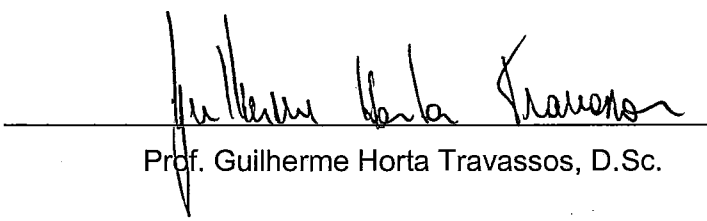
INFRA-ESTRUTURA COMPUTACIONAL DE APOIO AO PROCESSO DE INSPEÇÃO DE SOFTWARE

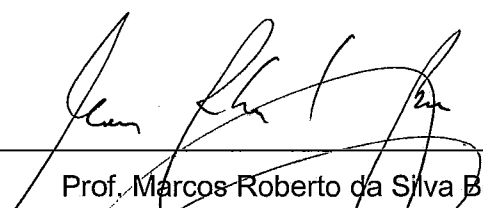
Marcos Kalinowski

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

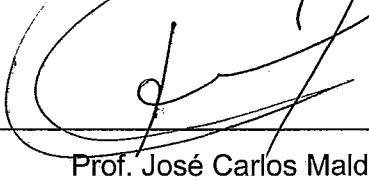
Aprovada por:



Prof. Guilherme Horta Travassos, D.Sc.



Prof. Marcos Roberto da Silva Borges, Ph.D.



Prof. José Carlos Maldonado, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 2004

KALINOWSKI, MARCOS

Infra-Estrutura Computacional de Apoio ao Processo de Inspeção de Software [Rio de Janeiro] 2004

X, 109 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2004)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Inspeção de Software

2. Ferramenta CASE

I. COPPE/UFRJ II. Título (série)

À minha família.

Agradecimentos

Ao Professor Guilherme Horta Travassos por me guiar pelos caminhos da vida acadêmica, por me apresentar a engenharia de software experimental e outros conceitos que até então para mim eram desconhecidos, por orientar minha tese, por participar do meu estudo de caso desempenhando o papel de moderador da inspeção, por ter adotado a infra-estrutura resultante deste trabalho em inspeções reais realizadas na COPPE/UFRJ, por participar da banca examinadora da minha tese, por sua amizade e atenção, por sua compreensão e confiança em relação a minha capacidade de realizar este trabalho e por ter me ajudado a desenvolver uma visão mais crítica a respeito de trabalhos de pesquisa e a crescer como pesquisador e engenheiro de software.

Ao Professor Marcos Roberto da Silva Borges, pelo aprendizado durante a realização de disciplinas de graduação no DCC/UFRJ e do meu projeto final de curso, que foi utilizado como base para a elaboração deste trabalho, e por participar da banca examinadora da minha tese.

Ao Professor José Carlos Maldonado por suas sugestões significativas durante os workshops realizados na COPPE/UFRJ e na USP São Carlos e por participar da banca examinadora da minha tese.

À Professora da área de Engenharia de Software da COPPE/UFRJ, Claudia Maria Lima Werner, pela contribuição ao meu aprendizado no decorrer do curso.

À Professora da área de Engenharia de Software da COPPE/UFRJ, Ana Regina da Rocha, pela contribuição ao meu aprendizado no decorrer do curso.

Aos demais professores da COPPE/UFRJ com que eu tive a oportunidade de realizar disciplinas que contribuíram para o meu aprendizado.

Aos professores do curso Bacharelado em Ciência da Computação do IM-DCC/UFRJ, pela excelência do curso, que me forneceu a base necessária para cursar o mestrado.

Aos participantes voluntários do meu experimento. Em especial ao Professor Marcio Barros, que se prontificou a ser o primeiro participante.

À equipe de Engenharia de Software Experimental (http://www.cos.ufrj.br/~ese) da COPPE/UFRJ por suas contribuições neste trabalho. Em especial ao aluno Leonardo Reis, que se prontificou a ter o documento de requisitos para a elaboração de sua tese inspecionado utilizando a infra-estrutura elaborada neste trabalho, desempenhando o papel do autor do documento no meu estudo de caso e fornecendo sugestões interessantes.

Aos alunos da disciplina Tópicos Especiais em Engenharia de Software IV (2004), que participaram do meu estudo de caso fornecendo importantes sugestões de melhoria. Estes alunos têm ainda utilizado a infra-estrutura em inspeções reais realizadas ao longo do curso.

Aos amigos da COPPE/UFRJ. Em especial Alexandre Corrêa, Alexandre Dantas, Aline Pires, Ana Cândida Natali, Arilo Neto, Cristina Mendonça, Gladys Lima, Gleison Santos, Gustavo Veronese, Hamilton Oliveira, Hélio Costa, Jeann Andrade, Leonardo Murta, Leonardo Reis, Luis Felipe Silva, Marcelo Costa, Marco Antônio Araújo, Marco Mangan, Marco Lopes, Mariano Montoni, Paula Mian, Rafael Barcelos, Rodrigo Oliveira Spínola, Sávio Figueiredo, Sômulo Mafra, Tayana Conte e Wladmir Chappeta. Pelo convívio gratificante que temos.

Aos meus demais amigos, principalmente, Ana Olívia, Alexandre Meire, Alex Sandro, Aldinei Bastos, Bruno Crotman, Carola Kurzhals, Fabio Foward, Gustavo Azevedo, Guilherme Duncan, Igor Mayerhofer, Karlos Eduardo Gribel, Larissa Erbacher, Maria Cecília Terra, Natally Rocha, Rafael Azevedo, Renata Araújo, Rodrigo Kalinowski e Sascha Kalinowski pelo incentivo e motivação.

À CAPES e ao CNPq pelo apoio financeiro.

E, finalmente, mas não menos importante, a Deus por permitir a conclusão de mais esta etapa importante da minha vida.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

INFRA-ESTRUTURA COMPUTACIONAL DE APOIO AO PROCESSO DE INSPEÇÃO DE SOFTWARE

Marcos Kalinowski

Junho / 2004

Orientador: Guilherme Horta Travassos
Programa: Engenharia de Sistemas e Computação

Inspeções de software melhoram a qualidade do software pela análise de seus artefatos, detectando seus defeitos para serem removidos antes que estes artefatos sejam passados para as próximas atividades do processo de desenvolvimento de software. Conhecimento a respeito de inspeções de software tem sido adquirido através de estudos experimentais. No entanto, muito deste conhecimento não é considerado nas propostas de apoio computacional existentes.

Nesta tese é descrita uma infra-estrutura computacional, cujo conjunto de requisitos foi derivado de conhecimento adquirido através de estudos experimentais, para apoiar inspeções de software. Para avaliar a viabilidade desta infra-estrutura dois estudos foram conduzidos: um estudo de caso, que mostrou a viabilidade de se utilizar a infra-estrutura em inspeções reais e um estudo experimental que avaliou o apoio à atividade de planejamento de inspeções de software. Os resultados do estudo experimental sugerem que participantes inexperientes são capazes de elaborar planos de inspeções mais eficientes e em menos tempo, quando a infra-estrutura é utilizada.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

COMPUTATIONAL FRAMEWORK FOR SUPPORTING THE SOFTWARE INSPECTION PROCESS

Marcos Kalinowski

June / 2004

Advisor: Guilherme Horta Travassos
Department: Computer and System Engineering

Software inspections improve software quality by its artifacts' analysis, detecting their defects for removal before these artifacts are delivered to the following software life cycle activities. Some knowledge regarding software inspections have been acquired by empirical studies. However, many of this knowledge are not considered in the current computational support proposals.

This work describes a computational framework whose requirements set was derived from knowledge acquired by empirical studies to support software inspections. To evaluate the feasibility of such framework two studies have been accomplished: one case study, which has shown the feasibility of using the framework to support inspections, and an experimental study that evaluated the supported software inspection planning activity. Preliminary results of this experimental study suggested that unexperienced subjects are able to plan inspections with higher defect detection effectiveness, and in less time, when using this computational framework.

Sumário

Capítulo 1
Introdução

Neste capítulo, apresentamos as questões que levaram a realização deste trabalho e a organização desta dissertação.

1.1 – Motivação

O custo da correção de defeitos aumenta exponencialmente com o decorrer do ciclo de vida de software (BOEHM, 1981). Portanto, atividades devem ser agendadas para encontrar defeitos tão cedo quanto possível. Inspeções de software (FAGAN, 1976) têm se apresentado como uma abordagem eficiente e de baixo custo para encontrar defeitos, reduzindo o retrabalho e aumentando a qualidade dos produtos (BOEHM e BASILI, 2001).

Ao longo dos anos, muitas teorias e técnicas referentes a inspeções de software têm sido propostas, incluindo variantes do processo de inspeção, técnicas de estimativa para o número de defeitos em documentos e a eficiência de inspeções, técnicas de leitura de artefatos visando a aumentar a eficiência de inspetores em detectar defeitos, e diretrizes de apoio a tarefas do processo de inspeção referentes a tomadas de decisão. Algumas destas teorias e técnicas têm sido avaliadas através de estudos experimentais, e podem ser considerados como conhecimento da área de inspeções de software. SAUER *et al.*, (2000), por exemplo, descrevem uma reorganização do processo de inspeção baseado em resultados de estudos experimentais. SHULL *et al.* (2003), por sua vez, extraíram conhecimento sobre técnicas de leitura de resultados de estudos experimentais.

Resultados de um *survey* (CIOLKOWSKI *et al.*, 2003) mostram que embora muitas organizações de software realizem inspeções, elas o fazem de forma pouco sistematizada e utilizam pouco conhecimento sobre inspeções. Assim, o verdadeiro potencial das inspeções raramente é alcançado na prática.

Muitas propostas de suporte ferramental surgiram para resolver problemas relacionados à aplicação de inspeções de software na prática (MACDONALD e MILLER, 1999). A maioria dessas propostas tem foco na detecção de defeitos ou na

inspeção de artefatos específicos. Recentemente, duas ferramentas têm sido utilizadas em estudos experimentais e merecem atenção especial, GRIP (HALLING *et al.*, 2001) e IBIS (LANUBILE e MALARDO, 2002). No entanto, analisando conhecimento de pesquisa a respeito de inspeções de software, parece ser possível explorar informações experimentalmente avaliadas para fornecer mais apoio a alguns pontos de tomada de decisão do processo de inspeção. Não identificamos na literatura, por exemplo, propostas que utilizassem o conhecimento descrito em (ADAMS, 1999) (BIFFL *et al.*, 2003) (CARVER, 2003) e (VITHARANA e RAMAMURTHY, 2003).

Baseado neste cenário, este trabalho apresenta uma nova proposta de infra-estrutura computacional para apoiar o processo de inspeção de software.

1.2 – Objetivo da Tese

O objetivo deste trabalho é a elaboração e disponibilização de uma infra-estrutura computacional de apoio ao processo de inspeção de software que possa ser utilizada para inspecionar os diferentes artefatos produzidos durante o ciclo de vida de software por equipes geograficamente distribuídas. Para fornecer um apoio adequado os requisitos para esta infra-estrutura são derivados de conhecimento selecionado sobre inspeções de software, muito do conhecimento selecionado foi adquirido por estudos experimentais. Além disto, os aspectos colaborativos do processo de inspeção precisam ser considerados e a infra-estrutura deve fornecer soluções viáveis para a colaboração entre os participantes e a realização de suas tarefas de inspeção, dentro do contexto tecnológico da atualidade.

A partir do conjunto de requisitos, algumas decisões de projeto foram tomadas e a infra-estrutura, denominada ISPIS, foi implementada. ISPIS trata do apoio ao processo de inspeção como um todo, tendo como foco principal apoiar seus pontos de tomada de decisão. ISPIS foi construída de forma a servir como o arcabouço de uma arquitetura extensível para apoio à inspeção de software pela integração de ferramentas independentes desenvolvidas em projetos de pesquisa da COPPE/UFRJ. Assim, ela pode ser utilizada em conjunto com outras ferramentas que fornecem apoio detalhado para a aplicação de técnicas de detecção de defeitos em diferentes tipos de artefatos.

Existem duas expectativas principais para ISPIS: (1) sua disponibilização possibilita a realização de inspeções de forma mais sistematizada por organizações na

prática e (2) o apoio para a tomada de decisão induz decisões corretas ao longo do processo de inspeção de software, possibilitando inspeções mais eficientes.

Visando dar um passo inicial no sentido de avaliar estas expectativas um estudo experimental para avaliar a proposta de apoio de ISPIS ao planejamento de inspeções e um estudo de caso para avaliar a viabilidade de utilizar ISPIS em inspeções reais foram conduzidos.

1.3 – Organização deste Trabalho

Além desta introdução, esta tese é composta por mais cinco capítulos e três apêndices.

No Capítulo 2, *Inspeção de Software*, o conceito de inspeção de software é introduzido e os seus benefícios de sua aplicação são apresentados. Nesse capítulo é ainda discutido o estado da prática e o apoio ferramental atualmente existente.

No Capítulo 3, *Proposta de Infra-Estrutura de Apoio ao Processo de Inspeção de Software*, uma proposta para apoio ao processo de inspeção de software, elaborada utilizando como conjunto básico de requisitos conhecimento adquirido através de estudos experimentais relacionados a inspeções de software, é apresentada.

No Capítulo 4, *Infra-Estrutura Computacional de Apoio ao Processo de Inspeção de Software*, é descrita a infra-estrutura implementada com base na proposta de apoio apresentada no Capítulo 3. São descritos nesse capítulo, de forma detalhada, as decisões de projeto, a arquitetura da infra-estrutura e seu funcionamento detalhado.

No Capítulo 5, *Avaliação da Infra-Estrutura*, são descritos um experimento para avaliar o apoio da infra-estrutura ISPIS à atividade de planejamento do processo de inspeção e um estudo de caso avaliando a viabilidade de se utilizar ISPIS em inspeções reais.

O Capítulo 6, *Considerações Finais*, conclui a tese descrevendo suas contribuições e seus trabalhos futuros.

O Apêndice A, *Diagrama e Descrição das Classes de Domínio*, contém os diagramas de classes das classes de domínio de ISPIS e da PatternFlow. Além disto, a descrição de cada uma destas classes é fornecida.

O Apêndice B, *Modelo Entidade-Relacionamento para a Persistência dos Dados*, apresenta o esquema relacional utilizado para a persistência, identificando quais tabelas fazem parte da PatternFlow e quais formam sua extensão para ISPIS.

O Apêndice C, *Instrumentos do Experimento*, contém os seguintes instrumentos utilizados na operação do experimento: consentimento de participação, caracterização dos participantes, problema modelado e questionário de acompanhamento.

Capítulo 2
Inspeção de Software

Neste capítulo, inspeções de software, seus benefícios e o processo de inspeção são apresentados. Por fim, o estado da prática e o apoio ferramental atualmente existente são descritos.

2.1 – Introdução

Na engenharia de software, assim como em outras disciplinas de engenharia, é necessário considerar variáveis como esforço, produtividade, tempo e custo de desenvolvimento. Essas variáveis são afetadas negativamente quando artefatos defeituosos são produzidos, devido ao retrabalho para corrigir defeitos. Sabe-se, ainda, que o custo do retrabalho para correção de defeitos aumenta na medida que o processo de desenvolvimento progride (BOEHM e BASILI, 2001). Desta forma, iniciativas devem ser realizadas no sentido de encontrar e corrigir defeitos tão logo sejam introduzidos. Uma abordagem que tem se mostrado eficiente e de baixo custo para encontrar defeitos, reduzindo o retrabalho e melhorando a qualidade dos produtos é a revisão dos artefatos produzidos ao longo do processo de desenvolvimento de software (FAGAN, 1976) (GILB e GRAHAM, 1993).

Inspeção de software (FAGAN, 1976) é um tipo particular de revisão que pode ser aplicado a todos os artefatos de software e possui um processo de detecção de defeitos rigoroso e bem definido. A Figura 2.1 ilustra a possibilidade de se realizar inspeções nos diferentes artefatos de software.

De forma resumida, o processo tradicional de inspeção envolve o planejamento da inspeção, indivíduos revisando um determinado artefato, um encontro em equipe para discutir e registrar os defeitos, a passagem dos defeitos para o autor do artefato para que possam ser corrigidos e uma avaliação final sobre a necessidade de uma nova inspeção.

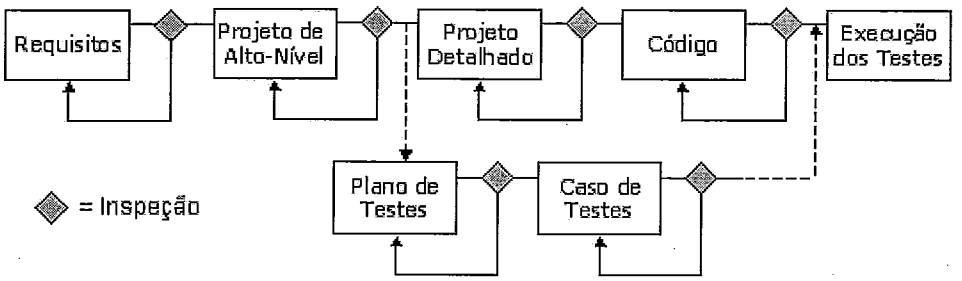


Figura 2.1. Inspeções de Software nos Diferentes Artefatos- Adaptado de (ACKERMAN *et al.*, 1989)

A importância de inspeções na redução do retrabalho e na garantia da qualidade de software está bem documentada na literatura (FAGAN, 1976) (ACKERMAN *et al.*, 1989) (GILB, GRAHAM, 1993) (SAUER *et al.*, 2000) e é discutida em maiores detalhes na seção 2.3.

A seção seguinte apresenta a definição de alguns conceitos utilizados ao longo deste trabalho. Na seção 2.3, alguns benefícios de se realizar inspeções em artefatos produzidos ao longo do processo de desenvolvimento de software são descritos. Em seguida, na seção 2.4, conceitos sobre técnicas de leitura para detecção de defeitos em artefatos de software são apresentados. A seção 2.5 descreve o processo de inspeção de software e suas características. O estado atual da utilização de inspeções na prática e do suporte ferramental existente é discutido na seção 2.6. Por fim, a seção 2.7 concluí este capítulo.

2.2 – Definição dos Conceitos

Nesta seção, são definidos termos utilizados neste trabalho, evitando ambigüidades, uma vez que terminologias inconsistentes sobre estes termos podem ser encontradas na literatura. Dois termos serão definidos aqui: defeito e discrepância.

O termo defeito muitas vezes é utilizado de forma genérica. No entanto é importante ter em mente que sua interpretação dependerá do contexto em que ele for utilizado. Defeitos encontrados através de revisões estarão relacionados às faltas no artefato sendo revisado. Quando um defeito se manifesta através de atividades de teste, por sua vez, estaremos lidando com uma falha no software. Estas definições

seguem a terminologia padrão para Engenharia de Software do IEEE (IEEE 610.12, 1990):

- **Erro.** É um defeito cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta.
- **Falta.** É uma manifestação concreta de um erro num artefato de software. Um erro pode resultar em diversas faltas.
- **Falha.** É o comportamento operacional do software diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversas faltas e algumas faltas podem nunca causar uma falha.

Em alguns momentos o termo discrepância será utilizado. Este termo refere-se a um suposto defeito encontrado. No nosso contexto, uma discrepância poderá ser considerada um defeito de fato ou um chamado falso positivo.

Para obter uma classificação para os defeitos encontrados nas revisões (as faltas), partimos do fato de que todos os artefatos gerados durante o desenvolvimento de software utilizam como base o documento de requisitos ou artefatos gerados a partir deste. Desta forma, as classes de defeito seriam os tipos de defeito presentes em documentos de requisitos acrescidos dos tipos de defeitos introduzidos pela transformação de artefatos ao longo do desenvolvimento de software.

Um padrão IEEE (IEEE 830, 1998), que recomenda práticas para especificação de requisitos de software, define atributos de qualidade que um documento de requisitos deve possuir. Foi considerado que a falta de qualquer um destes atributos constituiria um tipo de defeito (SHULL, 1998). Assim, a seguinte taxonomia foi definida:

- **Omissão.** (1) Algum requisito importante relacionado à funcionalidade, ao desempenho, às restrições de projeto, ao atributo, ou à interface externa não foi incluído; (2) não está definida a resposta do software para todas as possíveis situações de entrada de dados; (3) faltam seções na especificação de requisitos; (4) faltam referências de figuras, tabelas, e diagramas; (5) falta definição de termos e unidades de medidas.
- **Ambigüidade.** Um requisito tem várias interpretações devido a diferentes termos utilizados para uma mesma característica ou vários significados de um termo para um contexto em particular.
- **Inconsistência.** Dois ou mais requisitos são conflitantes.

- **Fato Incorreto.** Um requisito descreve um fato que não é verdadeiro, considerando as condições solicitas para o sistema.
- **Informação Estranha.** As informações fornecidas no requisito não são necessárias ou mesmo usadas.
- **Outros.** Outros defeitos como a inclusão de um requisito numa seção errada do documento.

Analisando as transformações da informação no desenvolvimento de software, ilustradas na Figura 2.2, TRAVASSOS *et al.*, (2001) identificaram a introdução de defeitos que se encaixam nas mesmas classes definidas acima. Portanto, acreditamos que esta classificação possa ser utilizada de forma genérica para os diferentes tipos de artefatos gerados ao longo do processo de desenvolvimento de software.

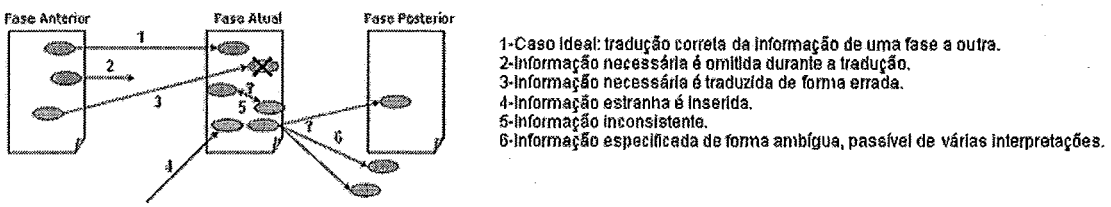


Figura 2.2. Transformação da Informação no Desenvolvimento de Software. Adaptado de (TRAVASSOS *et al.*, 2001).

É importante ressaltar que estas classes genéricas de defeitos podem ser divididas em classes mais específicas, dependendo da necessidade. Além disto, esta classificação não pretende ser definitiva e cada organização pode acrescentar mais tipos de defeito de acordo com suas necessidades (SHULL *et al.*, 2000).

2.3 – Benefícios da Aplicação de Inspeções de Software

Nesta seção, são apresentados benefícios da aplicação de inspeções. No entanto, para obter tais benefícios é necessário aplicar o processo de inspeção de forma sistemática, fazendo com que a inspeção retenha conhecimento obtido durante sua aplicação e que este conhecimento seja utilizado de forma adequada.

2.3.1 – Esforço, Produtividade, Tempo e Custo

De acordo com (BOEHM e BASILI, 2001) o esforço gasto por organizações de software com retrabalho varia em média entre 40% e 50% do esforço total do desenvolvimento de um projeto. Uma estimativa da distribuição do retrabalho pelas atividades de desenvolvimento de software (WHEELER *et al.*, 1996) está ilustrada na Figura 2.3.

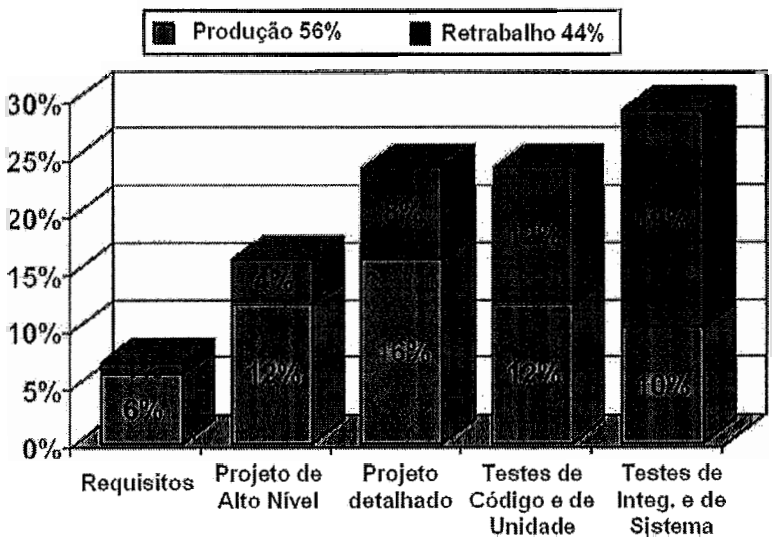


Figura 2.3. Distribuição do retrabalho pelas atividades de desenvolvimento de software. Adaptado de (WHEELER *et al.*, 1996).

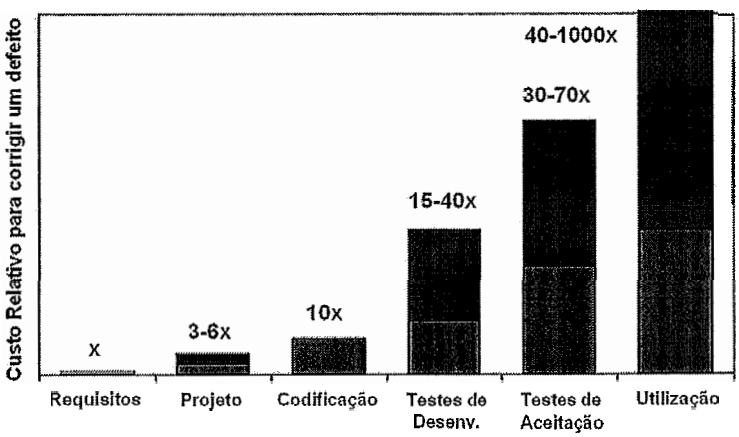


Figura 2.4. Custo relativo para corrigir um defeito. Adaptado de (BOEHM, 1981).

Analisando a Figura 2.3, é possível verificar que o retrabalho tende a aumentar na medida que o desenvolvimento progride. Uma das razões para isto é o aumento no esforço para corrigir defeitos nas atividades finais do processo de desenvolvimento. Através da análise de 63 projetos, BOEHM (1981) apresenta o custo relativo da correção de defeitos encontrados em cada uma das atividades de desenvolvimento (Figura 2.4).

Assim, um dos maiores benefícios de se utilizar inspeções de software é a detecção de defeitos nas fases iniciais do processo de desenvolvimento de software, facilitando a correção destes defeitos com menor esforço e custo. Desta forma, de acordo com (JONES, 1991), o esforço com retrabalho é reduzido em média para 10% a 20% do esforço total de desenvolvimento. Esta redução no retrabalho pode implicar em melhorias significativas para a produtividade de software. De acordo com (BOEHM *et al.*, 2000) a maior redução de esforço é gerada pela melhoria de (1) maturidade de processos de software, (2) arquiteturas de software e (3) gerência de riscos é proveniente da redução do retrabalho. Resultados experimentais mostram como este benefício pode afetar as variáveis esforço, produtividade, tempo e custo, mencionadas na seção anterior:

Esforço. O departamento de desenvolvimento da Ericsson em Oslo, Noruega, calculou uma redução bruta do esforço total de desenvolvimento em 20% aplicando inspeções (CONRADI *et al.*, 1999). Além disto, resultados de (LAITENBERGER e ATKINSON, 1999) mostram que a introdução de inspeções de projeto pode reduzir o esforço com retrabalho em 44%.

Produtividade. De acordo com (GILB e GRAHAM, 1993), inspeções aumentam a produtividade de 30% a 50%;

Tempo. De acordo com (GILB e GRAHAM, 1993), inspeções reduzem o tempo de desenvolvimento de 10% a 30%;

Custo. Resultados de (LAITENBERGER e ATKINSON, 1999) mostram que a introdução de inspeções de código pode reduzir os custos de implementação de projetos em 39%.

Uma estimativa de (WHEELER *et al.*, 1996) sobre o custo de desenvolvimento quando inspeções são aplicadas, quando comparado ao desenvolvimento sem utilizar inspeções, se encontra na Figura 2.5. Esta estimativa representa bem os benefícios apresentados nesta seção. É possível observar que utilizando inspeções se obtém um

aumento na produtividade, já que projetos são concluídos em menos tempo e envolvem menos gastos.

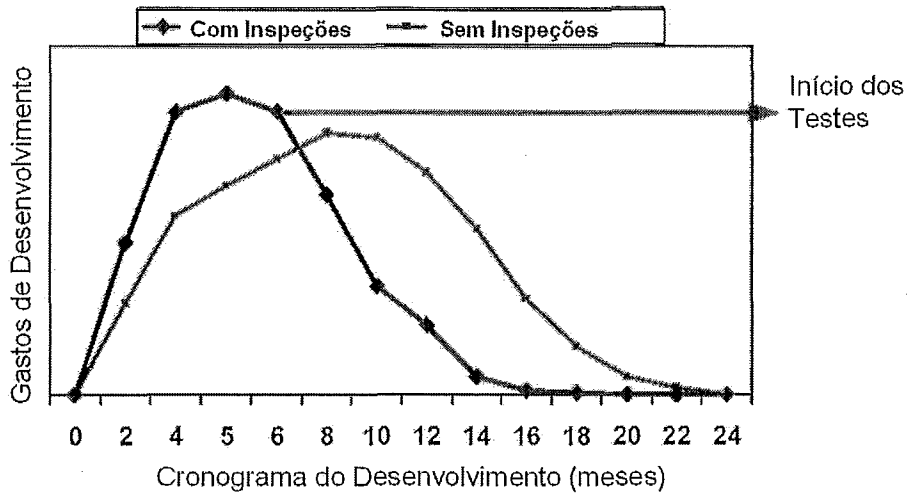


Figura 2.5. Estimativa dos gastos de desenvolvimento utilizando e não utilizando inspeções. Adaptado de (WHEELER et al., 1996).

2.3.2 – Qualidade de Software

De acordo com Pressman (2001), qualidade de software é a conformidade a: (1) requisitos funcionais e não funcionais que têm sido explicitamente declarados, (2) padrões de desenvolvimento que tenham sido claramente documentados e (3) características implicitamente esperadas de todo software a ser desenvolvido. De forma resumida, qualidade consiste de um conjunto de requisitos e de um produto ou serviço que esteja em conformidade com estes requisitos e, por esta razão, atenda completamente às necessidades dos clientes. Entre as atividades que podem ser utilizadas para verificar a qualidade de software se encontram:

- Revisões de software;
- Testes;
- Padrões e procedimentos formais;
- Controle de mudanças;
- Métricas de software;
- Procedimentos para coleção e disseminação de informações.

A importância das revisões na garantia da qualidade é destacada pelo modelo CMMI, que identifica revisões como uma das áreas chave (*key process area*) do processo de desenvolvimento de software (CMMI Product Team, 2001). Sabe-se ainda que inspeções de software, em particular, capturam em torno de 60% dos defeitos de artefatos (BOEHM e BASILI, 2001) o que deixa explícita a sua contribuição para a melhoria da qualidade.

Uma outra maneira de detectar defeitos em artefatos é através da aplicação de testes. No entanto, a aplicação de testes descobre apenas sintomas de problemas e, desta forma, pode ocasionar um trabalho refinado e custoso para detectar o defeito que causou o sintoma. De acordo com BASILI e SELBY (1987) inspeções descobrem defeitos que podem não ser descobertos nos testes. BOEHM e BASILI (2001) ressaltam ainda que inspeções e testes capturam diferentes tipos de defeito e em diferentes momentos do processo de desenvolvimento de software. Portanto, é interessante aplicar tanto inspeções quanto testes para detectar defeitos em artefatos de software.

Além disto, precedendo os testes com as inspeções, defeitos podem ser removidos nas fases iniciais do processo de desenvolvimento e os desenvolvedores terão uma visão mais ampla da complexidade do sistema. Esta visão os deixa mais bem preparados para o momento em que se confrontam com esta complexidade e com os defeitos que podem possivelmente surgir durante os testes (MELO *et al.*, 2001).

2.3.3 – Outros Benefícios

A aplicação de inspeções de forma bem planejada pode trazer diversos outros benefícios, entre eles se encontram:

- **Aprendizado.** Inspetores experientes podem tentar detectar padrões de como os defeitos ocorrem e definir diretrizes que ajudem na detecção destes. Além disto, bons padrões de desenvolvimento podem ser observados durante a inspeção, sendo possível sua descrição como melhores práticas para a organização.
- **Integração entre processos de detecção e de prevenção de defeitos.** Saber onde e quando os defeitos ocorrem pode ajudar a estabelecer planos de contingência que evitem a sua ocorrência.
- **Produtos mais inteligíveis.** Os autores dos diversos artefatos, sabendo que estes serão inspecionados, passarão a produzir artefatos de uma forma que

sua compreensão seja facilitada. A produção de artefatos mais inteligíveis, além de facilitar a inspeção, trará benefícios para as fases seguintes do processo de desenvolvimento, incluindo principalmente a fase de manutenção.

- **Dados defeituosos ajudam a melhorar o processo de software do projeto.** Analisando a causa dos defeitos encontrados (*root cause analysis*) é possível fazer ajustes no processo para evitar a ocorrência de defeitos deste mesmo tipo.

A aplicação de técnicas de leitura (ou de detecção de defeitos) de artefatos pode fazer com que mais defeitos sejam encontrados em inspeções, aumentando os benefícios de sua aplicação. A seção seguinte descreve algumas destas técnicas.

2.4 – Técnicas de Leitura de Artefatos de Software

Desenvolvedores de software atualmente são treinados a escrever artefatos como documentos requisitos, documentos de projeto, código, entre outros. No entanto, eles não são treinados em como proceder para ler tais artefatos.

Técnicas de leitura são uma série de etapas (heurísticas) preparadas para a análise individual de um artefato que permitem alcançar a compreensão necessária para uma determinada tarefa (BASILI *et al.*, 1996). A disponibilização destas técnicas visa aumentar o custo-eficiência[1] de inspetores e reduzir a influência do fator humano nos resultados de uma inspeção. Estas técnicas podem ainda fornecer modelos para escrever artefatos de maior qualidade.

Entre as técnicas de leitura mais conhecidas estão a *ad-hoc* e o uso de *checklists*[2]. Um exemplo do uso de *checklists* está na técnica de inspeção por fases (KNIGHT e MEYERS, 1991), onde checklists são utilizadas tanto para detectar defeitos quanto para avaliar características de qualidade do artefato.

---

[1] O custo-eficiência de um inspetor pode ser definido como sendo o número de defeitos encontrados por hora pelo inspetor.

[2] Uma *checklist,* no contexto de inspeções de software, é uma lista de perguntas que guiam o inspetor na detecção de defeitos.

Existem ainda técnicas de leitura mais específicas, como a baseada em perspectiva (PBR) (SHULL *et al.*, 2000) e as de projetos orientados a objeto (OORT's) (TRAVASSOS *et al.*, 1999). PBR apóia a detecção de defeitos em documentos de requisitos, enquanto OORT's apóiam a detecção de defeitos em projetos descritos em UML (BOOCH et al., 1998).

As técnicas de leitura PBR estão fundamentadas no fato de existirem diferentes papéis durante o desenvolvimento do software. PBR provê um conjunto de instruções de acordo com o papel do "consumidor" de um documento de requisitos de software (usuário, projetista, testador). PBR tem sido submetida a diversos estudos e há evidências experimentais dos seus benefícios (SHULL *et al.*, 2000). Estas evidências mostram que, com um foco mais específico, um desenvolvedor é mais efetivo na busca por defeitos, levando-se em conta a perspectiva adotada e que a combinação de diferentes perspectivas promove uma maior cobertura total de defeitos em comparação com um processo de inspeção *ad-hoc*.

As OORT's, por sua vez, constituem uma família de sete técnicas de leitura utilizadas para inspecionar artefatos UML produzidos na fase de projeto de alto-nível. Em um estudo experimental realizado na Ericsson, na Noruega (CONRADI et al., 2003), o uso da técnica melhorou o desempenho na detecção de defeitos em diagramas UML, quando comparada com a técnica até então utilizada na empresa.

2.5 – O Processo de Inspeção de Software

Nesta seção o processo tradicional de inspeção de software e algumas propostas de variantes deste processo são apresentados. É ainda apresentada uma proposta para gerenciar o processo de inspeção.

2.5.1 – O Processo Tradicional de Inspeção de Software (FAGAN, 1976)

FAGAN (1976) desenvolveu o processo tradicional de inspeção de software, uma forma detalhada de se realizar uma revisão. Neste processo, existem seis atividades principais:

- **Planejamento.** Um usuário, desempenhando o papel de moderador da inspeção, define o contexto da inspeção (descrição da inspeção, técnica a

ser utilizada na detecção de defeitos, documento a ser inspecionado, autor do documento, entre outros), seleciona os inspetores e distribui o material a ser inspecionado.

- **Apresentação.** Os autores dos artefatos a serem inspecionados apresentam as características destes. Esta fase pode ser omitida se os inspetores possuem conhecimento sobre o projeto e os artefatos que devem ser inspecionados.
- **Preparação.** Os inspetores estudam os artefatos individualmente, e eventualmente fazem anotações sobre estes produzindo uma lista de discrepâncias. O fornecimento de técnicas de leitura pode facilitar a execução desta tarefa.
- **Reunião.** Uma reunião em equipe ocorre, envolvendo o moderador, os inspetores e os autores do documento. Discrepâncias são discutidas, e classificadas como defeito ou falso positivos. A decisão final sobre a classificação de uma discrepância sendo discutida é do moderador. A solução dos defeitos não é discutida durante a reunião, que não deve exceder duas horas, uma vez que após este tempo a concentração e a capacidade de análise dos inspetores costuma reduzir drasticamente. No caso em que uma reunião precisar de mais de duas horas, é sugerido que o trabalho de inspeção continue no próximo dia.
- **Retrabalho.** O autor corrige os defeitos encontrados pelos inspetores e confirmados pelo moderador.
- **Continuação.** O material corrigido pelos autores é repassado para o moderador, que faz uma análise da inspeção como um todo e re-avalia a qualidade do artefato inspecionado. Ele tem a liberdade de decidir se uma nova inspeção deve ocorrer ou não.

A forma como estas atividades se relacionam no processo de inspeção está ilustrada na Figura 2.6. Note que a atividade de apresentação, opcional, não está representada na figura.

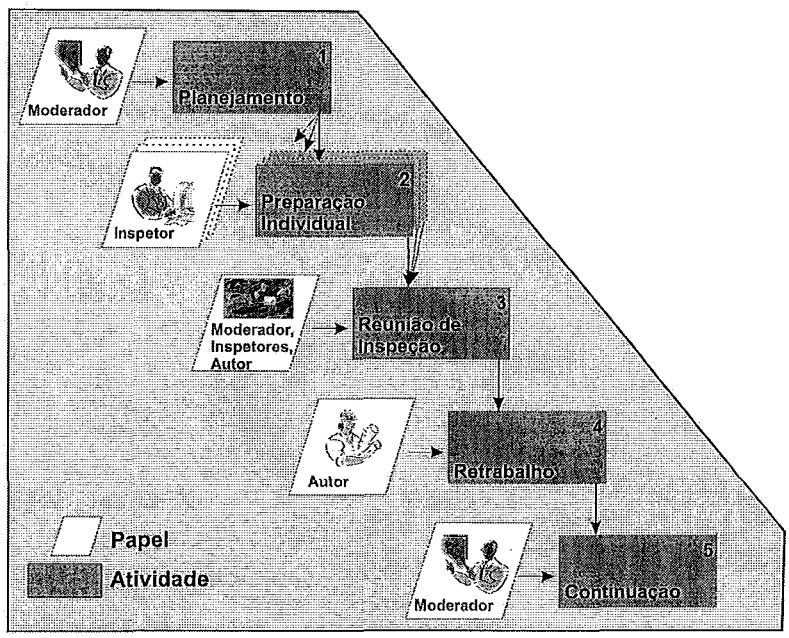


Figura 2.6. Processo de inspeção de software, conforme definido em (FAGAN, 1976).

Entre as características deste processo, temos que ele pode ser aplicado a todos os artefatos produzidos ao longo do processo de desenvolvimento, permitindo a utilização de técnicas de leitura de artefatos específicos na atividade de preparação individual. Além disto, ele possui uma estrutura rígida, com aspectos colaborativos, onde papéis, atividades e os relacionamentos entre atividades estão bem definidos. Algumas diretrizes que podem ser seguidas na instanciação de uma inspeção estão descritas em (MELO *et al.*, 2001).

Desde que o processo tradicional de inspeção de software foi definido, diversos trabalhos surgiram apresentando variantes do processo ou questionando e tentando avaliar sua estrutura.

2.5.2 – O Processo de Inspeção de Humphrey (1989)

HUMPHREY (1989), descreve uma variante do processo em que o foco da atividade de preparação individual muda de tentar entender o artefato a ser inspecionado para efetivamente encontrar seus defeitos. Assim, cada um dos inspetores entrega uma lista de discrepâncias para o moderador da inspeção antes da reunião de inspeção se iniciar. O objetivo da reunião passa então a ser discutir as discrepâncias encontradas e classificá-las como defeito ou falso positivo.

2.5.3 – Inspeções Assíncronas: FTArm (JOHNSON, 1994)

VOTTA (1993), tendo em vista a forma de se realizar inspeções descrita por HUMPHREY (1989), argumenta que reuniões de inspeção podem ser evitadas, reduzindo custos e conflitos de alocação de recursos sem sacrificar a eficiência da inspeção. O processo FTArm (acrônimo em inglês para o termo "Método de Revisão Técnica Formal Assíncrona") proposto por JOHNSON (1994) considera este argumento. As atividades do processo FTArm são: planejamento, apresentação, revisão particular, revisão pública, consolidação e reunião.

A detecção de defeitos propriamente dita começa na fase de revisão particular. Os revisores podem cadastrar pontos de dúvida, requisitar que alguma ação seja tomada ou simplesmente tecer um comentário. Somente os comentários estarão visíveis para os demais revisores. Entretanto, o moderador tem acesso a todos estes registros, o que lhe permite monitorar o progresso das revisões particulares.

Na fase de Revisão Pública todos os registros (dúvidas, ações e comentários) são visíveis aos inspetores que podem votar (concordância, discordância ou neutralidade) de forma assíncrona ou gerar novos registros. A fase só se encerra quando todos os registros estiverem estabilizados.

Um relatório condensado dos resultados da Revisão Pública é gerado pelo moderador na fase de Consolidação. Se não houver consenso, uma última reunião síncrona é realizada para que os itens pendentes sejam decididos por voto ou pela decisão unilateral do moderador.

Automatizando este processo com a ferramenta CSRS (JOHNSON et al., 1993) foi possível conduzir um estudo experimental (JOHNSON e TJAHJONO, 1997) para avaliar o argumento de VOTTA (1993), sobre a não necessidade de uma reunião de inspeção no processo de inspeção de software. Em (PORTER e JOHNSON, 1997) este estudo foi analisado em conjunto com outro estudo experimental, projetado independentemente na universidade de Maryland. Ambos indicaram não haver uma diferença substancial na efetividade da identificação de defeitos do processo com e sem reuniões de inspeção. Entretanto, os autores observaram que o processo com reuniões encontrou, ainda que em pequeno número, classes de defeitos não encontradas quando somente a inspeção individual foi realizada. Uma outra observação interessante foi a geração de um menor número de falso positivos, quando as reuniões de inspeção

ocorreram. Estes resultados são também apresentados em (JOHNSON e TJAHJONO, 1998).

Em continuação a estes trabalhos, JOHNSON (1998), em seu artigo entitulado "*Reengineering Inspections*", apresenta reuniões assíncronas como uma de suas sete recomendações para o futuro das revisões técnicas formais. As sete recomendações são:

- (1) Providenciar uma maior integração entre a inspeção e o método de desenvolvimento;
- (2) Minimizar os encontros e maximizar a assincronicidade nas inspeções;
- (3) Mudar o foco da detecção de defeitos para melhoria da qualidade dos desenvolvedores;
- (4) Construir bases de conhecimento baseadas na revisão;
- (5) Ter como saída a revisão, mas reter o conhecimento da realização desta;
- (6) Investigar tecnologias de revisão mediadas por computador;
- (7) Acabar com a limitação do tamanho da equipe de revisores.

2.5.4 – A Reorganização do Processo de Inspeção por SAUER et al. (2000)

Baseados em diversos estudos experimentais sobre inspeções de software SAUER *et al.*, (2000) propuseram uma reorganização do processo tradicional de inspeção. A reorganização visa a adequação do processo tradicional a inspeções com reuniões assíncronas e equipes geograficamente distribuídas. Assim, mudanças para reduzir o custo e o tempo total para a realização deste tipo de inspeção foram introduzidas. Este projeto alternativo para inspeções de software mantém a estrutura rígida e os aspectos colaborativos do processo tradicional e consiste basicamente em substituir as atividades de preparação e de reunião do processo tradicional por três novas atividades seqüenciais: detecção de defeitos, coleção de defeitos e discriminação de defeitos. Estas atividades podem ser descritas da seguinte forma:

- **Detecção de Defeitos.** Cada um dos inspetores selecionados pelo moderador no planejamento realizará uma atividade de detecção de defeitos. A principal tarefa desta atividade consiste em buscar defeitos no documento a ser inspecionado e assim produzir uma lista de discrepâncias.

- **Coleção de Defeitos.** O moderador agrupa as listas de discrepâncias dos inspetores. Esta atividade envolve eliminar discrepâncias repetidas (encontradas por mais de um inspetor), mantendo só um registro para cada discrepância.
- **Discriminação de Defeitos.** O moderador, o autor do documento e os inspetores discutem as discrepâncias de forma assíncrona. Durante esta discussão, algumas discrepâncias serão classificadas como falso positivo e outras como defeito. Os falso positivos serão descartados e os defeitos serão registrados em uma lista de defeitos, que então será passada para o autor para que a correção possa ocorrer.

A forma como as atividades se relacionam na reorganização do processo de inspeção de SAUER *et al.* (2000) está ilustrada na Figura 2.7.

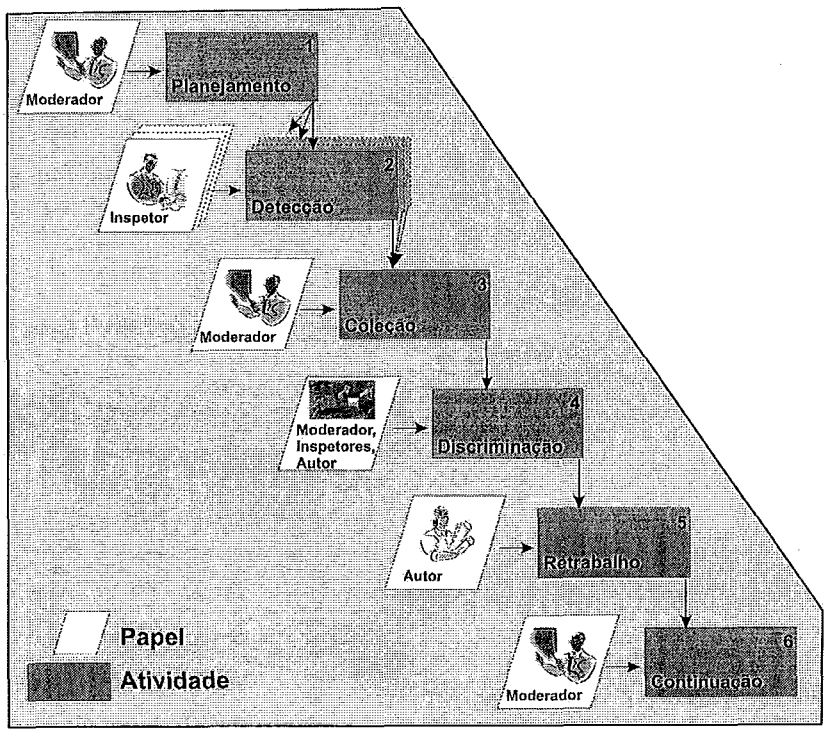


Figura 2.7. Reorganização do processo de inspeção, adaptada de (SAUER *et al.*, 2000).

Este processo permite a utilização de um número grande de inspetores em paralelo para a detecção de defeitos e, assim, aumentar a probabilidade de se encontrar defeitos difíceis de serem encontrados (LANUBILE e MALLARDO, 2002). Um

grande número de inspetores tem um efeito no custo, mas não no tempo de detecção de defeitos e não implicará em problemas de coordenação, afinal discrepâncias encontradas por mais de um inspetor são encaminhadas diretamente para a atividade de retrabalho e as discrepâncias encontradas por apenas um inspetor não precisam ser discutidas necessariamente por todos os inspetores.

2.5.5 – Gerenciando o Processo de Inspeção de Software

Sabe-se que um processo de gerência sistemático pode auxiliar no aprimoramento das inspeções. O gerenciamento ativo de inspeções (AIM) (HALLING et al., 2002) visa formalizar este processo de gerência de inspeções e é baseado na coleta, monitoramento e análise contínua dos dados sobre inspeções.

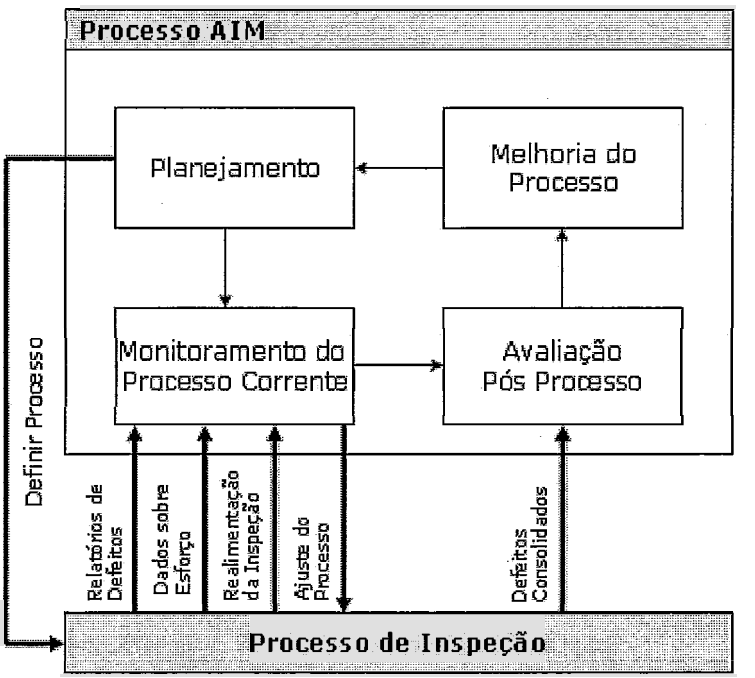


Figura 2.8. Gerenciamento Ativo de Inspeções (AIM). Adaptado de (HALLING *et al.*, 2002).

O processo AIM pode ser aplicado em paralelo ao processo de inspeção. A única sobreposição é a atividade de planejamento. A atividade de planejamento do AIM

complementa a atividade de planejamento do processo de inspeção, fazendo com que o moderador possa contar com dados que o auxiliam na tomada de decisões. Uma representação desta abordagem se encontra na Figura 2.8 e suas atividades são descritas a seguir:

- **Planejamento.** Esta atividade é semelhante à dos processos de inspeção: seleção dos inspetores, seleção da técnica de inspeção, realização ou não de reuniões, etc. Entretanto, as decisões são tomadas com base na avaliação de inspeções passadas.
- **Monitoramento do Processo Corrente.** A coleta de dados é feita em tempo real permitindo o monitoramento do processo de inspeção. Estes dados permitem a alteração de determinados parâmetros como restrições de tempo e inclusão/exclusão de inspetores.
- **Avaliação Pós Processo.** Os resultados finais do processo realizado são avaliados para identificar pontos potenciais de melhoria. O foco é entender a relação entre processo de inspeção, inspetores e artefato inspecionado.
- **Melhoria do Processo.** As informações coletadas podem permitir não só a melhoria do processo de inspeção, como também do próprio processo de desenvolvimento de software. O objetivo desta fase é otimizar aspectos como treinamento dos inspetores e técnicas de identificação de defeitos.

É importante ressaltar que o processo para gerência de inspeções AIM necessita ser experimentalmente avaliado para assegurar que ele realmente implica em melhorias para o processo de inspeção.

2.6 – Estado Atual: Prática de Revisões em Organizações de Software e Suporte Ferramental Existente

Ao longo dos anos, muito conhecimento tem sido produzido na área de inspeções de software. Incluindo variantes do processo tradicional de inspeção, técnicas de estimativa do número de defeitos de documentos e da cobertura de inspeções, técnicas de leitura de documentos visando aumentar o número de defeitos encontrados por inspetores, diretrizes para pontos de tomada de decisão do processo

de inspeção, dentre outras. Parte deste conhecimento tem sido avaliado em estudos experimentais e se mostrado adequado.

No entanto, muito deste conhecimento não tem sido utilizado na prática por organizações de software ao realizarem suas inspeções, e é negligenciado na maioria das propostas de apoio ferramental ao processo de inspeção de software.

Esta seção apresenta o estado da prática de revisões de software e o ferramental de apoio atualmente existente.

2.6.1 – Prática de Revisões em Organizações de Software

Em um artigo comparando as diversas abordagens sobre inspeções de software encontradas na literatura[3], LAITENBERGER e DEBAUD (2000) mostraram que, conforme representado na Figura 2.9, inspeções em código são as mais comuns. No entanto, sabe-se que os benefícios de inspeções são maiores para os artefatos produzidos no início do ciclo de desenvolvimento.

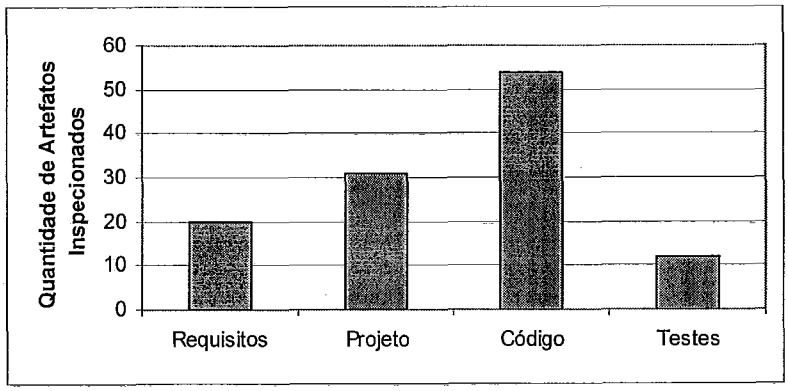


Figura 2.9. Distribuição do Uso de Inspeção nos Diferentes Artefatos. Adaptado de (LAITENBERGER e DEBAUD, 2000).

Um *survey* foi realizado em 2002 visando avaliar o estado da prática de revisões de software (CIOLKOWSKI *et al.*, 2003). De acordo com seus resultados, embora muitas organizações de software realizem revisões, a forma como as revisões são

[3] Os autores analisaram 97 artigos e como alguns destes artigos tratavam de vários artefatos, o total de artefatos inspecionados foi 112.

realizadas ainda é pouco sistematizada, pouco conhecimento da área de inspeções de software é utilizado e assim o verdadeiro potencial das revisões raramente é explorado. Este argumento é baseado em três observações sobre as organizações participantes do *survey*:

- (1) Revisões raramente cobrem sistematicamente as fases de desenvolvimento de software. Cerca de 40% das organizações responderam realizar inspeções em requisitos e 30% inspeções em código.
- (2) Muitas vezes a atividade de detecção de defeitos não é realizada sistematicamente. Cerca de 60% das organizações responderam não conduzir uma atividade de detecção de defeitos regularmente.
- (3) Organizações raramente utilizam revisões de software no contexto de um programa sistemático de avaliação e melhoria do processo. Mais da metade das empresas participantes do *survey* respondeu não coletar dados (40%) ou coletar dados e não os utilizar para realizar uma análise (18%).

A utilização pouco sistematizada de revisões na prática tem sido um fator de confusão entre os resultados esperados e os obtidos através de sua aplicação. A Figura 2.10 ilustra a cobertura de defeitos estimada das revisões realizadas pelas organizações participantes do *survey*.

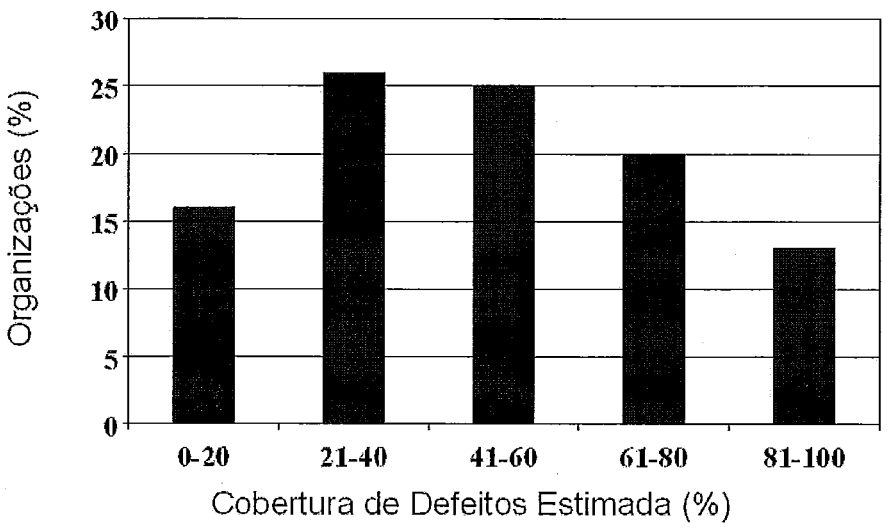


Figura 2.10. Cobertura de defeitos estimada para as organizações participantes do *survey*. Adaptado de (CIOLKOWSKI *et al.*, 2003)

Diversas propostas de apoio ferramental visando apoiar a realização de revisões na prática de forma mais sistemática foram elaboradas. Estas propostas se encontram descritas na sessão seguinte.

2.6.2 – Ferramentas de Apoio ao Processo de Inspeção

Entre as ferramentas mais conhecidas de apoio ao processo de inspeção estão ICICLE (*Intelligent Code Inspection Environment in a C Language Environment*) (BROTHERS *et al.*, 1990), InspeQ (*Inspecting software in phases to ensure Quality*) (KNIGHT e MEYERS, 1991), CSI (*Collaborative Software Inspection*) (MASHAYEKHI *et al.*, 1993), CSRS (*Collaborative Software Review System*) (JOHNSON et al., 1993), Scrutiny (GINTELL *et al.*, 1995), WIT (*Web Inspection Tool*) (TERVONEN *et al.*, 1999), GRIP (*GRoupware supported Inspection Process*) (HALLING et al., 2001) e IBIS (*Internet Based Inspection System*) (LANUBILE e MALLARDO, 2002). Uma descrição destas ferramentas se encontra a seguir:

- **ICICLE.** Visa apoiar o processo tradicional de inspeção de software. Foi desenvolvida para o contexto específico de inspeção de código escrito em C e C++. A ferramenta não se aplica a inspeções mais genéricas, limitando o tipo de artefatos a ser inspecionado e a técnica de detecção de defeitos. Na reunião, a ferramenta exige a presença física de todos os participantes;

- **InspeQ.** Visa apoiar um processo muito parecido com o descrito por HUMPHREY (1989), utilizando a técnica de inspeção por fases (KNIGHT e MEYERS, 1991) para a detecção de defeitos. Assim, *checklists* específicos são utilizados para detectar defeitos e avaliar aspectos de qualidade dos artefatos. Desta forma, os efeitos dos aspectos humanos na detecção de defeitos são reduzidos. A ferramenta pode ser utilizada para inspecionar apenas artefatos descritos textualmente, restringe a técnica a ser utilizada e na reunião exige a presença física de todos os participantes;

- **CSI.** Foi projetada para apoiar o processo de inspeção de HUMPHREY (1989) e ser aplicada a todos os artefatos do processo de software. No entanto, estes artefatos precisam estar descritos textualmente para que possam ser visualizados na ferramenta, que então enumera suas linhas e permite a adição

de anotações a estas linhas. A associação rigorosa de anotações às linhas do artefato não captura alguns tipos de defeito, como as omissões. Estes tipos de defeito precisam então ser descritos separadamente com uma descrição textual de sua localização. A reunião ocorre de forma síncrona, mas não há necessidade da presença física dos participantes, bastando que estes estejam diante de uma estação de trabalho com o CSI;

- **CSRS.** Apóia o processo FTArm, formalizado em (JOHNSON, 1994), onde reuniões assíncronas são empregadas. O artefato, que precisa estar descrito textualmente, é organizado em uma base de dados como um conjunto de nós (*nodes*), onde cada nó representa uma parte do artefato. Os nós estão conectados através de *links* de hipertexto, o que permite que o inspetor percorra o documento. Na detecção de defeitos (referenciada nesta proposta por *revisão particular*) o inspetor lê cada um dos nós existentes no artefato e então é capaz de adicionar novos nós, representando suas anotações. A informação de quais nós foram cobertos e quais ainda precisam ser analisados por cada um dos inspetores é apresentada tanto para o inspetor em questão quanto para o moderador. CSRS possui ainda a facilidade de notificar automaticamente os participantes da inspeção, através de e-mail, quando um novo nó é adicionado. As técnicas de inspeção que podem ser utilizadas em CSRS se limitam ao uso de *checklists* ou da técnica ad-hoc. Na reunião (referenciada nesta proposta por *revisão pública*) todos os nós se tornam públicos e os inspetores podem, de forma assíncrona, votar concordando, discordando ou se mostrando neutros em relação a cada nó;

- **Scrutiny.** Apóia o processo de inspeção de uma organização particular (Bull HB Information Systems), que se assemelha muito ao processo tradicional de inspeção de software, utilizando também reuniões síncronas. A ferramenta visa ser genérica e permitir a inspeção de diferentes tipos de artefatos, no entanto, os artefatos precisam estar descritos textualmente. Na atividade de preparação individual não é fornecido apoio à aplicação de técnicas específicas para a detecção de defeitos. Na reunião, embora síncrona, os participantes não precisam estar presentes fisicamente. A reunião procede com o moderador guiando a leitura do documento pelos demais participantes, enquanto estes lêem

e discutem as anotações feitas. Votações podem ser iniciadas para resolver o status das diferentes anotações;

- **WIT.** HARJUMAA e TERVONEN (1998) apresentam um protótipo para apoiar inspeções assíncronas com equipes geograficamente distribuídas utilizando a *web* denominado WIP (*Web Inspection Prototype*). Posteriormente este protótipo seria evoluído para a ferramenta WIT (TERVONEN *et al.*, 1999). Esta ferramenta leva em consideração as sugestões de JOHNSON (1998) para o futuro das revisões técnicas formais. Além disto, alguns aspectos similares aos que seriam mais tarde formalizados por SAUER et al. (2000), como uma atividade para eliminar duplicatas, são considerados nesta proposta. A ferramenta visa ser genérica para possibilitar a inspeção de diferentes tipos de artefatos, no entanto os artefatos precisam estar representados através de documentos HTML. A técnica para detecção de defeitos desta proposta está limitada a *ad-hoc* ou ao uso de *checklists*. O procedimento de trabalho assíncrono é similar ao da ferramenta CSRS, utilizando revisões particulares e públicas, sendo uma das diferenças a realização da eliminação de duplicatas antes da revisão publica e a separação das atividades de revisão publica e de classificação de defeitos;

- **GRIP.** Nasceu da iniciativa de se adaptar um sistema COTS (*Comercial Off The Shelf*) de apoio a trabalho colaborativo (GSS) (Groupware Support System) para realizar inspeções em requisitos de software (HALLING *et al.*, 2001). Assim, GRIP fornece um *framework* e ferramentas colaborativas para a realização de inspeções assíncronas com equipes geograficamente distribuídas. GRIP fornece suporte a um processo próprio de inspeção que envolve quatro atividades: planejamento, inspeção individual, reunião assíncrona, e avaliação da inspeção. Muitas das mudanças propostas por SAUER *et al.*, 2000 para reduzir o custo e o tempo total para a realização de inspeções assíncronas não são seguidas nesta ferramenta. Não é utilizada, por exemplo, uma fase de coleção de defeitos para eliminar discrepâncias duplicadas e evitar esforço desnecessário na reunião de inspeção. A ferramenta não fornece apoio à técnicas específicas de detecção de defeitos, apenas *ad-hoc.* Uma taxonomia é fornecida junto com o artefato a ser inspecionado. Os elementos desta taxonomia representam, de forma hierárquica, os tópicos abordados no artefato a ser inspecionado. As

discrepâncias dos inspetores são então agrupadas de acordo com os elementos da taxonomia. Assim, uma discrepância na descrição do caso de uso *Mantendo Usuário* poderia ser adicionada utilizando a seguinte navegação pela taxonomia: *Requisitos Funcionais* > *Casos de Uso* > *Mantendo Usuário* > *Descrição*. Os artefatos em si são lidos na ferramenta mais conveniente ao inspetor. Desta forma, GRIP é capaz de lidar com diferentes tipos de artefato, bastando que uma taxonomia para o artefato seja criada descrevendo sua estrutura. Na reunião GRIP fornece apoio para a classificação das discrepâncias que de acordo com um estudo experimental, relatado em (GRÜNBACHER *et al.*, 2003), mostrou reduzir o esforço da reunião e apoiar a classificação correta de discrepâncias (para isto *tresholds* para a classificação automática de defeitos baseado nas opiniões dos participantes da reunião podem ser utilizados);

- **IBIS.** Utiliza a *web* em conjunto com notificações por e-mail para apoiar inspeções assíncronas com equipes geograficamente distribuídas. Uma das diferenças entre estas ferramentas é que IBIS visa explicitamente apoiar a reorganização do processo de inspeção proposta em SAUER *et al.* (2000) e permitir a sua realização de forma sistematizada. IBIS não limita o tipo de artefato a ser inspecionado e na detecção de defeitos atualmente permite apenas a utilização das técnicas *ad-hoc* e de *checklists*. Na reunião de inspeção desta proposta as discrepâncias encontradas na detecção de defeitos são tratadas como tópicos de discussão, onde mensagens podem ser acrescentadas e o moderador pode realizar a classificação das discrepâncias como defeito ou falso positivo. IBIS não fornece apoio aos pontos de tomada de decisão do processo de inspeção, sendo as atividades de planejamento e de continuação do processo tratadas como simples cadastros, sem apoio para a realização de suas tarefas. IBIS tem sido utilizada em estudos experimentais recentes para obter conhecimento na área de inspeções de software e para avaliar aspectos da reorganização do processo de inspeção (LANUBILE e MALLARDO, 2003).

A Tabela 2.1 exibe características básicas das ferramentas descritas acima. As características descritas são: processo de inspeção, técnicas de inspeção que podem ser utilizadas, tipos de artefatos que podem ser inspecionados e se permite equipes de inspeção geograficamente distribuídas.

Ferramenta	Processo	Tipo de Reunião	Técnicas	Artefatos	Equipes Distribuídas
ICICLE	Processo Tradicional (Fagan, 1976)	Síncrona	Técnica Própria	Código C e C++	Não
InspeQ	Processo Próprio	Síncrona	Inspeção por Fases (com checklists específicas)	Artefatos descritos textualmente	Não
CSI	Processo de Humphrey (HUMPHREY, 1989)	Síncrona	Ad-hoc	Artefatos descritos textualmente	Não
CSRS	FTArm (JOHNSON, 1994)	Assíncrona	Ad-hoc e *checklists*	Artefatos descritos textualmente	Sim
Scrutiny	Processo da BULL HB Information Systems, variante do processo tradicional.	Síncrona	Ad-hoc	Artefatos descritos textualmente	Sim
WIT	Processo Próprio	Assíncrona	Ad-hoc e *checklists*	Artefatos descritos em HTML	Sim
GRIP	Processo Próprio	Assíncrona	Ad-hoc	Todos os tipos de artefatos	Sim
IBIS	Sauer et al. (2000)	Assíncrona	Ad-hoc e checklists	Todos os tipos de artefatos	Sim

Tabela 2.1. Características básicas de algumas ferramentas de apoio a inspeções.

MACDONALD *et al.* (1995) analisaram as ferramentas ICICLE, InspeQ, CSI, CSRS e Scrutiny. Quatro anos depois, MACDONALD e MILLER (1999) realizaram uma nova revisão sobre ferramental de apoio a inspeções de software. Desta vez 16 ferramentas foram avaliadas, incluindo todas as ferramentas aqui descritas exceto GRIP e IBIS, que foram desenvolvidas após este trabalho.

Muitas ferramentas de apoio ao processo de inspeção podem ser encontradas na literatura. A análise das ferramentas citadas neste texto e dos artigos sobre revisão de ferramental de apoio para inspeções (MACDONALD et al., 1995) e (MACDONALD e MILLER, 1999) nos permite apontar as seguintes limitações nas ferramentas atualmente existentes:

- (1) Em sua maioria possuem foco exagerado na detecção de defeitos e não no processo de inspeção como um todo;
- (2) Não exploram efetivamente conhecimento da área de inspeções de software para apoiar pontos de tomada de decisão do processo. Acreditamos que o

conhecimento gerado em (ADAMS, 1999) (BIFFL *et al.*, 2003) (CARVER, 2003) e (VITHARANA e RAMAMURTHY, 2003) ainda não tenha sido utilizado em nenhuma abordagem;

- (3) Limitam as técnicas de inspeção a serem utilizadas na detecção de defeitos;
- (4) Não possibilitam integração a outras ferramentas de detecção de defeitos;
- (5) Muitas vezes limitam os tipos de artefatos que podem ser inspecionados;
- (6) Oferecem pouco apoio à atividade de reunião de inspeção;
- (7) As ferramentas que visam apoio ao processo tradicional se tornaram obsoletas por lidarem com reuniões síncronas que mostraram através de estudos experimentais envolver custos desnecessários (veja seção 2.5 para maiores detalhes). Além disto, a avaliação de *groupware* de apoio a atividades síncronas é por sua natureza uma tarefa complexa (PINELLE e GUTWIN, 2000). Em (MANGAN *et al.*, 2002) foram descritas dificuldades encontradas na avaliação de componentes de percepção (*awareness*) para detecção de defeitos síncrona em diagramas de engenharia de software por uma dupla de inspetores.
- (8) Das ferramentas que visam apoio ao processo de inspeção utilizando reuniões assíncronas apenas IBIS utiliza a reorganização proposta em (SAUER et al., 2000);
- (9) Poucos estudos avaliando as propostas foram realizados, o que torna difícil verificar se, e quanto, as ferramentas realmente melhoram o desempenho de inspeções de software.

2.7 – Conclusão

O objetivo de inspeções de software é melhorar a qualidade de artefatos de software através de sua análise, detectando e removendo defeitos antes que o artefato seja passado para a próxima fase do processo de desenvolvimento de software. Conforme visto na seção 2.3, a aplicação de inspeções entre as atividades do ciclo de vida de software pode trazer diversos benefícios para organizações de software.

O processo de inspeção foi inicialmente elaborado por FAGAN (1976). Argumentos da literatura e estudos experimentais, visando avaliar este processo, vêm indicando reuniões assíncronas para inspeções de software. Tendo em vista as inspeções com reuniões assíncronas SAUER *et al.*, (2000), baseados em estudos

experimentais, apresentam uma reorganização com mudanças para reduzir o custo e o tempo total da realização deste tipo particular de inspeção.

Atualmente muitas organizações realizam revisões, mas que a forma como as revisões são realizadas ainda é pouco sistematizada e pouco conhecimento da área de inspeções de software é utilizado. Assim o verdadeiro potencial das revisões raramente é explorado, introduzindo um fator de confusão entre os resultados esperados pelas revisões e os obtidos na prática.

Muitas ferramentas de apoio ao processo de inspeção surgiram visando apoiar a realização de inspeções de forma mais sistematizada em organizações. A análise de 16 destas ferramentas nos mostrou algumas limitações das abordagens atuais, sendo uma delas o pouco uso de conhecimento existente e efetivamente validade sobre inspeções de software para apoiar os pontos de tomada de decisão do processo. O conhecimento descrito em (ADAMS, 1999) (BIFFL et al., 2003) (CARVER, 2003) e (VITHARANA e RAMAMURTHY, 2003), por exemplo, não é utilizado em nenhuma das ferramentas analisadas.

Analisando o conhecimento da literatura sobre inspeções de software e as ferramentas existentes, identificou-se ser possível elaborar uma proposta de infra-estrutura genérica de apoio ao processo de inspeção de software, que pudesse ser utilizada para inspecionar todos os tipos de artefatos produzidos ao longo do ciclo de vida de software, utilizando equipes geograficamente distribuídas e reuniões assíncronas. Esta proposta automatiza a reorganização do processo de inspeção de SAUER *et al.* (2000), possibilitando a integração a outras ferramentas de detecção de defeitos e explorando informações experimentalmente validadas para fornecer apoio aos pontos de tomada de decisão do processo. A proposta em questão está descrita no Capítulo 3.

Capítulo 3
Proposta de Infra-Estrutura para Apoio ao Processo de Inspeção

Neste capítulo, apresentamos uma proposta para apoio ao processo de inspeção de software elaborada utilizando como conjunto básico de requisitos conhecimento adquirido através de estudos experimentais relacionados a inspeções de software.

3.1 – Introdução

Analisando as características do processo de inspeção de software e as propostas de apoio atualmente existentes, identificamos ser possível elaborar uma proposta que explorasse mais o conhecimento existente na literatura, principalmente para apoiar os pontos de tomada de decisão do processo.

Este capítulo descreve uma nova proposta de apoio ao processo de inspeção de software, considerando as mudanças para reduzir o custo e o tempo total para a realização de inspeções assíncronas com equipes geograficamente distribuídas da reorganização do processo de inspeção de software de SAUER et al. (2000). Uma solução arquitetural foi elaborada para que a proposta pudesse contemplar apoio adequado para inspecionar diferentes artefatos de software.

Os requisitos de apoio a cada uma das atividades do processo de inspeção de software foram adquiridos através de conhecimento teórico sobre inspeções de software que tem se mostrado adequado em resultados de estudos experimentais obtidos da literatura (BIFFL e GUTJAHR, 2002) (BIFFL *et al.*, 2003) (CARVER, 2003) (LANUBILE e MALLARDO, 2003) (VITHARANA e RAMAMURTHY, 2003). Este conhecimento é explicitado ao longo deste capítulo, onde o apoio fornecido pela proposta é descrito de forma detalhada. A proposta considera ainda requisitos para possibilitar e facilitar a gerência do processo de inspeção.

Este capítulo está organizado da seguinte forma: na seção 3.2, a solução arquitetural da proposta é apresentada; na seção 3.3, o apoio a cada uma das

atividades do processo de inspeção é descrito; a seção 3.4, apresenta o apoio para a gerência do processo de inspeção; por fim, a seção 3.5, concluí o capítulo.

3.2 – Proposta de Solução Arquitetural

O processo de inspeção proposto em (SAUER et al., 2000) é genérico e pode ser instanciado para a inspeção de diferentes tipos de artefatos. Considerando isto, um dos requisitos básicos para a infra-estrutura de apoio é não limitar os tipos de artefatos que podem ser inspecionados e possibilitar o uso de técnicas de leitura específicas para determinados artefatos.

Assim, uma arquitetura que possibilita o uso de ferramentas externas, para apoiar a aplicação de técnicas de leitura específicas para determinado tipo de artefato, foi projetada. Um integrador de ferramentas é então utilizado para possibilitar o intercâmbio de dados entre as ferramentas externas e a infra-estrutura. A Figura 3.1 ilustra a arquitetura proposta.

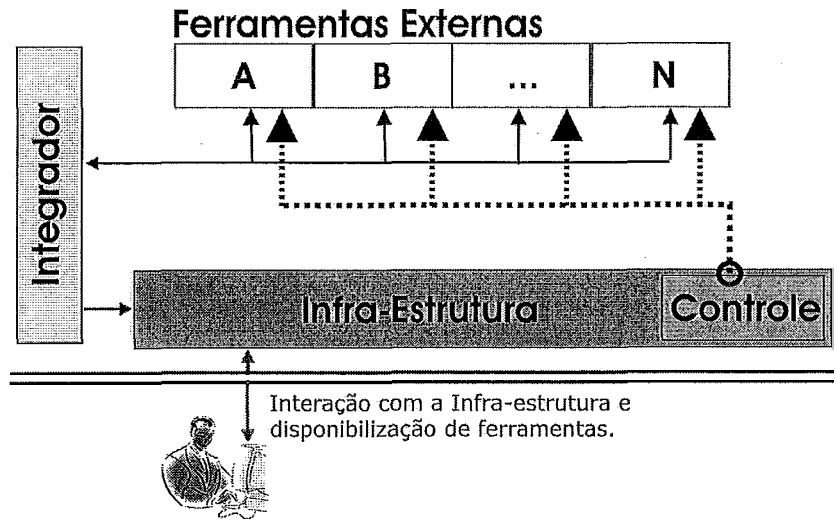


Figura 3.1. Arquitetura Proposta para a Infra-Estrutura (KALINOWSKI *et al.*, 2004).

Nesta arquitetura, a infra-estrutura em si é responsável pela interação com o usuário pela *web* e por garantir que o processo de inspeção seja seguido de forma sistemática. Seu módulo de controle tem conhecimento das ferramentas externas existentes e disponibiliza as apropriadas para o usuário, de acordo com as técnicas que

devem ser utilizadas na inspeção. Os dados produzidos através da utilização das ferramentas externas são então retornados para a infra-estrutura utilizando o módulo integrador.

Uma das características desta solução é o baixo acoplamento, permitindo que ferramentas independentes sejam utilizadas para complementar as funcionalidades da infra-estrutura, possibilitando o apoio customizado e extensível a técnicas de leitura específicas.

3.3 – Apoio às Atividades do Processo de Inspeção

Esta seção apresenta o apoio da proposta de infra-estrutura a cada uma das atividades do processo de inspeção de SAUER *et al.*, (2000).

3.3.1 – Planejamento

Na atividade de planejamento da inspeção o moderador da inspeção é responsável por (1) definir o contexto da inspeção, (2) selecionar os inspetores e (3) distribuir o material a ser utilizado durante a inspeção.

A definição do contexto descreve os objetivos da inspeção e deve conter os seguintes dados: moderador, data da solicitação, prazo para a detecção de defeitos, descrição da inspeção, técnica a ser utilizada na detecção de defeitos, ênfase nos tipos de defeito a ser encontrados na inspeção, autor do artefato, lingua do artefato, tipo de artefato e o domínio da aplicação do artefato.

A seleção dos inspetores, por sua vez, envolve uma tomada de decisão que pode afetar a eficiência do processo de inspeção como um todo. Considerando isto, algumas propostas da literatura, como o gerenciamento ativo de inspeções (AIM) (HALLING *et al.*, 2002), sugerem que esta decisão seja tomada com base na avaliação de dados históricos sobre inspeções passadas. Assim, a proposta considera os dados históricos relativos ao desempenho dos inspetores em inspeções passadas sobre o mesmo tipo de artefato. Além disto, a caracterização dos inspetores é considerada para apoiar a escolha de inspetores com um perfil adequado para inspeção em questão.

Em relação aos dados históricos, o moderador deve ter acesso às seguintes informações sobre a última inspeção e a média de todas as inspeções, sobre o mesmo

artefato, de cada um dos inspetores: número de defeitos encontrados, número de falsos positivos relatados, tempo dedicado à detecção de defeitos e custo-eficiência (número de defeitos por hora). Além disto, a distribuição dos tipos de defeitos encontrados na inspeção por cada inspetor deve poder ser consultada.

A caracterização, por sua vez, deve ser utilizada através da aplicação de um resultado de CARVER (2003), onde o impacto das diferentes caracterizações de inspeções e inspetores no desempenho destes na detecção de defeitos foi avaliado. Isto foi realizado através da elaboração e utilização de uma metodologia para obter hipóteses fundamentadas[4] (*grounded hypotheses*) através da exploração de dados históricos de estudos experimentais sobre inspeções. Foram analisados dois estudos conduzidos na NASA em 1994 e 1995 (no *Goddard Space Flight Center* em Greenbelt, Maryland) (BASILI *et al.*, 1996), três conduzidos na universidade de Maryland em 97, 98 e 99 (SHULL, 1998) (SHULL *et al.*, 1999) (SHULL et al., 2003), um estudo conduzido na universidade de Southern Carlifornia pelo Dr. Barry Boehm em 2000 e 2001 e quatro repetições dos estudos da NASA realizados na Universidade de São Paulo em 2001 (SHULL *et al.*, 2003).

Utilizando estas hipóteses fundamentadas é possível, dado o contexto de uma inspeção, conhecer o valor desejado para algumas características dos inspetores. Em nossa proposta consideramos que um inspetor possui uma característica desejada quando o valor desta característica do inspetor for próximo ao valor desejado desta característica para a inspeção. Assim, a infra-estrutura deve fornecer uma lista ordenada dos inspetores mais indicados para a detecção de defeitos, utilizando como critério de ordenação o número de características desejadas para a inspeção que o inspetor possui.

É importante ressaltar que a lista ordenada de inspetores deve ter como intuito apenas auxiliar a decisão do moderador sobre quais inspetores escolher e não automatizar esta decisão. A decisão final deve continuar sob responsabilidade do moderador, que pode consultar os dados sobre a caracterização e o desempenho de cada um dos inspetores da lista e escolher os que ele considerar mais convenientes. Vale lembrar ainda que variáveis de um contexto mais amplo podem afetar esta

[4] Uma hipótese fundamentada é uma hipótese elaborada que se confirmou em resultados de estudos experimentais.

decisão, como a própria disponibilidade dos recursos humanos dentro da organização em questão.

Por fim, a distribuição do material depende diretamente do contexto da inspeção e não deve apresentar limitações quanto ao tipo de material que pode ser carregado na infra-estrutura.

3.3.2 – Detecção de Defeitos

De acordo com a solução arquitetural apresentada na seção 3.1, a infra-estrutura deverá disponibilizar ferramentas externas para apoiar a aplicação de técnicas de leitura de artefatos específicos na atividade de detecção de defeitos. A disponibilização da ferramenta adequada para o contexto da inspeção definido na atividade de planejamento deverá ser feita por um módulo de controle da infra-estrutura. Para cada discrepância as seguintes informações devem poder ser cadastradas: localização, tipo de defeito (de acordo com os tipos de defeito descritos na seção 2.2), severidade e descrição.

Além disto, para viabilizar a inspeção de todos os tipos de artefatos, a infra-estrutura, em sua configuração básica, deve possibilitar o uso da técnica *ad-hoc.* Assim, discrepâncias devem poder ser cadastradas para o artefato sem a necessidade de se utilizar uma ferramenta externa.

3.3.3 – Coleção de Defeitos

Na coleção de defeitos a infra-estrutura deve utilizar os resultados do estudo experimental apresentado em (LANUBILE e MALLARDO, 2003), que mostram que a discriminação deve ser restrita a discrepâncias encontradas por apenas um inspetor.

Neste experimento, nove inspeções assíncronas utilizando a ferramenta IBIS foram realizadas sobre documentos de requisitos distintos. Cada uma das inspeções fez uso de equipes geograficamente distribuídas com tamanho variando entre quatro e seis participantes (incluindo o autor do documento e o moderador). Em todas as inspeções tanto as discrepâncias encontradas por um só inspetor quanto as encontradas por mais de um inspetor passaram pela atividade de discriminação de defeitos. Os resultados do experimento foram os seguintes: (1) discrepâncias encontradas por um só inspetor têm maior chance de representar falso positivos do que discrepâncias encontradas por mais

de um inspetor, (2) houve um número maior de votos individuais do tipo 'falso positivo' para discrepâncias encontradas por um único inspetor e (3) o número de mensagens na discussão não teve diferença significativa entre as discrepâncias encontradas por um único inspetor e as encontradas por mais de um inspetor. Assim, embora tivessem pouca probabilidade de representar falso positivos, as discrepâncias encontradas por mais de um inspetor exigiram o mesmo esforço durante a discriminação de defeitos. Estes resultados são condizentes com a hipótese do experimento de que a discussão de discrepâncias encontradas por mais de um inspetor não vale a pena, uma vez que esforço desnecessário durante a discriminação poderia ser evitado classificando-as diretamente como defeito.

Desta forma, na coleção de defeitos, a infra-estrutura deve possibilitar que o moderador identifique discrepâncias encontradas por mais de um inspetor. Uma vez que uma discrepância repetida é encontrada pelo moderador, ela deve ser classificada pela infra-estrutura como defeito e encaminhada diretamente para a atividade de retrabalho do autor do documento.

3.3.4 – Discriminação de Defeitos

Nesta atividade, o processo de inspeção de (SAUER *et al.*, 2000) envolve uma discussão assíncrona das discrepâncias. A infra-estrutura deve possibilitar esta discussão para participantes geograficamente distribuídos.

Além disto, resultados de um estudo experimental (VITHARANA e RAMAMURTHY, 2003) revelando uma característica interessante da discriminação de defeitos devem ser utilizados. De acordo com estes resultados, o anonimato dos participantes pode ajudar na classificação correta das discrepâncias. Neste experimento, 159 alunos de graduação foram utilizados formando 53 equipes de inspeção compostas por três participantes cada. Estas equipes foram divididas em dois tratamentos distintos, tendo e não tendo conhecimento dos demais participantes da equipe. A tarefa consistia em identificar defeitos em dois artefatos do tipo código fonte, descritos na linguagem de programação C, de complexidades distintas. A atividade de detecção de defeitos foi realizada de forma isolada pelos participantes de cada equipe. A discriminação dos defeitos, por sua vez, foi apoiada por um sistema de *groupware* (chamado *visionQuest*), que foi instanciado para que os participantes soubessem quem eram os demais participantes e não o soubessem, atendendo respectivamente aos dois

tratamentos. Os resultados do estudo mostraram que: (1) os grupos que fizeram uso de anônimidade foram mais eficientes por conseguirem classificar um maior número de discrepâncias corretamente e (2) os participantes dos grupos que fizeram uso de anônimidade desenvolveram atitudes mais favoráveis a inspeções de software, mostrando mais participação na realização de suas tarefas. A Figura 3.2 mostra o desempenho das equipes com participantes anônimos e não anônimos em encontrar defeitos nos artefatos utilizados no estudo.

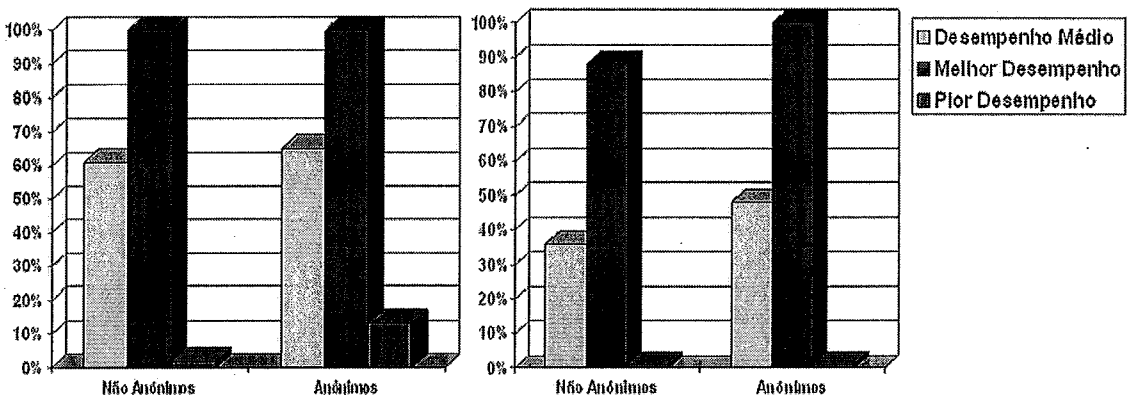


Figura 3.2. Desempenho das equipes em encontrar defeitos em cada um dos artefatos inspecionados. O gráfico à esquerda é referente ao artefato de menor complexidade. Adaptado de (VITHARANA e RAMAMURTHY, 2003).

Desta forma, durante a reunião a infra-estrutura deve manter os nomes dos participantes da atividade de discriminação em sigilo, revelando apenas o papel que o participante desempenha na inspeção.

3.3.5 – Retrabalho

A correção nos artefatos inspecionados propriamente dita está fora do escopo do processo de inspeção de software. No entanto, nesta atividade, a infra-estrutura deve permitir a elaboração de um relatório de correção dos defeitos, para que possa ser utilizado como informação para a avaliação da qualidade do artefato na atividade de continuação do processo de inspeção.

Assim, para cada defeito um comentário de correção deve poder ser adicionado pelo autor do artefato. Além disto, o artefato corrigido deve poder ser anexado na infra-

estrutura, para que tanto o artefato com os defeitos quanto o corrigidos possam ser visualizados posteriormente pelo moderador.

3.3.6 – Continuação

A continuação envolve uma importante avaliação da qualidade do artefato pelo moderador. Ele precisa decidir se o artefato possui qualidade suficiente, em termos de densidade de defeitos, para ser passado para as próximas atividades do ciclo de vida de software ou se uma nova inspeção precisa ser realizada no artefato. Esta decisão envolve ainda estimar se os defeitos encontrados em uma nova inspeção são capazes de aumentar a cobertura de defeitos[5] do artefato.

A infra-estrutura deve apoiar esta tomada de decisão utilizando os seguintes dados: (1) estimativas da cobertura de defeitos para a inspeção atual e para a reinspeção, e (2) dados históricos do número médio de defeitos encontrados em inspeções passadas sobre o mesmo tipo de artefato.

Para calcular a cobertura de defeitos atual de um documento é preciso conhecer o número total de defeitos presentes neste, o que na prática não ocorre. BIFFL *et al.*, (2003) apresentam duas formas de estimar o número de defeitos de documentos. O *Weighed Average of Individual Estimates* (WAE) e o *Weighed Average Of Individual Offsets* (WAO). Em ambas estimativas subjetivas individuais do número de defeitos são utilizadas para se obter uma estimativa subjetiva da equipe de inspetores como um todo. Um estudo experimental utilizando 177 participantes divididos em 30 equipes solicitadas para realizar a estimativa de defeitos em um documento de requisitos com um número de defeitos conhecido está descrito neste mesmo artigo. Nos resultados do estudo experimental a estimativa subjetiva de equipe utilizando o WAO foi mais precisa (tanto para inspeções utilizando *checklists* quanto para inspeções utilizando leitura baseada em perspectiva) do que a estimativa utilizando WAE, do que a estimativa do modelo objetivo de captura e recaptura *Jack Knife*[6] e do que os modelos de estimativas

[5] A cobertura de defeitos é o número de defeitos encontrados no documento dividido pelo número de defeitos presentes no documento.

[6] O modelo *Jack Knife* é um modelo que, dentre os objetivos, tem mostrado melhores resultados em estudos experimentais (BIFFL e GROSSMANN, 2001).

subjetivas individuais. A Figura 3.3 ilustra a comparação dos resultados do experimento para os modelos utilizando o WAO, utilizando o WAE e o modelo *Jack Knife*.

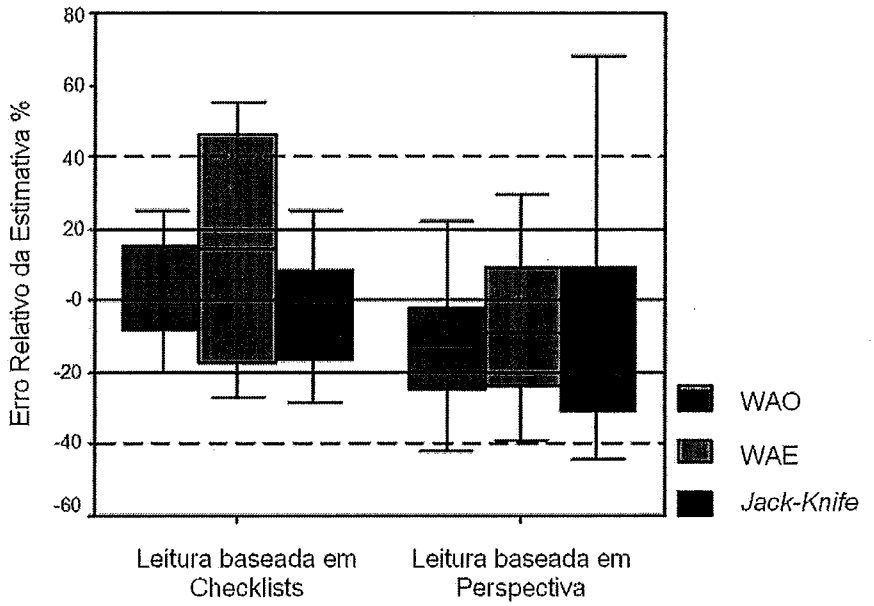


Figura 3.3. Comparação dos resultados do experimento para os modelos utilizando o WAO, utilizando o WAE e o modelo Jack-Knife. Adaptado de (BIFFL et al., 2003).

A estimativa utilizando o WAO pode ser calculada de acordo com a fórmula da Figura 3.4. O número de defeitos do documento nesta técnica é estimado como sendo a soma do número de defeitos encontrados na inspeção anterior com a média ponderada das diferenças entre o número de defeitos estimado por cada um dos inspetores e o número de defeitos que ele efetivamente encontrou na inspeção. Para este modelo, a estimativa individual de cada um dos inspetores é obtida logo após a conclusão da atividade de detecção de defeitos, quando o inspetor conhece apenas os defeitos que ele mesmo encontrou no documento.

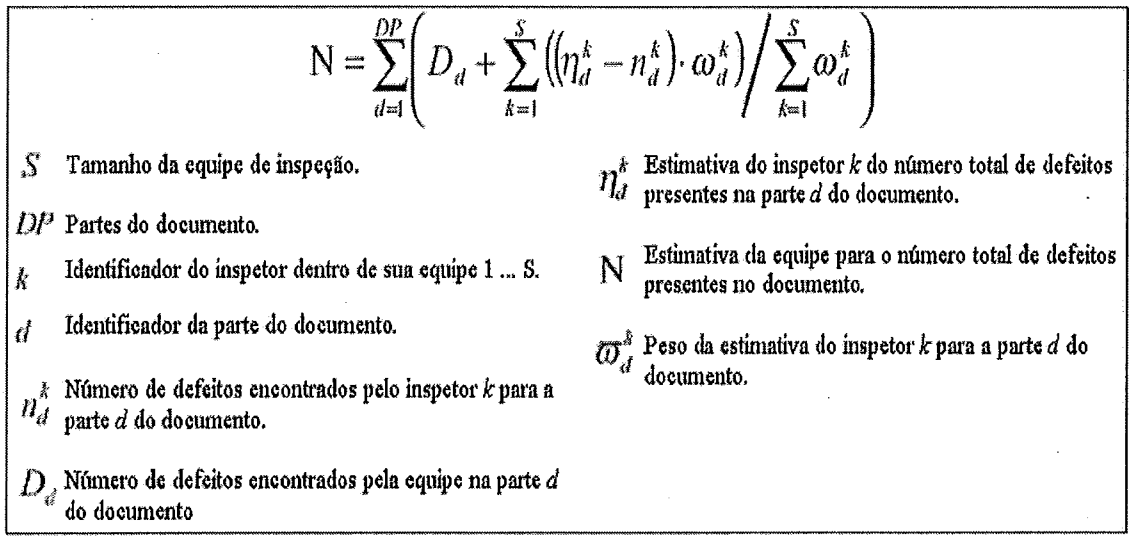


Figura 3.4. Estimativa do número de defeitos de documentos utilizando o WAO. Adaptado de (Biffl *et al.*, 2003).

Para calcular a cobertura de defeitos da próxima inspeção, o modelo ILM (*Improved Linear Model*) (ADAMS, 1999) pode ser utilizado. Em (BIFFL e GUTJAHR, 2002) um modelo de aumento de confiabilidade (*reliability growth model*) (RGM) para esta estimativa foi apresentado e um estudo experimental foi conduzido para avaliar técnicas de estimativa da cobertura de defeitos. O experimento utilizou 159 participantes que realizaram duas inspeções consecutivas em um documento de requisitos, divididos em 29 equipes de inspeção. Os dados dos defeitos encontrados na primeira inspeção foram utilizados para que as estimativas da performance da equipe na segunda inspeção pudessem ser feitas. Neste experimento a estimativa do modelo ILM foi mais precisa do que a dos modelos triviais[7], do modelo linear original (OLM) (ADAMS, 1999) e do que o modelo RGM proposto no artigo. A tabela 3.1 apresenta o erro relativo[8] das estimativas dos modelos OLM, ILM e RGM neste experimento.

O modelo ILM faz sua estimativa de acordo com a fórmula da Figura 3.5. Neste modelo, em uma nova inspeção, a cobertura da inspeção é acrescida da cobertura

[7] Sempre re-inspecionar e nunca re-inspecionar.

[8] O erro relativo é a diferença entre a cobertura de defeitos estimada e a cobertura real obtida após a segunda inspeção.

anterior, normalizada pelo esforço envolvido em cada uma das inspeções, vezes o percentual de defeitos restantes no documento.

	OLM		ILM		RGM	
Equipes com Eficiência	Média %	Desvio Padrão %	Média %	Desvio Padrão %	Média %	Desvio Padrão %
Todas	20,4	13,1	-0,1	6,1	-2,9	5,9
Cobertura da primeira inspeção < 50%	9,8	10,1	-1,3	8,0	-1,0	7,5
Cobertura da primeira inspeção entre 50% e 70%	25,2	11,6	0,5	5,2	-3,8	5,1

Tabela 3.1. Erro relativo dos modelos de estimativa OLM, ILM e RGM. Adaptado de (BIFFL e GUTJAHR, 2002).

$$CE[I+1] = CR[I] + ((CR[I]/E[I]) \cdot E[I+1]) \cdot (1 - CR)$$

Variáveis $(0 < CE[I], CR[I], CR < 1)$

I	Número de vezes que o documento foi inspecionado.
CE[I]	Cobertura Estimada para a inspeção de número I.
CR[I]	Cobertura Real obtida na inspeção de número I.
CR	Cobertura Real total considerando todas as inspeções já realizadas neste documento.
E[I]	Esforço realizado na inspeção de número I.

Figura 3.5. O modelo ILM. Adaptado de (ADAMS, 1999).

As estimativas da cobertura de defeitos atual e após a próxima inspeção devem ser utilizadas da seguinte forma: se a cobertura de defeitos da inspeção atual for baixa (abaixo de 70%) e da reinspeção for alta (maior do que 70%), a infra-estrutura deve sugerir uma nova inspeção, visando uma boa cobertura de defeitos para a inspeção.

Por fim, em relação ao dado histórico do número médio de defeitos em inspeções passadas sobre o mesmo tipo de artefatos, uma diretriz, sugerida por FAGAN (1976), pode ser aplicada. Seguindo esta diretriz, a infra-estrutura deve sugerir uma nova inspeção se o documento apresentou 5% a mais de defeitos do que o número médio de defeitos encontrados em inspeções neste tipo de artefato. Esta sugestão visa a qualidade do produto, evitando que artefatos inicialmente muito defeituosos sejam passados adiante no processo de desenvolvimento de software.

3.4 – Apoio à Gerência do Processo de Inspeção

Seguindo a proposta do gerenciamento ativo de inspeções (AIM) (HALLING et al., 2002) a infra-estrutura deve: (1) auxiliar as decisões tomadas no planejamento com base na avaliação de inspeções passadas (veja seção 3.3.1), (2) permitir o monitoramento das inspeções correntes, (3) permitir avaliações pós-processo visando entender a relação entre processo de inspeção, inspetores e artefato inspecionado e assim (4) permitir a melhoria do processo.

O monitoramento das inspeções correntes deve permitir que se saiba quando uma determinada atividade foi concluída e por quem. Além de permitir o monitoramento das inspeções correntes, a infra-estrutura deve possibilitar o monitoramento das inspeções passadas, para que o moderador possa realizar comparações do tempo gasto na realização das tarefas nas diferentes inspeções. Poder monitorar as diferentes inspeções facilita a tarefa do moderador de coordenação do processo.

Para realizar a avaliação pós-processo, o moderador deverá ter acesso a dados históricos detalhando o desempenho dos inspetores em inspeções passadas. Os dados fornecidos para a avaliação pós-processo estão explicitados na seção 3.3.1, e em nossa proposta este apoio está contemplado na atividade de planejamento.

Em relação à melhoria do processo, as informações coletadas na realização das inspeções podem ser utilizadas para questionar a estrutura do processo de inspeção (ou mesmo do processo de desenvolvimento de software) ou o apoio fornecido pela infra-estrutura.

Além de considerar a proposta de gerenciamento ativo de inspeções, uma outra facilidade para a coordenação do processo de inspeção deve ser fornecida: a notificação por e-mail para avisar aos participantes da inspeção de acessarem a infra-estrutura sempre que uma atividade precisa ser realizada.

3.5 – Conclusão

Neste capítulo apresentamos uma proposta de apoio para a reorganização do processo de inspeção de software proposta por SAUER et al. (2000), visando permitir a realização sistemática de inspeções assíncronas com equipes geograficamente distribuídas.

Uma solução arquitetural para possibilitar a realização de inspeções em diferentes tipos de artefatos de software, permitindo a aplicação de técnicas de leitura específicas na atividade de detecção de defeitos, foi descrita.

Conhecimento da área de inspeções de software, obtido através de estudos experimentais da literatura, foi utilizado para se obter um conjunto de requisitos para o apoio a cada uma das atividades do processo de inspeção. Além disto, requisitos para o apoio à gerência do processo de inspeção foram identificados.

O capítulo seguinte descreve a infra-estrutura implementada baseada nesta proposta de apoio e seu funcionamento detalhado.

Capítulo 4

Infra-estrutura Computacional para Apoio ao Processo de Inspeção de Software

Neste capítulo a infra-estrutura implementada com base na proposta de apoio do Capítulo 3 é apresentada. Decisões de projeto, a arquitetura da infra-estrutura e seu funcionamento detalhado são descritos.

4.1 – Introdução

Neste capítulo é descrita a solução computacional que implementa a infra-estrutura de apoio ao processo de inspeção de software, contemplando o conjunto de requisitos apresentado no capítulo 3. Esta infra-estrutura, denominada ISPIS (KALINOWSKI *et al.*, 2004), visa possibilitar a realização sistematizada de inspeções assíncronas, sobre todos os tipos de artefatos que podem ser construídos durante o processo de desenvolvimento de software, com equipes geograficamente distribuídas.

Uma análise cuidadosa da proposta e de seu conjunto de requisitos resultou na tomada de algumas decisões a respeito do projeto da solução computacional. Estas decisões de projeto estão descritas na seção 4.2.

Em seguida, na seção 4.3 descrevemos como foi possível instanciar a proposta de solução arquitetural da seção 3.2, que considera ferramentas externas para possibilitar o uso de técnicas de leitura específicas para determinados artefatos, através da implementação de projetos da COPPE/UFRJ.

Uma visão detalhada das funcionalidades de ISPIS é fornecida na seção 4.4 e na seção 4.5 o capítulo é concluído.

4.2 – Decisões de Projeto

Nesta seção são relatadas as decisões de projeto tomadas para a solução computacional de ISPIS para o conjunto de requisitos identificado na proposta de apoio ao processo de inspeção de software do Capítulo 3.

4.2.1 – Utilização de uma Ferramenta de Workflow

Baseada na estrutura rígida do processo de inspeção de software e em seus aspectos colaborativos, a utilização de uma ferramenta de *workflow* foi cogitada para sua automatização. Segundo CHAFFEY (1998), tais ferramentas podem trazer diversos benefícios para garantir o controle, o acompanhamento e a auditoria de processos. Utilizar uma ferramenta de *workflow* para automatizar o processo de inspeção de software torna possível usufruir destes benefícios para a realização de inspeções.

No entanto, a definição do processo de inspeção muda dinamicamente, de acordo com as decisões que vão sendo tomadas no seu decorrer (como, por exemplo, quando a equipe de inspeção é selecionada na atividade de planejamento). Assim, em determinados momentos ISPIS precisa interagir com a ferramenta de *workflow* para redefinir a estrutura do processo. Por este motivo a seguinte decisão de projeto foi tomada: ISPIS, ao invés de utilizar uma ferramenta de *workflow*, foi desenvolvida como sendo a extensão de uma destas ferramentas.

Muitas ferramentas de *workflow* existem atualmente no mercado, informações e estudos comparativos sobre algumas destas ferramentas podem ser encontrados em (WARIA, 2004). A ferramenta de *workflow* escolhida para ser estendida em ISPIS foi a PatternFlow (KALINOWSKI, 2001). Um motivo para a adoção desta ferramenta foi o fato dela ter sido desenvolvida na UFRJ e possuir projeto e código abertos e conhecidos. A ferramenta adere a grande parte de um conjunto de padrões para ferramentas de *workflow* conhecido como *workflow patterns* (VAN DER AALST *et al.*, 2000) e com isto se mostrou adequada aos requisitos de coordenação do processo de inspeção de software.

Assim, ISPIS está implementada em dois módulos, conforme ilustra a Figura 4.1. Um destes módulos é o de funcionalidades específicas para apoiar inspeções de software. Este módulo trata das particularidades do processo de inspeção e de suas

atividades. Para isto faz uso tanto de dados específicos para apoiar inspeções de software[9] quanto dos metadados do PatternFlow[10], onde interage com a definição do processo.

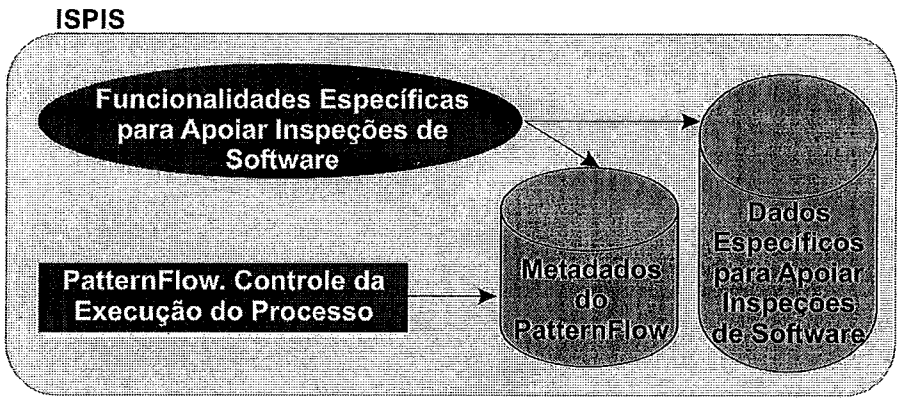


Figura 4.1. Módulos de ISPIS.

O outro módulo é a ferramenta PatternFlow. Este módulo é responsável por controlar a execução do processo seguindo as informações descritas em seus metadados.

4.2.2 – Plataforma de Desenvolvimento

Um dos requisitos para a infra-estrutura é possibilitar a realização de inspeções por equipes geograficamente distribuídas. As plataformas de desenvolvimento para a *web*, utilizando como cliente os *browsers*, atendem a este requisito e por este motivo foi tomada a decisão de projeto de utilizar uma plataforma de desenvolvimento para a *web*.

Existem diversas plataformas de desenvolvimento para a web. No entanto, a ferramenta estendida, PatternFlow, utiliza a plataforma J2EE (*Java 2 Enterprise Edition*). Assim, para facilitar a criação de ISPIS como extensão da PatternFlow, ISPIS

[9] Entre estes dados específicos se encontram dados como a caracterização dos inspetores, a caracterização das inspeções que estão sendo realizadas, dados históricos sobre inspeções passadas, entre outros.

[10] Os metadados do PatternFlow contêm as informações relativas à definição e à encenação de processos.

utiliza esta mesma plataforma. Tendo em vista os propósitos da infra-estrutura esta plataforma se mostrou adequada para sua implementação.

Visando o ambiente web, ISPIS, assim como a PatternFlow, foi implementada seguindo o padrão arquitetural de aplicações para a *web Thin Web Client* (CONALLEN, 1999), que visa manter pouco processamento no cliente (que numa aplicação *web* é o browser). A visão lógica do padrão arquitetural *Thin Web Client* é exibida na Figura 5.2.

Em ISPIS, seguindo este padrão, páginas servidoras (*Server Pages*) (implementadas através de *Java Servlets* e *Java Server Pages*) residem em um servidor de aplicação (*Web Server*). O servidor de aplicação escolhido foi o *Jakarta Tomcat,* no entanto não foram utilizados recursos específicos deste servidor e a troca de servidor pode ocorrer sem maiores impactos. Estas páginas servidoras se comunicam com classes de domínio (*Business Objects*) (classes implementadas em *Java*) que então realizam as funcionalidades principais do sistema e garantem a persistência dos dados em um esquema relacional. O banco de dados escolhido foi o *MySQL*, no entanto a solução não está limitada a este banco e outros podem ser utilizados.

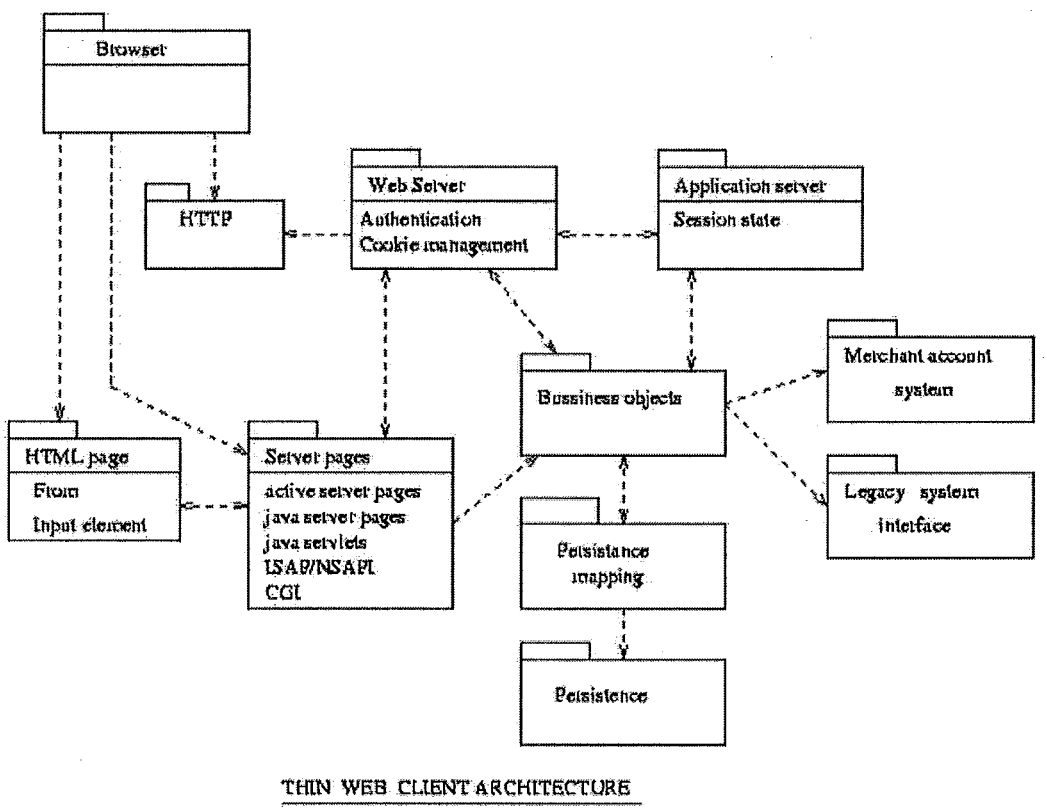


Figura 4.2. Visão lógica do padrão arquitetural *Thin Web Client* (CONALLEN, 1999).

Assim, utilizando a mesma plataforma e padrão de desenvolvimento, ISPIS é capaz de estender as funcionalidades da PatternFlow através da simples importação dos pacotes que contém as classes do domínio da PatternFlow, que então podem ser utilizadas diretamente sempre que preciso.

Na Figura 4.3 isto é ilustrado, mostrando a classe *Inspector* de ISPIS importando o pacote *patternFlow.entities.User* para assim poder estender as funcioncionalidades da classe *User* da PatternFlow. O diagrama das classes de domínio da PatternFlow, descrito em UML (BOOCH *et al.*, 1998) se encontra no Apêndice A.1. O diagrama das classes de domínio de ISPIS, por sua vez, se encontra no Apêndice A.2. Um modelo entidade-relacionamento, descrevendo o esquema relacional utilizado na persistência dos dados se encontra no Apêndice B.

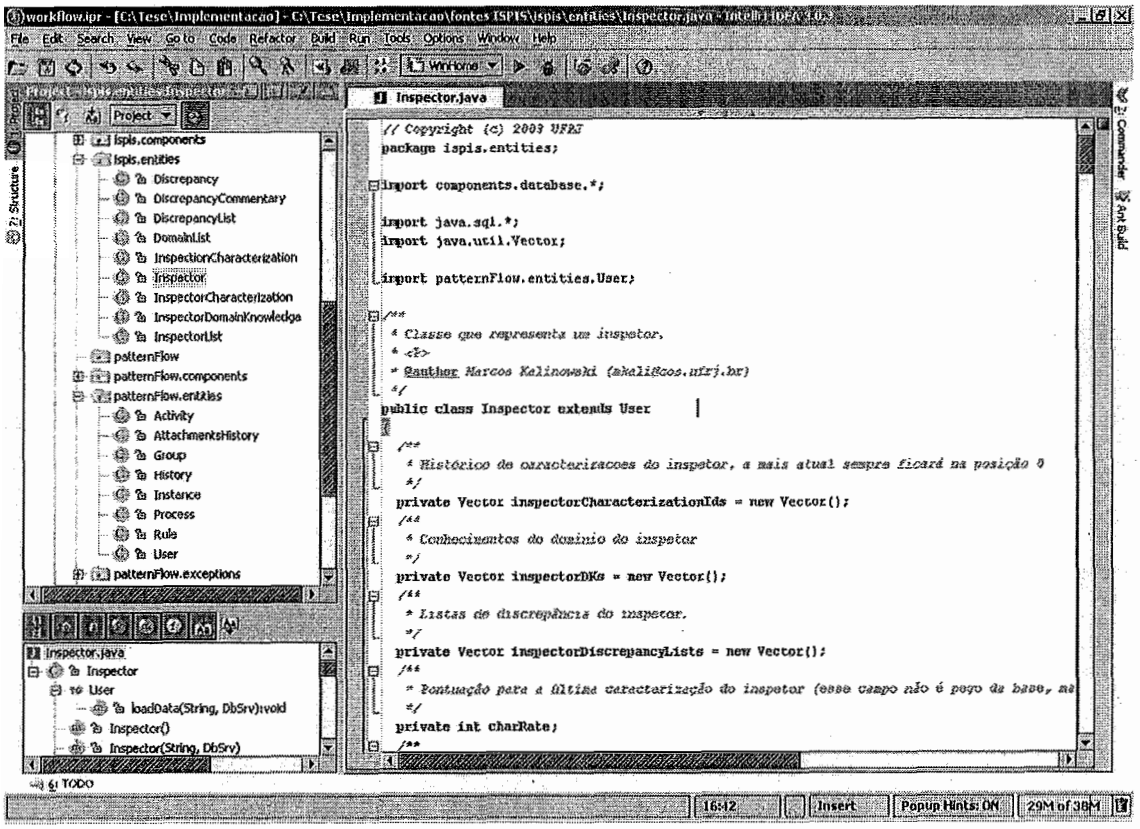


Figura 4.3. Classe *Inspector* de ISPIS estendendo a classe *User* da PatternFlow.

Todos os softwares utilizados para a implementação de ISPIS são gratuitos e independentes do sistema operacional escolhido para a máquina servidora da aplicação.

4.2.3 – Internacionalização

Ainda tendo em vista o requisito de possibilitar a realização de inspeções por equipes geograficamente distribuídas, é preciso considerar o fato da equipe de inspeção poder envolver participantes de diferentes idiomas nativos e acostumados a lidar com ferramentas em diferentes idiomas.

Por este motivo, uma decisão de projeto foi implementar ISPIS tendo um núcleo de programa cujos recursos de projeto e código não presumem um único idioma ou localidade. Assim, ISPIS utiliza palavras-chave para a saída de dados. Estas palavras chave são então convertidas para frases no idioma nativo do usuário que está utilizando o sistema. Isto é feito através da utilização de arquivos de tradução para os diferentes idiomas. Os idiomas atualmente suportados em ISPIS são: alemão, inglês (*default*) e português. Para disponibilizar a ferramenta para usuários de outros idiomas basta que novos arquivos de tradução sejam criados.

4.2.4 – Apresentação Customizada

Para amenizar as diferenças de apresentação de ISPIS com outras ferramentas que possam estar sendo utilizadas nas diferentes organizações, os elementos de saída produzidos para o *browser* são formatados em arquivos CSS (Cascading Style Sheet) (MEYERS, 2004) que formatam os diferentes elementos da linguagem HTML. Assim, neste arquivo pode ser configurado o tamanho e cor da fonte de ISPIS, cor de fundo, cor de tabelas, estilos de botões, entre outras coisas, sem que haja a necessidade de se alterar código fonte.

4.3 – Instanciação da Arquitetura Proposta

Na seção 3.2 foi apresentada uma proposta de solução arquitetural para que a infra-estrutura pudesse lidar com diferentes tipos de artefatos através do uso de

ferramentas externas. Esta seção apresenta a instanciação desta solução e identifica as ferramentas que formam a configuração atual do apoio existente para inspeções de software na COPPE/UFRJ. A instanciação está ilustrada na Figura 4.4.

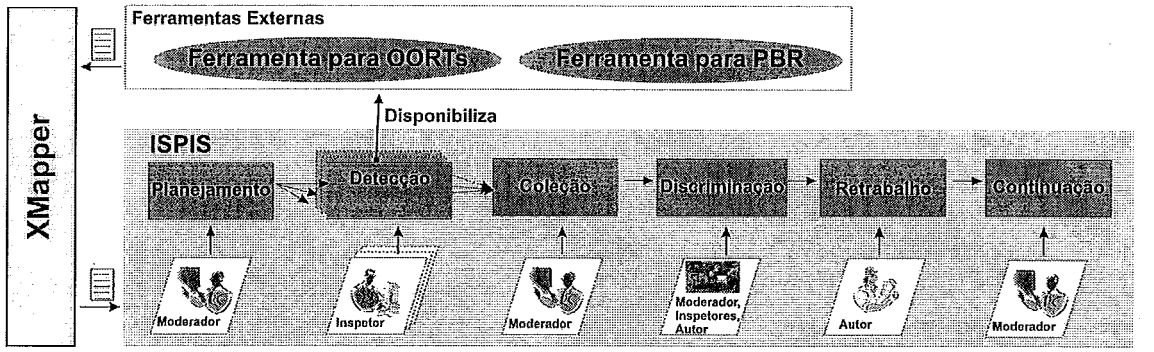


Figura 4.4. Instanciação da solução arquitetural proposta (KALINOWSKI *et al.*, 2004).

ISPIS desempenha o papel da infra-estrutura principal, automatizando o controle do processo de inspeção. Na atividade de detecção de defeitos ISPIS disponibiliza as ferramentas externas adequadas para o contexto da inspeção definido na atividade de planejamento. Atualmente, duas ferramentas externas para apoiar a aplicação de técnicas de leitura de artefatos específicos estão disponíveis na COPPE/UFRJ. Elas apóiam a aplicação de PBR em documentos de requisitos (SILVA e TRAVASSOS, 2004) e das OORT's em documentos de projeto (REIS e TRAVASSOS, 2003). Os dados gerados pela utilização destas ferramentas podem ser carregados em ISPIS mediante a utilização de um conversor XML gerado pela ferramenta de integração xMapper (SPINOLA et al., 2004).

Apresentada essa visão, a seguir o xMapper e as ferramentas de apoio à PBR e OORT's serão descritos em maiores detalhes:

- **xMapper.** Ferramentas inseridas no mesmo domínio de aplicação podem representar dados de semântica equivalente de forma diferente. As diferenças podem ter sua origem nas:
    - possíveis abordagens para armazenamento e manipulação dos dados, discrepância sintática;
    - diversas formas possíveis de se organizar um documento, discrepância estrutural;

- variadas possibilidades para definição de um mesmo conceito, discrepância semântica.

Assim, para garantir a interoperabilidade é preciso que haja um mecanismo para transformações estruturais e semânticas nos dados e uma tecnologia de armazenamento comum. O intercâmbio dos dados é feito utilizando o padrão XML. Desta forma, o problema da discrepância sintática é amenizado. Para lidar com a questão das discrepâncias estruturais e semânticas, o mecanismo implementado no xMapper utiliza esquemas, contendo as informações estruturais, e ontologias, contendo informações semânticas. Através da ontologia é possível lidar, por exemplo, com problemas como nomes distintos que fazem referência a um mesmo conceito, e através dos esquemas, considerar como os conceitos estão estruturados nos documentos.

De forma simplificada, a idéia é mapear na ontologia os esquemas dos documentos a serem integrados, obtendo assim um mapa de integração. Isto centraliza informações semânticas e estruturais tornando possível a geração automatizada de conversores entre os artefatos. Maiores detalhes sobre esta abordagem de integração de ferramentas podem ser encontrados em (SPINOLA *et al.*, 2004).

- **Ferramentas para aplicação de PBR e para Casos de Uso.** A ferramenta para a aplicação de PBR fornece um guia para a execução das tarefas especificadas pela técnica para a perspectiva do usuário e facilita o relato das discrepâncias identificadas.

  Este apoio ferramental para a identificação de discrepâncias pode trazer uma série de benefícios para a inspeção de requisitos de software. A ferramenta provê uma série de funcionalidades para auxiliá-los no entendimento da técnica, diminuindo a necessidade de se consultar materiais de referência e reduzindo assim o tempo total de inspeção. Além disso, a experiência na aplicação manual da técnica tem mostrado que, muitas vezes, os inspetores identificam pontos no documento onde existem defeitos, mas não são capazes de expressar claramente o defeito existente.

  Entre as tarefas propostas por PBR, o inspetor deve produzir um artefato de alto nível de acordo com a perspectiva assumida. Na perspectiva do usuário, por exemplo, um esboço dos casos de uso do sistema deve ser descrito. Vale

destacar que estes artefatos inicialmente esboçados devem ser detalhados em fases posteriores do desenvolvimento. Neste caso, os modelos de caso de uso poderão ser exportados para uma ferramenta de Casos de Uso (SILVA *et al.*, 2004). Esta ferramenta foi definida e construída para auxiliar na construção de modelos de casos de uso, capturando os conceitos que permitem auxiliar atividades subseqüentes do desenvolvimento de software.

- **Ferramenta para aplicação de OORT's.** Esta ferramenta objetiva fornecer apoio automatizado à aplicação e configuração de OORT's (REIS e TRAVASSOS, 2003). Nesta proposta, o apoio à aplicação das OORT's consiste em disponibilizar mecanismos que auxiliem e orientem o inspetor na execução de tarefas que compõem a técnica de leitura selecionada, executem automaticamente o maior número possível de tarefas, e auxiliem o armazenamento e a identificação de discrepâncias.

4.3 – Funcionamento Detalhado

Nesta seção, o decorrer de um processo de inspeção na infra-estrutura de apoio é apresentado de forma a ilustrar seu funcionamento detalhado. As sub-seções seguintes apresentam o apoio fornecido a cada uma das atividades do processo de inspeção.

4.3.1 - Planejamento

Na atividade de planejamento ISPIS auxilia o moderador na seleção de inspetores, visando aumentar a cobertura de defeitos da inspeção. Para isto permite a definição do contexto da inspeção (Figura 4.5) e a edição da caracterização dos diferentes inspetores. Em seguida, estes dados são utilizados em conjunto com o desempenho dos inspetores em inspeções passadas para apoiar a seleção de inspetores.

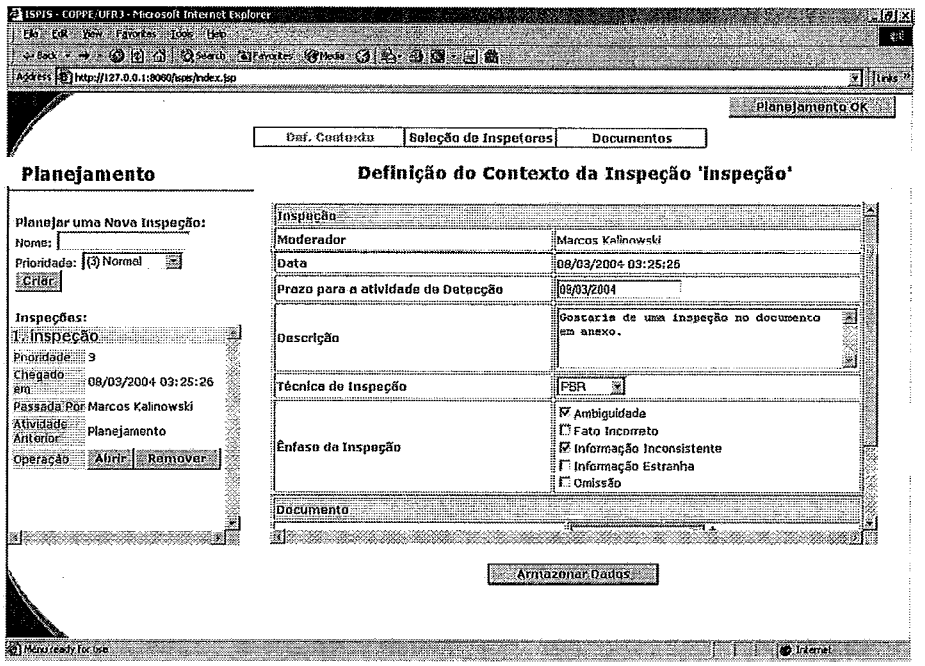


Figura 4.5. Definição do contexto de uma inspeção.

Seguindo a proposta descrita na seção 3.3.1, ISPIS apóia a seleção fornecendo uma lista ordenada dos inspetores mais indicados para a fase de inspeção individual (Figura 4.6).

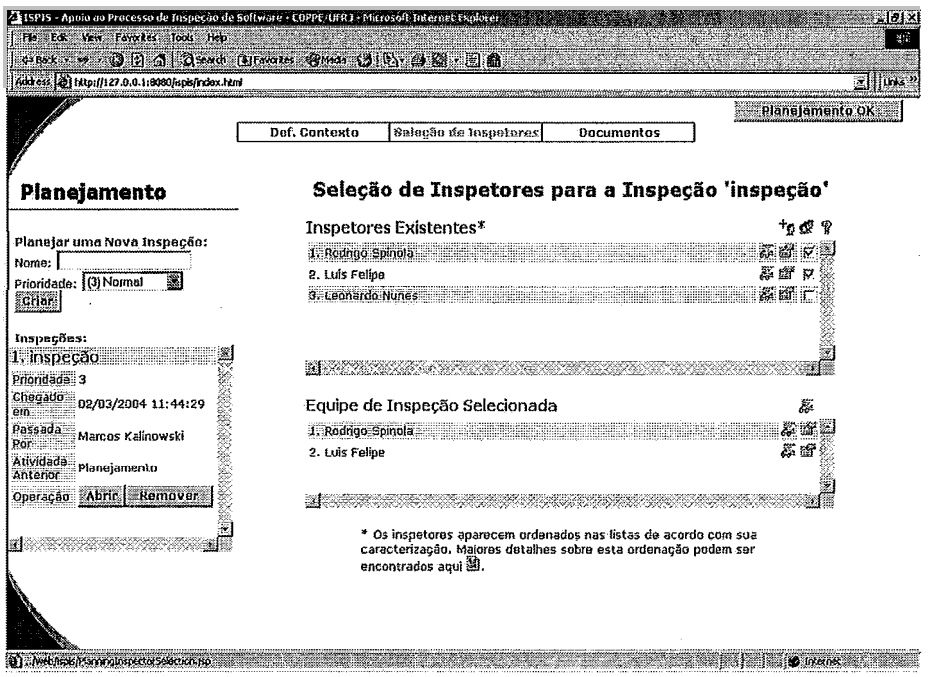


Figura 4.6. Lista ordenada dos inspetores mais indicados para a inspeção.

Para implementar esta proposta, um conjunto inicial de seis das hipóteses fundamentadas de CARVER (2003) foi selecionado para inspeções de requisitos e de três para inspeções de documentos de projeto. Estas hipóteses estão descritas através de expressões em lógica de primeira ordem que ilustram as características do inspetor e da inspeção que combinadas implicam em mais defeitos encontrados:

- Hipóteses Fundamentadas para Documentos de Requisitos:
  - R1: O inspetor possui experiência com desenvolvimento de software;
  - R2: O inspetor possui experiência escrevendo requisitos;
  - R3: (O inspetor possui experiência escrevendo casos de uso) ^ *not* (a inspeção utiliza uma *checklist* procedural, bem definida);
  - R4: (O inspetor possui experiência com Projetos de Software) ^ *not* (a inspeção utiliza uma checklist procedural, bem definida);
  - R5: (O inspetor possui conhecimento do domínio) ^ *not* (a inspeção utiliza uma técnica procedural);
  - R6: (O inspetor possui experiência revisando requisitos) ^ (o inspetor possui familiaridade com a língua de trabalho do artefato) ^ ((a inspeção é de um documento não familiar) ^ (a inspeção é de um documento de domínio não familiar)).

- Hipóteses Fundamentadas para Documentos de Projeto:
  - P1: O inspetor possui experiência com desenvolvimento de software;
  - P2: O inspetor possui conhecimento do domínio;
  - P3: O inspetor possui experiência revisando projetos orientados a objeto.

Para que as hipóteses fundamentadas pudessem ser aplicadas, a caracterização dos inspetores em ISPIS captura as características presentes como variáveis nas hipóteses. As características que atualmente fazem parte da caracterização dos inspetores em ISPIS são as que faziam parte da caracterização dos inspetores nos estudos analisados no próprio trabalho de CARVER (2003). São elas: habilidade de ler e compreender textos em inglês; habilidades para trabalhar em inglês; experiência prática com desenvolvimento de software; experiência escrevendo requisitos; experiência escrevendo casos de uso; experiência revisando requisitos; experiência revisando casos de uso; experiência modificando requisitos para manutenção; experiência em projeto de sistemas; experiência em desenvolver projetos

a partir de requisitos e casos de uso; experiência criando projetos orientado a objetos; experiência lendo projetos orientado a objetos; experiência com UML; experiência alterando projeto para manutenção; experiência com gerência de projetos de software; experiência com inspeções de software; conhecimento em diferentes domínios. As primeiras três características têm seus valores variando entre 1 e 4, as demais entre 1 e 5.

Além de contar com a lista ordenada, o moderador pode visualizar o perfil de cada um dos inspetores disponíveis e fazer uma análise mais subjetiva. O perfil de um inspetor é formado por sua caracterização e pelo seu desempenho em inspeções passadas sobre o mesmo tipo de artefato a ser inspecionado. A visualização do perfil do inspetor está ilustrada nas Figura 4.7 e 4.8.

Na caracterização, ilustrada na Figura 4.7, é possível ver o valor que o inspetor recebeu para cada uma das características bem como o valor desejado e a pontuação recebida para a característica em questão (desde que a característica tenha valor desejado conhecido para a inspeção).

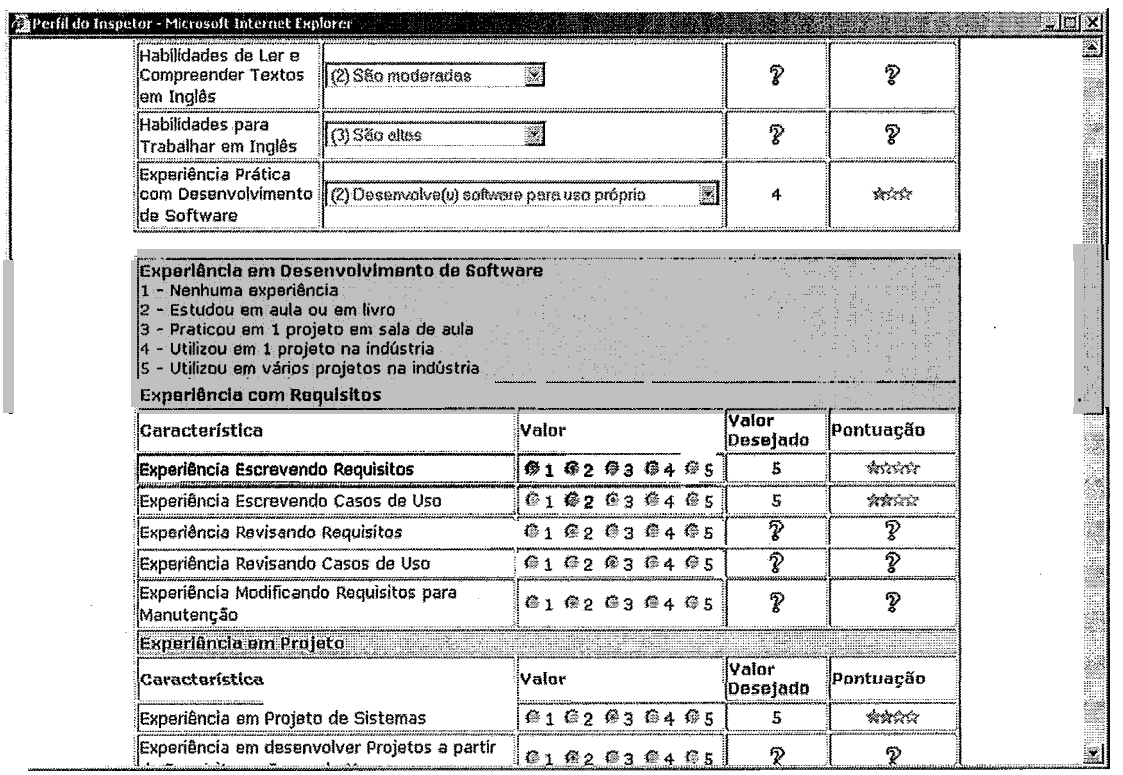


Figura 4.7. Perfil do Inspetor, parte da caracterização.

No desempenho em inspeções passadas neste mesmo tipo de artefato, ilustrado na Figura 4.8, é possível consultar a eficiência média do inspetor e a eficiência em sua última inspeção. Além disto, a ênfase no tipo de defeitos a serem encontradas na inspeção corrente é exibida e um gráfico para ver a distribuição dos tipos de defeitos encontrados pelo inspetor pode ser consultado.

É importante ressaltar que ISPIS apenas faz sugestões visando apoiar a seleção de inspetores. A decisão final sobre a montagem da equipe continua sob a responsabilidade do moderador.

Em relação à distribuição de material, ISPIS possibilita a disponibilização controlada de diversos tipos de documentos, que podem ser carregados (*upload*) para o servidor da aplicação e anexados à inspeção que está sendo planejada (Figura 4.9). Estes documentos podem representar o artefato a ser inspecionado propriamente dito ou material de apoio à realização da inspeção, como por exemplo checklists ou outras técnicas de leitura.

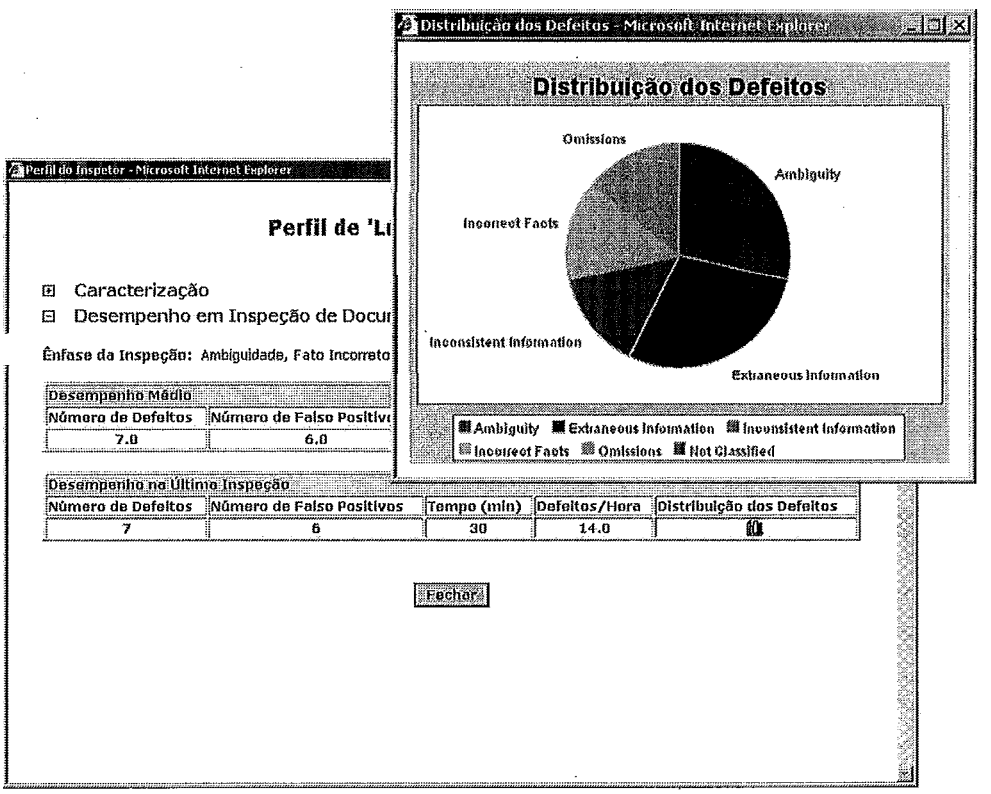


Figura 4.8. Perfil do Inspetor. Parte de desempenho em inspeções passadas sobre o mesmo tipo de artefato.

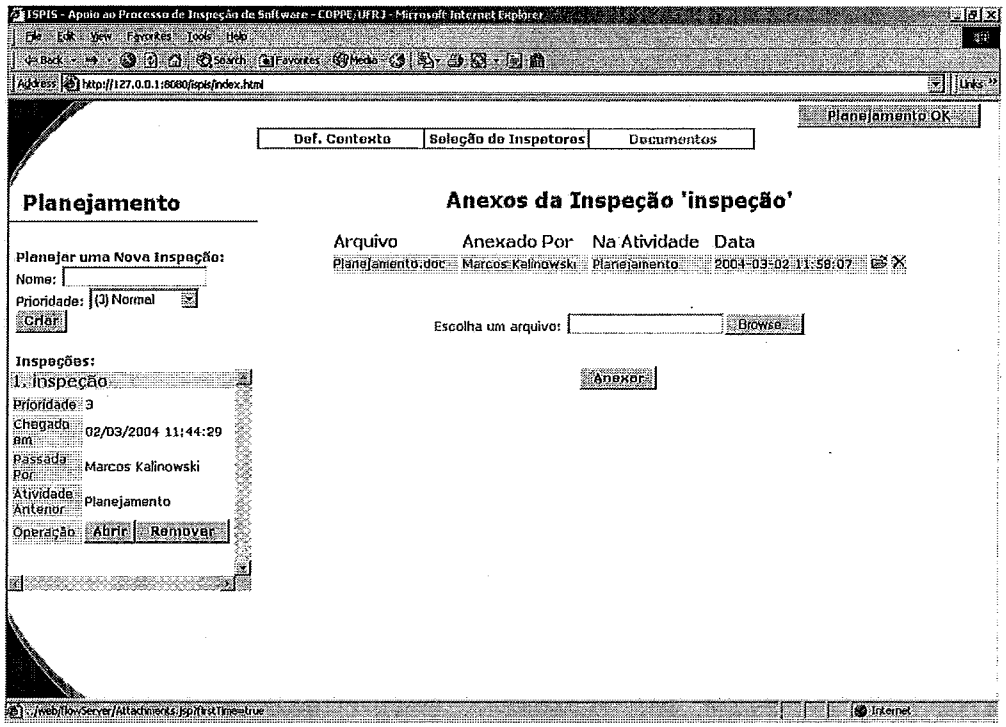


Figura 4.9. Anexando documentos a inspeções em ISPIS.

No momento em que o planejamento se encerra, ISPIS possui informação suficiente para definir a estrutura do processo de inspeção de (SAUER et al., 2000). São criadas nos metadados do PatternFlow:

- Uma atividade de detecção de defeitos para cada um dos inspetores selecionados (caso ainda não exista). Para cada uma das atividades criadas é dado acesso ao inspetor em questão.
- Uma atividade de coleção de defeitos para a inspeção em questão, que sincroniza os dados das fases de detecção criadas para a inspeção.
- Uma atividade de discriminação de defeitos para a inspeção em questão. O autor do documento, o moderador e os inspetores recebem acesso a esta atividade.
- Uma atividade de retrabalho para o autor do documento (caso ainda não exista). Somente o autor do documento recebe acesso a esta atividade.
- Uma atividade de continuação para o moderador da inspeção (caso ainda não exista). Somente o moderador da inspeção recebe acesso a esta atividade.

- As regras de desvio entre estas atividades, de forma que elas estejam encadeadas de acordo com o processo de inspeção de (SAUER et al., 2000).

Esta forma dinâmica de definição do processo possibilita a ISPIS coordenar a realização de diversas inspeções ocorrendo ao mesmo tempo. Note que, na definição do processo existirão tantas atividades de coleção e de discriminação quanto forem as inspeções planejadas em ISPIS. As regras de desvio possibilitam que a atividade correta seja acionada para cada uma das inspeções.

Após redefinir a estrutura do processo, ISPIS encaminha a inspeção para as atividades de detecção dos inspetores selecionados.

4.3.2 – Detecção de Defeitos

Para que possa ser utilizada em qualquer tipo de artefato, ISPIS possibilita, em sua configuração básica, inspeções *ad-hoc*. Como visto na seção 4.2, um apoio mais específico para a inspeção de determinados artefatos é fornecido através da integração a ISPIS de ferramentas externas.

Na detecção de defeitos *ad-hoc*, ilustrada na Figura 4.10, as tarefas do inspetor se resumem a ler o documento e cadastrar discrepâncias.

Na detecção de defeitos através da integração ISPIS disponibiliza a ferramenta adequada para a inspeção em questão para *download*, o inspetor então baixa a ferramenta e a utiliza para realizar a detecção de defeitos. Durante a detecção, um documento XML, descrevendo as discrepâncias encontradas é produzido pela ferramenta. Estes documentos podem ser carregados em ISPIS. Após carregar os documentos, eles podem ser ativados. A ativação de um documento faz com que a lista de discrepâncias contida neste se torne a lista de discrepâncias de ISPIS. A Figura 4.11 ilustra a utilização de ISPIS em conjunto com a ferramenta de apoio à aplicação de PBR.

A ativação de uma lista de discrepâncias ocorre internamente em dois passos. Primeiramente ISPIS transforma o XML do documento a ser ativado para que possa ser lido, utilizando o conversor entre os artefatos das ferramentas gerado através da utilização do xMapper. Depois ISPIS lê o documento XML e carrega seus dados em sua base de dados específicos para apoiar inspeções de software.

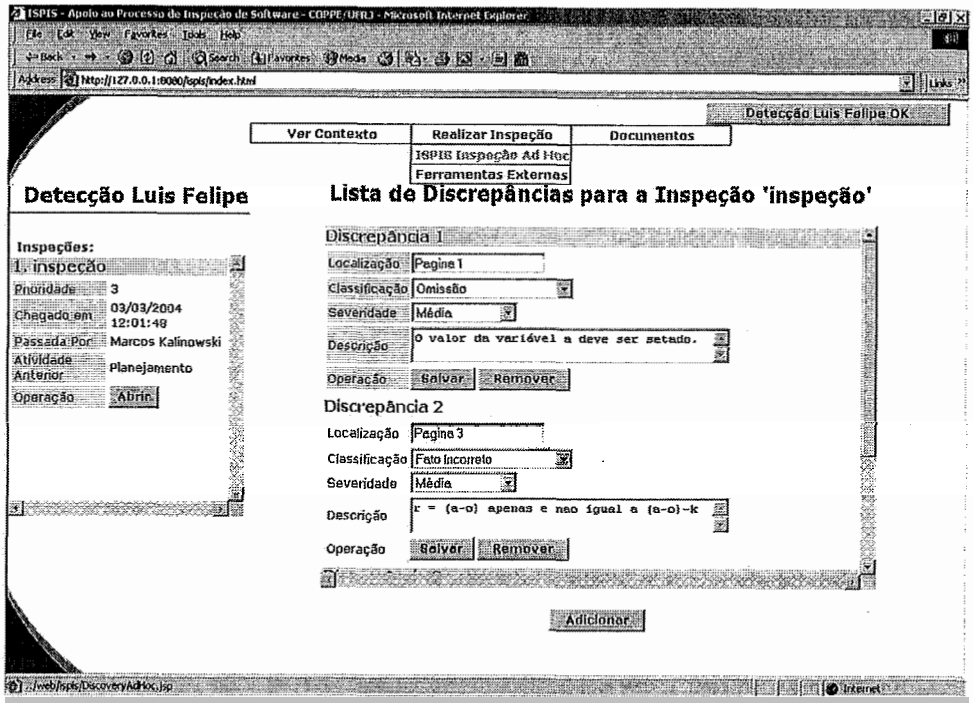


Figura 4.10. Detecção de defeitos ad-hoc em ISPIS.

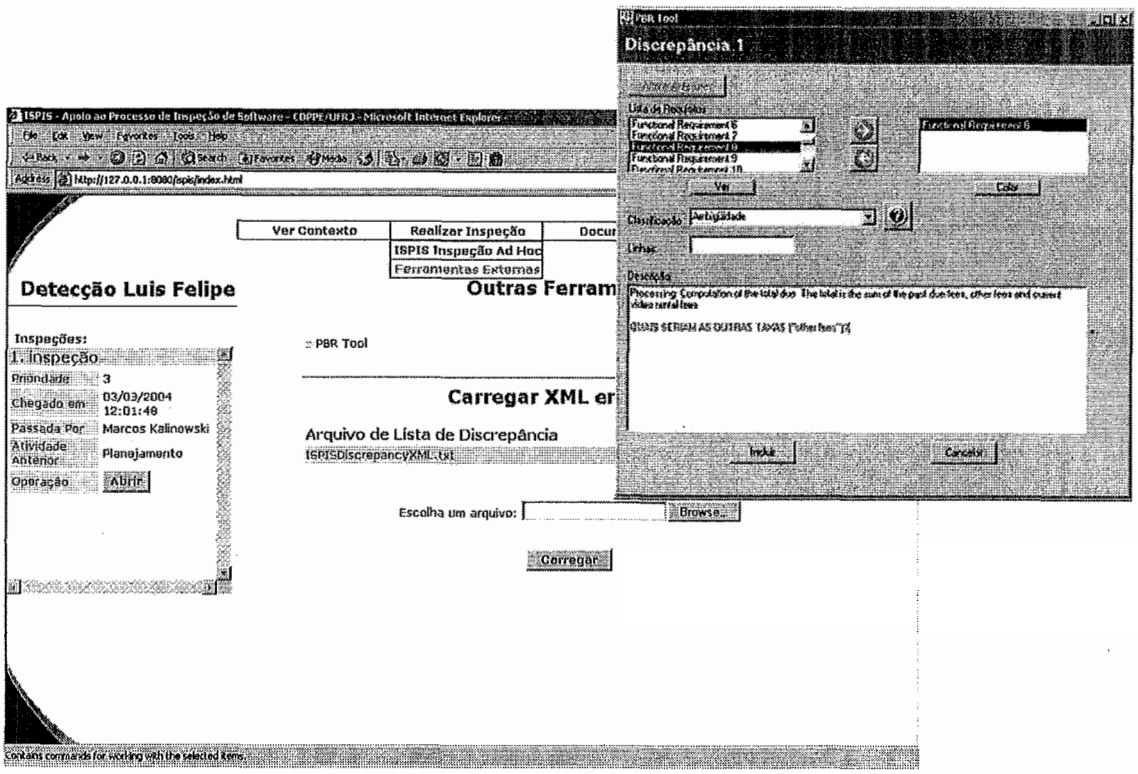


Figura 4.11. Detecção através da integração para uma inspeção aplicando a técnica PBR, utilizando a ferramenta correspondente.

Ao final da atividade de detecção o inspetor é solicitado por ISPIS a informar o tempo que ele levou efetivamente inspecionando o documento e a fazer uma estimativa do número de defeitos que ele acredita estarem presentes neste. Estes dados são utilizados respectivamente na atividade de planejamento, para calcular a eficiência (defeitos/hora) do inspetor e na atividade de continuação da inspeção, para estimar o número de defeitos do documento utilizando o WAO (BIFFL *et al.*, 2003) e estimar a cobertura de uma nova inspeção utilizando o modelo ILM (ADAMS, 1999).

Quando uma atividade de detecção de um inspetor é concluída, seus dados são sincronizados na atividade de coleção de defeitos da inspeção. A inspeção é disponibilizada em sua atividade de coleção de defeitos no momento que a última atividade de detecção de defeitos é concluída.

4.3.3 – Coleção de Defeitos

Na atividade de coleção de defeitos, o moderador da inspeção tem acesso a todas as listas de discrepâncias produzidas na atividade de detecção de defeitos, conforme ilustra a Figura 4.12. Ele poderá então selecionar discrepâncias destas listas e as descartar ou classificar como replicadas, caso haja mais de uma discrepância representando o mesmo defeito. Quando uma discrepância é descartada ela é classificada como um falso positivo e não será mais exibida.

Quando discrepâncias são classificadas como replicadas, por sua vez, fica sob responsabilidade do moderador selecionar uma das versões da discrepância. A versão selecionada poderá ainda ser editada, de modo que informações importantes contidas nas réplicas possam ser acrescentadas. Seguindo a sugestão experimentalmente avaliada de (LANUBILE e MALLARDO, 2003), a versão selecionada é então classificada como defeito e encaminhada diretamente para a atividade de retrabalho. As demais versões da discrepância são classificadas como defeito também, para manter os dados estatísticos sobre o desempenho dos inspetores consistente, no entanto não serão mais exibidos no decorrer da inspeção. A Figura 4.13 ilustra uma discrepância sendo classificada como replicada.

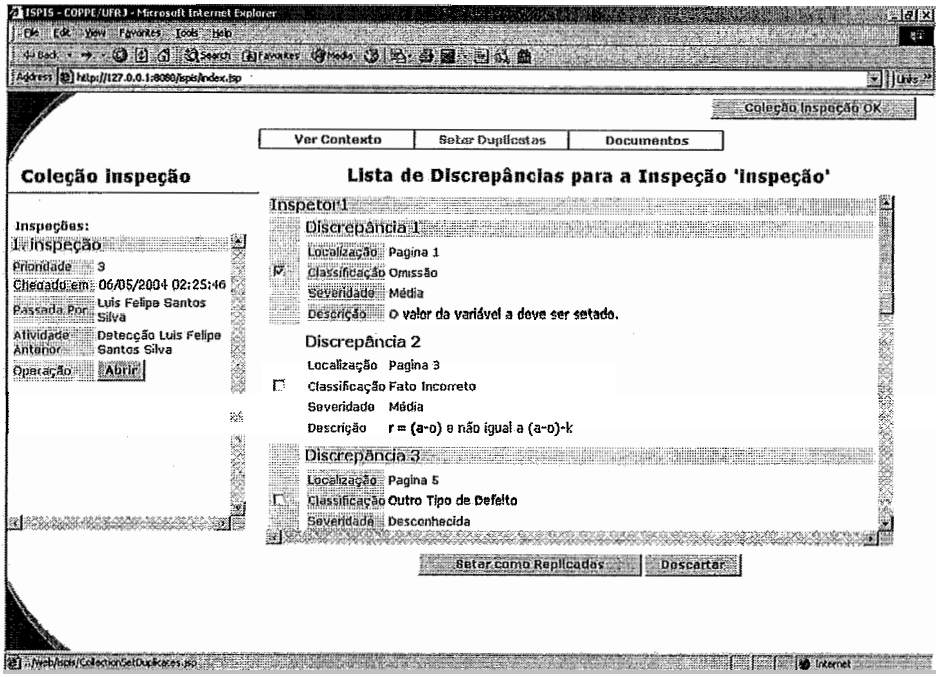


Figura 4.12. Listas de discrepância dos inspetores.

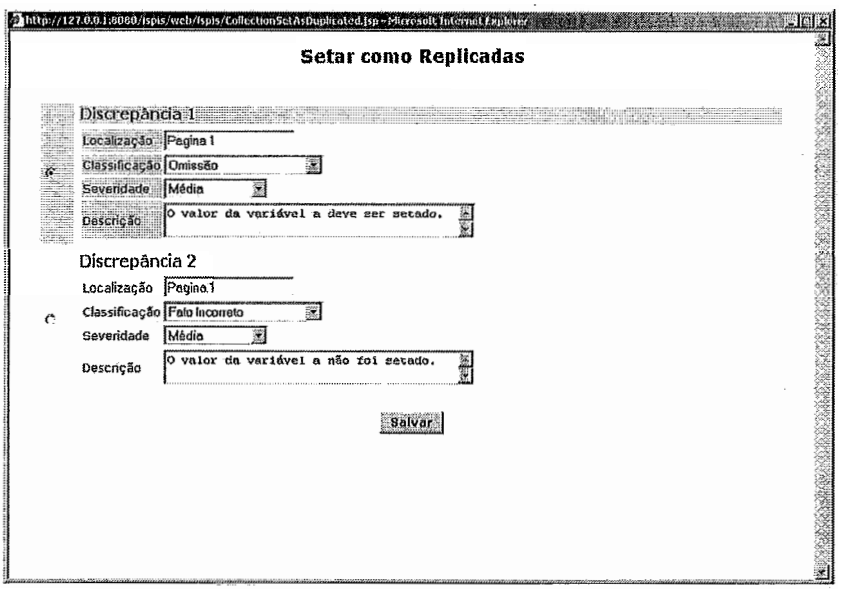


Figura 4.13. Classificando uma discrepância como replicada.

No momento que o moderador define a atividade de coleção de defeitos como concluída a inspeção é passada para sua atividade de discriminação de defeitos. Além disto ocorre uma mudança na definição do processo, removendo o acesso do moderador à atividade de coleção desta inspeção.

4.3.4 – Discriminação de Defeitos

O moderador, o autor do documento e os inspetores têm participação nesta atividade. Durante a discriminação as discrepâncias são tratadas como tópicos de discussão. Cada participante pode acrescentar seus comentários relativos a cada uma das discrepâncias, que fica disponível como tópico de discussão enquanto o moderador não decidir se representa um defeito ou um falso positivo (Figura 4.14).

Seguindo a proposta da seção 3.3.4, os participantes da discriminação em ISPIS são anônimos. Assim, nos comentários é exibido o papel desempenhado pelo participante da discriminação ao invés do seu nome.

A caixa de seleção da discrepância aparece em destaque para todos os participantes que já a comentaram, indicando para eles que estão efetivamente participando da discussão desta discrepância. Na Figura 4.15 as discrepâncias 1 e 2 do inspetor 2 aparecem em destaque para o moderador, depois que este as comentou. Este recurso foi sugerido pelos participantes de um estudo de caso, descrito no capítulo 5, para facilitar a orientação dos participantes em discriminações com muitas discrepâncias.

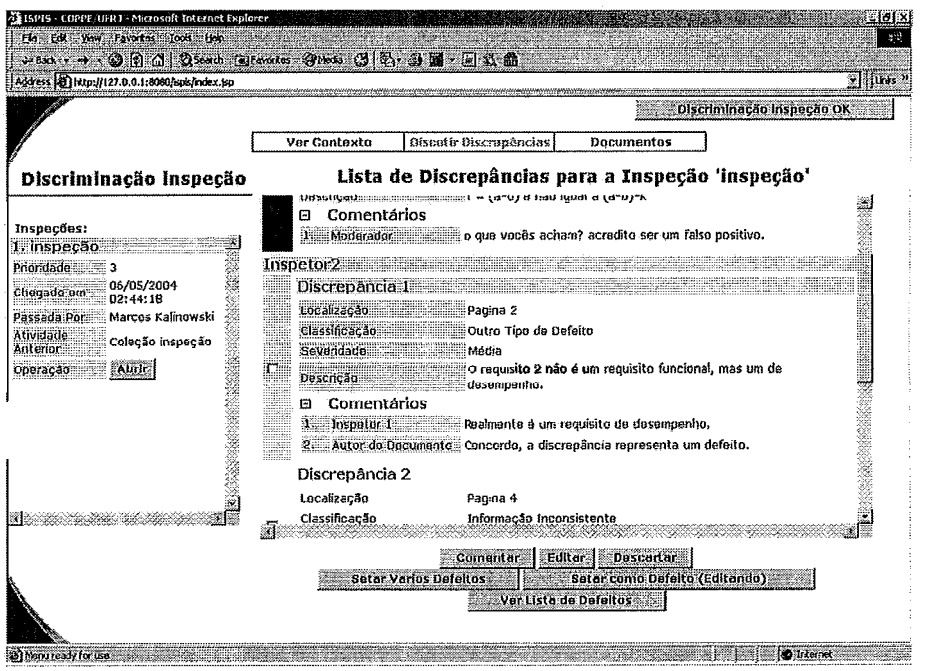
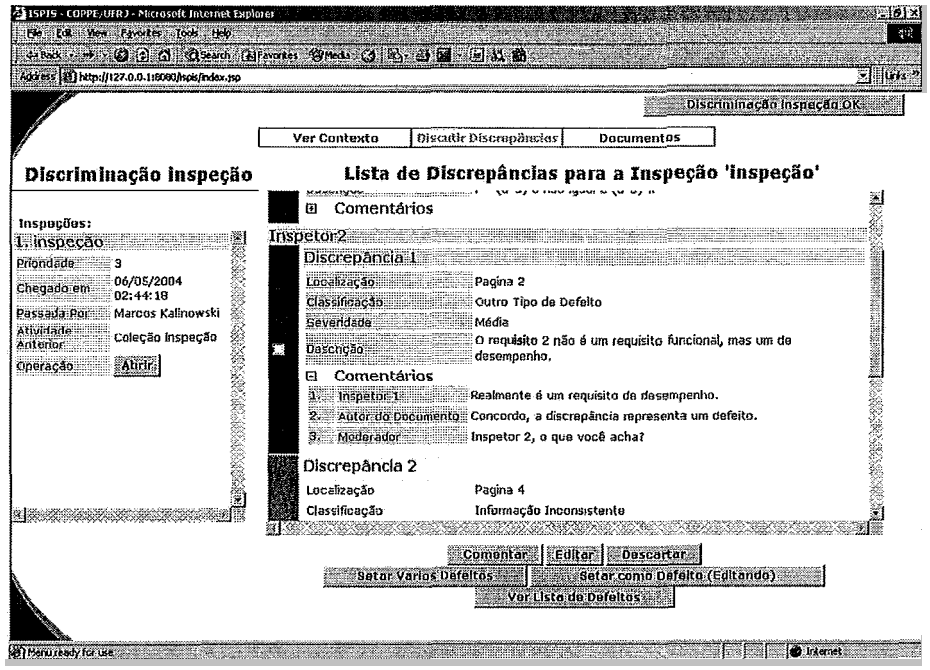


Figura 4.14. Discriminação de Defeitos em ISPIS (visão do moderador).



4.15. Discrepâncias destacadas já comentadas pelo participante aparecem para ele em destaque.

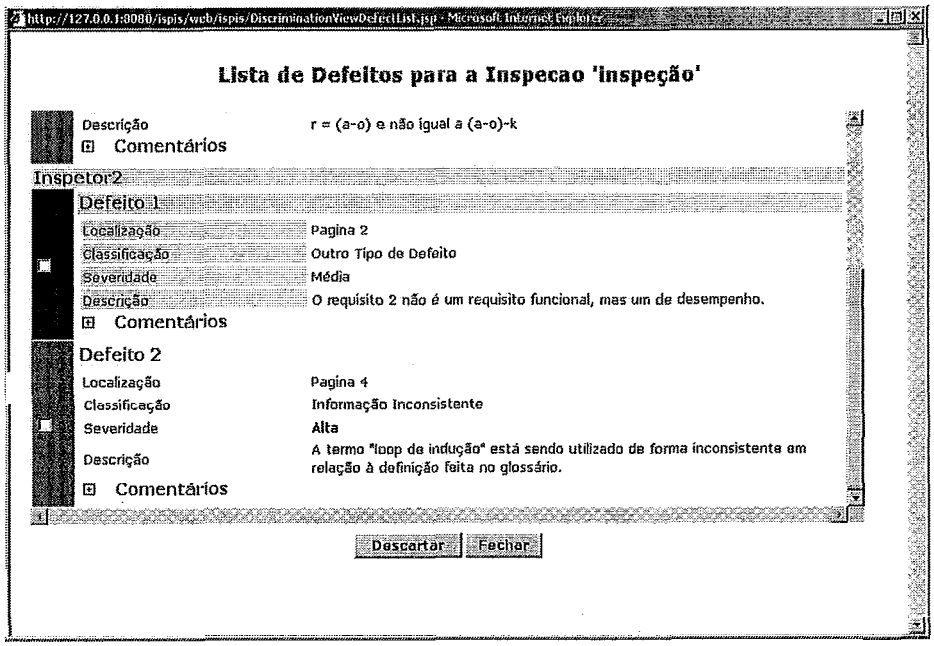


Figura 4.16. Visualização da lista de defeitos pelo moderador durante a discriminação.

Só o moderador tem acesso a editar, descartar e classificar discrepâncias como defeitos e ver a lista de defeitos atual em uma nova janela. Defeitos podem ser descartados da lista de defeitos atual (Figura 4.16) e assim ser retornados para a

discussão, possibilitando que uma decisão errada de classificação possa ser corrigida ou dúvidas esclarecidas.

Os demais participantes estão limitados a adicionar comentários. Além disto, só o moderador tem acesso à funcionalidade de definir a discriminação como concluída.

No momento que o moderador define a discriminação como concluída a inspeção é encaminhada para a atividade de retrabalho do autor do documento sendo inspecionado. Além disto o acesso dos participantes à atividade de discriminação desta inspeção, já concluída, é removido.

4.3.5 – Retrabalho

Seguindo a proposta da seção 3.3.5, nesta atividade, ISPIS não fornece apoio à correção dos defeitos no artefato. Após a correção do artefato, a versão corrigida pode ser anexada, neste momento ISPIS automaticamente gera um backup da versão antiga do artefato, que também continua anexado à inspeção.

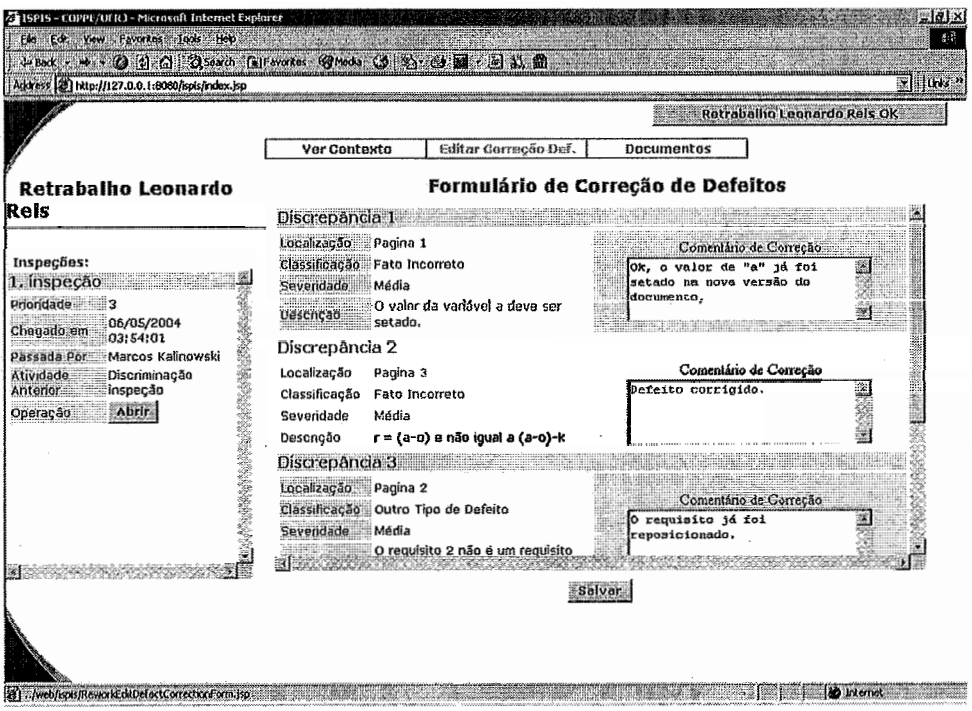


Figura 4.17. Edição do formulário de correção de defeitos pelo autor do artefato.

Além disto, as correções dos defeitos podem ser descritas, criando um formulário de correção de defeitos. Assim, para cada defeito (proveniente diretamente da fase de coleção ou da fase de discriminação) um comentário de correção pode ser adicionado. A edição do formulário de correção de defeitos pelo autor está ilustrada na Figura 4.17.

No momento em que o autor define o retrabalho como concluído, a inspeção é encaminhada para a fase de continuação.

4.3.6 – Continuação

Nesta fase, o moderador tem acesso ao contexto da inspeção, definido na fase de planejamento, às versões defeituosas e corrigidas do artefato e ao relatório de correção de defeitos.

Seguindo a proposta descrita na seção 3.3.6, uma das formas utilizadas em ISPIS para apoiar a decisão do moderador de realizar ou não uma nova inspeção é apresentar uma estimativa da cobertura de defeitos atual e da cobertura de defeitos após a próxima inspeção.

Para estimar a cobertura de defeitos atual é preciso estimar o número de defeitos presentes no documento. Esta estimativa é feita utilizando o WAO (BIFFL et al., 2003). Para a estimativa da cobertura de defeitos da próxima inspeção o modelo ILM (ADAMS, 1999) é utilizado. Nesta estimativa, ISPIS considera que a variável subjetiva 'esforço' do modelo ILM tenha o mesmo valor nas duas inspeções. Portanto a estimativa da cobertura é para uma inspeção com o mesmo esforço.

ISPIS então sugere uma nova inspeção se a cobertura de defeitos da inspeção atual for menor do que 70% e a cobertura de defeitos após a próxima inspeção maior do que 70%. São apresentados ao moderador: a cobertura estimada da inspeção atual, a cobertura estimada para a próxima inspeção e a sugestão de ISPIS.

Outra forma de apoiar a decisão utilizada em ISPIS é utilizar dados históricos sobre inspeções no mesmo tipo de artefato. ISPIS apresenta ao usuário o número de defeitos encontrados na inspeção atual e a média do número de defeitos em inspeções anteriores sobre o mesmo tipo de artefato. Seguindo a diretriz de (FAGAN, 1976), ISPIS então sugere uma nova inspeção se o número de defeitos encontrados no documento for 5% maior do que a média.

A Figura 4.18 mostra ISPIS sugerindo uma nova inspeção para uma inspeção que teve como cobertura atual estimada o valor 62.5% e cobertura estimada após a próxima inspeção 85.9375%. No entanto, de acordo com os dados históricos o artefato se mostrou pouco defeituoso e ISPIS não sugere uma nova inspeção.

Por seguirem fundamentos diferentes, as duas sugestões de ISPIS podem ser conflitantes. No entanto os dados são apresentados explicitamente ao moderador, que é responsável pela decisão final sobre a realimentação do processo de inspeção ou não, independente das sugestões.

No momento em que o moderador define a atividade de continuação como concluída, a inspeção atual é finalizada em ISPIS. Se o moderador decidiu re-inspecionar, uma nova inspeção é criada em ISPIS e movida para a atividade de planejamento aproveitando parcialmente a definição do contexto da inspeção anterior. Esta forma de proceder torna possível distinguir as diferentes ocorrências da inspeção e seus dados históricos.

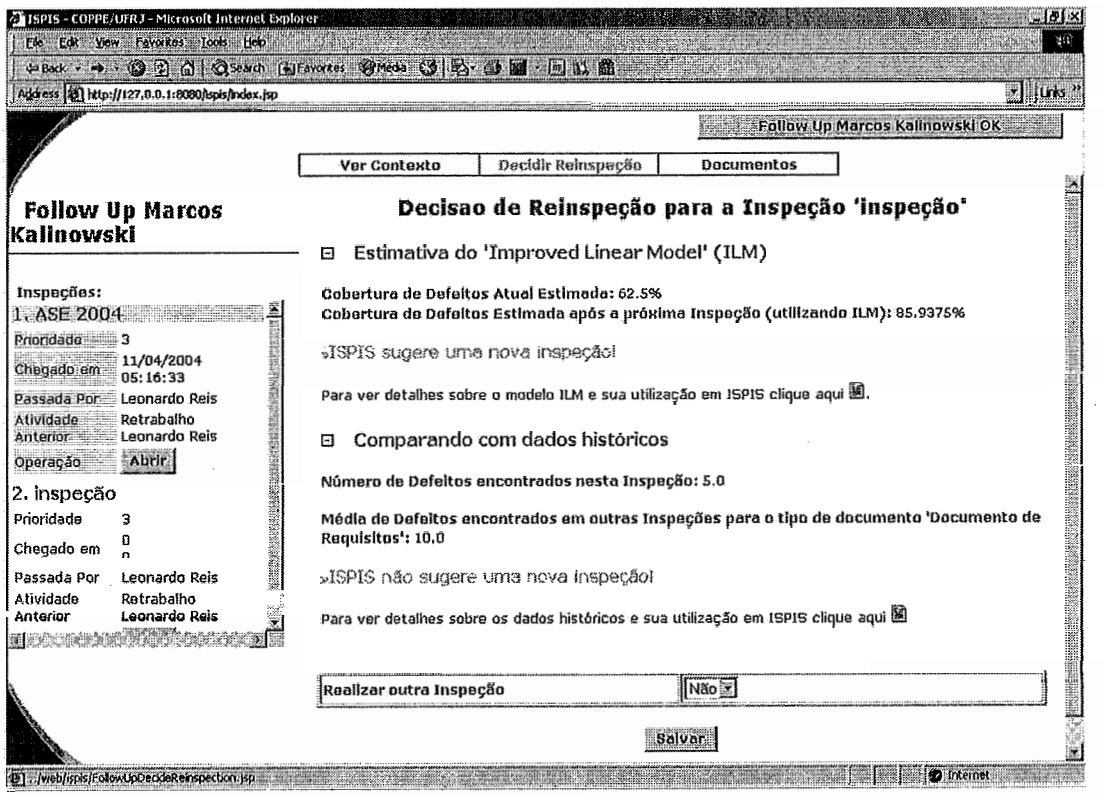


Figura 4.18. Apoio à decisão de realizar ou não uma nova inspeção.

4.3.7 – Apoio à Gerência do Processo de Inspeção

Baseado no gerenciamento ativo de inspeções (HALLING et al., 2002), conforme descrito na proposta da seção 3.4, ISPIS: (1) auxilia as decisões tomadas no planejamento com base na avaliação de inspeções passadas, (2) permite o monitoramento das inspeções correntes, (3) permite avaliações pós-processo visando entender a relação entre processo de inspeção, inspetores e artefato inspecionado e assim (4) permite a melhoria do processo.

No monitoramento cada instância de inspeção que foi ou está sendo realizada pode ser acompanhada pelo seu moderador. As Figuras 5.19 e 5.20 ilustram o monitoramento das inspeções em ISPIS. Na Figura 5.19 a inspeção a ser monitorada deve ser escolhida entre as inspeções ativas ou as inspeções concluídas. A Figura 5.20, por sua vez, ilustra o monitoramento de uma determinada inspeção, onde são exibidos para cada passagem entre atividades: a atividade de origem, a atividade de destino, quem realizou a atividade de origem e o número de instâncias desta inspeção que se encontram na atividade de destino. Este último dado é útil na atividade de coleção de defeitos, onde ilustra quantos inspetores já concluíram sua atividade de detecção de defeitos.

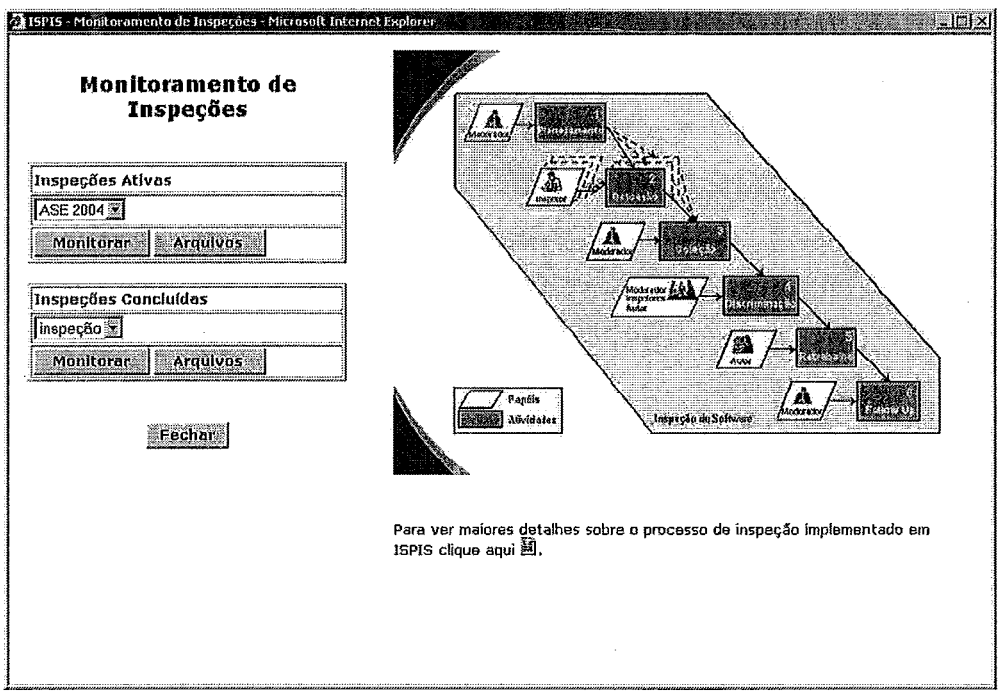


Figura 5.19. Monitoramento de inspeções: escolhendo a inspeção a ser monitorada.

Figura 5.20. Monitoramento de inspeções: monitorando uma determinada inspeção.

Ainda referente ao monitoramento, o moderador tem acesso a entrar em ISPIS como cada um dos inspetores envolvidos em suas inspeções para ter uma visão mais detalhada do progresso na detecção de defeitos e poder encerrar estas atividades quando lhe convier.

Em ISPIS, o apoio à avaliação pós-processo, onde o moderador deve ter acesso a dados históricos detalhando o desempenho dos inspetores em inspeções passadas, está contemplado na atividade de planejamento. Assim, basta o moderador entrar na atividade de planejamento para ter acesso a estes dados.Para possibilitar mudanças na estrutura do processo a definição desta estrutura está encapsulada em ISPIS está encapsulada e utiliza as facilidades do PatternFlow para definição de processos, permitindo fácil alteração. Esta facilidade pode ser explorada para fins experimentais.

Além disto, ISPIS utiliza notificações automáticas por e-mail para os participantes da inspeção sempre que uma atividade precisa ser realizada por estes. O que facilita a coordenação do processo de inspeção.

4.4 – Conclusão

Este capítulo apresentou ISPIS, a infra-estrutura computacional de apoio ao processo de inspeção obtida a partir da implementação da proposta descrita no capítulo 3. Tendo em vista a motivação para implementar ISPIS, existem duas expectativas para ela: (1) tornar revisões mais sistematizadas possíveis na prática e (2) fornecer apoio para a tomada de decisões corretas durante o processo de inspeção de software, para que inspeções mais eficientes possam ser alcançadas.

Para avaliar a primeira destas expectativas um estudo de caso foi realizado. A avaliação da segunda expectativa, por sua vez, é uma tarefa mais complexa. Muitas formas de apoiar a tomada de decisões em ISPIS se baseiam em resultados de estudos experimentais, o que nos leva a acreditar que o apoio seja adequado. No entanto, algumas formas de apoio são novas propostas deste trabalho, como o apoio à atividade de planejamento. Assim, o primeiro passo em direção a avaliar se a utilização de ISPIS pode resultar em inspeções mais eficientes foi um estudo experimental para avaliar o apoio de ISPIS ao planejamento. O estudo experimental e o estudo de caso estão descritos no capítulo seguinte.

Capítulo 5
Avaliação da Infra-estrutura

Este capítulo apresenta um estudo experimental para avaliar o apoio da infra-estrutura ISPIS à atividade de planejamento do processo de inspeção e um estudo de caso avaliando a viabilidade de se utilizar ISPIS em inspeções reais.

5.1 - Introdução

Experimentação é o centro do processo científico. No campo da engenharia de software, a utilização de métodos experimentais pode trazer alguns benefícios dentre os quais: acelerar o progresso através da rápida eliminação de abordagens não fundamentadas, suposições errôneas e modismos, ajudando a orientar a engenharia e a teoria para direções promissoras. É possível encontrar na literatura três estratégias experimentais: *survey*, estudo de caso e experimento. Uma descrição destas estratégias, pode ser encontrada em (TRAVASSOS *et al.*, 2002).

A estratégia experimental mais apropriada em uma situação concreta depende dos objetivos do estudo, das propriedades do processo de software usado durante a experimentação e dos resultados finais esperados (TRAVASSOS *et al.*, 2002).

Neste capítulo é apresentado um estudo experimental para avaliar o apoio fornecido pela infra-estrutura ISPIS para a atividade de planejamento de inspeções (seção 5.2) e um estudo de caso avaliando a viabilidade de se utilizar ISPIS em inspeções reais (seção 5.3).

5.2 – Estudo Experimental: Avaliando o Apoio de ISPIS ao Planejamento de Inspeções.

O estudo experimental visa a avaliar o apoio fornecido por ISPIS ao planejamento de inspeções, uma vez que esta atividade utiliza uma proposta de apoio

nova, elaborada neste trabalho. O processo de experimentação seguido foi o descrito em (WOHLIN *et al.*, 2000) que está ilustrado na Figura 5.1. Uma descrição resumida deste processo, baseada em (TRAVASSOS et al., 2002) se encontra no parágrafo seguinte.

A definição é a primeira atividade, onde o experimento é expresso em termos dos problemas e objetivos. A atividade de planejamento vem em seguida, onde o projeto do experimento é determinado, a instrumentação é considerada e os aspectos da validade do experimento são avaliados. A execução do experimento segue o planejamento. Neste momento os dados experimentais são coletados para serem analisados e avaliados na atividade de análise e interpretação. Finalmente, os resultados são apresentados e empacotados durante a atividade de apresentação e empacotamento.

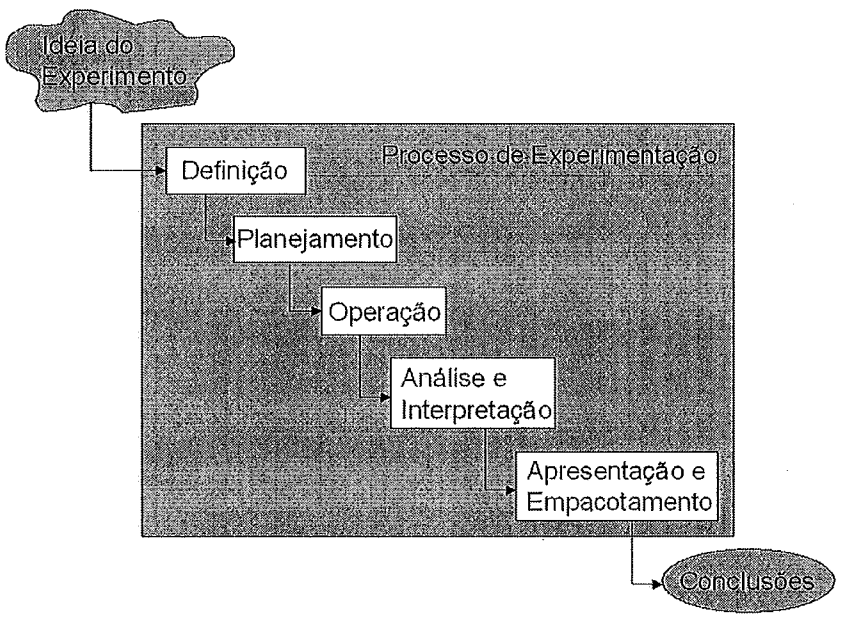


Figura 5.1. Processo de experimentação. Adaptado de (WOHLIN et al., 2000).

A apresentação do estudo experimental segue o esboço para relatar experimentos sugerido em (WOHLIN et al., 2000). Assim, o experimento é relatado em cinco seções: a definição do experimento, o planejamento do experimento, a operação do experimento propriamente dita, a análise dos dados e a interpretação dos resultados. O pacote do experimento está disponível na COPPE/UFRJ e instrumentos deste podem ser encontrados no Apêndice C.

5.2.1 Definição do Estudo Experimental

A fase de definição do estudo experimental descreve os objetivos de sua realização. No contexto do nosso estudo o objetivo é avaliar se o apoio fornecido pela infra-estrutura ISPIS para a atividade de planejamento de inspeções é adequado e se traz benefícios quando comparado à realização manual desta atividade.

A estrutura a seguir, baseada no método GQM (BASILI et al., 1994), define o estudo experimental.

Analisar o apoio ferramental fornecido pela infra-estrutura ISPIS ao planejamento de inspeção de software
com o propósito de caracterizar
com respeito a cobertura de defeitos das inspeções seguindo os planos elaborados, tempo de elaboração dos planos e grau de satisfação dos usuários na realização das tarefas
do ponto de vista do pesquisador
no contexto de desenvolvedores de software utilizando ISPIS para elaborar planos de inspeções de software, quando comparada à elaboração destes planos sem apoio ferramental.

Assim, pretende-se avaliar as seguintes questões gerais:

- Q1: A utilização de ISPIS no planejamento de inspeções implica em uma cobertura de defeitos maior, quando comparada ao planejamento sem apoio ferramental?
- Q2: A utilização de ISPIS no planejamento de inspeções implica em um tempo menor de elaboração de planos, quando comparada ao planejamento sem apoio ferramental?
- Q3: A utilização de ISPIS no planejamento de inspeções implica em um grau de satisfação maior dos usuários na realização das tarefas, quando comparada ao planejamento sem apoio ferramental?

Conforme será esclarecido em maiores detalhes na próxima seção, a caracterização de ISPIS de acordo com a definição acima será feita a partir da coleta e análise de dados quantitativos e qualitativos durante a operação do experimento.

5.2.2 Planejamento do Estudo Experimental

A fase de planejamento do experimento descreve como o experimento deve ser realizado. Esta descrição envolve: (1) a seleção do contexto, (2) a formulação das hipóteses, (3) a seleção das variáveis, (4) a seleção dos indivíduos, (5) o projeto do experimento, (6) a descrição da instrumentação e (7) a avaliação da validade do experimento. A Figura 5.2 ilustra as fases envolvidas no planejamento de experimentos.

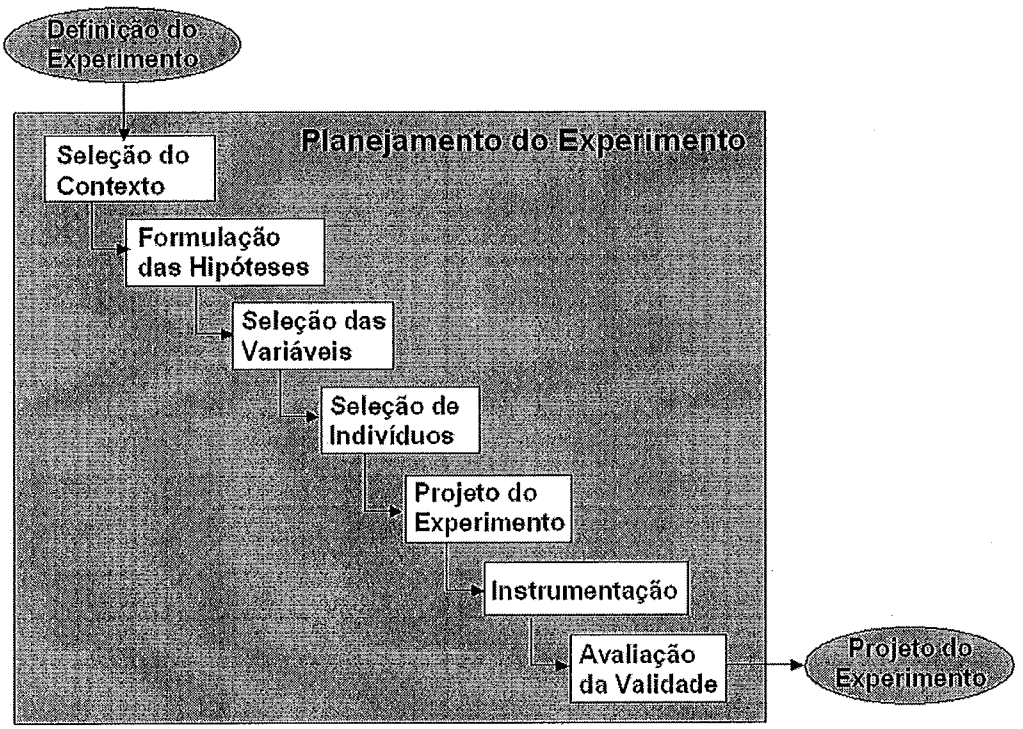


Figura 5.2. Planejamento do Experimento. Adaptado de (WOHLIN et al., 2000).

A realização de cada uma das fases para o estudo experimental que visa caracterizar ISPIS para o planejamento de inspeções se encontra a seguir.

Seleção do Contexto. De acordo com (WOHLIN, 2000), o contexto pode ser caracterizado conforme quatro dimensões:

- processo: *on-line* / *off-line*;
- participantes: alunos / profissionais;
- realidade: problema real / modelado;
- generalidade: específico / geral.

Este estudo se propõe a ser executado de forma *off-line*, uma vez que não haverá um monitoramento contínuo sobre as atividades dos participantes. Os indivíduos utilizados no estudo serão alunos de pós-graduação da COPPE/UFRJ e eles estarão trabalhando com um problema modelado. A generalidade do estudo é específica para este contexto.

Formulação das Hipóteses. Uma hipótese nula e três hipóteses alternativas foram elaboradas para o plano do experimento:

- **Hipótese nula (H_0):** A utilização da infra-estrutura ISPIS para planejar inspeções não traz alterações quando comparada ao planejamento de inspeções sem suporte ferramental.

  $H_0 : (\mu_1 = \mu_2) \wedge (\eta_1 = \eta_2) \wedge (\iota_1 = \iota_2)$, onde:

  - μ_1 é o tempo de elaboração de um plano de inspeção sem utilizar a infra-estrutura.
  - μ_2 é o tempo de elaboração de um plano de inspeção utilizando a infra-estrutura.
  - η_1 é a cobertura de defeitos seguindo o processo de inspeção de acordo com o plano de inspeção produzido sem utilizar a infra-estrutura.
  - η_2 é a cobertura de defeitos seguindo o processo de inspeção de acordo com o plano de inspeção produzido utilizando a infra-estrutura.
  - ι_1 é o grau de satisfação na execução manual das tarefas de planejamento.
  - ι_2 é o grau de satisfação na execução das tarefas com apoio da infra-estrutura.

- **Hipótese alternativa (H_1):** A cobertura de defeitos é maior quando o processo segue um plano elaborado com o uso da infra-estrutura.

  $H_1 : \eta_1 < \eta_2$, onde:

  - η_1 é a cobertura de defeitos seguindo o processo de inspeção de acordo com o plano de inspeção produzido sem utilizar a infra-estrutura.
  - η_2 é a cobertura de defeitos seguindo o processo de inspeção de acordo com o plano de inspeção produzido utilizando a infra-estrutura.

- **Hipótese alternativa (H_2):** O tempo de elaboração de planos para o processo de inspeção é menor quando a infra-estrutura é utilizada.

  $H_2 : \mu_1 > \mu_2$, onde:

  - μ_1 é o tempo de elaboração de um plano de inspeção sem utilizar a infra-estrutura.
  - μ_2 é o tempo de elaboração de um plano de inspeção utilizando a infra-estrutura.

- **Hipótese alternativa (H_3):** O uso da infra-estrutura resulta em um maior grau de satisfação dos usuários no planejamento de inspeções.

  $H_3 : \iota_1 < \iota_2$, onde:

  - ι_1 é o grau de satisfação na execução manual das tarefas.
  - ι_2 é o grau de satisfação na execução das tarefas com apoio da infra-estrutura.

Seleção de Variáveis. O experimento utiliza as seguintes variáveis:

- Variável independente:
  - O plano do processo de inspeção de software.
- Variáveis dependentes:
  - O tempo de elaboração do plano para a inspeção de software.
  - A cobertura de defeitos de uma inspeção seguindo o plano elaborado. Esta variável representa o percentual do total de defeitos encontrados

no artefato durante a inspeção. O valor para esta variável pode ser encontrado de acordo com a fórmula abaixo:
Cobertura = (número de defeitos encontrados) / (número de defeitos do artefato) * 100%

- O grau de satisfação do usuário na execução da tarefa. Esta variável terá seus valores obtidos a partir de análise qualitativa.

Seleção de Indivíduos. Os participantes selecionados devem representar uma população de desenvolvedores de software. Um questionário deve ser entregue a estes participantes para que possam ser caracterizados em termos de sua experiência prática com desenvolvimento de software e sua formação acadêmica.

Os participantes devem então ser divididos em dois grupos utilizando o questionário de caracterização para formar uma divisão de grupos equilibrada em termos de experiência dos participantes com: (1) gerenciamento de projetos de software, (2) planejamento de inspeções de software e (3) aplicação prática de inspeções de software.

Esta divisão equilibrada facilita o uso de agrupamento (*blocking*) (WOHLIN *et al.*, 2000) para eliminar fatores de confusão da análise dos resultados.

Projeto do Experimento. O projeto do estudo experimental envolve um fator: o plano para uma inspeção de software. Dois tratamentos são dados a este fator: não utilizar a infra-estrutura ISPIS para a elaboração do plano de inspeção e utiliza-la. O experimento faz uso de um projeto balanceado, com um grupo para cada tratamento, dividindo os participantes conforme descrito na seção *Seleção de Indivíduos*. Cada tratamento deve ter o mesmo número de participantes e cada um dos participantes utilizará apenas um tratamento em um único objeto.

No experimento, os participantes dos dois tratamentos recebem um problema modelado solicitando a elaboração de um plano para a realização de uma inspeção em um documento de requisitos para um sistema de controle de uma rede de caixa eletrônicos de banco (ATM). Fora o problema modelado, definindo o contexto da inspeção, e o documento de requisitos ATM, os participantes terão acesso aos seguintes dados: (1) dados históricos reais sobre o desempenho de inspetores em inspeções passadas e (2) a caracterização atual dos inspetores disponíveis para a inspeção. A elaboração do plano consiste em definir explicitamente o contexto da

inspeção, selecionar os inspetores para a atividade de detecção individual e distribuir o material para inspeção aos inspetores.

Instrumentação. Para a realização do experimento a instrumentação conta com dados de sete inspetores que realizaram duas detecções de defeitos consecutivas, em diferentes documentos de requisitos, utilizando PBR. Estes dados são de detecções realizadas em um documento de requisitos para um sistema de controle de estacionamento (PGCS) e um documento de requisitos para uma rede de caixas eletrônicos de banco (ATM), respectivamente. Assim, para instrumentar o experimento, a caracterização destes sete inspetores deve ser cadastrada em ISPIS e a detecção de defeitos no documento PGCS utilizada para tornar dados históricos sobre o desempenho dos inspetores disponíveis. Para o tratamento manual a caracterização e a lista de discrepâncias da detecção PGCS (com os defeitos destacados) de cada um dos inspetores devem ser preparados em formato de papel.

Os dados da detecção de defeitos no documento ATM, por sua vez, não são disponibilizados para os participantes e serão utilizados apenas na análise dos resultados.

Além dos dados históricos e da caracterização dos inspetores o experimento conta com os seguintes instrumentos:

- Um formulário de consentimento (Apêndice C.1).
- Um questionário de caracterização (Apêndice C.2).
- Um problema modelado, representando a tarefa a ser desempenhada pelos participantes (Apêndice C.3).
- Um questionário de acompanhamento, para possibilitar uma análise qualitativa (veja Apêndice C.4).

Avaliação da Validade. A validade interna, externa, de construção e de conclusão do experimento se encontra discutida a seguir:

- **Validade interna.** Como se trata de um estudo envolvendo mais de um grupo, onde cada grupo é submetido a um tratamento distinto, a maior ameaça de validade interna é a relação dos resultados com a seleção dos participantes do estudo. Para evitar esta ameaça e aumentar a validade interna, o estudo deverá

utilizar os dados do questionário de caracterização para a divisão dos participantes em grupos de acordo com suas características individuais.

- **Validade externa.** O estudo envolve estudantes e utiliza um problema modelado. Ele se propõe a ser válido dentro deste contexto. Não é possível generalizar os resultados para o contexto empresarial e problemas reais, sendo necessário para isto elaborar novos estudos para ampliar a validade externa.
- **Validade de construção.** Para obter validade de construção precisamos verificar duas coisas: que os tratamentos representem bem a construção da causa e que a saída do experimento represente bem a construção do efeito. Como o experimento possui apenas um fator e dois tratamentos, a causa pode ser facilmente mapeada no uso ou não da infra-estrutura. A saída por sua vez está ligada diretamente ao efeito, uma vez que ambos podem ser extraídos a partir dos valores das variáveis dependentes.
- **Validade de conclusão.** O teste estatístico *t-test* deve ser aplicado para comparar os valores obtidos para cada uma das variáveis dependentes obtidas através de dados quantitativos, utilizando os dois tratamentos: com uso de suporte ferramental fornecido por ISPIS e sem o uso. Maiores detalhes sobre *t-test* e seu potencial estatístico podem ser encontrados em (MARASCUILO e SERLIN, 1988). As hipóteses devem ser verificadas aplicando, para cada uma delas, o critério do *t-test* com nível de significância 10%.

5.2.3 Operação

O experimento foi conduzido utilizando 12 participantes voluntários, representados por 10 alunos de mestrado e doutorado, um ex-aluno de mestrado e um de doutorado, todos da COPPE/UFRJ.

Inicialmente os participantes assinaram um formulário de consentimento de participação no experimento e foram solicitados a preencher um questionário de caracterização. Este questionário foi utilizado para dividir os participantes em dois grupos balanceados de acordo com sua experiência (veja seção *Seleção dos Indivíduos* para maiores detalhes). Todos os participantes relataram ter experiência prática com desenvolvimento de software. Um resumo da experiência dos participantes do estudo se encontra na Tabela 5.1.

Identificação	Experiência Prática Desenvolvendo Software	Nível
Subject 01	Quatro anos.	D.Sc.
Subject 02	Quinze anos.	D.Sc.
Subject 03	Três anos.	M.Sc.
Subject 04	Três anos.	D.Sc.
Subject 05	Dois anos.	D.Sc.
Subject 06	Dois anos.	M.Sc.
Subject 07	Quatro anos.	D.Sc.
Subject 08	Quatorze anos.	D.Sc.
Subject 09	Quatorze anos (Dez anos programador e quatro gerente).	D.Sc. Concluído
Subject 10	Quatorze anos.	D.Sc.
Subject 11	Cinco anos.	M.Sc. Concluído
Subject 12	Seis anos.	M.Sc.

Tabela 5.1. Experiência prática com desenvolvimento de software dos participantes.

Em seguida, o problema modelado e o documento de requisitos do sistema ATM foram entregues aos participantes. Além disto, os participantes do tratamento manual receberam os dados históricos sobre o desempenho dos inspetores na inspeção PGCS e a caracterização destes inspetores em formato impresso. Os participantes do tratamento utilizando ISPIS, por sua vez, receberam um *link* para a ferramenta e uma conta com usuário e senha.

Enquanto elaboravam seus planos para a inspeção os participantes não sabiam que dados sobre o desempenho dos inspetores na detecção de defeitos no documento de requisitos ATM já estavam disponíveis. Portanto, eles não tinham como prever que dados, como a cobertura de defeitos de uma inspeção seguindo o plano que eles estavam elaborando podiam ser medidos.

Após o procedimento do experimento, dados qualitativos foram coletados utilizando um questionário de acompanhamento.

5.2.4 Análise e Interpretação dos Dados

A Tabela 5.2 apresenta os valores obtidos para as variáveis 'tempo' e 'cobertura de defeitos' de cada um dos participantes que utilizaram o tratamento com ISPIS. Os resultados dos participantes do tratamento manual se encontram na Tabela 5.3. Os resultados marcados como *outlier* são dados fora da distribuição normal. Nesta análise consideramos como *outlier* todos os resultados mais afastados da média do que o

desvio padrão. As tabelas apresentam ainda a média, o desvio padrão e a média desconsiderando os *outliers* para as duas variáveis.

Participantes do Tratamento com ISPIS	Tempo	Cobertura de Defeitos %
Subject 01	20,00	26,83 (outlier)
Subject 02	25,00	46,34 (outlier)
Subject 03	9,00 (outlier)	31,71
Subject 04	25,00	43,90
Subject 05	27,00	43,90
Subject 06	13,00	31,71
Média	19,83	37,40
Desvio Padrão	7,33	8,26
Média sem *outliers*	22,00	37,80

Tabela 5.2. Resultados para as variáveis 'tempo' e 'cobertura de defeitos' dos participantes do tratamento com ISPIS.

Participantes do Tratamento Manual	Tempo	Cobertura de Defeitos %
Subject 07	30,00	31,71
Subject 08	30,00	29,27
Subject 09	25,00	46,34 (outlier)
Subject 10	25,00	31,71
Subject 11	20,00(outlier)	41,46
Subject 12	31,00	31,71
Média	26,83	35,37
Desvio Padrão	4,26	6,85
Média sem *outliers*	28,20	33,17

Tabela 5.3. Resultados para as variáveis 'tempo' e 'cobertura de defeitos' dos participantes do tratamento manual.

Analisando os resultados preliminares, as seguintes observações podem ser feitas a respeito das hipóteses apresentadas no planejamento do experimento:

- **Hipótese alternativa (H_1): A cobertura de defeitos é maior quando o processo segue um plano elaborado com o uso da infra-estrutura.** Aplicando t-test desconsiderando os outliers obtemos t_0 = 1,27, da tabela de valores críticos obtemos valor de *t* para nível de significância 10% e 7 graus de liberdade $t_{0.05,7}$ = 1,89. Como t_0 não é maior do que $t_{0.05,7}$ e em relação à cobertura de defeitos a hipótese nula não pôde ser refutada através da aplicação

do *t-test*. No entanto, a cobertura média foi 14% maior nos planos de inspeção elaborados pelos participantes do tratamento que utiliza ISPIS. Os participantes mais experientes tiveram bom desempenho em ambos os tratamentos. O resultado máximo de cobertura (cobertura de 46,34%), por exemplo, foi obtido pelos dois participantes mais experientes do estudo. Ambos professores universitários com ampla experiência em desenvolvimento de software. Conforme mencionado em (KALINOWSKI e TRAVASSOS, 2004), acreditamos que os participantes com menos experiência foram auxiliados pelo conhecimento de (CARVER, 2003) embutido em ISPIS. Ressaltamos que o estudo precisa ser repetido utilizando conjuntos de dados maiores, para que conclusões mais precisas a respeito da cobertura de defeitos possam ser tiradas.

- **Hipótese alternativa (H_2): O tempo de elaboração de planos para o processo de inspeção é menor quando a infra-estrutura é utilizada.** Aplicando t-test desconsiderando os outliers obtemos $t_0 = 8,17$, da tabela de valores críticos obtemos o valor de *t* para nível de significância 10% e 8 graus de liberdade $t_{0.05,8} = 1,86$. Como $t_0 > t_{0.05,8}$ a hipótese nula foi refutada. Todos os participantes realizaram o planejamento em menos de 40 minutos. Assim, o tempo gasto na atividade planejamento parece não representar significância em relação à eficiência da inspeção como um todo. Mesmo assim, vale ressaltar que os participantes do tratamento que utiliza ISPIS gastaram 22% menos tempo na realização de suas tarefas.
- **Hipótese alternativa (H_3): O uso da infra-estrutura resulta em um maior grau de satisfação dos usuários no planejamento de inspeções.** Todos os participantes informaram se sentir ao menos satisfeitos na realização da tarefa. No entanto, os participantes que utilizaram ISPIS relataram um grau de satisfação muito alto, enquanto os participantes do tratamento manual relataram apenas um grau de satisfação alto. O principal problema associado ao grau de satisfação dos participantes do tratamento manual foi lidar com os dados históricos em formato de papel (*paper shuffling*), o que não ocorre em ISPIS.

Apesar dos bons resultados preliminares, acreditamos que mais participantes precisam ser utilizados para confirmar estes resultados, principalmente para avaliar melhor a questão da cobertura de defeitos. Além disto, o plano do experimento deve ser

discutido com outros pesquisadores para que assim se possa obter melhorias e para que futuras conclusões possam ser tiradas.

5.3 – Estudo de Caso: Avaliando a Viabilidade da Utilização de ISPIS em Inspeções Reais

O estudo de caso foi conduzido com 12 participantes, representando um professor e 11 alunos de mestrado e doutorado da COPPE/UFRJ. Todos os participantes relataram ter experiência prática com desenvolvimento de software. Antes do início do estudo de caso os participantes assinaram um formulário de consenso de participação no estudo. Uma demonstração da ferramenta foi dada aos participantes em sala de aula.

Cinco dos participantes tinham experiência com inspeções de documentos de requisitos utilizando a técnica PBR, uma vez que participaram de um estudo anterior para analisar apoio ferramental para PBR (SILVA e TRAVASSOS, 2004). O professor desempenhou o papel do moderador da inspeção. Um dos demais participantes desempenhou o papel do autor do documento, uma vez que o documento a ser inspecionado seria o documento de requisitos para a implementação de sua tese de mestrado. Trata-se de um documento de requisitos real, para a implementação de uma ferramenta de apoio às OORT's. Os 10 participantes restantes foram agrupados em duas equipes de inspetores, sendo uma equipe formada pelos 5 participantes com experiência em PBR. A outra equipe de inspetores foi formada por 5 participantes sem experiência anterior em inspeções de software.

Para avaliar a viabilidade de utilizar a infra-estrutura ISPIS para apoiar inspeções com e sem utilizar técnicas de inspeção específicas, o moderador foi solicitado a gerenciar duas inspeções no documento de requisitos. Uma destas inspeções utilizando PBR como técnica de detecção de defeitos e a outra utilizando *ad-hoc*. Para isto, a equipe de inspetores experiente com PBR e a não experiente foram utilizadas, respectivamente. Uma vez que o moderador planejou as duas inspeções estas foram encaminhadas pela infra-estrutura para as atividades de detecção de defeitos.

Na inspeção utilizando PBR, ISPIS forneceu uma ferramenta externa para apoiar a aplicação desta técnica na detecção de defeitos. A ferramenta externa disponibilizada foi a descrita em (SILVA e TRAVASSOS, 2004). Três dos 5 inspetores efetivamente

utilizaram a ferramenta externa, que produz um documento XML descrevendo uma lista de discrepâncias. Este documento pôde então ser carregado e ativado em ISPIS. Os outros 2 participantes da inspeção PBR aplicaram a técnica manualmente e registraram suas listas de discrepância diretamente em ISPIS. Na inspeção utilizando *ad-hoc*, por sua vez, todos os inspetores registraram suas discrepâncias diretamente em ISPIS, uma vez que nenhum apoio ferramental foi disponibilizado.

O moderador então realizou a atividade de coleção de defeitos para cada uma das duas inspeções paralelas e disponibilizou as listas de discrepância para a atividade de discriminação de defeitos.

Na discriminação de defeitos, o moderador, autor do documento e os inspetores discutiram a classificação correta de cada uma das discrepâncias. A Figura 5.3 mostra parte da captura de uma tela com a discussão de uma das discrepâncias. A tela em questão é a visão do moderador e está em inglês devido a um pedido deste para ter inglês cadastrado como sua língua nativa, para que durante o estudo ele pudesse realizar uma revisão dos termos empregados na versão ISPIS em inglês.

Por fim, as atividades de retrabalho e de continuação foram realizadas pelo autor e pelo moderador, respectivamente. Na atividade de continuação o moderador decidiu não realizar uma nova inspeção.

Ambos os tratamentos alcançaram com sucesso o fim do processo de inspeção de software e os dados qualitativos mostraram a satisfação dos usuários com o uso da infra-estrutura. No entanto, algumas sugestões de facilidades de usabilidade para tratar de listas de discrepâncias com um número grande de discrepâncias foram feitas. No estudo de caso as listas de discrepância de ambos os tratamentos tinham mais de 90 discrepâncias e algumas facilidades para lidar com listas de discrepâncias grandes não tinham sido consideradas até então. Estas sugestões foram implementadas em ISPIS e uma nova versão foi gerada. Basicamente estas mudanças foram:

- Reposicionamento da barra de rolagem após a realização das operações sobre discrepâncias.
- Marcação para cada usuário das discrepâncias de cujas discussões ele está efetivamente participando. Assim, agora as discrepâncias já comentadas pelo usuário aparecem com suas caixas de seleção destacadas.

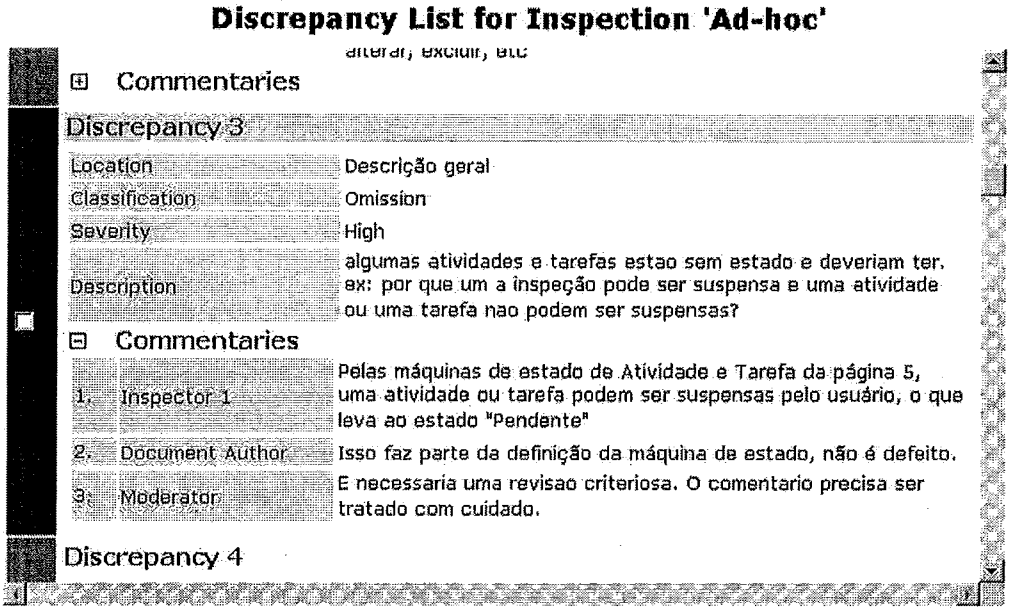


Figura 5.3. Uma discriminação real ocorrendo em ISPIS.

5.4 – Conclusões

Uma das formas de se avaliar propostas de métodos, ferramentas e processos na engenharia de software é fazendo uso da experimentação. A literatura descreve três estratégias experimentais: *survey*, estudo de caso e experimento. A escolha da estratégia experimental depende do contexto e dos objetivos do estudo.

O plano de um estudo experimental para avaliar o apoio fornecido pela infra-estrutura ISPIS para o planejamento de inspeções de software foi elaborado. Os resultados preliminares da condução deste experimento são satisfatórios e dão indícios de que o uso da infra-estrutura, quando comparada ao planejamento manual, pode fazer com que planos para inspeções obtenham uma maior cobertura de defeitos, sejam elaborados em menos tempo e com um grau de satisfação maior dos usuários.

Além disto, um estudo de caso mostrou a viabilidade de se utilizar ISPIS em inspeções reais. Por este motivo acreditamos que a disponibilização da infra-estrutura permite a sua utilização por organizações e parceiros de pesquisa. Desta forma, organizações podem realizar suas inspeções de maneira mais sistematizada e usufruir os benefícios da aplicação de conhecimento experimentalmente avaliado e da coleta e uso de dados históricos.

Capítulo 6
Considerações Finais

Este capítulo descreve as contribuições desta tese e alguns de seus possíveis trabalhos futuros.

6.1 – Visão Geral

Há um crescente interesse em estudos experimentais na área de engenharia de software. Este fato fica evidenciado pelo crescente número de publicações contendo métodos experimentais (ZELKOWITZ e WALLACE, 1998). A experimentação tem sido utilizada para acelerar o progresso através da rápida eliminação de abordagens não fundamentadas, suposições errôneas e modismos, ajudando a orientar a engenharia e a teoria para direções promissoras. Realçando a importância da realização de experimentos na engenharia de software SHULL et al., (2001) descrevem uma metodologia para a transferência de tecnologias para a indústria baseada em métodos experimentais.

Este trabalho representa a concretização de uma proposta de apoio ao processo de inspeção de software pautada nestes aspectos, tendo seu conjunto básico de requisitos básicos derivado de resultados de estudos experimentais da literatura. Tendo em vista o estado atual da prática de inspeções de software e as limitações das ferramentas de apoio existentes, duas expectativas principais foram criadas para que esta infra-estrutura pudesse efetivamente trazer benefícios para a aplicação de inspeções de software. A primeira destas expectativas foi que sua disponibilização pudesse possibilitar a realização de inspeções de forma mais sistematizada por organizações na prática. A segunda expectativa era fornecer um apoio adequado para a tomada de decisões, induzindo decisões corretas ao longo do processo de inspeção de software.

Visando verificar se estas expectativas foram alcançadas a infra-estrutura computacional ISPIS, resultante do trabalho, foi avaliada utilizando métodos

experimentais, mais especificamente, um estudo experimental e um estudo de caso foram conduzidos.

O estudo experimental avaliou o apoio de ISPIS ao planejamento de inspeções de software, que utiliza uma proposta de apoio nova, ainda não avaliada. Os resultados do experimento mostraram benefícios no planejamento com ISPIS, quando comparado ao planejamento manual de inspeções.

O estudo de caso, por sua vez, mostrou a viabilidade de se utilizar ISPIS em inspeções reais. Assim, acreditamos que esta infra-estrutura fornece apoio adequado a inspeções de software, permitindo sua realização sistematizada e eficaz na prática.

6.2 – Contribuições

Nesta seção algumas contribuições identificadas neste trabalho de pesquisa são descritas.

Revisão da literatura. O Capítulo 2 descreve conceitos relacionados a inspeções de software, benefícios de sua aplicação e fornece um cenário do estado atual da prática e do apoio ferramental existente. Esta revisão pode ser utilizada como base para fornecer um entendimento inicial a respeito de inspeções de software;

Elaboração de uma proposta de infra-estrutura para apoio ao processo de inspeção de software. No Capítulo 3 esta proposta, pautada em conhecimento sobre inspeções de software proveniente de estudos experimentais, é descrita em detalhes;

Implementação desta proposta. A implementação da proposta resultou em ISPIS (Capítulo 4), uma infra-estrutura que:

- Pode ser utilizada para a inspeção de todos os artefatos produzidos ao longo do processo de desenvolvimento de software;
- possibilita a integração de ferramentas que dêem apoio à aplicação de técnicas de detecção de defeitos específicas;
- pode ser utilizada por equipes geograficamente distribuídas;
- possui seu conjunto de requisitos de apoio baseado no conhecimento sobre inspeções de software proveniente de estudos experimentais identificado na proposta do Capítulo 3;

- efetivamente utiliza dados históricos sobre inspeções passadas.

A Tabela 6.1 compara as características básicas das demais ferramentas descritas nesta tese com as de ISPIS.

Ferramenta	Processo	Tipo de Reunião	Técnicas	Artefatos	Equipes Distribuídas
ICICLE	Processo Tradicional (Fagan, 1976)	Síncrona	Técnica Própria	Código C e C++	Não
InspeQ	Processo Próprio	Síncrona	Inspeção por Fases (com checklists específicas)	Artefatos descritos textualmente	Não
CSI	Processo de Humphrey (HUMPHREY, 1989)	Síncrona	Ad-hoc	Artefatos descritos textualmente	Não
CSRS	FTArm (JOHNSON, 1994)	Assíncrona	Ad-hoc e *checklists*	Artefatos descritos textualmente	Sim
Scrutiny	Processo da BULL HB Information Systems, variante do processo tradicional.	Síncrona	Ad-hoc	Artefatos descritos textualmente	Sim
WIT	Processo Próprio	Assíncrona	Ad-hoc e *checklists*	Artefatos descritos em HTML	Sim
GRIP	Processo Próprio	Assíncrona	Ad-hoc	Todos os tipos de artefatos	Sim
IBIS	Sauer et al. (2000)	Assíncrona	Ad-hoc e checklists	Todos os tipos de artefatos	Sim
ISPIS	Sauer et al. (2000)	Assíncrona	Ad-hoc e demais técnicas através da integração de ferramentas externas.	Todos os tipos de artefatos	Sim

Tabela 6.1. Características básicas de algumas ferramentas de apoio a inspeções.

A principal diferença em relação a estas características está na técnica de detecção de defeitos, onde ISPIS permite a integração de ferramentas externas, possibilitando fornecer apoio à aplicação de diferentes técnicas de detecção de defeitos. Desta forma, a infra-estrutura pode ser estendida para apoiar da forma customizada a inspeção dos diferentes artefatos produzidos ao longo de processo de desenvolvimento.

Um outro diferencial é que ISPIS contempla o conhecimento experimentalmente avaliado descrito em (ADAMS, 1999) (BIFFL *et al.*, 2003) (CARVER, 2003) (VITHARANA e RAMAMURTHY, 2003), que acreditamos não ter sido utilizado em nenhuma abordagem. Além disto, acreditamos que o conhecimento obtido no estudo experimental (LANUBILE e MALLARDO, 2003) esteja sendo efetivamente utilizado apenas em IBIS (LANUBILE e MALLARDO, 2002) e ISPIS.

Disponibilização de ISPIS. A infra-estrutura se encontra disponível na COPPE/UFRJ. Assim, acesso a ela para parceiros de pesquisa e organizações pode ser proporcionado.

Avaliação de ISPIS. Um passo em direção à avaliação da infra-estrutura foi dado com a realização de um estudo experimental e um estudo de caso (Capítulo 5).

6.3 – Trabalhos Futuros

Entre os trabalhos futuros sugeridos no intuito de dar continuidade a esta pesquisa podemos citar:

- Implementação de um número maior das hipóteses fundamentadas de CARVER (2003) para assim fornecer um apoio mais completo ao planejamento de inspeções, utilizando mais conhecimento. O estudo experimental descrito nesta tese poderia então ser repetido para verificar se a adição de conhecimento aumenta os benefícios do planejamento;
- Permitir uma caracterização flexível dos inspetores, de modo que informações consideradas importantes para determinadas organizações pudessem ser acrescentadas. Isto poderia ser feito com uma adaptação que tornasse possível cadastrar as características a serem consideradas na caracterização dos inspetores para cada organização;
- A adaptação descrita no tópico anterior poderia ainda permitir que combinações entre as características da inspeção e dos inspetores benéficas para a inspeção pudessem ser cadastradas e assim fazer com que as hipóteses fundamentadas de CARVER (2003) sejam configuradas dinamicamente. Assim, cada

organização poderia fazer uso de ISPIS com um conjunto diferente de hipóteses fundamentadas ou mesmo com conhecimento interno da organização;

- Implementação de recursos de votação para capturar de forma objetiva a opinião dos participantes durante a discriminação de defeitos. Um apoio que se mostrou adequado para a classificação das discrepâncias que de acordo com um estudo experimental, relatado em (GRÜNBACHER *et al.*, 2003), é o implementado na ferramenta GRIP (HALLING *et al.*, 2001). Em GRIP o esforço da discriminação é reduzido e a classificação correta de discrepâncias é apoiada. Isto é feito através da utilização de *tresholds* para a classificação automática de defeitos baseado nas opiniões dos participantes da discriminação;
- A realização de novos estudos experimentais utilizando ISPIS. Esses estudos poderiam, por exemplo, questionar a estrutura do processo proposto por SAUER *et al.* (2000), comparando sua eficiência em inspeções reais com outras variantes do processo de inspeção;
- Integração de ISPIS a uma infra-estrutura de experimentação, de modo que experimentos sobre inspeções pudessem ser realizados utilizando a infra-estrutura na atividade de operação do experimento;
- Implementação de novas ferramentas externas apoiando a aplicação de outras técnicas de leitura em ISPIS, como por exemplo as *checklists*. Esta atividade envolve a construção das ferramentas e a geração de um conversor para integrar a ferramenta a ISPIS. Este conversor pode ser gerado utilizando a ferramenta xMapper (SPINOLA et al., 2004).

Referências

ACKERMAN, A., BUCHWALD, L., LEWSKI, F., 1989, "Software Inspections: An Effective Verification Process", IEEE Software, vol. 6, no. 3, pp.31-37.

ADAMS, T., 1999, "A formula for the re-inspection decision", Software Engineering Notes 24(3): 80.

BASILI, V.R., SELBY, R.W., 1987, "Comparing the Effectiveness of Software Testing Strategies", IEEE Transactions on Software Engineering 13 (12): 1278-1296.

BASILI, V., CALDERA, C., ROMBACH, H., 1994, "Goal Question Metric Paradigm", Encyclopaedia of Software Engineering (Marciniak J. editor), vol. 1, John Wiley & Sons, 528-532.

BASILI, V.R., GREEN, S., LAITENBERGER, O., LANUBILE, F., SHULL, F., SORUMGARD, S., ZELKOWITZ, M.V., 1996, "The empirical investigation of perspective based reading", Empirical Software Engineering Journal vol. 1, no. 2.

BIFFL, S., GROSSMANN, W., 2001, "Evaluating the accuracy of objective estimation models based on inspection data from multiple inspection cycles", Proc. ACM/IEEE ICSE, Toronto, Canada, IEEE Comp. Soc. Press.

BIFFL, S., GUTJAHR, W., 2002, "Using a Reliability Growth Model to Control Software Inspection", Empirical Software Engineering: An international journal; vol.7, pp. 257-284, Kluwer Academic Publishers.

BIFFL, S., HALLING, M., KOESZEGI, S., 2003, "Investigating the Accuracy of Defect Estimation Models for Individuals and Teams Based on Inspection Data", Proceedings of the 2nd International Symposium on Empirical Software Eng. (ISESE 2003), Roman Castles (Rome), Italy, IEEE Computer Society Press.

BOEHM, B.W., 1981, *Software Engineering Economics*, Prentice Hall.

BOEHM, B.W., ABTS, C., BROWN, A.W., CHULANI, S., CLARK, B.K., HOROWITZ, E., MADACHY, R., REIFER, D., STEECE, B., 2000, *Software Cost Estimation with COCOMO II*, Prentice Hall.

BOEHM, B. W., BASILI, V.R., 2001, "Software Defect Reduction Top 10 List.", IEEE Computer 34 (1): 135-137.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., 1998, *The Unified Modeling Language User Guide,* 1 ed., Addison-Wesley.

BROTHERS, L., SEMBUGAMOORTHY, V., MULLER, M., 1990, "ICICLE: Groupware for Code Inspection", In: Proceedings of the ACM conference on computer-supported cooperative work (CSCW'90), Los Angeles, California, United States.

CARVER, J., 2003, "The Impact of Background and Experience on Software Inspections", PhD Thesis, University of Maryland, USA.

CHAFFEY, D., 1998, *Groupware, workflow and Intranets: Reengineering the Enterprise with Collaborative Software*, 1ed. Butterworth-Heinemann.

CIOLKOWSKI, M., LAITENBERGER, O., BIFFL, S., 2003, "Software Reviews: The State of the Practice", IEEE Software 20 (6): 46-51.

CMMI PRODUCT TEAM, 2001, "Capability Maturity Model Integration (CMMI), Version 1.1", acessado em 10/02/2004, disponível em http://www.sei.cmu.edu/publications/documents/02.reports/02tr029.html.

CONALLEN, J., 1999, "Building Web Applications with UML", 1ed. Addison-Wesley.

CONRADI, R., MARJARA, A., HANTHO, O., FROTVEIT, T., SKATOVIC, B., 1999, "A study of inspections and testing at Ericsson", Proceedings of the International Conference on Product Focused Software Process Improvement (PROFES'99), Oulu, Finland.

CONRADI, R., MOHAGHEGHI, M., ARIF, T., HEDGE, L.C., BUNDE, G.A., PEDERSEN, A., 2003, "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, Springer LNCS No. 2743, pp. 483-501.

FAGAN, M.E., 1976, "Design and Code Inspection to Reduce Errors in Program Development", IBM Systems Journal, vol. 15, no. 3, pp. 182-211.

GILB, T., GRAHAM, D., 1993, "Software Inspection", Addison-Wesley.

GINTELL, J.W., HOUDE, M.B., MCKENNEY, R.F., 1995, "Lessions learned by building and using Scrutiny, a collaborative software inspection system", Proceedings of the Seventh International Workshop on Computer-Aided Software Engineering.

GRÜNBACHER, P., HALLING, M., BIFFL, S., 2003, "An Empirical Study on Groupware Support for Software Inspection Meetings", Proceedings of the 18th IEEE International Conference on Automated Software Engineering.

HALLING, M., GRÜNBACHER, P., BIFFL, S., 2001, "Tailoring a COTS Group Support System for Software Requirements Inspection", In: 9th IEEE International Conference on Automated Software Engineering, San Diego, USA.

HALLING, M., BIFFL, S., GRÜNBACHER, P., 2002, "A Groupware-Supported Inspection Process for Active Inspection Management", EUROMICRO 2002, pp. 251-258, Dortmund Germany: IEEE CS.

HARJUMAA, L., TERVONEN, I., 1998, "A WWW-based tool for software inspection", Proceedings of the 31st HICSS conference, vol III, 1998, pp. 379-388.

IEEE, 1990, "IEEE Standard Glossary of Software Engineering Terminology, Standard 610", IEEE Press.

IEEE, 1998, "IEEE Recommended Practice for Software Requirements Specifications - Description, Standard 830", IEEE Press.

JOHNSON, P. M., TJAHJONO, D., WAN, D., BREWER, R., 1993, "Experiences with CSRS: An Instrumented Software Review Environment", In Proceedings of the 11th Annual Pacific Northwest Software Quality Conference, Portland, USA.

JOHNSON, P., 1994, "An Instrumented Approach to Improving Software Quality Through Formal Technical Review," in Proc. 16th International Conference on Software Engineering, Sorrento, Italy.

JOHNSON, P.M., TJAHJONO, D., 1997, "Assessing Software Review Meetings: A Controlled Experimental Study Using CSRS", Proceedings of ICSE 97, Boston, USA.

JOHNSON, P.M., TJAHJONO, D., 1998, "Does Every Inspection Really Need A Meeting?", Journal of Empirical Software Engineering, Volume 4, Number 1, pages 9-35, January.

JOHNSON, P.M., 1998, "Reengineering Inspection", Communications of the ACM, vol. 41, no 2.

JONES, C., 1991, "Applied Software Measurement", McGraw Hill.

KALINOWSKI, M., 2001, "PatternFlow: um software de workflow para a implementação e execução de processos de negócio customizados", Projeto Final do Curso de Ciência da Computação, DCC/UFRJ.

KALINOWSKI, M., SPINOLA, R.O., TRAVASSOS, G.H., 2004, "Infra-estrutura Computacional para Apoio ao Processo de Inspeção de Software", Simpósio Brasileiro em Qualidade de Software (SBQS 2004), Brasília.

KALINOWSKI, M., TRAVASSOS, G.H., 2004, "A Computational Framework for Supporting Software Inspections", 19th IEEE International Conference on Automated Software Engineering (ASE 2004), Linz, Austria.

KNIGHT, J.C., MEYERS, E.A., 1991, "Phased Inspections and their Implementation", Software Engineering Notes, Vol.16, No.3, pp. 29-35.

LAITENBERGER, O., ATKINSON, C., 1999, "Generalized Perspective Based Inspection to handle Object Oriented Development Artifacts", Proceedings of ICSE 99, p.494-503, Los Angeles, CA, USA.

LANUBILE, F., MALLARDO, T., 2002, "Tool support for distributed inspection", Proc. of the 26th Annual International Computer Software & Applications Conference (COMPSAC 2002), Oxford, England, August 2002.

LANUBILE, F. MALLARDO, T., 2003, "An Empirical Study of Web-Based Inspection Meetings", Proc. of the 2nd International Symposium on Empirical Software Eng. (ISESE 2003), Roman Castles (Rome), Italy, IEEE Computer Society Press, September/October.

MACDONALD, F., MILLER, J., BROOKS, A., ROPER, M., WOOD, M., 1995, "A review of tool support for software inspection", Proceedings of the Seventh International Workshop on Computer-Aided Software Engineering.

MACDONALD, F., MILLER, J., 1999, "A Comparison of Computer Support Systems for Software Inspection", Automated Software Engineering, 1999. 6, pp. 291-313.

MANGAN, M.A.S., ARAÚJO, R.M., KALINOWSKI, M., BORGES, M.R.S., WERNER, C.M.L., 2002, "Towards the evaluation of awareness information support applied to peer reviews of software engineering diagrams", 7th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD'2002), Rio de Janeiro, September 2002, pp. 49-54.

MARASCUILO, L.A., SERLIN, R.C., 1988, "Statistical methods for the social and behavioral sciences", W.H. Freeman (New York).

MARTIN, J., MCCLURE, C., 1985, *Diagramming Techniques for Analysts and Programmers*, Prentice Hall, USA.

MASHAYEKHI, V., DRAKE, J.M., TSAI, W.T., RIEDL, J., 1993, "Distributed, collaborative software inspection", IEEE Software, Volume10 Issue 5, Sept. 1993.

MELO, W., TRAVASSOS, G.H., SHULL, F., 2001, "Software Review Guidelines", Technical Report ES – 556/01 – COPPE/UFRJ, August 2001.

MEYER, E., 2004, *Cascading Style Sheets: The Definitive Guide*, 2nd Edition, O'Reilly.

PINELLE, D., GUTWIN, C., 2000, "A Review of Groupware Evaluations". In: Proceedings of the 9th IEEE WETICE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Gaithersburg, Maryland, USA.

PORTER, A.A., and JOHNSON, P.M., 1997, "Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies." IEEE Transactions on Software Engineering. Vol. 23, no. 3, pp. 129-145.

PRESSMAN, R.S., 2001, *Software Engineering: A Practitioner's Approach*, Fifth Edition, McGraw Hill.

REIS, L.N.M, TRAVASSOS, G.H., 2003, "Apoio Automatizado à Configuração e Aplicação de OORT's", Anais do WTES 2003 (SBES) - Workshop de Teses em Engenharia de Software, p.59-64, Manaus.

SAUER, C., JEFFERY, D.R., LAND, L., YETTON, P., 2000, "The Effecticveness of Software Development Technical Review: A Behaviorally Motivated Program of Research", IEEE Transactions on Software Engineering, 26 (1): 1-14, January.

SHULL, F., 1998, "Developing Techniques for Using Software Documents: A Series of Empirical Studies", Ph.D. thesis, University of Maryland, College Park.

SHULL, F., TRAVASSOS, G.H., CARVER, J., BASILI, V.R., 1999, "Evolving a Set of Techniques for OO Inspections." University of Maryland Technical Report CS-TR-4070.

SHULL, F., RUS, I., BASILI, V., 2000, "How Perspective-Based Reading Can Improve Requirements Inspections", July, IEEE Software, pp. 73-79.

SHULL, F., CARVER, J., TRAVASSOS, G.H., 2001, "An Empirical Methodology for Introducing Software Processes." In Proceedings of the Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9), Vienna, Austria, p. 288-296.

SHULL, F., CARVER, J., TRAVASSOS, G.H., MALDONADO, J.C., CONRADI, R., BASILI, V., 2003, "Replicated Studies: Building a Body of Knowledge about Software Reading Techniques." To Appear in Volume on Software Engineering Empirical Methods, Eds. Natalie Juristo and Ana Moreno.

SILVA, L.F.S, TRAVASSOS, G.H., 2004, "Tool-Supported Unobtrusive Evaluation of Software Engineering Process Conformance", Proceedings of the International Symposium on Empirical Software Engineering, ISESE, California, USA.

SPINOLA, R.O, KALINOWSKI, M., TRAVASSOS, G.H., 2004, "Um Mecanismo para a Integração de Ferramentas CASE", Simpósio Brasileiro em Engenharia de Software (SBES) , Brasília, DF. (submetido).

TERVONEN, I., HARJUMAA, L., IISAKKA, J., 1999, "The Web generation of software inspection: a process with virtual meetings and on-line recording", Software Technology and Engineering Practice, STEP'99, Proceedings.

TRAVASSOS, G. H., SHULL, F., FREDERICKS, M., BASILI, V. R., 1999, "Detecting Defects in Object Oriented Designs: Using Reading Techniques to increase Software Quality", ACM Sigplan Notices. Estados Unidos, v.34, n.10, p.47 - 56.

TRAVASSOS, G.H., SHULL, F., CARVER, J., 2001, "Working with UML: A Software Design Process Based on Inspections for the Unified Modeling Language", in: Advances in Computers, vol. 54, Academic Press.

TRAVASSOS, G.H., GUROV, D., AMARAL, E.A.G.G., 2002, "Introdução à Engenharia de Software Experimental", Relatório Técnico ES – 590/02 – COPPE/UFRJ, Abril 2002.

VAN DER AALST, W.M.P, BARROS, A.P., HOFSTEDE, A.H.M., KIEPUSZEWSKI, B., 2000, "Advanced Workflow Patterns", Proceedings Fifth International Conference on Cooperative Information Systems, CoopIS'00, Eilat, Israel.

VITHARANA, P., RAMAMURTHY, K., 2003, "Computer Mediated Group Support, Anonymity, and the Software Inspection Process: An Empirical Investigation", IEEE Transactions on Software Engineering, vol. 29, number 2.

VOTTA, L.G., 1993, "Does Every Inspectin Need a Meeting?", Proceedings of the first ACM Symposium on Foundations of Software Engineering, pp. 107-114.

WARIA, 2004, "Workflow and Reengineering International Association", http://www.waria.com, acesso em 13/05/2004.

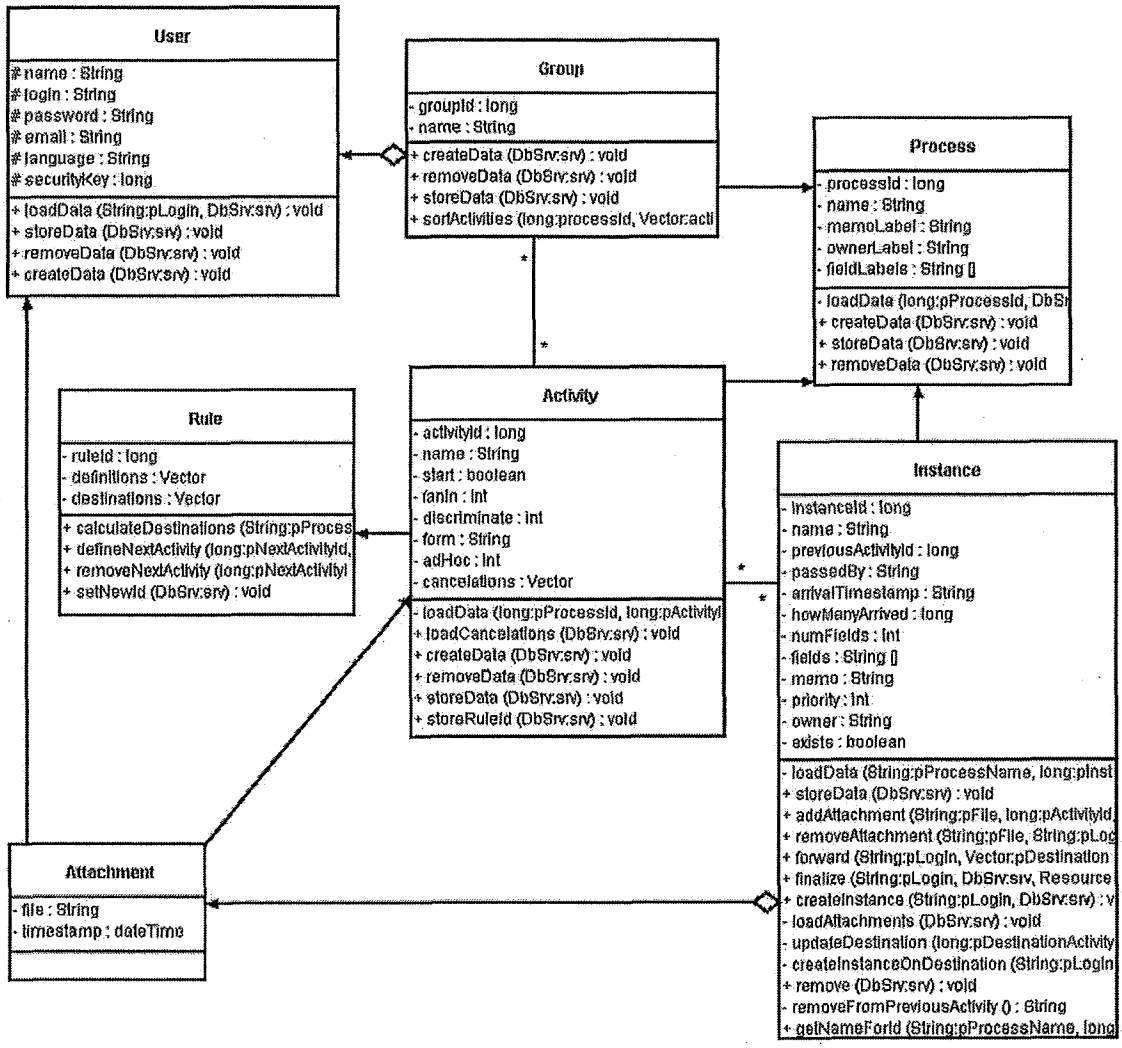
WHEELER, D.A., BRYKEZYNSKI, B., MEESON, R.N., 1996, *Software Inspections: An Industry Best Practice*, IEEE Computer Society.

ZELKOWITZ, M. V., WALLACE, D., 1998, "Experimental models for validating computer technology", IEEE Computer 31, 5 (May, 1998), pp. 23-31.

Apêndice A – Diagramas das Classes de Domínio

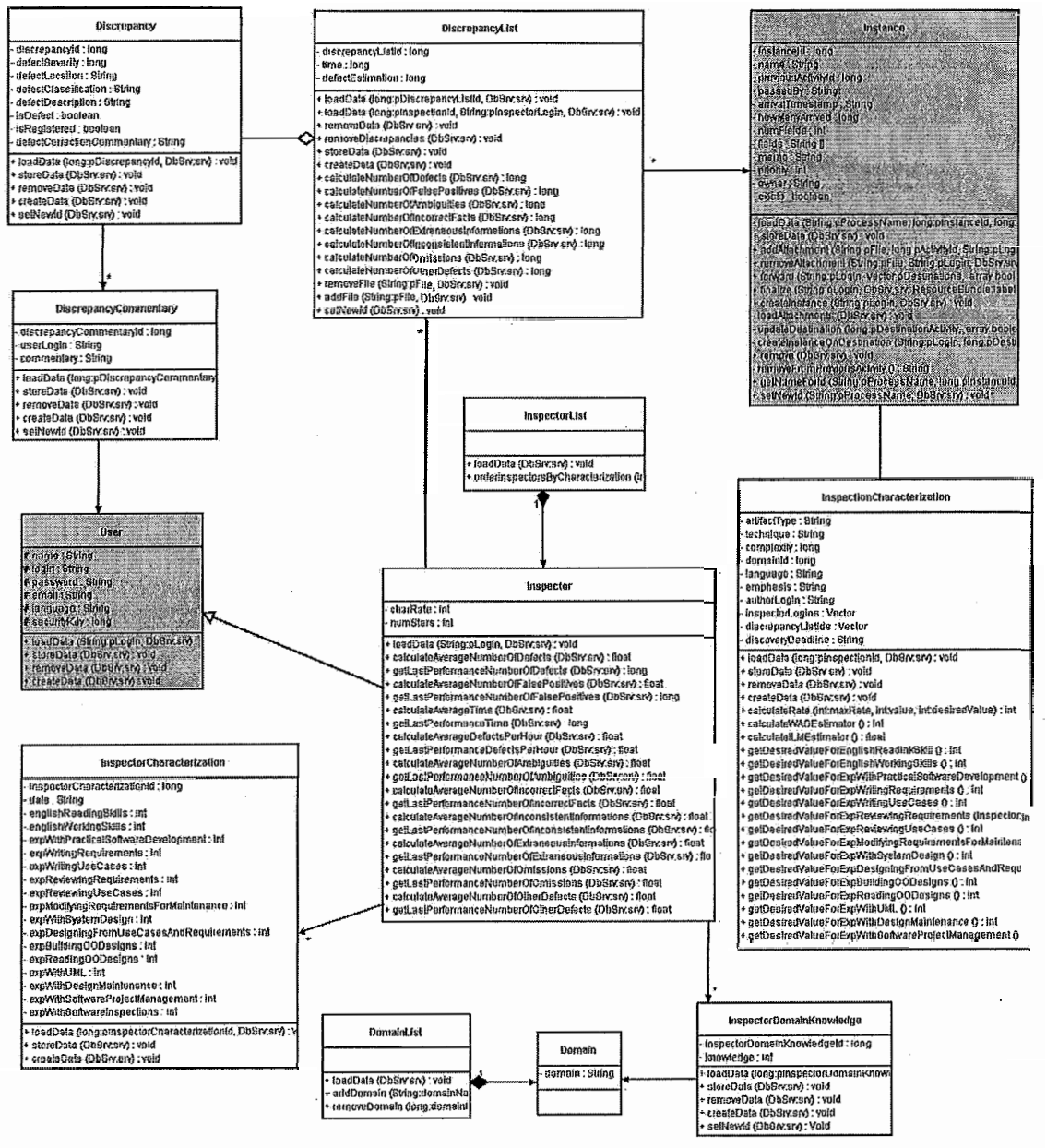
A.1 – Diagrama das Classes de Domínio da PatternFlow

Segue abaixo o diagrama das classes de domínio da ferramenta PatternFlow descrito na linguagem UML (BOOCH *et al.*, 1998).



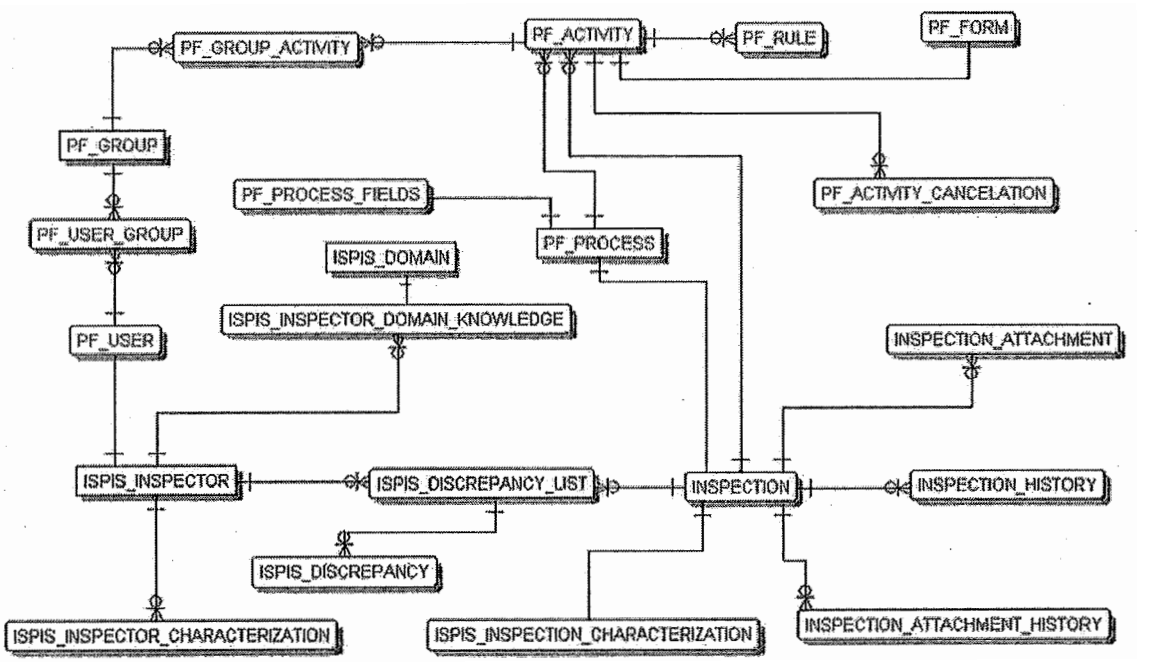
A.2 – Diagrama e Descrição das Classes de Domínio de ISPIS

Segue abaixo o diagrama das classes específicas do domínio de ISPIS, descrito em UML (BOOCH et al., 1998). Estas classes foram criadas para estender as funcionalidades da PatternFlow. As classes destacadas são as do PatternFlow, para maiores detalhes veja o diagrama da seção A.1.



Apêndice B – Modelo Entidade-Relacionamento

O modelo entidade relacionamento usado para a persistência dos dados de ISPIS, na notação James Martin (MARTIN e MCCLURE, 1985), se encontra abaixo. As tabelas com nome iniciado em 'PF_' formam os metadados da PatternFlow. As iniciadas em 'ISPIS_' constituem os dados específicos de apoio a inspeções de software. Por fim, as iniciadas com 'INSPECTION_' tratam da encenação do processo de inspeção seguindo a descrição encontrada nos metadados da PatternFlow.



Apêndice C – Instrumentos do Experimento

C.1 – Consentimento de Participação

Eu declaro ter mais de 18 anos de idade e que concordo em participar de um estudo conduzido pelo aluno Marcos Kalinowski e pelo Prof. Guilherme Horta Travassos. Este estudo visa compreender aspectos relativos ao planejamento de inspeções de software baseado em dados históricos.

PROCEDIMENTO

Eu entendo que serei solicitado a planejar uma inspeção. Neste exercício alguns métodos experimentais serão aplicados visando avaliar aspectos do planejamento de inspeções.

Os pesquisadores conduzirão o estudo consistindo da coleta, análise e relato dos dados do exercício. Eu entendo que não tenho obrigação alguma em contribuir com informação sobre meu desempenho no exercício, e que posso solicitar a retirada de meus resultados do experimento a qualquer momento. Eu entendo também que quando os dados forem coletados e analisados, meu nome será removido dos dados e que este não será utilizado em nenhum momento durante a análise ou quando os resultados forem apresentados.

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a manter sigilo das tarefas solicitadas e dos documentos apresentados e que fazem parte do experimento.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo do estudo experimental de livre e expontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas, ferramentas e processos para a Engenharia de Software.

RESPONSÁVEIS
Marcos Kalinowski
Prof. Guilherme Horta Travassos
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Nome (em letra de forma): _____

Assinatura: _____ Data: _____

C.2 – Caracterização dos Participantes

Nome _____ **Nível (Ms.c/D.Sc.):** ____

Experiência Prática Como Desenvolvedor De Software

Qual é sua experiência anterior com desenvolvimento de software na prática ? (marque aqueles itens que melhor se aplicam)

__ nunca desenvolvi software.
__ tenho desenvolvido software para uso próprio.
__ tenho desenvolvido software como parte de uma equipe, relacionado a um curso.
__ tenho desenvolvido software como parte de uma equipe, na indústria.

Por favor, explique sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em desenvolvimento (E.g. "Eu trabalhei por 10 anos como programador na indústria")

_____
_____
_____

Experiência em Desenvolvimento de Software

Por favor, indique o grau de sua experiência nesta seção seguindo a escala de 5 pontos abaixo:

1 = nenhum
2 = estudei em aula ou em livro
3 = pratiquei em 1 projeto em sala de aula
4 = usei em 1 projeto na indústria
5 = usei em vários projetos na indústria

Experiência com Requisitos

- Experiência escrevendo requisitos 1 2 3 4 5
- Experiência escrevendo casos de uso 1 2 3 4 5
- Experiência revisando requisitos 1 2 3 4 5
- Experiência revisando casos de uso 1 2 3 4 5
- Experiência modificando requisitos para manutenção 1 2 3 4 5

Experiência em Projeto

- Experiência em projeto de sistemas 1 2 3 4 5
- Experiência em desenvolver projetos a partir de requisitos e casos de uso 1 2 3 4 5
- Experiência criando projetos Orientados a Objetos 1 2 3 4 5
- Experiência lendo projetos Orientados a Objetos 1 2 3 4 5

- Experiência com Unified Modeling Language (UML) 1 2 3 4 5
- Experiência alterando projeto para manutenção 1 2 3 4 5

Outras Experiências

- Experiência com gerenciamento de projeto de software? 1 2 3 4 5
- Experiência com inspeções de software? 1 2 3 4 5
- Experiência Planejando Inspeções de Software? 1 2 3 4 5

Experiência em Contextos Diferentes

Nós usaremos esta seção para compreender quão familiar você está com vários sistemas que poderão ser utilizados como exemplos ou para exercícios durante o curso.

Por favor, indique o grau de experiência nesta seção seguindo a escala de 3 pontos abaixo:
1 = Eu não tenho familiaridade com a área. Eu nunca fiz isto.
3 = Eu utilizo isto algumas vezes, mas não sou um especialista.
5 = Eu sou muito familiar com esta área. Eu me sentiria confortável fazendo isto.

Quanto você sabe sobre...

- Utilizar um terminal de caixa eletrônico de banco? 1 3 5
- Utilizar ferramental de apoio a inspeções de software? 1 3 5

C.3 – Problema Modelado

C.3.1 – Versão Entregue aos Participantes sem Suporte Ferramental

Operação do Experimento

Juntamente com este documento você recebeu: (1) um documento de requisitos de um sistema para uma rede de máquinas de caixa eletrônico de banco e, (2) dados de inspetores disponíveis em uma organização fictícia. Estes dados capturam a caracterização dos inspetores e seu desempenho na última inspeção que realizaram aplicando a técnica PBR.

Um dos funcionários da organização, João Silva, elaborou o documento de requisitos que você recebeu. A empresa é responsável por todo o desenvolvimento do sistema. A fim de reduzir o retrabalho e aumentar a produtividade no desenvolvimento do sistema, a empresa deseja realizar uma inspeção no documento de requisitos. Esta inspeção deverá ter as seguintes características:

- Prazo de dois dias para a detecção de defeitos pelos inspetores;
- Os inspetores deverão utilizar a técnica PBR na detecção de defeitos;
- Não possui ênfase em encontrar um tipo específico de defeito.

A sua tarefa é, desempenhando o papel de moderador, elaborar um plano para a realização da inspeção do documento de requisitos dentro da organização, preenchendo os tópicos 1 e 2 do *template* abaixo.

1. DEFINIÇÃO DO CONTEXTO DA INSPEÇÃO

1.1. Sobre a Inspeção

Moderador: _____

Data: _____

Prazo para a atividade de detecção de defeitos: _____

Descrição: _____

_____

_____

Técnica de Inspeção: ( ) *ad-hoc;* ( ) *checklist;* ( ) PBR; ( ) OORT's.

Ênfase da Inspeção: ( ) Ambigüidades; ( ) Fato Incorreto; ( ) Informação Estranha; ( ) Informação Inconsistente ( ) Omissão.

1.2. Sobre o documento

Autor: _____

Tipo de artefato: ( ) Documento de Requisitos; ( ) Documento de Projeto; ( ) Código Fonte; ( ) Plano de Teste.

Língua do artefato: _____

Domínio da aplicação: _____

2. EQUIPE DE INSPETORES SELECIONADA

1) _____

2) _____

3) _____

3. DISTRIBUIÇÃO DO MATERIAL

Enviar por e-mail, para cada um dos inspetores selecionados, a definição do contexto da inspeção e o documento a ser inspecionado.

C.3.2 – Versão Entregue aos Participantes com Suporte Ferramental

Operação do Experimento

Juntamente com este documento estão disponibilizados para você: (1) um documento de requisitos de um sistema para uma rede de máquinas de caixa eletrônico de banco e, (2) um link (http://ese.cos.ufrj.br:8081/ispisexp) para uma a infra-estrutura ISPIS, que fornece apoio ao processo de inspeção de software. Nesta infra-estrutura, dados sobre inspetores de uma organização fictícia podem ser explorados. Estes dados capturam a caracterização dos inspetores e seu desempenho na última inspeção que realizaram aplicando a técnica PBR.

Um dos funcionários da organização, João Silva, elaborou o documento de requisitos que você recebeu. A empresa é responsável por todo o desenvolvimento do sistema. A fim de reduzir o retrabalho e aumentar a produtividade no desenvolvimento do sistema, a empresa deseja realizar uma inspeção no documento de requisitos. Esta inspeção deverá ter as seguintes características:

- Prazo de dois dias para a detecção de defeitos pelos inspetores;
- Os inspetores deverão utilizar a técnica PBR na detecção de defeitos;
- Não possui ênfase em encontrar um tipo específico de defeito.

A sua tarefa é, desempenhando o papel de moderador, utilizar a infra-estrutura para elaborar um plano para a realização da inspeção do documento de requisitos dentro da organização. Este plano envolve definir o contexto da inspeção, selecionar uma equipe com três inspetores e distribuir o material a ser utilizado na inspeção.

C.4 - Questionário de Acompanhamento

C.4.1 – Versão Entregue aos Participantes sem Suporte Ferramental

Nome:

Estas são perguntas que precisamos fazer de forma a tornar mais eficazes os dados obtidos ao longo do estudo.

1. Em relação ao seu grau de satisfação na realização da tarefa:

__ Eu fiquei muito satisfeito realizando a tarefa.
__ Eu fiquei satisfeito realizando a tarefa.
__ Eu fiquei insatisfeito realizando a tarefa.
__ Eu fiquei muito insatisfeito realizando a tarefa.

2. Ao realizar a tarefa você pensou em funcionalidades que poderiam ser oferecidas por ferramentas e que poderiam aumentar o seu grau de satisfação na execução da tarefa? Quais?

_____
_____
_____
_____
_____

3. Que critérios você utilizou para selecionar os inspetores? Justifique.

_____
_____
_____
_____
_____

4. Informe o tempo gasto na realização da tarefa (em minutos): _____

C.4.2 – Versão Entregue aos Participantes com Suporte Ferramental

Nome:

Estas são perguntas que precisamos fazer de forma a tornar mais eficazes os dados obtidos ao longo do estudo.

1. Em relação ao seu grau de satisfação na realização da tarefa através do suporte ferramental:

__ Eu fiquei muito satisfeito realizando a tarefa.
__ Eu fiquei satisfeito realizando a tarefa.
__ Eu fiquei insatisfeito realizando a tarefa.
__ Eu fiquei muito insatisfeito realizando a tarefa.

2. Ao utilizar o suporte ferramental você pensou em funcionalidades que poderiam aumentar o seu grau de satisfação na execução da tarefa? Quais?

_____
_____
_____

3. Você considera o suporte funcional oferecido pela ferramenta adequado para a realização da tarefa? (Justifique)

_____
_____
_____

4. Ao realizar a tarefa com suporte ferramental você pensou em funcionalidades que poderiam ser oferecidas e que estão ausentes? Quais?

_____
_____
_____

5. Ao realizar a tarefa com suporte ferramental você pensou em funcionalidades que foram oferecidas pela ferramenta e que atrapalharam a realização da tarefa ou que você tenha considerado inúteis? Quais?

6. Que critérios você utilizou para selecionar os inspetores? Justifique.

7. Informe o tempo gasto na realização da tarefa (em minutos): _____