Indefinição quanto ao que preencher no campo “Nome” dos “Formulários de Anotação de Defeitos”.
- Dificuldades na interpretação da tabela de “Severidade do Defeito”.
- Necessidade de utilização do documento de requisitos em leituras onde este não faz parte.
- Dificuldades em entender o porquê da utilização da Descrição de Classes de apenas uma classe.
- Indefinição quanto ao que preencher no campo “Requisitos” do “Formulário de Anotação de Defeitos”.
- As tabelas de “Severidade do Defeito” e de “Classificação de Defeitos” estavam em uma determinada ordem nas definições e invertidas no preenchimento de discrepâncias nos “Formulário de Anotação de Defeitos”.
- Seguindo as leituras seqüencialmente, ou seja, da OORT 1 a OORT 7, os conhecimentos obtidos do Diagrama de Estados faziam com que a matriz de adjacência (MA) desenvolvida na OORT 7 parecesse tendenciosa.

- Necessidade de exemplos no preenchimento do “Formulário de Anotação de Defeitos”.

Procurando solucionar os problemas destacados acima, as seguintes ações foram tomadas:

- Revisão das OORTs, visando tornar seus procedimentos mais claros.

- Checagem geral dos defeitos, averiguando se um defeito injetado com o objetivo de ser encontrado em uma determinada leitura, não levaria à indicação de outro defeito em uma outra leitura.

- Inclusão de uma explicação no arquivo “Instruções”, relativo ao que preencher no campo “Nome”, que passou a se chamar “Palavra Chave” no “Formulário de Anotação de Defeitos”.

- Redefinição dos conceitos da tabela de “Severidade do Defeito” que passou a se chamar “Grau de Severidade do Defeito”, para evitar interpretação errônea ou ambigüidade nas definições.

- Reformulação da explicação existente no arquivo “Instruções”, relativo à utilização do lápis vermelho para identificação e numeração de requisitos na Leitura Vertical (OORT 5, OORT 6 e OORT 7).

- Destaque de informações no arquivo “Instruções”, para facilitar o entendimento da OORT e o preenchimento dos formulários.

- Redefinição da tabela de “Classificação de Defeitos”, que passou a se chamar “Classificação do Defeito”, evitando má interpretação ou ambigüidade nas definições, sendo também incluído alguns exemplos;

- Reorganização do “Formulário de Anotação de Defeitos” das Leituras Horizontal e Vertical, procurando proporcionar uma seqüencialidade no preenchimento das informações.

- Inclusão de uma explicação mais detalhada, no arquivo “Instruções”, sobre a documentação do sistema que seria inspecionado.

3.3. REALIZAÇÃO DO ESTUDO

O estudo envolveu o planejamento e inspeção propriamente dita.

Nesta etapa de planejamento foram realizadas as seguintes atividades:

- Reestruturação das OORTs, que ficaram conforme apresentadas no Anexo 1.

- Definição do Formulário de Anotação de Defeitos para a Leitura Horizontal (ver Anexo 2).

- Definição do Formulário de Anotação de Defeitos para a Leitura Vertical (ver Anexo 2).

- Definição do Formulário de Relato Sobre Uso da Técnica OORT, para identificação dos especialistas, assim como seus níveis de conhecimento e suas opiniões a respeito da técnica (ver Anexo 3).

- Realização de contato com 5 especialistas em UML, que participaram como inspetores durante o processo.

- Preparação da Tabela de Classificação do Defeito (ver Anexo 4), com as definições de cada classificação.

- Preparação dos artefatos (especificação do sistema de controle de matrícula acadêmica) com defeitos injetados para a inspeção.

- Envio do material aos participantes da pesquisa.

Para realizar a Inspeção, cada inspetor recebeu a documentação do sistema de controle de matrícula (SOUZA, 2003). Os diagramas UML foram inspecionados individualmente, por cada um dos inspetores, especialistas em UML, utilizando a técnica OORT com seus sete tipos de leituras, os formulários de identificação de defeitos, o formulário de relato, uma classificação de defeitos e as instruções para a realização da inspeção.

3.4. PERGUNTAS DA PESQUISA E METODOLOGIA

As perguntas que o estudo buscou responder foram: *Qual o nível de eficiência das técnicas OORT, em termos de quantidade de defeitos detectados em especificações de software representadas através de Unified Modeling Language? E qual é o nível de satisfação dos especilistas que inspecionaram a especificação UML através das técnicas OORT, em relação a aspectos importantes envolvidos na aplicação dessas técnicas?*

O estudo realizado configura um estudo experimental descritivo, que, futuramente, poderá ser generalizado por metodologias de inferência, considerando-se, desta forma, dois campos básicos da estatística: a estatística descritiva e a inferência estatística. Segundo Levine (2000), a estatística descritiva inclui métodos que descrevem as várias características de um conjunto de dados. A inferência estatística, por sua vez, refere-se aos métodos que tornam possível a estimativa de uma característica de uma população ou a tomada de decisão referente à população, com base em resultados da amostra.

Conforme Levine (2000) e Mattar (1997), é importante registrar que população, no sentido estatístico, é a totalidade dos itens ou objetos considerados, e que a amostra é qualquer parte dessa população. Quando se realiza um processo de escolher amostras de uma população efetua-se uma amostragem.

Existem duas possibilidades quando se deseja conhecer aspectos referentes a uma população ou universo estatístico, um deles é o censo, onde todos os elementos são pesquisados, e o outro é a amostra, onde se estimam dados a respeito da população.

Para o presente estudo, realizar uma pesquisa a partir de um censo seria inviável, pois a população de especialistas em UML, apesar de ser seleta, não é pequena e os dados não podem ser facilmente obtidos. Conseqüentemente optou-se por uma amostragem.

Segundo Mattar (1997), para o processo de seleção ou coleta de amostras, alguns passos devem ser seguidos, onde se define uma população de pesquisa, identificam-se todas as unidades amostrais e o tamanho da amostra, seleciona-se um procedimento específico do qual a amostra será determinada e por fim a seleção física da amostra.

Existem dois tipos de amostragem, a amostragem probabilística e a amostragem não probabilística. Apesar das amostras probabilísticas serem as que fornecem estimativas com maior precisão sobre a população, optou-se pela amostragem não probabilística.

Para o presente estudo, adotou-se uma amostragem não probabilística, devido este ser um estudo que avalia, a partir da opinião de especialistas em UML, uma proposta recente de uma técnica de inspeção, que é a OORT. Desta forma, os especialistas que trabalharam como inspetores durante este estudo foram escolhidos de forma intencional, a partir de um critério, que exigia a participação de pessoas que trabalham ou trabalharam por um período acima

de dois anos com a linguagem UML. Isto caracteriza uma amostragem não probabilística intencional.

Conhecida a população e determinada a amostra, o material desenvolvido para a inspeção e coleta dos dados foi distribuído aos participantes. Para esta distribuição foram enviados e-mails aos que ficaram impossibilitados de comparecer a uma reunião e, para os demais participantes, o material foi entregue em mãos, seguido de uma reunião. As respostas, foram recebidas através de e-mail (ver VOTTA, 1993), com exceção de um participante, que entregou pessoalmente o material.

As respostas obtidas foram analisadas e a partir delas foram preparados gráficos, tabelas e observações. Esta abordagem constituiu-se em uma análise de dados qualitativos amparada em representações visuais (PEREIRA, 1999). A escala de Likert (PEREIRA, 1999) foi utilizada no Formulário de Relato Sobre Uso da Técnica OORT (ver Anexo 3). No qual as opiniões dos participantes a respeito da técnica foram capturadas, através de questões onde as respostas variavam de 1 a 5 (onde 1 representava a menor satisfação e 5 a maior satisfação).

Para a análise dos dados, algumas questões foram agrupadas, simplificando as observações ou pré-conclusões e tornando-as mais objetivas. Tanto as tabelas como os gráficos foram apresentados segundo as normas científicas (VIEIRA, 1991).

3.5. ANÁLISE DOS RESULTADOS

Para este estudo, oito profissionais, especialistas em UML, foram contatadas, das quais cinco se dispuseram a realizar a inspeção e responder os formulários enviados junto ao restante do material. Estes profissionais possuíam mais de 3 anos de experiência com o uso de UML, sendo que nenhum deles conhecia as técnicas OORT.

A partir do material preenchido pelos participantes, com as respostas em cada formulário, uma análise detalhada foi realizada focando dois pontos. O primeiro, diz respeito aos resultados obtidos através dos formulários de defeitos de cada uma das leituras da OORT e o segundo sobre o formulário de relato sobre a OORT.

a) Análise da Eficiência das Técnicas OORT

Nesta primeira análise, sobre os formulários para preenchimento de defeitos encontrados a partir da aplicação da inspeção através da OORT, pode-se observar que a maioria dos defeitos injetados nos diagramas e documentos foram encontrados. Houve uma única exceção relativa à OORT 7, onde dos seis defeitos injetados, nenhum foi encontrado. A figura 11 ilustra os defeitos injetados e os defeitos detectados, através de cada OORT.

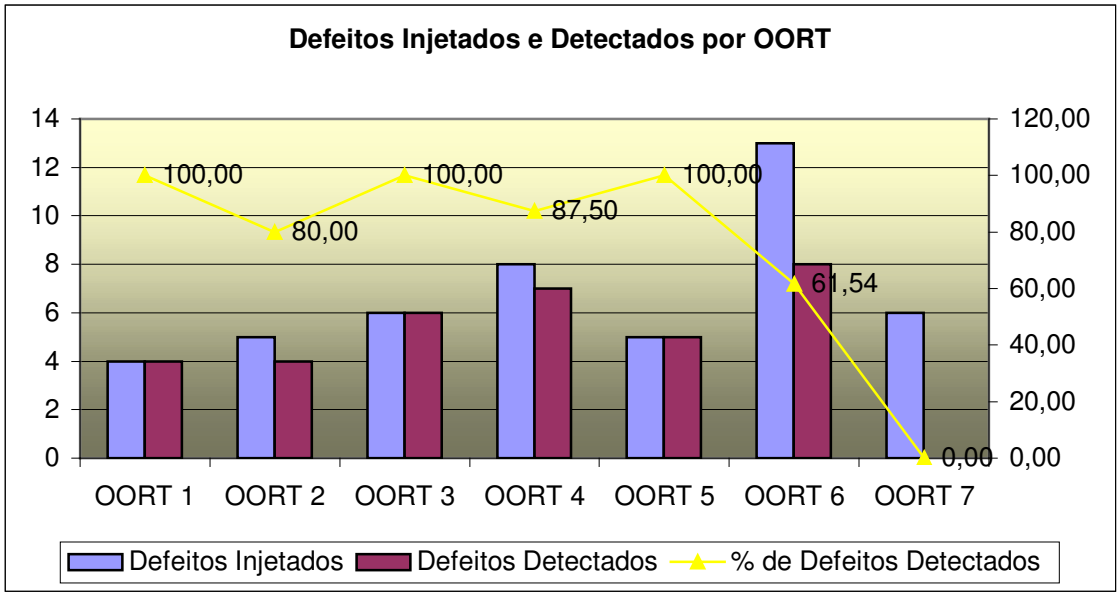


Figura 11 - Defeitos injetados e Detectados por OORT

Ainda com relação à Figura 11, pode-se observar que, em três casos, ocorreram 100% dos defeitos encontrados, através das OORT 1, OORT 3 e OORT 5. Os maiores índices percentuais dos defeitos encontrados ocorreram

através das OORTs que fazem parte da Leitura Horizontal onde se encontram 49% dos defeitos injetados (ver Figura 12 – primeiro gráfico). Ressalta-se ainda que 27% dos defeitos injetados (ver Figura 12 – segundo gráfico) não foram encontrados, defeitos estes que, na sua maioria deveriam ser detectados pelas OORTs referentes à Leitura Vertical. Isto pode ser observado na Figura 12 em gráficos comparativos.

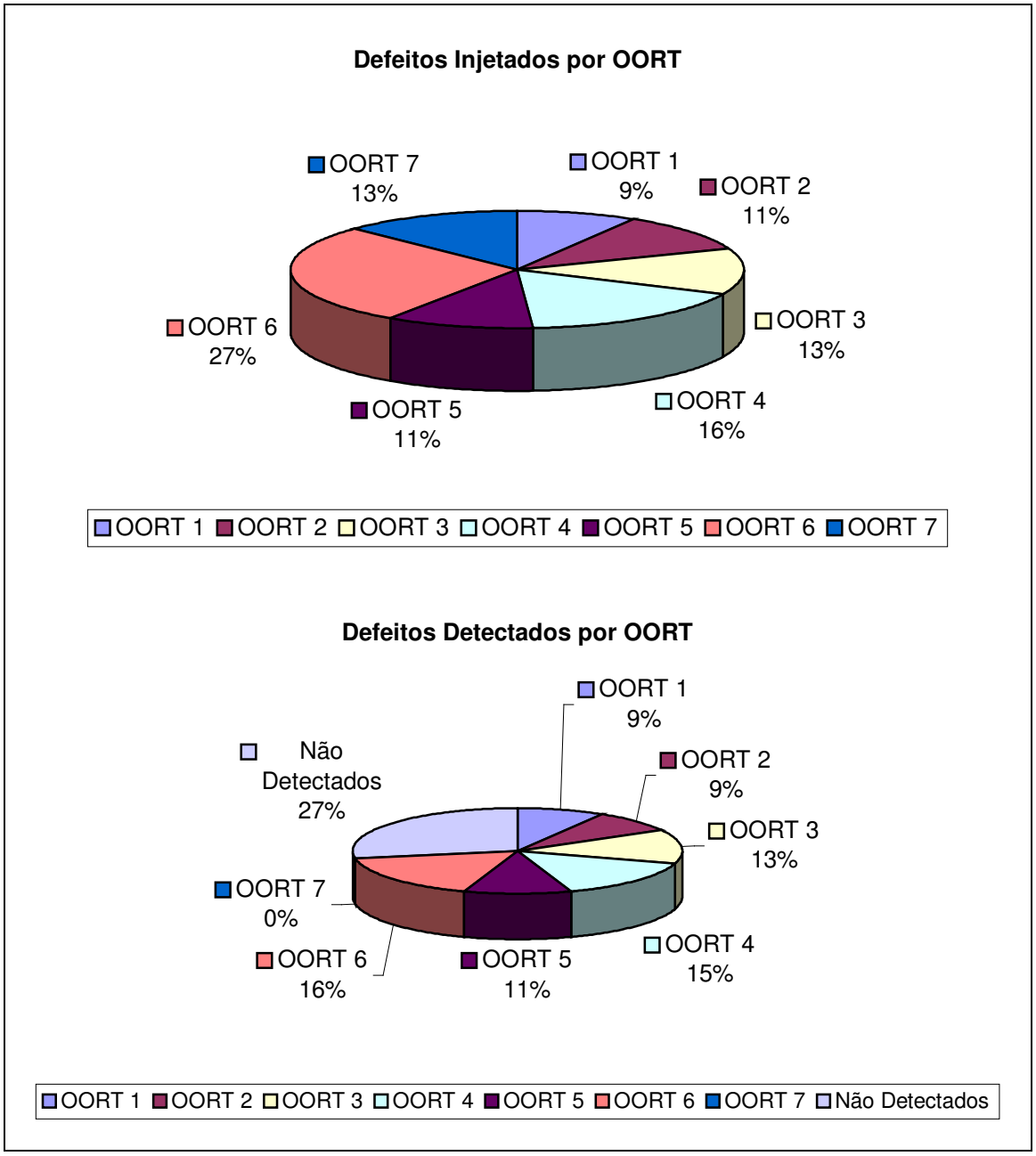


Figura 12 - Defeitos injetados por OORT em relação aos Defeitos Detectados por OORT

O gráfico mostrado na figura 13 apresenta os defeitos injetados e detectados, de acordo com a classificação adotada (ver definições em Anexo 4). O gráfico indica que os maiores percentuais de defeitos encontrados ocorreram para as classificações de Inconsistência (88,24%), Ambigüidade (100%) e Informação Estranha (75%), ocorrendo também um índice médio para Omissão (50%). O índice mais baixo ocorreu para os defeitos do tipo Fato Incorreto (37,50%).

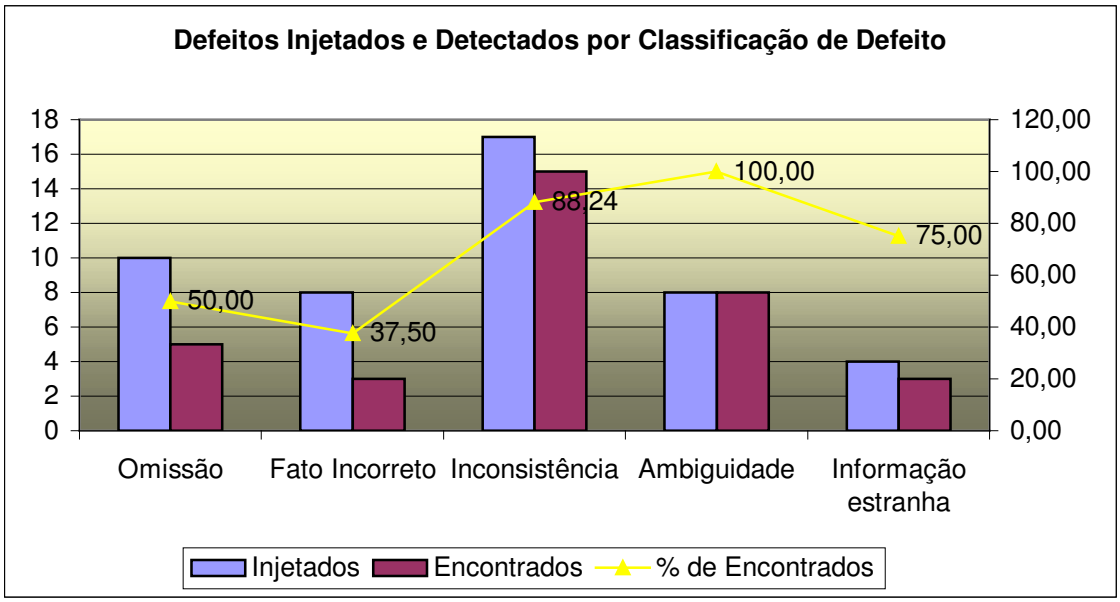


Figura 13 - Defeitos injetados e Detectados por Classificação

Apesar de existir um diferencial nas quantidades de defeitos injetados, a Figura 14 sugere que este não foi um fato decisivo para as quantidades de defeitos encontrados. Altos índices dos encontrados foram observados, como por exemplo, nas classificações de Inconsistência e de Ambigüidade, onde respectivamente representam 36% e 17% (ver Figura 14 – primeiro gráfico) dos defeitos injetados. Ressalta-se ainda que 27% dos defeitos injetados (ver Figura 14 – segundo gráfico) não foram encontrados pelos especialistas. Estes defeitos, na sua maioria, estavam nas classificações referentes a Omissão e a Fato Incorreto, como pode ser observado na Figura 14 em gráficos comparativos.

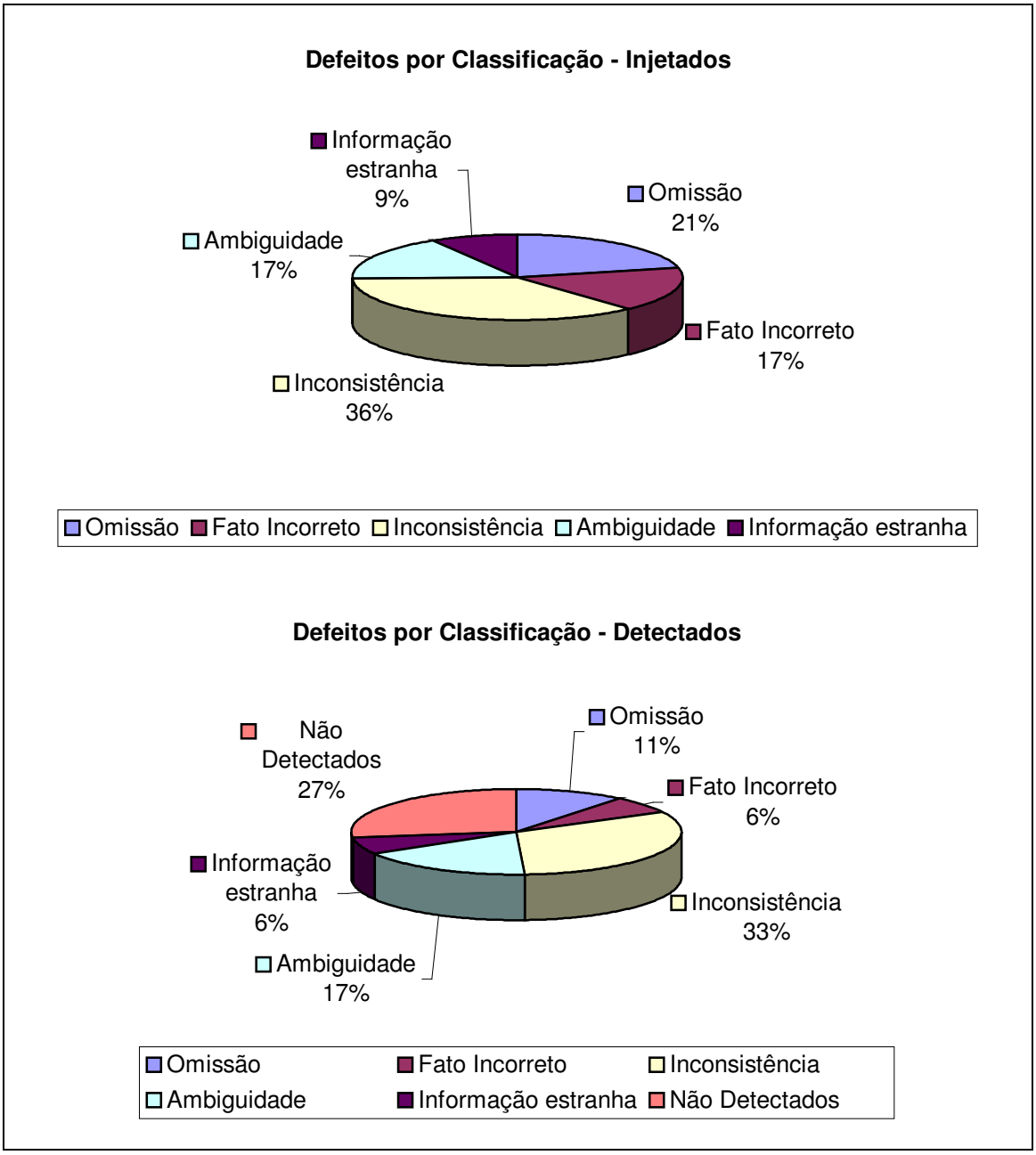


Figura 14 - Defeitos injetados em relação aos Defeitos Detectados por Classificação

Como a OORT aborda dois tipos de leitura, a Horizontal (OORT 1, OORT 2, OORT 3 e OORT 4) e a Vertical (OORT 5, OORT 6 e OORT 7), a Figura 15 mostra um gráfico composto de informações das quantidades de defeitos injetados em cada uma destas leituras, assim como as quantidades dos defeitos detectados, com seus respectivos percentuais. Através desta figura, verifica-se um percentual elevado de defeitos detectados pela Leitura

Horizontal, atingindo 91,30%. Um percentual mais baixo é verificado para a Leitura Vertical, atingindo 54,17%.

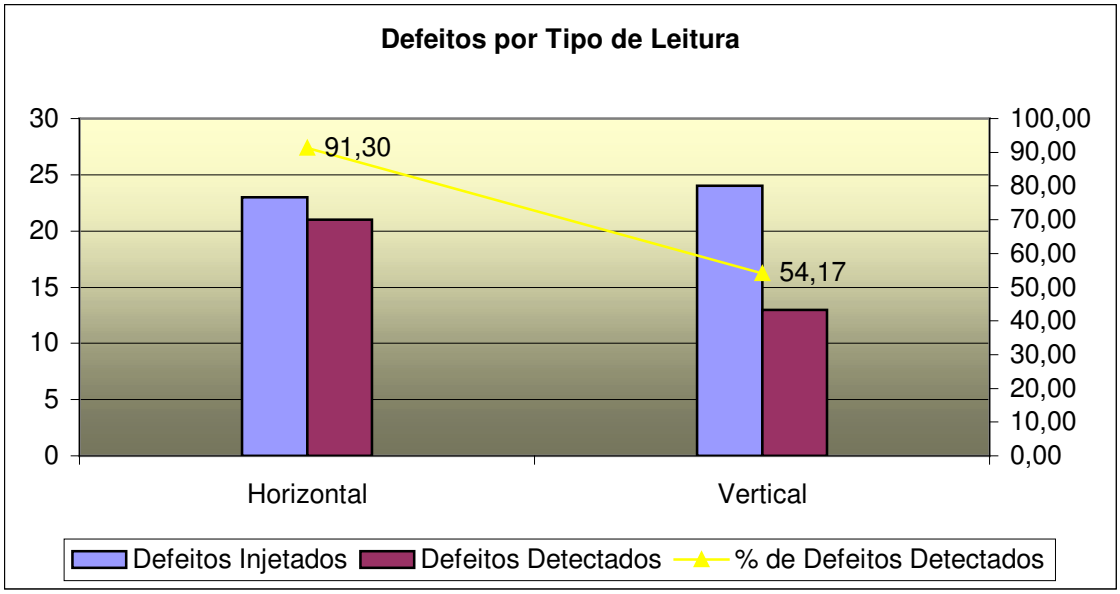


Figura 15 - Defeitos Injetados e Detectados nas Leituras Horizontal e Vertical

A partir da Figura 16 em que pode ser observado os resultados dos defeitos encontrados pela Leitura Horizontal, de acordo com a Classificação do Defeito, observa-se que os defeitos injetados, foram encontrados na sua maioria, gerando desta forma altos índices percentuais para as classificações de Omissão (100%), Ambigüidade (100%) e Informação Estranha (100%), com exceção somente para a classificação de Fato Incorreto onde nenhum defeito foi injetado para a leitura em questão.

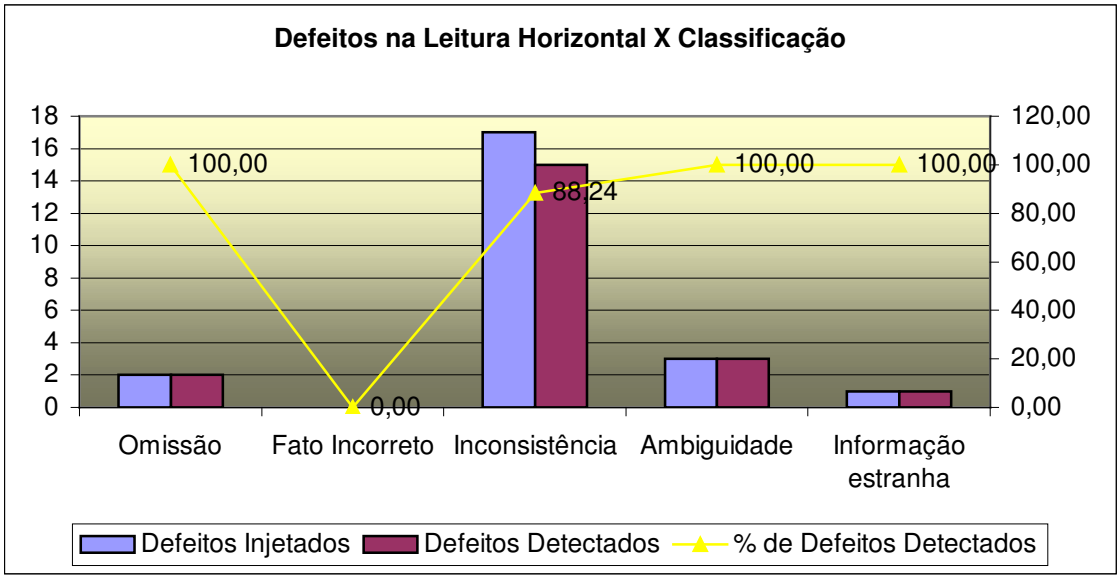


Figura 16 - Defeitos Injetados e Detectados da Leitura Horizontal segundo Classificação

Com relação à Figura 17, mas focando o resultado para a Leitura Vertical, pode-se observar que os resultados não foram tão satisfatórios como para a Leitura Horizontal, pois índices percentuais mais baixos foram obtidos, como para as classificações de Omissão (37,50%) e Fato Incorreto (37,50%), com exceção somente para a classificação de Inconsistência, onde nenhum defeito foi injetado para a leitura em questão.

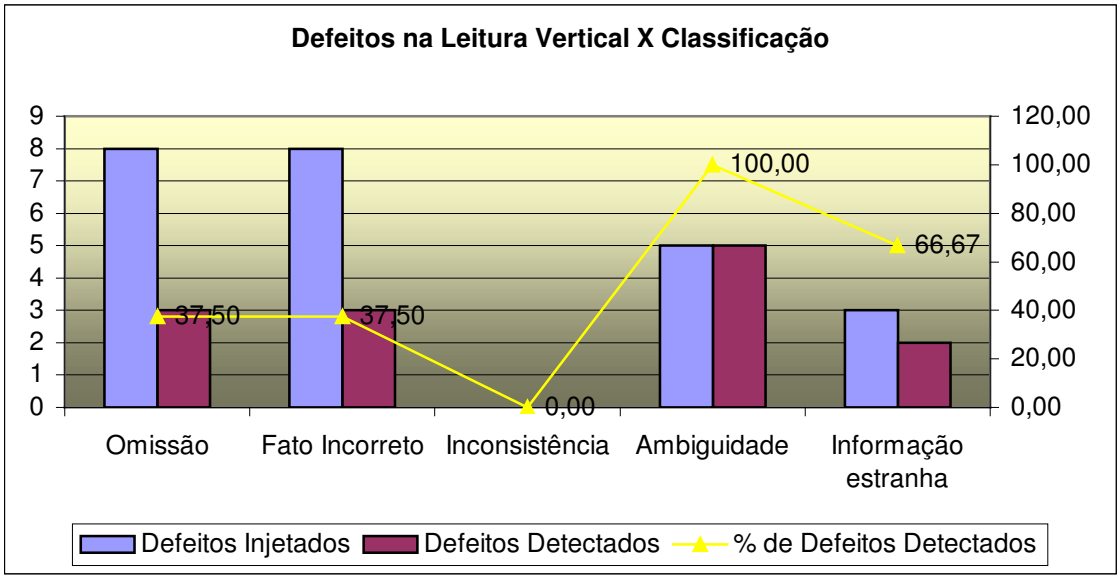


Figura 17 - Defeitos Injetados e Detectados da Leitura Vertical segundo Classificação

É importante informar que os defeitos comentados e expostos em gráficos e tabelas foram detectados pelos especialistas e anotados por estes nos formulários fornecidos para a inspeção, mas nem todos classificaram os defeitos de forma correta. Além disso, alguns defeitos corretamente encontrados não foram injetados para a inspeção ou não foram encontrados pela OORT prevista, desta forma representando um total de 14 defeitos, sendo que destes 14 defeitos, 7 foram injetados, mas foram identificados por uma OORT que não era a planejada para encontrá-lo.

b) Análise da Utilização das Técnicas OORT

Nesta segunda análise, a respeito do Formulário de Anotações de Defeitos Sobre Uso da OORT, algumas questões foram agrupadas, para efeito de análise, devido à relação de um tópico em comum. Isto se deve, tanto para facilitar a análise como para o entendimento dos resultados.

As questões 1 e 2 referiam-se a suficiência de conhecimento dos especialistas em relação à aplicação da técnica, ou seja, se estes consideraram ter conhecimento suficiente para sua aplicação, respectivamente sobre inspeção e sobre UML. A partir da Figura 18, pode-se observar que este não foi

considerado um obstáculo, pois, em ambas as questões, as respostas, em sua maioria, foram da categoria de muita satisfação, atingindo 60%, e satisfação (questão 2) ou média satisfação (questão 1) atingindo 40%.

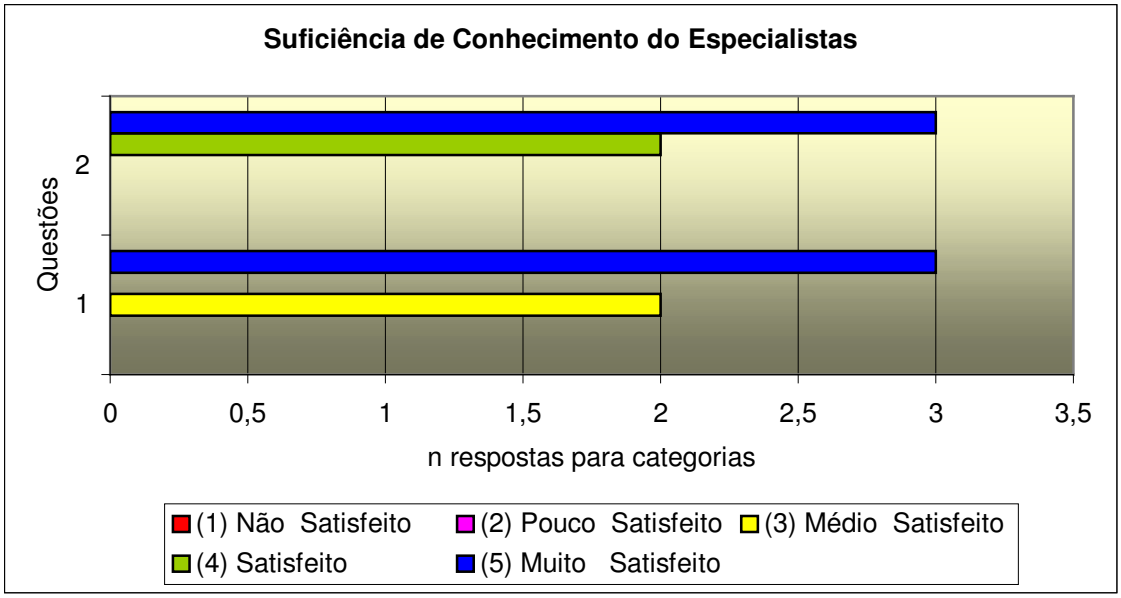


Figura 18 - Suficiência de Conhecimento

A Figura 19 corresponde à avaliação da Tabela de Classificação dos Defeitos fornecida para a inspeção. Esta figura mostra, respectivamente pelas questões 3 e 4, o entendimento da classificação e a sua completude. É perceptível que, quanto ao entendimento (questão 3), os índices mostram uma satisfação por parte dos especialistas a 80%. Quanto à completude, houve pouca satisfação, atingindo 60% das respostas.

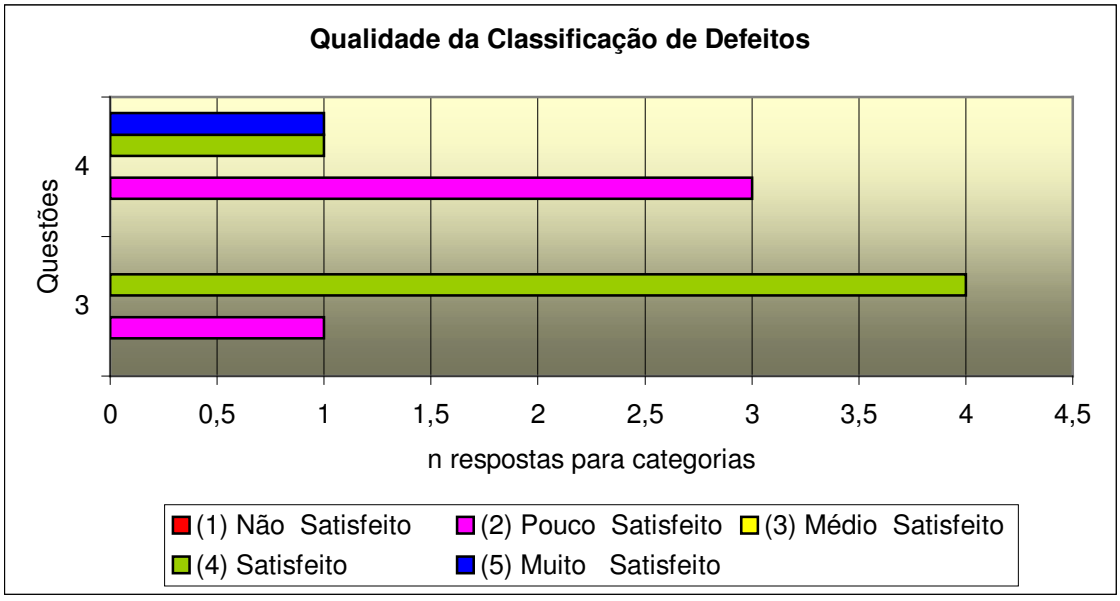


Figura 19 - Avaliação da Classificação de Defeitos

Para a avaliação dos Tipos de Defeitos, as mesmas questões de entendimento e completude foram feitas, respectivamente questões 5 e 6. A Figura 20 mostra, como resposta de maior freqüência, a categoria de média satisfação representando em ambas as questões 40% das respostas. Evidencia-se também uma tendência para uma maior satisfação, quando observado que outros 40% estão igualmente divididos entre as categorias de satisfação e de muita satisfação.

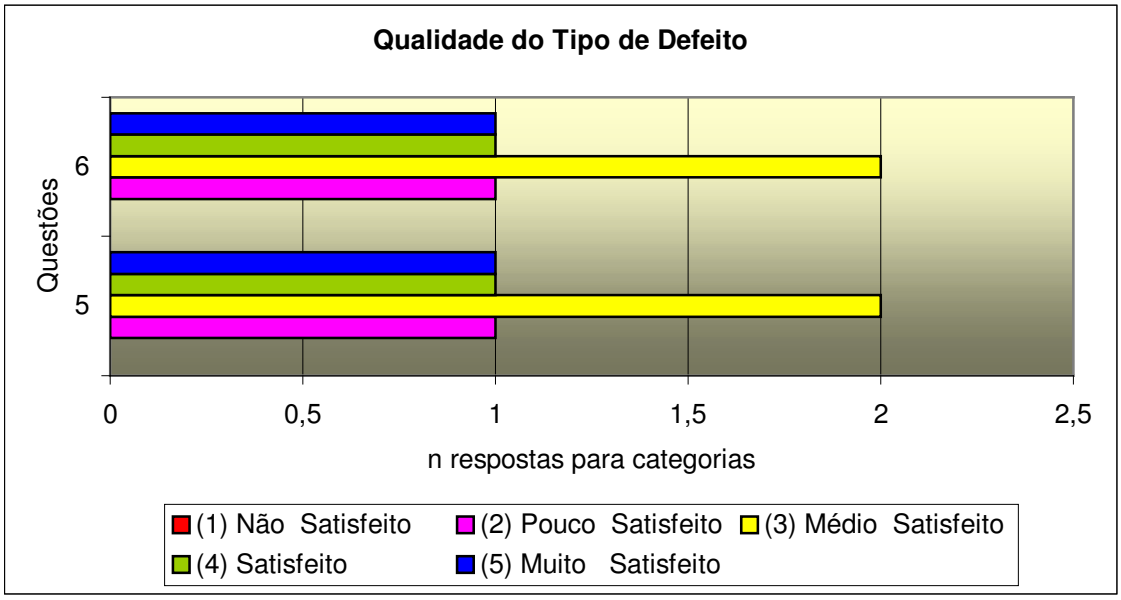
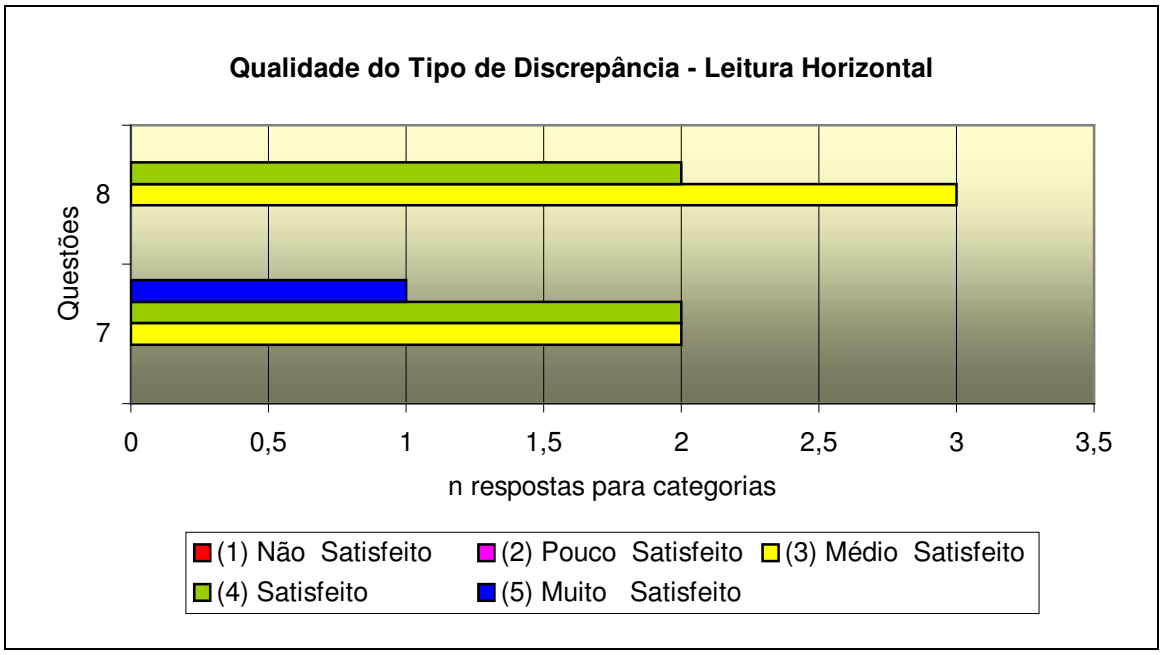


Figura 20 - Avaliação do Tipo de Defeito

A Figura 21 mostra o resultado da avaliação do Tipo de Discrepância para as Leituras Horizontal (Figura 21 – primeiro gráfico) e Vertical (Figura 21 – segundo gráfico), novamente quanto ao entendimento (questões 7 e 9) e quanto à completude (questões 8 e 10). Para a Leitura Horizontal, os índices são considerados bons, pois em ambas as questões, as respostas estiveram acima da categoria de média satisfação, chegando a 40% na categoria de satisfação. A Leitura Vertical também obteve uma boa avaliação quanto aos temas abordados pelas questões, havendo somente uma exceção correspondendo a 20% de pouca satisfação para a questão relativa à completude.



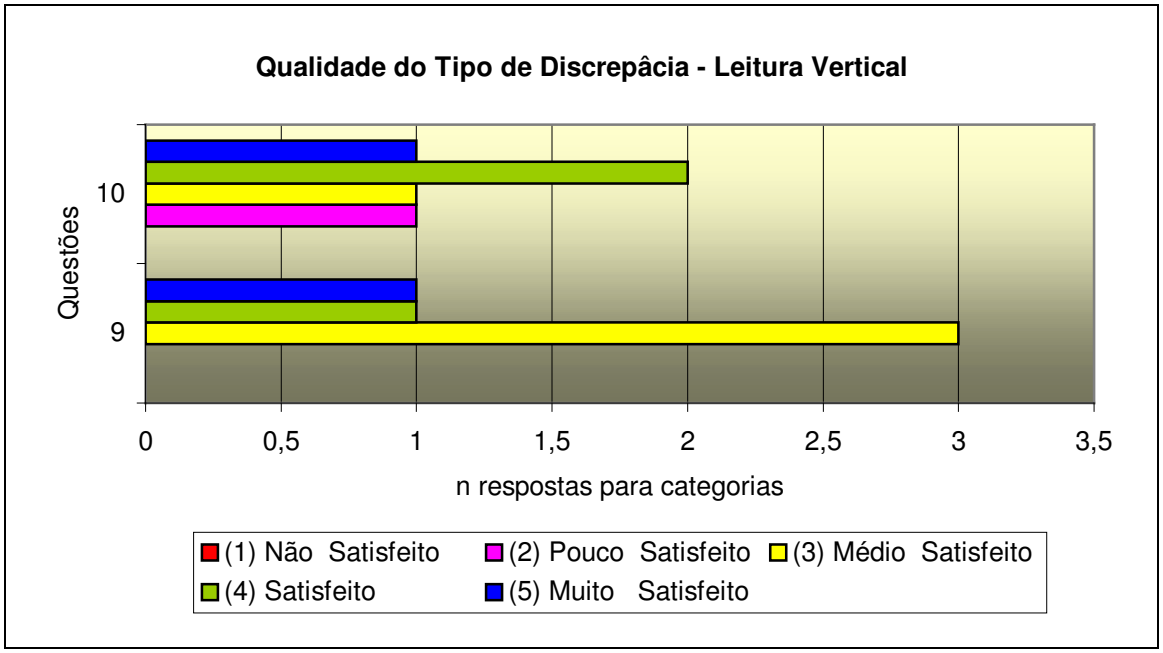


Figura 21 - Avaliação do Tipo de Discrepância

O Grau de Severidade do Defeito também foi avaliado, através de questões de entendimento e completude, respectivamente questões 11 e 12. Na Figura 22 pode-se perceber que, quanto ao entendimento deste grau de severidade, a categoria apontada com maior freqüência foi a de muita satisfação, atingindo 80%. Quanto à completude, a categoria de maior freqüência foi a de pouca satisfação, que obteve um índice de 40% das respostas.

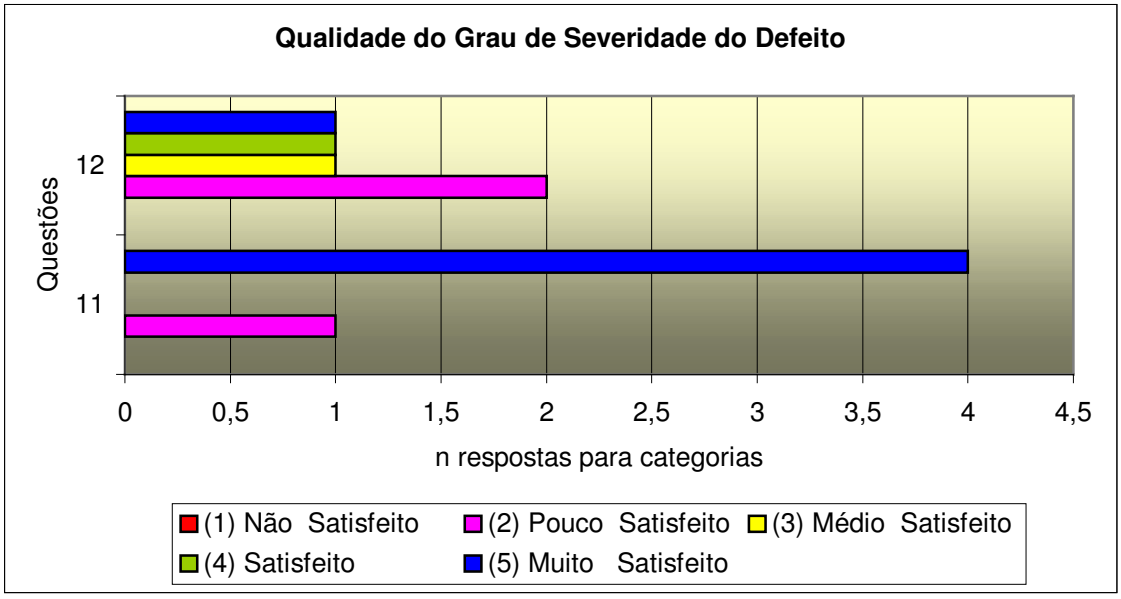
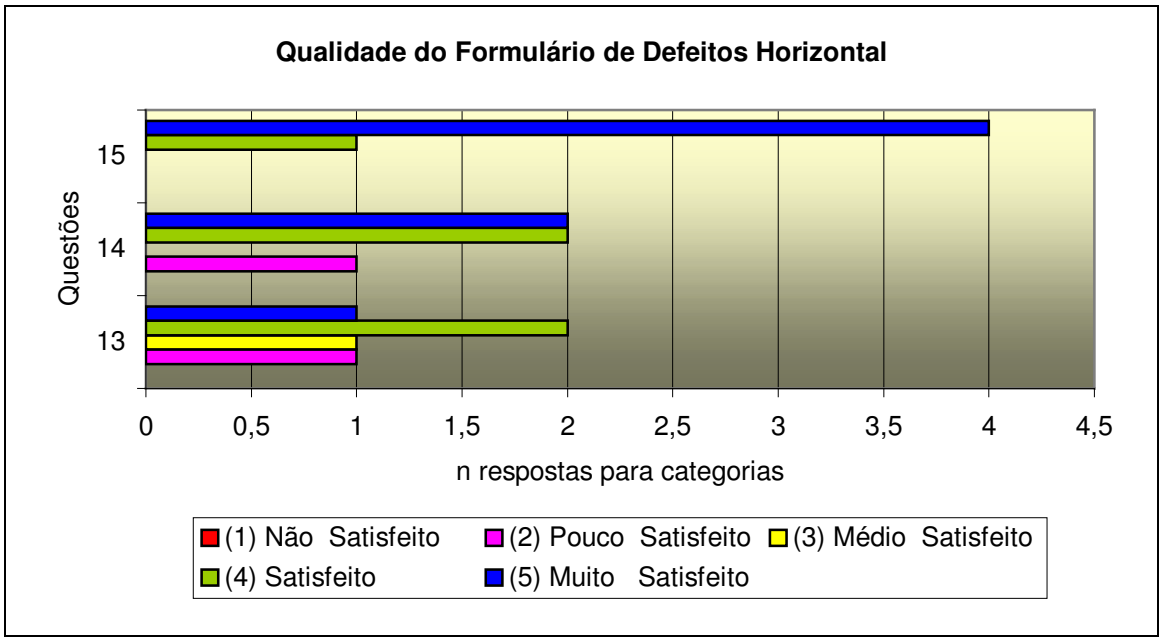


Figura 22 - Avaliação do Grau de Severidade do Defeito

Os Formulários de Anotação de Defeitos para a Leitura Horizontal e para a Leitura Vertical, além de serem avaliados quanto ao entendimento (questões 13 e 16) e completude (questões 15 e 18), foram também avaliados quanto à facilidade de preenchimento (questões 14 e 17). Ambas as leituras tiveram uma boa avaliação (Figura 23), ocorrendo maiores freqüências de respostas para as categorias de satisfação e muita satisfação, em todas as questões. Houve pequenas diferenças para as questões de entendimento e de facilidade de preenchimento, onde foram apontadas as categorias de média satisfação e pouca satisfação ao menos uma vez, representando somente 20% das respostas a categoria de pouca satisfação.



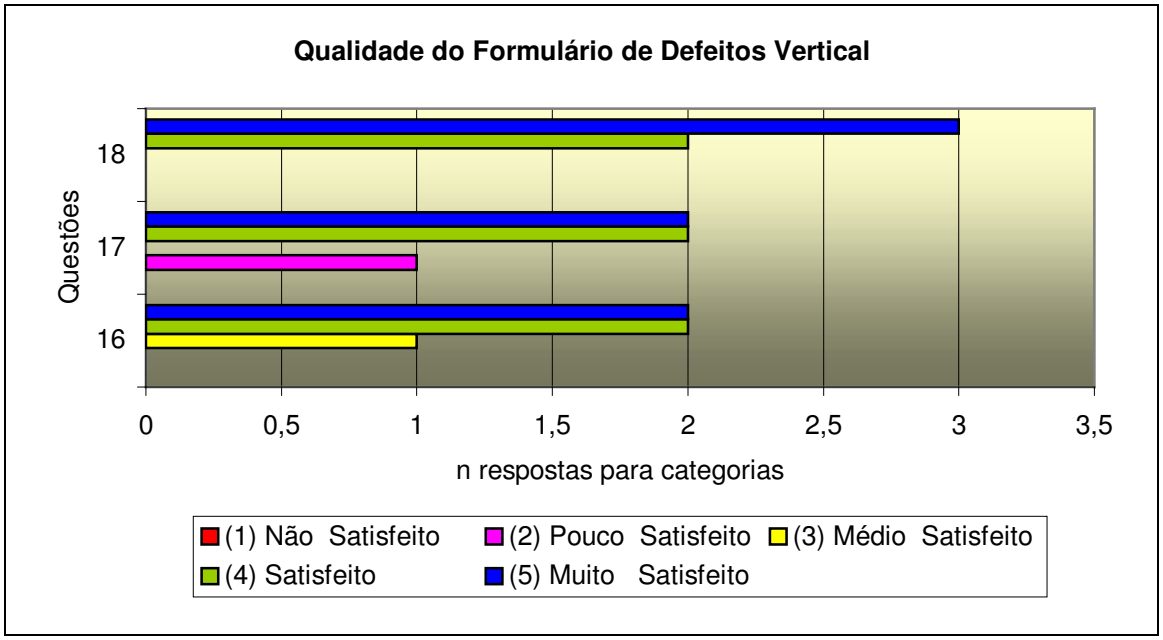
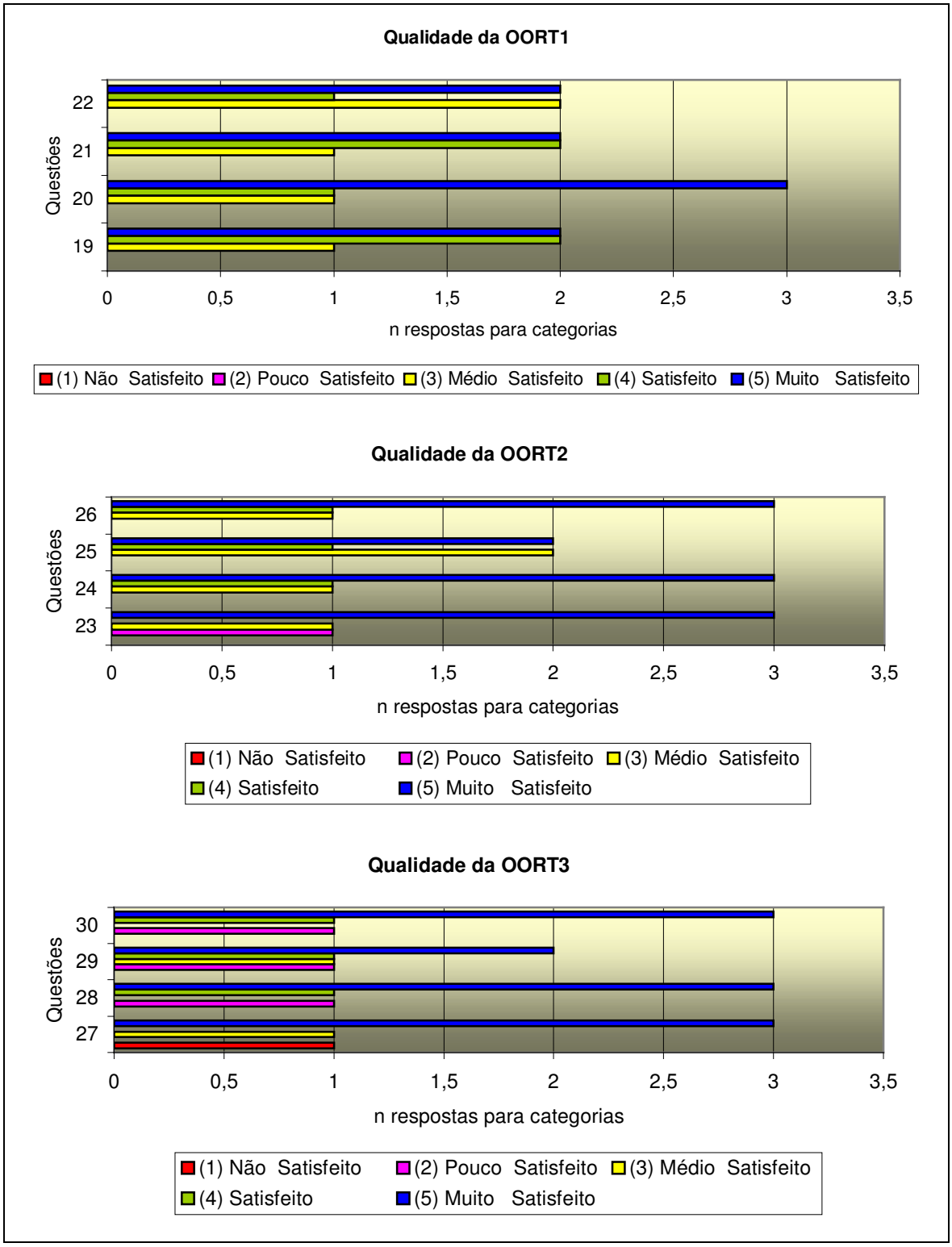


Figura 23 - Avaliação do Formulário de Defeitos para Leitura Horizontal e Vertical

A Figura 24 apresenta quatro gráficos, cada qual referente a uma OORT das que compõe a Leitura Horizontal, considerando, desta forma, a OORT 1, a OORT 2, a OORT 3 e a OORT 4. Em cada gráfico, são representadas as respostas ocorridas para as perguntas referentes aos *checklists* de cada uma. As perguntas enfocavam a compreensão (questões 19, 23, 27 e 31), a necessidade (questões 20, 24, 28 e 32) e a suficiência (questões 21, 25, 29 e 33) das questões existentes nos *checklists* e, além destas três, mais uma referente à suficiência de artefatos abordados nos *checklists* (questões 22, 26, 30 e 34) para realizar uma OORT.

Observando os resultados na Figura 24, para todas as perguntas feitas no Formulário de Relato Sobre Uso da Técnica OORT, é perceptível que as categorias de satisfação e muita satisfação são predominantes, ocorrendo, para todas, um índice de no mínimo 40% de muita satisfação. Mas, é importante frisar que índices de pouca satisfação e de não satisfação, atingindo o valor de 20%, apareceram nas OORTs 2, 3 e 4. Destaca-se a OORT 3, onde pelo menos um destes índices apareceram em cada uma das perguntas do relato, sendo, conseqüentemente, a OORT da Leitura Horizontal de pior avaliação.


Qualidade da OORT1
Questões
22
21
20
19
0
0,5
1
1,5
2
2,5
3
3,5
n respostas para categorias
(1) Não Satisfeito
(2) Pouco Satisfeito
(3) Médio Satisfeito
(4) Satisfeito
(5) Muito Satisfeito
Qualidade da OORT2
Questões
26
25
24
23
0
0,5
1
1,5
2
2,5
3
3,5
n respostas para categorias
(1) Não Satisfeito
(2) Pouco Satisfeito
(3) Médio Satisfeito
(4) Satisfeito
(5) Muito Satisfeito
Qualidade da OORT3
Questões
30
29
28
27
0
0,5
1
1,5
2
2,5
3
3,5
n respostas para categorias
(1) Não Satisfeito
(2) Pouco Satisfeito
(3) Médio Satisfeito
(4) Satisfeito
(5) Muito Satisfeito

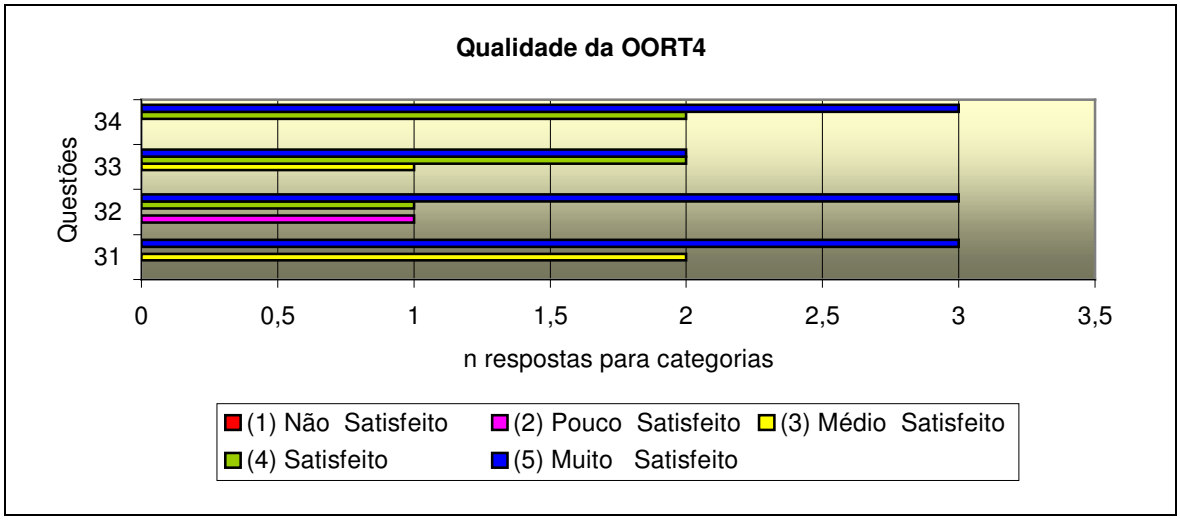


Figura 24 - Avaliação dos Checklists das OORTs da Leitura Horizontal

A Figura 25 apresenta três gráficos, cada qual referente a uma OORT das que compõe a Leitura Vertical, considerando desta forma a OORT 5, a OORT 6, e a OORT 7. Em cada gráfico, são representadas as respostas ocorridas para as perguntas referentes aos *checklists* de cada uma. As perguntas enfocavam a compreensão (questões 35, 39 e 43), a necessidade (questões 36, 40 e 44) e a suficiência (questões 37, 41 e 45) das questões existentes nos *checklists* e, além destas três, mais uma referente à suficiência de artefatos abordados nos *checklists* (questões 38, 42 e 46) para realizar uma OORT.

Observando-se os resultados na Figura 25, verifica-se que a OORT 5 foi a de melhor desempenho, pois se observa no gráfico que em todas as perguntas ocorreram um indicativo igual ou acima de 40% da categoria de muita satisfação para todas as perguntas abordadas no formulário de relato, não ocorrendo categorias abaixo de média satisfação. A OORT 6 e a OORT 7 não obtiveram uma avaliação tão boa quanto a da OORT 5, pois, apesar de terem na maioria das perguntas, um índice bem presente da categoria de muita satisfação, ainda assim não se pode ignorar o fato de que em todas as perguntas ocorreram as categorias de pouca satisfação ou de não satisfação, com um mínimo de 20%.

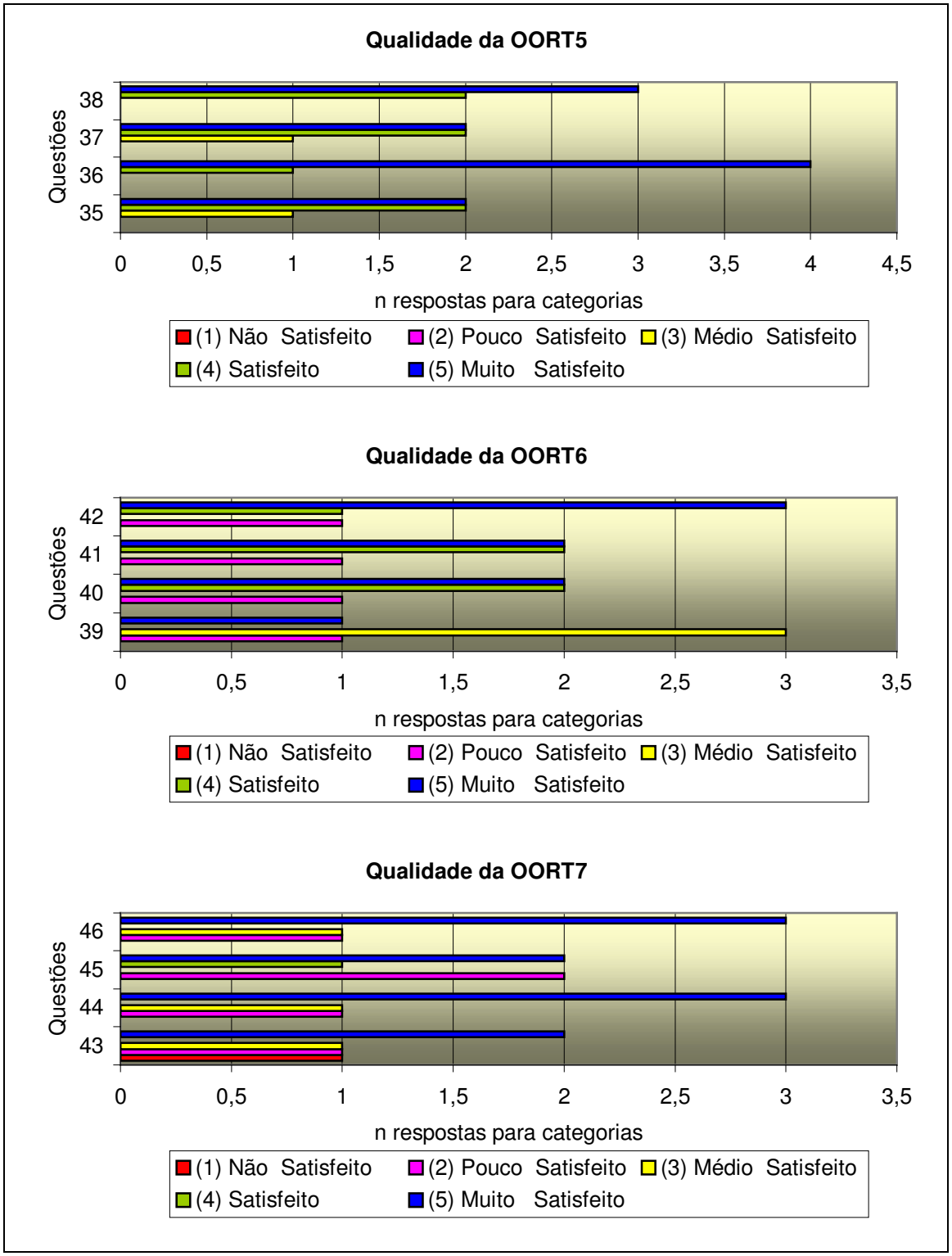


Figura 25 - Avaliação dos Checklists das OORTs da Leitura Vertical

A questão 47 avaliava o entendimento dos especialistas sobre a OORT, ou seja, um entendimento como um todo da técnica, relativo não somente ao material que compõe a técnica, como também ao procedimento envolvido na

sua aplicação. Obteve-se, nesta questão, um indicador de satisfação de 60% por parte dos especialistas (Figura 26), ressaltando-se que houve também indicadores de média e de pouca satisfação, no caso de dois especialistas.

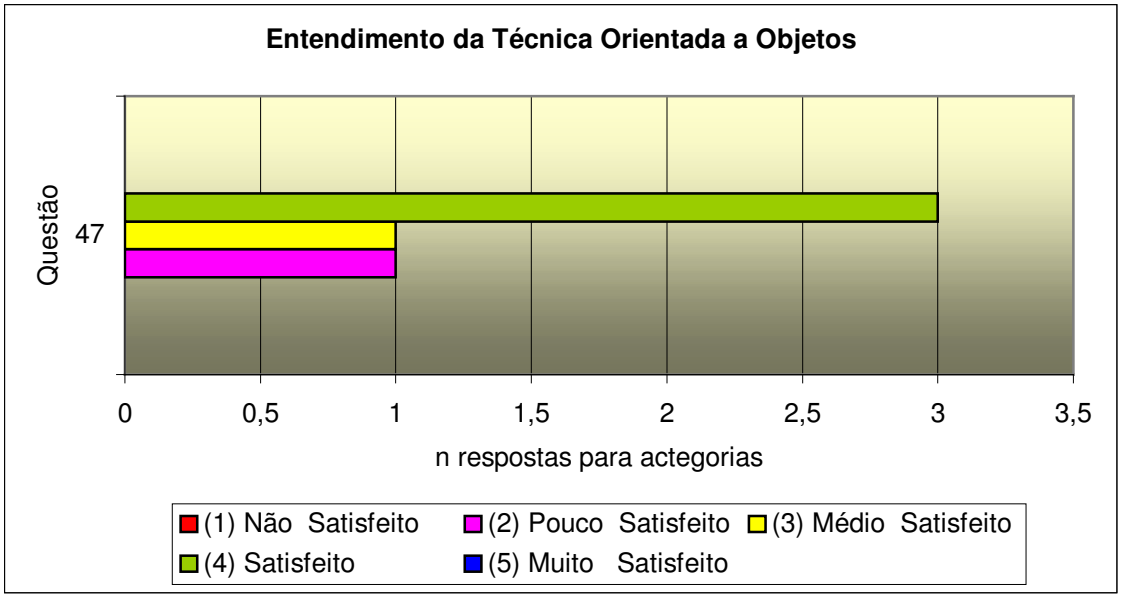


Figura 26 - Avaliação do Entendimento da Técnica Orientada a Objetos

O aumento de conhecimento após a inspeção utilizando a técnica OORT, também foi avaliado. As questões 48 e 49 tinham esta finalidade e respectivamente representavam as avaliações quanto à inspeção e à UML. Pôde-se perceber um apontamento (Figura 27) de média satisfação e de não satisfação a 80%, quanto à UML, e de média satisfação e satisfação a 80%, para a inspeção. Tais resultados sugerem um indicativo de baixa satisfação quanto à UML e um alto quanto à inspeção.

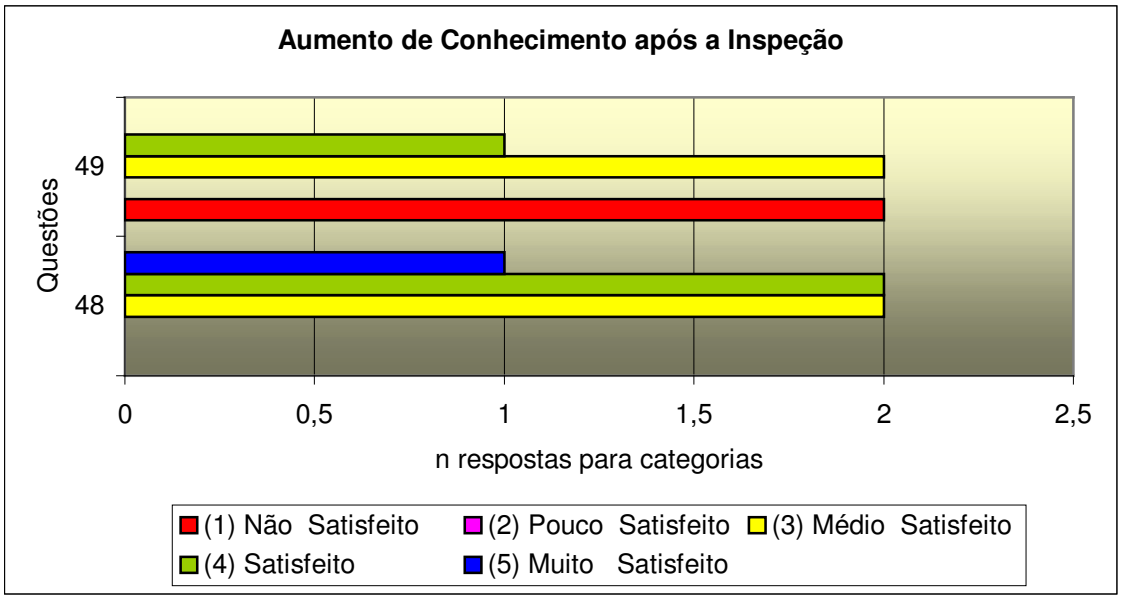


Figura 27 - Avaliação do Aumento de Conhecimento após a Inspeção

Como última questão, foi perguntado aos especialistas se estes consideravam, de alguma forma, a possibilidade de utilizar a OORT como parte de um projeto de desenvolvimento de um software. Na Figura 28, é possível observar que existe uma tendência de rejeição quanto a esta possibilidade, representando 40% na categoria de pouca satisfação. Mesmo assim, é importante observar também, que, apesar de divididos, 60% dos especialistas tiveram avaliações favoráveis quanto ao assunto, variando de média satisfação a muita satisfação.

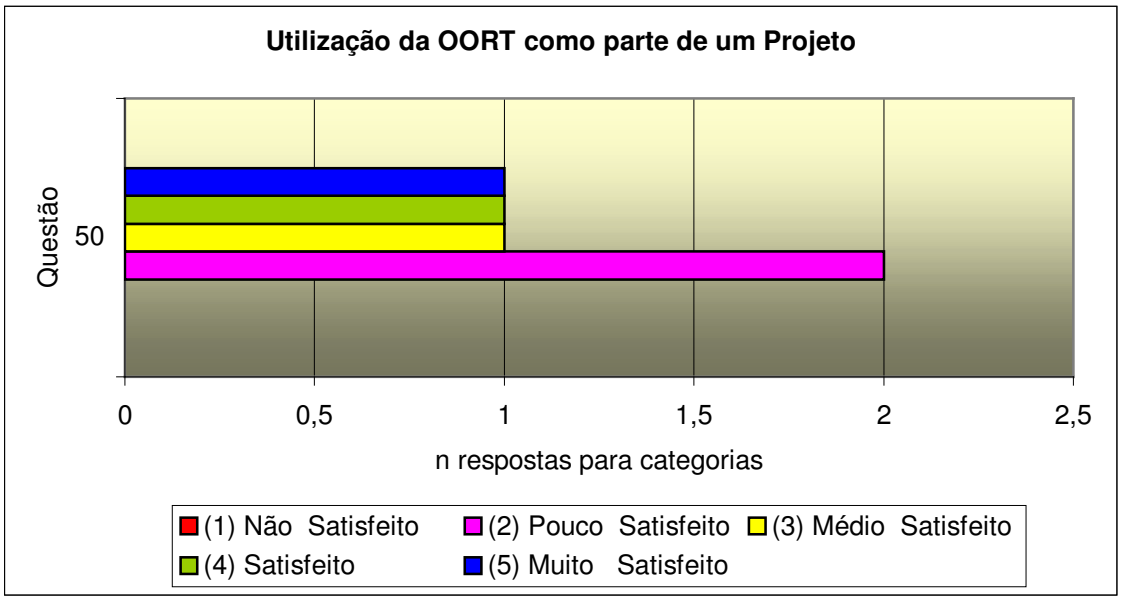


Figura 28 - Avaliação da Utilização da OORT como parte de um Projeto

3.6. CONSIDERAÇÕES FINAIS

A partir da análise dos resultados, ficaram evidenciados os seguintes pontos:

a) Quanto à eficiência das técnicas OORT

Pode-se concluir que, para a pesquisa realizada, incluindo os especialistas que dela participaram e o sistema inspecionado:

- De maneira geral, as técnicas OORT mostraram-se significativamente eficientes, uma vez que 73% dos defeitos contidos na especificação do sistema, preparada através dos diagramas UML, foram detectados.

- Considerando-se as sete técnicas, a técnica OORT7 mostrou-se menos eficiente, uma vez que os defeitos que deveriam ser detectados através dela não foram detectados pelos participantes.

- Considerando-se os cinco tipos de defeitos adotados, o índice de detecção de defeitos mais baixo ocorreu para os defeitos referentes à “fato incorreto” com 37,5% de defeitos detectados, seguida pela classificação de “omissão” com 50%.

- Considerando-se os dois tipos de leitura, a Leitura Horizontal mostrou-se significativamente mais eficiente do que a Leitura Vertical, na detecção dos defeitos relacionados a cada uma delas, sendo que a Leitura Horizontal detectou 91,30% dos defeitos e a Leitura Vertical detectou 54,17% dos defeitos.

b) Quanto à utilização das técnicas OORT

Pode-se concluir que, para a pesquisa realizada, incluindo os especialistas que dela participaram e o sistema inspecionado:

- Considerando-se a classificação de defeitos utilizada, a grande maioria (80%) dos participantes declarou estar satisfeita quanto ao entendimento propiciado por ela. No entanto, apenas 20% informaram estar satisfeitos com a completude da classificação adotada. Tais resultados podem sugerir a necessidade de se revisar e testar mais detalhadamente a classificação, buscando estendê-la de forma a possibilitar a cobertura de outros tipos de defeitos que porventura possam ocorrer.

- Considerando-se a descrição fornecida a cada tipo de defeito constante na classificação adotada, 80% dos participantes declararam estar “muito satisfeitos”, “satisfeitos” ou com “satisfação média”, tanto em relação ao quesito de entendimento quanto ao quesito de completude. Tal resultado pode ser um indicativo de que, de maneira geral, os defeitos estão definidos adequadamente.

- Considerando-se o tipo de discrepância, que obteve uma boa avaliação, averigua-se um problema de completude em relação à descrição fornecida para a Leitura Vertical, o que demonstra a necessidade de sua revisão, buscando cobrir as possibilidades que a classificação se dispõe.

- Considerando-se a severidade do defeito, 80% dos participantes declararam estar “muito satisfeitos” em relação ao entendimento. No entanto, quanto à completude, 40% dos participantes informaram estar “pouco satisfeitos”. Tais resultados podem sugerir a necessidade de uma complementação ou de uma avaliação da necessidade desta classificação junto à técnica OORT.

- Considerando-se os formulários para preenchimento de defeitos das leituras horizontal e vertical, ambos obtiveram uma boa avaliação, sendo considerados completos por 100% dos participantes. Ficando somente algumas observações negativas para a grande quantidade de informações a

serem preenchidas neles e também para os itens da tabela de identificação dos defeitos. Tal resultado pode sugerir uma revisão, buscando formulários claros e concisos.

- Considerando-se as OORTs que compõem a Leitura Horizontal, constatou-se que a OORT 3 (que analisa comparativamente Diagrama de Seqüência x Diagrama de Estado) foi a que recebeu um menor nível de satisfação, conforme declarado pelos especialistas (20% somando “pouca satisfação” e “não satisfação”). Tal resultado pode indicar a necessidade de se revisar a OORT3, visando melhorar sua qualidade.

- Considerando-se as OORTs que compõem a Leitura Vertical, identificou-se que a OORT6 e a OORT7 receberam um índice menor referente à satisfação dos participantes, em relação à OORT5. De maneira geral, a OORT7 foi indicada como a de menor índice de satisfação por parte dos especialistas. Tais resultados podem ser vistos como um indicativo para se revisar as duas OORTs, especialmente a OORT7, a fim de se promover melhorias nas mesmas.

- Considerando-se o conhecimento propiciado aos especialistas através da participação na pesquisa, 100% dos participantes declararam que seu conhecimento relativo à inspeção (incluindo o uso das técnicas OORT) foi incrementado. Tal resultado é bastante significativo e pode sugerir que a abordagem seguida na pesquisa seria útil para a definição de treinamento de profissionais para uso das técnicas OORT.

- Considerando-se a propensão demonstrada pelos especialistas em utilizar as técnicas OORT em um projeto de software, no ambiente empresarial, 40% dos participantes posicionaram-se desfavoravelmente a esta possibilidade. Tal resultado é significativo e aponta para a necessidade de se estabelecer estratégias de utilização das referidas técnicas, de maneira a tornar o processo de inspeção através das OORTs menos desgastante. É importante esclarecer que, na pesquisa, cada participante inspecionou toda a

especificação do sistema, utilizando as sete OORTs. Uma estratégia diferente, em que cada profissional ficasse responsável pela aplicação de apenas uma parte das OORTs, tenderia a ser menos cansativa para os inspetores.

CAPÍTULO 4

CONCLUSÃO

O trabalho realizado, cumpriu com os objetivos inicialmente propostos, através do estudo da literatura relacionada às técnicas OORT, sua preparação juntamente com a definição do processo de inspeção e da realização de um estudo empírico, onde dados e informações foram obtidos e analisados.

É importante salientar que o estudo empírico apresentado representou um passo inicial para um domínio mais aprofundado das técnicas OORT, não apenas visando identificar a eficiência dessas técnicas na detecção de defeitos em especificações elaboradas através de UML, mas também visando identificar aspectos que poderiam ser melhorados nas técnicas, como foi abordado na presente pesquisa.

Desta forma, o trabalho presta uma contribuição à melhoria da qualidade de software, através da discussão de técnicas de avaliação da qualidade de especificações de requisitos e da apresentação de um estudo empírico com resultados significativos. Tais resultados, permitiram também a contribuição no âmbito do refinamento do processo de inspeção de UML, a possibilidade da melhoria dos requisitos representados em UML e a possibilidade de melhoria do processo de desenvolvimento de software e do produto final.

Considerando-se as sugestões de melhorias, conclui-se que são necessárias as revisões dos seguintes itens relacionados à técnica OORT:

- Classificação de Defeitos.
- Tipo de Discrepância.
- Severidade do Defeito.
- Tabela de Identificação de Defeitos.

- Formulário para Preenchimento de Defeitos.
- OORT 3, OORT 6 e OORT 7.

Analisando o futuro das técnicas OORT, observa-se que elas têm grandes possibilidades de serem amplamente utilizadas tanto no ambiente acadêmico como no empresarial. Salienta-se no entanto, que para que este futuro seja uma realidade, as técnicas necessitam de uma reavaliação quanto à estratégia de aplicação, tornado-a menos cansativa. Uma das possíveis maneiras de obter este resultado seria a realização de um trabalho em relação ao desenvolvimento da automatização de tarefas das técnicas OORT, ou seja, a criação de uma ferramenta que auxiliasse e tornasse mais rápida a inspeção através das técnicas OORT.

Como trabalho futuro, destaca-se ainda, a importância de se realizar as melhorias sugeridas e também a realização de uma nova pesquisa, envolvendo maior número de especialistas, o que tornaria possível a análise dos resultados através de técnicas de inferência estatística, levando, assim, a resultados passíveis de serem estendidos a um escopo maior referente à utilização das técnicas OORT.

A aceitação da técnica OORT depende tanto destas melhorarias como de trabalhos futuros, para uma possível evolução da técnica.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, A. e DAROLT, R. **Pesquisa e Desenvolvimento em UML**. Universidade do Sul de Santa Catarina. Araranguá. 2001. Disponível em: <http://www.oodesign.com.br/forum/index.php?act=Attach&type=post&id=911>. Acesso em: 26 junho. 2003.

ARIF, T. e HEDGE, L. C. **Inspection of Object Oriented Construction: A study of Reading Techniques tailored for inspection of Desing Models expressed in UML**. Prediploma Thesis. Norwegian University of Science and Technology. Trondheim - Norway. Novembro 2001.

BARROS, P. **Linguagem de Modelagem Unificada**. São Paulo. [2000]. Disponível em: <http://www.uml.com.br/>. Acesso em: 26 junho. 2003.

BIFFL, S. e HALLING, M. **Software Product Improvement with Inspection**. Proceeedings of the 26th EUROMICRO Conference. Universitat Wien. Institute of Software Technology. Wien. 2000. p. 2262-2269.

BOOCH G.; RUMBAUGH, J. e JACOBSON, I. **UML Guia do Usuário**. 8ª ed. Rio de Janeiro. ed Campus. 2000.

BUNDE, G. A. e PEDERSEN, A., **Defect Reduction by Improving Inspection of UML Diagrams in the GPRS Project**. Master Thesis. Agder University College. Grimstad - Norway. 2002.

CARVER, J. **Impact of Background Experience on Software Inspections**. Ph.D. Thesis Proposal. University of Maryland. Maryland - EUA. [2000].

CIOLKOWSKI, M. **Evaluating the Effectiveness of Different Inspection Techniques on Informal Requirements Documents**. Master Thesis. University of Kaiserslautern. Kaiserslautern. 1999.

CONRADI, R. et al. **Object-Oriented Reading Techniques for Inspection of UML Models - An Industrial Experiment**. Proceedings of the European Conference on Object-Oriented Programming ECOOP'03. Darmstadt. Germany. July 2003. Springer LNCS 2743. p. 483-501.

DOOLAN, E. P. **Experience with Fagan's Inspection Method**. SOFTWARE – PRACTICE AND EXPERIENCE. Vol. 22(2). John Wiley & Sons. Rijswijk - Netherlands. Fevereiro 1992. p. 173-182.

EBENAU, R. G. e STRAUSS, S. H., **Software Inspection Process**. 1ª ed. New York - USA. ed McGraw-Hill. 1994.

ERIKSSON, H. E.; MAGNUS, P. **UML Toolkit**. 1ª ed. New York - USA. ed John Wiley & Sons. 1998.

FAGAN, M. E., **Desing and Code Inspections to Reduce Errors in Program Development**. IBM Systems Journal. Vol. 15. Number 3. [S.I.]. 1976. p. 182-211.

FAGAN, M. E., **Advances in Software Inspections**. IEEE Transactions on Software Engineering. Vol. 12. Number 7. [S.I.]. 1986. p .744-751.

GILB, T. **Optimizing Software Inspections**. Crosstalk: The Journal of Defense Software Engineering. [S.I.]. Março 1998. p. 16-19.

KIRNER, T.; ABIB, J. C. **Inspection of Software Requirements Specification Documents: A Pilot Study**. In: XV INTERNATIONAL CONFERENCE ON SYSTEM DOCUMENTATION – SIGDOC'97. Salt Lake City – USA. Proceedings of the XV ACM International Conferece on System Documentation – SIGDOC 97. SALT LAKE CITY, UT, USA : ACM Press. 1997. p. 161-171.

KYLVAG, K., **Inspect Inspections – A Method of How to Investigate the use of Software Inspections in Companies**. Bachelor Thesis. Bachelor os Science in Software Engineering. School of Engineering in Helsingborg. Lund Institute of Technology. Helsingborg. 2001.

LEVINE, D.M.; BERENSON, M.L. e STEPHAN, D. **Estatística: teoria e aplicações**. 1ª ed. Rio de Janeiro. ed LTC. 2000.

MATTAR, F.N. **Pesquisa de Marketing**. 4ª ed. São Paulo. ed Atlas. 1997.

PAGLIUSO, P. B. B.; TAMBASCIA, C. A. e VILLAS-BOAS, A. **Melhoria da Inspeção de Requisitos Segundo a Técnica de Leitura Baseada em Perspectiva**. XI SEMINCO – Seminário de Computação. Universidade Regional de Blumenau. Blumenau. 2002.

PEREIRA, J.C.R. **Analise de Dados Qualitativos**. 3ª ed. São Paulo. ed Edusp. 1999. cap. 4. p. 77-100.

PORTER, A.H.; SIY, H. e VOTTA, L. **A Review of Software Inspections**. Advances in Computer. Vol. 42. Naperville - Illinois. 1996. p. 40-76.

ROCHA, A. R. C.; MALDONADO, J. C. e WEBER, K. C. **Qualidade de Software – Teoria e Prática**. 1ª ed. São Paulo. ed Makron Books . 2001. p. 1-4.

SINGER, J. e VINSON, N. **Ethical Issues in Empirical Studies of Software Engineering**. Montreal. 2002. Disponível em: <http://iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-44912.pdf>. Acesso em: 16 janeiro. 2004.

SILVA, T. A., **Um Estudo sobre a UML – Unified modeling Langua*ge***. Faculdade de Ciências Exatas e da Natureza. Universidade Metodista de Piracicaba. Piracicaba. 1999.

SOUZA, R. C. G. **Características de Testabilidade nos Diagramas UML (Unified Modeling Language): Apoio aos Testes de Sistemas de Software Orientados a Objetos**. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo. São Paulo. 2003. Cap. 6. p. 110-141.

TOMAYKO, J., **Inspections**. In: Marciniak, J. Encyclopedia of Software Engeneering. 1ª ed. [S.I.]. ed John Wiley & Sons. 1994. p. 572 – 573.

TRAVASSOS, G. H. **Revisões de Software**. In: ROCHA, A. R. C.; MALDONADO, J. C. e WEBER, K. C. Qualidade de Software – Teoria e Prática. 1ª ed. São Paulo. ed Makron Books. 2001. p. 84 – 94.

TRAVASSOS, G. H.; SHULL, F. e CARVER, J., **Evolving a Process for Inspecting OO Desings**. In: Workshop Qualidade de Software – SBES'99. Florianópolis. 1999.

TRAVASSOS, G. H. et al. **Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality**. Proceedings of the OOPSLA'99. Denver - Colorado. (in ACM SIGPLAN Notices,34(10), Oct. 1999). 1999. p. 47-56.

TRAVASSOS, G. H. et al. **Reading Techniques for OO Design Inspections**. In: Proceedings of the 34th Software Engineering Workshop – NASA/SEL. Greenbelt. 2002.

VIEIRA, S. e HOFFMANN, R. **Elementos de Estatística**. 2ª ed. São Paulo. ed Atlas. 1991.

VOTTA JR, L. **Does Every Inspection Need a Meeting?**. ACM SIGSOFT Software Engineering Notes. Vol. 18. Number 5. Naperville - Ilinois. Dezembro 1993. p. 107-114.

ANEXO 1

TÉCNICAS DE LEITURA ORIENTADA A OBJETOS (OORT)

ANEXO 1.1. OORT 1 – DIAGRAMA DE SEQÜÊNCIA X DIAGRAMA DE CLASSES

Objetivo: Verificar se o Diagrama de Classe do sistema descreve as classes e seus relacionamentos de tal forma que o comportamento especificado no Diagrama de Seqüência seja corretamente capturado.

Entradas:
1. Um Diagrama de Classes.
2. Diagrama de Seqüência.

Saídas:
1. Versões anotadas de ambos os diagramas.
2. Relatórios de discrepâncias.

Instruções: Execute os passos R 1.1 e R 1.2.

PASSO R 1.1: A PARTIR DO DIAGRAMA DE SEQÜÊNCIA – IDENTIFIQUE OS OBJETOS, SERVIÇOS E CONDIÇÕES DO SISTEMA.

Entradas:

1. Diagrama de Seqüência.

Saídas:

1. Objetos, classes e atores do sistema (sublinhados com **azul** no Diagrama de Seqüência).
2. Serviços do sistema (sublinhados com **verde** no Diagrama de Seqüência).
3. Restrições/Condições nas mensagens/serviços (circulados com **amarelo** no Diagrama de Seqüência).

Isto é, um Diagrama de Seqüência marcado será produzido, e será usado em R 1.2.

Instruções: iguais as saídas acima

Q 11.a: Sublinhe os objetos, classes e atores do sistema com **azul** no Diagrama de Seqüência.

Q 11.b: Sublinhe os serviços do sistema em **verde** no Diagrama de Seqüência.

Q 11.c: Circule restrições/condições nas mensagens/serviços com **amarelo** no Diagrama de Seqüência.

PASSO R 1.2: CHEQUE OS DIAGRAMAS DE CLASSES RELACIONADOS, PARA AVERIGUAR SE TODOS OS OBJETOS DO SISTEMA SÃO CONSIDERADOS.

Entradas:

1. Diagramas de Seqüência marcados do passo R 1.1.
2. Diagramas de Classes.

Saídas:

1. Relatórios de discrepância.

Instruções:

Q 12.a: Todo objeto/classe/ator do Diagrama de Seqüência pode ser encontrado no Diagrama de Classe?

Possível inconsistência.

Q 12.b: Todo serviço/mensagem do Diagrama de Seqüência pode ser encontrado no Diagrama de Classes, e com os parâmetros apropriados?

Possível inconsistência.

Q 12.c: Todos os serviços (baixo nível) são considerados pelas mensagens no Diagrama de Seqüência?

Possível omissão.

Q 12.d: Existe uma associação ou outro tipo de relacionamento entre duas classes no caso de troca de mensagens?

Possível omissão.

Q 12.e: Existe uma má combinação no comportamento do argumento ou em como os restrições/condições são formulados entre os dois documentos?

Possível inconsistência.

Q 12.f: As restrições/condições do Diagrama de Seqüência no passo R 1.1 podem ser cumpridas?Por exemplo:

- Número de objetos que podem receber uma mensagem (cheque a cardinalidade no Diagrama de Classes)?
- Extensão dos valores de dados?
- Dependências entre dados e objetos?
- Restrição/condição de tempo?

Possível inconsistência.

Q 12.g: Comentários globais do projeto, baseado em experiência própria, domínio de conhecimento, e compreensão, por exemplo:

- As mensagens e seus parâmetros fazem sentido para este objeto?
- As condições/restrições declaradas são apropriadas?
- Todos os atributos necessários foram definidos?
- Os atributos/funções definidos na classe fazem sentido?
- As classes/atributos/funções possuem nomes significativos?
- As relações de classe são razoáveis e do tipo correto?(exemplo: associação x relacionamento de composições).

Possível fato incorreto.

ANEXO 1.2. OORT 2 – DIAGRAMA DE ESTADOS X DESCRIÇÃO DE CLASSES

Objetivo: Verificar se as classes estão definidas, de forma que elas possam capturar a funcionalidade especificada pelo Diagrama de Estados.

Entradas:
1. Descrição de Classes.
2. Diagrama de Estados para os objetos do sistema.

Saídas:
1. Relatórios de discrepâncias.

Instruções: Execute os passos R 2.1 a R 2.3, para cada Diagrama de Estados.

PASSO R 2.1: LEIA O DIAGRAMA DE ESTADOS PARA ENTENDER POSSÍVEIS ESTADOS E SUAS TRANSIÇÕES.

Entradas:

1. Diagrama de Estados.
2. Descrição de Classes.

Saídas:

1. Estados de Objetos (marcados com **azul** no Diagrama de Estados).
2. Transição de ações/condições (marcados com **verde** no Diagrama de Estados).

Isto é, um Diagrama de Estados com marcações é produzido e utilizado em R 2.2 e R 2.3.

3. Relatórios de discrepâncias.

Instruções:

Q 21.a: Identifique a verdadeira classe do Diagrama de Estados. Inexistente? Possível omissão.

Q 21.b: Sublinhe o nome de cada estado de objeto (com caneta **azul**).

Q 21.c: Sublinhe as transições de ações/condições/restrições (com caneta **verde**).

Q 21.d Você pode entender o comportamento dos objetos a partir de Q 21.b-c? *Possível ambigüidade.*

PASSO R 2.2: IDENTIFIQUE AS CLASSES ASSOCIADAS, SEUS ATRIBUTOS E COMPORTAMENTOS.

Entradas: parcialmente do Diagrama de Estados

1. Descrição de Classes.
2. Estados de objetos (marcados com **azul** no Diagrama de Estados – do passo R 2.1).
3. Transições de ações/condições/restrições (marcados com **verde** no Diagrama de Estados – do passo R 2.1).

Saídas:

1. Relatórios de discrepância.

Instruções:

Q 22.a: Na Descrição de Classes, identifique a classe sendo modelado pelo Diagrama de Estados. Inexistente?

Possível omissão.

Q 22.b: Descubra como um estado em **azul** está representado, isto é a classe capturou cada estado modelado de modo único?

- Por exemplo, através de um atributo explícito.
- Por exemplo, através de um atributo implícito (meramente através de controle de fluxo).
- Por exemplo, através de combinação de atributos.
- Por exemplo, através do subtipo do verdadeiro objeto (consulte a hierarquia de classes).

Possível inconsistência ou ambigüidade.

Q 22.c: Todas as transições de ações/condições/restrições em **verde** estão dentro do comportamento das classes?

Possível inconsistência.

Q 22.d: Todas as transições de condições/restrições estão usando dados de objetos que são definidos como atributos de classes com nomes correspondentes?

Possível inconsistência.

PASSO R 2.3: COMPARE A DESCRIÇÃO DE CLASSES COM O DIAGRAMA DE ESTADOS.

Q 23.a: A partir do seu domínio de conhecimento, todos os estados relevantes estão definidos no Diagrama de Classes?
Possível fato incorreto.
Q 23.b: Para cada estado marcado, avalie se este é apropriado e essencial.
Possível fato incorreto ou informação estranha.
Q 23.c: Para cada transição de ações/condições/restrições marcadas: ausência de informação.
Possível inconsistência.

ANEXO 1.3. OORT 3 – DIAGRAMA DE SEQÜÊNCIA X DIAGRAMA DE ESTADOS

Objetivo: Verificar se toda transição de estado para um objeto pode ser atingida pelas mensagens transmitidas e recebidas por aquele objeto.

Entradas:
1. Diagrama de seqüência.
2. Diagrama de Estados para diversos objetos.

Saídas:
1. Relatórios de discrepâncias.

Instruções: Execute os passos R 3.1 - R 3.3, para cada Diagrama de Estados.

PASSO R 3.1: LEIA O DIAGRAMA DE ESTADOS PARA ENTENDER POSSÍVEIS ESTADOS DE OBJETOS, SUAS TRANSIÇÕES E AÇÕES CORRESPONDENTES.

Entradas:

1. Diagrama de Estados.

Saídas:

1. Diagrama de Estados marcado, com transição de ações identificadas em **verde.**
2. Relatórios de discrepâncias.

Instruções:

Q 31.a: Determine qual classe está sendo modelada. Inexistente?

Possível omissão.

Q 31.b: Rastreie todas as transições do começo do estado para o fim do estado, e marque as ações correspondentes com um nome único (**A1, A2, etc**) com **verde.**

Q 31.c: De modo geral, estas transições de ações e estados fazem sentido e são compreensíveis para tal objeto?

Possível ambigüidade.

PASSO R 3.2: LEIA O DIAGRAMA DE SEQÜÊNCIA PARA ENTENDER COMO AS TRANSIÇÕES DE AÇÕES SÃO ATINGIDAS PELAS MENSAGENS ENVIADAS PARA/DE OBJETOS RELEVANTES.

Entradas:

1. Diagrama de Estados marcado (com transição de ações identificadas em **verde** – do passo R 3.1).
2. Diagrama de Seqüência.

Saídas:

1. Diagrama de Seqüência marcado, com as mensagens de objetos combinando identificadas em **verde.**
2. Relatórios de discrepância.

Instruções:

Q 32.a: Pegue o Diagrama de Seqüência em relevância com o Diagrama de Estados. Existe algum problema em identificá-los?

Possível omissão ou informação estranha.

Para cada Diagrama de Seqüência relevante faça Q 32.b-e:

Q 32.b: Leia o Diagrama de Seqüência para identificar os serviços de sistemas e mensagens associados.

Q 32.c: Identifique os estados dos objetos no Diagrama de Estados, sendo semanticamente relacionado com o serviço de sistema.

Q 32.d: Mapeie as setas de mensagens (uma ou mais) no Diagrama de Seqüência para expor transições no Diagrama de Estados. Existem mensagens "suficientes" para realizar uma dada transação?

Possível omissão.

Marque mensagens do Diagrama de Seqüência relacionadas e transições do Diagrama de Estados com um nome (A1, A2, etc) em **verde** no Diagrama de Seqüência**.**

Q 32.e: Procure por restrições e condições nas mensagens acima do Diagrama de Seqüência. Cheque se a mesma informação de restrição/condição se sustenta em ambos os diagramas.

Possível inconsistência.

Tal informação do Diagrama de Estados deve ser correspondentemente expressada no Diagrama de Seqüência por: (1) Informação de condição/restrição (exemplo: t=verdadeiro ou t>0). (2) Informação de transição (o que ocorre quando t>0?). (3) Nada (não relevante para o Diagrama de Estados).

PASSO R 3.3: REVEJA OS DIAGRAMAS MARCADOS PARA TER CERTEZA QUE TODAS AS TRANSIÇÕES DE AÇÕES FORAM CONSIDERADAS.

Entradas:

1. Transações de ações do Diagrama de Estados (marcadas com **verde** – do passo R 3.1)
2. Mensagens de objetos do Diagrama de Seqüência (marcadas com **verde**–do passo R 3.2)

Saídas:

1. Relatórios de discrepâncias.

Instruções:

Q 33.a: Procure por transições de ações não identificadas no Diagrama de Estados, isto é, aquelas não implementadas por mensagens disponíveis no Diagrama de Seqüência.

Possível inconsistência.

Q 33.b: A ordem de eventos é a mesma no Diagrama de Estados e no Diagrama de Seqüência, isto é cheque se as mensagens/transições identificadas no Diagrama de Seqüência aparecem em ordem lógica?

Possível inconsistência.

Exemplo, a ação Ax em uma transição posterior do Diagrama de Seqüência na verdade ocorre depois de uma ação Ay em uma transição anterior.

ANEXO 1.4. OORT 4 – DIAGRAMA DE CLASSES X DESCRIÇÃO DE CLASSES

Objetivo: Verificar se a descrição detalhada das classes possui todas as informações necessárias, e se a Descrição das Classes tem senso semântico.

Entradas:	**Saídas:**
1. Um Diagrama de Classes.	1. Relatórios de discrepância.
2. Descrição de Classes.	

Instruções:
Execute os passos R 4.1 e R 4.2, para cada classe no Diagrama de Classes.

PASSO R 4.1: LEIA O DIAGRAMA DE CLASSES PARA ENTENDER AS PROPRIEDADES NECESSÁRIAS.

Entradas:

1. Uma determinada classe do Diagrama de Classes.
2. Descrição de Classes.

Saídas:

1. Relatórios de discrepância.

Instruções:

Q 41.a: Existe uma Descrição de Classes para esta classe?

Possível omissão.

Marque com um **asterisco azul** na Descrição de Classes, quando for possível encontrar, veja R 4.2.

Q 41.b: O nome e as descrições textuais da classe são significantes na Descrição de Classes?

Possível ambigüidade.

Q 41.c: Os atributos e tipos são consistentes entre o Diagrama de Classes e as Descrição de Classes?

Possível inconsistência.

Q 41.d: Esta classe pode encapsular significantemente todos estes atributos e com determinados tipos?

Possível ambigüidade ou fato incorreto.

Q 41.e: Em comportamento e restrições/condições:

- Cheque a consistência entre comportamentos e restrições/condições entre o Diagrama de Classes e as Descrição de Classes.
- Os comportamentos na Descrição de Classes estão descritos no mesmo nível de detalhe/pseudocódigo?
  Possível inconsistência.
- De modo geral, esta classe deveria realmente conter todos estes comportamentos e restrições/condições?
  Possível fato incorreto.
- Os comportamentos e restrições/condições da Descrição de Classes utilizam comportamentos ou atributos de algum outro lugar, e estão eles definidos?
  Possível omissão ou ambigüidade.
- Os comportamentos e restrições/condições da Descrição de Classes baseiam-se “excessivamente” em atributos de classes remotas?Isto é, muita junção?
  Possível outros.

Q 41.f: Para os casos que utilizam herança no Diagrama de Classes:

- A herança esta inclusa na Descrição de Classes?
  Possível omissão.
- De modo geral, é “significativo” que uma dada classe seja supertipo/subtipo de uma dada subclasses/superclasse?
  Possível outros.

Q 41.g: Cheque se todos os relacionamentos são corretamente descritos:

- Eles têm as cardinalidades corretas, e elas são descritas na Descrição de classe?
  Possível inconsistência.
- Os papéis dos objetos no Diagrama de Classes também são definidos na Descrição de Classes?
  Possível inconsistência.
- A notação gráfica correta está sendo usada no Diagrama de Classes?
  Possível inconsistência.

- De modo geral, as relações declaradas “fazem sentido”, assim como composição vs. agregação vs. associação vs. Herança, etc?
  Possível informação estranha.
- Um atributo é utilizado para representar um relacionamento, e este tem o tipo (uma referência ou referências) correto?
  Possível inconsistência.

PASSO R 4.2: REVEJA A CLASSE A PROCURA DE INFORMAÇÕES ESTRANHAS.

Entradas:

1. Descrição de Classes.

Saídas:

1. Relatórios de discrepância.

Instruções:

Q 42.a: Há alguma classe não marcada com um asterisco (isto é supérflua) na Descrição de Classes?*Possível informação estranha.*

ANEXO 1.5. OORT 5 – DESCRIÇÃO DE CLASSES X DESCRIÇÃO DE REQUISITOS

Objetivo: Verificar se os conceitos e serviços que são descritos pelos requisitos funcionais são capturados pela descrição da classe.

Entradas:
1. Descrição de Requisitos.

2. Descrição de Classes.

Saídas:
1. Relatórios de discrepâncias.

Instruções: Faça os passos R 5.1 - R 5.3.

PASSO R 5.1: LEIA OS REQUISITOS PARA ENTENDER AS FUNCIONALIDADES DESCRITAS.

Entradas:

1. Descrição de Requisitos.

Saídas:

1. Classes, objetos, atributos candidatos (marcados com **azul** na Descrição de Requisitos).
2. Serviços candidatos (marcados com **verde** na Descrição de Requisitos).
3. Restrições e condições dos serviços (marcados com **amarelo** na Descrição de Requisitos).

Isto é, Descrição de Requisitos marcada é produzida, e usada nos passos R 5.2 e R 5.3 a seguir.

Instruções:

Q 51.a: Encontre os nomes, candidatos a classes/objetos/atributos. Sublinhe com **azul.**

Q 51.b: Encontre os verbos ou descrições de ações, candidatos a serviços ou comportamentos. Sublinhe com **verde.**

Q 51.c: Procure por restrições e condições em nomes/verbos acima, por exemplo, para relacionamentos, limitando quantidades, ou requisitos não-funcionais. Sublinhe com **amarelo.**

PASSO R 5.2: COMPARE A DESCRIÇÃO DE CLASSES COM OS REQUISITOS.

Entradas:

1. Descrição de Classes.
2. Descrição de Requisitos – do passo R 5.1.

Saídas:

1. Conceitos correspondentes marcados na Descrição de Requisitos e na Descrição de Classes.
2. Relatórios de discrepância.

Instruções:

Q 52.a: Para cada ação/verbo sublinhado em **verde** na Descrição de Requisitos:

- Ache comportamentos associados na Descrição de Classes.
- As classes/objetos recebem a informação correta para completar seus comportamentos requisitados, e resultados apropriados são produzidos?
  Possível fato incorreto.

Q 52.b: Para cada nome/conceito sublinhado em **azul** na Descrição de Requisitos, tente achar a classe associada na Descrição de Classes, e marque ambos com um **asterisco azul**.

- A Descrição de Classes contêm informações claras e suficientes para este conceito, e o nome da classe se assemelha ao nome que você assinalou?
  Possível ambigüidade.
- A classe encapsula atributos relacionados (marcados em **azul**), e a classe encapsula comportamentos relacionados (marcados em **verde**), e todas as restrições e condições foram identificadas para esta classe descrita na Descrição de Classes?
  Possível omissão.

Q 52.c: Para cada nome/conceito sublinhado em **azul** remanescente na Descrição de Requisitos, tente encontrar um atributo correspondente na Descrição de Classes, e marque ambos com um **asterisco azul.**

- De modo geral, a Descrição de Classes estão usando tipos apropriados para representar a informação da Descrição de Requisitos, e as restrições e condições destes atributos sublinhados em **amarelo** estão contidos na Descrição de Classes?
  Possível fato incorreto.

Passo R 5.3: Reveja a Descrição de classes e documentos de requisitos para assegurar que todos os conceitos correspondem mutuamente.

Entradas:

1. Descrição de Requisitos marcados – do passo R 5.1.
2. Descrição de Classes marcadas – do passo R 5.2.

Saídas:

1. Relatórios de discrepâncias.

Instruções:

Q 53.a: Existe ainda algum nome sublinhado com **azul** ou alguma atividade sublinhada com **verde** na Descrição de Requisitos que ainda não foram consideradas, isto é, representadas na Descrição de Classes?
Possível omissão.
Nota: alguns conceitos da Descrição de Requisitos podem ter sido utilizados somente para explicação.

ANEXO 1.6. OORT 6 – DIAGRAMA DE SEQÜÊNCIA X DIAGRAMA DE CASOS DE USO

Objetivo: Verificar se o Diagrama de Seqüência descreve uma combinação apropriada de objetos e mensagens que capturam a funcionalidade da especificação dos Casos de Uso.

Entradas:
1. Um Diagrama de Casos de Uso para uma parte do sistema, com seus serviços.
2. Um ou mais Diagramas de Seqüência para objetos e serviços relevantes do sistema.

Saídas:
1. Relatórios de discrepância.

Instruções:
Execute os passos R 6.1 a R 6.3 (somente o passo R 6.3 encontra defeitos).

PASSO R 6.1: IDENTIFIQUE AS PRINCIPAIS FUNCIONALIDADES DOS CASOS DE USO E SEUS CONCEITOS DE SISTEMA IMPORTANTES.

Entradas:

1. Diagrama de Casos de uso.

Saídas:

1. Conceitos de sistema (marcados com **azul** no Diagrama de Casos de Uso).
2. Serviços de sistema fornecidos (marcados com **verde** no Diagrama de Casos de Uso).
3. Dado necessário para atingir tais serviços (marcados com **amarelo** no Diagrama de Casos de Uso).

Instruções: (similar ao passo R 5.1 para a Descrição de Requisitos, mas aqui para o Diagrama de Casos de Uso).

Q 61.a: Encontre os nomes/conceitos únicos no Diagrama de Casos de Uso. Sublinhe e **numere** consecutivamente com **azul** (usado em Q 61.d).

Q 61.b: Para cada nome, encontre verbos/ações “de ou para” aquele nome.

Sublinhe com **verde** e **numere** em ordem de performance.

Q 61.c: Marque os restrições/condições **duplamente** sublinhados com **verde** (parte do serviço de marcar).

Q 61.d: Ache também a informação ou dado a ser enviado/recebido de forma a realizar uma determinada ação. Assinale o dado com **amarelo** como "**D x,y**", onde **x,y** são os nomes envolvidos.

PASSO R 6.2: IDENTIFIQUE E INSPECIONE O DIAGRAMA DE SEQÜÊNCIA RELACIONADO, PARA IDENTIFICAR SE A FUNCIONALIDADE CORRESPONDENTE ESTÁ DESCRITA CORRETAMENTE E SE OS COMPORTAMENTOS E DADOS ESTÃO REPRESENTADOS NA ORDEM CORRETA.

Entradas:

1. Diagrama de Casos de Uso, marcado com os conceitos, serviços, e dados do passo R 6.1.
2. Diagrama de Seqüência

Saídas:

1. Conceitos de sistema (marcados com **azul** no Diagrama de Seqüência).
2. Serviços de sistema (marcados com **verde** no Diagrama de Seqüência).
3. Intercâmbio de dados entre objetos (marcados com **amarelo** no Diagrama de Seqüência).

Instruções:

Q 62.a: Para cada Diagrama de Seqüência, sublinhe com **azul** os objetos de sistema, e com a mesma denominação **numérica** (de Q 61.a) como no Diagrama de Casos de Uso.

Q 62.b: Identifique os serviços descritos no Diagrama de Seqüência.

Isto é, observe as setas horizontais de mensagens entre os objetos, e possivelmente diversas setas em um serviço (cluster). Sublinhe os serviços identificados com **verde**, e **numere-os** em ordem ocorrida (do topo para baixo) no Diagrama de Seqüência.

Q 62.c: Identifique o intercâmbio de informações/dados entre duas classes de sistema (x, y).
Assinale o dado com amarelo como "**D x,y**", como em R 6.1.

PASSO R 6.3: COMPARE O DIAGRAMA DE CASOS DE USO MARCADO COM O DIAGRAMA DE SEQÜÊNCIA PARA DETERMINAR SE ELES REPRESENTAM O MESMO DOMÍNIO DE CONCEITOS.

Entradas:

1. Diagrama de Casos de Uso, com conceitos, serviços, e dados marcados do passo R 6.1.
2. Diagrama de Seqüência, com marcações similares do passo R 6.2.

Saídas:

1. Relatórios de discrepância.

Instruções:

Q 63.a: Para cada nome marcado com **azul** no Diagrama de Casos de Uso, procure por um similar no Diagrama de Seqüência.
Marque com um **asterisco azul** quando encontrado.
Para nomes não marcados com um asterisco no Diagrama de Casos de Uso, cheque também se possivelmente eles são atributos em alguma classe.
Os nomes remanescentes, não marcados com um asterisco do Diagrama de Casos de Uso podem representar defeitos, conforme eles não aparecem no projeto (Diagrama de Seqüência).
Possível omissão.
Q 63.b: Similarmente, para cada nome não marcado no Diagrama de Seqüência, ele pode pertencer a algum projeto interno ou pior: um conceito não utilizado.
Possível informação estranha.
Q 63.c: Para cada serviço marcado com **verde** no Diagrama de Seqüência, procure pelo correspondente no Diagrama de Casos de Uso.

- As classes/objetos do Diagrama de Seqüência estão trocando mensagens na mesma ordem que no Diagrama de Casos de Uso? Caso contrário isto pode ser um defeito.
- Os parâmetros das mensagens no Diagrama de Seqüência estão corretamente descritos no Diagrama de Casos de Uso, por exemplo, dado correto entre **D x,y,** etc?
- É possível "entender" a funcionalidade esperada, por exemplo, de um dado sendo enviado/recebido, simplesmente através da leitura do Diagrama de Seqüência?

Relatar qualquer problema nestes tópicos.

Possível inconsistência ou ambigüidade.

Q 63.d: As restrições/condições marcadas **duplamente** em **verde** do Diagrama de Casos de Uso são consideradas pelo Diagrama de Seqüência?

Possível fato incorreto.

ANEXO 1.7. OORT 7 – DIAGRAMA DE ESTADOS X (DESCRIÇÃO DE REQUISITOS/DIAGRAMA DE CASOS DE USO)

Objetivo: Verificar se o Diagrama de Estado descreve estados apropriados dos objetos e eventos que disparam mudanças de estados como descrito pela especificação dos Casos de Uso.

Entradas:
1. Diagrama de Estados.
2. Descrição de Requisitos.
3. Diagrama de Casos de Uso.

Saídas:
1. Relatórios de discrepâncias.

Instruções: Para cada Diagrama de Estados, faça os passos R 7.1 - R 7.4.

PASSO R 7.1: LEIA O DIAGRAMA DE ESTADOS PARA ENTENDER BASICAMENTE QUAL OBJETO ELE ESTÁ MODELANDO (NADA MAIS AQUI).

PASSO R 7.2: LEIA OS REQUISITOS FUNCIONAIS PARA DETERMINAR OS POSSÍVEIS ESTADOS DOS OBJETOS, QUAIS ESTADOS SÃO ADJACENTES UNS COM OS OUTROS, E QUAIS EVENTOS/AÇÕES CAUSAM AS MUDANÇAS DE ESTADOS.

Entrada:

1. Descrição de Requisitos.

Saídas:

1. Objetos de estados (marcados com **azul** na Descrição de Requisitos).
2. Matriz de adjacência (MA), recordando se existe uma transição de estado de um estado para outro.

Instruções: (somente leitura)

Q 72.a: Reserve o Diagrama de Estados e apague qualquer asterisco previamente marcado na Descrição de Requisitos.

Leia a Descrição de Requisitos e marque ligeiramente – com um **asterisco** – todos os lugares onde os verdadeiros conceitos do Diagrama de Estados são utilizados.

Q 72.b: Localize todos os lugares correspondentes na Descrição de Requisitos para todos estados deste objeto, marque estes lugares com **azul** e **numere-os** de 1 a N.

Q 72.c: Identifique qual dos estados numerados como sendo o estado inicial ("I"), e similarmente o estado final ("**F**").

Q 72.d: Faça uma **matriz** de adjacência (MA) de **N por N** em uma folha de papel separada. Tente identificar possíveis transições de estado ij, isto é se o estado i pode levar ao estado j. Coloque uma marca de checado ("**V**") nestas entradas **MA-ij**.

PASSO R 7.3: LEIA OS CASOS DE USO E DETERMINE QUAIS EVENTOS PROVOCAM MUDANÇA DE ESTADOS.

Entradas:

1. Diagrama de Casos de Uso.
2. Matriz de adjacência preliminar (MA) – do passo R 7.2.

Saídas:

1. Matriz de adjacência completa (MA).

Instruções:

Q 73.a: Leia os Diagramas de Casos de Uso e encontre aqueles em que os objetos participam.

Q 73.b: Para cada entrada **MA-ij** marcada (isto é tendo uma transição), documente precisamente o evento e/ou restrição/condição associada em algum lugar na MA.

Q 73.c: Para as entradas em branco, veja se ainda pode haver eventos que podem causar a transição. Caso contrário escreva um "**X**" na entrada.

PASSO R 7.4: LEIA O DIAGRAMA DE ESTADOS PARA DETERMINAR SE OS ESTADOS DESCRITOS ESTÃO CONSISTENTES COM OS REQUISITOS, E SE AS TRANSIÇÕES SÃO CONSISTENTES COM OS REQUISITOS E CASOS DE USO.

Entradas:

1. Descrição de Requisitos marcados – do passo R 7.2.
2. Diagrama de Estados.
3. Matriz de adjacência completa (MA) – do passo R 7.3.

Saídas:

1. Relatórios de discrepâncias.

Instruções: Repita os seguintes passos para cada Diagrama de Estados:

Q 74.a: Para cada estado numerado na Descrição de Requisitos, encontre o estado correspondente no Diagrama de Estados e marque-o com **azul** e **número** correspondente.

Nota: nomes de estados podem ser diferentes na Descrição de Requisitos e nos Diagramas de Estados, e nomes sobrepondo podem não representar estados idênticos.

- Todos os estados nas Descrições de Requisitos foram encontrados no Diagrama de Estados?

  Possível omissão.

  Ou talvez alguns estados da Descrição de Requisitos foram combinados em único estado do Diagrama de Estados, mas esta não foi uma combinação sensível?

  Possível fato incorreto.

- Inversamente, existem estados extras no Diagrama de Estados?

  Possível informação estranha.

  Ou talvez um estado da Descrição de Requisitos foi dividido em mais de um estado no Diagrama de Estados, mas novamente esta não foi uma combinação sensível?

  Possível fato incorreto.

Q 74.b: Cheque transições de eventos e ações na matriz MA:

- Todos os eventos na MA aparecem no Diagrama de Estados?
  Possível omissão.
- Todos os eventos no Diagrama de Estados aparecem na MA?
  Possível informação estranha.

Q 74.c: Cheque transições de restrições/condições e ações na matriz MA:

- Todas as restrições/condições na MA aparecem no Diagrama de Estados?
  Possível omissão.
- Todas as restrições/condições no Diagrama de Estados aparecem na MA?
  Possível informação estranha.

ANEXO 2

FORMULÁRIOS DE ANOTAÇÕES DE DEFEITOS

ANEXO 2.1. FORMULÁRIO DE ANOTAÇÃO DE DEFEITOS PARA LEITURA HORIZONTAL

Nome:		**Técnica de Leitura Horizontal:**
Data:	**Hora de Início:**	**Hora de Término:**
Documento 1:		**Documento 2:**

Tipo de Defeitos:		
Diagrama de Classes	**Diagrama de Seqüência**	**Diagrama de Estados**
(AT) Atributos	**(AO)** Ator	**(ES)** Estado
(CO) Comportamento	**(OB)** Objeto	**(EV)** Evento
(CN) Condição	**(ME)** Mensagem	**(CO)** Comportamento
(HE) Herança	**(DA)** Dado	**(CN)** Condição
(RE) Relacionamento	**(CN)** Condição	**(RS)** Restrição
(CA) Cardinalidades	**(RS)** Restrição	
(PA) Papel		
(CL) Classe		
(IN) Interface		
(RS) Restrição		

Tipo de Discrepância:	**Classificação do Defeito:**
(1) Presente no Documento 1, mas não no Documento 2.	**(OM) Omissão:** Ausência de conceito.
	(FI) Fato Incorreto: Interpretação errônea.
(2) Presente no Documento 2, mas não no Documento 1.	**(IN) Inconsistência:** Representação em desacordo.
	(AM) Ambigüidade: Representação não clara.
(3) Presente em ambos os documentos, mas inconsistente ou ambíguo.	**(IE) Informação Estranha:** Não se aplica ao domínio.
	(OU) Outro: Não presente na classificação.
(4) Presente em ambos os documentos, mas usando uma representação ou notação incorreta.	**Grau de Severidade do Defeito:**
	Não Sério (NS): requer apenas correção de um diagrama.
(5) Presente em ambos os documentos, mas representa informação estranha.	**Intermediário (IN):** requer correção nos dois diagramas.
(6) Faltando em ambos os documentos.	**Sério (SE):** requer uma análise do sistema, para reestruturação do(s) diagrama(s).

Preencha a tabela com as discrepâncias encontradas, procurando utilizar todos os campos existentes referente:

Discrepância	Tipo de Defeito	Palavra Chave	Tipo de Discrepância	Classificação do Defeito	Grau de Severidade do Defeito	Questão relativa ao Defeito
[1]						
[2]						
[3]						
[4]						
[5]						
[6]						
[7]						
[8]						
...						

Utilize a estrutura a seguir para comentar cada uma das discrepâncias encontradas durante a inspeção:

Discrepância	Comentário
[1]	
[2]	
[3]	
...	

ANEXO 2.2. FORMULÁRIO DE ANOTAÇÃO DE DEFEITOS PARA LEITURA VERTICAL

Nome:		**Técnica de Leitura Vertical:**
Data:	**Hora de Início:**	**Hora de Término:**
Documento 1:		**Documento 2:**

Tipo de Defeitos:		
Diagrama de Classes	**Diagrama de Seqüência**	**Diagrama de Estados**
(AT) Atributos	**(AO)** Ator	**(ES)** Estado
(CO) Comportamento	**(OB)** Objeto	**(EV)** Evento
(CN) Condição	**(ME)** Mensagem	**(CO)** Comportamento
(HE) Herança	**(DA)** Dado	**(CN)** Condição
(RE) Relacionamento	**(CN)** Condição	**(RS)** Restrição
(CA) Cardinalidades	**(RS)** Restrição	
(PA) Papel		
(CL) Classe		
(IN) Interface		
(RS) Restrição		

Tipo de Discrepância:	**Classificação do Defeito:**
(1) Funcionalidade do sistema ou conceito necessário foi omitido.	**(OM) Omissão:** Ausência de conceito.
	(FI) Fato Incorreto: Interpretação errônea.
(2) O diagrama está incorreto em relação aos requisitos do sistema.	**(IN) Inconsistência:** Representação em desacordo.
	(AM) Ambigüidade: Representação não clara.
(3) Como o diagrama implementa estes requisitos é ambíguo, não completamente especificado ou inconsistente.	**(IE) Informação Estranha:** Não se aplica ao domínio.
	(OU) Outro: Não presente na classificação.
(4) Alguma informação do diagrama é estranha, isto é, não mencionada pelos requisitos.	**Grau de Severidade do Defeito:**
	Não Sério (NS): requer apenas correção de um diagrama.
(5) Outro problema que aparece no(s) diagrama(s).	**Intermediário (IN):** requer correção nos dois diagramas.
	Sério (SE): requer uma análise do sistema, para reestruturação do(s) diagrama(s).

Preencha a tabela com as discrepâncias encontradas, procurando utilizar todos os campos existentes:

Discrepância	Tipo de Defeito	Palavra Chave	Tipo de Discrepância	Classificação do Defeito	Grau de Severidade do Defeito	Identificação do Requisito	Questão relativa ao Defeito
[1]							
[2]							
[3]							
[4]							
[5]							
[6]							
[7]							
[8]							
...							

Utilize a estrutura a seguir para comentar cada uma das discrepâncias encontradas durante a inspeção:

Discrepância	Comentário
[1]	
[2]	
[3]	
...	

ANEXO 3

FORMULÁRIO DE RELATO SOBRE USO DA TÉCNICA OORT

DADOS PESSOAIS	
Nome:	
Telefone para Contato:	
E-mail:	
Profissão:	
Nível de Formação:	**Conhecimento sobre o Método UML:**
( ) Superior ( ) Mestrando ( ) Mestre ( ) Doutorando ( ) Doutor	( ) 1 ano a 1 ano e 11 meses ( ) 2 anos a 2 anos e 11 meses ( ) 3 anos a 3 anos e 11 meses ( ) 4 anos ou mais
Experiência sobre Inspeção de Software:	**Conhecimento sobre a Técnica OORT:**
( ) Nenhuma Experiência ( ) Experiência em Sala de Aula ( ) Experiência em Empresa ( ) Experiência em Sala de Aula e Empresa	( ) Nenhuma Experiência ( ) Experiência em Sala de Aula ( ) Experiência em Empresa ( ) Experiência em Sala de Aula e Empresa

A partir das perguntas abaixo, aponte os pontos positivos e as dificuldades da utilização da técnica OORT, de acordo com a escala de 1 (um) a 5 (cinco), onde 1 (um) representa a menor satisfação e 5 (cinco) a maior satisfação. Comente cada pergunta, buscando indicar possíveis problemas, dificuldades e/ou sugestões. Após responder as perguntas, inclua comentários de ordem geral:

[1] Seu conhecimento sobre inspeção foi suficiente para a utilização da Técnica OORT?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[2] Seu conhecimento sobre UML foi suficiente para a utilização da Técnica OORT?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[3] A classificação de defeitos utilizada nos formulários de defeitos é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[4] A classificação de defeitos utilizada nos formulários de defeitos é completa?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[5] O tipo de defeitos utilizado nos formulários de defeitos é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[6] O tipo de defeitos utilizado nos formulários de defeitos é completo?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[7] O tipo de discrepância utilizado no formulário para preenchimento de defeitos da leitura horizontal é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[8] O tipo de discrepância utilizado no formulário para preenchimento de defeitos da leitura horizontal é completo?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[9] O tipo de discrepância utilizado no formulário para preenchimento de defeitos da leitura vertical é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[10] O tipo de discrepância utilizado no formulário para preenchimento de defeitos da leitura vertical é completo?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[11] O grau de severidade do defeito utilizado nos formulários de defeitos é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[12] O grau de severidade do defeito utilizado nos formulários de defeitos é completo?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[13] O formulário para preenchimento de defeitos da leitura horizontal é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[14] O formulário para preenchimento de defeitos da leitura horizontal é de fácil preenchimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[15] O formulário para preenchimento de defeitos da leitura horizontal é completo?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[16] O formulário para preenchimento de defeitos da leitura vertical é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[17] O formulário para preenchimento de defeitos da leitura vertical é de fácil preenchimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[18] O formulário para preenchimento de defeitos da leitura vertical é completo?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[19] Todas as questões referentes à OORT 1 puderam ser compreendidas?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[20] Todas as questões referentes à OORT 1 são convenientes ou necessárias?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[21] As questões referentes à OORT 1 são suficientes para detectar os defeitos?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[22] Os artefatos (diagramas e documentos) utilizados na OORT 1 são suficientes?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[23] Todas as questões referentes à OORT 2 puderam ser compreendidas?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[24] Todas as questões referentes à OORT 2 são convenientes ou necessárias?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[25] As questões referentes à OORT 2 são suficientes para detectar os defeitos?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[26] Os artefatos (diagramas e documentos) utilizados na OORT 2 são suficientes?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[27] Todas as questões referentes à OORT 3 puderam ser compreendidas?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[28] Todas as questões referentes à OORT 3 são convenientes ou necessárias?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[29] As questões referentes à OORT 3 são suficientes para detectar os defeitos?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[30] Os artefatos (diagramas e documentos) utilizados na OORT 3 são suficientes?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[31] Todas as questões referentes à OORT 4 puderam ser compreendidas?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[32] Todas as questões referentes à OORT 4 são convenientes ou necessárias?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[33] As questões referentes à OORT 4 são suficientes para detectar os defeitos?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[34] Os artefatos (diagramas e documentos) utilizados na OORT 4 são suficientes?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[35] Todas as questões referentes à OORT 5 puderam ser compreendidas?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[36] Todas as questões referentes à OORT 5 são convenientes ou necessárias?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[37] As questões referentes à OORT 5 são suficientes para detectar os defeitos?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[38] Os artefatos (diagramas e documentos) utilizados na OORT 5 são suficientes?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[39] Todas as questões referentes à OORT 6 puderam ser compreendidas?
1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )
Comentário:

[40] Todas as questões referentes à OORT 6 são convenientes ou necessárias?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[41] As questões referentes à OORT 6 são suficientes para detectar os defeitos?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[42] Os artefatos (diagramas e documentos) utilizados na OORT 6 são suficientes?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[43] Todas as questões referentes à OORT 7 puderam ser compreendidas?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[44] Todas as questões referentes à OORT 7 são convenientes ou necessárias?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[45] As questões referentes à OORT 7 são suficientes para detectar os defeitos?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[46] Os artefatos (diagramas e documentos) utilizados na OORT 7 são suficientes?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[47] De modo geral, a Técnica de Inspeção Orientada a Objetos (OORT) é de fácil entendimento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[48] A partir da inspeção utilizando a Técnica OORT, você considera que seus conhecimentos sobre inspeção tiveram um aumento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[49] A partir da inspeção utilizando a Técnica OORT, você considera que seus conhecimentos sobre UML tiveram um aumento?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

[50] Você utilizaria a Técnica OORT novamente, seja somente para uma inspeção ou como parte de um projeto de desenvolvimento de software?

1 ( ) 2 ( ) 3 ( ) 4 ( ) 5 ( )

Comentário:

Comentários Gerais:

ANEXO 4

TABELA DE CLASSIFICAÇÃO DO DEFEITO

TIPO DE DEFEITO	DESCRIÇÃO DO DEFEITO
Omissão	Um ou mais diagramas, que deveriam conter algum conceito da descrição do sistema, não contém uma representação para o conceito. **Exemplo:** Um método aparece no diagrama de classes, mas não aparece no diagrama de seqüência.
Fato Incorreto	O diagrama contém uma interpretação errônea de um conceito existente na descrição do sistema. **Exemplo:** A descrição do sistema indica que o tempo de execução deve ser menor que um minuto (<1). Mas, em um determinado diagrama este mesmo tempo é representado como menor ou igual a um minuto (<=1).
Inconsistência	A representação de um conceito em um diagrama, está em desacordo com a representação do mesmo conceito no mesmo diagrama ou em outro diagrama. **Exemplo:** Uma restrição que pede que t seja maior que zero (t>0) na descrição do sistema ou em um diagrama, mas no diagrama em comparação, a restrição é representada como t maior ou igual a zero (t>=0).
Ambigüidade	A representação de um conceito em um projeto não está clara, e poderia causar ao usuário (desenvolvedor) do documento uma interpretação errônea do significado do conceito. **Exemplo:** Um diagrama de estados, possui um evento "tempo até 12 segundos".
Informação Estranha	O projeto inclui informações que, apesar de aparentemente verdadeiras, não se aplicam ao domínio do sistema e não deveriam ser incluídas nos diagramas. **Exemplo:** Uma classe "Cliente" no diagrama de classe possui um atributo "código do fornecedor" em um sistema de controle de entregas nos clientes.