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

Article · January 2004

CITATIONS
10

READS
213

4 authors, including:

Marcos Kalinowski
Universidade Federal Fluminense
88 PUBLICATIONS **369** CITATIONS
SEE PROFILE

Guilherme Horta Travassos
Federal University of Rio de Janeiro
289 PUBLICATIONS **2,943** CITATIONS
SEE PROFILE

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

Marcos Kalinowski Rodrigo Oliveira Spínola Guilherme Horta Travassos
Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ
{mkali, ros, ght}@cos.ufrj.br

Resumo

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. Muito conhecimento tem sido produzido na área de inspeções de software e se mostrado adequado através de estudos experimentais. Entretanto, pouco apoio computacional capturando este conhecimento está disponível. Este artigo apresenta uma infra-estrutura computacional para apoio ao processo de inspeção de software utilizando como conjunto básico de requisitos o conhecimento adquirido através de estudos experimentais relacionados à inspeção de software.

Palavras-Chave: Inspeção de software, infra-estrutura de apoio ao processo de inspeção.

Abstract

Software inspections aim to improve software quality by the analysis of software artifacts, detecting and removing their defects before these artifacts can be delivered to the following software life cycle activity. Some knowledge regarding software inspections have been acquired by empirical studies. However, there is no indication that do exist computational support exploring such knowledge. This paper describes a computational framework to support the software inspection process whose requirements set is derived from knowledge acquired by empirical studies.

Keywords: Software inspection, framework support for the software inspection process.

1. Introdução

O custo da correção de defeitos aumenta na medida que o processo de desenvolvimento progride. Desta forma, iniciativas devem ser realizadas no sentido de encontrar e corrigir defeitos tão logo sejam introduzidos. Revisões de artefatos de software têm se mostrado uma abordagem eficiente e de baixo custo para encontrar defeitos, reduzindo o retrabalho e melhorando a qualidade dos produtos.

Inspeção de software [11] é um tipo particular de revisão que possui um processo de detecção de defeitos rigoroso e bem definido. A importância de inspeções na garantia da qualidade de software está bem documentada na literatura [1, 11, 12, 22] .

O processo de inspeção tradicional [11] 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.

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 através de estudos experimentais e se mostrado adequado [4, 23]. Sauer *et al.* [22] apresentaram uma reorganização do processo de inspeção de software baseada em resultados de estudos experimentais.

Além disto, resultados de um *survey*, descritos por Ciolkowski *et al.* [8], relatam que, embora muitas organizações de software realizem revisões, a forma como as revisões são realizadas ainda é pouco sistematizada, pouco conhecimento da área de inspeções de software

é utilizado e assim o potencial 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 (40% realizam inspeções em requisitos e 30% realizam inspeções em código), (2) muitas vezes a atividade de detecção de defeitos não é realizada sistematicamente (cerca de 60% não conduzem uma atividade de detecção de defeitos regularmente), e (3) cerca de 40% das organizações não coletam dados e 18% coletam dados e não os utilizam em suas análises.

Para atender a problemas relativos à realização de inspeções de software, diversas propostas de apoio ferramental surgiram. Entre as mais conhecidas estão: ICICLE [6], CSI [19], CSRS [14], Scrutiny [13], WIT [28] e IBIS [17]. Muitas destas ferramentas têm como foco a detecção de defeitos e a inspeção de um tipo de artefato específico. Uma das características de WIT e IBIS é permitir a realização de inspeções por equipes geograficamente distribuídas. Além disto, IBIS automatiza a reorganização do processo de inspeção proposta em [22]. Analisando o conhecimento da literatura, identificou-se ser possível explorar informações experimentalmente validadas para fornecer mais apoio aos pontos de tomada de decisão do processo.

Baseado neste cenário, este trabalho apresenta uma infra-estrutura computacional de apoio ao processo de inspeção de software que pode ser utilizada para inspecionar diferentes artefatos produzidos ao longo do processo de desenvolvimento de software; objetivando que revisões sejam realizadas na prática de forma mais sistemática e explorando melhor o seu potencial, utilizando dados históricos e informações experimentalmente avaliadas sobre inspeções de software.

Além desta introdução, este artigo consta de mais cinco seções. A seção 2 apresenta as características do processo de inspeção. Na seção 3 a proposta da infra-estrutura é descrita. A seção 4 apresenta sua arquitetura e configuração atual. Em seguida, na seção 5, é mostrado de forma detalhada como a infra-estrutura apóia inspeções de software. Por fim, a seção 6 relata as contribuições e perspectivas de utilização da infra-estrutura.

2. Características do Processo de Inspeção

Desde que o processo tradicional de inspeção de software foi definido, diversos trabalhos surgiram questionando e tentando avaliar sua estrutura. Votta [31], por exemplo, argumenta que evitando reuniões de inspeção de forma síncrona, custos e conflitos de alocação de recursos podem ser reduzidos sem sacrificar a eficiência da inspeção. Estudos experimentais, como os analisados em [15] reforçam este argumento. Baseado nestes resultados, entre outros, uma reorganização do processo de inspeção é proposta em [22] e está ilustrada na Figura 1.

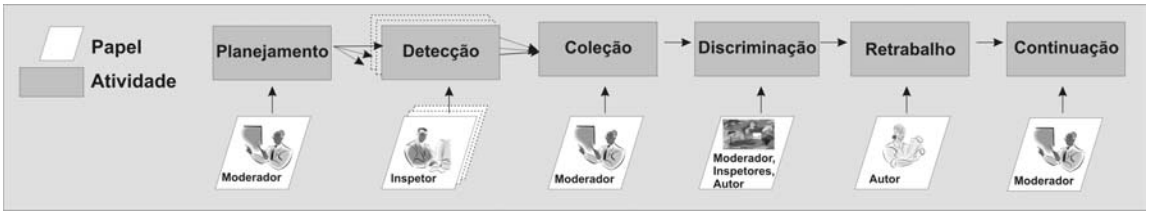


Figura 1. Reorganização do processo de inspeção, adaptada de [22].

De forma resumida, as atividades deste processo podem ser descritas da seguinte forma:

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.

Detecção de Defeitos. Os inspetores selecionados pelo moderador realizam a 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 (possíveis defeitos).

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. Durante esta discussão, algumas serão classificadas como defeito e outras falso positivo. Os falso positivos serão descartados e os defeitos serão registrados em uma lista de defeitos.

Retrabalho. O autor do documento corrige os defeitos da lista de defeitos e produz um relatório de correção dos defeitos.

Continuação. O moderador decide se uma nova inspeção deve ou não ocorrer.

A reorganização do processo mantém a estrutura rígida e os aspectos colaborativos do processo tradicional, 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 [20].

3. Proposta de Apoio ao Processo de Inspeção de Software

Apresentadas as características do processo de inspeção, uma proposta de apoio foi formulada a partir de conhecimento teórico sobre inspeções de software, que tem se mostrado adequado em resultados de estudos experimentais obtidos da literatura [4, 5 7, 18, 30].

3.1 Planejamento

No planejamento, a infra-estrutura deve fornecer apoio à tomada de decisão da seleção de inspetores, de acordo com (1) sua caracterização, e (2) utilizando dados históricos sobre o desempenho do inspetor em inspeções passadas sobre o mesmo tipo de artefato.

A caracterização deve ser utilizada através da aplicação de um resultado de Carver [7]. Neste trabalho, 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[1] (*grounded hypotheses*) através da exploração de dados históricos de estudos experimentais sobre inspeções. Assim, uma lista ordenada dos inspetores mais indicados para a detecção de defeitos pode ser fornecida de acordo com a forma como suas caracterizações se adequam à caracterização desejada para o contexto da inspeção, obtida a partir das hipóteses fundamentadas. O critério de ordenação de inspetores seria então o número de características desejadas para a inspeção as quais o inspetor se adequa. Isto ocorre quando o valor de sua caracterização para esta característica for próximo ao seu valor desejado para a inspeção em questão.

3.2 Detecção de Defeitos

Na detecção de defeitos a infra-estrutura deve disponibilizar técnicas de inspeção e auxiliar em sua aplicação. A técnica disponibilizada depende do planejamento feito para a

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

inspeção em questão. Entre as técnicas mais conhecidas estão a *ad-hoc* e o uso de *checklists*[2]. Existem ainda técnicas de leitura mais específicas, como a baseada em perspectiva (PBR) [23] e as de projetos orientados a objeto (OORT's) [29]. PBR apóia a detecção de defeitos em documentos de requisitos, enquanto OORT's apóia a detecção de defeitos em projetos descritos através de diagramas UML. Deve ser possível ainda estender a infra-estrutura para prover apoio a novas técnicas de inspeção.

3.3 Coleção de Defeitos

Na coleção de defeitos, a infra-estrutura deve utilizar os resultados do estudo experimental apresentado em [18], que mostram que a discriminação deve ser restrita a discrepâncias encontradas por apenas um inspetor. Desta forma, as discrepâncias repetidas, encontradas pelo moderador, devem ser diretamente classificadas pela infra-estrutura como defeito e encaminhadas para a atividade de retrabalho.

3.4 Discriminação de Defeitos

Nesta atividade, o processo de inspeção de [22] envolve uma discussão assíncrona das discrepâncias. A infra-estrutura deve possibilitar esta discussão para participantes geograficamente distribuídos.

Resultados de um estudo experimental [30] revelaram uma característica interessante da discriminação de defeitos, o anonimato dos participantes pode ajudar na classificação correta das discrepâncias. Desta forma, 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.

3.5 Retrabalho

O retrabalho nos artefatos inspecionados 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 na atividade seguinte do processo.

3.6 Continuação

Na continuação, a infra-estrutura deve apoiar a tomada de decisão de realizar uma nova inspeção ou não, utilizando os seguintes dados: (1) estimativas da cobertura de defeitos[3] para a inspeção atual e para a reinspeção, e (2) dado histórico 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.*, [5] apresentam uma forma de estimar o número de defeitos de documentos. Esta utiliza o *Weighed Average Of Individual Offsets* (WAO), onde estimativas subjetivas individuais do número de defeitos são utilizadas para se obter uma estimativa subjetiva da equipe de inspetores como um todo. Em um estudo experimental apresentado neste mesmo artigo, a estimativa subjetiva de equipe utilizando o WAO teve desempenho superior ao modelo objetivo de estimativa *Jack Knife*[4] e do que modelos de estimativas subjetivas individuais. A estimativa utilizando o WAO pode ser calculada de acordo com a fórmula da Figura 2.

---

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

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

[4] O modelo Jack Knife é um modelo objetivo que tem mostrado bons resultados em estudos experimentais [3].

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 para o documento e o número de defeitos que ele efetivamente encontrou na inspeção. Para este modelo, a estimativa individual de cada um dos inspetores utilizada 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.

$$N = \sum_{d=1}^{DP}\left(D_d + \sum_{k=1}^{S}\left(\left(\eta_d^k - n_d^k\right)\cdot \omega_d^k\right) \Big/ \sum_{k=1}^{S} \omega_d^k \right)$$

S	Tamanho da equipe de inspeção.	η_d^k	Estimativa do inspetor k do número total de defeitos presentes na parte d do documento.
DP	Partes do documento.		
k	Identificador do inspetor dentro de sua equipe 1 ... S.	N	Estimativa da equipe para o número total de defeitos presentes no documento.
d	Identificador da parte do documento.	ϖ_d^k	Peso da estimativa do inspetor k para a parte d do documento.
n_d^k	Número de defeitos encontrados pelo inspetor k para a parte d do documento.		
D_d	Número de defeitos encontrados pela equipe na parte d do documento		

Figura 2. Estimativa do número de defeitos utilizando o WAO. Adaptado de [5].

Para calcular a cobertura de defeitos da próxima inspeção, o modelo ILM (*Improved Linear Model*) [2] pode ser utilizado. Em um estudo experimental envolvendo 169 indivíduos [4], o modelo ILM foi mais eficiente do que os modelos triviais[5], o modelo linear original e do que o modelo de aumento de confiabilidade (*reliability growth model*) proposto neste mesmo artigo.

O modelo ILM faz sua estimativa de acordo com a fórmula da Figura 3. Este modelo supõe que em uma nova inspeção, com mesmo esforço da inspeção anterior, a cobertura será acrescida da taxa de detecção (cobertura da inspeção anterior normalizada pelo esforço envolvido em cada uma das inspeções) vezes o percentual de defeitos restantes no documento.

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

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. O modelo ILM. Adaptado de [2].

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 e da reinspeção for alta, a infra-estrutura deve sugerir uma nova inspeção, visando uma boa cobertura de defeitos para a inspeção.

Em relação ao dado histórico do número médio de defeitos em inspeções passadas sobre o mesmo tipo de artefatos, diretrizes, como a sugerida por [11], podem ser aplicadas. De acordo com 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.

4 Proposta de Infra-estrutura Computacional

A partir deste cenário, foi elaborada uma arquitetura que possibilitasse a instanciação da infra-estrutura computacional para realização de inspeções de diferentes artefatos com

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

equipes geograficamente distribuídas que contemplasse a proposta de apoio apresentada na seção 3. As subseções seguintes apresentam a arquitetura projetada e sua instanciação através de projetos de pesquisa realizados pela equipe de engenharia de software experimental da COPPE/UFRJ.

4.1 Arquitetura

Três tipos de componentes foram identificados para a arquitetura: (1) um arcabouço para a infra-estrutura, (2) ferramentas externas de apoio à aplicação de técnicas de inspeção específicas para determinado tipo de artefato e, (3) um integrador de ferramentas para possibilitar o intercâmbio de dados. A Figura 4 ilustra a arquitetura proposta.

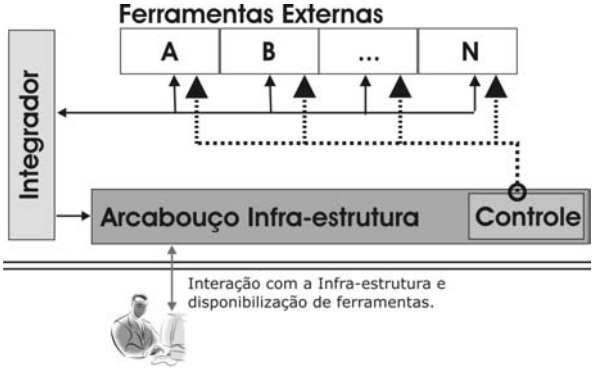


Figura 4. Arquitetura Proposta para a Infra-Estrutura.

Nesta arquitetura, o arcabouço da infra-estrutura é responsável, dentre outras coisas, por tornar a infra-estrutura acessível pela *web* e 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 que está sendo realizada. Os dados das ferramentas externas são então retornados utilizando o módulo integrador.

Uma das características desta arquitetura é o fato dela tornar a infra-estrutura fracamente acoplada, permitindo que ferramentas independentes sejam utilizadas em conjunto para prover mais funcionalidades à infra-estrutura, tornando-a customizável e extensível.

4.2 Instanciação da Infra-estrutura

Baseado na arquitetura apresentada na seção anterior, a infra-estrutura de apoio ao processo de inspeção de software foi implementada. O componente do arcabouço da infra-estrutura recebeu o nome ISPIS. Entre outras características, ISPIS automatiza o controle do processo de inspeção e, na fase de detecção de defeitos disponibiliza ferramentas externas a depender do contexto da inspeção que está sendo realizada. Estas ferramentas de apoio à aplicação de técnicas específicas de detecção de defeitos utilizadas são as atualmente disponíveis na COPPE/UFRJ. Elas apóiam a aplicação de PBR [25] e de OORT's [21]. Os dados gerados pela utilização destas ferramentas podem ser carregados em ISPIS mediante a utilização do *driver* de transformação gerado pela ferramenta de integração [26]. A configuração atual da infra-estrutura está ilustrada na Figura 5.

Apresentada essa visão geral da infra-estrutura serão descritas a seguir, em maiores detalhes, as ferramentas que formam sua configuração atual.

ISPIS. É o arcabouço da infra-estrutura através do qual o usuário acessa o sistema. Tarefas específicas são solicitadas de acordo com o papel que este usuário desempenha dentro do processo de inspeção.

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 [9], 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.

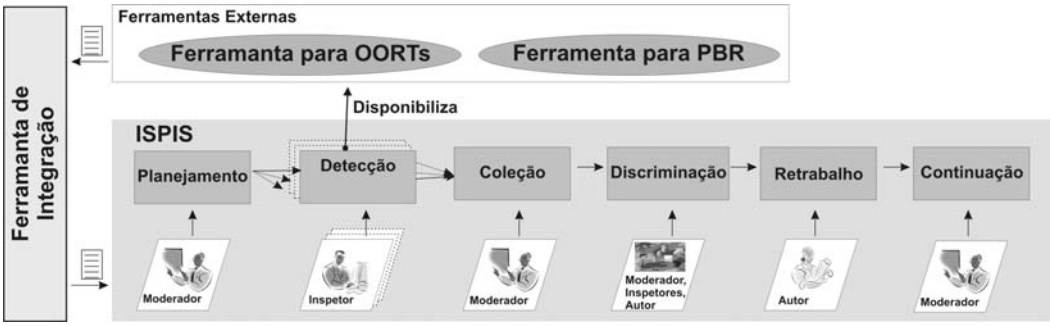


Figura 5. Configuração atual da infra-estrutura.

No entanto, ISPIS precisa interagir com a ferramenta de *workflow* para redefinir a estrutura do processo de acordo com as decisões que vão sendo tomadas ao longo do processo. Por este motivo, ao invés de utilizar uma ferramenta de *workflow*, ISPIS foi desenvolvida como sendo a extensão de uma destas ferramentas. A ferramenta de *workflow* escolhida para ser estendida foi a PatternFlow [16].

Para fornecer um apoio customizado à inspeção de tipos de artefatos específicos ISPIS utiliza a integração a outras ferramentas na atividade de detecção de defeitos. Entre estas ferramentas se encontram as de apoio à aplicação de PBR e OORT's.

Ferramenta de Integração. Para garantir a interoperabilidade entre ferramentas independentes é preciso que haja um mecanismo para transformações estruturais e semânticas nos dados e uma tecnologia de armazenamento comum. A ferramenta de integração utiliza esquemas e ontologias para efetuar as transformações estruturais e semânticas, respectivamente. Ela assume como premissa que o intercâmbio dos dados seja feito utilizando o padrão XML.

De forma simplificada, a idéia é mapear na ontologia os esquemas dos documentos a serem integrados. Isto centraliza informações semânticas e estruturais tornando possível a geração automatizada de *drivers* de transformação entre os artefatos.

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 a ferramenta de Casos de Uso [24]. 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 é configurável para permitir a aplicação customizada de OORT's. Desta forma, ela objetiva fornecer apoio

automatizado à aplicação e configuração de OORT's [21]. 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.

5 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. Cada instância de inspeção que foi ou está sendo realizada pode ser acompanhada pelo seu moderador. O acompanhamento mostra quando uma determinada atividade foi concluída e por quem. Desta forma o moderador é auxiliado na tarefa de gerência do processo.

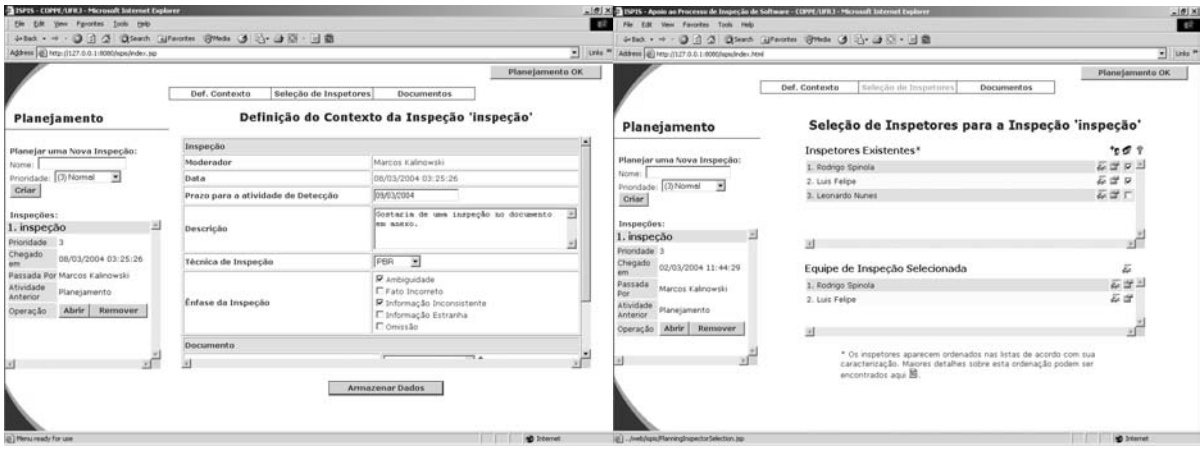
As seções seguintes apresentam o apoio fornecido a cada uma das atividades do processo de inspeção.

5.1 Planejamento

Na fase 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 6(a)) 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.

Seguindo a proposta descrita na seção 3.1, ISPIS apóia a seleção fornecendo uma lista ordenada dos inspetores mais indicados para a fase de inspeção individual (Figura 6(b)).

Para implementar esta proposta, algumas hipóteses fundamentadas de Carver [7] foram selecionadas. 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:



(a) (b)

Figura 6. (a) Definição do contexto. (b) Lista ordenada de inspetores.

- 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);

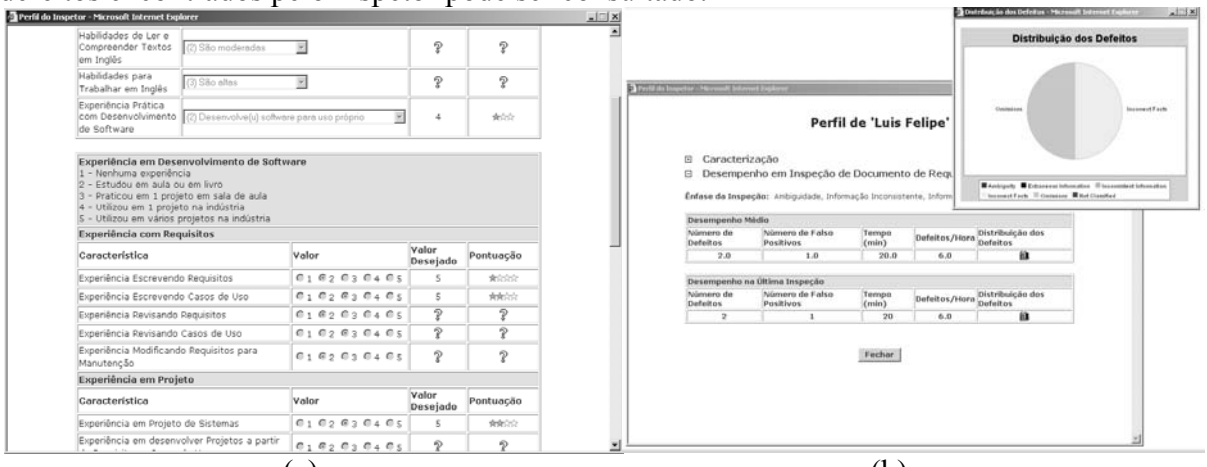
- R4: (O inspetor possui experiência com Projetos de Software) ^ *not* (a inspeção utiliza uma checklist procedural, bem definida);
- R5: (O inspetor possui conhecimento do domínio) ^ *not* (a inspeção utiliza uma técnica procedural);
- R6: (O inspetor possui experiência revisando requisitos) ^ (o inspetor 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. Para cada uma das características o inspetor recebe uma pontuação de 1 a 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 na Figura 8.

Na caracterização, ilustrada na Figura 7 (a), 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). No desempenho em inspeções passadas neste mesmo tipo de artefato, ilustrado na Figura 7 (b), é possível consultar a eficiência média do inspetor e a eficiência da última inspeção. São exibidos os seguintes dados: o número de defeitos, número de falso positivos, tempo gasto e a eficiência (defeitos/hora). Além disto, a ênfase no tipo de defeitos a serem encontradas pela inspeção é exibido e um gráfico para ver a distribuição dos tipos de defeitos encontrados pelo inspetor pode ser consultado.



(a) (b)

Figura 7. Perfil do inspetor. (a) Parte de caracterização do inspetor. (b) Parte de desempenho em inspeções sobre o mesmo tipo de artefato do inspetor.

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, bastando que estes sejam anexados à inspeção que está sendo planejada.

No momento em que o planejamento se encerra, ISPIS possui informações suficientes para definir a estrutura do processo de inspeção de [22]. São criadas:

- 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 descrito em [22].

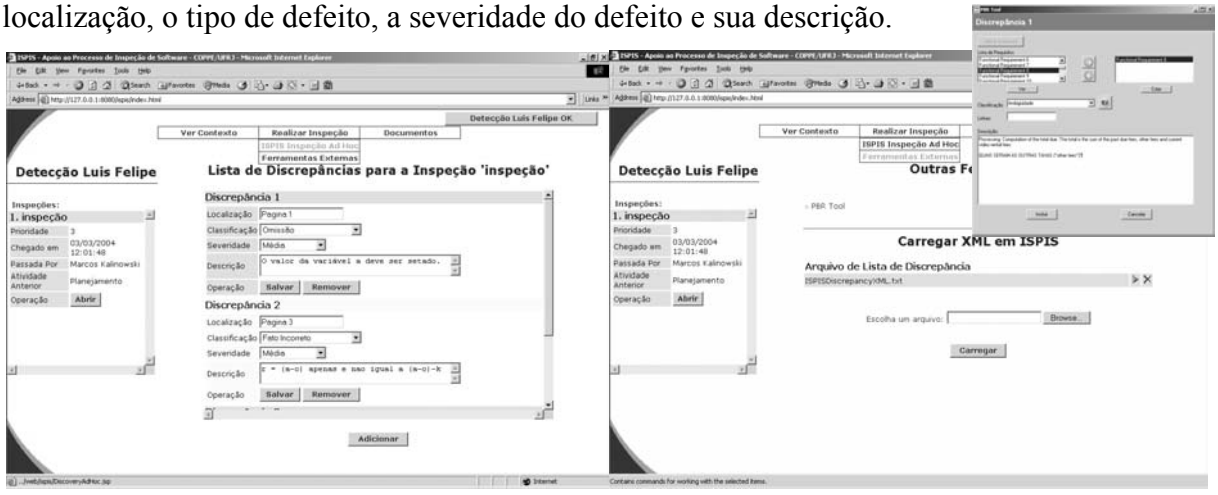
Esta forma dinâmica de definição do processo possibilita ISPIS coordenar a realização de diversas inspeções ocorrendo ao mesmo tempo. Note que, na definição do processo existirão tantas atividades de coleção e de discriminação quanto forem as inspeções planejadas em ISPIS. As regras de desvio possibilitam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

5.2 Detecção de Defeitos

Para que possa ser utilizada em qualquer tipo de artefato, ISPIS possibilita, em sua configuração básica, inspeções *ad-hoc*. Um apoio mais específico para a inspeção de determinados artefatos é fornecido através da integração a ISPIS das ferramentas de apoio a PBR e OORT's.

Na detecção *ad-hoc*, ilustrada na Figura 8 (a), as tarefas do inspetor se resumem a ler o documento e cadastrar discrepâncias. Cada discrepância encontrada é registrada com sua localização, o tipo de defeito, a severidade do defeito e sua descrição.



(a) (b)

Figura 8. (a) Detecção *ad-hoc*. (b) Detecção através da integração para uma inspeção aplicando a técnica PBR, utilizando a ferramenta correspondente.

Na detecção através da integração, ilustrada na Figura 8 (b), ISPIS disponibiliza a ferramenta adequada para a inspeção em questão para *download*, o inspetor então utiliza esta

ferramenta para realizar a detecção. 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, estes 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.

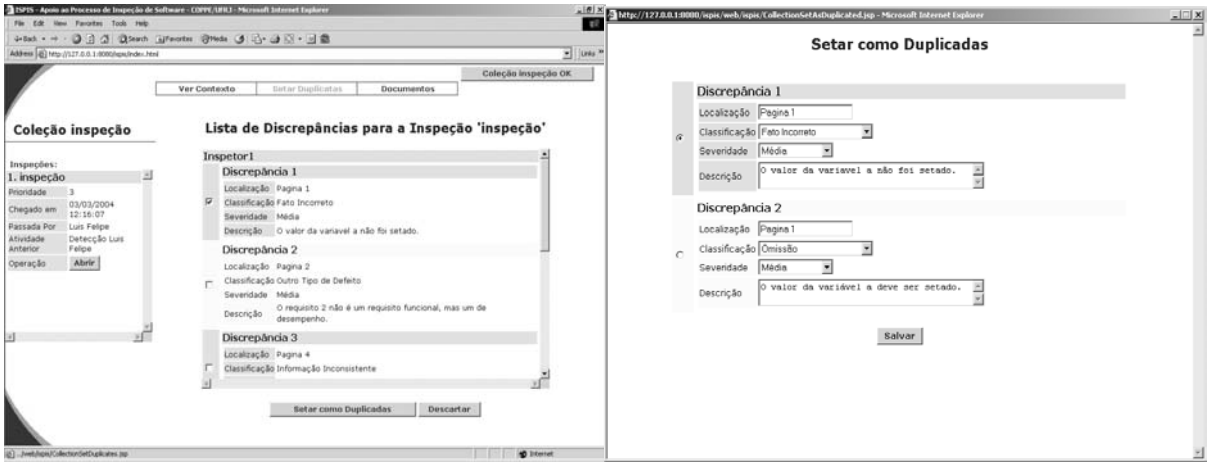
A ativação internamente ocorre em dois passos. Primeiramente ISPIS transforma o XML do documento a ser ativado para que possa ser lido. Depois ISPIS lê o documento XML e carrega seus dados em sua base de dados específicos para apoiar inspeções de software. Para a transformação do documento ISPIS utiliza um *driver* gerado pela Ferramenta de Integração.

Ao final da atividade de detecção o inspetor é direcionado 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 [5] e estimar a cobertura de uma nova inspeção utilizando o modelo ILM [2].

Na medida que as atividades de detecção dos inspetores são concluídas, 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.

5.3 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 9 (a). Ele poderá então selecionar discrepâncias destas listas e as descartar ou classificar elas como duplicatas. No momento em que uma discrepância é descartada ela é classificada como um falso positivo e não será mais exibida.



(a) **(b)**

Figura 9. (a) Listas de discrepâncias dos inspetores. (b) Classificando uma discrepância como duplicata.

A classificação de uma discrepância como duplicata está ilustrada na Figura 9 (b). Fica sob responsabilidade do moderador selecionar uma das versões da discrepância. A discrepância selecionada poderá ainda ser editada, de modo que informações importantes contidas nas duplicatas possam ser acrescentadas. No momento que uma discrepância é classificada como duplicata, suas réplicas são classificadas como falso positivo e não serão mais exibidas. Seguindo a sugestão experimentalmente avaliada de [18], a discrepância

selecionada é classificada como defeito e encaminhada diretamente para a atividade de retrabalho.

No momento que o moderador define a atividade de coleção 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.

5.4 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. A Figura 10 ilustra o apoio de ISPIS à fase de discriminação.

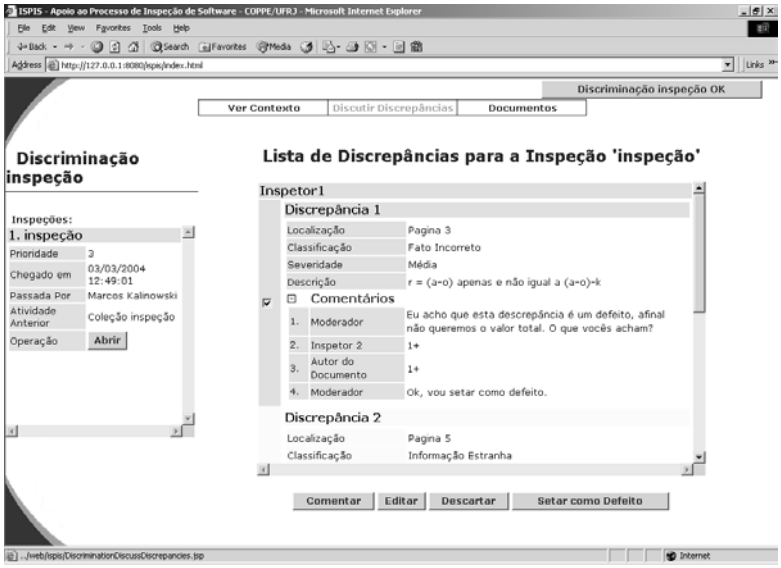


Figura 10. Discriminação de Defeitos em ISPIS

Seguindo a proposta da seção 3.4, os participantes da discriminação em ISPIS são anônimos. Nos comentários é exibido o papel desempenhado pelo participante da discriminação e não o seu nome.

Só o moderador tem acesso a editar, descartar e classificar discrepâncias como defeitos. 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 é removido.

5.5 Retrabalho

Nesta atividade, ISPIS não fornece apoio à correção dos defeitos no documento, uma vez que esta tarefa não faz parte do processo de inspeção. Após a correção do documento, ele pode ser anexado. 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.

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

5.6 Continuação

Nesta fase, o moderador tem acesso ao contexto da inspeção, definido na fase de planejamento, às versões defeituosas e corrigidas do documento e ao relatório de correção de defeitos. O moderador deve então decidir se uma nova inspeção deve ou não ocorrer. Esta

tomada de decisão é importante para garantir a qualidade do produto sendo desenvolvido e recebe apoio de ISPIS.

Uma das formas utilizadas em ISPIS para apoiar a decisão 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 [5]. Para a estimativa da cobertura de defeitos da próxima inspeção o modelo ILM [2] é utilizado. Nesta estimativa, ISPIS considera que a variável subjetiva 'esforço' do modelo ILM seja o mesmo 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%. ISPIS apresenta ao moderador: o valor estimado da inspeção atual, o valor estimado para a próxima inspeção e a sugestão de ISPIS. A forma como ISPIS faz esta sugestão está ilustrada na Figura 11.

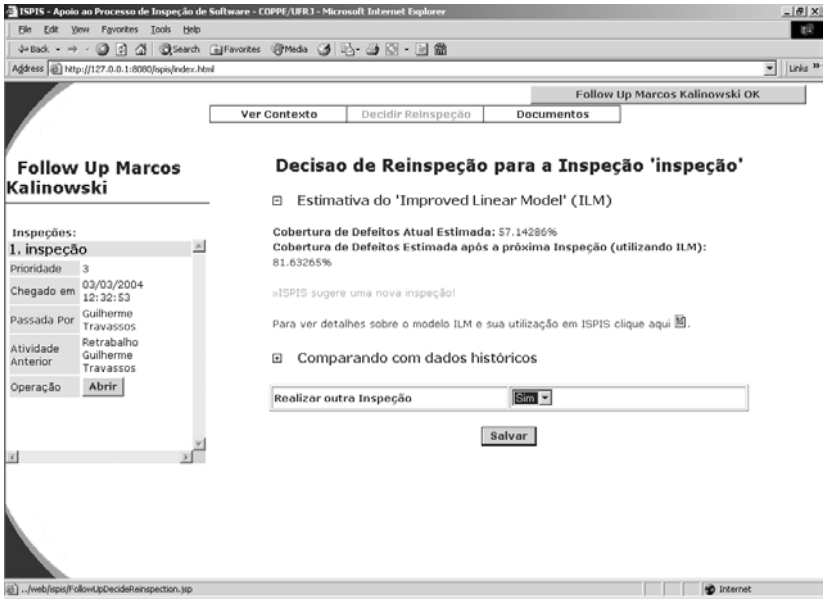


Figura 11. Apoio de ISPIS à decisão de realizar ou não uma nova inspeção

A figura mostra ISPIS sugerindo uma nova inspeção considerando a cobertura atual estimada de 57.14% e cobertura estimada após a próxima inspeção de 81.63%.

Uma 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 [11] ISPIS então sugere uma nova inspeção se o número de defeitos encontrados no documento for 5% maior do que a média. Esta sugestão visa a qualidade do produto, tentando evitar que um documento inicialmente muito defeituoso seja passado para as próximas atividades do processo de desenvolvimento de software. Embora ISPIS faça sugestões, a decisão final sobre a realimentação do processo de inspeção ou não continua sob responsabilidade do moderador.

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

6 Contribuições e Perspectivas de Utilização

Um dos fundamentos para desenvolver software com sucesso é focar na qualidade de software. Uma das formas de aumentar a qualidade é a aplicação de inspeções nos artefatos desenvolvidos ao longo do processo de desenvolvimento. A importância da aplicação de

inspeções na garantia da qualidade é destacada pelo modelo CMMI, que identifica revisões como uma das áreas chave do processo de desenvolvimento de software [10].

Embora as vantagens de se utilizar inspeções de software sejam conhecidas, a forma como inspeções são realizadas na prática nas organizações de software ainda é pouco sistematizada e seu potencial raramente é explorado. Visando isto, uma infra-estrutura que faz uso de conhecimento sobre inspeções de software adquirido através de estudos experimentais e de dados históricos para apoiar pontos de tomada de decisão foi implementada.

Um estudo experimental foi conduzido para avaliar o apoio de ISPIS para a atividade de planejamento e seus resultados preliminares foram positivos. Em complemento, um estudo de caso mostrou a viabilidade de se utilizar ISPIS em inspeções reais e a infra-estrutura têm sido efetivamente utilizada em inspeções realizadas na COPPE/UFRJ. O detalhamento destes estudos pode ser encontrado em [32].

Acreditamos que a disponibilização desta infra-estrutura poderá permitir a sua utilização por diferentes organizações e parceiros de pesquisa. Desta forma, as organizações poderão realizar suas inspeções de forma mais sistemática e usufruir os benefícios fornecidos pela aplicação automatizada de conhecimento experimentalmente avaliado.

7. Agradecimentos

Os autores deste trabalho gostariam de agradecer a CAPES, ao CNPq e ao projeto NSF-CNPq READERS pelo apoio financeiro e à Equipe de Engenharia de Software Experimental da COPPE/UFRJ (http://www.cos.ufrj.br/~ese) pelas sugestões significativas.

Referências

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

[2] ADAMS, T., 1999, “A formula for the re-inspection decision”, Software Engineering Notes 24(3): 80.

[3] BIFFL, S., GROSSMANN, W., 2001, “Evaluating the accuracy of objective estimation models based on inspection data from multiple inspection cycles”, ICSE 2001, Toronto, Canada, IEEE Comp. Soc. Press, May.

[4] 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.

[5] BIFFL, S., HALLING, M., KOESZEGI, S., 2003, “Investigating the Accuracy of Defect Estimation Models for Individuals and Teams Based on Inspection Data”, ISESE 2003, Roman Castles (Rome), Italy, IEEE Computer Society Press, September/October.

[6] BROTHERS, L., SEMBUGAMOORTHY, V., MULLER, M., “ICICLE: Groupware for Code Inspection”, Conference on CSCW, Sept. 1990.

[7] CARVER, J.C., BASILI, V. R., 2003, “The Impact of Background and Experience on Software Inspections”, Tese de doutorado, University of Maryland, EUA, 2003.

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

[9] CHAFFEY, D., 1998, “Groupware, workflow and Intranets: Reengineering the Enterprise with Collaborative Software”, 1ed. Butterworth-Heinemann.

[10] 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.

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

[12] GILB, T., GRAHAM, D., 1993, “Software Inspection”, Addison-Wesley.

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

[14] JOHNSON, P. M., TJAHJONO, D., WAN, D., BREWER, R., 1993, "Experiences with CSRS: An Instrumented Software Review Environment", 11th Annual Pacific Northwest Software Quality Conference, Portland, OR. 1993.

[15] JOHNSON, P.M., TJAHJONO, D., 1996, "Assessing Software Review Meetings: A Controlled Experimental Study Using CSRS", Technical Report 96-06, Dept. of Information and Computer Sciences, Univ. of Hawaii.

[16] 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.

[17] LANUBILE, F., MALLARDO, T., 2002, "Tool support for distributed inspection", COMPSAC 2002, Oxford, England.

[18] LANUBILE, F. MALLARDO, T., 2003, "An Empirical Study of Web-Based Inspection Meetings", ISESE 2003, Roman Castles (Rome), Italy, IEEE Computer Society Press.

[19] MASHAYEKHI, V., DRAKE, J.M., TSAI, W.T., RIEDL, J., "Distributed, collaborative software inspection", IEEE Software, Volume:10 Issue: 5, Sept. 1993.

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

[21] REIS, L.N.M, TRAVASSOS, G.H., 2003, "Apoio Automatizado à Configuração e Aplicação de OORT's", WTES 2003 (SBES), p.59-64, Manaus.

[22] 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.

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

[24] SILVA, L.F.S., CHAPETTA, W.A., TRAVASSOS, G.H., 2004, "Inspeções de Requisitos de Software Utilizando PBR e Apoio Ferramental", SBQS 2004.

[25] SILVA, L.F.S., TRAVASSOS, G.H., 2004, "Tool-Supported Unobtrusive Evaluation of Software Engineering Process Conformance", ISESE 2004, California.

[26] SPINOLA, R.O., TRAVASSOS, G.H., 2003, "Uma Abordagem para Integração de Ferramentas", WTES 2003 (SBES), p.59-64, Manaus.

[27] STEIN, M., RIEDL, J., HARNER, S.J., MASHAYEKHI, V., 1997, "A Case Study of Distributed, Assyncronous Software Inspection", Int'l Conf. Software Eng., pp. 107-117.

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

[29] TRAVASSOS, G. H., SHULL, F., FREDERICKS, M., BASILI, V. R. 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, 1999.

[30] 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, February 2003.

[31] VOTTA, L.G., 1993, "Does Every Inspectin Need a Meeting?", Proc. First ACM Symp. Foundations of Software Engineering, pp. 107-114.

[32] KALINOWSKI, M., TRAVASSOS, G.H., 2004. "A Computational Framework for Supporting Software Inspections", 19th IEEE Automated Software Engineering (submetido)