TÉCNICA PARA INSPEÇÃO DE DIAGRAMAS DE ATIVIDADES

Rafael Maiani de Mello

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Ciências em Engenharia de Sistemas e Computação.

Orientador: Guilherme Horta Travassos

Rio de Janeiro
Fevereiro de 2011

TÉCNICA PARA INSPEÇÃO DE DIAGRAMAS DE ATIVIDADES

Rafael Maiani de Mello

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) 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.

Examinada por:

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Prof. Geraldo Bonorino Xexéo, D.Sc.

_____

Prof. Leonardo Gresta Paulino Murta, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2011

Mello, Rafael Maiani de

Técnica para Inspeção de Diagramas de Atividades/ Rafael Maiani de Mello – Rio de Janeiro: UFRJ/COPPE, 2011.

XV, 166 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos.

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 107-115.

1. Inspeção de Software. 2. Especificação de Requisitos. 3. Diagrama de Atividades. 4. Engenharia de Software Experimental. I. Travassos, Guilherme Horta II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao nosso primeiro filho *(in memorian)*

Ao nosso segundo filho

Agradecimentos

Agradeço a Deus que, através de Sua Palavra, me deu fé e discernimento para superar todas as adversidades e perseverar até aqui.

À minha esposa, Fernanda, por tudo que ela é, tudo que ela representa para mim e tudo que fez e faz por mim.

Aos meus pais e minha avozinha, tão presentes na minha educação e meus incentivadores desde sempre.

Ao Prof. Guilherme Travassos, pelo empenho, compreensão, amizade e irretocável dedicação.

A todos os demais professores da COPPE, por manterem este centro de excelência acadêmica, em especial aqueles com os quais tive o privilégio de ser aluno: Profa. Ana Regina, Profa. Cláudia Werner e Prof. Jano.

A todo o Grupo de Engenharia de Software Experimental pelo companheirismo, apoio e relevantes contribuições, em especial ao Jobson, ao Wallace e à Silvia, parceiros de aulas e de pesquisa; à Taísa, pelo seu fundamental apoio e dedicação nas questões administrativas.

Aos demais professores e amigos que tanto me incentivaram e me inspiram ao longo da minha carreira acadêmica: Prof. Nery Machado Filho (*in memorian*), Prof. Paulo Eustáquio, Prof. Rogério Alvarenga e Profa. Mariluci Portes, Eduardo Carvalho, Elaine Zancanela e Rômulo Magnus.

Aos meus alunos, com os quais tanto aprendi em tão pouco tempo.

Aos grandes amigos que Deus me deu, em especial aqueles com quem sempre pude contar nos bons momentos e nos ruins: Padre Geraldo, Isaac e Bárbara, Rodrigo, Denise, Reinaldo, Rômulo, entre outros.

Aos meus colegas da Diretoria de Tecnologia do Banco do Brasil, com quem pude contar com a compreensão necessária para cumprir esta jornada.

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

TÉCNICA PARA INSPEÇÃO DE DIAGRAMAS DE ATIVIDADES

Rafael Maiani de Mello

Fevereiro/2011

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

Este trabalho apresenta ActCheck, uma técnica de inspeção configurável e baseada em *checklist* que tem como objetivo principal apoiar a identificação de defeitos nos diagramas de atividades em especificações de requisitos. A técnica proposta foi desenvolvida a partir de uma metodologia científica baseada em experimentação onde, inicialmente, foi conduzida uma *quasi*-revisão sistemática que apontou a carência de tecnologias que apoiassem a inspeção de modelos de *workflow* em geral. A primeira versão da técnica de inspeção foi submetida a uma prova de conceito e, posteriormente, a técnica foi evoluída, sendo desenvolvida ActCheck, que foi submetida a um estudo experimental *in vitro* onde, embora tenha sido identificada sua viabilidade como técnica de inspeção, não foi possível afirmar que esta apresente diferença significativa de eficácia ou eficiência quando seus resultados são comparados com os resultados de inspeções *ad hoc*.

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

A TECHNIQUE FOR ACTIVITY DIAGRAMS INSPECTION

Rafael Maiani de Mello

February/2011

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

This work presents ActCheck, a checklist-based and configurable inspection technique mainly developed to help defect detection on requirements specifications. This technique were developed using a scientific methodology based on experimentation. First, a *quasi*-systematic review was conducted and its results showed a lack of technologies to support workflow models inspections. The first version of the inspection technique was submitted to a proof of concept and, after that, the technique was evolved, when it was developed ActCheck, submitted to an empirical study *in vitro*. Based on this study, although the viability of ActCheck as inspection technique was identified, we could not affirm it has efficiency or effectiveness significantly different when its results are compared with the results of *ad hoc* inspections.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

CAPÍTULO 1 - INTRODUÇÃO

Neste capítulo será apresentado o contexto e a motivação para esta pesquisa, mencionando trabalhos relacionados. O capítulo também define os objetivos da pesquisa e, ao final, apresenta a organização do texto desta dissertação.

1.1 Contexto e Motivação: Descrição do Problema

Na engenharia de software, a descrição dos detalhes das funcionalidades e dos processos de negócio de um projeto costuma ser feita através de uma simples redação ou mesmo através de uma descrição textual apoiada por padrões, como casos de uso. Entretanto, a especificação de requisitos de aplicações contemporâneas costuma envolver uma grande quantidade de informação. Esta informação pode estar relacionada às atividades, bem como aos fluxos, dependências entre esses fluxos, desvios ou mesmo regras de negócio, contexto que dificulta a compreensão do escopo do projeto, quando esta informação está organizada somente através de uma descrição textual.

Neste contexto, Gutiérrez et al. (2008) sugerem que o uso de diagramas de atividades para representação de casos de uso permite vislumbrar o comportamento global de um caso de uso, especialmente seus aspectos dinâmicos. Gross e Doerr (2009) apontam que, atualmente, a modelagem de processos, apoiada por notações gráficas como o diagrama de atividades, possui um importante papel na engenharia de requisitos. Desde a sua primeira versão, em 1998, a UML tem se destacado como uma linguagem padrão para modelagem de projetos orientados a objetos, provendo modelos para a representação estática e dinâmica de sistemas. Entretanto, com o advento da versão 2.0 da UML em 2003 (OMG, 2009), o diagrama de atividades ganhou melhor definição e tornou-se um modelo de representação dinâmica mais abrangente, sendo sua aplicação não mais limitada a projetos de sistemas orientados a objeto, passando a ser, inclusive, adotado como notação para representação de *workflows* em geral (Wynn et al., 2009).

Atualmente, o diagrama de atividades provê recursos sintáticos para representação de diversas situações que podem ocorrer no fluxo de controle e de dados de uma aplicação tais como, por exemplo, fluxos em paralelo, loops, interrupções e exceções, tornando-se assim um potencial candidato para captura dos

requisitos de aplicações de software que demandam recursos multitarefa e distribuição. O modelo de atividades da UML 2 é considerado uma versão bastante melhorada dos fluxogramas (Travassos, 2008), uma das mais antigas ferramentas de modelagem de software existentes. As estruturas do fluxograma clássico (instruções, decisões e fluxo de controle) possuem correspondentes nos diagramas de atividades. Apesar dos fluxogramas terem sido utilizados inicialmente para descrição do fluxo de controle de programas, à sua época, Yourdon (1992) já observava que os analistas utilizavam os fluxogramas como um meio para documentar as especificações de processos. O diagrama de atividades também possui primitivas inspiradas nas redes de Petri (Machado et al., 2005; Xexéo, 2007), e pode ser utilizado tanto para modelar processos de negócio quanto para especificar métodos, operações e algoritmos complexos (Travassos, 2008). A OMG2 (2009) estabelece os seguintes contextos passíveis de aplicação do modelo de atividades:

- Descrição de computação procedural, representando métodos correspondentes a operações sobre classes;
- Modelagem de sistemas de informação para especificar processos no nível de sistema;
- Modelagem organizacional para engenharia de processo de negócio e *workflow*.

Enquanto a descrição procedural é voltada paro o paradigma de desenvolvimento orientado a objetos, a especificação de processos no nível de sistemas é independente de paradigma. Neste contexto, Pastor et al. (2001) apresentam o método de desenvolvimento de aplicações Web OOWS, em que os diagramas de caso de uso não são utilizados como referência, mas também preconizam o diagrama de atividades como modelo para especificação dos processos do sistema. Já Almendros-Jiménez e Iribarne (2004) propõem um método para descrição de Casos de Uso, que basicamente consiste em três etapas: identificação dos atores e dos casos de uso; descrição dos casos de uso com o Diagrama de Atividades; identificação de relacionamentos nos casos de uso (inclusão e generalização) por meio de diagramas de atividades.

Massollar (2008) propõe uma abordagem em que cada diagrama de atividades descreve um caso de uso, apresentando 15 regras para construção do Diagrama de Atividades, baseada em regras para a construção de casos de uso, tais como: cada passo do caso de uso deve estar associado a uma ação atômica de um Ator ou do Sistema; o diagrama deve ser composto das ações do Ator e do Sistema, com o devido fluxo de controle e pontos de decisão. A Figura 1.1 apresenta um exemplo de

caso de uso descrito através do diagrama de atividades, apoiado pela utilização de estereótipos específicos da abordagem proposta, apresentados na Tabela 1.1.

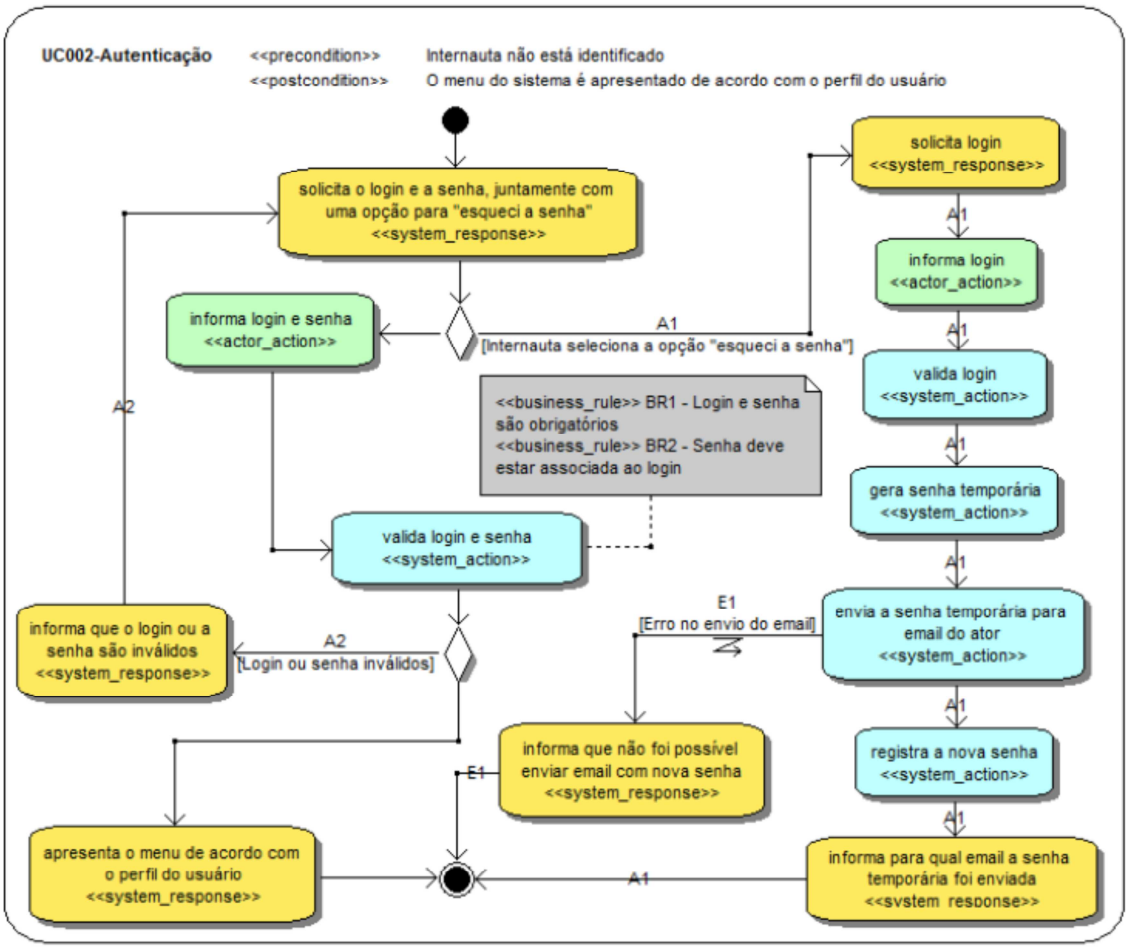

Figura 1.1. Exemplo de caso de uso descrito através de um diagrama de atividades, conforme Massollar (2008).

Tabela 1.1. Lista de estereótipos para descrição de casos de uso.

Estereótipo	Objetivo
<<include>>	Representa o relacionamento de inclusão de caso de uso
<<extend>>	Representa o relacionamento de extensão de caso de uso
<<system action>>	Representa uma ação interna do sistema
<<actor action>>	Representa uma ação do ator
<<system response>>	Representa uma ação de resposta do sistema
<<business rule>>	Indica que uma regra de negócio é referenciada

Já a abordagem apresentada por Pereira et al. (2010), tem como objetivo apoiar a concepção de *workflows* científicos, que é uma tecnologia usada para automatizar procedimentos experimentais baseados em simulação e dependentes de recursos computacionais. Nesta abordagem, a concepção de um experimento é representada através de uma especificação, composta por modelos descritos por

diagramas de atividades e formulários que capturam informações e restrições dos procedimentos experimentais considerando três tipos de elementos: atividade, artefato e ferramenta.

Cada *workflow* científico pode sofrer refinamentos sucessivos, sendo representado em diagramas de atividades em níveis de detalhes distintos. Pereira et al. (2010) classificam o *workflow* no maior nível de abstração de *workflow abstrato* e, no menor nível de abstração, de *workflow concreto*. Entre estes dois níveis, pode haver diversos *workflows intermediários de abstração.* A Figura 1.2 apresenta um exemplo de *workflow* descrito através desta abordagem. O oráculo, ou seja, a base para desenho do diagrama de atividades virá da descrição textual dos requisitos em linguagem natural.

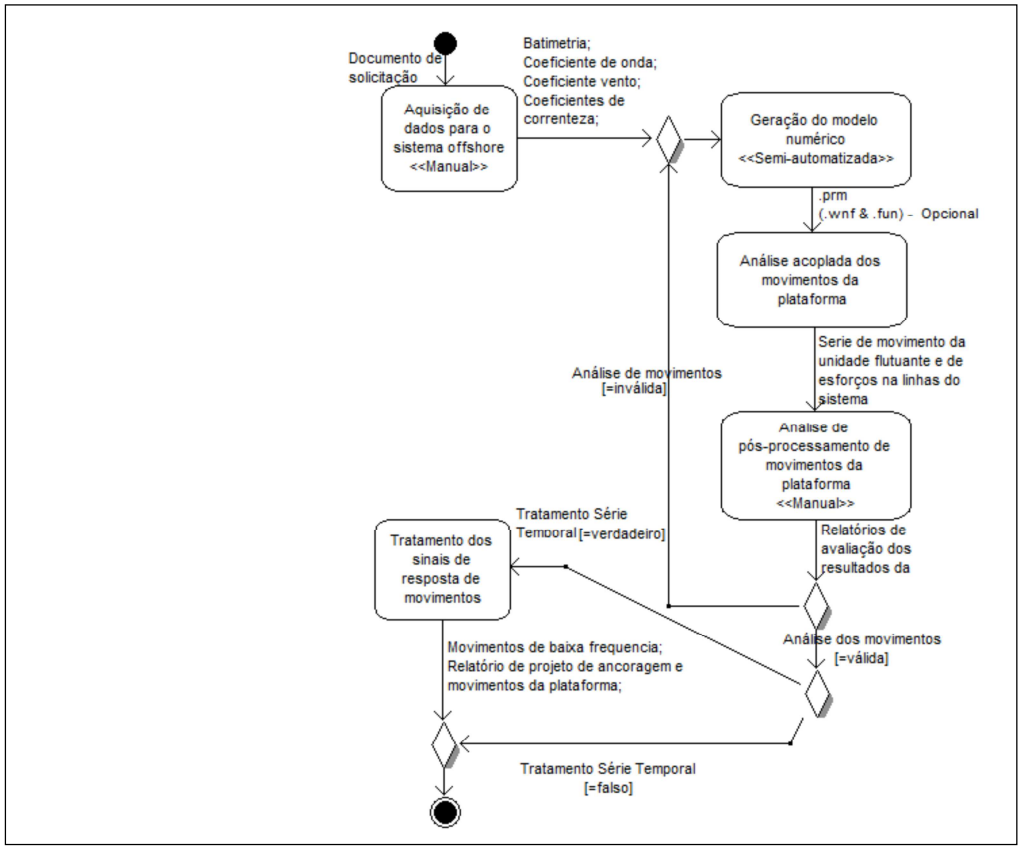

Figura 1.2- Exemplo de *workflow* científico descrito através de um diagrama de atividades.

Na última década, também é perceptível o esforço significativo da comunidade científica no aprimoramento de atividades que promovam a garantia da qualidade do produto final não só através da verificação do código-fonte, mas também dos demais artefatos elaborados ao longo do processo de desenvolvimento. A garantia da qualidade do produto de software é o objetivo principal das atividades de VV&T

(Verificação, Validação e Teste), onde se inclui a Inspeção de Software, uma atividade revisional que busca identificar defeitos nos mais diversos artefatos de software a fim de prevenir a ocorrência de falhas. Devido ao potencial de redução de esforço para um projeto de software, diferentes técnicas para inspeção, mais especificamente da especificação de requisitos e de modelos iniciais dos projetos de software têm sido propostas e aplicadas em estudos experimentais. Entretanto, uma *quasi*-revisão sistemática conduzida recentemente (de Mello et al., 2010) não identificou tecnologias que explorassem suficientemente a inspeção de diagramas de atividades.

Técnicas que apoiem a detecção de defeitos em diagramas de atividades e, em especial, técnicas para a inspeção de diagramas de atividades em especificação de requisitos, também são uma necessidade relacionada a dois temas de pesquisa do Grupo de Engenharia de Software Experimental da COPPE. No primeiro tema, Massollar (2008) propõe um processo para a engenharia e garantia da qualidade de aplicações Web, que prevê a inspeção dos modelos desenvolvidos na fase de especificação de requisitos, mais especificamente o diagrama de atividades, que deve estar em conformidade com a descrição textual dos requisitos.

A segunda pesquisa diz respeito ao desenvolvimento de uma abordagem para apoiar a concepção de *workflows* científicos em experimentos em Engenharia de Software, mais especificamente, experimentos *in virtuo* e *in silico* de larga escala (Pereira et al., 2009). Neste contexto, o diagrama de atividades será utilizado para representação dos *workflows* científicos em diversos níveis de abstração, que também devem ser inspecionados sob o ponto de vista da especificação textual dos requisitos.

A motivação pessoal para atuar nesta pesquisa baseia-se na percepção da importância das atividades de inspeção na promoção da qualidade do produto de software e na redução do esforço dispendido ao longo do processo de desenvolvimento. Esta percepção baseia-se não somente na revisão da literatura especializada e do conhecimento teórico adquirido, mas também em experiências vividas durante o curso de mestrado, participando de estudos envolvendo a aplicação das técnicas de leitura OORTs (Travassos et al., 1999) e OO-PBR (Mafra e Travassos, 2006) em projetos reais, bem como na realização de inspeção de artefatos de requisitos em projetos na indústria.

1.2 Trabalhos Relacionados

Além dos trabalhos de Massollar (2008) e Pereira et al. (2010), é possível encontrar outros trabalhos que propõem a utilização do diagrama de atividades na especificação de requisitos, tais como Pastor et al. (2001), Almendros-Jiménez e

Iribarne (2004) e Gutiérrez et al.(2008). Sobre a utilização de outros modelos de *workflow* na especificação de requisitos, Shinha e Paradkar (2010) apresentam uma proposta para a utilização de um modelo BPMN (Business Process Modeling Notation), linguagem de modelagem estruturalmente semelhante ao diagrama de atividades para representar cada caso de uso. Já Gross e Doer (2009) apresentam dois experimentos conduzidos para avaliar a utilidade de duas notações gráficas na engenharia de requisitos: o diagrama de atividades e o EPC (Event Process Chains). Através destes experimentos, os autores concluíram que, sob o ponto de vista do engenheiro de requisitos, o diagrama de atividades apresenta um desempenho melhor para a especificação de requisitos do que o EPC. A aplicabilidade do diagrama de atividades na representação de *workflows* é tratada em Störrle (2004), Wohed et al. (2005) e Wynn et al. (2009).

Em Shull et al. (2000), Mafra e Travassos (2006) e Belgamo e Fabbri (2004), são apresentadas técnicas para a inspeção da especificação de requisitos (textual) através de perspectivas, mas sem considerar a inspeção de modelos que descrevam estes requisitos. Para a inspeção de modelos, destacamos OORTs (Travassos et al. 1999), que é composto por diversas técnicas de leitura que orientam a verificação da rastreabilidade entre alguns modelos UML, não estando incluído o diagrama de atividades.

No que se refere à revisão de modelos de *workflow*, tais como o diagrama de atividades, são apresentadas na literatura diversas propostas para a verificação de regras sintáticas na elaboração destes modelos (Eshuis e Wieringa, 2004; Choi e Watanabe, 2005; Guelfi e Mammar, 2005; Xu et al., 2006; Qian et al., 2007), sem dar suporte à identificação de defeitos semânticos, ou seja defeitos que dependam de uma interpretação contextual, diferentemente dos defeitos sintáticos, que podem ser facilmente detectados com apoio de ferramentas automatizadas, através da verificação de regras pré-definidas. Não obstante, Tanriöver e Bilgen (2007) sugerem uma abordagem para a elaboração de técnicas de inspeção *on the fly* dos artefatos conforme o contexto de cada projeto, apresentando exemplos referentes à inspeção do diagrama de atividades sem, no entanto, explorar diversos recursos do modelo de atividades. No contexto da solução proposta, de Mello et al. (2010) apresentam a versão inicial da técnica de inspeção apresentada nesta dissertação e uma prova de conceito referente à aplicação desta técnica.

1.3 Objetivo da Solução Proposta

Esta dissertação de mestrado tem como objetivo apresentar ActCheck, uma técnica para inspeção de diagramas de atividades configurável e baseada em *checklists*. A técnica proposta é considerada configurável na medida em que determinados itens de avaliação (questões) que compõem seus *checklists* podem ser desconsiderados conforme o contexto de cada projeto, sendo que a técnica dá suporte para a configuração destes *checklists* através de um questionário de caracterização e de tabelas de configuração, relacionando os itens de avaliação às repostas do questionário.

ActCheck tem como foco principal a identificação de defeitos semânticos entre a descrição textual dos requisitos e um ou mais diagramas de atividades que descrevem graficamente estes requisitos com base no conceito de defeito definido pela norma IEEE (1998) e na categorização de defeitos apresentada em Travassos (1999): omissão, fato incorreto, inconsistência, ambiguidade e informação estranha. Através dos itens de avaliação, ActCheck busca a verificação da consistência, da completude e da conformidade do diagrama de atividades em relação aos requisitos de software.

Adicionalmente, a técnica considera em seus itens de avaliação a verificação da consistência interna do diagrama de atividades e a da consistência entre diagramas de atividades. Além dos componentes originais do diagrama de atividades da UML 2.2, a utilização dos estereótipos apresentados por Massollar (2008) para a descrição de casos de uso também são abordados nos *checklists* da técnica. Entretanto, é importante observar que não faz parte do escopo de ActCheck:

- A verificação sintática dos modelos inspecionados;
- O suporte a identificação de defeitos entre o diagrama de atividades e outros modelos UML ou qualquer outro artefato que não seja a especificação de requisitos do sistema;
- A inspeção de diagramas de atividades que utilizem recursos sintáticos descartados a partir da UML 2;
- Prover suporte para a definição de outros fatores que possam contribuir para a qualidade da inspeção, tais como critérios para seleção de inspetores e critérios para reinspeção.

1.4 Estruturação da Pesquisa

Para apoiar o desenvolvimento de uma técnica para inspeção do Diagrama de Atividades, seguimos a abordagem baseada em evidências para definição de novas tecnologias de software, apresentada em (Mafra et al., 2006). Tal abordagem consiste numa extensão da proposta de Shull et al. (2001) para introdução de tecnologias de software na indústria, aplicada em pesquisas realizadas pelo grupo Engenharia de Software Experimental da COPPE/UFRJ (Mafra et al., 2006; Conte et al., 2007; Massollar, 2008). Conforme pode ser visualizado na Figura 1.3, Inicialmente, para que a definição de novas tecnologias esteja baseada em evidências da literatura, devem ser executados estudos secundários, mais precisamente revisões sistemáticas. Após esta atividade, é elaborada a versão inicial da tecnologia, baseando-se nos resultados obtidos nos estudos secundários. Com a versão inicial da tecnologia definida, são conduzidos os seguintes estudos primários diversos, nesta ordem:

- Estudos de viabilidade, para determinar a usabilidade e a eficiência da tecnologia proposta;
- Estudos de observação, para melhorar a compreensão da tecnologia, buscando meios de torná-la mais eficiente;
- Estudos de caso em um ciclo de vida de desenvolvimento real, caracterizando a aplicação da tecnologia neste contexto;
- Estudos de caso na indústria, para identificar se a aplicação da tecnologia proposta integra-se ao ambiente industrial.

No escopo dessa dissertação de mestrado, a abordagem baseada em evidências foi parcialmente aplicada. Primeiramente, foi conduzida uma *quasi*-revisão sistemática para identificação de técnicas para inspeção de modelos de *workflow*, apresentada no Capítulo 4. A partir dos resultados desta revisão e de uma breve revisão de literatura prévia, foi elaborada a primeira versão da técnica de inspeção, que foi submetida a uma prova de conceito. Após os resultados positivos da prova de conceito, a técnica sofreu modificações significativas na sua estrutura, que deram origem a segunda versão da técnica, que recebeu o nome de ActCheck. Em seguida, foi conduzido um estudo experimental *in vitro* sobre a viabilidade de ActCheck, cujos resultados são apresentados e analisados no capítulo 6.

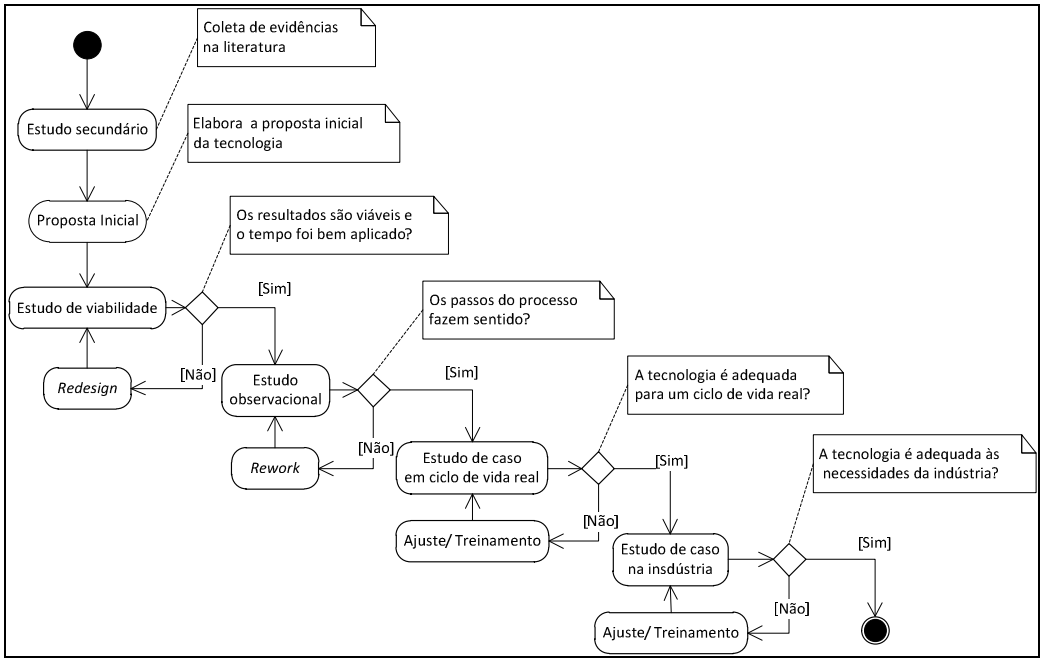

Figura 1.3- Abordagem para definição de novas tecnologias baseadas em evidências, adaptado de Mafra et al. (2006)

1.5 Organização do Trabalho

Além deste capítulo introdutório, a dissertação está organizada em outros seis capítulos. O capítulo dois aborda a primeira parte do referencial teórico: o diagrama de atividades. São apresentados os diversos recursos sintáticos que compõe este modelo, bem como são feitas considerações sobre a sua aplicabilidade, mais especificamente no que tange à especificação de requisitos e representação de fluxos de trabalho (*workflow*) em geral. Em adição, este capítulo apresenta os *workflow patterns,* que também fazem parte do material estudado para a composição de ActCheck.

O capítulo três aborda a segunda parte do referencial teórico: inspeção de software. É apresentado neste capítulo o processo de inspeção de software, a classificação de defeitos e conceitos relacionados à classificação e organização das técnicas de inspeção, mais especificamente técnicas para inspeção de modelos. Ao fim deste capítulo, são apresentadas os exemplos de técnicas para inspeção de modelos: OORTs (modelos UML) e ArqCheck (modelos arquiteturais).

O capítulo quatro trata da *quasi*-revisão sistemática que foi conduzida para identificar a literatura relevante sobre o tema de pesquisa, buscando seguir o primeiro passo para a definição de novas tecnologias baseada em evidências. O conceito de *quasi*-revisão sistemática é introduzido e, em seguida, os principais aspectos do protocolo da revisão são apresentados e o resultado obtido analisado.

O capítulo cinco apresenta a técnica para inspeção de diagramas de atividades, descrevendo sua primeira versão, que foi submetida a uma prova de conceito, e apresentando com detalhes ActCheck, a segunda versão da técnica de inspeção. São apresentados os artefatos que compõem ActCheck, tornando-a configurável e baseada em *checklist*, e o conceito de casos discrepantes, que deram origem aos itens de avaliação da técnica de inspeção.

O capítulo seis trata do *quasi*-experimento conduzido *in vitro* para avaliar a viabilidade ActCheck e comparar seus resultados da aplicação da técnica proposta , em projetos de software reais com inspeções *ad hoc*. O planejamento do estudo é apresentado, bem como os resultados obtidos são analisados.

Por fim, o capítulo sete apresenta as conclusões deste trabalho, destacando as considerações finais, os resultados obtidos e as contribuições identificadas para pesquisa. São mencionadas também as limitações da pesquisa e propostas de pesquisas futuras.

CAPÍTULO 2 - DIAGRAMA DE ATIVIDADES

Neste capítulo, são apresentados os diversos recursos sintáticos que compõem o modelo de atividades da UML, bem como são feitas considerações sobre a aplicabilidade dos diagramas de atividades, mais especificamente no que diz respeito à especificação de requisitos de software e representação de fluxos de trabalho (workflow) em geral. Em adição, o capítulo também apresenta os workflow patterns, que também fazem parte do referencial teórico relacionado à pesquisa.

2.1 Introdução

A UML (*Unified Modeling Language*) tem como objetivo prover aos profissionais envolvidos com o processo de desenvolvimento ferramentas para análise, projeto e apoio à implementação de sistemas baseados em software bem como para modelagem de negócios e processos similares (OMG2, 2009). Busca-se, com a UML, prover uma interoperabilidade entre ferramentas de modelagem visual de objetos, através de uma linguagem de propósito geral que possa ser utilizada em todos os domínios de aplicação.

A UML possui modelos para representação estática (estrutural) e para a representação dinâmica (comportamental) dos sistemas. No caso da representação dinâmica, a UML possui três modelos comportamentais: atividades, máquinas de estado e interações. O foco do modelo de atividades está na sequência, nas condições e nas entradas e saídas para invocar outros comportamentos. Já as máquinas de estado mostram como os eventos causam mudanças no estado do objeto e invocam outros comportamentos, enquanto as interações descrevem a troca de mensagens entre os objetos que causam a invocação de outros comportamentos (Bock, 2003).

Para Machado et al. (2005), a UML permite uma modelagem padrão sobre múltiplos pontos de vista, sendo que a modelagem das atividades de um sistema através do Diagrama de Atividades suporta o ponto de vista funcional, representando os processos do sistema. Entretanto, até a versão 1.5 da UML, o Diagrama de Atividades possuía restrições semânticas expressivas, e era considerado confuso pelos usuários, resumindo-se a uma forma de máquina de estados com notação modificada para assemelhar-se a um diagrama de fluxo.

A partir da UML 2.0 ocorreu a redefinição do modelo de atividades, estendendo as suas capacidades e permitindo o suporte à modelagem de fluxos sobre uma larga variedade de domínios, do computacional até o físico (Bock, 2003). O novo Diagrama de Atividades pode ser utilizado para especificar sistemas em geral e sistemas de software, independentemente de como será sua implementação, não se limitando somente ao paradigma da orientação a objetos. Atualmente a UML encontra-se na versão 2.2 (OMG, 2009).

2.2 O Modelo de Atividades

A UML 2.0 organiza os recursos sintáticos do modelo de atividades nos pacotes apresentados na Figura 2.1. O pacote de atividades básicas contém os recursos que são considerados comuns a qualquer aplicação, tais como os conceitos básicos de ação, fluxo de dados e de controle, e de entrada/ saída, que são a raiz das atividades. A partir daí, podem ser seguidas duas direções: o pacote das atividades estruturadas, para modelagem de software ou os pacotes de atividades intermediárias e completas, para modelagem de processos em geral. Cada pacote de atividades possui seus construtos específicos, mas este direcionamento é ortogonal, podendo haver uma combinação entre os recursos sintáticos utilizados (Bock, 2009).

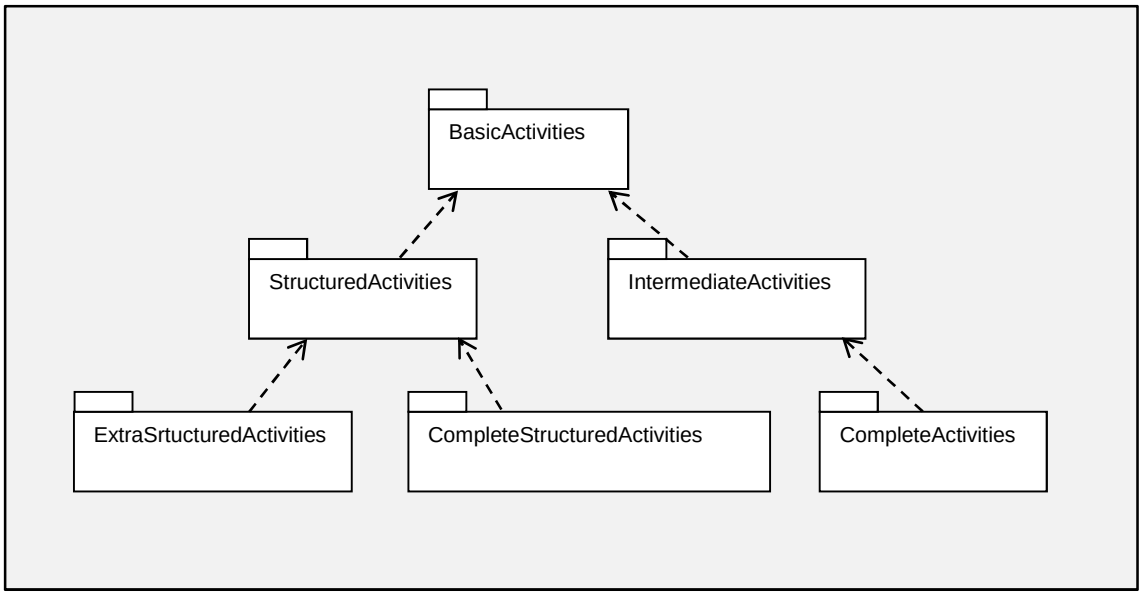

Figura 2.1. Os pacotes que compõem o modelo de atividades, baseado em (Bock, 2003)

2.2.1 Ações

As ações são frequentemente consideradas os elementos “primitivos” da UML, por serem os únicos elementos da UML que podem requisitar objetos, terem um efeito persistente neles, e invocar diretamente comportamentos e operações. Uma ação não pode ser considerada por si só um comportamento porque se entende que os comportamentos são definidos pelo usuário, enquanto que as ações são definidas no modelo. A ação é a base para todos os modelos comportamentais da UML (Bock2, 2003).

Na UML, uma ação só pode ser diretamente contida no modelo de atividades (Bock2, 2003), representando um passo simples que não é mais decomposto dentro do Diagrama de Atividades que o contém, mas podendo implicar em muitos outros passos detalhados, em outros níveis de abstração. Uma ação também pode representar uma chamada para outra atividade, o que favorece a reutilização (Xexéo, 2007). Todas as ações são pré-definidas, sendo que algumas delas invocam comportamentos definidos pelo usuário, incluindo outras atividades. São exemplos de ações (Bock1, 2003):

- Invocação de comportamentos definidos pelo usuário
- Criação de objetos;
- Definição de valor de atributos;
- Ligação entre objetos

Uma ação é representada por um retângulo arredondado, que pode ser ligada a outra ação através de setas que representam o fluxo de controle de uma atividade. Uma ação pode conter objetos de entrada e de saída para representar o fluxo de dados, bem como pré e/ou pós-condições (chamadas de pré e pós-condições locais) para sua realização (Bock2, 2003). A Figura 2.2 apresenta um exemplo de como uma ação “Entrega bebida”, que contém um objeto de entrada, um objeto de saída e pré e pós-condições locais pode ser representada.

Cada ação que compõe uma atividade é considerada um nó da atividade, sendo conectada a outro nó através de suas arestas de modo a formar um grafo de fluxo completo (Bock2, 2003). Deste modo, pode-se dizer que a execução de uma atividade depende fundamentalmente da execução das ações contidas nesta atividade (OMG2, 2009).

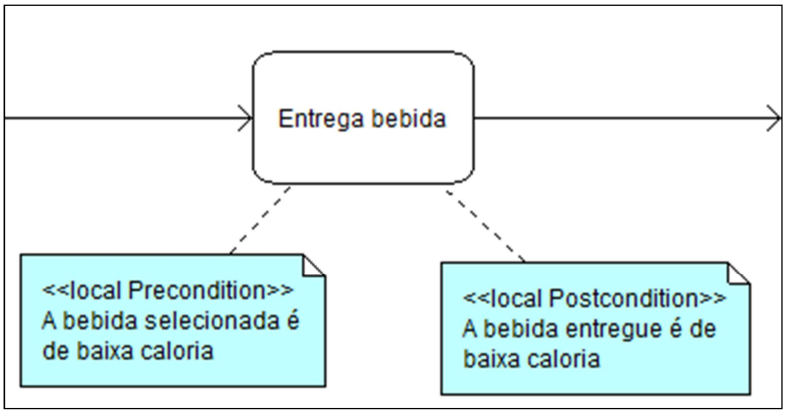

Figura 2.2. Exemplo de uma ação com pré e pós-condições locais

A UML trata ainda com representações específicas as ações de envio e recebimento de sinais de uma atividade, que podem ser visualizadas na Figura 2.3, itens a e b, respectivamente. No caso de recebimento de sinal baseado em tempo, é utilizada uma representação especial (item c). Importante observar que as relações de envio/recebimento de sinal podem envolver mais de uma atividade.

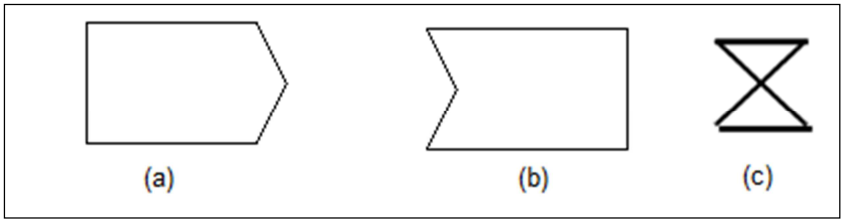

Figura 2.3. Representações para ações de envio de sinal (a) e recebimento de sinal (b) e (c), baseado em OMG (2009).

2.2.2 Nós de Controle

Além dos nós de ação, existem mais dois tipos de nós de atividades: os nós de controle e os nós de objeto. Ao contrário do que o nome sugere, os nós de controle orientam não somente o fluxo de controle, mas também o fluxo de dados, consistindo em sete tipos, com cinco representações distintas, apresentadas na Figura 2.4.

O nó de início de atividade é único para cada atividade e o fluxo de controle parte deste nó para o primeiro recurso sintático a ser executado numa atividade. Em contrapartida, o nó de fim de atividade deve aparecer uma ou mais vezes numa atividade, indicando que a execução de uma atividade foi finalizada. Já os nós de fim de fluxo permitem que ocorra, por exemplo, um desvio para fora de determinado

trecho da atividade que seja feito repetidamente até que determinada condição seja alcançada sem, no entanto, finalizar a atividade.

O nó de decisão deve ser utilizado quando se deseja optar por um único fluxo dentre um ou mais fluxos alternativos. Para cada fluxo alternativo, deve haver uma condição de guarda, definindo os critérios que precisam ser atendidos para que o fluxo de controle da atividade siga este fluxo. Para reunir um ou mais fluxos alternativos, é utilizado um nó de *merge*, que podem representar ao mesmo tempo um nó de decisão, estando ligado a um ou mais fluxos alternativos com suas respectivas condições de guarda.

O nó de bifurcação (*fork*) permite que dois ou mais fluxos de uma atividade sejam executados de modo concorrente. Entretanto, é importante observar que, em uma bifurcação, também pode haver condições de guarda restringindo a execução de cada fluxo. Fluxos executados em paralelo podem ser reunidos através de um nó de junção (*join*), que pode apresentar condições para que esta junção ocorra, representadas através do recurso *JoinSpec*.

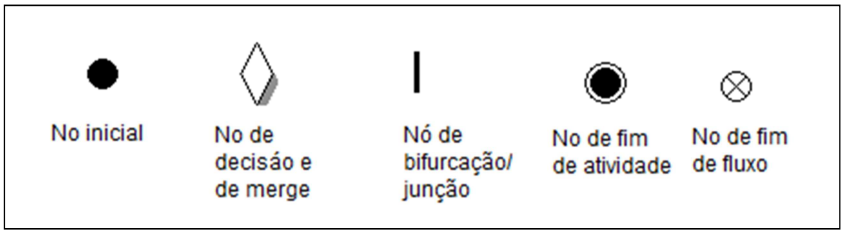

Figura 2.4. Nós de controle, baseado em (Bock3, 2003)

A Figura 2.5 apresenta um pequeno exemplo de um diagrama de atividades em que quatro das cinco notações apresentadas são aplicadas. A partir da execução da primeira ação (1), ocorre uma bifurcação (2), permitindo que paralelamente uma decisão seja tomada (3) enquanto o problema relatado é corrigido (4). A partir da decisão tomada, dois fluxos podem ser seguidos, mas, ao final, é feito um *merge* (5) para o mesmo ponto. Paralelamente, após o problema ser consertado, é feito o teste do conserto (6). É feito, então, um junção entre os dois fluxos que estavam sendo executados paralelamente (7), o que faz a atividade voltar a ter um só fluxo, contendo a ação que consiste na entrega do conserto (8). Após esta ação, a atividade é finalizada.

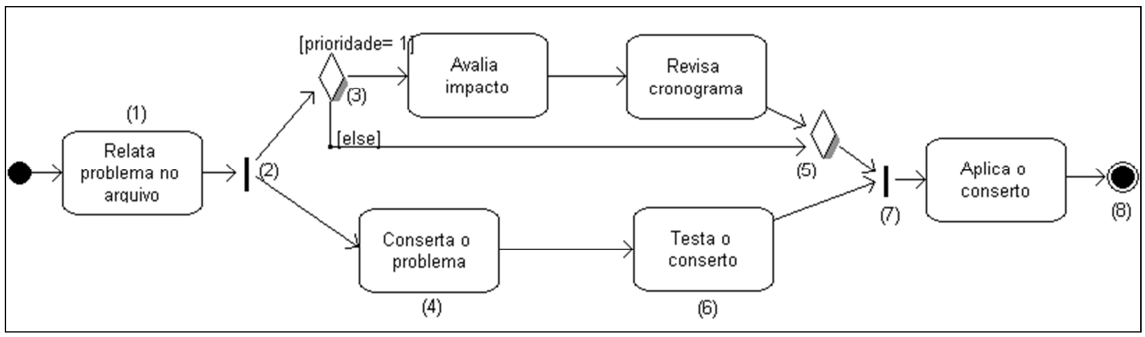

Figura 2.5. Exemplo de Diagrama de Atividades, adaptado de (Bock3, 2003)

No exemplo da figura 2.6, é feita uma bifurcação sem posterior junção, gerando dois fluxos concorrentes que levam a duas formas distintas de concluir a atividade. No primeiro fluxo, o cliente aguarda receber a confirmação do itinerário escolhido enviado, para efetuar a reserva. O segundo fluxo só ocorrerá caso esta confirmação não chegue em 48 horas, levando ao cancelamento do itinerário.

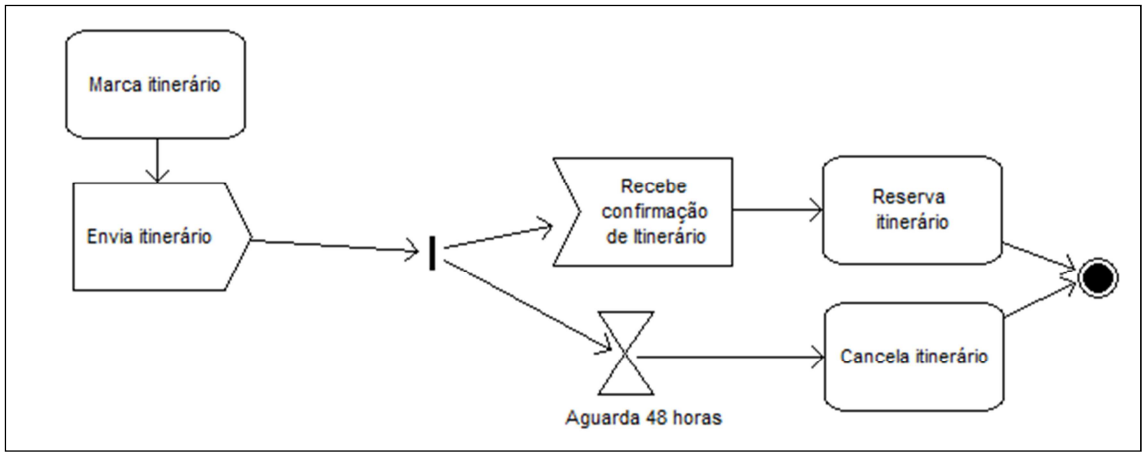

Figura 2.6. Exemplo de diagrama de atividades com ações de envio/ recebimento de sinal e bifurcação. Adaptado de Fowler (2004).

2.2.3 Nós de Objeto

Os nós de objeto manipulam temporariamente os dados produzidos e consumidos ao longo das ações de uma atividade enquanto estes dados esperam ser movidos ao longo de uma atividade (Bock, 2004), no que é conhecido como fluxo de objetos. Existem quatro tipos de nós de objeto, conforme pode ser visualizado na Figura 2.7:

- Parâmetros de atividades (A), que são nós que ficam no início e no fim dos

fluxos de uma atividade, provendo meios para aceitar as entradas de uma atividade e saídas resultantes da execução desta atividade (OMG, 2009)

- *Pins* (D), que representam uma entrada ou uma saída de uma ação (OMG, 2009), descrevendo os tipos de dados que fluem em uma ação (Bock, 2004);
- *Buffers* centrais (B), utilizados em situações onde *tokens* (valores de controle e de dados que fluem ao longo de uma atividade) em competição são provenientes de múltiplas fontes;
- Abastecimento (*store*) de dados (C), que suportam fluxo e armazenamento de dados passivo, não esgotável e persistente, ao contrário dos outros três tipos, que suportam fluxo e armazenamento de dados ativo, esgotável e transiente.

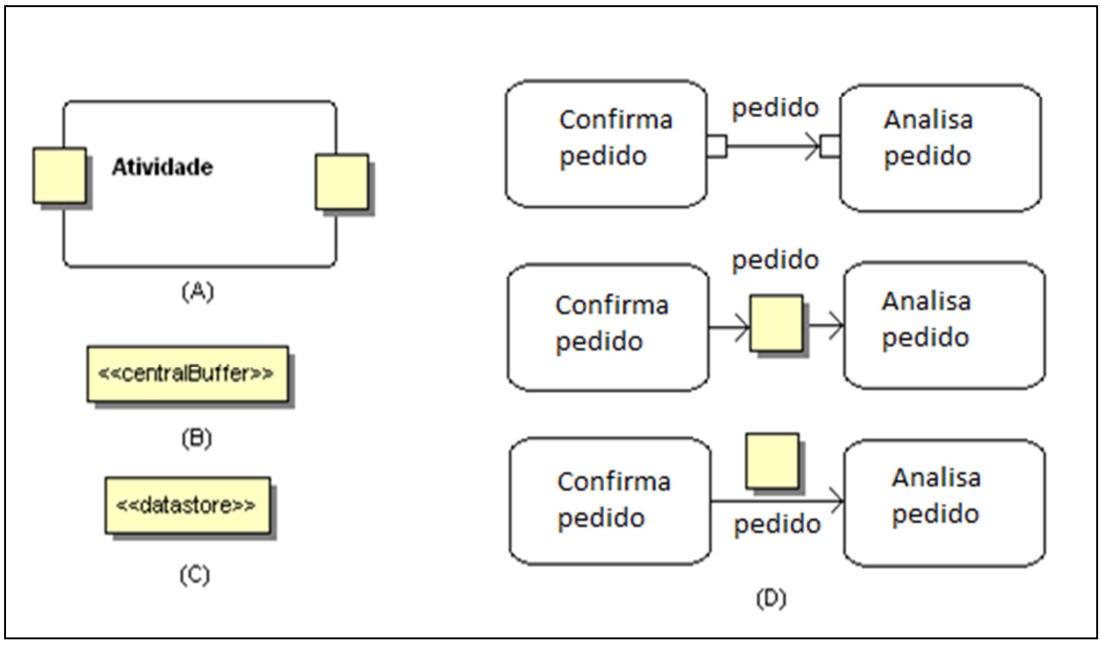

Figura 2.7. Nós de objeto, baseado em (Bock, 2004)

Cada nó de objeto podem possuir diversos estereótipos que caracterizam um objeto ou que descrevem uma transformação entre objetos (OMG, 2009). São eles:

- Ordenação (*ordering*): define os critérios para ordenação das instâncias (*tokens*) de um mesmo nó de objeto
- Estado (*inState*): define os possíveis estados de um nó de objeto em determinado ponto da atividade;
- Seleção (*selection*): estabelece critérios para seleção de objetos;

- Limite Superior (*upperBound*): estabelece o número máximo de instâncias permitidas para um mesmo nó de objeto;
- Controle (*isControlType*): define quando o tipo de nó de objeto deve ser considerado como um nó de controle ou não;
- Transmissão múltipla (*isMulticast*) e recebimento múltiplo (*isMultireceive*): define quando os objetos são passados através de *multicasting* ou quando nós de objeto recebem a partir de multicasting;
- Transformação (*transformation*): indica a transformação de um nó de objeto transmitido em outro nó de objeto recebido.

A Figura 2.8 apresenta dois exemplos de aplicação de um dos tipos de nó de objeto, os *pins*. No primeiro exemplo, o estado do objeto "pedido" é definido como "preenchido" e a prioridade para seleção de um objeto de pedido é feita considerando-se o primeiro objeto inserido na pilha (FIFO) e a prioridade do pedido. Já no segundo exemplo, um objeto do tipo "pedido" é passado pela ação "Fecha pedido", mas a ação "Notifica clientes" espera um objeto do tipo "cliente", denotando uma transformação de objetos no fluxo de dados.

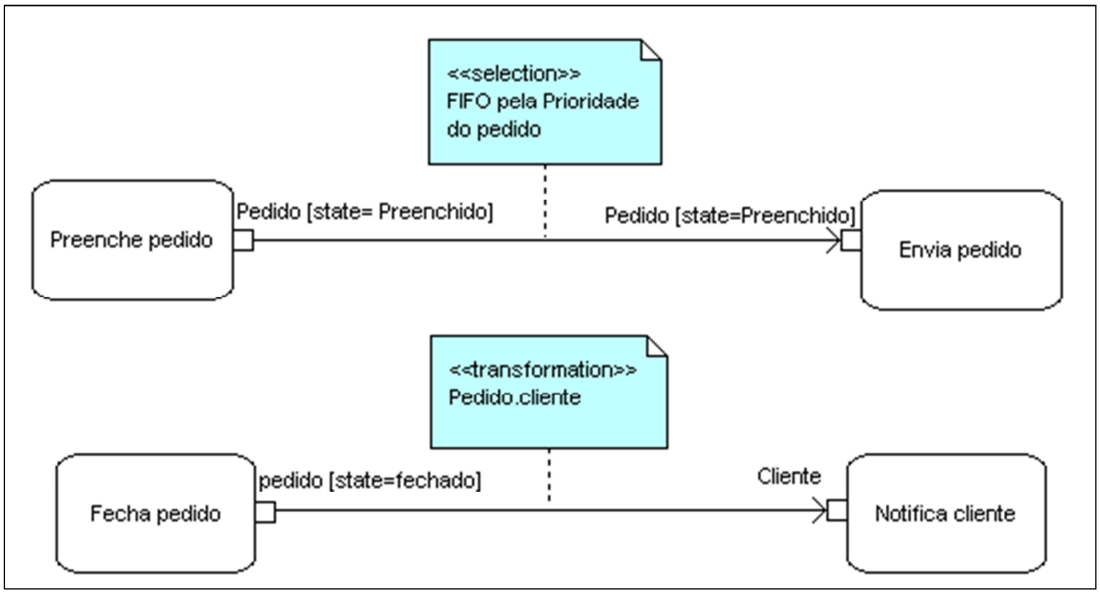

Figura 2.8. Exemplos de nós de objeto

Nós de objeto também podem ser encapsulados de modo a constituir um conjunto de parâmetros necessários para se executar uma ação ou uma atividade. Segundo especificado na UML (OMG, 2009), pode existir mais de um conjunto de parâmetros como *pin* de entrada ou de saída de uma ação, conforme é exemplificado

na Figura 2.9. Nesta atividade, o ajuste de salário é feito tanto para o empregado recém-contratado quanto para o empregado que já foi avaliado, sendo que cada contexto demanda um conjunto de objetos distinto para a mesma ação.

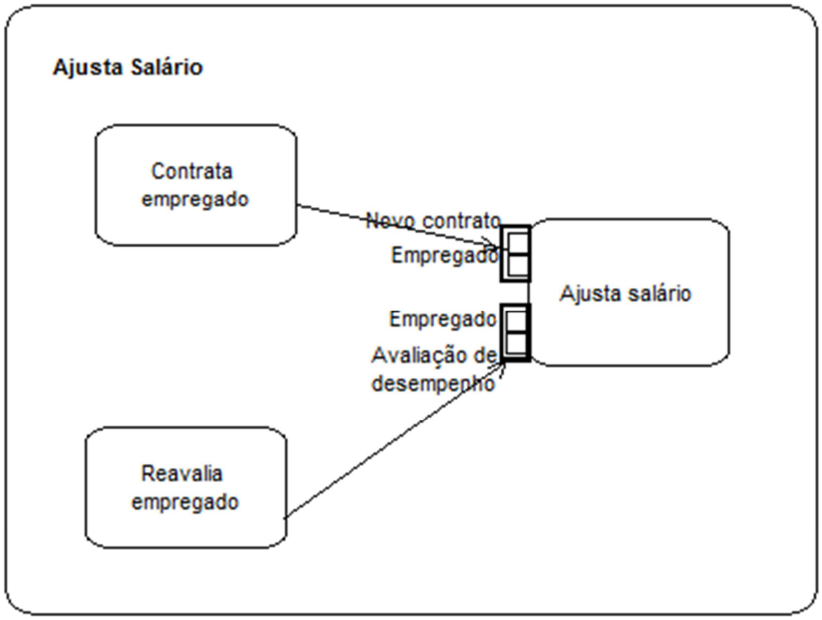

Figura 2.9. Exemplo de nós de objeto encapsulados. Adaptado de Pilone e Pitman (2005)

2.2.4 Raias

O modelo de atividades conta ainda com as partições, que consistem numa forma de agrupar as ações que possuem alguma característica em comum, tais como: um mesmo executante, uma mesma classe ou uma mesma propriedade (Bock2, 2004). Uma forma comum de representar as partições de um diagrama de atividades é através do uso de *swimlanes* (raias de piscina), que também podem ser subdivididas através de sub-raias, como pode ser visualizado no exemplo da Figura 2.10. Neste exemplo, são caracterizados dois executantes das ações da atividade: o cliente e a empresa. Entretanto, dentro da empresa, existem dois setores envolvidos na atividade: um setor que realiza ações específicas de avaliação de pedidos (setor de atendimento) e outro setor responsável pela geração de boletos (setor financeiro).

Outra possibilidade de aplicação de raias envolve a utilização de raias ortogonais que capturem duas dimensões de agrupamento numa mesma atividade. Podemos, por exemplo, agrupar os recursos de uma atividade em um conjunto de raias verticais que identifique quem executa cada ação ao mesmo tempo em que

agrupamos estes mesmos recursos um conjunto de raias horizontais que identifiquem as classes que são instanciadas ao longo da execução desta mesma atividade.

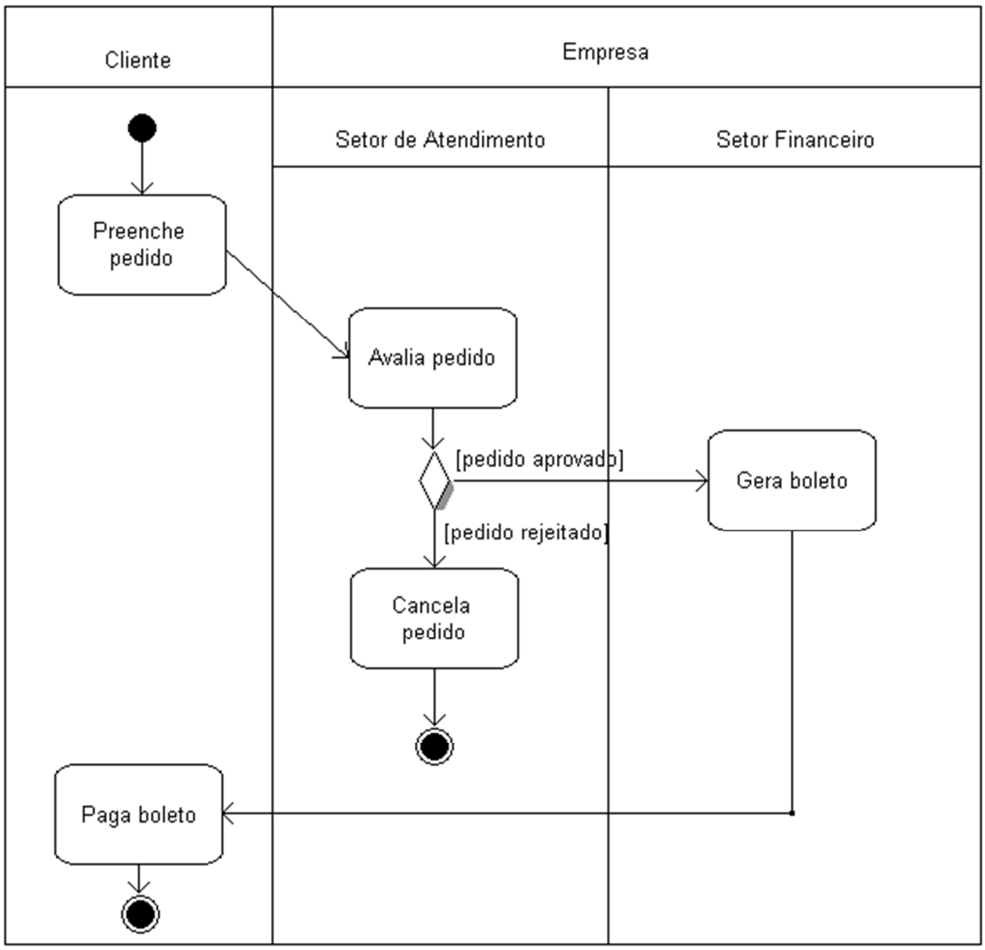

Figura 2.10. Exemplo de uso de raias. Baseado em (OMG, 2009)

2.2.5 Regiões de Expansão e de Interrupção

Outros recursos do Diagrama de Atividades incluem as regiões, formadas por trechos de atividade que precisam ser repetidos (regiões de expansão), ou por trechos de atividade que podem sofrer uma interrupção (regiões de interrupção). Uma região de expansão deve possuir um ou mais elementos de entrada (*inputElement*), que formam a coleção de dados de entrada que definem quantas vezes a região de expansão deverá ser executada. Uma região de expansão pode possuir também elementos de saída, que constituirão a coleção de objetos disponibilizada após conclusão da execução da região (OMG, 2009). As regiões de expansão podem ser classificadas através de um estereótipo que defina como as múltiplas execuções irão interagir:

- *iterative*- estabelece que todas as interações ocorrerão na ordem dos elementos da região;
- *paralell*- estabelece que todas as interações são independentes, podendo ou não ocorrerem paralelamente;
- *stream*- define que haverá uma única execução da região, recebendo a coleção de dados de entrada.

A Figura 2.11 apresenta um exemplo de região de expansão. Neste exemplo, um conjunto de testadores e um conjunto de módulos são passados para uma região de expansão. A cada iteração, é selecionado um testador e um módulo, a fim de testar cada módulo (ação). Esta seleção de um objeto "testador" e de um objeto "módulo" é feita obedecendo ao critério FIFO (*First In, First Out)* de ordenação. Após cada teste de módulo é disponibilizada uma planilha de teste. O conjunto destas planilhas está representado como saída da região de expansão.

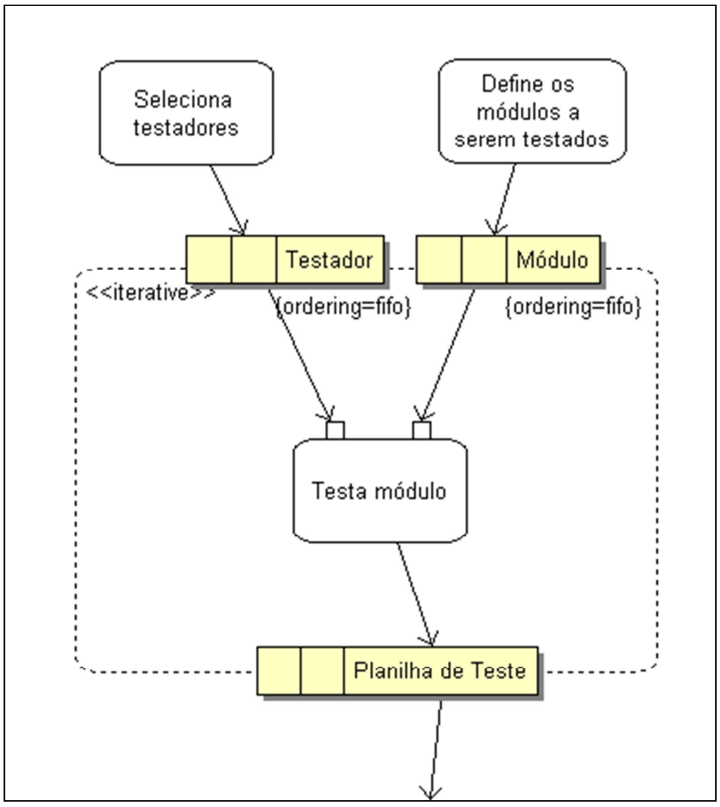

Figura 2.11. Exemplo de aplicação de uma região de expansão.

Uma região de interrupção estabelece que, dentro desta região, a atividade pode ser interrompida a qualquer momento, sendo redirecionada para outra parte da

atividade. O objetivo desta estrutura, definida a partir da UML 2.0 é flexibilizar o tratamento de terminações de fluxo não locais em um diagrama de atividades. Na Figura 2.12, por exemplo, podemos identificar que, ao longo do processo de recebimento e encaminhamento de um pedido, pode ser requerido o cancelamento deste pedido. Se isto acontecer, é gerada uma interrupção que redireciona o fluxo de controle da atividade para o cancelamento do pedido e, em seguida, para o fim da atividade.

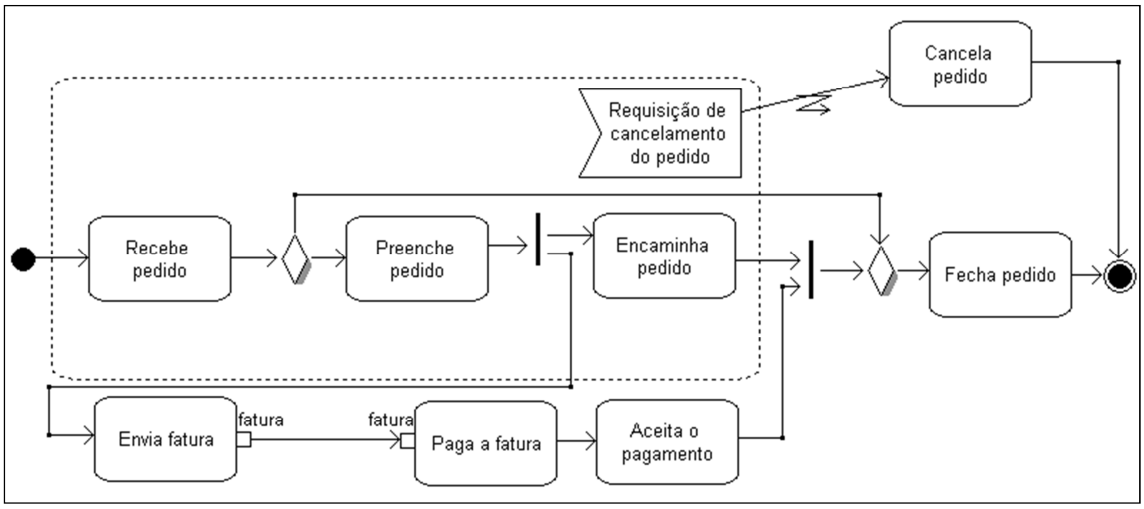

Figura 2.12. Exemplo de aplicação de uma região de interrupção.

2.2.6 Grupos de Atividade

Um diagrama de atividades pode ser empacotado total ou parcialmente em um grupo de atividade (*activity group*). Os grupos de atividade podem ser úteis, por exemplo, para caracterizar os nós protegidos, que são as áreas do diagrama de atividades que podem sofrer uma determinada exceção mapeada. No caso de uma atividade ser empacotada por um grupo, podem ser especificados os nós de objeto de entrada e de saída desta atividade, chamados de nós de parâmetro da atividade (OMG, 2009). A Figura 2.13 apresenta um exemplo de atividade agrupada, com os nós de entrada “Produtos” e “Cliente”. Como nó de saída, é passado o “númeroSedex”.

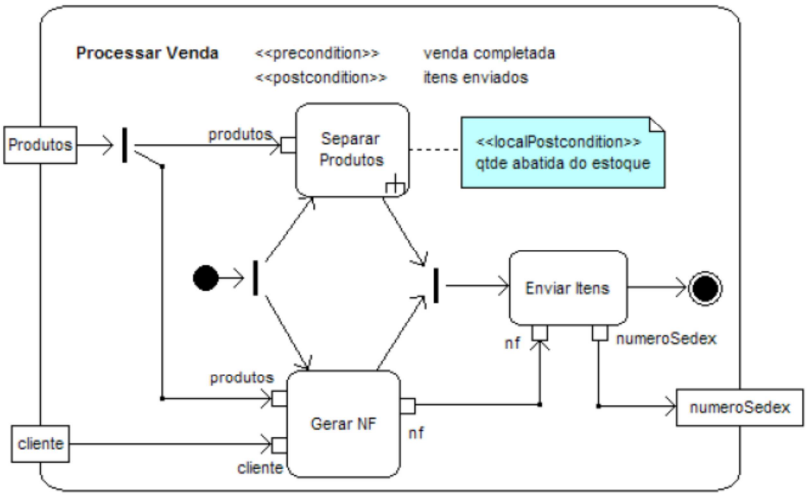

Figura 2.13. Exemplo de grupo de atividade.

2.2.7 Exceções

Em um diagrama de atividades, uma ação específica ou um grupo de atividade pode ser servido de tratamentos para eventuais exceções mapeadas. O grupo de uma atividade que possui tratamento para uma exceção é chamado de nó protegido (OMG, 2009). A Figura 2.14 apresenta um exemplo onde são tratadas duas exceções, ao longo de uma atividade envolvendo operações com matrizes. Neste exemplo, são tratadas duas exceções: “matriz singular” e “overflow”. Para cada exceção, é prevista a substituição de um dos vetores da matriz, dando continuidade à execução da atividade.

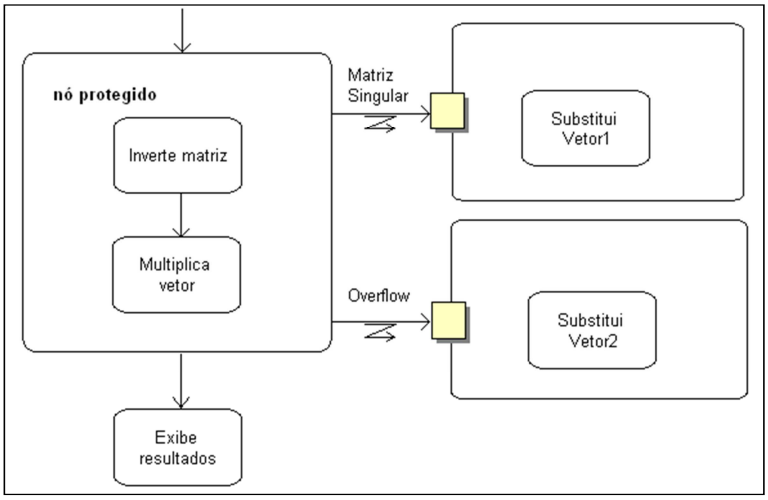

Figura 2.14. Exemplo de tratamento de exceção, baseado em OMG (2009).

2.2.7.1 Nós de *Loop* e Nó Condicional

Uma forma de representar repetições de um trecho de atividade é através de nós de *loop* (loop Node). Este nó corresponde a uma região dividida basicamente em três setores. O primeiro setor (*setup*) define os valores iniciais para a execução do que está definido no *body*, o terceiro setor. Já o segundo setor (*test*) comporta os elementos que determinam se a execução do "body" será realizada mais uma vez (OMG, 2009). A Figura 2.15 apresenta um exemplo de nó de *loop*.

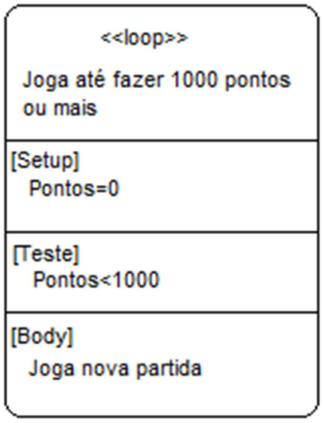

Figura 2.15. Exemplo de nó de loop.

Os nós condicionais são uma forma estruturada de representar decisões numa atividade. Cada nó condicional deve possuir ao menos uma sessão *body* e uma sessão *test*. Basicamente, o resultado das verificações contidas nas sessões *test* irão determinar qual dos *body* de um nó condicional será executado, ou se nenhum body será executado. A Figura 2.16 exemplifica a utilização de um nó condicional.

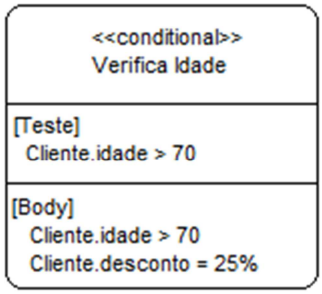

Figura 2.16. Exemplo de nó condicional.

2.2.8 Chamadas de Atividades

Uma ação de uma atividade pode representar uma chamada a outra atividade. Com esta chamada, o fluxo de controle passa da atividade chamadora para a atividade chamada que, ao concluir, sua execução, volta para a atividade chamadora. Na Figura 2.17, por exemplo, a atividade “Processar venda” em determinado momento, chama a atividade “Separar produto”. Essa atividade passa, então, a ser executada. Quando a ação “Separar produto” for concluída, o fluxo de controle retorna para “Processar venda”, do ponto onde parou, aguardando a junção ocorrer para executar a atividade “Enviar itens”.

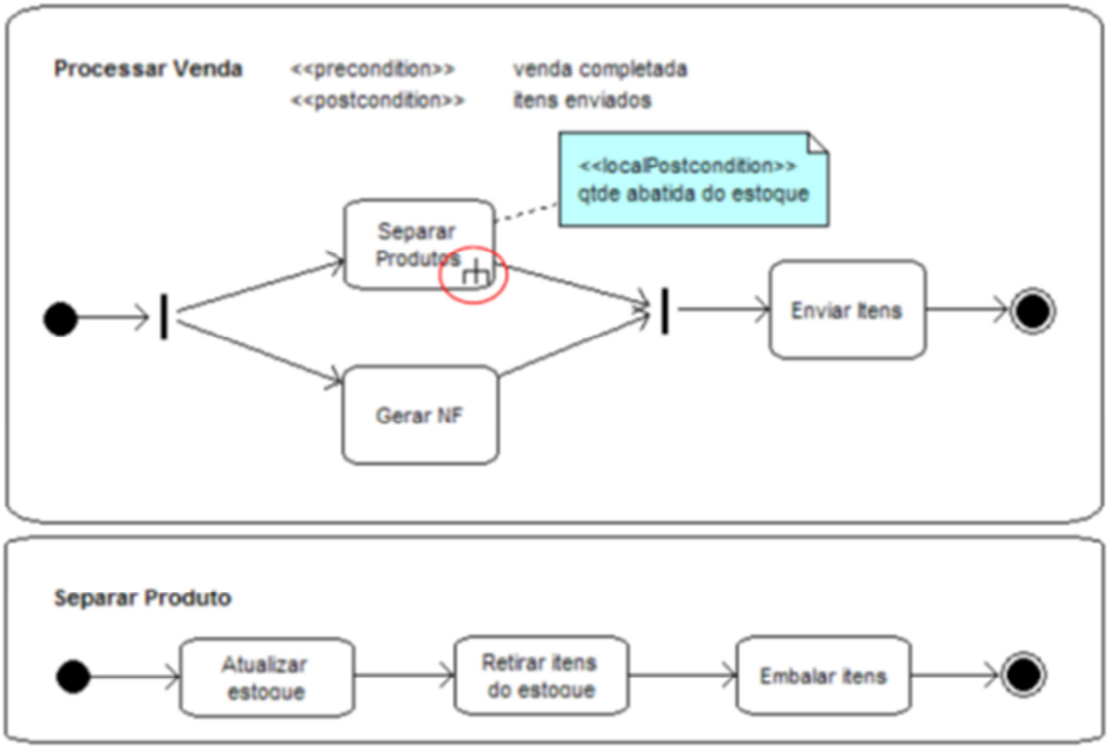

Figura 2.17. Exemplo de chamada de atividade.

2.3 Considerações Finais do Capítulo

Até a reestruturação do diagrama de atividades na UML 2.0, eram perceptíveis as limitações deste modelo para a descrição de processos de um sistema, mesmo no caso de projetos de sistemas orientados a objeto. Atualmente, os recursos sintáticos do diagrama de atividades permitem a representação de diversas situações que podem ser identificadas no fluxo de controle e de dados de *workflows* em geral.

Deste modo, a modelagem de atividades tornou-se uma opção para descrever graficamente a especificação de requisitos de projetos de software. A aplicação de

diagramas de atividades na especificação de requisitos pode contribuir para o entendimento de estruturas mais elaboradas de sistemas. Entretanto, há também o risco de surgimento de novos defeitos numa especificação, decorrentes dos diagramas de atividades elaborados.

Neste contexto, torna-se necessária a verificação da conformidade, a completude e a corretude dos diagramas de atividades com a descrição textual dos requisitos, que pode apresentar um nível de detalhamento maior ou menor que os diagramas, conforme a aplicação dos diagramas de atividades na técnica de modelagem adotada. De um modo geral, é importante verificar se o fluxo de controle e o de dados de uma atividade oferecem cenários coerentes com o escopo do software, sem omissões de informações necessárias para a completeza da atividade e sem informações inconsistentes. Também é importante que de cada nó e condição de um diagrama esteja descrita de modo legível e livre de informações estranhas, que possam prejudicar a interpretação do modelo e levar ao desenvolvimento de um software inconsistente com a especificação.

CAPÍTULO 3 - INSPEÇÃO DE SOFTWARE

Neste capítulo, é apresentado o processo de inspeção de software, a classificação de defeitos e conceitos relacionados à classificação e organização das técnicas de inspeção, mais especificamente no que diz respeito a técnicas para inspeção de modelos. Ao fim deste capítulo, são apresentadas as técnicas de inspeção OORTs e ArqCheck.

3.1 Introdução

O padrão IEEE 610.12 (1990) define inspeção como o exame visual de um produto de software que detecta e identifica anomalias de software, incluindo erros e desvios de padrões e de especificações. Entretanto, conforme a terminologia estabelecida neste padrão, as inspeções tendem a identificar defeitos e não erros pois, segundo esta norma, “falta” consiste em um passo, processo ou definição de dados incorretos, enquanto que “erro” consiste em qualquer estado ou resultado inesperado na execução de um programa. Neste sentido, Travassos (2001) acrescenta que as inspeções de software buscam defeitos que se referem à faltas, diferentemente dos defeitos encontrados nos testes, que se referem a falhas.

É importante observar que existe uma confusão na literatura quanto ao uso dos termos “revisão”, “inspeção” e “*walkthrough*” (Wong, 2006). A revisão propriamente dita pode ser considerada um processo ou reunião em que um produto de software é apresentado ao pessoal do projeto, clientes, usuários e demais interessados para comentários ou aprovação, não havendo necessariamente a identificação de defeitos. Para Wong (2006), inspeção é, portanto, um tipo específico de revisão, assim como *walkthrough*. Podemos ainda inserir as inspeções entre as atividades de revisão de software contempladas pela Área de Processo de Verificação, preconizada no nível de maturidade 3 do CMMI (SEI, 2008) e no nível D do MPS-BR (SOFTEX, 2006).

A Inspeção de Software foi introduzida por Michael Fagan em 1976, inspirado pelos métodos estatísticos de qualidade utilizados na manufatura de hardware. Fagan basicamente formalizou em um processo a prática de se perguntar a um colega de trabalho se tudo está correndo bem em um projeto de software (Petersson, 2002). Diferentemente dos testes de software, as inspeções de software podem ser realizadas nas etapas iniciais do processo de desenvolvimento, permitindo a identificação de defeitos desde a descrição de requisitos, evitando a propagação

destes defeitos e o possível surgimento de falhas. As inspeções de software podem ser aplicadas a todos os tipos de artefatos de software existentes (Petersson, 2002), tendo como foco a verificação semântica dos artefatos.

3.2 O Processo de Inspeção de Software

Ao ser o primeiro a introduzir a inspeção de software, Fagan apresentou um processo que consiste em seis fases (Wong, 2006), apresentadas na Figura 3.1. A primeira fase consiste no Planejamento, onde o processo de inspeção de software é preparado e organizado. Abrange tipicamente a preparação dos procedimentos de inspeção, da reunião de inspeção e do pessoal envolvido, com a devida definição de seus papéis.

A segunda fase, a de visão geral (*overview*) visa treinar os revisores no artefato a ser inspecionado e sobre o escopo geral da inspeção. A condução de uma reunião de *overview* justifica-se quando o artefato a ser inspecionado é de difícil compreensão ou quando é desejável compreender melhor o relacionamento entre o artefato e o projeto de software (Wong, 2006).

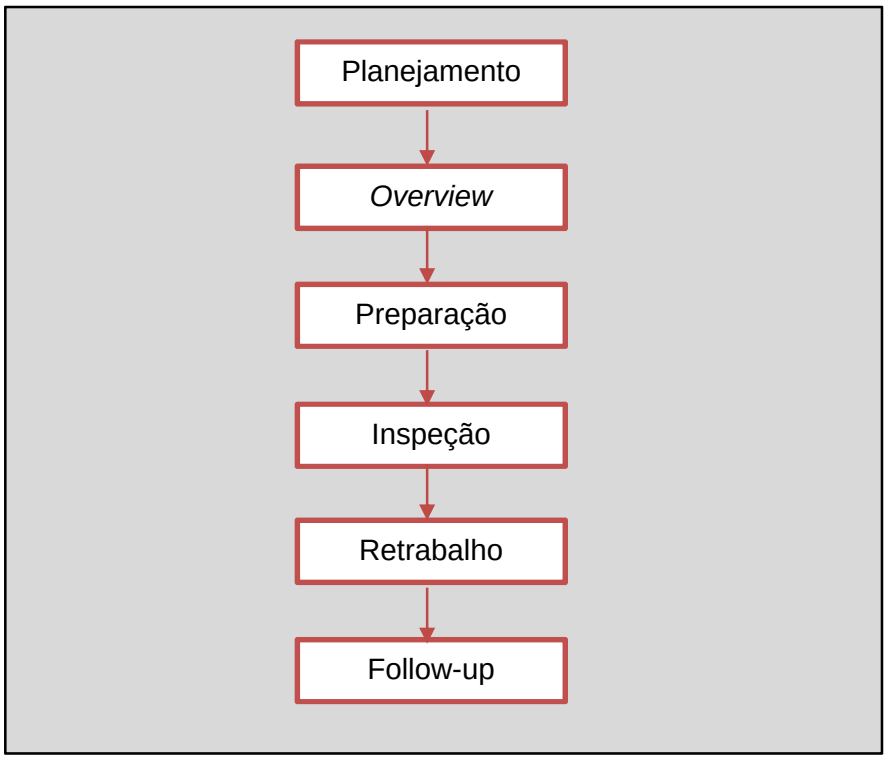

Figura 3.1. O processo de Inspeção de Software de Fagan (Wong, 2006)

Na fase posterior, a de preparação, os participantes da inspeção tentam compreender o artefato individualmente antes da próxima fase, a reunião de inspeção (Petersson, 2002), onde o autor apresenta o artefato para os demais participantes, que chamaremos de revisores, que buscam identificar os possíveis defeitos existentes no artefato.

Segundo Wong (2006), diversos estudos sugerem que as fases de preparação e de reunião de inspeção seriam custosas, tendo sido apresentadas propostas que adotam inspeções individuais ao invés de reuniões de inspeção ou reuniões de inspeção em pequenos grupos, visando manter uma melhor interação. Segundo o autor, estudos sugerem que não há necessidade da fase de preparação em inspeções individuais e que estas, inclusive, podem trazer melhores resultados.

Após a inspeção propriamente dita, vem a fase de Retrabalho (*rework*), onde o autor corrige os defeitos encontrados e a fase de Acompanhamento (*Follow-up*), onde a qualidade das correções aplicadas pelo autor é validada. Neste ponto, o moderador decide se será necessário realizar uma nova inspeção. Tal decisão pode ser apoiada por métricas de inspeção de software, que visam avaliar a qualidade da revisão, além de ajudar a entender os potenciais problemas e questões com o processo de desenvolvimento e apresentar valioso *feedback* para o processo de revisão. Algumas métricas de inspeção são citadas por Petersson (2002) e Wong (2006) , tais como:

- Número total de defeitos;
- Tempo dedicado à preparação individual e reunião de inspeção
- Numero de revisores envolvidos;
- Eficácia da inspeção, que consiste na razão entre os defeitos encontrados durante a inspeção e o número total de defeitos.

Após o processo de Fagan ter sido introduzido, diversas variantes foram sendo desenvolvidas com foco na melhoria de diversos aspectos do processo de inspeção. Tais melhorias podem ser divididas em três aspectos: a estrutura do processo, os passos do processo e o suporte ao processo (Petersson, 2002). As melhorias na estrutura do processo de inspeção focam as variações de projeto (*design*) de como as inspeções são executadas, dando origem a novos modelos de inspeção, tais como: a inspeção livre de reuniões; a inspeção de duas pessoas (*two-person*), somente envolvendo o autor do artefato e um revisor; a inspeção em fases, onde são realizadas diversas inspeções parciais.

Não somente a estrutura do processo de inspeção pode ser objeto de avaliação, mas também é possível buscar a melhoria da execução dos passos do processo. Incluem-se neste aspecto questões como a definição do número de

revisores adequados para uma inspeção e o uso de técnicas de leitura. Adicionar revisores pode aumentar a chance de encontrar novos defeitos, mas ao mesmo tempo aumenta o esforço. Deve-se buscar um balanceamento entre as restrições de custo, de tempo e de qualidade. Outra questão apontada por Petersson (2002) como importante para a melhoria dos passos do processo de inspeção consiste na adoção de técnicas de leitura para orientar a execução de uma inspeção.

O apoio ao processo de inspeção envolve métodos e ferramentas que apoiem a melhoria da eficácia e/ou eficiência do processo de inspeção. Dentre as ferramentas, frameworks computacionais como ISPIS (Kalinowski e Travassos, 2004) podem ser adotados para apoiar o processo de inspeção como um todo, mas podem ser adotadas também ferramentas de suporte parcial, tais como ferramentas de apoio à preparação da reunião de inspeção. Dentre os métodos de suporte ao processo de inspeção, Petersson (2002) destaca a importância das técnicas de estimativa de falhas, como o cálculo da densidade de defeitos e a análise de dados históricos.

Fagan (1986) também aponta os fatores que contribuem para a qualidade de uma inspeção através de um diagrama de causa/efeito ou diagrama de Ishikawa, apontando como causas primárias da qualidade da inspeção (efeito): a capacidade de um produto ser inspecionado (*inspectability*), o processo de inspeção, os gestores, e os programadores (desenvolvedores).

3.3 Categorias de Defeito

Existem diversas classificações de defeitos na literatura para inspeção de software conforme o tipo de artefato que está sendo inspecionado (requisitos, modelos, código). Para inspeção de requisitos, Porter e Votta (1994) e Porter et al. (1995), organizam os defeitos em dois níveis, sendo o primeiro nível composto por duas categorias: omissões e inadequações (*comission*). No segundo nível, uma omissão pode ser categorizada conforme o tipo de informação ausente na documentação: falta de funcionalidade, falta de desempenho, falta de ambiente ou falta de interface. Já os defeitos identificados como *comission*, podem ser classificados em: informação ambígua, informação inconsistente, informação extra/ incorreta ou sessão incorreta. Entretanto, esta proposta não comporta uma categoria para informações que possam estar corretas, mas que são desnecessárias no contexto observado. Como alternativa, a categorização adotada por Basili et al. (1996) possui apenas um nível composto de cinco classes não ortogonais, que abrangem as omissões (omissão de informação), os *comission* (informação ambígua, inconsistência entre informações e fato incorreto) e informação estranha.

Para a inspeção de modelos, Parnas e Weiss (1985) propõem a seguinte categorização: inconsistências, ineficiências (parte do modelo impõe barreiras para programação ou utilização do próprio modelo), ambiguidades ou inflexibilidades (a parte do modelo que é considerada defeituosa não propicia a acomodação de mudanças). Importante observar que esta categorização considera como defeito aspectos relacionados a critérios de qualidade como manutenibilidade e reusabilidade do modelo, que podem ser considerados subjetivos por não dependerem de um oráculo para comparação. Também se observa que a categorização citada não considera informações estranhas como defeito.

Outra categorização de defeitos para modelos é sugerida por (Travassos et al., 1999b) e (Shull et al., 1999), que adaptaram a categorização de Basili et al. (1996) para defeitos de requisitos em seus estudos. Esta categorização também tem sido aplicada na concepção de técnicas para inspeção de modelos, como no caso de ArqCheck (Barcellos e Travassos, 2006). A Tabela 3.1 apresenta as cinco categorias, com a redefinição dos conceitos e exemplos.

Tabela 3.1. Classes de Defeitos (adaptado de Travassos, 2001)

Categoria	Definição	Exemplo
Omissão	Informações necessárias foram omitidas do artefato	A representação de um conceito dos requisitos gerais do domínio ou do documento dos requisitos não está presente em nenhum dos diagramas do projeto
Fato incorreto	Algumas informações no artefato de software contradizem informações presentes na especificação de requisitos ou o conhecimento geral do domínio	Um diagrama de projeto contém uma representação errada de um conceito descrito nos requisitos gerais do domínio ou no documento de requisitos
Inconsistência	As informações em uma parte do artefato de software estão inconsistentes com outras no artefato de software	Uma representação de um conceito em um diagrama de projeto está em desacordo com a representação do mesmo conceito no mesmo ou em outro diagrama do projeto
Ambiguidade	As informações no artefato de software são ambíguas, isto é, é possível ao desenvolvedor interpretar as informações de diferentes maneiras, podendo não levar a uma implementação correta.	Uma representação de um conceito no projeto não está clara e poderia causar má interpretação ou entendimento errado do seu significado por parte do usuário do documento
Informação Estranha	As informações são fornecidas, mas não são necessárias ou mesmo usadas	O projeto inclui informações que embora talvez verdadeiras, não se aplicam a esse domínio e não deveriam ser incluídas no projeto.

Para a inspeção de código, Basili (1987) apresenta a categorização de defeitos em dois esquemas, que procuram distinguir as razões que levam os programadores a inserirem falhas no software. Primeiramente, o inspetor deve definir se um defeito trata-se de omissão de código ou de código incorreto (*comission*). Em seguida, o inspetor classifica o defeito conforme o tipo de código envolvido: inicialização imprópria (estrutura de dados), computação (causa a avaliação incorreta de valores de variáveis); controle (causa o fluxo incorreto de uma entrada de dados), interface (passagem de argumentos), uso incorreto de estruturas de dados, cosmético. Percebe-se que esta classificação, no entanto, não aborda códigos desnecessários, despropositados em relação ao contexto da aplicação.

3.4 Técnicas para Inspeção de Software

Um dos fatores decisivos no planejamento e nos resultados da inspeção de um projeto de software é a definição da técnica de inspeção que será utilizada. Uma inspeção pode ser realizada de modo *ad hoc,* dependendo exclusivamente da experiência do revisor, como também pode ser apoiada por técnicas estruturadas através de *checklists* ou de técnicas de leitura, que possuem um grau de formalidade maior e dependem menos da experiência do revisor para alcançar bons resultados do que os *checklists*. Existe também a alternativa do uso de heurísticas para apoiar a inspeção. A inspeção *ad hoc*, como o nome indica, baseia-se na experiência dos revisores, não havendo direcionamento sobre como proceder ou o que deve ser verificado especificamente durante a atividade de leitura. É também considerada como a “leitura sem técnica” (Petersson, 2002).

A inspeção baseada em *checklists* utiliza-se de uma estrutura em que questões do tipo “sim/não” devem ser respondidas pelos inspetores enquanto leem um documento de software (Laitenberger et al., 2001). Apesar da inspeção baseada em *checklists* ser citada frequentemente como uma técnica de leitura (Petersson 2002; Sabaliauskaite et al., 2003; Thelin et al., 2003; Wong 2006), a CBR (Checklist- Based Reading) é considerada menos formal que as demais técnicas de leitura, por não ser repetível e depender fortemente do revisor, não havendo uma formalização da análise realizada (Laitenberger et al., 2001). Da mesma forma, Porter et al. (1995) classificam tanto a inspeção *ad hoc* como *checklist* como técnicas de leitura não sistemáticas. Já Shull et al. (2000) classificam *checklist* como uma técnica de leitura parcialmente sistemática e não focada. Ao longo desta dissertação, quando citarmos técnicas de leitura, não estaremos incluindo as técnicas baseadas em *checklist*, tal como concluem He e Carver (2006).

As técnicas de leitura podem ser definidas como uma série de procedimentos que podem ser adotados por um revisor para obter um entendimento do artefato sob revisão, provendo um guia sistemático para a identificação de defeitos (Wong, 2006). He e Carver (2006) observam que, enquanto as inspeções *ad hoc* e via *checklists* são intuitivas e baseadas em procedimentos não sistemáticos, as técnicas de leitura possuem procedimentos explícitos e sistemáticos. Uma técnica de leitura é constituída de dois componentes principais: procedimentos para focar o revisor nos objetivos específicos da inspeção e questões que levem o revisor a refletir sobre a informação discrepante, de modo a encontrar defeitos (Shull et al. 2003), componentes estes que aumentam a eficiência dos revisores individuais (Travassos, 2001).

Shull et al. (1996) organizam as técnicas de leitura em famílias. No que tange ao escopo do problema, cada família de técnicas de leitura está associada a metas genéricas e específicas, atuando em um tipo de artefato e notação particular, conforme pode ser visualizado na Figura 3.2. Dentre as famílias de técnicas presentes na literatura, podemos destacar:

- DBR (*Defect- Based Reading*)- focam a inspeção de documentos de requisitos, onde cenários são descritos para apoiar a identificação de tipos específicos de defeito, tais como: inconsistência de tipos de dados, funcionalidades incorretas e ambiguidades (Porter et al., 1995);
- PBR (*Perspective- Based Reading*)- família de técnicas de leitura para inspeção de requisitos baseada em três perspectivas: usuário, projetista e testador. Para cada perspectiva foi definida uma abstração específica com uma técnica de leitura correspondente: para o usuário, foi estabelecida a modelagem de casos de uso; para o projetista, a construção de DFDs; para o testador; a construção de casos de teste (Shull, 1998). Em 2006, uma nova técnica baseada em perspectiva, a OO-PBR (*Object-Oriented Perspective- Based Reading*) foi apresentada em Mafra e Travassos (2006), com objetivo de realizar a inspeção dos requisitos através da modelagem do diagrama de classes.
- UBR (*Usage-Based Reading*)- tem como ideia principal direcionar o esforço de leitura para a detecção dos defeitos considerados mais críticos do objeto inspecionado sob o ponto de vista do usuário, que são considerados os defeitos com impacto mais negativo (Thelin et al., 2003);
- TBR (*Traceability- Based Reading*)- consiste na família de técnicas de leitura OORTs (Travassos et al., 1999), por esta orientar a verificação da rastreabilidade entre requisitos e artefatos de projetos orientados a objeto.

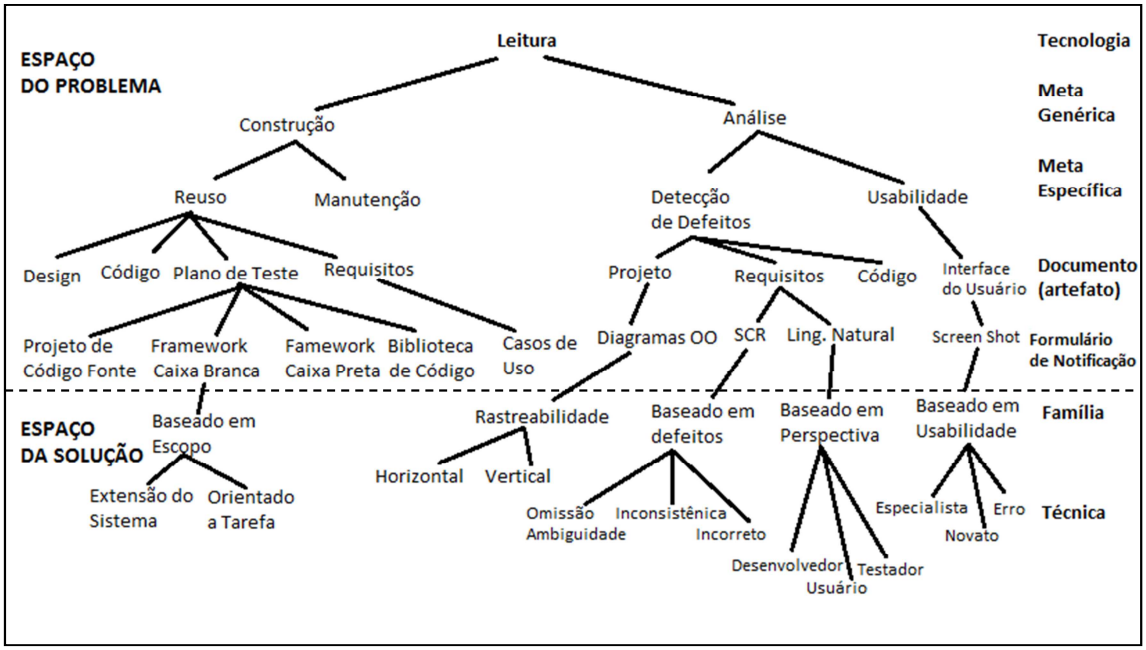

Figura 3.2. Famílias de técnicas de leitura (Shull et al., 1999)

O uso de heurísticas é uma opção alternativa às técnicas de leitura e aos *checklists* adotada por Conte et al. (2007) em sua técnica WDP (Web Design Perspectives-based Usability), que tem como objetivo avaliar a Usabilidade em projetos Web sob a perspectiva conceitual, navegacional, estrutural e de apresentação, valendo-se das heurísticas de avaliação de Nielsen (Nielsen, 1994), como é exemplificado na Tabela 3.1.

Conforme o tipo de técnica aplicada, uma inspeção de software pode variar no seu custo e na sua eficácia. Estudos sugerem que a adoção de técnicas de leitura apresentam melhores resultados do que a aplicação de *checklists* (Porter et al., 1995; Laitenberger et al., 2001; He e Carver, 2006), mas não há uma unanimidade (Sabaliauskaite et al., 2003). No estudo experimental conduzido por Laitenberger et al. (2000), por exemplo, foi identificado que a aplicação de PBR resultou numa eficácia 41% maior do que a aplicação de *checklist*, enquanto que o custo médio total de tempo por defeito detectado foi de 132 minutos por defeito para *checklist* e 56 minutos por defeito para PBR. É importante ressaltar, entretanto, que mesmo uma inspeção *ad hoc* em artefatos iniciais de um projeto de software já pode representar uma economia significativa para o projeto.

Tabela 3.2. Heurísticas da WDP relacionadas à Perspectiva de Navegação, baseado em (Conte et al., 2007)

#	Heurística
3	Avalie se a interface permite que o usuário navegue facilmente através dos diferentes passos de uma tarefa. Avalie se a inferface tem fazer/desfazer ou funções similares que permitem que o usuário saia em caso de escolhas erradas. Avalie se a interface permite que o usuário retorne para o fluxo principal de uma tarefa depois de um desvio ou depois de realizar uma tarefa secundária
5	Avalie se a interface previne erros de navegação, ou seja, se as opções disponíveis definem claramente quais resultados ou estados serão alcançados
7	Avalie se a interface provê formas diferentes de acessar as principais tarefas. Avalie se a interface provê teclas de aceleração ou atalhos quando o usuário realiza as tarefas principais Avalie se os acessos providos pela interface minimizam o esforço físico do usuário
9	Avalie se o sistema mostra como acessar soluções alternativas quanto são apresentadas mensagens de erro
10	Avalie se a interface provê um modo fácil de acessar ajuda e documentação de uma tarefa específica

3.4.1 Exemplos de Técnicas de Inspeção

Esta seção apresenta duas técnicas que apoiam a detecção de defeitos em modelos de software, cujas características estruturais influenciaram a elaboração da técnica de inspeção proposta nesta dissertação. São elas: OORTs, um conjunto de técnicas de leitura para inspeção de modelos UML e ArqCheck, técnica configurável baseada em *checklist* para inspeção de modelos arquiteturais.

OORTs

OORTs (Object-Oriented Reading Techniques) foi criada em 1998, constituindo-se numa família de técnicas de leitura (sete técnicas no total), desenvolvidas para detecção de defeitos em modelos UML de projeto de alto nível, entendendo-se por projeto (*design*) de alto nível o conjunto de artefatos referentes à representação dos conceitos do mundo real (Travassos et al., 2002). OORTs tem sido avaliada em diversos estudos experimentais, que tem comprovado sua eficácia e estudos também foram conduzidos para seu aprimoramento (Massollar, 2008), conforme pode ser observado em Conradi et al. (2003).

OORTs não visa à inspeção individual de artefatos, consistindo em três técnicas de leitura para inspeção horizontal e quatro técnicas de leitura para inspeção vertical, como pode ser observado na Figura 3.. Enquanto a inspeção horizontal foca a verificação da consistência dos artefatos, a inspeção vertical foca a verificação da rastreabilidade entre os artefatos. Estudos realizados sobre a aplicação OORTs (Travassos et al., 2002) sugerem que as técnicas de inspeção horizontais tendem a identificar defeitos de ambiguidade e inconsistência, enquanto que as inspeções verticais tendem a identificar defeitos de omissão e de funcionalidade incorreta.

Apesar de OORTs ser orientada para a inspeção de modelos UML, nem todos os modelos UML são contemplados por esta família de técnicas. Além da descrição dos requisitos, são objetos de inspeção em OORTs somente os seguintes modelos: diagrama de classes, descrição das classes, diagrama de estados e diagrama de sequência (interação). Tais modelos eram considerados, na época da criação das OORTs, os principais diagramas UML utilizados (Travassos et al., 2002). A Figura 3. apresenta um trecho de uma técnica de leitura OORTs para a inspeção entre Diagrama de Sequência e Casos de Uso.

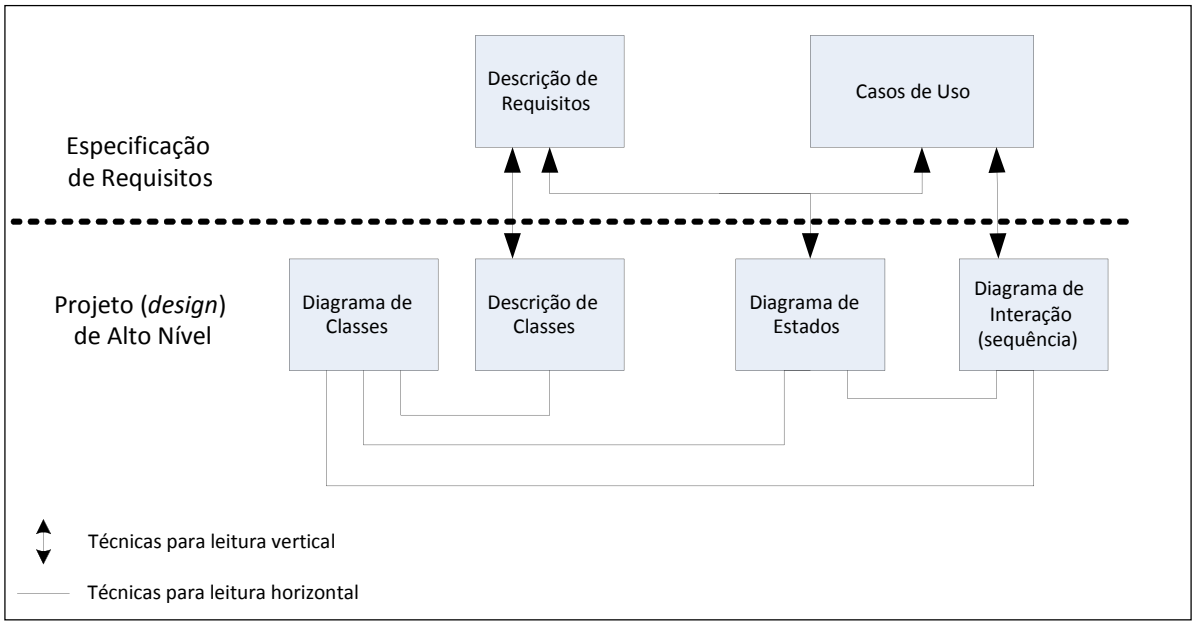

Figura 3.3. Técnicas de leitura que compõem OORTs, baseado em (Travassos et al., 1999).

Cada técnica de leitura de OORTs possui um objetivo definido, as entradas para o processo e as tarefas a serem realizados pelo revisor. Cada tarefa possui uma lista de entradas e saídas esperadas e os passos que devem ser executados, cada passo podendo conter orientações para preenchimento no Relatório de Discrepâncias, conforme estas forem sendo identificadas.

Leitura 5 – Descrições de Classes x Descrição de Requisitos

Objetivo: Verificar se os conceitos e serviços descritos pelos requisitos funcionais estão capturados apropriadamente pela descrição das classes.

Entradas para o Processo:

1. Um conjunto de requisitos funcionais que descrevem os conceitos e serviços que são necessários no sistema final.
2. Um conjunto de descrições de classes que lista as classes de um sistema juntamente com seus atributos e comportamentos.

I. **Leia a descrição de requisitos para entender a funcionalidade descrita.**

ENTRADAS: Conjunto de Requisitos Funcionais (FR).
SAÍDAS : Classes/Atributos candidatos (marcados em azul nos FRs); Serviços candidatos (marcados em verde nos FRs); Restrições e condições para os serviços (marcados em amarelo nos FRs).

A. Leia totalmente cada um dos requisitos funcionais para entender a funcionalidade que ele escreve.
B. Encontre os substantivos na descrição do requisito; eles são candidatos a se tornar classes, objetos, ou atributos no projeto do sistema. Sublinhe os substantivos com uma caneta azul.
C. Encontre os verbos, ou descrições de ações, que são candidatos a serem serviços ou comportamentos no sistema. Sublinhe os verbos ou descrições de ações com uma caneta verde.
D. Procure por descrições de restrições ou condições nos substantivos e verbos que você identificou nos dois passos anteriores. Preste atenção especialmente aos requisitos não funcionais, que tipicamente contem restrições e condições relacionadas às funcionalidades do sistema. Por exemplo, examine se os relacionamentos entre os conceitos foram identificados. Pergunte se existem restrições explícitas ou limitações na forma que as ações são executadas. Tente observar se quantidades definidas foram especificadas em alguma parte do requisito (veja Exemplo 4). Sublinhe estas condições e restrições com uma caneta amarela.

II. **Compare as descrições de classes aos requisitos para verificar se os requisitos foram capturados apropriadamente.**

ENTRADAS: Conjunto de Requisitos Funcionais (FR); Descrição das Classes (CD).
SAIDAS : Conceitos relacionados que foram marcados no FR e CD; Relatórios de Discrepância.

A. Para cada descrição de ação sublinhada em verde nos requisitos funcionais, tente encontrar um comportamento associado ou combinação de comportamentos na descrição da classe. Utilize dicas sintáticas (i.e. nome do comportamento que é similar ou sinônimo para uma descrição de ação) para ajudar você na busca, mas certifique que o significado *semântico* da função nos requisitos e projeto de alto nível é o mesmo. Quando encontrado, marque ambos o nome do comportamento(s) na descrição da classe e a descrição da atividade nos requisitos com um símbolo (*) verde.
1) Esteja certo que as classes recebem a informação correta para desempenhar os comportamentos requisitados. Certifique que resultados *viáveis* são produzidos. Se não, as classes não podem implementar a funcionalidade de forma apropriada. Indique isto no formulário de relato de discrepância e marque se isto é provocado por funcionalidade omitida ou incorreta ou então informação ambígua.

Figura 3.4. Trecho de técnica de leitura OORTs para inspeção de descrição de classes e descrição de requisitos, baseado em (Travassos et al., 2002)

ArqCheck

ArqCheck (Barcelos e Travassos, 2006) é uma técnica de inspeção baseada em *checklist* que tem por objetivo a melhoria da qualidade da arquitetura de um software através da revisão do documento arquitetural. Para ser aplicado ArqCheck, o documento arquitetural não precisa seguir nenhum padrão, mas deve conter algumas requisitos definidos pelas recomendações da norma IEEE 1417. São eles:

- O documento arquitetural deve identificar os elementos que compõem a solução;
- O documento deve descrever o papel dos elementos que compõem a arquitetura;
- O documento deve identificar como os requisitos relevantes para a arquitetura estão sendo atendidos por esta arquitetura;
- O documento deve representar o software através de diferentes perspectivas.

Os itens de avaliação (questões) de ArqCheck são classificados em três grupos: os itens que avaliam a consistência do documento, através das visões arquiteturais modular, dinâmica, de alocação e de contexto geral; os itens que avaliam o atendimento aos requisitos, verificando se todas as funcionalidades especificadas foram atendidas pela arquitetura e se todos os elementos arquiteturais foram definidos com base nos requisitos especificados; os itens que avaliam a abordagem utilizada no que tange ao atendimento dos requisitos de qualidade: desempenho, disponibilidade, modificabilidade, segurança, testabilidade e usabilidade. A Tabela 3.1 apresenta um trecho de ArqCheck, contendo itens para a verificação da consistência do documento.

Tabela 3.3. Trecho de ArqCheck baseado em (Barcelos e Travassos, 2006)

Itens de avaliação da consistência de representações entre os diagramas (específicos à abordagem e documentação arquitetural utilizada)				
Nº	**Visão Modular**	**Sim**	**Não**	**NA**
3	Os módulos internos de cada cluster foram descritos em algum diagrama da visão Modular?			
4	Todo relacionamento definido com um cluster foi devidamente mapeado para um d6e seus módulos internos?			
Nº	**Visão Dinâmica**	**Sim**	**Não**	**NA**
5	Toda porta/interface possui um nome, é utilizada com um único propósito e de forma única?			
6	Os fluxos de execução, descritos na visão dinâmica, alocam todos os módulos definidos na visão Modular?			
7	Todo módulo/cluster representado na visão dinâmica foi descrito na visão Modular?			

8	Todo fluxo entre dois elementos arquiteturais pode ser mapeado para algum relacionamento da visão Modular?			
9	Todo relacionamento descrito na visão modular pode ser mapeado para algum fluxo de comunicação, de dados ou de controle da visão Dinâmica?			
Nº	**Visão de alocação**	**Sim**	**Não**	**NA**
10	Todo módulo/cluster, representado na visão de alocação, foi descrito na visão Modular?			
11	Toda dependência, representada na visão de Alocação, pode ser mapeada para um ou mais relacionamentos da visão Modular?			
12	Dado os módulos/clusters representados na visão de Alocação, todos os relacionamentos definidos entre eles na visão Modular também foram representados na visão de Alocação			

3.5 Considerações Finais do Capítulo

Este capítulo apresentou uma visão geral do processo tradicional de inspeção de software, abordando fatores que podem influenciar na qualidade das inspeções. Dentre estes fatores, estão as técnicas de inspeção, que podem ser identificadas na literatura apoiando a identificação de defeitos em diversos tipos de artefato de software, sendo tipicamente estruturadas através de *checklists*, técnicas de leitura ou heurísticas. Estudos sugerem que as inspeções realizadas com o apoio de *checklists* apresentam resultados superiores às inspeções *ad hoc.* Em contrapartida, uma técnica de inspeção baseada em *checklist* pode servir como base para a posterior estruturação de uma técnica de leitura.

Para a elaboração de uma técnica de inspeção, dentre outros fatores, é necessário definir qual será o artefato de software a ser inspecionado, o escopo de aplicação da técnica e, quando possível, qual será o oráculo, ou seja, o artefato que será considerado como referência para a identificação de defeitos no artefato inspecionado. Com estes elementos definidos, é importante conhecer como cada artefato é estruturado e buscar por padrões e exemplos de sua utilização no escopo de aplicação da técnica a ser elaborada. Deste modo, será possível mapear possíveis casos de discrepância semântica.

Juntamente com o mapeamento dos casos de discrepância, também é importante considerar a categorização de defeitos que será seguida, que pode variar conforme as perspectivas que desejam ser observadas na inspeção e conforme o tipo de artefato que será inspecionado. A categorização de defeitos servirá como referência para a estruturação das situações de discrepância em itens de avaliação da técnica. Tais itens de avaliação podem ser apresentados na forma de questões, como num *checklist*, ou encadeados em um roteiro, como no caso de uma técnica de leitura.

CAPÍTULO 4 - IDENTIFICAÇÃO DE TÉCNICAS PARA INSPEÇÃO DE MODELOS DE WORKFLOW

Este capítulo trata da quasi-revisão sistemática que foi conduzida para identificar a literatura relevante sobre o tema de pesquisa, buscando seguir o primeiro passo para a definição de novas tecnologias baseada em evidências. Neste capítulo é introduzido o conceito de quasi-revisão sistemática e, em seguida os principais aspectos do protocolo da revisão são apresentados e o resultado obtido analisado.

4.1 Introdução

Este capítulo apresenta um resumo do protocolo e considerações sobre os resultados de uma *quasi*-revisão sistemática conduzida em 2009 com o objetivo de identificar técnicas para a inspeção de modelos de workflow, conforme define a abordagem para definição de novas tecnologias de software baseada em evidências (Mafra et al., 2006). Esta revisão sistemática, entretanto, foi precedida por uma revisão prévia da literatura, em que se buscou identificar técnicas para inspeção do diagrama de atividades. Nesta revisão de literatura, foi encontrada apenas uma única proposta, a de Tanriöver e Bilgen (2007). Esta proposta consiste numa técnica elaborada para inspeção do "diagrama de *workflow*", uma adaptação do Diagrama de Atividades da UML, pertencente a uma notação específica (KAMA).

Os autores relatam que a técnica (sem nome) foi elaborada *on the fly* com o objetivo de exemplificar a aplicação de um *framework* proposto para o desenvolvimento de técnicas de inspeção conforme a demanda de projetos (Tanriöver e Bilgen, 2007). A técnica consiste numa lista de recomendações (próxima a um *checklist*) para inspeção individual do Diagrama de Atividades, chamada de inspeção intra-diagrama e outra lista de recomendações para apoiar a inspeção entre diagramas de atividades e descrições de casos de uso, chama da de inspeção inter-diagramas.

Observou-se que, nesta proposta, a verificação semântica dos modelos é pouco explorada, incluindo poucas orientações para a identificação de defeitos, sendo a maioria das orientações focada na verificação interna do diagrama de atividades. Observou-se também que, diversos recursos semânticos do diagrama de atividades da UML 2.0 não foram contemplados, tais como: bifurcações/ sincronizações, objetos,

envio/ recebimento de sinal e pré e pós-condições locais. Importante ressaltar que, tais recursos não contemplados são relevantes para especificar os requisitos de aplicações contemporâneas.

Ao longo do desenvolvimento da pesquisa, fora dos resultados das duas revisões, foi identificado também um estudo comparativo entre *checklists* e técnicas de leitura (Sabaliauskaite et al. 2003), cujo *checklist* avaliado apresenta algumas questões (quatro) para identificação de defeitos em diagrama de atividades. Entretanto, duas destas questões são muito abrangentes e as outras duas são direcionadas para os diagramas de atividades anteriores à UML 2.0, por tratarem da transição de estados. Uma técnica de leitura avaliada pelos autores também repete estas mesmas questões.

4.2 *quasi*-Revisão Sistemática

Kitchenham et al. (2004) observam que as evidências atualmente relacionadas às tecnologias de engenharia de software são fragmentadas e limitadas, não sendo integradas adequadamente e desprovidas de padrões ou guias acordados entre os pesquisadores. Na Engenharia de Software, ainda não há uma cultura amplamente disseminada de pesquisa que reivindique a aplicação de revisões sistemáticas e de replicações de estudos, diferentemente do que ocorre em áreas como a medicina.

A revisão sistemática de literatura é uma forma de estudo secundário que consiste num meio de identificação, avaliação e interpretação de toda pesquisa relevante para uma questão de pesquisa em particular, área de um tópico, ou fenômeno de interesse (Kitchenham, 2004). Ao contrário de uma revisão tradicional de literatura, a revisão sistemática possui seu protocolo de pesquisa pré-definido e documentado, tornando-a capaz de ser repetida e reduzindo a influência do viés dos pesquisadores. Para Kitchenham (2004) a condução de uma revisão sistemática consiste nos seguintes estágios:

1. Planejamento da Revisão
    - 1.1 Identificação da necessidade de revisão
    - 1.2 Desenvolvimento de protocolo de revisão
2. Condução da Revisão
    - 2.1 Identificação da Pesquisa
    - 2.2 Seleção dos estudos primários
    - 2.3 Avaliação da qualidade dos estudos
    - 2.4 Extração de dados e monitoração
    - 2.5 Síntese de dados

3. Relato da revisão

De modo complementar, Pai et al. (2004) detalham uma sequência de passos para a condução de uma revisão sistemática, que se inicia pela definição da questão de pesquisa, formulada a partir da definição dos elementos do PICO: População (*Population*), também conhecido como *Patient*, no caso da medicina); Intervenção (*Intervention*), o que está sendo avaliado; Comparação (*Comparison*) com a intervenção, quando for o caso; Resultados (Outcome) esperados. A técnica PICO viabiliza a elaboração de questões de pesquisa mais focadas, auxiliando na condução de pesquisas em bases mais específicas e na criação de critérios de seleção não-ambíguos.

A Revisão Sistemática pode ser utilizada também para identificar não somente estudos primários, mas também artigos teóricos em geral e/ou aplicações na indústria, a fim de caracterizar determinada tecnologia. Desta forma, a revisão não oferecerá elementos para a condução de uma posterior meta-análise bem como pode não possuir elementos para Comparação (*Comparison*). Uma revisão com tais características é considerada uma *quasi*-Revisão sistemática (Travassos, 2008), que é a forma de revisão aplicada nesta pesquisa, cujo protocolo pode ser visualizado no Apêndice A.

4.3 Planejamento da Revisão

Conforme explanado, para ser proposta uma técnica de inspeção do Diagrama de Atividades faz-se necessário, primeiramente, identificar na literatura especializada se já existem tecnologias que atendam a esta necessidade. Entretanto, as características inerentes ao Diagrama de Atividades e sua aplicabilidade na fase de especificação de requisitos e, principalmente, na especificação de *workflows*, sugerem a importância de serem pesquisadas não somente técnicas para inspeção do próprio Diagrama de Atividades da UML, como também técnicas para inspeção de modelos de *workflow* em geral ou técnicas que porventura existam para inspeção de outras notações com estrutura e aplicabilidade semelhantes ao Diagrama de Atividades. A seguir, são apresentadas outras três importantes notações identificadas na literatura como linguagens para representação de *workflow* (Wynn et al., 2009):

- BPMN (Business Process Modeling Notation): têm por objetivo principal prover uma notação de leitura compreensível para todos os usuários, dos analistas de negócio aos desenvolvedores, criando uma ponte padronizada entre o desenho de processos de negócio e a implementação do processo (OMG3, 2009).

Visualmente, esta notação é bem semelhante ao Diagrama de Atividades, que, inclusive, foi uma das notações examinadas para sua elaboração;

- EPCs: Event Process Chains, linguagem de modelagem que, juntamente com sua forma estendida eEPCs, tornaram-se bastante difundidas por serem adotadas em ferramentas famosas de Planejamento de Recursos Empresariais (ERP) e de Gestão de Workflow (WFM), como SAP R/3 e ferramentas de Reengenharia de Processo de negócio, como ARIS (van der Aalst, 1999).
- YAWL (Yet Another Workflow Language), linguagem apresentada por van der Aalst e Hofstede (2005), com o objetivo de ser uma linguagem de *workflow* mais completa que abranja todas as perspectivas (fluxo de controle, dados, de recursos e operacional) definidas pelos *workflow patterns* (van der Aalst et al., 2003). Também é baseada nas Redes de Petri, como o Diagrama de Atividades.

Outro motivador para a realização de uma revisão envolvendo as demais linguagens de *workflow* é o fato de que, em uma busca recente utilizando a máquina de busca Scopus (Elsevier, 2009) para localizar os termos *"inspection"* e *"activity diagram"* em resumos, títulos e palavras-chave de artigos, somente foi localizado um artigo, de Tanriöver e Bilgen (2007), que apresenta uma técnica para inspeção de um diagrama adaptado do Diagrama de Atividades.

Baseado principalmente no guia de Kitchenham (2004), foi montado um protocolo da *quasi*-revisão sistemática, sendo apresentadas nesta sessão alguns de seus itens. Inicialmente, foi definida a única questão que norteou a revisão: *"Quais são as técnicas para inspeção aplicáveis a modelos de workflow em projetos de software?".* Em seguida, foi aplicada a técnica PICO (Pai et al., 2004) para formulação da questão de pesquisa, obtendo-se o seguinte resultado:

- ***Population:*** Artigos que descrevam técnicas de inspeção aplicáveis a modelos de workflow em projetos de software.
- ***Intervention:*** Técnicas de Inspeção aplicáveis a Diagrama de Atividades, BPMN, EPC, YAWL, Fluxogramas ou outras linguagens de workflow.
- ***Comparison:*** Não há.
- ***Outcome:*** Identificar as técnicas, suas descrições, heurísticas utilizadas, sua aplicabilidade.

A partir da questão de pesquisa e da análise PICO, foi estruturada a *string* de pesquisa, cuja Figura 4.1 apresenta a versão definitiva para a máquina de busca Scopus. Entretanto, pode-se observar na figura que o termo "técnica de inspeção" não

foi aplicado para evitar limitação dos resultados que utilizassem outros termos simples, até mesmo “validação”. Também para não restringir o escopo da pesquisa, apesar da População citar “projetos de software”, também foram considerados artigos relacionados a processos de negócio. No caso do modelo adotado, além do termo genérico *workflow*, foram adotadas *strings* que correspondessem ao diagrama de atividades e a outros modelos semelhantes, descritos na Intervenção.

TITLE-ABS-KEY (((inspection OR review* OR reading* OR verification* OR validation* OR checklist*) AND (software* OR "business process" OR UML OR "Unified Modeling Language")) AND ("activity diagram" OR "activity diagrams" OR "activity model" OR "activity models" OR BPMN* OR "Business Process Modeling Notation" OR EPC* OR eEPC* OR "Event-driven Process Chain" OR Flowchart* OR Workflow* OR YAWL* OR "Yet Another Workflow Language"))*

Figura 4.1. String de pesquisa adotada na ferramenta SCOPUS

Foram considerados tanto artigos teóricos, como estudos experimentais, aplicações na indústria ou mesmo uma combinação entre estes. Além dos critérios para seleção de fontes, que incluíram principalmente máquinas de busca disponíveis na Web, foram definidos critérios para inclusão e exclusão dos artigos retornados pelas buscas. São eles:

- O título e resumo dos artigos devem estar disponíveis na Web;
- O resumo (*abstract*) deve indicar a caracterização ou aplicação de uma técnica de inspeção de modelos de *workflow*;
- Os artigos com resumo aprovado (pré-selecionados) deverão estar preferencialmente disponíveis na web, acessíveis através do *link* da máquina de busca. Caso não estejam disponíveis, serão utilizados até dois e-mails disponíveis dos primeiros autores de cada artigo para solicitá-los. Caso não haja algum retorno no prazo de 30 dias corridos a contar da data de solicitação, as referências faltantes serão consideradas indisponíveis;
- O título e resumo dos artigos devem estar escritos em inglês, espanhol ou português;
- Os artigos devem estar escritos em inglês ou espanhol (máquinas de busca), ou em português (anais do SBES e SBQS);
- Os artigos devem permitir a identificação de pelo menos uma técnica de inspeção de modelos de *workflow*;
- Não serão considerados artigos que não caracterizem ou que não apresentem

uma referência bibliográfica que caracterize a técnica de inspeção mencionada;

- Não serão considerados artigos que apresentem técnicas de inspeção cuja descrição não esteja em linguagem natural;
- Não serão considerados artigos que apresentem técnicas de inspeção que não inspecionem o próprio modelo de workflow e que somente o comparem com artefatos que não sejam nem a descrição dos requisitos nem modelos UML;
- Não serão considerados artigos cuja técnica de inspeção dependa de transformações dos modelos de *workflow* em outras representações, como redes de Petri, algoritmos ou grafos;
- Não serão considerados artigos cuja técnica de inspeção dependa de apoio computacional;
- Não serão considerados artigos que somente descrevam regras e/ou critérios de qualidade, sem especificar ou referenciar uma técnica de inspeção;
- Não serão considerados artigos cuja técnica de inspeção não apresente oportunidades para identificação de defeitos, como técnicas baseadas em regras de redução e refatorações.

Tais critérios procuraram evitar a seleção de técnicas que não dessem foco ao objetivo principal da inspeção de software: identificação de defeitos através do exame visual de um artefato (IEEE, 1998). Espera-se que o apoio deste exame seja feito através de uma tecnologia de fácil compreensão (leitura) e que norteie as atividades do inspetor.

Como procedimento para seleção dos artigos, foi estabelecido que um pesquisador aplicasse os critérios de pesquisa sobre as fontes e que os artigos identificados fossem classificados em incluídos, excluídos ou não definidos por este mesmo pesquisador. Um segundo pesquisador realizaria, então, uma avaliação da classificação feita pelo primeiro pesquisador e, ao final desta avaliação, os artigos que continuassem classificados como não-definidos seriam incluídos. Finalmente, o primeiro pesquisador extrairia as informações previstas dos artigos incluídos e o segundo pesquisador avaliaria a qualidade destes artigos. O protocolo previa a extração de informações básicas sobre o artigo e de indicadores da aplicação das técnicas apresentadas, tais como:

- Nome do modelo inspecionado (Diagrama de Atividades/ EPC/ BPMN/ Fluxograma/ YAWL/ outro <especificar>/ independente de modelo)
- A técnica visa inspecionar o modelo de workflow individualmente? (Sim/ Não)
- A técnica visa inspecionar o modelo de workflow em relação a outro(s) artefato(s) num mesmo nível de abstração? (Sim/ Não) Quais são estes

artefatos?

- A técnica visa inspecionar o modelo de workflow em relação a outro(s) artefato(s) em níveis de abstração distintos? (Sim/ Não) Quais são estes artefatos?
- Tipo de técnica de inspeção (*checklist*/ técnica de leitura/ heurísticas)

Adicionalmente, se fosse o caso do artigo mencionar um estudo sobre determinada técnica, o protocolo orientava a extração de informações que caracterizasse este estudo e os resultados obtidos. Cada artigo selecionado deveria também ser avaliado conforme as questões de qualidade e pontuações possíveis descritas na Tabela 4.2, sendo atribuído um conceito referente à qualidade total de cada artigo:

Tabela 4.2. Questões e conceitos para avaliação da qualidade dos artigos selecionados

Questão	Pontuações possíveis
Onde se localiza a descrição da técnica de inspeção?	0 (fora do artigo); 1 (no próprio artigo)
O artigo utiliza a terminologia adequada?	0 (Não) ou 0,5 (Sim)
O artigo explicita as restrições e as condições de aplicação da técnica de inspeção?	0 (Nenhum), 0,5; 1; (Parcial) 1,5 (Todos)
O artigo apresenta algum estudo sobre a técnica de inspeção?	0 (Não); 0,5 (Prova de conceito ou aplicação na indústria); 1 (Sim, estudo)
Os resultados do estudo sobre a técnica de inspeção são descritos?	0 (Não) ou 0,5 (Sim)

4.4 Execução e Resultados Obtidos

Uma rodada de teste foi executada em maio de 2009 na máquina de busca principal (Scopus) com uma versão inicial da *string* de pesquisa, sendo identificados 559 artigos sendo que 97 foram pré-selecionados através da análise do resumo (*abstract*), mas apenas um destes artigos atendeu a todos os critérios de inclusão e exclusão. Este artigo é o mesmo previamente definido como artigo de controle (Tanriöver e Bilgen, 2007)

Após a análise dos resultados, para a execução da revisão completa, decidiu-se pelo aprimoramento da *string* de pesquisa, resultando na string apresentada na figura 4.1. No caso dos anais do SBES e do SBQS, os termos da *string* foram pesquisados manualmente, sendo que não foi identificado nenhum artigo que atendesse à *string* de pesquisa. A execução nas máquinas de busca deu-se entre 04 de setembro de 2009 (Scopus e Compendex) e 05 de setembro de 2009 (IEEEXplore), sendo retornadas 1486 referências no total. Destas referências, apenas 929 eram distintas, havendo repetições devido ao uso de máquinas de busca aglutinadoras de bibliotecas digitais (Scopus e Compendex).

Destes 929 artigos, 170 foram pré-selecionados após a leitura do resumo, sendo que 151 destes artigos foram disponibilizados para leitura. A pré-seleção destes artigos permitiu encontrar algumas propostas para a verificação de Diagramas de Atividades e outras notações aplicáveis a *workflow*, diferentes da proposta de Tanriöver e Bilgen (2007), tais como (Eshuis e Wieringa, 2004; Choi e Watanabe, 2005; Guelfi e Mammar, 2005; Xu et al., 2006; Qian et al., 2007). Entretanto, em sua maioria, tais propostas focam na verificação sintática e automatizada dos modelos, sem qualquer menção a procedimentos que possam orientar um inspetor na realização da revisão e na exploração dos aspectos semânticos do modelo utilizando linguagem natural.

4.5 Ameaças à Validade da Revisão

Além de 19 artigos pré-selecionados através da leitura do abstract não terem sido avaliados, destacamos também como ameaça à validade da revisão o risco da pesquisa de strings em *abstract*, dada especificidade do tema. Conforme já exemplificado em OORTs (Travassos et al., 1999), existe a possibilidade de uma abordagem de inspeção agregar um conjunto de técnicas. Desta forma, expressões de pesquisa (*search strings*) que referenciem um modelo específico, como no caso "*activity diagrams*", podem, por exemplo, não constar no resumo de um artigo que apresente uma abordagem de inspeção mais ampla que apenas este diagrama. Uma alternativa seria pesquisar tais *strings* em todo o artigo ao invés de somente no título, *abstract* e palavras-chave, mas observamos que a aplicação de *search strings* para todo conteúdo do artigo é limitada pelas ferramentas de busca devido à existência de artigos que precisam ser comprados para serem lidos, sendo que seu conteúdo completo não fica disponível para pesquisa.

4.6 Considerações Finais do Capítulo

Os cenários apresentados pelas revisões de literatura e pela *quasi*-Revisão Sistemática indicam a carência de tecnologias que apoiem a inspeção de diagramas de atividades e modelos de workflow em geral, realçando a necessidade e a oportunidade de construção de uma tecnologia para apoiar a inspeção de especificações de requisitos descritas com diagramas de atividades para atender à demanda de projetos de software que utilizam esta forma de especificação.

Através dos artigos analisados, percebe-se também que o conceito de revisão de um modelo de software muitas vezes é considerado como limitado à verificação das regras sintáticas de modelagem, não abrangendo a verificação semântica inerente às atividades de inspeção. Entretanto a verificação sintática de um modelo pode ser tratada facilmente através de uma verificação automatizada.

A única proposta de detecção de defeitos semânticos através do exame visual de um artefato (Tanriöver e Bilgen, 2007) identificada é citada como um exemplo para um contexto específico, não há indícios de sua aplicação em provas de conceito ou mesmo em estudos experimentais. Além disto, verificou-se que poucos conceitos do diagrama de atividades são explorados. Neste cenário, optamos por desenvolver uma técnica para inspeção do diagrama de atividades, que será apresentada no capítulo seguinte.

CAPÍTULO 5 - TÉCNICA PARA INSPEÇÃO DE DIAGRAMAS DE ATIVIDADES EM PROJETOS DE SOFTWARE

Este capítulo apresenta a técnica proposta, desde sua primeira versão, que foi submetida a uma prova de conceito. Além dos artefatos que compõem a técnica, também é apresentada a estratégia de pesquisa que foi aplicada para o desenvolvimento e a organização destes artefatos.

5.1 Introdução

As poucas contribuições identificadas nas revisões realizadas não exploram as possibilidades de aplicação de diversos recursos semânticos que podem ser aplicados a um diagrama de atividades. Além disto, a utilização de uma especificação de requisitos como referência para a identificação de defeitos no modelo também não é explorada.

Considerando-se a aplicabilidade do diagrama de atividades já mencionada no capítulo 4, a detecção de defeitos deste modelo pode envolver diversos cenários, o que sugere a caracterização de outras técnicas de inspeção, além da já mencionada e necessária técnica de inspeção entre o diagrama de atividades e a descrição textual dos requisitos. Na proposta de Massollar (2008), por exemplo, está previsto o desenvolvimento de diagramas baseados em outros modelos da UML, tais como o diagrama de classes e o diagrama de sequencia. Deste modo, a rastreabilidade entre os diagramas de atividades e estes diagramas poderia ser verificada.

A Figura 5.1, baseada em OORTs (Travassos et al.,1999), apresenta possíveis sugestões de técnicas para inspeção horizontal e vertical relacionadas ao diagrama de atividades, sendo que, neste caso, os casos de uso são substituídos pelo diagrama de atividades. Através de análises preliminares baseadas no conjunto de técnicas de OORTs (Travassos et al., 2002), identificamos a oportunidade de desenvolvimento de técnicas de inspeção verticais para apoiar a identificação de defeitos entre o diagrama de atividades e diagrama de sequência (2) e entre o diagrama de atividades, descrição de requisitos e diagrama de estados (3). Massollar (2008) acrescenta que o diagrama de classes conceitual pode ser elaborado a partir do diagrama de classes que

representa o modelo de domínio e de informações extraídas do Diagrama de Atividades, o que representaria outra técnica de inspeção (4).

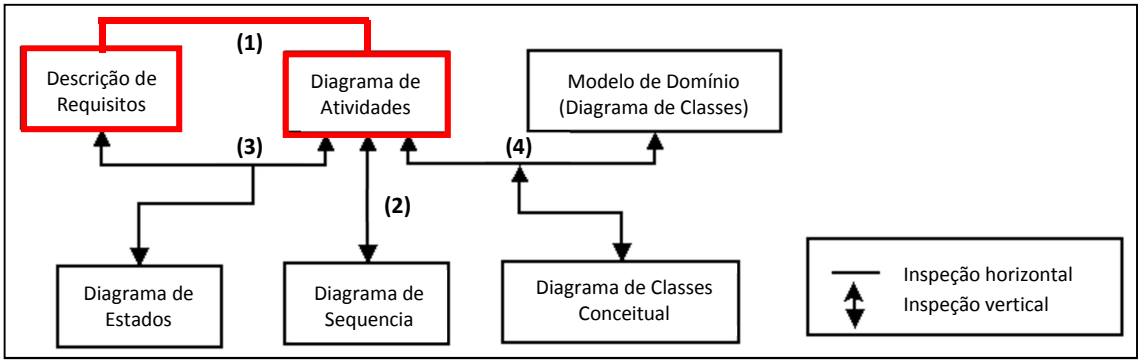

Figura 5.1. Possíveis técnicas para inspeção do diagrama de atividades.

Entretanto, tendo como base a motivação apresentada neste documento e nos resultados das revisões, definimos como escopo de pesquisa a elaboração de uma técnica de inspeção de especificações de requisitos descritas com diagramas de atividades (1), incluindo evidências sobre a sua viabilidade.

5.2 Casos Discrepantes

Para direcionar a elaboração da proposta inicial da técnica foi realizado um levantamento dos possíveis casos que podem sugerir defeitos entre a especificação de requisitos e um diagrama de atividades, que foram chamados de casos discrepantes. Neste sentido, caracterizamos como casos discrepantes situações genéricas que podem ser encontradas em um artefato de modo a qualificar uma discrepância, ou seja, um possível defeito.

Cada caso discrepante está relacionado a uma ou mais categorias de defeito, conforme a categorização de Basili (1996), adaptada por Travassos et al. (1999): omissão, fato incorreto, inconsistência, ambiguidade, informação estranha. Esta categorização é adotada por técnicas para inspeção de modelos, como OORTs (Travassos et al., 1999) e ArqCheck (Barcelos e Travassos, 2006) e também por técnicas para inspeção de requisitos, como PBR (Shull et al, 2000) e OO-PBR (Mafra e Travassos, 2006). Adicionalmente, podem ser apresentados exemplos de discrepância para um caso discrepante.

Para definição dos casos discrepantes, foram exploradas as possibilidades de aplicação de cada recurso sintático do diagrama de atividades, através do estudo da superestrutura do Diagrama de Atividades especificada na UML 2.2 (OMG, 2009), apresentada no Capítulo 2 desta dissertação e do estudo da aplicabilidade destes

recursos sintáticos para atender a representação dos *workflow patterns,* apresentados por van der Aalst (2003) e Russel et al. (2005). Também foram considerados os contextos específicos de aplicação do diagrama de atividades para especificação de requisitos, conforme apresentado por Pereira et al. (2009) e Massollar (2008).

5.2.1 *Workflow Patterns*

Segundo van der Aalst et al., (2003), as especificações de fluxo de trabalho (*workflow)* podem ser compreendidas através das seguintes perspectivas:

- perspectiva de fluxo de controle, que descreve as atividades (que correspondem às ações no contexto da UML) e sua ordem de execução;
- perspectiva de dados, que consistem nas camadas de dados de negócio e de processamento da perspectiva de fluxo controle;
- perspectiva de recurso, que provê uma âncora de estrutura organizacional para o *workflow* na forma de papéis de pessoas e de dispositivos responsáveis pela execução das atividades;
- perspectiva operacional, que descreve as ações elementares executadas pelas atividades.

Com o objetivo de definir requisitos para as possíveis notações (linguagens) para representação de *workflow*, tais como o diagrama de atividades, van der Aalst et al. (2003) propuseram padrões direcionados para a perspectiva de fluxo de controle, dando origem aos *Workflow Patterns*. Tais padrões foram inspirados pela definição de padrões de Riehle e Züllighoven (1996) e pelos *design patterns* de Gamma et al. (1995). van der Aalst et al. (2003) apresentaram ao todo 20 padrões, divididos em seis grupos:

1) Padrões de fluxo de controle básico- são os padrões considerados elementares em um *workflow*: Sequência, *Split* paralelo, Sincronização, Escolha exclusiva e *Merge* Simples;
2) Padrões de desvio e de sincronização avançada- ao contrário do primeiro grupo, aqui são definidos os padrões que os autores consideraram nem sempre ser suportados de modo prático pelas linguagens de workflow. São eles: Escolha múltipla, *Merge* sincronizado, *Merge* Múltiplo e *Discriminator*.
3) Padrões estruturais- envolvem dois padrões: os ciclos arbitrários, referentes à implementação de *loops*; a terminação implícita, que diz respeito à sinalização do fim de um *workflow* ou do fim de um fluxo.

4) Padrões de múltiplas instâncias- reúnem quadro padrões para execução de iterações num *workflow*. São eles: instâncias múltiplas (IM) sem sincronização, IM com conhecimento prévio em tempo de projeto, IM com conhecimento prévio em tempo de execução, MI sem conhecimento prévio em tempo de execução;
5) Padrões baseados em estados- trata de três situações especiais fluxo de controle que dependem da ativação de estados. São elas: a escolha diferida, o roteamento paralelo interligado (*interleaved parallel routing*) e o marco (*milestone*);
6) Padrões de cancelamento- considera padrões para a desativação de um evento ativo ou de todo um *workflow*. São eles: atividade de cancelamento e caso de cancelamento.

Cada um dos 20 padrões apresentados no trabalho são descritos e exemplificados. Também são informados alguns termos sinônimos para cada padrão. A Tabela 5.1 apresenta a descrição de um dos padrões de Sincronização Avançada, a Escolha Múltipla. A partir da proposta dos primeiros *Workflow Patterns*, surgiram outros estudos para a elaboração de padrões focando outras perspectivas. No caso da perspectiva de dados, Russel et al. (2005) apresentam 40 padrões, subdivididos em: visibilidade de dados, interação de dados (interno), interação de dados (externo), transferência de dados e roteamento baseado em dados.

Tabela 5.1. Descrição do padrão de Escolha Múltipla. Baseado em (van der Aalst et al., 2003)

Padrão 6- Escolha Múltipla
Descrição- É um ponto no processo de Workflow onde, baseado em uma decisão ou data de controle de workflow, um determinado número de caminhos (*branches*) são escolhidos
Sinônimos- Roteamento Condicional, Seleção, OR-*split*
Exemplo- Após executar a atividade avaliar_dano a atividade contatar_bombeiros ou a atividade contatar_companhia_seguro é executada. Pelo menos uma destas atividades é executada. Entretanto, é possível que ambas precisem ser executadas.

Desde a criação dos *Workflow Patterns*, análises têm sido realizadas sobre abrangência das consideradas linguagens de *workflow*, em relação à capacidade de representação destes padrões. Estas análises permitem explorar as possibilidades de aplicação de linguagens de *workflow*, tais como BPMN (Wohed et al., 2006), o diagrama de atividades (Russel et al., 2006; Wohed et al., 2005) e outros (van der

Aalst e Hofstede, 2005). Em uma avaliação realizada por Russel et al. (2006), o diagrama de atividades mostrou-se significativamente abrangente quanto à representação dos padrões de fluxos de controle previstos nos *Workflow Patterns*, sendo verificado que quase todos os padrões da perspectiva de fluxo de controle (16 de 20 no total) são diretamente suportados pelo Diagrama de Atividades da UML 2.0, como pode ser visualizado na Tabela 5.2. O resultado é semelhante à alternativa de modelagem BPMN (Wohed et al., 2006). Já os *Workflow Patterns* que fazem parte da perspectiva de dados foram considerados parcialmente atendidos pelo Diagrama de Atividades, o que também ocorreu com BPMN (Wohed et al., 2006).

Tabela 5.2. Avaliação da cobertura da UML 1.4 e 2.0 em relação aos workflow patterns, baseado em (Wohed et al., 2006)

No.	Padrão	2.0	1.4		No.	Padrão	2.0	1.4
1	Sequência	+	+		11	Terminação Implícita	+	-
2	*Split* paralelo	+	+		12	MI sem sincronização	+	-
3	Sincronização	+	+		13	MI com conhecimento prévio em tempo de projeto	+	+
4	Escolha exclusiva	+	+		14	MI com conhecimento prévio em tempo de execução	+	+
5	Merge simples	+	+		15	MI sem conhecimento prévio em tempo de execução	-	-
6	Escolha múltipla	+	-		16	Escolha diferida	+	+
7	Merge sincronizado	-	-		17	Roteamento paralelo interligado	-	-
8	Merge múltiplo	+	-		18	Marco	-	-
9	*Discriminator*	+	-		19	Atividade de cancelamento	+	+
10	Ciclos arbitrários	+	-		20	Caso de Cancelamento	+	+

Importante observar também que o diagrama de atividades da UML 1.4 é capaz de suportar diretamente apenas metade dos *workflow patterns*, o que vai ao encontro do entendimento da ampliação da aplicabilidade do diagrama de atividades a partir da UML 2.0.

5.2.2 Identificação dos Casos Discrepantes

Para a elaboração da maior parte dos casos discrepantes, cada recurso sintático do diagrama de atividades foi analisado individualmente e em conjunto sob o ponto de vista das diversas possibilidades de aplicação descritas nos padrões das perspectivas de fluxo de controle e de dados que compõem os *workflow patterns* (van der Aalst et al., 2003).

Para evitar redundâncias e facilitar a elaboração da técnica de inspeção, os casos discrepantes que fossem identificados foram categorizados em nove grupos distintos, sendo que sete destes grupos classificam os casos discrepantes conforme o recurso sintático do diagrama de atividades mais diretamente envolvido com a discrepância (grupos A-G). São eles:

A) Nós de Ação e Fluxo de ações;
B) Nós de Controle- Início, Fim de Atividade, Fim de Fluxo;
C) Nós de Controle- Nós de Bifurcação e Sincronização;
D) Nós de Controle- Nós de Decisão e de Merge;
E) Regiões de Interrupção, Regiões de Expansão, Nós de Loop, Nós Condicionais, Nós Protegidos e Exceções;
F) Nós de Objeto e Fluxo de Objetos;
G) Raias.

As Tabela 5.3 e 5.4 listam alguns casos discrepantes definidos para os grupos A e D, respectivamente. A partir da análise específica das ações e do fluxo de ações.

Tabela 5.3. Exemplos de casos discrepantes envolvendo ações e fluxo de ações.

A- Nós de Ação e Fluxo de ações		
Cód.	**Descrição**	**Categoria de defeito**
A02	Alguma pré ou pós condição deixou de ser foi informada	Omissão
A03	Alguma ação foi omitida numa sequência de ações	
A06	O tempo especificado no recebimento de sinal não condiz com o especificado no oráculo ou o conhecimento do domínio	Fato Incorreto
A07	Uma sequência de ações não condiz com a especificação ou com o conhecimento do domínio.	
A10	O tempo especificado para um recebimento de sinal diverge do especificado em outra parte do modelo	Inconsistência
A11	Existe uma sequência cujas ações estão sendo executadas concorrentemente em outra parte do modelo	
A12	O rótulo da ação e/ou pré/pós condições não estão claras, sendo utilizadas expressões/verbos genéricos ou de duplo sentido	Ambiguidade
A13	A relação entre envio/recebimento de algum sinal não está clara, devido à descrição ambígua, podendo ser relacionado dois sinais diferentes para um mesmo evento	
A16	Uma Pré ou pós condição apresenta justificativas ou detalhes desnecessários	Informação Estranha
A17	Existe uma pré ou pós condição desnecessária, redundante, que já está representada no fluxo das ações	

Tabela 5.4. Exemplos de casos discrepantes identificados para os nós de decisão e de *merge*

Grupo D- Nós de decisão e de *merge*		
Cód.	**Descrição**	**Categoria de defeito**
D01	As condições de guarda não contemplam todas as possibilidades; está faltando alguma condição ou o "senão"	Omissão
D02	Alguma condição de guarda contraria a especificação	Fato Incorreto
D03	Existe uma condição de guarda que nunca poderá ser atendida.	
D06	Não deveria haver decisão/ *merge* em determinada sequência de ações ou o local da decisão/ *merge* não é o adequado	Inconsistência
D07	Uma mesma condição de guarda ou semelhante é adotada em decisões distintas com ações divergentes	
D08	Falta clareza na condição de guarda	Ambiguidade
D09	Existe "senão" sem necessidade, em redundância com as demais condições de guarda.	Informação Estranha

Os outros dois grupos dizem respeito à rastreabilidade entre a atividade inspecionada as demais atividades da especificação (grupo H), incluindo casos referentes à técnica de Pereira et al. (2009), e à aplicação do diagrama de atividades inspecionado na descrição de um caso de uso (grupo I), com base na técnica de modelagem de Massollar (2008).

A Tabela 5.5 apresenta exemplos dos casos discrepantes identificados para o grupo I. Importante observar que dois destes casos discrepantes não estão associados a uma única categoria de defeito (I09 e I12). Esta dupla categorização também ocorreu com outros casos discrepantes de outros grupos, envolvendo as categorias "Fato Incorreto" e "Inconsistência", devido às possibilidades de interpretações. Por exemplo, no caso discrepante "I09" além de uma inconsistência na descrição do modelo (se o modelo permite a situação descrita, ele está inconsistente com o restante da atividade), também poderíamos identificar que esta inconsistência ocorre por que os requisitos estão mal representados, o que nos leva a um fato incorreto.

Ao longo da pesquisa, foram identificados 98 casos discrepantes distintos, que foram utilizados como base para a construção dos *checklists* que compõem a técnica de inspeção apresentada nesta dissertação. Parte destes casos discrepantes também foi utilizada para o treinamento dos inspetores na técnica, para a execução do estudo de viabilidade da técnica. A tabela a seguir apresenta a quantidade de casos discrepantes identificados por categoria de defeito e grupo, sendo que 15 destes casos discrepantes podem ser categorizados tanto como fato incorreto quanto como

inconsistência. A lista completa dos casos discrepantes pode ser visualizada no Apêndice B.

Tabela 5.5. Exemplos de casos discrepantes identificados para a descrição de casos de uso

Grupo I- Estereótipos para descrição de casos de uso		
Cód.	**Descrição**	**Categoria de defeito**
I02	Uma regra de negócio especificada, relevante para uma ação, deixou de ser referenciada na atividade	Omissão
I06	Uma ação que representa uma inclusão não foi representada através do estereótipo <<include>>	
I08	Uma resposta do sistema <<system response>> não define as informações que são apresentadas ao ator	
I09	Uma condição de guarda de uma decisão imediatamente posterior a uma ação do ator (<<ator action>>) não está relacionada a esta ação	Fato Incorreto, Inconsistência
I12	Uma ação imediatamente posterior a uma resposta do sistema (<<system response>>) não está relacionada a uma ação do ator	
I17	Uma ação da atividade descreve a implementação do sistema, o que é inadequado para a descrição de um caso de uso	Informação Estranha

Tabela 5.6. Quantidade de casos discrepantes identificados por grupo e categoria de defeito

Grupo	**Categoria de Defeito**				
	Omissão	**Fato Incorreto**	**Inconsistência**	**Ambiguidade**	**Informação Estranha**
A) Nós de Ação e Fluxo de Ações	3	4(+2)	2(+2)	3	3
B)Nós de Controle- Início, Fim de Atividade e Fim de Fluxo	1	0(+2)	0(+2)	1	1
C)Nós de Controle- Nós de Bifurcação e Sincronização	3	4 (+3)	1(+3)	2	0
D)Nós de Controle- Nós de Decisão e de Merge	1	0(+4)	2(+4)	1	1
E)Regiões de Interrupção, Regiões de Expansão, Nós de Loop, Nós Condicionais e Exceções	3	6	1	2	1
F)Nós de Objeto e Fluxo de Objetos	1	6	3	5	2
G)Raias	0	1	1	1	1
H)Rastreabilidade entre Atividades	2	0	0	0	1
I)Estereótipos para descrição	8	0 (+4)	4(+4)	0	1

de casos de uso					
Total	**22**	**21 (+15)**	**29 (+15)**	**15**	**11**

5.3 Primeira Versão da Técnica de Inspeção

Com base em 92 casos discrepantes, foi elaborada a primeira versão da técnica de inspeção para apoiar a identificação de defeitos em especificações de requisitos descritas com diagrama de atividades, apresentada em de Mello et al. (2010). A aplicação da técnica tem como premissa que existe uma descrição textual de requisitos e que esta descrição já foi devidamente inspecionada.

A primeira versão da técnica de inspeção é composta de um *checklist* para apoiar a detecção de defeitos em especificações de requisitos descritas com diagramas de atividades, de um questionário de caracterização da aplicação para apoiar a configuração do *checklist* e de uma tabela de configuração, que contém as informações de rastreabilidade entre os itens de avaliação do *checklist* e os itens do questionário de caracterização da aplicação. Assim, conforme as respostas fornecidas ao questionário de caracterização, é possível reduzir o número de itens de avaliação do *checklist* antes de encaminhar o material para o inspetor, conforme o esquema apresentado na Figura 5.2. A técnica também possui um modelo de relatório para registro das discrepâncias identificadas pelo inspetor.

Como demonstração de sua aplicabilidade, esta primeira versão da técnica foi aplicada na inspeção da especificação de requisitos de uma aplicação real baseada em *workflow* científico (*e-Science*) utilizada por uma organização. A comparação entre os resultados obtidos através de aplicação da técnica de inspeção e os resultados obtidos anteriormente pela equipe de desenvolvimento, através de inspeções *ad-hoc*, indicam a viabilidade da aplicação da técnica em projetos de software e apontam algumas vantagens relacionadas ao número e tipos de defeitos encontrados (de Mello et al., 2010).

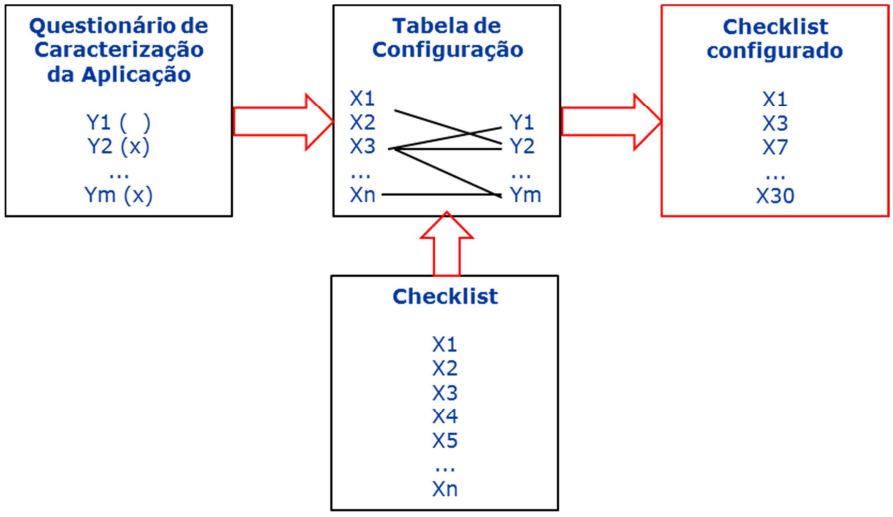

Figura 5.2. Aplicação dos artefatos da técnica para configuração do checklist.

5.3.1 *Checklist*

Com base nos 92 casos discrepantes, foi construído um *checklist* contendo 34 itens de avaliação. A princípio, cada caso discrepante poderia ser reescrito como um itens de avaliação (na forma de uma pergunta). Entretanto, observou-se que um *checklist* com 92 itens de avaliação seria muito extenso, além de tornar determinadas verificações do diagrama muito repetitivas. Deste modo, cada caso discrepante foi analisado quanto à sua afinidade com os demais, no que diz respeito a recursos sintáticos envolvidos e categorias de defeito, a fim de que mais de um caso discrepante pudesse ser contemplado em cada item de avaliação.

Para cada item do *checklist*, o inspetor deve assinalar uma das seguintes respostas: “Sim”, “Não” ou “N.A”. (não avaliável). O inspetor também deve informar no *checklist* preenchido o tempo consumido na realização de sua inspeção. No fim, o inspetor deve preencher um relatório de discrepâncias, descrevendo as discrepâncias encontradas, especificando o tipo de defeito e a questão que ajudou a identificar a discrepância.

Alguns itens de avaliação do *checklist* estão relacionados a um único caso discrepante, enquanto um item de avaliação do *checklist* representou 11 casos discrepantes. A agregação de casos discrepantes em uma única questão do *checklist* visou eliminar itens de avaliação semanticamente muito próximas, buscando fornecer maior objetividade na realização da inspeção.

Procurou-se organizar os itens de avaliação do *checklist* numa sequência conforme a perspectiva da atividade predominantemente observada nos itens de avaliação, conforme van der Aalst (2003): fluxo de controle, fluxo de dados, recursos

e, ao final, outros itens que dizem respeito a dependências entre atividades e o uso de estereótipos para descrição de casos de uso, conforme apresentado em (Massollar, 2008).

Fluxo de controle

Na perspectiva de fluxo de controle, com base na descrição textual dos requisitos, os itens de avaliação orientam a identificação de defeitos relativos à descrição das ações e restrições do diagrama de atividades e a forma como estes recursos sintáticos estão distribuídos no diagrama, utilizando-se de recursos como nós de decisão/*merge* e nós de bifurcação/ sincronização. Considerou-se como uma restrição qualquer recurso sintático que aplicasse uma condição que pudesse impactar o fluxo da atividade: pré e pós condições locais, condições de guarda, *JoinSpecs* e exceções. Posteriormente, o termo "restrição" foi devidamente substituído para "condição". A Tabela 5.7 apresenta os 15 itens de avaliação do *checklist* (1-14 e 16) que compõem esta perspectiva.

Tabela 5.7. Itens de avaliação do checklist referentes à perspectiva de fluxo de controle.

#	Item de Avaliação	Resposta
1	**As ações da atividade estão descritas com clareza e objetividade, fornecendo minimamente o verbo e o objeto da ação?**	( ) Sim ( ) Não ( ) N.A.
2	**O rótulo de cada ação está consistente e adequado à descrição dos requisitos?**	( ) Sim ( ) Não ( ) N.A.
3	**Cada fluxo de ações, fluxo de objetos e desvios da atividade levam a uma interpretação clara e coerente com a atividade?**	( ) Sim ( ) Não ( ) N.A.
4	**Cada fluxo de ações e desvios da atividade está consistente e adequado à descrição de requisitos?**	( ) Sim ( ) Não ( ) N.A.
5	**Alguma ação deixou de ser inserida na atividade?**	( ) Sim ( ) Não ( ) N.A.
6	**O início da atividade ou algum nó de fim de atividade ou fim de fluxo deixou de ser informado?**	( ) Sim ( ) Não ( ) N.A.
7	As restrições de tempo dos nós de recebimento de sinal da atividade estão em conformidade com a descrição de requisitos e coerentes com a própria atividade?	( ) Sim ( ) Não ( ) N.A.
8	A relação envio x recebimento de sinal está clara para todas as ações deste tipo descritas na atividade?	( ) Sim ( ) Não ( ) N.A.
9	Todos os recebimentos de sinal da atividade são viáveis?	( ) Sim ( ) Não ( ) N.A.
10	A atividade contempla as restrições (pré e pós condições locais,	( ) Sim ( ) Não ( ) N.A.

	condições de guarda, JoinSpecs, exceções) adequadas para representar o especificado na descrição dos requisitos?	
11	Todas as restrições da atividade (pré e pós condições locais, condições de guarda, JoinSpecs, exceções) foram descritas com clareza, objetividade e são viáveis?	( ) Sim ( ) Não ( ) N.A.
12	Existe alguma decisão (escolha exclusiva) que deveria ser tratada como escolha múltipla para representar corretamente a descrição dos requisitos?	( ) Sim ( ) Não ( ) N.A.
13	Existe alguma situação representada na atividade como escolha múltipla (bifurcação) que deveria ser representada como escolha exclusiva (decisão) para representar corretamente a descrição dos requisitos?	( ) Sim ( ) Não ( ) N.A.
14	Existem ações sendo executadas concorrentemente, mas que são executadas sequencialmente em outras partes do modelo?	( ) Sim ( ) Não ( ) N.A.
16	Todas as regiões de expansão esclarecem os critérios para repetição e estes critérios estão coerentes com a atividade e em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.

Importante observar que os seis primeiros itens de avaliação do *checklist*, embora mencionem outros recursos sintáticos, tratam dos recursos essenciais que toda atividade minimamente precisa ter (OMG, 2009): o nó de início da atividade, um nó de fim de atividade e uma ou mais ações encadeadas entre o início e o fim da atividade. Deste modo, os seis primeiros itens de avaliação do *checklist* são os únicos dos 34 itens que não dependem do questionário de caracterização para aparecerem no *checklist* de uma inspeção específica.

Fluxo de dados

Esta perspectiva está contemplada na primeira versão do *checklist* através de seis itens de avaliação. Nesta categoria de avaliação, não somente o fluxo dos dados propriamente dito é verificado, mas também a descrição de cada nó de objeto e propriedade que compõe este fluxo. A Tabela 5.8 apresenta estes seis itens.

Tabela 5.8. Itens de avaliação do *checklist* referentes à perspectiva de fluxo de dados.

#	Item de Avaliação	Resposta
15	Os objetos de entrada e saída das regiões de expansão estão consistentes e adequados ao especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
17	Os objetos representados na atividade e suas respectivas propriedades (critério de ordenação, estados, critério de seleção, limite) estão descritos com clareza, objetividade e em conformidade com a descrição	( ) Sim ( ) Não ( ) N.A.

	dos requisitos?	
18	Algum objeto externo utilizado/ retornado por uma subatividade empacotada deixou de ser informado como parâmetro de subatividade?	( ) Sim ( ) Não ( ) N.A.
19	Existe algum objeto que é especificado como entrada num empacotamento mas não é utilizado dentro da região empacotada da atividade?	( ) Sim ( ) Não ( ) N.A.
20	As propriedades dos objetos estão sendo respeitadas ao longo da atividade?	( ) Sim ( ) Não ( ) N.A.
21	Todas as transformações entre objetos estão descritas com clareza e em conformidade com a descrição de requisitos?	( ) Sim ( ) Não ( ) N.A.

Como a utilização de nós de objeto tipicamente não é obrigatória no desenho de uma atividade, foram evitadas questões que tratassem da omissão de objetos e propriedades. Somente no caso da passagem de objetos externos a uma atividade empacotada (questão 19) é que esta categoria de defeito foi considerada.

Recursos

A perspectiva de recursos envolve essencialmente a verificação da utilização adequada de raias em uma atividade. Entretanto, não foram considerados defeitos de omissão devido ao fato do uso de raias ser um recurso sintático também opcional. Nos dois itens de avaliação desta perspectiva, é verificada a compreensibilidade de cada raia definida e a consistência e corretude das ações executadas dentro de cada raia, conforme a descrição textual dos requisitos. A Tabela 5.9 apresenta os três itens de avaliação desta perspectiva.

Tabela 5.9. Itens de avaliação do *checklist* referentes à perspectiva de recursos.

#	Item de Avaliação	Resposta
22	Os fluxos agrupados em cada raia/ sub-raia da atividade estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
23	As raias e sub-raias da atividade estão claramente descritas?	( ) Sim ( ) Não ( ) N.A.

Dependências entre Atividades

Outros três itens de avaliação da primeira versão do *checklist* tratam da relação de dependência entre dois diagramas de atividades, seja utilizando o recurso sintático

de chamada de atividade especificado na UML (OMG, 2010), seja verificando a rastreabilidade entre dois diagramas que representam uma mesma atividade em níveis de detalhe distintos. A Tabela 5.10 apresenta estes três itens de avaliação

Tabela 5.10. Itens de avaliação do checklist referentes a dependências entre atividades.

#	Item de Avaliação	Resposta
24	Todas as outras atividades referenciadas nas ações existem no projeto?	( ) Sim ( ) Não ( ) N.A.
25	Alguma ação corresponde à outra atividade do projeto, mas esta atividade não é referenciada no diagrama?	( ) Sim ( ) Não ( ) N.A.
34	As ações de uma atividade num nível de detalhe menor estão claramente mapeadas nas ações de uma atividade correspondente num nível de detalhe maior?	( ) Sim ( ) Não ( ) N.A.

Descrição de Casos de Uso

A primeira versão do *checklist* orientava o inspetor na verificação da utilização adequada de estereótipos para a descrição de casos de uso (Massollar, 2008) através de oito itens de avaliação. Importante observar que, embora esta abordagem possa ser ou não considerada no desenho de um diagrama de atividades, se ela for considerada, a aplicação de seus estereótipos torna-se obrigatória para viabilizar a interpretação de cada diagrama de atividade como um caso de uso.

Tabela 5.11. Itens de avaliação do *checklist* referentes à descrição de casos de uso.

#	Item de Avaliação	Resposta
26	As ações referentes ao comportamento interno do sistema estão devidamente representadas através do estereótipo <<function>>?	( ) Sim ( ) Não ( ) N.A.
27	Os objetos que representam classes que mapeiam uma visão do modelo conceitual estão devidamente representados através do estereótipo <<view>>?	( ) Sim ( ) Não ( ) N.A.
28	Algum estereótipo foi utilizado indevidamente ao longo da atividade?	( ) Sim ( ) Não ( ) N.A.
29	Todas as coleções de dados geradas ou consumidas estão devidamente representadas através do estereótipo <<collection>>?	( ) Sim ( ) Não ( ) N.A.
30	Todas as ações em que o ator interage com o sistema e o sistema com o ator estão representadas com os estereótipos <<input>> e <<output>>, respectivamente?	( ) Sim ( ) Não ( ) N.A.
31	As consultas de informações para processamento estão devidamente	( ) Sim ( ) Não ( ) N.A.

	representadas (<<search>>)?	
32	Cada regra de negócio referenciada na atividade existe e seu uso está coerente com a descrição de requisitos?	( ) Sim ( ) Não ( ) N.A.
33	Os relacionamentos de inclusão e extensão estão devidamente referenciados no modelo através dos estereótipos <<include>> e <<extend>>, respectivamente?	( ) Sim ( ) Não ( ) N.A.

5.3.2 Questionário de Caracterização da Aplicação

Para apoiar a configurabilidade da técnica de inspeção, foi elaborado um questionário de caracterização da aplicação. Espera-se que o questionário seja preenchido pelo engenheiro de software responsável pelo projeto, devido à necessidade de conhecimento prévio da técnica de modelagem que foi aplicada aos diagramas de atividades que serão inspecionados.

Este questionário busca capturar essencialmente as informações relacionadas aos recursos sintáticos dos diagramas de atividades que podem ser aplicados na especificação de requisitos de um projeto de software. Nesta primeira versão do questionário, os recursos sintáticos opcionais do diagrama de atividades foram distribuídos nos 14 grupos a seguir:

1. Início/ Fim de atividade e Nós de Ação simples em sequência
2. Nó de decisão/ merge
3. Nó de bifurcação/ sincronização
4. Raias (*swimlanes*)
5. Nó de fim de fluxo
6. Nós de objeto simples
7. Nós de objeto com propriedades, estereótipos e transformações
8. Pré e pós-condições locais
9. Empacotamento de Atividades
10. Exceções
11. Região de Expansão
12. Região de Interrupção, Envio e Recebimento de Sinal
13. Referência a outras atividades
14. Estereótipos para descrição de casos de uso

O questionário de caracterização da aplicação também contempla uma questão para identificar se a técnica de modelagem adotada no projeto contempla a possibilidade de dois diagramas descreverem uma mesma atividade em níveis de

detalhe distintos. Esta situação é prevista, por exemplo, na abordagem de concepção de *workflows* científicos abstratos, apresentada em Pereira et al. (2009).

Importante destacar que deve ser considerado o que está *previsto* na técnica de modelagem, e não somente aquilo que é *aplicado* nos diagramas de atividades de um projeto. Por exemplo, pode estar prevista a utilização de chamadas de atividade e o desenvolvedor não identificar as ações de chamada de atividade devidamente, embora elas existam. Logo, a mera observância dos modelos a serem inspecionados para a caracterização da aplicação implicaria no risco de discrepâncias, especialmente omissões, serem ignoradas pelo inspetor.

Com o questionário de caracterização da aplicação devidamente preenchido, é possível utilizar uma tabela contendo informações de rastreabilidade entre os itens do questionário e os itens de avaliação do *checklist* para identificar o conjunto de itens de avaliação que devem fazer parte do *checklist* de determinado projeto, direcionando o foco do inspetor para os itens de avaliação realmente pertinentes ao contexto do projeto. Por exemplo, se a técnica de modelagem aplicada no projeto não prevê a utilização de nós de objeto no diagrama de atividades, seis dos 34 itens de avaliação do *checklist* já poderão ser descartados. Em outra situação, quando não estiver prevista a descrição de casos de uso através dos estereótipos já mencionados, oito itens também poderão ser previamente eliminados. Conforme já mencionado, dos 34 itens de avaliação do *checklist* original, apenas os 6 primeiros são considerados essenciais para qualquer diagrama de atividades.

5.4 Prova de Conceito

Para exemplificar a aplicabilidade da versão inicial da técnica de inspeção, uma prova de conceito aplicando a técnica a uma especificação de requisitos de software baseada em *workflow* científico foi realizada e seus resultados apresentados em de Mello et al. (2010). Esta especificação faz parte do projeto de colaboração de pesquisa e desenvolvimento Galileu, conduzido pela COPPE/UFRJ em parceria com a Petrobras, para apoiar na simulação da extração de petróleo por plataformas em águas profundas através do uso de dutos, chamados *risers*.

Os *risers* são fixados ao poço e, por consequência, sofrem a ação de diversas condições ambientais, como ventos e ondas. Tais ações podem provocar fissuras nos *risers*, devido à fadiga. Para aperfeiçoar a configuração entre plataforma, *risers* e ambiente, os pesquisadores utilizam um conjunto de programas que simulam a passagem do tempo e os possíveis eventos que podem ocorrer na estrutura. A combinação ordenada da aplicação destes representa uma rodada de experimento.

Para especificar a ordem de execução destes programas e apresentar todos os recursos e restrições aplicáveis aos diferentes cenários de execução, utiliza-se a representação de *workflows* científicos. A Tabela 5.12 apresenta um exemplo de formulário de atividade, baseado na abordagem de Pereira et al. (2010). A Figura 5.3 apresenta um extrato do *workflow científico* utilizado nesta prova de conceito.

Tabela 5.12. Exemplo de formulário de atividade (de Mello et al., 2010).

Nome	ANÁLISE ACOPLADA DE MOVIMENTOS DA PLATAFORMA		
Descrição	Realiza as análises de movimentos da plataforma, empregando uma formulação acoplada que incorpora um modelo hidrodinâmico, para representação do casco da unidade flutuante, e um modelo de elementos finitos, para representar as linhas de ancoragem e risers. Nesta atividade também há a geração das séries temporais de movimentos da unidade flutuante e de esforços de suas linhas, quando submetida a várias combinações ambientais (coeficiente de onda, coeficiente de vento e coeficiente de correnteza).		
Tipo de atividade	Automatizada	Obrigatoriedade	Obrigatória
Ferramentas	Prosim		
Insumos	Arquivo .wnf ; Arquivo .prm; Arquivo .fun		
Produtos	Série de movimento da unidade flutuante e de esforços nas linhas do sistema[1]		
Papeis	Projetista de ancoragem e/ou projetista de risers.		
Pré-condições	Requer monitoramento da execução.		
Pós-condições	Nenhuma.		
Pré-atividades	Geração de modelo numérico.		
Subatividades	Nenhuma.		
Capacidade de paralelismo	Distribuição das análises de casos ambientais em diferentes nós. Paralelismo da alocação de uma única análise de um caso ambiental em diversos nós (paralelização das malhas de Elementos Finitos que representam as linhas do modelo).		
Riscos associados	A execução pode ser interrompida por influência de um problema numérico.		
Frequência de utilização	Executada para cada número de combinações ambientais e por ciclos de projetos.		
Custo computacional	Relativo à complexidade do modelo, número de combinações ambientais e tempo total de simulação.		
Outros comentários	[1] contém grupos de arquivos de saída para cada caso de carregamento.		

No contexto do projeto Galileu, esta especificação de requisitos já havia sido inspecionada, de modo *ad hoc,* por cinco pesquisadores da COPPE/UFRJ com conhecimento prévio de UML, mais especificamente de diagramas de atividades e com conhecimento parcial do domínio do problema. No total de suas inspeções individuais, estes pesquisadores detectaram 81 defeitos distintos. Desse conjunto de defeitos, seis estavam relacionados ao *workflow* científico propriamente dito,

representado através de um diagrama de atividades. Entretanto, destes seis defeitos, apenas dois são defeitos relacionados à correspondência semântica entre o diagrama de atividades e a descrição textual de requisitos, enquanto que os outros defeitos detectados são referentes às inadequações sintáticas do modelo.

Diferentemente das inspeções ad hoc previamente realizadas, na prova de conceito o escopo da inspeção foi reduzido, focando a identificação defeitos apenas no *workflow*, com base no restante da especificação de requisitos,

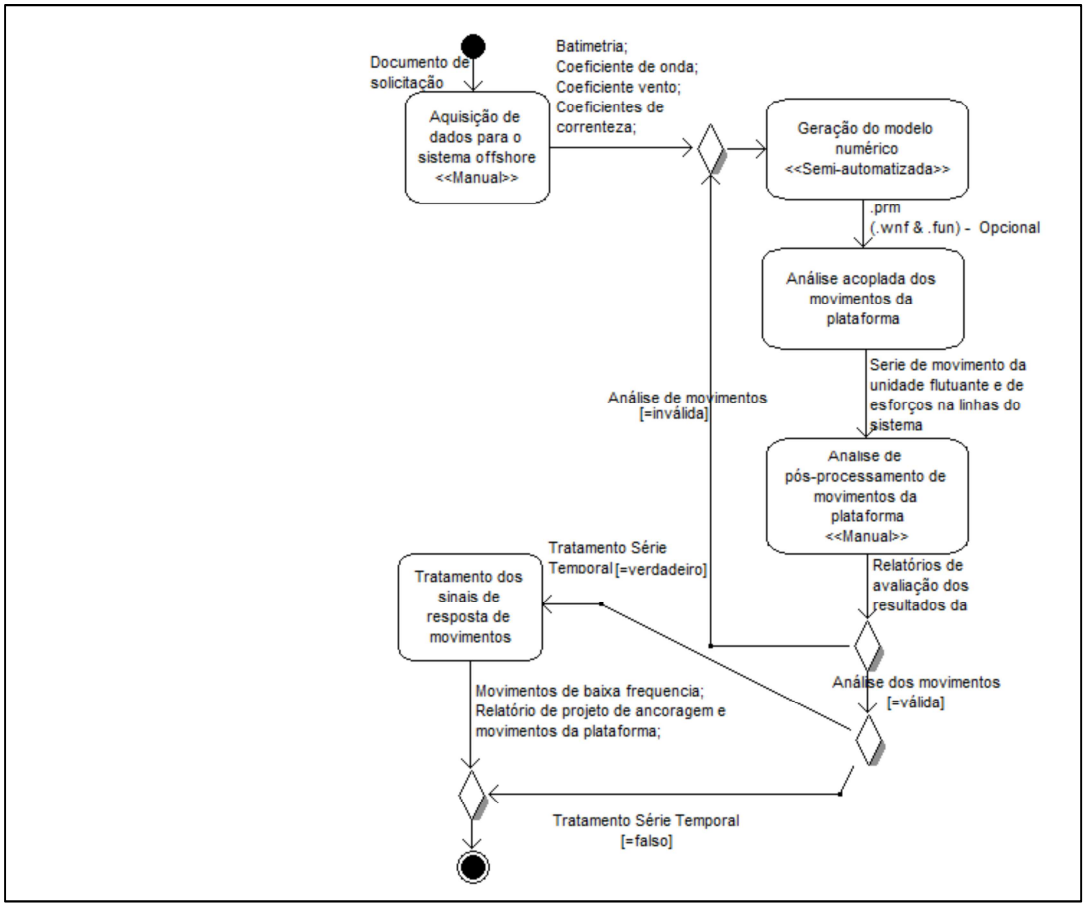

Figura 5.3. Extrato do workflow científico (de Mello et al., 2010).

5.4.1 Execução e Resultados Obtidos

Para a execução da prova de conceito, o questionário de caracterização da aplicação da técnica de inspeção foi preenchido, considerando-se os recursos previstos na técnica de modelagem adotada, o que levou a serem descartados 15 itens de avaliação do *checklist* completo. Este *checklist* configurado, contendo 19 itens de avaliação, foi utilizado por um inspetor externo com conhecimento de diagrama de

atividades e nenhum conhecimento no domínio do problema para realizar a inspeção da especificação do *workflow*.

Com o apoio dos 19 itens de avaliação do *checklist* configurado, este inspetor detectou 25 discrepâncias em 117 minutos dedicados à inspeção. Estas discrepâncias foram encaminhadas para análise do desenvolvedor responsável pela especificação e, após uma reunião entre o inspetor e o desenvolvedor para avaliação das discrepâncias, foram descartados cinco falsos positivos. Também foi observado que a origem de quatro destes cinco falsos positivos foi devido a problemas na descrição textual dos requisitos, mais especificamente nos formulários originais utilizados na inspeção para efeito de se estabelecer um *baseline* comum com a inspeção *ad hoc* realizada previamente. Neste sentido, é importante observar que a aplicação do *checklist* deve considerar que a descrição dos requisitos está correta para que se possam encontrar defeitos no diagrama de atividades.

Quanto aos 20 defeitos detectados na prova de conceito, apenas um deles fazia parte da relação de defeitos semânticos detectados na inspeção *ad hoc* conduzida pelos cinco inspetores. Este defeito é relativo a uma inconsistência na decisão sobre a ação de "análise da série temporal" (Figura 5.3), capturada através da questão 4 do *checklist*: *"Cada fluxo de ações e desvios da atividade está consistente e adequado à descrição de requisitos?".*

O outro defeito semântico capturado pela inspeção *ad hoc,* referente ao diagrama de atividades, e que não foi detectado na prova de conceito trata-se da não utilização do termo "arquivo" em um objeto da atividade "Geração do modelo numérico" (Figura 5.3). Apesar da não-detecção deste defeito, foi observado que o defeito fazia parte do escopo do item de avaliação 17 do *checklist ("Os objetos representados na atividade e suas respectivas propriedades estão descritos com clareza, objetividade e em conformidade com a descrição dos requisitos?")*, não tendo sido detectado, portanto, devido à falha humana do inspetor. Não obstante, este item de avaliação é muito genérico, o que pode também ter contribuído para não-detecção do defeito. Entretanto, é importante observar que, apesar de uma inspeção através de *checklist* ser mais estruturada que uma inspeção *ad hoc*, seus resultados ainda sofrem influência do inspetor.

Considerando-se as categorias dos defeitos encontrados no diagrama de atividades através das inspeções *ad hoc* e da inspeção através de *checklist*, foram obtidos os valores reproduzidos na Tabela 5.13. É possível observar que a maior parte dos defeitos detectados na prova de conceito, mais da metade, foram classificados como inconsistências, o que sugere a importância de uma técnica de inspeção na verificação da rastreabilidade entre a descrição textual de requisitos e um modelo que

a descreve, no caso o diagrama de atividades. Quanto a pouca quantidade de ambiguidades e a não-detecção de informações estranhas, destacamos que a abordagem proposta por Pereira et al. (2009) exige uma formalização significativa da especificação dos requisitos através de formulários padronizados, o que pode estimular os desenvolvedores que fazem uso desta abordagem a evitar defeitos destas categorias.

Tabela 5.13. Quantidade de defeitos no diagrama de atividades detectados nas inspeções *ad hoc* e na prova de conceito, conforme a categoria de defeito relatada (de Mello et al., 2010)

Tipo de defeito	*ad hoc*		*checklist*	
	Qtd.	%	Qtd.	%
Omissão	2	100	5	25
Ambiguidade	0	0	1	5
Inconsistência	0	0	11	55
Fato incorreto	0	0	3	15
Informação estranha	0	0	0	0
Total	**2**	**100**	**20**	**100**

Numa análise geral dos resultados das inspeções conduzidas, observa-se que 10 vezes mais defeitos foram identificados através da aplicação do *checklist* por apenas um inspetor sem qualquer conhecimento anterior sobre o domínio do problema. Considerando-se os defeitos distintos encontrados pelos cinco inspetores em todo o documento (81), é importante observar também que cada inspetor gastou, em média, 107 minutos para realizar sua inspeção *ad-hoc*. Já o inspetor que aplicou o *checklist* nesta prova de conceito, levou 117 minutos para detectar 20 defeitos somente numa parte da documentação. Não se pode afirmar, entretanto, que a técnica de inspeção reduz o tempo investido pelo inspetor. Contudo, estes números indicam que, com uma quantidade de tempo semelhante, o inspetor que aplicou o *checklist* encontrou mais defeitos no total, mesmo buscando defeitos em apenas uma parte da documentação, o diagrama de atividades.

Como ameaças à validade desta prova de conceito, ressaltamos o fato do escopo da inspeção *ad hoc* ser mais amplo (toda a especificação de requisitos) do que o escopo da inspeção apoiada pela técnica de inspeção, apesar de ambas as inspeções utilizarem a mesma versão da documentação. Outra ameaça reside no fato do inspetor que aplicou o *checklist* nesta prova de conceito, também ser o pesquisador que elaborou a técnica de inspeção, o que pode ter contribuído para garantir um resultado favorável à inspeção.

5.5 ActCheck

Após a aplicação da prova de conceito, a técnica foi revisada, sendo identificadas oportunidades de melhoria na sua estrutura. Esta segunda versão da técnica recebeu o nome de ActCheck. No que se refere a modificações no *checklist*, alguns itens de avaliação foram reescritos, enquanto outros foram acrescentados ou substituídos. Além disto, percebeu-se que alguns destes itens não levavam a identificação de defeitos entre a especificação de requisitos e o diagrama de atividades, mas sim representavam uma oportunidade para a verificação da consistência interna do próprio modelo ou entre diagramas, o que também poderia contribuir para detecção de defeitos na própria descrição textual dos requisitos. Como consequência, os itens de avaliação de ActCheck foram divididos entre dois *checklists*: o *checklist* A, contendo itens de avaliação específicos para verificação da rastreabilidade entre os diagramas de atividades e a descrição textual dos requisitos; e o *checklist* B, a ser aplicado após o *checklist* A para verificar a consistência interna de um diagrama de atividades e a sua relação com os outros diagramas de atividades em especificações de requisitos.

O questionário de caracterização da aplicação também sofreu modificações, sendo reorganizados os recursos sintáticos do diagrama de atividades na questão 1 e sendo acrescentados novos itens de avaliação. Consequentemente, as informações de rastreabilidade entre os itens de avaliação e os itens do questionário também foram modificadas. As seções a seguir apresentam o processo de aplicação de ActCheck numa inspeção e os artefatos que compõem a técnica. Os artefatos completos podem ser visualizados no Apêndice C desta dissertação.

5.5.1 Processo de Aplicação de ActCheck

A Figura 5.4 apresenta o processo de aplicação de ActCheck na inspeção da especificação de requisitos de um projeto. Primeiramente, é necessário que a equipe de inspetores esteja definida e que a descrição textual dos requisitos esteja previamente inspecionada. Em seguida, com base na técnica de modelagem adotada no projeto e no perfil dos inspetores, o responsável pelo desenvolvimento da aplicação irá preencher o Questionário de Caracterização da Aplicação. O processo segue com a configuração dos *checklists* A' e B', ou seja, a identificação dos itens de avaliação dos *checklists* A e B que serão utilizadas na inspeção em questão. Esta definição é feita através da correspondência entre as respostas do Questionário de Caracterização da Aplicação e as informações de rastreabilidade contidas nas tabelas

de configuração referentes aos dois *checklists* originais de ActCheck. Em seguida, cada inspetor realiza sua inspeção individual, preenchendo uma cópia dos checklists A' e B' e informando as discrepâncias identificadas em seu relatório de discrepâncias.

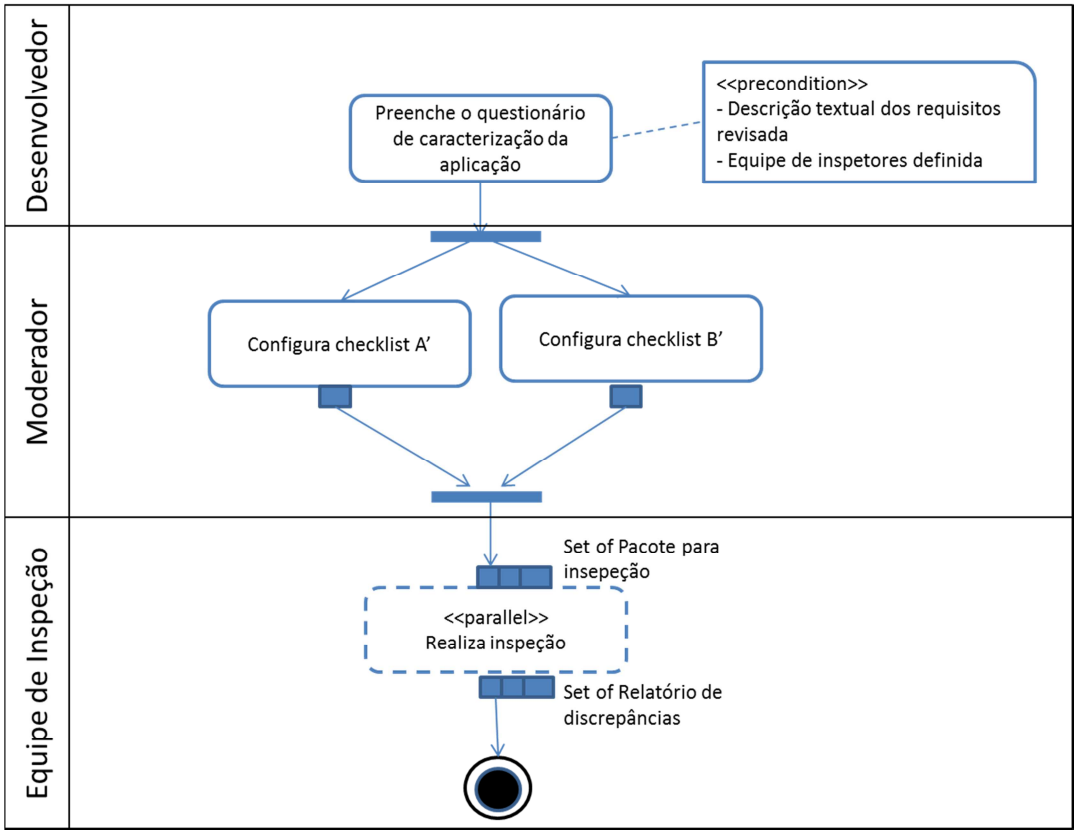

Figura 5.4. O processo de aplicação de ActCheck.

5.5.2 Checklist A

O Checklist A é composto de 39 itens de avaliação que orientam o inspetor a identificar discrepâncias entre os diagramas de atividades de uma especificação de requisitos e a descrição textual destes requisitos. Os itens de avaliação do Checklist A independem de como a descrição textual dos requisitos foi feita, embora valha ressaltar que a aplicação de técnicas estruturadas como as de Massollar (2008) e Pereira et al. (2009) possam facilitar a identificação do escopo da inspeção de requisitos a partir da rastreabilidade entre os artefatos da especificação.

Buscando facilitar a inspeção, os itens de avaliação do Checklist A foram agrupadas conforme seus respectivos focos de verificação:

- ações e condições;
- fluxo de controle;
- objetos e fluxo de dados;

- raias;
- estereótipos para casos de uso.

Em relação ao agrupamento da primeira versão técnica, pode-se observar que o grupo de ações e fluxo de controle foi dividido em dois grupos (ações e condições-fluxo de controle), separando as verificações individuais de cada recurso sintático que compõe o fluxo de controle da verificação do próprio fluxo de controle.

Diferentemente da primeira versão da técnica, em cada um destes cinco grupos, os itens de avaliação foram organizados de modo a orientar o inspetor a buscar, nesta ordem, discrepâncias referentes às seguintes categorias de defeito: omissões, inadequações (ambiguidades, inconsistências e fatos incorretos) e informações estranhas.

Ações e Condições

O primeiro grupo de itens de avaliação de ActCheck orienta a detecção de discrepâncias nas ações e condições que compõem a atividade, como pode ser visto na Tabela 3.14. Entende-se como condição alguma regra que possa afetar o fluxo da atividade: pré e pós-condições locais, condições de guarda, restrições para junção (*joinspecs*), exceções, interrupções e critérios de repetição (OMG, 2009). Para os critérios de repetição das regiões de expansão e para ações especiais que representam envio/ recebimento de mensagens e ações baseadas em tempo, foram adotadas questões específicas.

Tabela 5.14. Itens de Avaliação do Checklist A referentes a Ações e Condições

#	Item de Avaliação	Resposta
1	**Todos os rótulos das ações da atividade informam qual é a ação e qual é o objeto de cada ação?**	**( ) Sim ( ) Não ( ) N.A.**
2	Todos os nós de loop esclarecem os critérios para sua repetição?	( ) Sim ( ) Não ( ) N.A.
3	**As ações e condições da atividade estão em conformidade com o especificado nos requisitos?**	**( ) Sim ( ) Não ( ) N.A.**
4	As ações temporais (sinal baseado em condição temporal) da atividade estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
5	Os critérios para repetição dos nós de loop estão claramente descritos?	( ) Sim ( ) Não ( ) N.A.
6	**As ações e condições (pré e pós condições locais, condições de**	**( ) Sim ( ) Não ( ) N.A.**

	guarda, JoinSpecs, exceções, interrupções, critérios de teste dos nós condicionais) da atividade foram descritas com clareza?	
7	Os critérios para repetição dos nós de loop estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
8	**Alguma ação ou condição da atividade contém informação desnecessária?**	**( ) Sim ( ) Não ( ) N.A.**

A Figura 5.5 apresenta um exemplo de fato incorreto em um trecho do diagrama de atividades, que pode ser detectado com o auxílio do item de avaliação 3. Neste diagrama, a ação de autorizar desconto é realizada pelo caixa, embora a descrição textual dos requisitos informe que o gerente autoriza qualquer desconto. A Figura 5.5 apresenta duas condições de guarda que estão ambíguas, caracterizando dois defeitos no diagrama de atividades. Neste exemplo, percebe-se que não ficaria claro para o leitor do diagrama o que significa “entregar fora” ou “entregar dentro”. Estes defeitos poderiam ser evitados se as condições de guarda fossem “fora do prazo de devolução” e “dentro do prazo de devolução”, respectivamente.

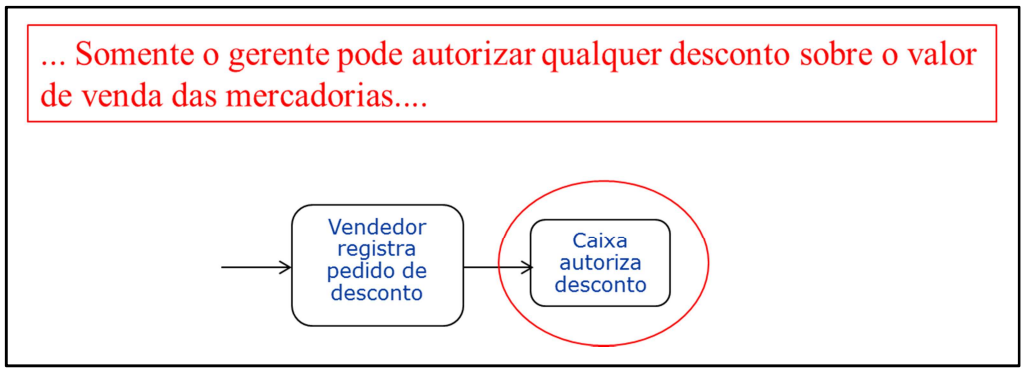

Figura 5.5. Exemplo de defeito numa ação, capturado pelo item 3 do checklist A

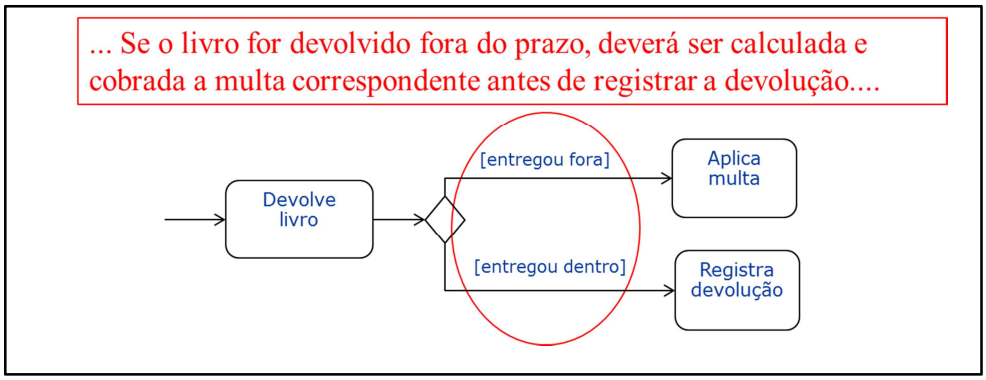

Figura 5.6. Exemplo de defeito em condições de guarda de uma atividade, capturado pelo item 6 do checklist A

Fluxo de Controle

Após a verificação das ações e das condições da atividade, o checklist A orienta o inspetor a detectar defeitos no fluxo de controle da atividade comparando-a com a interpretação feita da descrição textual dos requisitos através de sete itens de avaliação conforme pode ser visualizado na Tabela 5.15. Por exemplo, a situação apresentada na Figura 5.7, levaria a resposta do item de avaliação 12 a ser negativa, uma vez que as duas ações apresentadas estão em sequência, quando deveriam estar em paralelo para atender à descrição textual dos requisitos.

Tabela 5.15. Itens de avaliação do Checklist A referentes a fluxo de controle.

#	Item de Avaliação	Resposta
9	**Alguma ação ou condição deixou de ser inserida na atividade, numa região ou num nó protegido?**	( ) Sim ( ) Não ( ) N.A.
10	**Algum nó de fim de atividade ou de fim de fluxo deixou de ser informado?**	( ) Sim ( ) Não ( ) N.A.
11	As regiões de expansão, de interrupção, nós de loop, nós condicionais e as regiões protegidas (exceções) descrevem suas ações de maneira coerente com a especificação?	( ) Sim ( ) Não ( ) N.A.
12	**Cada fluxo da atividade está em conformidade com o especificado nos requisitos?**	( ) Sim ( ) Não ( ) N.A.
13	Existe alguma decisão (escolha exclusiva) que deve ser tratada como escolha múltipla (bifurcação) para representar corretamente a descrição dos requisitos?	( ) Sim ( ) Não ( ) N.A.
14	Existe alguma escolha múltipla (bifurcação) que deve ser representada como uma escolha exclusiva (decisão) para representar corretamente a descrição dos requisitos?	( ) Sim ( ) Não ( ) N.A.
15	**Cada fluxo da atividade permite interpretar claramente a atividade?**	( ) Sim ( ) Não ( ) N.A.

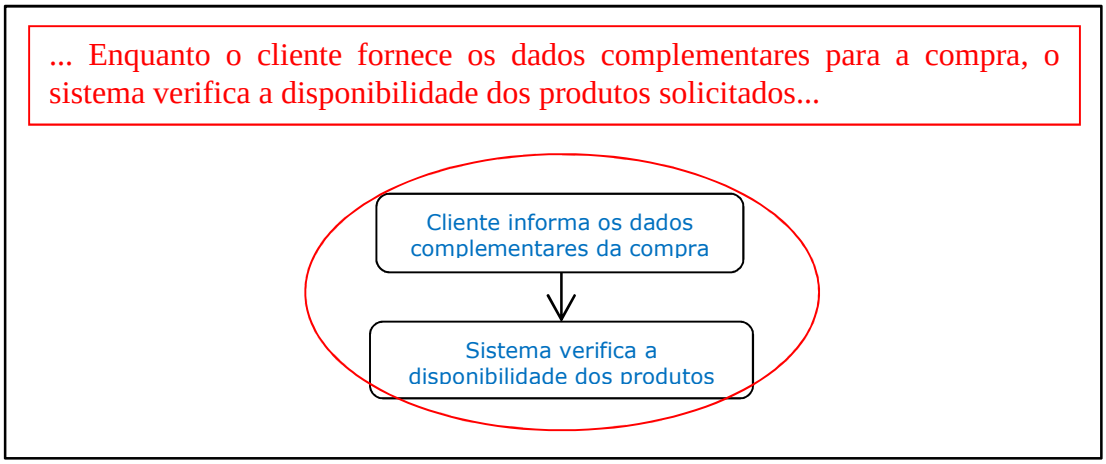

Figura 5.7. Exemplo de defeito no fluxo de controle de um diagrama de atividades, capturado pelo item de avaliação 12 do Checklist A.

Objetos e Fluxo de Dados

O terceiro conjunto de itens de avaliação do Checklist A orienta o inspetor buscar discrepâncias no fluxo de dados do diagrama de atividades, representado através dos nós de objeto e de suas respectivas propriedades. Estes itens são apresentados na Tabela 5.16. Como a utilização dos nós de objeto depende da necessidade/ contexto de cada projeto, tomou-se o cuidado de evitar a identificação de omissões na criação destes recursos sintáticos. A única exceção está descrita no item 16, quando se espera que objetos externos a uma atividade agrupada sejam fornecidos.

Tabela 5.16. Itens de avaliação do Checklist A referentes a objetos e fluxo de dados

#	Item de Avaliação	Resposta
16	Algum objeto externo utilizado/ retornado por uma subatividade agrupada não está sendo informado como parâmetro de subatividade?	( ) Sim ( ) Não ( ) N.A.
17	Todas as transformações entre objetos estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
18	Os objetos de entrada e de saída de cada ação, região de expansão ou subatividade estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
19	Os objetos da atividade e suas respectivas propriedades estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
20	Os objetos da atividade e suas respectivas propriedades (critérios de ordenação, estados, critérios de seleção, limites, persistência, isMulticast, isMultireceive, isControlType) estão claramente descritos?	( ) Sim ( ) Não ( ) N.A.
21	Todas as transformações entre objetos estão claramente descritas?	( ) Sim ( ) Não ( ) N.A.
22	Algum objeto da atividade, propriedade de objeto ou transformação de objeto apresenta informação desnecessária/irrelevante para o contexto da atividade?	( ) Sim ( ) Não ( ) N.A.

A Figura 5.8 apresenta um defeito em que o critério de ordenação do objeto “Empregado” aparentemente está descrito com clareza, mas a propriedade “data de registro” mencionada está ambígua, pois na descrição textual dos requisitos existem “data de registro da admissão” e “data de registro no cadastro”. Este defeito poderia ser capturado com o auxílio do item de avaliação 19.

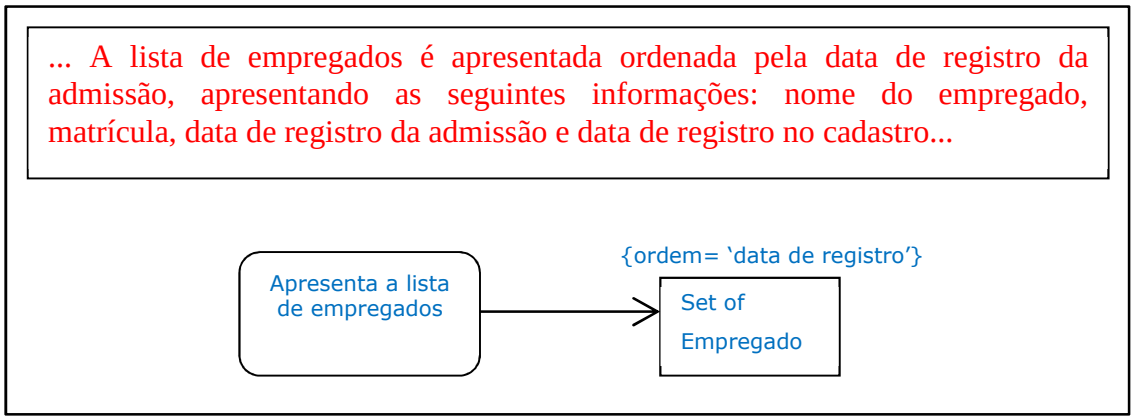

Figura 5.8. Exemplo de defeito na declaração das propriedades de um objeto, capturado através do item de avaliação 19 do Checklist A.

A Figura 5.9 apresenta um exemplo de fato incorreto (ou inconsistência) que indica o não-atendimento do diagrama de atividades inspecionado ao item de avaliação 18 do Checklist A. Esperava-se que o *pin* de entra da ação "Aprova projeto de lei" fosse "projeto de lei", e não "lei", que seria o *pin* de saída desta ação.

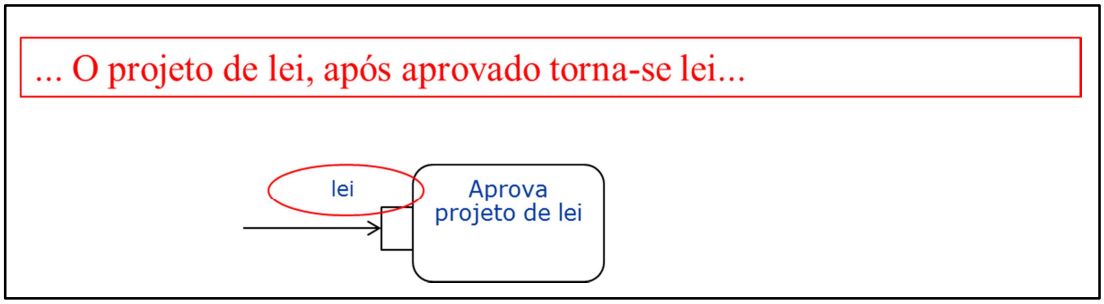

Figura 5.9. Exemplo de defeito em um *pin* de entrada de uma ação.

Raias

A utilização adequada do recurso sintático de raias no *checklist* A é verificada através das dos itens de avaliação 23, 24 e 25, apresentados na Tabela 5.17. Assim como o uso dos nós de objeto, não há como determinar que uma raia foi omitida, mas é possível verificar a descrição da raia e se existe consistência entre o escopo de atuação de cada raia e a descrição textual dos requisitos.

A figura 5.10 apresenta um exemplo contendo dois defeitos: no primeiro, não está clara a definição de uma das raias (item de avaliação 24), rotulada como "Equipe". Conforme a descrição textual dos requisitos, esta raia deveria se chamar "Equipe de contabilidade". O outro defeito consiste no fluxo agrupado à raia "Gerente de Equipe", que não está em conformidade com a descrição dos requisitos (item 23).

De fato, quem avalia o pedido de desligamento é a equipe de RH, e não o gerente de equipe.

Tabela 5.17. Itens de avaliação do Checklist A referentes ao uso de raias.

#	Item de Avaliação	Resposta
23	Os fluxos agrupados em cada raia/ sub-raia da atividade estão em conformidade com o especificado nos requisitos?	( ) Sim ( ) Não ( ) N.A.
24	As raias e sub-raias da atividade estão claramente descritas?	( ) Sim ( ) Não ( ) N.A.
25	Alguma raia ou sub-raia da atividade apresenta alguma informação desnecessária/irrelevante ao contexto da atividade?	( ) Sim ( ) Não ( ) N.A.

Estereótipos para Descrição de Casos de Uso

O último grupo de itens de avaliação do Checklist A destina-se basicamente à verificação da utilização adequada dos estereótipos para descrição de um caso de uso. Neste grupo houve mudanças nos itens de avaliação, sendo retirados itens de avaliação referentes a estereótipos que foram removidos da versão mais recente da abordagem apresentada por Massollar (2008). Também foram acrescentados itens de avaliação referentes à coerência do fluxo de controle da atividade com o fluxo dos passos de um caso de uso. A Tabela 5.18 apresenta os itens que fazem parte deste grupo.

A Figura 5.10 apresenta um exemplo de defeito que poderia ser capturado com o auxílio do item de avaliação 26 do Checklist A. De acordo com a descrição textual dos requisitos, existe uma regra importante relacionada ao cadastro de uma senha provisória para o usuário, a RN01. Entretanto, esta regra foi omitida na ação “Cadastra uma senha provisória para o usuário” do diagrama de atividades correspondente.

Sobre fatos incorretos e/ou inconsistências, são apresentados exemplos nas Figura 5.11, 5.13 e 5.14.. No primeiro exemplo, tratado pelo item de avaliação 34, a ação que precede a resposta do sistema “Apresenta relatório atualizado” não corresponde a uma ação do ator, apesar de ser rotulada com o estereótipo <<actor action>>. Na verdade, de acordo com a descrição textual dos requisitos, a ação “Verifica atualizações na base de clientes” é uma ação do sistema, que deveria vir *antes* da resposta do sistema. No segundo exemplo, Figura 5.13, de fato incorreto/discrepância, tratado pelo item 33, deduz-se que a condição de guarda “Sai do sistema” não pode estar relacionada à ação do ator “Verifica incidência de multa”.

Como exemplo de inconsistência relacionada ao item de avaliação 37, a Figura 5.13 apresenta uma ação que referencia uma regra de negócio 17 (RN17) que está grafada de outra forma na descrição textual de requisitos.

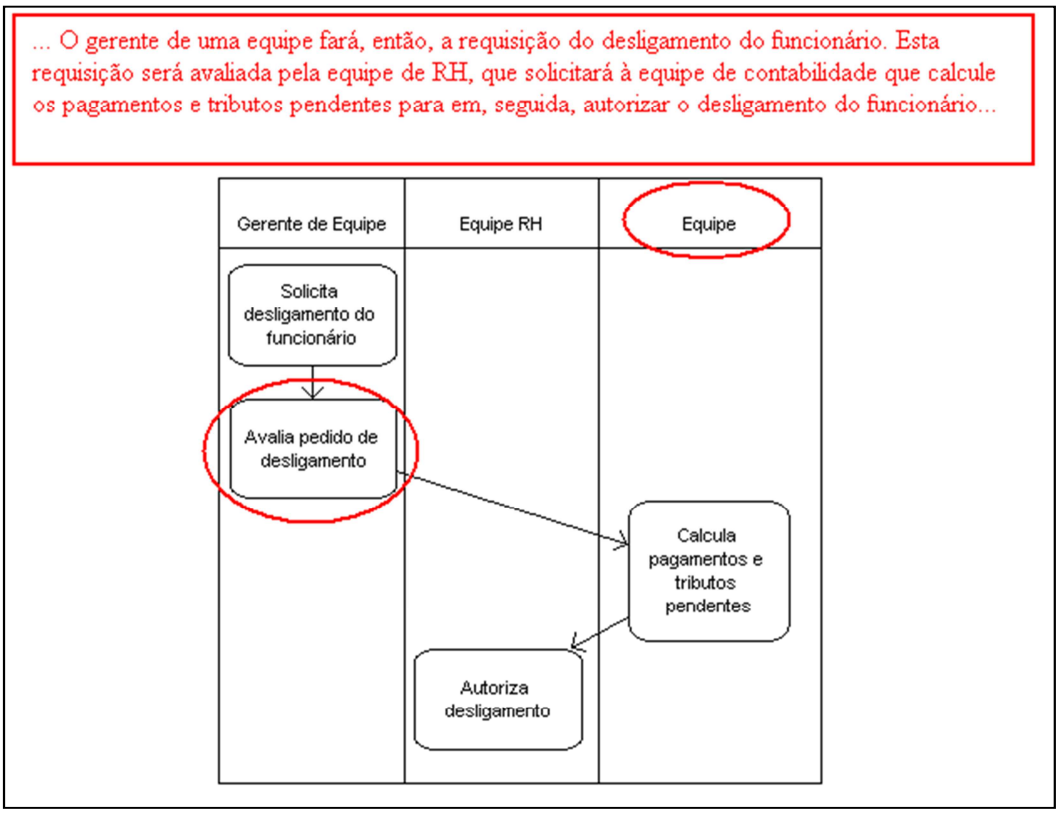

Figura 5.10. Exemplo de defeito envolvendo o uso de raias em um diagrama de atividades.

Tabela 5.18. Itens de avaliação do Checklist A referentes à descrição de casos de uso.

#	Item de Avaliação	Resposta
26	Alguma regra de negócio relevante para o contexto de uma determinada ação da atividade e especificada nos requisitos deixou de ser referenciada na atividade?	( ) Sim ( ) Não ( ) N.A.
27	Os relacionamentos de inclusão e extensão estão devidamente referenciados no modelo através dos estereótipos <<include>> e <<extend>>?	( ) Sim ( ) Não ( ) N.A.
28	As ações referentes ao comportamento interno do sistema estão devidamente representadas através do estereótipo <<system action>>?	( ) Sim ( ) Não ( ) N.A.
29	Todas as ações em que o ator interage com o sistema e o sistema com o ator estão representadas, respectivamente, através dos estereótipos <<actor action>> e <<system response>>?	( ) Sim ( ) Não ( ) N.A.
30	Nas ações do ator (<<actor action>>) estão definidas as informações que este fornece ao sistema?	( ) Sim ( ) Não ( ) N.A.
31	Nas respostas do sistema (<<system response>>), estão definidas as informações que são apresentadas ao ator?	( ) Sim ( ) Não ( ) N.A.
32	As condições de guarda que precedem uma ação do ator (<<actor action>>) estão relacionadas com as informações fornecidas nessa ação?	( ) Sim ( ) Não ( ) N.A.

33	As condições de guarda que precedem uma ação do sistema (<<system action>>) estão relacionadas com os resultados gerados nessa ação?	( ) Sim ( ) Não ( ) N.A.
34	Cada condição de guarda ou a ação que precede uma resposta do sistema está relacionada a uma ação do ator?	( ) Sim ( ) Não ( ) N.A.
35	Toda ação da atividade está descrita num nível de abstração adequado para a descrição de um caso de uso?	( ) Sim ( ) Não ( ) N.A.
36	Alguma ação adota uma regra de negócio que não tem a ver com seu escopo?	( ) Sim ( ) Não ( ) N.A.
37	As regras de negócio referenciadas na atividade existem na especificação?	( ) Sim ( ) Não ( ) N.A.
38	Algum estereótipo está sendo indevidamente utilizado ao longo da atividade?	( ) Sim ( ) Não ( ) N.A.
39	Alguma ação da atividade detalha parte da solução computacional (como é feito)?	( ) Sim ( ) Não ( ) N.A.

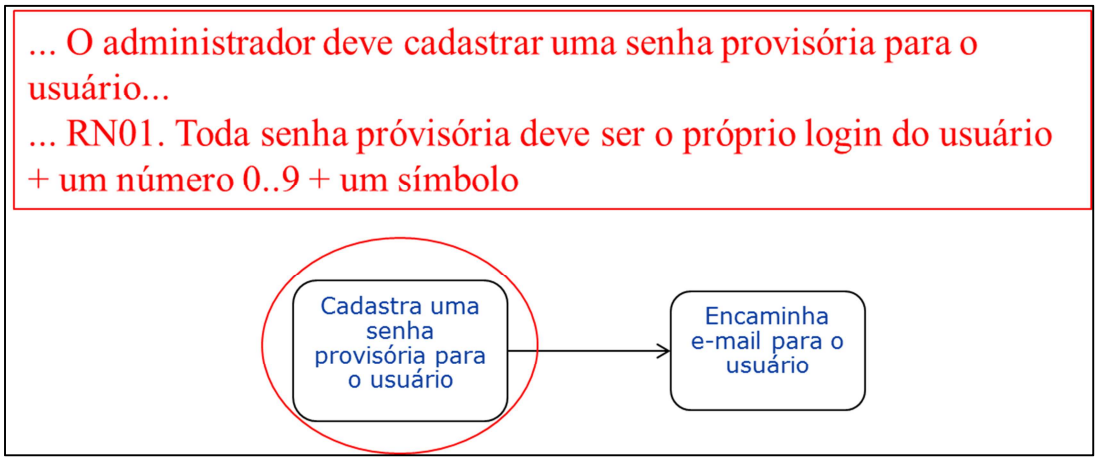

Figura 5.10. Exemplo de omissão em uma ação, onde uma regra de negócio relevante deixou de ser referenciada (item 26 do Checklist A).

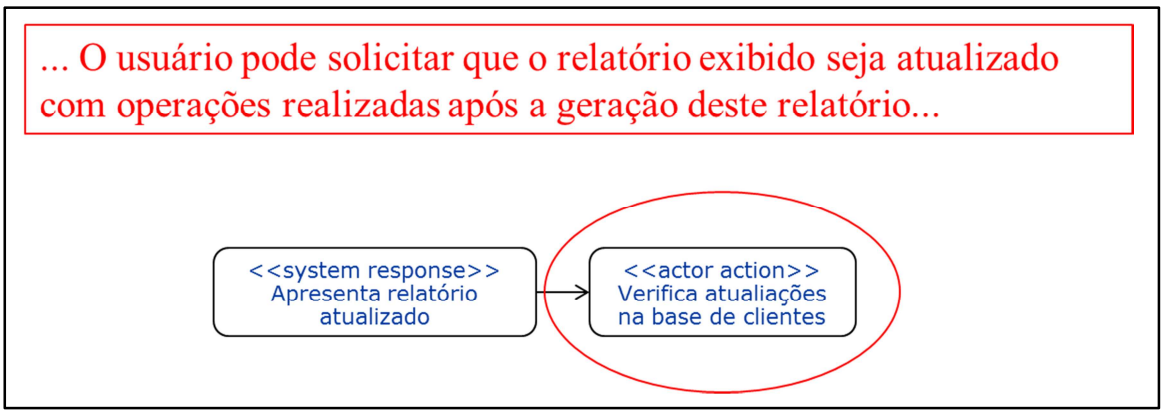

Figura 5.11. Exemplo de defeito em um diagrama de atividades, relacionado ao item de avaliação 34 do checklist A.

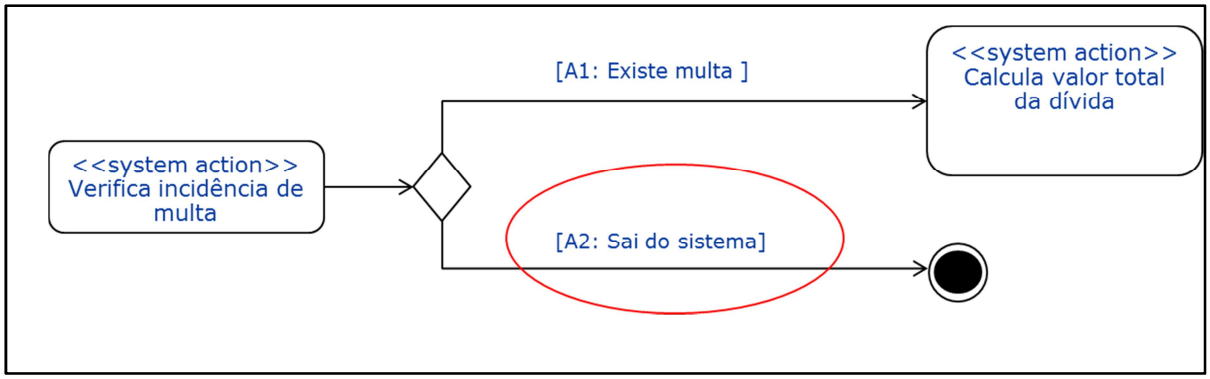

Figura 5.123. Exemplo de defeito em um diagrama de atividades, relacionado ao item de avaliação 33 do checklist A.

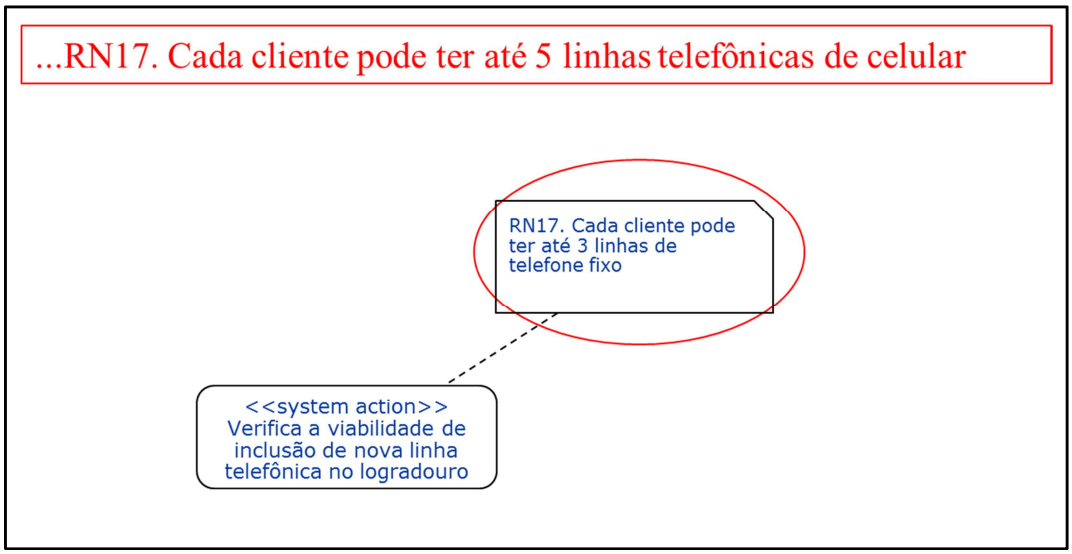

Figura 5.13. Exemplo de inconsistência no diagrama de atividades, detectável através do item de avaliação 37 do Checklist A.

5.5.3 Checklist B

Durante a elaboração da técnica de inspeção, além dos 39 itens de avaliação apresentados na seção anterior, também foram identificados 14 itens que podem envolver a análise da descrição textual dos requisitos, mas não necessariamente este será o fator decisivo para a detecção de uma discrepância nos diagramas de atividades inspecionados.

O Checklist B é composto de 14 itens de avaliação, sendo que a maioria destes itens orientam o inspetor a verificar a consistência interna do diagrama de atividades, categorizadas como verificação intra-diagrama, bem como orientam a detecção de defeitos entre diagramas, categorizadas como verificação Inter diagramas, conforme citado em (Tanriöver e Bilgen, 2007).

A verificação intra-diagrama representa uma nova oportunidade de identificação de discrepâncias na descrição textual dos requisitos, na medida em que defeitos eventualmente ignorados na inspeção prévia da descrição textual dos requisitos tendem a não ser também capturados através da aplicação do Checklist A, que considera esta descrição correta (oráculo). A elaboração de itens de verificação intra-diagrama e inter-diagramas baseia-se no entendimento de que, assim como a elaboração de diagramas de atividades pode contribuir para uma melhor compreensão dos requisitos descritos textualmente, a detecção de defeitos nesta descrição textual também poderia ser facilitada através do exame visual de diagramas de atividades.

Importante observar que, ocasionalmente, um inspetor pode se basear no seu conhecimento do domínio para identificar discrepâncias em qualquer um dos 14 itens de avaliação deste Checklist, bem como também no Checklist A. Entretanto, destacamos que os itens 9, 10 e 11 do Checklist B dependem necessariamente deste conhecimento, pois orientam o inspetor a verificar a viabilidade de fluxos, condições e recebimentos de sinal. Além disto, três itens do Checklist B estão relacionados à rastreabilidade entre atividades: dois itens verificam o uso adequado do recurso sintático de chamada de atividade do Diagrama de Atividades, enquanto um terceiro item se refere à rastreabilidade entre dois diagramas que representam a mesma atividade em níveis de detalhamento distintos.

Para ilustrar a aplicabilidade dos itens de avaliação do Checklist B, a Figura 5.14 exemplifica uma situação em que um trecho dos requisitos está mal descrito (é o operador quem deveria informar os dados da operação de venda, e não o gerente), mas o defeito foi negligenciado na inspeção da descrição textual dos requisitos. Deste modo, o defeito foi replicado para o diagrama de atividades (ação “Informa os dados da operação de venda de título” ligada à raia “Gerente”), e também não foi capturado pelo Checklist A.

Tabela 5.19. Itens de avaliação do Checklist B.

#	Item de Avaliação	Resposta
1	**As ações e condições da atividade estão consistentes entre si?**	( ) Sim ( ) Não ( ) N.A.
2	**Os fluxos da atividade estão consistentes entre si?**	( ) Sim ( ) Não ( ) N.A.
3	Existem ações sendo executadas concorrentemente, mas que são executadas sequencialmente em outras partes do modelo?	( ) Sim ( ) Não ( ) N.A.
4	Existem ações semelhantes/ idênticas sendo executadas por raias ou sub-raias distintas?	( ) Sim ( ) Não ( ) N.A.

5	Os critérios das regiões de expansão da atividade e dos nós de loop estão consistentes entre si?	( ) Sim ( ) Não ( ) N.A.
6	A manipulação dos objetos está consistente ao longo da atividade?	( ) Sim ( ) Não ( ) N.A.
7	As propriedades dos objetos estão sendo respeitadas ao longo da atividade?	( ) Sim ( ) Não ( ) N.A.
8	As relações de envio x recebimento de sinal da atividade estão representadas com clareza?	( ) Sim ( ) Não ( ) N.A.
9	De acordo com o seu conhecimento do domínio, todo fluxo em paralelo da atividade é viável?	( ) Sim ( ) Não ( ) N.A.
10	De acordo com o seu conhecimento do domínio, todas as condições da Atividade são viáveis?	( ) Sim ( ) Não ( ) N.A.
11	De acordo com o seu conhecimento do domínio, todos os recebimentos de sinal da atividade são viáveis ?	( ) Sim ( ) Não ( ) N.A.
12	Alguma ação corresponde à outra atividade do projeto, mas esta referência não está explícita no Diagrama de Atividades?	( ) Sim ( ) Não ( ) N.A.
13	É possível identificar cada fluxo da atividade em outro Diagrama de Atividades do projeto representando a mesma atividade num nível maior de detalhamento?	( ) Sim ( ) Não ( ) N.A.
14	Todas as outras atividades referenciadas nas ações do Diagrama de Atividades existem no projeto?	( ) Sim ( ) Não ( ) N.A.

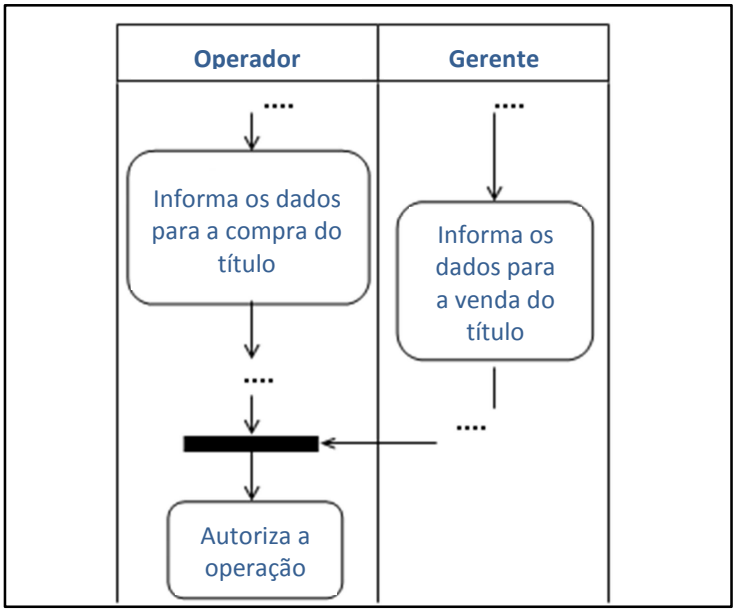

Figura 5.14. Exemplo de inconsistência no diagrama de atividades.

O inspetor que aplicasse o Checklist B, orientado pela questão 4 (“Existem ações semelhantes/ idênticas sendo executadas por raias ou sub-raias distintas?”), poderia suspeitar deste defeito, pois operações semelhantes naquele contexto (compra) enquanto o operador informa os dados da compra e confirma todas as operações do dia, o gerente informa os dados das operações de venda. Caberá,

posteriormente, ao analista de requisitos confirmar se esta discrepância é de fato um defeito ou apenas um falso positivo.

Outro exemplo de inconsistência no diagrama de atividades pode ser visualizado na Figura 5., onde são apresentados dois fluxos distintos (que poderiam pertencer a dois diagramas de atividades distintos), cada um contendo uma primeira ação e a condição de guarda subsequente idênticas. Entretanto, a ação posterior de um dos fluxos (aplicar desconto) contraria o especificado no outro fluxo, que não aplica desconto nenhum. Já a figura 5.17 apresenta um fato incorreto, que depende do conhecimento do domínio do inspetor para ser identificado, supondo-se que a condição está em conformidade com a descrição textual dos requisitos. Se o inspetor soubesse que não existe temperatura negativa na unidade de medição Kelvin, ele poderia assinalar um fato incorreto.

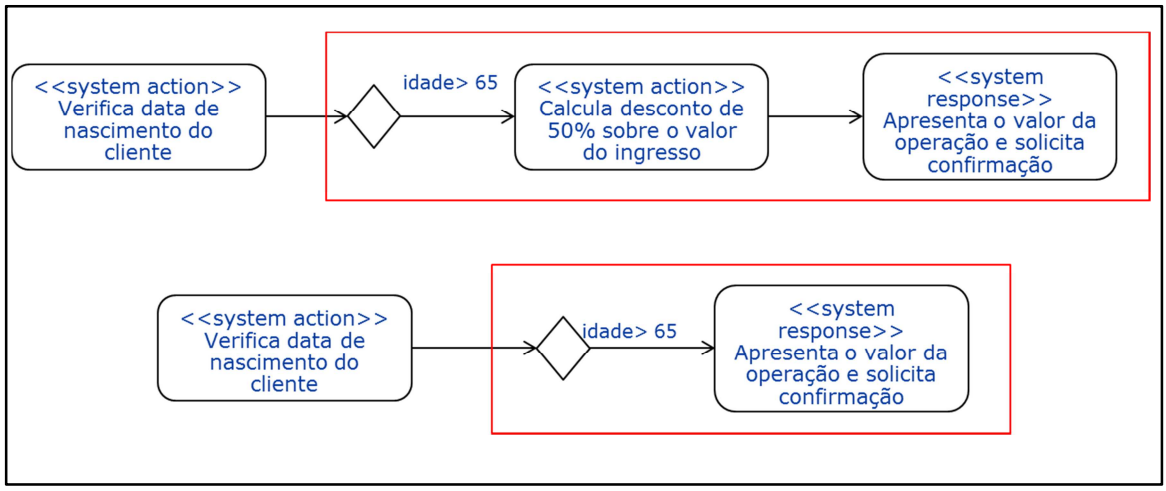

Figura 5.16. Outro exemplo de inconsistência no diagrama de atividades.

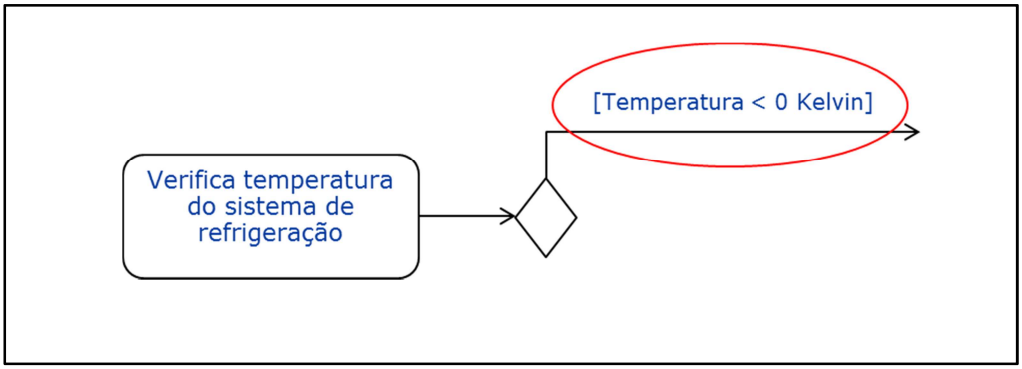

Figura 5.17. Exemplo de defeito identificado através do conhecimento do domínio pelo inspetor.

5.5.4 Novo Questionário de Caracterização da Aplicação

O novo Questionário de Caracterização da Aplicação é formado por quatro questões que buscam capturar determinados aspectos da inspeção para viabilizar a configuração dos *checklists* da técnica. No primeiro item do questionário, assim como na primeira versão da técnica, o questionário solicita que o desenvolvedor informe quais são os recursos sintáticos do diagrama de atividades que estão previstos na técnica de modelagem utilizada na construção dos modelos. Entretanto, devido às modificações realizadas na primeira versão da técnica para ActCheck, uma das opções foi descartada, os nós de fim de fluxo. A marcação das opções deste item influenciará na configuração do Checklist A e do Checklist B

O segundo item do questionário solicita que o desenvolvedor responda se existe mais de um diagrama de atividades na especificação de requisitos. Em caso positivo, o terceiro item solicita ao desenvolvedor informar se a técnica de modelagem prevê que dois ou mais diagramas de atividades possam representar uma mesma atividade em níveis de detalhamento distintos, semelhante ao outro item do questionário na primeira versão da técnica. Um quarto item solicita que o desenvolvedor informe se algum dos inspetores ou se o único inspetor possui conhecimento do domínio do problema. Deste modo, os itens 2, 3 e 4 do novo questionário influenciarão na configuração do Checklist B. Este questionário pode ser visualizado no Apêndice C desta dissertação.

5.5.5 Tabelas de Configuração dos *checklists*

Para orientar o planejador da inspeção a configurar o Checklist A e o Checklist B, foram elaboradas duas tabelas listando todos os itens de avaliação de cada um dos checklists. Entretanto, estas tabelas não devem ser utilizadas pelos inspetores, agregando a cada item de avaliação as seguintes informações:

- Item (itens) do Questionário de caracterização da aplicação, que é a fonte de informação para configurar os checklists de uma inspeção;
- Resposta esperada para cada item de avaliação (Sim / Não);
- Casos Discrepantes relacionados, base para a elaboração de cada item de avaliação do *checklist*, podendo contribuir para o treinamento da técnica;
- Categorias de Defeito Relacionadas, sendo tipicamente apenas uma categoria, mas no caso de inconsistências, fatos incorretos e

ambiguidades, alguns itens de avaliação podem corresponder a defeitos de mais de uma destas categorias.

A Tabela 5.20 apresenta a Tabela de configuração do Checklist B. Importante observar que, quando há um traço (“-“) no campo “Item do Questionário”, significa que esta questão é básica, devendo aparecer em qualquer versão configurada do Checklist B. Esta tabela e a tabela de configuração do Checklist A podem ser visualizadas no Apêndice C desta dissertação.

Tabela 5.20. Tabela de configuração do Checklist B

#	Item de Avaliação	Resposta Esperada	Casos Discrepantes Relacionados	Classes de Defeito Relacionadas	Item do Questionário de Carac. da Aplicação
1	As ações e condições da atividade estão consistentes entre si?	Sim	A10, D07	Inconsistência	-
2	Os fluxos da atividade estão consistentes entre si?	Sim	A11, B04	Inconsistência	-
3	Existem ações sendo executadas concorrentemente, mas que são executadas sequencialmente em outras partes do modelo?	Não	A11, C11	Inconsistência	1) II
4	Existem ações semelhantes/ idênticas sendo executadas por raias ou sub-raias distintas?	Não	G02	Inconsistência	1) III
5	Os critérios das regiões de expansão da atividade e dos nós de loop estão consistentes entre si?	Sim	E10	Inconsistência	1) VII
6	A manipulação dos objetos está consistente ao longo da atividade?	Sim	F09, F10	Inconsistência	1) IV
7	As propriedades dos objetos estão sendo respeitadas ao longo da atividade?	Sim	F08	Inconsistência	1) IV
8	As relações de envio x recebimento de sinal da atividade estão representadas com clareza?	Sim	A13	Ambiguidade	1) VIII
9	De acordo com o seu conhecimento do domínio, todo fluxo em paralelo da atividade é viável?	Sim	C10	Fato Incorreto, Inconsistência	1) II e 4) Sim
10	De acordo com o seu conhecimento do domínio, todas as condições da Atividade são viáveis?	Sim	A09, D03	Fato Incorreto, Inconsistência	4) Sim
11	De acordo com o seu conhecimento do domínio, todos os recebimentos de sinal da atividade são viáveis ?	Sim	A08	Fato Incorreto, Inconsistência	1) VIII e 4) Sim
12	Alguma ação corresponde a outra atividade do projeto, mas esta referência	Não	H01	Omissão	1) IX e 2)Sim

	não está explícita no Diagrama de Atividades?				
13	É possível identificar cada fluxo da atividade em outro Diagrama de Atividades do projeto representando a mesma atividade num nível maior de detalhamento?	Sim	H02	Omissão	1) IX e 2)Sim
14	Todas as outras atividades referenciadas nas ações do Diagrama de Atividades existem no projeto?	Sim	H03	Informação Estranha	2) Sim e 3) Sim

5.5.6 Relatório de Discrepâncias

O modelo do relatório de discrepâncias de ActCheck considera a aplicação de inspeções individuais e orienta o inspetor a descrever e a classificar livremente as discrepâncias encontradas, embora seja necessário informar referências da documentação inspecionada sobre a localização da discrepância e quantas vezes a discrepância se repete (se for o caso). O relatório de discrepâncias também solicita que o inspetor informe quanto tempo levou para realizar a inspeção. Este modelo de relatório de discrepâncias pode ser visualizado no Apêndice C desta dissertação.

5.6 Considerações Finais do Capítulo

Este capítulo apresentou uma técnica de inspeção configurável para apoiar a detecção de defeitos nos diagramas de atividades de especificações de requisitos. Foi apresentado o conceito de caso discrepante, que serviu de base para a elaboração das duas versões da técnica, descritas neste capítulo.

Após uma breve apresentação da versão inicial da técnica de inspeção, são mostrados os resultados de uma prova de conceito conduzida sobre a primeira versão inicial da técnica proposta, que apresentou resultados positivos no que diz respeito à identificação de defeitos e cuja análise contribuiu para a reestruturação de técnica na sua configuração atual.

ActCheck, a segunda versão da técnica de inspeção, foi apresentada com mais detalhes, sendo descritos os artefatos que compõe a técnica: dois *checklists* que compõem a técnica, o questionário de caracterização de aplicação, as tabelas de configuração dos *checklists* e o modelo de relatório de discrepâncias. Também foram apresentados diversos exemplos de aplicação dos itens de avaliação dos *checklists* na identificação de defeitos.

Da mesma forma que a primeira versão da técnica de inspeção, ActCheck também precisa ser avaliada, porém desta vez através de um processo experimental, de modo a buscar evidências estatisticamente significantes referentes à sua viabilidade e, consequentemente, à sua eficácia e sua eficiência.

CAPÍTULO 6 - AVALIAÇÃO DA VIABILIDADE DE ACTCHECK

Neste capítulo, é apresentado o estudo experimental conduzido para avaliar a viabilidade da técnica para inspeção do diagrama de atividades. São introduzidos os principais aspectos de seu planejamento, são apresentadas as análises realizadas sobre os resultados identificados e as ameaças à validade do estudo. Também é apresentada brevemente uma nova versão de ActCheck que foi desenvolvida a partir dos resultados e conclusões obtidas neste estudo.

6.1 Introdução

Entre novembro e dezembro de 2010 foi conduzido um estudo experimental no âmbito da COPPE *(in vitro)* para avaliar a técnica de inspeção apresentada no Capítulo 5, ActCheck. O objetivo geral do estudo experimental foi avaliar os efeitos da aplicação de ActCheck em parte de um projeto de uma aplicação real, no caso o módulo MFI, do sistema SIGIC, desenvolvido pelo Grupo de Engenharia de Software Experimental da COPPE/UFRJ para a Fundação COPPETEC. Além de avaliar a viabilidade da técnica, o estudo também envolveu a comparação dos resultados obtidos em inspeções realizadas com ActCheck e inspeções *ad hoc.* Como objetivos específicos do estudo, foram definidos os seguintes:

- Analisar: os efeitos da aplicação de ActCheck
- Com o propósito de: caracterizar
- Em relação à: sua viabilidade (opinião dos participantes, eficácia e eficiência na identificação de defeitos)
- Do ponto de vista de: estudantes de mestrado e doutorado em Engenharia de Software
- No contexto de: estudantes de pós-graduação da COPPE aplicando a técnica em artefatos de um projeto de uma aplicação real

O propósito de caracterizar a viabilidade de ActCheck visa corresponder ao primeiro estudo primário previsto por Shull et al. (2001) ao longo do processo para introdução de tecnologias de software na indústria, introduzido no Capítulo 1 desta

dissertação. Considera-se como a eficiência de uma inspeção a razão entre a quantidade de defeitos detectados e o tempo dedicado à inspeção (defeitos por hora). Considera-se como a eficácia de uma inspeção a taxa de defeitos detectados numa inspeção em relação a quantidade total de defeitos conhecidos para o artefato inspecionado. O estudo buscou responder às seguintes questões de pesquisa:

1. A aplicação de ActCheck auxilia na detecção de defeitos?
2. O tempo utilizado na inspeção com ActCheck é bem aplicado?
3. Uma inspeção aplicando ActCheck é mais eficaz do que uma inspeção ad hoc?
4. ActCheck estimula os inspetores à sua aplicação?

Para responder às três primeiras questões, foram extraídas as seguintes métricas básicas: quantidade de defeitos (conhecidos e detectados), quantidade de discrepâncias e tempo dedicado à inspeção. Para responder à quarta questão, também foi considerada a análise da opinião dos participantes do estudo sobre ActCheck. Deste modo, considera-se que o estudo é quantitativo, por coletar resultados para posterior análise da eficiência e da eficácia da técnica, mas também qualitativo, por abordar a opinião dos participantes.

A documentação do módulo MFI utilizada nas inspeções consistiu na especificação de requisitos previamente revisada do módulo financeiro do sistema MFI e em um conjunto de diagramas de atividades que descrevem quatro casos de uso que buscam atender a esta especificação. Entretanto, é importante ressaltar que a descrição dos requisitos foi feita diretamente em diagramas de atividades, que ainda seriam inspecionados, não havendo uma descrição de casos de uso textual previamente revisada (oráculo) para comparação. A descrição dos requisitos a partir de diagrama de atividades foi feita com base na técnica de modelagem proposta por Massollar (2008).

6.2 Planejamento do Estudo Experimental

Esta seção apresenta os aspectos mais relevantes relacionados ao planejamento do estudo. Mais detalhes podem ser verificados no Apêndice D desta dissertação, que apresenta de modo integral o Planejamento do Estudo Experimental. A partir das questões de pesquisa citadas na seção 6.1, foram formuladas as seguintes hipóteses nulas (H01 e H02) e suas hipóteses alternativas correspondentes (HA1 e HA2):

- H01: Não há diferença entre a eficiência de inspeções de diagramas de atividades que aplicam ActCheck e a eficiência de inspeções ad hoc.
- HA1: A eficiência de inspeções de diagramas de atividades que aplicam ActCheck é maior que a eficiência de inspeções ad hoc.
- H02: Não há diferença entre a eficácia de inspeções de diagramas de atividades que aplicam ActCheck e a eficácia de inspeções ad hoc.
- HA2: A eficácia de inspeções de diagramas de atividades que aplicam ActCheck é maior que a eficácia de inspeções ad hoc

O estudo contou com a participação de oito alunos (um de mestrado e sete de doutorado) do curso de VV&T (Verificação, Validação e Testes) promovido pelo Programa de Engenharia de Sistemas da Computação da COPPE/UFRJ. Cada participante assinou um termo de consentimento para participar deste estudo e preencheu um formulário de caracterização do participante. Os participantes foram organizados em quatro duplas com nível de experiência gradual, conforme o grau de experiência de cada um em conhecimentos relacionados à engenharia de software, especialmente em inspeção de software, em diagrama de atividades e em requisitos, conforme relatado em seus respectivos formulários de caracterização do participante. Importante observar que, em cada dupla, havia um participante que já possuía algum conhecimento do domínio da aplicação.

Para o estudo, foram selecionados quatro casos de uso do MFI (UC006, UC015, UC036 e UC040), sendo que havia dois diagramas de atividades para descrever cada caso de uso: uma versão desenvolvida através da ferramenta de modelagem BOUML (BOUML, 2011), adaptada com um *plug-in* para apoio a modelagem de diagramas de atividades e outra versão desenvolvida através da mesma ferramenta BOUML, sem apoio do *plug-in*. Deste modo, oito diagramas de atividades fizeram parte do estudo.

Foi definido que cada participante do estudo deveria realizar uma inspeção de cada caso e uso, sendo que duas inspeções seriam feitas *ad hoc*, numa primeira rodada, e duas inspeções seriam realizadas aplicando-se ActCheck, numa segunda rodada. Para equilibrar a distribuição dos artefatos entre os grupos de participantes, cada caso de uso foi classificado conforme a sua complexidade em duas categorias (complexidade menor e complexidade maior). Deste modo, os casos de uso UC015 e UC040 foram considerados de maior complexidade e os casos de uso UC006 e UC030 foram considerados de menor complexidade.

A Tabela 6.1 apresenta a distribuição de artefatos realizada ao longo das duas rodadas, sendo que os participantes marcados com um asterisco possuíam

previamente algum conhecimento do domínio do problema. Esta distribuição visou equilibrar a complexidade dos artefatos utilizados em cada grupo e fazer com que cada diagrama de atividades fosse inspecionado duas vezes em cada rodada. Para as duplas B, C e D, os artefatos foram distribuídos de modo que cada participante inspecionasse um caso de uso de maior complexidade e outro de menor complexidade a cada rodada. Já no grupo A, cada participante inspecionou dois artefatos de menor complexidade numa rodada e dois artefatos de maior complexidade na outra rodada.

Tabela 6.1. Distribuição dos artefatos entre os participantes por rodada.

Dupla	Nível de Experiência	Participante	Primeira Rodada (*ad hoc*)		Segunda Rodada (ActCheck)	
			sem *plug-in*	com *plug-in*	sem *plug-in*	com *plug-in*
A	Alto	P1	UC015	UC040	UC006	UC036
		P2*	UC006	UC036	UC015	UC040
B	Médio-Alto	P3	UC036	UC040	UC006	UC015
		P4*	UC006	UC015	UC036	UC040
C	Médio-Baixo	P5	UC036	UC006	UC040	UC015
		P6*	UC040	UC015	UC036	UC006
D	Baixo	P7*	UC015	UC006	UC040	UC036
		P8	UC040	UC036	UC015	UC006

Antes da execução de cada rodada, estavam previstos treinamentos específicos para preparação dos participantes. A preparação para a primeira rodada contava com uma exposição de 1 hora 30 minutos sobre o diagrama de atividades e a técnica utilizada nos modelos para descrição de casos de uso (Massollar, 2008). Também deveriam ser apresentadas as categorias de defeitos que deveriam ser utilizadas nas inspeções: omissão, fato incorreto, inconsistência, ambiguidade e informação estranha. Já antes da execução da segunda rodada, os participantes assistiria a uma exposição de 1 hora e 30 minutos sobre ActCheck. Neste treinamento, foram apresentados os artefatos da técnica e exemplos de defeitos para cada item de avaliação presente nos *checklists* configurados para o estudo.

Em cada rodada os participantes inspecionariam os diagramas, categorizando e relatando cada discrepância identificada em um relatório de discrepâncias conforme o modelo apresentado no Apêndice C. Entretanto, na segunda rodada, em que estava prevista a aplicação de ActCheck o relatório de discrepâncias recebeu mais uma coluna para que também fosse informado qual o item do *checklist* que havia colaborado para esta identificação de determinada discrepância. No caso da segunda

rodada, é importante observar que os Checklists A e B foram previamente configurados pelos pesquisadores antes de serem encaminhados aos participantes, conforme as características da técnica de modelagem utilizada na construção dos diagramas de atividades do projeto MFI.

6.3 Execução do Estudo Experimental

Os treinamentos foram ministrados dentro dos prazos previamente definidos, abordando o conteúdo previsto e contando com a presença de todos os participantes. Após o treinamento, cada participante recebeu por e-mail seu pacote contendo a especificação de requisitos do MFI e os dois diagramas de atividades correspondentes a dois casos de uso do sistema, além do relatório de discrepâncias a ser preenchido. No caso da segunda rodada (ActCheck), cada participante também recebeu os dois *checklists* configurados. Todos os pacotes foram encaminhados e recebidos com sucesso. Ao fim de cada rodada, todos os participantes encaminharam seus relatos de discrepâncias conforme previsto. Deste modo, nenhum dos participantes foi descartado do estudo.

Após os participantes encaminharem os artefatos preenchidos correspondentes à segunda rodada, houve uma entrevista conduzida pelos pesquisadores contando com a presença de todos os participantes, onde foram coletadas as percepções destes sobre o estudo em geral e sobre os aspectos negativos e positivos de ActCheck. Propostas para promover melhoria da técnica também foram coletadas

As discrepâncias relatadas pelos participantes em cada rodada foram analisadas por dois pesquisadores, extraindo-se os defeitos e descartando-se os falsos positivos. Quando necessário, os pesquisadores também alteraram a categorização dos defeitos e indicaram aqueles defeitos que foram detectados mais de uma vez numa mesma rodada e/ou entre rodadas. Para tal, primeiramente, para cada rodada, cada pesquisador conduziu sua análise individual das discrepâncias. Em seguida, através de uma reunião, os dois pesquisadores confrontaram suas análises até chegar a um consenso sobre cada item divergente. Um terceiro pesquisador acompanhou e revisou todo o processo. Com as discrepâncias devidamente tratadas, a listagem de cada rodada foi listada em planilhas Excel, contendo colunas para auxiliar o tratamento dos dados em arranjos mais específicos. Um dos arranjos, por exemplo, foi utilizado para alimentar a ferramenta estatística JMP4 (SAS, 2011), a fim de verificar as hipóteses formuladas no planejamento do estudo experimental. A Tabela 6.2 apresenta um exemplo de como os dados foram agrupados (por participante), para análise na ferramenta, onde a terminação “S” no nome do caso e uso significa “sem *plug-in*” e “C” significa “com *plug-in*”.

Tabela 6.2. Tempo dedicado à inspeção, discrepâncias e defeitos detectados por participante para cada artefato inspecionado.

Grupo	Participante	Técnica	Nome	Tempo (minutos)	Discrepâncias	Defeitos
A	P1	Adhoc	UC015S	40	6	5
A	P1	Adhoc	UC040C	35	5	3
A	P2	Adhoc	UC006S	22	5	3
A	P2	Adhoc	UC036C	24	8	6
B	P3	Adhoc	UC036S	90	10	7
B	P3	Adhoc	UC040C	105	10	10
B	P4	Adhoc	UC006S	25	6	4
B	P4	Adhoc	UC015C	74	20	5
C	P5	Adhoc	UC040S	42	13	10
C	P5	Adhoc	UC015C	34	10	5
C	P6	Adhoc	UC036S	25	5	3
C	P6	Adhoc	UC006C	25	2	0
D	P7	Adhoc	UC015S	45	15	9
D	P7	Adhoc	UC006C	10	2	2
D	P8	Adhoc	UC040S	32	11	11
D	P8	Adhoc	UC036C	26	4	1
A	P1	ActCheck	UC006S	45	2	2
A	P1	ActCheck	UC036C	55	1	1
A	P2	ActCheck	UC015S	55	6	6
A	P2	ActCheck	UC040C	30	14	14
B	P3	ActCheck	UC006S	50	5	3
B	P3	ActCheck	UC015C	200	10	4
B	P4	ActCheck	UC036S	93	6	6
B	P4	ActCheck	UC040C	47	7	7
C	P5	ActCheck	UC036S	62	10	8
C	P5	ActCheck	UC006C	52	3	2
C	P6	ActCheck	UC040S	40	11	8
C	P6	ActCheck	UC015C	50	16	12
D	P7	ActCheck	UC040S	60	16	16
D	P7	ActCheck	UC036C	35	3	3
D	P8	ActCheck	UC015S	39	5	2
D	P8	ActCheck	UC006C	31	6	3

6.4 Análise dos Resultados

Esta seção apresentará a análise realizada sobre os dados coletados ao longo da execução do estudo. Primeiramente serão apresentados os principais aspectos observados na análise quantitativa dos dados coletados, em seguida, são apresentadas considerações observadas através da análise qualitativa das informações obtidas na entrevista com o grupo.

6.4.1 Análise Quantitativa

A fim de colocar à prova as hipóteses do estudo, os resultados obtidos foram submetidos a diversas análises apoiadas pela ferramenta estatística JMP, de onde se buscou avaliar também outros aspectos relacionados às inspeções, como o comportamento da variável tempo e da variável número de defeitos.

A relevância estatística dos resultados coletados foi analisada pela ferramenta JMP4, considerando-se o teste da Distribuição de Student: média da população e diferenças entre médias, com alfa= 10%, devido a limitada quantidade de participantes do estudo. A Figura 6.1 representa a distribuição da eficiência, já com *outliers* removidos, a fim de testar a hipótese alternativa HA01, referente à eficiência das inspeções. Do mesmo modo, a Figura 6.2 representa a distribuição dos dados por participante e tipo de inspeção realizada a fim de testar a hipótese alternativa HA02, que diz respeito às eficácias das inspeções.

Cada ponto da Figura 6.1 representa a eficiência (quantidade de defeitos por minuto) de cada inspeção individual de cada rodada, enquanto cada ponto da Figura 6.2 representa a sua eficácia (taxa de defeitos detectados em relação a quantidade de total de defeitos conhecidos de cada diagrama). Em ambas as figuras, o canto direito da figura ilustra a falta de significância estatística uma vez que, quando a diferença entre as médias da população não é suficiente, os círculos possuem alguma interseção.

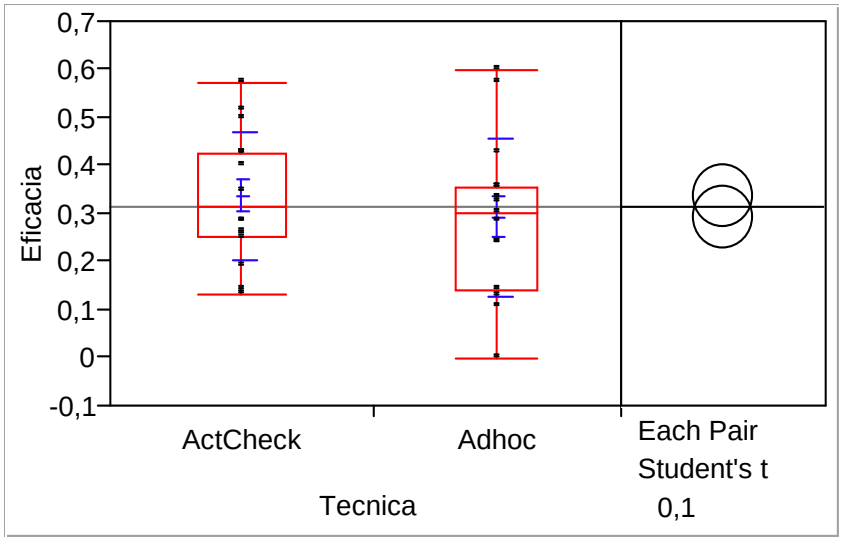

Figura 6.1. Distribuição da eficiência dos inspetores em cada rodada e representação correspondente do teste t de Student da amostra com alfa= 10%.

Apesar das hipóteses alternativas não terem sido confirmadas, observou-se que existe diferença estatisticamente significativa entre o tempo dedicado às inspeções *ad*

hoc e as inspeções com ActCheck, sendo que a técnica avaliada, na média, exige mais tempo do inspetor do que a inspeção sem técnica, conforme pode ser observado na Figura 6.3, onde a dimensão do tempo apresenta-se em minutos e cada ponto do gráfico corresponde ao tempo que um participante levou para realizar suas inspeções em uma rodada. Em contrapartida não é possível afirmar que inspeções com ActCheck propiciam a identificação de mais defeitos do que inspeções *ad hoc*, conforme pode ser observado na figura 6.4. Análises estatísticas adicionais foram feitas comparando os resultados entre os grupos e entre os participantes com e sem conhecimento prévio do domínio do problema. Entretanto, não foi possível afirmar que ActCheck tenha promovido nenhuma mudança significativa na eficácia ou na eficiência em algum destes agrupamentos.

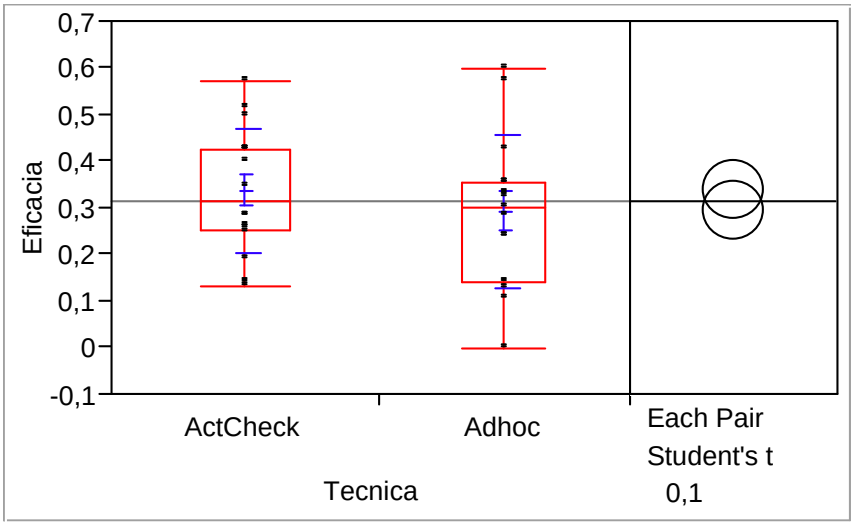

Figura 6.2. Distribuição da eficácia dos inspetores em cada rodada e representação correspondente do teste t de Student da amostra com alfa= 10%.

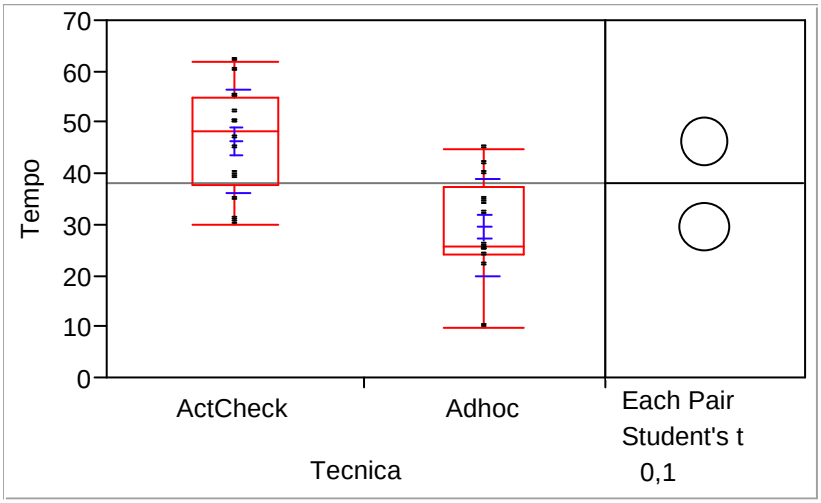

Figura 6.3. Distribuição do tempo dedicado às inspeções por inspetor em cada rodada e representação correspondente do teste de Student da amostra com alfa= 10%.

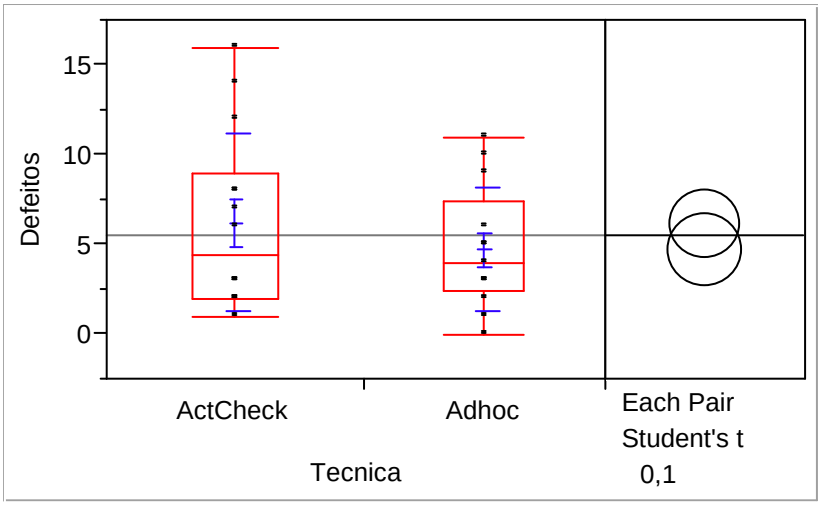

Figura 6.4. Distribuição dos defeitos detectados pelos inspetores por rodada e representação correspondente do teste de Student da amostra com alfa= 10%.

Para buscar compreender as causas das hipóteses alternativas não terem sido confirmadas, foi conduzida uma investigação mais detalhada sobre os dados coletados, a fim de identificar os possíveis pontos para promover a melhoria da técnica. A Tabela 6.3 apresenta resumidamente os resultados coletados em cada rodada:

Tabela 6.3. Resumo dos dados coletados nas duas rodadas de inspeção do estudo.

Item observado	Rodada 1 (ad hoc)	Rodada 2 (ActCheck)
Número de Discrepâncias detectadas	132	121
Tempo total dedicado às inspeções (minutos)	654	944
Total de defeitos detectados	84	97
Defeitos/hora	7,71	6,17
Total de defeitos distintos	75	88
Taxa de Defeitos em relação aos defeitos conhecidos	60%	70%

Analisando os dados da Tabela 6.3, percebe-se que a aplicação de ActCheck reduziu significativamente a quantidade de falsos positivos das inspeções (redução de 50%), embora pouco tenha contribuído para a detecção de mais defeitos (aumento de 15%). Deste modo, ocorreu com ActCheck um ligeiro aumento da cobertura de defeitos detectados (de 60% para 70%) em relação aos defeitos distintos detectados em ambas as rodadas. Em contrapartida, o tempo dedicado à inspeção de ActCheck foi bem maior (aumento de 44%), levando também a uma relevante queda na eficiência da técnica avaliada (20%), quando esta é comparada com a eficiência das inspeções ad hoc (em defeitos/hora).

Analisando a quantidade de defeitos detectados em comum nas duas rodadas, chama a atenção o fato de que menos de um terço dos defeitos detectados nas inspeções *ad hoc* também foram detectados nas inspeções através ActCheck, como pode ser visualizado na Figura 6.4. Ou seja, apesar de ActCheck ter detectado mais defeitos, não foi capaz de detectar mais de dois terços dos defeitos detectados sem auxílio de técnicas.

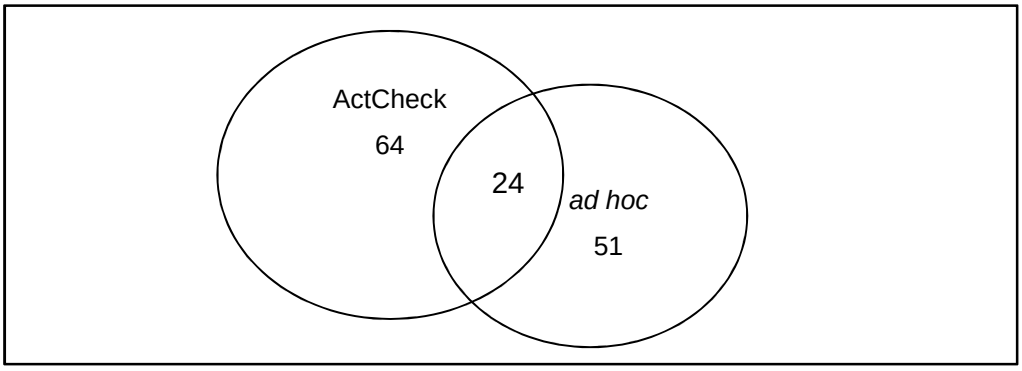

Figura 6.5. Quantidade de defeitos distintos detectados somente na segunda rodada (ActCheck), em ambas as rodadas e somente na primeira rodada (*ad hoc*).

Para identificar as causas desta ineficácia, inicialmente foram mapeados os defeitos distintos detectados em cada rodada por categoria, como pode ser observado na Tabela 6.4. Observando-se a quantidade de defeitos distribuída por categorias de defeito, percebe-se que a quantidade de omissões, fatos incorretos e inconsistências detectadas foram semelhantes, embora poucos foram os defeitos destas categorias que foram detectados nas duas rodadas.

Tabela 6.4- Distribuição dos defeitos detectados nas inspeções por categoria de defeito.

Categoria de Defeito	ad hoc	ActCheck	ambas
Omissão	46	45	20
Fato Incorreto	14	10	3
Inconsistência	10	8	1
Ambiguidade	4	9	0
Informação Estranha	1	16	0
Total	**75**	**88**	**24**

Chama a atenção também o significativo aumento das ambiguidades e de informações estranhas detectadas com ActCheck e que, além disto, nenhum dos defeitos detectados destas duas categorias foi identificado nas duas rodadas. Para entender melhor este comportamento, buscou-se identificar se cada um dos 51

defeitos detectados somente nas inspeções ad hoc poderia ter sido detectado com o apoio de algum item de avaliação do Checklist A ou do Checklist B. Após esta análise, entretanto, detectou-se que apenas 11 dos 51 detectados somente na primeira rodada (*ad hoc*) não estavam contemplados pelos itens de avaliação dos checklists da técnica, como pode ser visualizado na Tabela 6.5. Destes 11 defeitos não contemplados pelo checklist, a maior parte deles estava relacionada com a ausência de determinadas verificações referentes ao uso de regras de negócio para descrição de casos de uso.

Tabela 6.5- Distribuição dos defeitos detectados nas inspeções com ActCheck e defeitos detectados somente nas inspeções *ad hoc* por item de avaliação dos *checklists* de ActCheck.

Checklist/ Grupo	Item de Avaliação	Defeitos detectados ActCheck	Defeitos detectados somente ad hoc	% Cobertura
A- Ações e Condições	A-01	7	1	88%
	A-03	2	3	40%
	A-06	9	2	82%
	A-08	4	0	100%
A- Fluxo de Controle	A-09	18	13	58%
	A-10	3	0	100%
	A-11	1	0	100%
	A-12	1	12	8%
	A-15	1	1	50%
A- Objetos	A-16	0	0	-
	A-17	0	0	-
	A-18	0	0	-
	A-19	0	0	-
	A-20	1	0	100%
	A-21	0	0	-
	A-22	0	0	-
A- Casos de Uso	A-26	4	1	80%
	A-27	0	0	-
	A-28	1	0	100%
	A-29	1	0	100%
	A-30	3	0	100%
	A-31	3	0	100%
	A-32	2	0	100%
	A-33	0	0	-
	A-34	2	1	67%
	A-35	4	2	67%
	A-36	2	1	67%
	A-37	1	0	100%
	A-38	1	0	100%
	A-39	7	0	100%
Checklist B	B-01	3	2	60%
	B-02	2	1	67%

	B-06	0	0	-
	B-07	0	0	-
	B-10	2	0	100%
	B-12	0	0	-
	B-14	0	0	-
Sem item de avaliação		**3**	**11**	**21%**
Total		**88**	**51**	**63%**

A constatação de que a maioria os defeitos poderia ter sido detectada através dos itens de avaliação dos *checklists*, direcionou a análise para a qualidade destes itens. Na Tabela 6.5 podemos observar que, somente o grupo “Fluxo de Controle” do Checklist A foi responsável por quase metade dos defeitos não detectados, embora os demais itens de avaliação de outros grupos também não tenham apresentado a cobertura mínima esperada, ou seja, detectar os mesmos defeitos identificados nas inspeções ad hoc.

Importante observar que, como os casos de uso inspecionados não possuíam nós de objeto (embora fossem previstos pela técnica de modelagem), os itens de avaliação do Checklist A relacionados a este tipo de constructo fizeram parte da inspeção. De um modo geral, com os dados coletados não foi possível confirmar nem refutar as hipóteses alternativas do estudo, ou seja, não é possível afirmar que ActCheck seja mais eficiente ou mesmo mais eficaz do que inspeções *ad hoc*.

6.4.2 Análise Qualitativa

Ao longo da entrevista conduzida pelos pesquisadores no fim do estudo, os participantes foram ouvidos a respeito de suas impressões sobre o estudo e sobre as inspeções que realizaram. Os pesquisadores incentivaram os participantes a emitirem suas opiniões, evitando direcionar os temas. Enquanto os participantes opinavam, os pesquisadores realizavam anotações e tiravam dúvidas para esclarecer as observações feitas pelos participantes, quando necessário. A seguir, serão apresentadas as opiniões coletadas e validadas com os oito participantes ao longo da entrevista.

Avaliação dos treinamentos

Os participantes alegaram que, apesar dos treinamentos terem sido importantes para a aquisição de conhecimento para realização das inspeções, estes foram insuficientes, por abordarem um grande conteúdo de tópicos em pouco tempo.

De fato, conforme apresentado no Capítulo 2 desta dissertação, o diagrama de atividades possui uma grande quantidade de componentes e uma vasta aplicabilidade.

Apesar dos treinamentos terem focado na descrição de casos de uso a partir de diagramas de atividade, esta abordagem acrescenta novos elementos sintáticos e novas oportunidades de verificação, que também precisaram ser assimiladas pelos participantes em pouco tempo. Além disto, embora ActCheck tenha sido apresentada através de diversos exemplos no segundo treinamento, estes exemplos não estavam encadeados como num processo de inspeção real e o tempo não foi suficiente para abordar adequadamente cada item de avaliação de ActCheck.

Como proposta de melhoria, foi sugerido que os treinamentos fossem ministrados através de situações reais como, por exemplo, o uso de ActCheck poderia ser exemplificado ao longo de uma inspeção em grupo de um projeto real, realizada *on the fly* com os participantes. Outra proposta observada foi a criação de um guia para aplicação da inspeção com ActCheck, o que sugere que os participantes sentiram falta de um apoio mais sistemático ao longo das inspeções.

Avaliação de ActCheck

Os participantes relataram que perceberam um retrabalho muito grande na aplicação dos itens de avaliação de ActCheck. Para eles, a subdivisão dos itens de avaliação por categorias de defeito torna o processo cansativo e repetitivo, pois um mesmo construto teria que ser observado mais de uma vez para relatar categorias de defeito distintas. Conforme pode ser observado nas tabelas de configuração do Apêndice C, especialmente no caso dos grupos Checklist A, os itens de avaliação foram estruturados a fim de buscar primeiramente por omissões, em seguida por fatos incorretos, inconsistências e ambiguidade e, por fim são listados os itens para busca de informações estranhas.

Como aspectos positivos, os participantes relataram que os itens de avaliação descritos nos *checklists* os levaram a detectar defeitos que muito provavelmente não observariam nas inspeções *ad hoc*. Consequentemente, os participantes também relataram que a aplicação de ActCheck contribuiu para que estes internalizassem novas oportunidades de detecção de defeitos em diagramas de atividades. Quando questionados se estariam mais preparados para uma nova inspeção *ad hoc* após aplicarem ActCheck, os participantes concluíram que sim.

As considerações dos participantes foram muito importantes para explicar o porquê de muitos itens detectados nas inspeções *ad hoc* não terem sido detectados nas inspeções com ActCheck. Embora os participantes tenham percebido uma cobertura mais ampla da técnica no que diz respeito à possibilidade de identificar

novos defeitos, diferentes daqueles que habitualmente buscariam em inspeções *ad hoc*, a forma como os *checklists* foram estruturados não incentivou a exploração de seus recursos. Do mesmo modo, a repetição poderia explicar o porquê das inspeções com ActCheck terem sido, na média, significativamente mais demoradas do que as inspeções *ad hoc.*

Como proposta, os participantes da entrevista concluíram que os itens de avaliação precisam ser reorganizados, de modo a otimizar o resultados da aplicação da técnica e aproveitar melhor as oportunidades que ActCheck oferece para detecção de defeitos.

Avaliação da documentação inspecionada

Os participantes também relataram que esperavam que a documentação do projeto MFI fosse mais densa, havendo pouca referência para detecção de defeitos nos diagramas de atividades. Como exemplo, citaram que só havia descrição dos casos de uso no próprio diagrama de atividades. Independente do conhecimento prévio do domínio do problema, os participantes entenderam que possuíam pouco referencial para detectar discrepâncias. Entretanto, estava previsto na técnica de modelagem adotada no projeto MFI uma especificação de requisitos menos densa e a descrição de casos de uso diretamente através de diagramas de atividades.

6.5 Lições Aprendidas

Com a consolidação das análises quantitativa e qualitativa, foram extraídas relevantes lições para evolução da técnica. No entanto, entendemos também que estas lições poderão contribuir na futura elaboração/ evolução de outras técnicas de inspeção.

Primeiramente, destacamos a importância da observação prévia do comportamento dos inspetores, ou seja, buscar identificar como os inspetores buscam defeitos em determinado artefato de software. Atender a esta observação poderia contribuir para que os recursos existentes em ActCheck fossem melhor aproveitados. Neste sentido, ressaltamos que a organização dos itens de um *checklist* em subgrupos sintáticos não necessariamente contribui para melhores resultados. Ou seja, não basta um *checklist* estar *organizado* se o seu uso não é *prático.*

Do mesmo modo, foi observado que práticas como dividir os itens de avaliação em mais de um *checklist* ou mesmo utilizar itens de avaliação excessivamente padronizados e restritos a determinada categoria de defeito podem tornar as inspeções repetitivas e, consequentemente, desestimular os inspetores.

6.6 Ameaças à Validade

Como ameaça a validade deste estudo, destacamos a pequena quantidade de participantes (oito) e, consequentemente, a pequena quantidade de artefatos inspecionados. Além disto, destacamos também a dificuldade de controle do tempo das inspeções, relatado pelos participantes por e-mail nos relatórios das inspeções, embora a precisão deste tipo de variável também seja de difícil controle mesmo com controle presencial em sala de aula.

A forma como as inspeções *ad-hoc* e as inspeções com ActCheck foram distribuídas nas rodadas também pode ser considerada uma ameaça à validade, uma vez que todos as duplas inspetores só realizaram inspeções ActCheck após realizarem inspeções *ad hoc*. Deste modo, não pôde ser visualizado o efeito da fadiga dos inspetores na execução da segunda rodada. Com uma maior quantidade de participantes, seria possível, por exemplo, promover combinações de tipos de inspeção distintas entre os grupos ao longo das rodadas.

Outra ameaça à validade consiste na especificidade do material inspecionado, limitado a um único módulo (financeiro) de um sistema. Devido às características de técnica de modelagem, diversos itens de avaliação de ActCheck não chegaram a ser aplicados. A inexistência de uma lista prévia de defeitos conhecidos também pode ser considerada uma ameaça à validade, afetando diretamente o cálculo da cobertura das inspeções. Foram considerados como defeitos conhecidos apenas aqueles defeitos distintos detectados em ambas as rodadas de inspeção.

É importante ressaltar também que os pesquisadores que avaliaram as discrepâncias encaminhadas pelos participantes pertencem ao grupo de Engenharia de Software Experimental da COPPE, onde foi desenvolvida a técnica de inspeção estudada e também onde foi desenvolvida a documentação do módulo MFI.

6.7 Nova Versão de ActCheck

Com base nos resultados obtidos através do estudo experimental, foi desenvolvida uma nova versão de ActCheck (versão 2), com o objetivo principal de tornar a aplicação da técnica mais atrativa e menos trabalhosa para os inspetores. Nesta nova versão, os *checklists* foram unificados e alguns itens foram reescritos, desmembrados ou unificados. Os itens de avaliação de ActCheck também foram reagrupados, buscando reproduzir um encadeamento de itens mais próximo à análise realizada pelos inspetores. De modo semelhante a uma técnica de leitura, esta nova

versão de ActCheck busca orientar o inspetor sobre o que deve ser observado em cada grupo de itens de avalição.

Para evitar retrabalho, optou-se pela unificação dos *checklists*, pois, apesar do Checklist B possuir bem menos itens e apresentar um foco distinto do Checklist A, a divisão entre dois *checklists* fazia com que o inspetor tivesse que retornar a diversos componentes da atividade previamente verificados. Por exemplo, um item de avaliação baseado no Checklist B, questiona se *"Existe alguma pré ou pós condição local que, mesmo estando correta, é impossível de ser atendida?"* foi colocado imediatamente posterior a outro item que questiona se *"Todas as pré e pós condições locais de cada ação estão descritas corretamente?"*. É importante salientar que, mesmo que uma pré ou pós condição local esteja corretamente descrita sob o ponto de vista dos requisitos, o inspetor pode concluir que ela é impossível de ser alcançada, mediante seu conhecimento do domínio ou mesmo mediante o comportamento observado na atividade. Este checklist único da versão 2 de ActCheck pode ser visualizado no Apêndice E.

Outra mudança significativa em ActCheck 2 foi a eliminação do grupo de itens de avaliação específicos para o caso do diagrama de atividades descrever um caso de uso. Apesar dos 14 itens deste grupo apresentarem uma boa cobertura de detecção de defeitos no estudo experimental, os inspetores precisavam rever cada ação e fluxo da atividade que já haviam sido verificados em 22 itens anteriores. Para evitar retrabalho, estes itens foram redistribuídos entre os itens referentes à ações e os itens referentes ao fluxo de controle.

A reescrita de alguns itens de avaliação teve como premissa tornar mais explícita a condição que leva à discrepância ao invés de focar na categorização do defeito. Por exemplo, a expressão "em conformidade com os requisitos", que surgia diversas vezes na versão anterior de ActCheck, foi substituída por expressões como "está descrita corretamente" e variantes. Com o mesmo objetivo, alguns itens do novo checklist são compostos por mais de uma questão, como, por exemplo "Cada fluxo da atividade permite interpretar claramente a atividade? Existe algum trecho da atividade confuso?".

Na elaboração de novos itens de avaliação de ActCheck 2, foi dado um foco maior nos itens que apresentaram menor cobertura, como foi apresentado na Figura 6.5 e comentado neste Capítulo, especialmente nos itens de avaliação referentes a fluxo de controle. Diversos itens deste grupo foram reescritos e desmembrados em mais de um item, de modo a chamar a atenção do inspetor para determinados caso discrepantes que os inspetores capturaram em inspeções *ad hoc*, mas não

perceberam com o apoio de ActCheck. A verificação do fluxo de controle da atividade foi ampliada e é agora tratada por 20 itens no novo checklist.

6.8 Considerações Finais do Capítulo

Este capítulo apresentou o estudo experimental cuja primeira versão de ActCheck foi submetida para avaliação. Foram descritos os principais aspectos que nortearam seu planejamento e execução, bem como as análises quantitativa e qualitativa realizadas. Apesar dos resultados deste estudo sinalizarem a viabilidade de ActCheck, não foi possível detectar diferenças estatisticamente relevantes para afirmar que as inspeções com ActCheck sejam recomendadas como mais eficazes ou mais eficientes do que inspeções *ad hoc*. De um modo geral, verificou-se que a primeira versão de ActCheck proporciona a detecção de novos tipos de defeito, especialmente informações estranhas e ambiguidades.

Pôde-se perceber a importância da entrevista final para compreender o comportamento observado na análise quantitativa dos resultados. A técnica precisava ser restruturada para que os inspetores aproveitassem melhor seus recursos e não consumissem tanto tempo ao aplicarem-na. O aprendizado obtido através da análise qualitativa foi fundamental para a preparação da nova versão de ActCheck, apresentada ao fim do capítulo.

CAPÍTULO 7 - CONCLUSÕES

Neste capítulo, são apresentadas as considerações finais deste trabalho, destacando as suas contribuições da pesquisa para a engenharia de software. Também são descritas as limitações da pesquisa percebidas e finalmente, são apresentadas propostas de pesquisas futuras.

7.1 Considerações Finais

Esta dissertação apresentou o desenvolvimento de uma tecnologia para promover a qualidade de software cujo processo de desenvolvimento envolva a utilização de diagramas de atividades na especificação de seus requisitos. Esta é um demanda atual da indústria, percebida através dos projetos conduzidos pelo Grupo de Engenharia de Software Experimental da COPPE, especialmente no que tange a concepção de aplicações complexas como *workflows* científicos e aplicações Web.

Ao longo desta pesquisa buscou-se aplicar conceitos recomendados pelo processo de desenvolvimento de tecnologias baseadas em evidência, com a condução de um estudo secundário (*quasi*-revisão sistemática) e de um estudo primário (estudo de viabilidade), além de uma prova de conceito.

Como foi detectada, inicialmente, a carência de tecnologias que apoiassem a inspeção de diagramas de atividades e modelos de *workflow* em geral, houve um esforço significativo para o desenvolvimento da técnica. Primeiramente, buscou-se elaborar uma base de conhecimento prévia para a formulação da técnica, identificando-se casos discrepantes. Em seguida, optou-se por montar um *checklist* configurável contendo itens de avaliação distribuídos por categorias de defeito.

Após este *checklist* ter sido submetido a uma prova de conceito, foi desenvolvida ActCheck, onde novos itens de avalição foram incluídos e o *checklist* foi desmembrado em dois. Apesar de haver algum reagrupamento dos itens de avaliação, ActCheck ainda apresentava uma distribuição de itens de avaliação por categoria de defeito. O inspetor primeiramente deveria procurar por omissões para um dado grupo de itens, depois por fatos incorretos, inconsistências e ambiguidades até deparar-se com itens que o levassem a identificar informações estranhas.

A análise qualitativa realizada no estudo primário apontou que a organização de itens por categorias de defeito não era o melhor caminho a seguir, pois tornava a inspeção uma prática repetitiva e com um encadeamento lógico diverso da prática.

Esta percepção foi refletida nos resultados da análise quantitativa. Com ActCheck, as inspeções foram muito mais demoradas e menos eficientes, embora ligeiramente mais eficazes. Novos defeitos foram encontrados em relação as inspeções *ad hoc*, mas poucos defeitos detectados *ad hoc* foram detectados com ActCheck. Neste sentido, foi preparada uma nova versão de ActCheck, visando aproximar mais a organização dos itens do checklist da realidade prática.

7.2 Contribuições da Pesquisa

Relacionamos a seguir as principais contribuições desta pesquisa:

- Um protocolo de uma *quasi* Revisão Sistemática para investigar os resultados de pesquisa no tópico de inspeção de diagrama de atividades, cujos resultados demonstram atualmente carência de tecnologia;
- A apresentação da primeira versão de ActCheck, uma técnica de inspeção do Diagrama de Atividades baseada em *checklist* e configurável, elaborada a partir de uma versão inicial submetida a uma prova de conceito;
- Estudo que sugere a viabilidade de ActCheck, embora não tenham sido detectados resultados com significância estatística suficiente para afirma que a técnica é mais eficaz e eficientes que inspeções ad hoc;
- A apresentação de uma nova versão de ActCheck ainda não avaliada, buscando corrigir as deficiências sugeridas pelas conclusões extraídas do estudo de viabilidade conduzido sobre a primeira versão.

7.3 Limitações da Pesquisa

Esta pesquisa de mestrado buscou desenvolver um subtema da área de inspeção de software que, embora relevante perante as demandas atuais da indústria, pouco havia sido explorado: a inspeção de diagrama de atividades. Desta forma, não foram encontrados trabalhos que dessem um embasamento significativo para a elaboração da técnica. Também destacamos que foram utilizados poucos diagramas de projetos de software reais ao longo do desenvolvimento da técnica inspeção, embora estes tenham sido utilizados na prova de conceito e no estudo de viabilidade. Conforme relatado no capítulo 5, a versão inicial da técnica de inspeção foi

desenvolvida principalmente sobre o estudo do modelo de atividades da UML e possíveis padrões de modelagem de workflows *(workflow patterns)*.

Conforme relatado na seção 4.4, alguns artigos da *quasi*-Revisão Sistemática não foram acessíveis, bem como não houve uma re-execução mais recente da revisão, que foi realizada em 2009. Além disto, a técnica elaborada foi aplicada em dois projetos desenvolvidos no âmbito da COPPE, sendo submetida a apenas um estudo experimental *in vitro*, que contou com poucos participantes.

7.4 Propostas de Pesquisas Futuras

Como proposta para pesquisa futura, destacamos a continuidade da condução de estudos primários *in vitro* e *in vivo* para avaliação da nova versão de ActCheck e a repetição destes estudos, a fim de ampliar a confiabilidade dos resultados obtidos. Para explorar os recursos da técnica, consideramos importante que projetos de software com técnicas de modelagem distintas sejam utilizadas nos estudos como, por exemplo, *workflows* científicos e modelos que utilizem integralmente todos os recursos sintáticos do diagrama de atividades. Consequentemente, devem ser observados os novos casos discrepantes que podem surgir nas situações de aplicação da técnica para evoluí-la. Propomos também o desenvolvimento de apoio automatizado para aplicação de ActCheck, especialmente no que se refere à caracterização da aplicação, à configuração de *checklists* e aos relatos das inspeções.

Destacamos também a oportunidade futura para a evolução de ActCheck como uma técnica de leitura. À esta proposta, podemos adicionar o desenvolvimento técnicas para apoiar a identificação de defeitos entre o diagrama de atividades e outros modelos da UML, formando assim uma família de técnicas de leitura para apoiar a inspeção do diagrama de atividades, complementar às técnicas existentes em OORTs.

Com base no observado nos resultados do estudo de viabilidade, propomos também que sejam desenvolvidos estudos qualitativos que busquem observar o comportamento dos inspetores ao longo da inspeção de modelos, a fim de direcionar a usabilidade de futuras versões de técnicas de inspeção. Outra proposta extraída da análise qualitativa do estudo de viabilidade conduzido nesta pesquisa refere-se a avaliação do impacto de diferentes metodologias de treinamento na formação dos inspetores.

REFERÊNCIAS BIBLIOGRÁFICAS

(Almendros-Jiménez e Iribarne, 2004) Almendros-Jiménez, J., Iribarne, L. **Describing Use Cases with Activity Charts.** LNCS 3511, pp. 141-159, Springer-Verlag Heidelberg, 2005.

(Barcelos e Travassos, 2006) Barcelos, R. F, Travassos, G. H. **ArqCheck: Uma Abordagem para inspeção de documentos arquiteturais baseada em *checklist***. V Simpósio Brasileiro de Qualidade de Software, 2006.

(Basili et al., 1996) Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S., Zelkowitz, M. **The Empirical Investigation of Perspective-Based Reading**. Empirical Software Engineering: An International Journal, 1(2): 133-164, 1996.

(Basili et al., 1987) Basili, V.R., e Selby, R.W. **Comparing the Effectiveness of Software Testing Strategies.** IEEE Transactions on Software Engineering, SE-12 (7), pp. 1278-1296, Dezembro de 1987.

(Belgamo e Fabbri, 2004) Belgamo, A. e Fabbri, S. **GUCCRA: Contribuindo para a identificação de defeitos em documentos de requisitos durante a construção de modelos de casos de uso.** Workshop de Engenharia de Requisitos, 2004.

(Bock, 2003) Bock, C. **UML 2 Activity and Action Models- Part 2- Actions. Journal of** Object Technology, Vol. 2, No 4. 2003.

(Bock2, 2003) Bock, C. **UML 2 Activity and Action Models. Journal of Object** Technology, Vol. 2, No 5. 2003.

(Bock3, 2003) Bock, C. **UML 2 Activity and Action Models- Part 3- Control Nodes.** Journal of Object Technology, Vol. 2, No 6. 2003.

(Bock, 2004) Bock, C. **UML 2 Activity and Action Models- Part 4- Object Nodes.** Journal of Object Technology, Vol. 3, No 1. 2004.

(Bock2, 2004) Bock, C. **UML 2 Activity and Action Models- Part 5- Partitions.** Journal of Object Technology, Vol. 3, No 7. 2004.

(Choi e Watanabe, 2005) Choi, E., Watanabe, H. **Model Checking Class Specifications for Web Applications.** Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05), 2005.

(Conradi et al., 2003) Conradi, R., Mohagheghi, P., Arif, T., Hegde, L.C., Bunde, G.A., Pedersen, A. **Object-Oriented Reading Techniques for Inspection of UML Models – An Industrial Experiment.** LNCS (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2743, pp. 483-500. Springer- Verlag, 2003.

(Conte et al., 2007) Conte, T. U., Massollar, J., Mendes, E., Travassos, G. H., **Web Usability Inspection Technique Based on Design Perspectives.** Proc. XXI Simpósio Brasileiro de Engenharia de Software, 2007.

(de Mello et al., 2010) de Mello, R. M., Pereira, W. M., Travassos, G. H. **Activity Diagram Inspection on Requirements Specification.** XIV Simpósio Brasileiro de Engenharia de Software, 2010.

(Elsevier, 2009) **Scopus**, disponível em http://www.scopus.com. Último acesso: novembro de 2009.

(Eshuis e Wieringa, 2004) Eshuis, R., Wieringa, R. **Tool support for verifying UML activity diagrams**. IEEE Transactions On Software Engineering, vol. 30, no. 7, 2004.

(Fagan, 1986) Fagan, M. E. **Advances in Software Inspections.** IEEE Transactions on Software Engineering, Vol. SE-12, no.7, 1986.

(Fowler, 2004) M. **UML Distilled- A brief guide to the Standard Object Modeling Language.** 3ed. Addison-Wesley, 2004.

(Gamma et al., 1995) Gamma, E., Helm, H., Johnson, R., Vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software.** Addison-Wesley, Reading, Massachusetts, 1995.

(Guelfi e Mammar, 2005) Guelfi, N., Mammar, A. **A formal semantics of timed activity diagrams and its PROMELA translation**. 12th Asia-Pacific Software Engineering Conference, 2005.

(Gutiérrez et al., 2008) Gutiérrez, J, , Nebut C., Escalona, M., Mejías M., Ramos, I. **Visualization of Use Cases through Automatically Generated Activity Diagrams** LNCS 5301, pp. 83–96. Springer-Verlag, 2008.

(Gross e Doerr, 2009) Gross, A., Doerr, J. **EPC vs. UML Activity Diagram - Two Experiments Examining their Usefulness for Requirements Engineering.** 17th IEEE International Requirements Engineering Conference, 2009.

(He e Carver, 2006) He, L., Carver, J. **PBR vs. Checklist: A Replication in the N-Fold Inspection Context.** ISESE'06, September 2006, Rio de Janeiro, Brazil. ACM, 2006.

(IEEE, 1998) IEEE. **IEEE Standard 1028- IEEE Standard for Software Review.** Institute of Electrical and Electronics Engineers Inc., New York, 1998.

(IEEE, 1990) IEEE. **IEEE Standard glossary of software engineering terminology, Standard 610.12**. IEEE Press, 1990.

(Kalinwoski e Travassos, 2004) Kalinwoski, M., Travassos, G. H. **ISPIS: A Framework Supporting Software Inspection Process**. Proceedings of 19th International Conference on Automated Software Engineering (ASE'04). IEEE, 2004.

(Kitchenham, 2004) Kitchenham, B. **Procedures for Performing Systematic Reviews.** Joint Technical Report TR/SE-0401. Software Engineering Group, Department of Computer Science, Keele University, UK and Empirical Software Engineering National ICT Australia Ltd, 2004.

(Kitchenham et al., 2004) Kitchenham, B. A., Dybå, T., Jørgensen, M. **Evidence-based Software Engineering**. 2004.

(Laitenberger et al., 2000) Laitenberger, O., Atkinson, C., Schlich, M., El Anam, K. **An experimental comparison of reading techniques for defect detection in UML**

design documents. The Journal of Systems and Software 53, pp 183-204, ElSevier Science Inc., 2000.

(Laitenberger et al., 2001) Laitenberger, O., El Anam, K., Harbich, T. G. **An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective- Based Reading of Code Documents.** IEEE Transactions on Software Engineering, vol. 27, No. 5, 2001.

(Machado et al., 2005) Machado, R. J., Ramos, I., Fernandes, J. M., **Specification of Requirements Models.** Engineering and Managing Software Requirements, pp 47-68, Springer-Berlin Heidelberg, 2005.

(Mafra e Travassos, 2006) Mafra, S. N., Travassos, G. H., **Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos.** V Simpósio Brasileiro de Qualidade de Software- SBQS, 2006.

(Mafra et al., 2006) Mafra, S. N., Barcelos, R. F., Travassos, G. H. **Aplicando uma Metodologia Baseada em Evidência na Definição de Novas Tecnologias de Software.** In: XX SBES, Florianópolis, SC, Brasil, 2006.

(Massollar, 2008) Massollar, J. **Uma Abordagem Baseada na Engenharia para Desenvolvimento e Garantia da Qualidade de Aplicações Web.** Exame de Qualificação. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2008.

(Nielsen, 1994) Nielsen, J. **Heuristic evaluation.** In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods (John Wiley & Sons, 1994), pp. 25 – 62, 1994.

(OMG, 2009) OMG. **OMG Unified Modeling Language (OMG UML), Superstructure- Version 2.2.** Disponível em: http://www.omg.org/spec/UML/2.2/Superstructure

(OMG2, 2009) OMG. **OMG Unified Modeling Language (OMG UML)**, Infrastructure- Version 2.2.

(OMG3, 2009) OMG. **Business Process Modeling Notation- version 1.2** Disponível em http://www.omg.org/docs/formal/09-01-03.pdf. Último acesso: maio de 2009

(Pai et al., 2004) Pai, M., McCulloch, M., Gorman, J. D., Pai, N., Enanoria, W., Kennedy, G., Tharyan, P., Colford Jr., J. M. **Systematic reviews and meta-analyses: An illustrated, step-by-step guide.** The National Medical Journal of India, Vol. 17, No. 2, 2004.

(Parnas e Weiss, 1985) Parnas, D. Weiss, D. **Active Design Reviews: Principles and Practice.** Proceedings of 8th International Conference on Software Engineering, p132-136, 1985.

(Pastor et al., 2001) Pastor, O., Abrahão, S., Fons, J. **An Object-Oriented Approach to Automate Web Applications Development.** LNCS 2115, pp. 16-28. Springer-Verlag Berlin Heidelberg, 2001.

(Pereira et al., 2009) Pereira, W. M., Araújo, M. A. P., Travassos, G. H. **Apoio na Concepção de *Workflow* Científico Abstrato para Estudos *in virtuo* e *in silico* em Engenharia de Software**, 2009.

(Petersson, 2002) Petersson, H. **Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis.** Technical report 147, Department of Communication Systems, Lund Institute of Technology. Lund University, 2002.

(Pilone e Pitman, 2005) Pilone, B. , Pitman, N. **UML 2.0 in a Nutshell.** O'Reilly, June 2005.

(Porter e Votta, 1994) Porter, A. A., Votta, L. G. **An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections.** Proceedings of 16th International Conference on Software Engineering, p103-112, 1994.

(Porter et al., 1995) Porter, A. A., Votta, L. G., Basili, V. R. **Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment.** IEEE Transactions on Software Engineering, Vol. 21, No. 6, 1995.

(Qian et al., 2007) Qian, Y., Xu, Y., Wang, Z., Pu, G., Zhu, H., Cai, C. **Tool Support for BPEL Verifcation in ActiveBPEL Engine.** Proceedings of the 2007 Australian Software Engineering Conference (ASWEC'07). IEEE, 2007.

(Riehle. D. e Zullighoven, H., 1996) Riehle, D., Zullighoven, H. **Understanding and Using Patterns in Software Development.** Theory and Practice of Object Systems, 2(1):3-13, 1996.

(Russel et al., 2005) Russel, N, ter Hofsted, A. H. M., Edmond, D., van der Aalst, W. M. P. **Workflow Data Patterns: Identification, Representation and Tool Support.** LNCS 3716, pp 353-368. Springer-Verlag Berlin Heildeberg, 2005.

(Russel et al., 2006) Russel, N., van der Aalst, W. M. P., ter Hofsted, A. H. M., Wohed, P. **On the Suitability of UML 2.0 Activity Diagrams for Business Process Modeling.** Third Asia-Pacific Conference on Conceptual Modeling, Hobart, Australia, 2006.

(SAS, 2011) SAS. **JMP Software**, disponível em: http://www.jmp.com/.

(Sabaliauskaite et al., 2003) Sabaliauskaite, G., Matsukawa, F., Kusumoto, S., Inoue, K. **Further investigations of reading techniques for object-oriented design inspection.** Information and Software Technology 45, pp 571-585. ElSevier Science B. V., 2003.

(SEI, 2006) SEI. **CMMI for Development, version 1.2.** Carnegie Mellon University, 2006.

(Shinha e Paradkar, 2010) Shinha, A., Paradkar, A. **Use Cases to Process Specifications in Business Process Modeling Notation.** Proceedings of International Conference on Web Services, 2010.

(Shull, 1998) Shull, F. **Developing Techniques for Using Software Documents: A Series of Empirical Studies.** PhD Thesis, Department of Computer Science, University of Maryland, USA.

(Shull et al., 1999) Shull, F., Travassos, G. H., Carver, J., Basili, V. R. **Evolving a Set of Techniques for OO Inspections.** University of Maryland Technical Report CS-TR-4070, Outubro de 1999.

(Shull et al., 2000) Shull, F., Rus, I., Basili, V., **How Perspective-Based Reading Can Improve Requirements Inspections.** 2000.

(Shull et al., 2001) Shull, F., Carver, J., Travassos, G., **An Empirical Methodology for Introducing Software Processes.** Proceedings of the 8th ESEC - 9th ACM SIGSOFT FSE. ACM Press, p.p.288 – 296, 2001.

(Shull et al., 2003) Shull, F., Carver, J., Travassos, G. H., Maldonado, J. C., Conradi, R., and Basili, V. **Replicated Studies: Building a Body of Knowledge about Software Reading Techniques.** Lecture Notes on Empirical Software Engineering, pp 39-84, World Scientific Publishing Co., Inc., USA, NJ, 2003.

(SOFTEX, 2006) Softex. **MPS-BR- Melhoria de Processo de Software Brasileiro- Guia Geral- versão 1.1**. Maio de 2006.

(Störrle, 2004) Störrle, H. **Semantics of Control-Flow in UML 2.0 Activities.** Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC'04). IEEE, 2004.

(Tanriöver e Bilgen, 2007) Tanriöver, Ö., Bilgen, S. **An Inspection Approach for Conceptual Models in Notations Derived from UML: A Case Study.** ISCIS 2007: 22nd International International Symposium on Computer and Information Sciences, 7-9, pp 1-6, 2007.

(Thelin et al., 2003) Thelin, T., Runeson, P., Wohlin, C., **An Experimental Comparison of Usage-Based and Checklist-Based Reading.** IEEE Transactions on Software Engineering, Vol. 29, No. 8, 2003.

(Travassos et al., 1999) Travassos, G. H., Shull, F., Fredericks, M, Basili, V.R. **Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality.** Proc. International Conference on Object Oriented Programming Systems, Languages & Applications, 1999.

(Travassos et al., 1999b) Travassos, G. H., Shull, F., Carver, J. **Evolving a Process for Inspecting OO Designs.** XIII Simpósio Brasileiro de Engenharia de Software: *Workshop de Qualidade de Software.* Outubro, 1999.

(Travassos, 2001) Travassos, G. H. *apud* Rocha, A. R. C., Maldonado, J. C., Weber, K. C. **Qualidade de Software- Teoria e Prática.** São Paulo, 1ed., Prentice Hall, 2001.

(Travassos et al., 2002) Travassos, G. H., Shull, F., Carver, J., Basili, V. **Reading Techniques for OO Design Inspections.** Technical Report CS-TR-4353, University of Maryland Computer Science Department, 2002. Disponível em: http://www.cs.umd.edu/Library/TRs. Último acesso: fevereiro de 2009.

(Travassos, 2008) Travassos, G. H. **Notas de Aula- Disciplina de Engenharia de Software Orientada a Objetos.** Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2008.

(Travassos, 2008b) eSEE

(van der Aalst, 1999) van der Aalst, W. M. P. **Formalization and verification of event-driven process chains.** Information as Software Technology 41, pp 639-650. Elsevier, 1999.

(van der Aalst et al., 2003) van der Aalst, W. M. P., Stoffele, M., Wamelink, J. W. F. **Workflow Patterns.** Distributed and Parallel Databases vol. 14, n.1, pp. 5-51. 2003. Springer Netherlands.

(van der Aalst; Hofstede, 2005) van der Aalst, W.M.P., Ter Hofstede, A.H.M. YAWL: **Yet Another Workflow Language.** Information Systems vol. 30 Issue 4, pp 245-275. Elsevier, 2005.

(Wohed et al., 2005) Wohed, P., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M., Russel, N. **Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams.** LNCS 3716, pp. 63-78. Springer- Verlag Berlin Heidelberg, 2005.

(Wohed et al., 2006) Wohed, P., van der Aalst, W.M.P., Dumas, M., Ter Hofstede, A.H.M., Russell, N. **On the suitability of BPMN for Business Process Modeling.** LNCS 4102 pp. 161-176. Srpinger-Verlag Berlin Heidelberg, 2006.

(Wong, 2006) Wong, Y. K. **Modern Software Review- Techniques and Technologies.** IRM Press, 2006.

(Wynn et al., 2009) Wynn, M. T., Verbeek, H. M. W., van der Aalst, W. M. P, ter Hofstede, A. H. M., Edmond, D. **Business process verification- finally a reality!** Business Process Management Journal, vol. 15, no. 1, pp. 74-92. Emerald, 2009.

(Xexéo, 2007) Xexéo, G., **Modelagem de Sistemas de Informação- Da análise de requisitos ao modelo de interface.** Edição de agosto de 2007, disponível em http://wiki.xexeo.org. Último acesso: fevereiro de 2009.

(Xu et al., 2006) Xu K., Liu, Y., Wu C. **Guided Reasoning of complex e-business process with business bug patterns**, International Conference on e-busineess Engineering, 2006.

(Yourdon, 1992) Yourdon, E. **Análise Estruturada Moderna.** 3 ed. Editora Campus, 1992.

APÊNDICE A- PROTOCOLO DA *QUASI*- REVISÃO SISTEMÁTICA

A.1 Formulação da Questão

A.1.1 Análise PICO

Population: Artigos que descrevam abordagens de inspeção aplicáveis a modelos de workflow em projetos de software

Intervention: Abordagens de Inspeção aplicáveis a Diagrama de Atividades, BPMN, EPC, YAWL, Fluxogramas ou outras linguagens de workflow.

Comparison: Não há

Outcome: Identificar as abordagens, suas descrições, heurísticas utilizadas, sua aplicabilidade.

A.2 Qualidade da Questão e Amplitude

A.2.1 Descrição do Problema

Desde a sua primeira versão em 1998, a UML tem se destacado como uma linguagem padrão para Modelagem de projetos orientados a objetos, através de seus diagramas para representação estática e dinâmica de sistemas. Com o advento da versão 2.0 da UML em 2003, o Diagrama de Atividades ganhou uma melhor definição e tornou-se um modelo de representação dinâmica mais abrangente, não se limitando mais somente a projetos de sistemas orientados a objeto, passando a ser, inclusive, adotado como uma notação para representação de *workflows* em geral.

Na última década, também é perceptível um esforço significativo da comunidade científica no aprimoramento de atividades que promovam a garantia da qualidade do produto final não só através da verificação do código-fonte, mas também dos demais artefatos elaborados ao longo do processo de desenvolvimento. A garantia da qualidade do produto de software é o objetivo principal das atividades de VV&T (Verificação, Validação e Teste), onde se inclui a Inspeção de Software, uma atividade revisional que busca identificar defeitos nos mais diversos artefatos de software a fim de corrigi-los antes que futuras falhas ocorram.

Atualmente, devido ao potencial de redução de esforço para um projeto de

software, a inspeção de requisitos e de modelos iniciais dos projetos de software passou a contar com abordagens sistemáticas especialmente desenvolvidas para apoiar tais atividades, como é o caso de OORTs, uma abordagem de inspeção dotada de roteiros conhecidos como técnicas de leitura para identificar defeitos entre modelos UML. A princípio, entretanto, não é conhecida uma abordagem de inspeção que trate do Diagrama de Atividades.

Com a evolução da demanda de sistemas complexos, especialmente de aplicações Web, métodos específicos têm sido propostos a fim de apoiar de forma sistemática o desenvolvimento deste tipo de aplicação. Neste contexto, Massollar (2008) apresenta uma proposta de pesquisa que tem como objetivo principal a elaboração de *"uma abordagem baseada na Engenharia estruturada sobre um conjunto de atividades relacionadas à engenharia do produto e à garantia da qualidade, que apoie o desenvolvimento sistemático de aplicações Web baseando-se, principalmente, na construção de modelos que capturem as diversas perspectivas de projeto Web e na garantia da qualidade desses modelos através de técnicas de inspeção"*.

Desta forma, mais do que um método, a proposta de Massollar consiste na especificação e na experimentação de um processo para a engenharia e garantia da qualidade de aplicações Web. Dentre outras características, a proposta busca aproveitar a padronização já existente da UML como notação para modelagem de artefatos de projeto. Dentre os modelos adotados, o Diagrama de Atividades possui um papel fundamental na fase de especificação, onde cada Diagrama de Atividades deverá descrever um caso de uso da aplicação, com o auxílio de estereótipos.

Outro motivador para a identificação de uma abordagem para Inspeção do Diagrama de Atividades é a pesquisa que vem sido desenvolvida na COPPE para a definição de um abordagem para a concepção de *workflows* científicos em experimentos de Engenharia de Software, mais especificamente, experimentos *in virtuo* e *in silico* de larga escala . A princípio, esta abordagem prevê o uso do Diagrama de Atividades para representação dos *workflows* e prevê a inspeção dos *workflows* concebidos (Pereira, 2009).

Entende-se por inspeção como o exame visual de um produto de software que detecta e identifica anomalias de software, incluindo erros e desvios de padrões e de especificações (IEEE, 1998). Entretanto, conforme a terminologia estabelecida no padrão IEEE 610.12 (IEEE, 1990), as inspeções tendem a identificar defeitos e não erros. Segundo esta norma, "defeito" consiste em um passo, processo ou definição de dados incorretos, enquanto que "erro" consiste em qualquer estado ou resultado inesperado na execução de um programa.

É importante observar que existe uma confusão na literatura e na indústria quanto ao uso dos termos “revisão”, “inspeção” e “*walkthrough*” (Wong, 2006). A revisão propriamente dita pode ser considerada um processo ou encontro em que um produto de software é apresentado ao pessoal do projeto, clientes, usuários e demais interessados para comentários ou aprovação, não havendo necessariamente a identificação de defeitos.

Um dos fatores decisivos no planejamento e nos resultados da inspeção de um projeto de software é a definição da abordagem de inspeção que será utilizada. Uma inspeção pode ser realizada de modo *ad hoc,* dependendo exclusivamente da experiência do revisor, como também pode ser apoiada pelo uso de *checklists* ou pela aplicação de técnicas de leitura, que possuem um grau de formalidade maior e dependem menos da experiência do revisor para alcançar bons resultados. Ainda há a alternativa do uso de heurísticas para apoiar a inspeção.

As técnicas de leitura podem ser definidas como uma série de procedimentos que podem ser adotados por um revisor para obter um entendimento do artefato sob revisão, provendo um guia sistemático para a identificação de defeitos (Wong, 2006). He e Carver (2006) observam que, enquanto as inspeções *ad hoc* e via *checklists* são intuitivas e baseadas em procedimentos não-sistemáticos, as técnicas de leitura possuem procedimentos explícitos e sistemáticos. Independente do uso de *checklists*, técnicas de leitura ou heurísticas, espera-se que uma abordagem de inspeção não seja necessariamente dependente de apoio computacional, apresentando um roteiro em linguagem natural para orientar o inspetor.

Uma abordagem de inspeção pode ainda avaliar os artefatos individualmente ou pode realizar uma inspeção conjunta, relacionando dois ou mais artefatos. Tanriöver e Bilgen (2007), por exemplo, classificam a inspeção individual de modelos UML de Inspeção Intra-diagramas, enquanto que inspeção conjunta é chamada de Inspeção Inter-diagramas. Travassos *et al.* (2002) apresentam as duas dimensões em que a Inspeção entre artefatos de software podem ocorrer: a horizontal, quando os artefatos encontram-se em um mesmo nível de abstração ou a vertical, quando os artefatos encontram-se em níveis de abstração distintos. Enquanto a inspeção horizontal foca a verificação da consistência dos artefatos, a inspeção vertical foca a verificação da rastreabilidade entre os artefatos.

Para ser proposta, portanto, uma abordagem para inspeção do Diagrama de Atividades faz-se necessário, primeiramente, identificar na literatura especializada se já existem tecnologias que atendam a esta necessidade. Entretanto, as características inerentes ao Diagrama de Atividades e sua aplicabilidade na fase de especificação de requisitos e, principalmente, na especificação de *workflows*, sugerem a importância de

serem pesquisadas não somente abordagens para inspeção do próprio Diagrama de Atividades da UML, como também abordagens para inspeção de modelos de *workflow* em geral ou abordagens que porventura existam para inspeção de outras notações com estrutura e aplicabilidade semelhantes ao Diagrama de Atividades, como é o caso dos fluxogramas e das seguintes notações BPMN, YAWL e EPC (Wynn et al., 2009):.

- YAWL (Yet Another Workflow Language), linguagem apresentada pelo próprio criador dos *Workflow Patterns* (Van Der Aalst e Hofstede, 2005), com o objetivo de ser uma linguagem de *workflow* mais completa que abranja todos os *Workflow Patterns*. Também é baseada nas Redes de Petri, como o Diagrama de Atividades.
- BPMN (Business Process Modeling Notation): têm por objetivo principal prover uma notação de leitura compreensível para todos os usuários, dos analistas de negócio aos desenvolvedores, criando uma ponte padronizada entre o desenho de processos de negócio e a implementação do processo (OMG, 2009). Visualmente, esta notação é bem semelhante ao Diagrama de Atividades, que, inclusive, foi uma das notações examinadas para sua elaboração;
- EPCs: Event Process Chains, linguagem de modelagem de processos que, juntamente com sua forma estendida eEPCs, tornaram-se bastante difundidas por serem adotadas em ferramentas famosas de Planejamento de Recursos Empresariais (ERP) e de Gestão de Workflow (WFM), como SAP R/3 e ferramentas de Reengenharia de Processo de negócio, como ARIS (Van Der Aalst, 1999).

Em uma busca prévia utilizando a máquina de busca Scopus (Elsevier, 2009) para localizar as expressões *"inspection"* e *"activity diagram"* em resumos de artigos, somente foi localizado um artigo que apresenta uma abordagem para inspeção de um diagrama adaptado do Diagrama de Atividades, denominado de "diagrama de workflow" e pertencente à notação KAMA (Tanriöver e Bilgen, 2007).

A.2.2 Questão

µ0: Quais são as abordagens para inspeção aplicáveis a modelos de workflow em projetos de software?

A.2.3 Palavras-chave e Sinônimos

- Inspeção: revisão, verificação, validação, leitura, *checklist*

Inspection: review, verification, validation, reading, checklist

- Workflow: diagrama de atividades, modelo de atividades, BPMN, Business Process Modeling Notation, EPC, eEPC, Event-driven Process Chain, Fluxograma, YAWL, Yet Another Workflow Language

  Workflow: activity diagram, activity model, BPMN, Business Process Modeling Notation, EPC, eEPC, Event-driven Process Chain, Flowchart, YAWL, Yet Another Workflow Language

- Software: processos de negócio, UML, Unified Modeling Language

  Software, business process, UML, Unified Modeling Language

A.2.4 População

Artigos que descrevam técnicas de inspeção aplicáveis a modelos de workflow em projetos de software

A.2.5 Intervenção

Técnicas de Inspeção aplicáveis a Diagrama de Atividades, BPMN, EPC, YAWL, Fluxogramas ou outras linguagens de workflow.

A.2.6 Resultado

Identificar as técnicas, suas descrições, heurísticas utilizadas, sua aplicabilidade.

A.2.7 Medição dos Resultados

Avaliação das abordagens propostas, atribuição de conceito para a qualidade de cada artigo

A.2.8 Aplicação

Projetos de desenvolvimento de software que adotem a inspeção de modelos de workflow

A.2.9 Artigo de Controle

Özgur Tanriöver e Semih Bilgen - An Inspection Approach for Conceptual Models in Notations Derived from UML: A Case Study (Tanriöver e Bilgen, 2007)

A.3 Seleção de Fontes

A.3.1 Critérios para Seleção de Fontes

À exceção dos anais do SBES e do SBQS cuja pesquisa será parcialmente manual, as fontes deverão:

- Garantir resultados únicos através da busca de um mesmo conjunto de *strings*;
- Permitir pesquisas de *strings* somente no resumo (*abstract*) e título;
- Comportar as *strings* de pesquisa.

Linguagem das Fontes

Português (anais do SBES e do SBQS), Inglês e Espanhol

A.3.2 Identificação de Fontes

Método de Pesquisa de Fontes

À exceção dos anais do SBES e do SBQS, que serão obtidos a partir do portal BDCOMP, as demais fontes serão pesquisadas através da máquinas de busca existentes na web. Serão informadas *strings* ou conjuntos de *strings* de pesquisa com efeito semelhante conforme a linguagem de cada máquina de busca utilizada.

***String* de Pesquisa**

Tendo como base as palavras-chave e sinônimos enumerados neste protocolo, e considerando as linguagens específicas de cada máquina de busca, foi elaborada uma *string* de pesquisa para cada máquina de busca com efeito semelhante. No caso das publicações do SBES e SBQS, será verificada a incidência de tais termos sem apoio automatizado.

Scopus

TITLE-ABS-KEY (((inspection* OR review* OR reading* OR verification* OR validation* OR checklist*) AND (software* OR "business process" OR UML OR "Unified Modeling Language")) AND ("activity diagram" OR "activity diagrams" OR "activity model" OR "activity models" OR BPMN* OR "Business Process Modeling Notation" OR EPC* OR eEPC* OR "Event-driven Process Chain" OR

Flowchart* OR Workflow* OR YAWL* OR "Yet Another Workflow Language"))

EI Compendex

(((inspection* OR review* OR reading* OR verification* OR validation* OR checklist*)
AND (software* OR "business process" OR UML OR "Unified Modeling Language")) AND ("activity diagram" OR "activity diagrams" OR "activity model" OR "activity models" OR BPMN* OR "Business Process Modeling Notation" OR EPC* OR eEPC* OR "Event-driven Process Chain" OR Flowchart* OR Workflow* OR YAWL* OR "Yet Another Workflow Language")) WN KY

IEEE Xplore

(((inspection* OR review* OR reading* OR verification* OR validation* OR checklist*)<in>ti) OR ((inspection* OR review* OR reading* OR verification* OR validation* OR checklist*)<in>ab) OR ((inspection* OR review* OR reading* OR verification* OR validation* OR checklist*)<in>de)) AND
(((software* OR "business process" OR UML OR "Unified Modeling Language")<in>ti) OR ((software* OR "business process" OR UML OR "Unified Modeling Language")<in>ab) OR ((software* OR "business process" OR UML OR "Unified Modeling Language")<in>de)) AND
((("activity diagram" OR "activity diagrams" OR "activity model" OR "activity models" OR BPMN* OR "Business Process Modeling Notation" OR EPC* OR eEPC* OR "Event-driven Process Chain" OR Flowchart* OR Workflow* OR YAWL* OR "Yet Another Workflow Language")<in>ti) OR (("activity diagram" OR "activity diagrams" OR "activity model" OR "activity models" OR BPMN* OR "Business Process Modeling Notation" OR EPC* OR eEPC* OR "Event-driven Process Chain" OR Flowchart* OR Workflow* OR YAWL* OR "Yet Another Workflow Language")<in>ab) OR (("activity diagram" OR "activity diagrams" OR "activity model" OR "activity models" OR BPMN* OR "Business Process Modeling Notation" OR EPC* OR eEPC* OR "Event-driven Process Chain" OR Flowchart* OR Workflow* OR YAWL* OR "Yet Another Workflow Language")<in>de))

Lista de Fontes

Primeiramente, as seguintes fontes serão utilizadas:

- Anais do SBES dos anos de 1999 até 2008 (via portal BDComp);

- Anais do SBQS de 2002, 2003 e 2004 (via portal BDComp);
- Máquinas de busca aglutinadoras, nesta ordem: Scopus e Compendex

Para complementar a pesquisa, também serão utilizadas as seguintes máquinas de busca específicas:

- IEEE Digital Library;
- SpringerLink Digital Library.

Seleção de Fontes após Avaliação

A *priori*, todas as fontes listadas satisfazem os critérios de qualidade

Critérios para Seleção de Fontes

No caso da SpringerLink Digital Library, a *string* de pesquisa precisaria ser particionada mais de 50 vezes, sendo os resultados posteriormente agrupados devido a limitação da máquina de busca para somente 10 critérios. Além disto, a SpringerLink não permite pesquisa através de palavras-chave, somente sumário e título. Devido à estas limitações, a SpringerLink Digital Library foi excluída do grupo de máquinas de busca. As demais fontes forma aprovadas, com as seguintes ressalvas:

- No caso da máquina IEEEXplore, corre-se risco de resultados não serem exibidos, caso sejam retornados mais de 500 artigos;
- À exceção dos anais do SBES e do SBQS, não foi fixada uma data mínima de publicação, mas a máquina de busca Compendex estabelece como data mínima o ano de 1969.

A.4 Estratégia para Extração das Informações

Serão extraídas do conteúdo de cada artigo selecionado as seguintes informações:

- Título do Artigo
- Autores
- Fonte
- Tipo de Artigo (Teórico, Aplicação Industrial, Estudo Experimental)
- Nome da Abordagem de inspeção
- Nome do modelo inspecionado (Diagrama de Atividades/ EPC/ BPMN/ Fluxograma/ YAWL/ outro <especificar>/ independente de modelo)
- A abordagem visa inspecionar o modelo de workflow individualmente? (Sim/ Não)
- A abordagem visa inspecionar o modelo de workflow em relação a outro(s) artefato(s) num mesmo nível de abstração? (Sim/ Não) Quais são estes artefatos?
- A abordagem visa inspecionar o modelo de workflow em relação a outro(s) artefato(s) em níveis de abstração distintos? (Sim/ Não) Quais são estes artefatos?
- Tipo de abordagem (Checklist/ Técnica de leitura/ heurísticas)
- Foco da abordagem (sintaxe/ semântica/ ambos)
- Referência que contém a descrição da Abordagem (informar “o próprio artigo” ou descrever a referência bibliográfica)
- A abordagem já foi aplicada? (Sim/ Não)
- Em que tipos de projeto a abordagem já foi aplicada?
- Em que tipos de organização a abordagem já foi aplicada? (Academia, Indústria pequeno ou médio porte/ Indústria grande porte)

No caso do artigo apresentar um estudo, também serão extraídas:

- Data e local de execução do estudo
- Propósito do estudo (caracterizar/ avaliar/ predizer/ controlar/ melhorar)
- Descrição do estudo
- Participantes
- Material utilizado
- Projeto experimental
- Ameaças à validade do estudo
- Resultados do estudo
- Lições aprendidas
- Perspectivas futuras
- Comentários adicionais

A.5 Critérios para Avaliação da Qualidade dos Artigos

Cada artigo selecionado deverá ser avaliado conforme as questões da tabela a seguir. Ao final será atribuída uma nota referente à qualidade de cada artigo:

Questão	**Conceitos atribuídos**				
	Artigo 1	**Artigo 2**	**Artigo 3**	**Artigo 4**	**Artigo 5**
1. Onde se localiza a descrição da abordagem de inspeção?					
2. O artigo utiliza a terminologia adequada?					
3. O artigo explicita as restrições e as condições de aplicação da abordagem de inspeção?					
4. O artigo apresenta algum estudo sobre a abordagem de inspeção?					
5. Os resultados do estudo sobre a abordagem de inspeção são descritos?					

- Para a questão 1, será atribuído 1 ponto somente se a descrição da abordagem estiver no próprio artigo. Nenhum ponto será atribuído ao artigo caso a descrição encontre-se em outra referência.
- Para a questão 2, será atribuído 0,5 ponto caso o artigo não apresente equívocos na utilização de termos relativos aos conceitos de engenharia de software envolvidos. Exemplo: Usar o termo “erro” no lugar do termo “defeito”.
- Para a questão 3, será atribuído 0,5 ponto para cada item que seja explicitado: restrições da abordagem de inspeção e condições de aplicação da abordagem de inspeção.
- Para a questão 4, será atribuído 1 ponto caso o artigo apresente algum tipo de estudo (experimental ou não). 0,5 ponto será atribuído caso o artigo apresente apenas prova de conceito ou relato de aplicação industrial. Nenhum ponto será atribuído caso o artigo não apresente nenhum estudo.
- Para a questão 5, será atribuído 0,5 ponto ao artigo que apresente algum tipo de estudo, aplicação industrial ou prova de conceito e apresente os respectivos resultados.

A.6 Seleção de Artigos

A.6.1 Definição dos Artigos

Critérios de Inclusão e Exclusão de Artigos

Para a pré-seleção dos artigos, serão considerados os seguintes critérios:

- O título e resumo dos artigos devem estar disponíveis na Web.
- O resumo (*abstract*) deve indicar a caracterização ou aplicação de uma abordagem de inspeção de modelos de workflow;
- Os artigos com resumo aprovado (pré-selecionados), deverão estar preferencialmente disponíveis na web, acessíveis através do *link* da máquina de busca. Caso não estejam disponíveis, serão utilizados até 2 e-mails disponíveis dos primeiros autores de cada artigo para solicitá-los. Caso não haja algum retorno no prazo de 20 dias corridos a contar da data de solicitação, as referências faltantes serão consideradas indisponíveis.
- O título e resumo dos artigos devem estar escritos em inglês (máquinas de busca e BDComp) ou português (BDComp).

Para a seleção dos artigos, serão considerados os seguintes critérios:

- Os artigos devem estar escritos em inglês, espanhol ou em português (anais do SBES e SBQS);
- Os artigos devem permitir a identificação de pelo menos uma abordagem de inspeção de modelos de *workflow*;
- Não serão considerados artigos que não caracterizem ou que não apresentem uma referência bibliográfica que caracterize a abordagem de inspeção mencionada;
- Não serão considerados artigos que apresentem abordagens de inspeção cuja descrição não esteja em linguagem natural;
- Não serão considerados artigos que apresentem abordagens de inspeção que não inspecionem o próprio modelo de workflow e que somente o comparem com artefatos que não sejam nem a descrição dos requisitos nem modelos UML;
- Não serão considerados artigos cuja abordagem de inspeção dependa de transformações dos modelos de *workflow* em outras representações, como redes de Petri, algoritmos ou grafos;
- Não serão considerados artigos cuja abordagem de inspeção dependa de apoio computacional;

- Não serão considerados artigos que somente descrevam regras e/ou critérios de qualidade, sem especificar ou referenciar uma abordagem de inspeção;
- Não serão considerados artigos cuja abordagem de inspeção não apresentem oportunidades para identificação de defeitos, como abordagens baseadas em regras de redução e refatorações.

Definição de Tipos de Artigo

Serão considerados artigos teóricos, aplicações na indústria, estudos experimentais ou combinações entre estes.

Procedimentos para Seleção de Artigos

O pesquisador aplicará os critérios de pesquisa sobre as fontes. Os artigos identificados serão classificadas em pré-selecionados ou excluídos, conforme a leitura do resumo (*abstract*). Em seguida, os artigos pré-selecionados serão considerados incluídos, excluídos ou não definidos pelo mesmo pesquisador. Um segundo pesquisador realizará uma avaliação da classificação feita pelo primeiro pesquisador. Ao final, artigos não definidos serão incluídos. Em seguida, o primeiro pesquisador irá extrair as informações previstas dos artigos incluídos e o segundo pesquisador irá avaliar a qualidade dos artigos.

A.6.2 Execução da Seleção

Uma rodada de teste foi executada em maio de 2009 na máquina de busca principal (Scopus), onde foram identificados 559 artigos sendo que 97 foram e pré-selecionados através de análise do resumo (*abstract*), mas apenas um destes artigos atendeu a todos os critérios de inclusão e exclusão. Este artigo é o mesmo previamente definido como artigo de controle (Tanriöver e Bilgen, 2007)

Após a análise dos resultados, para a execução da revisão completa, decidiu-se pelo aprimoramento da *string* de pesquisa, de modo a incorporar alguns termos de significado semelhante e variações dos termos (como plurais). Foram elaboradas *strings* de pesquisa correspondentes para as máquinas de busca Compendex e IEEEXplore Digital Library. A execução da pesquisa nas máquinas de busca deu-se entre 04 de setembro de 2009 (Scopus e Compendex) e 05 de setembro de (IEEEXplore). Os gráficos na Figura 1 apresentam os resultados encontrados. Como a Scopus é a base principal, todos os artigos identificados nesta máquina de busca

foram considerados para efeito de identificação de artigos distintos, sendo desconsideradas repetições destes artigos na Compendex e na IEEEXplore Digital Library. Importante destacar que mais da metade dos artigos retornados pela IEEEXplore não foram retornados pela Scopus.

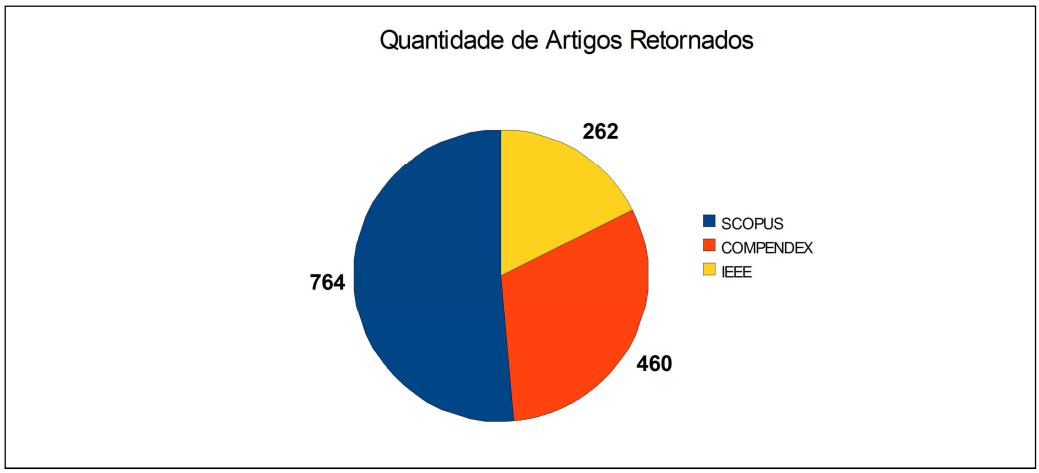

Figura 1- Quantidade de Artigos retornados e artigos distintos identificados

Dentre os artigos pré-selecionados (figura 2), nenhum pertencia à Compendex, diferentemente da IEEEXplore, onde mais de um terço dos artigos distintos foram aproveitados. Dos 170 artigos pré-selecionados das ferramentas de busca, 19 deixaram de ser lidos devido aos autores não terem encaminhado os artigos solicitados. No caso dos anais do SBES e do SBQS, os mesmos termos foram pesquisados manualmente no portal BDComp em 05 de setembro de 2009, sendo que não foi identificado nenhum artigo que nem correspondesse à *string* de pesquisa.

Apesar do incremento da *string* de pesquisa e da execução em todas as fontes pré-definidas, o resultado de artigos selecionados continuou sendo o mesmo: os critérios de inclusão e de exclusão somente foram atendidos pelo artigo de Tanriöver e Bilgen (2007).

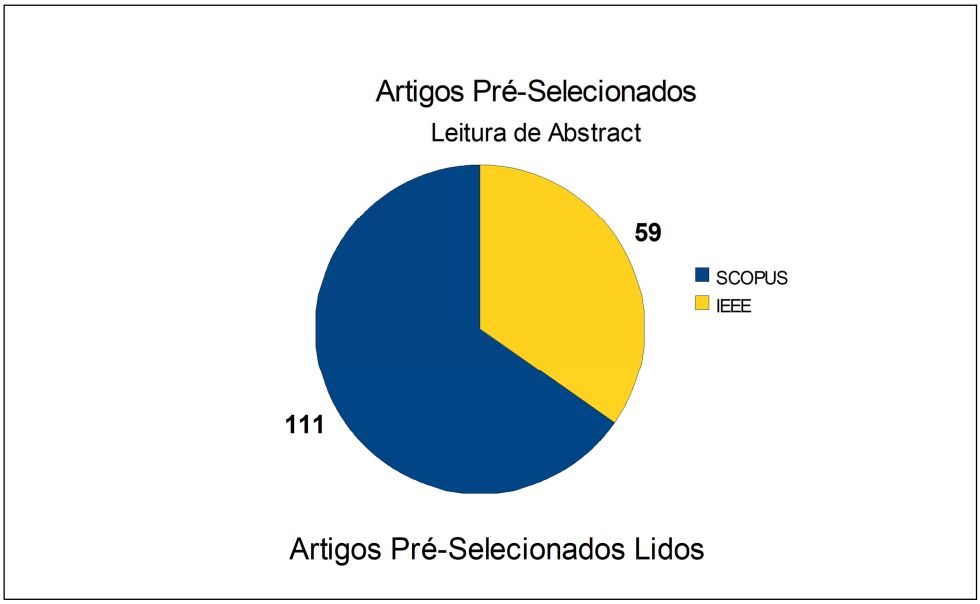

Figura 2- Quantidade de Artigos Pré-Selecionados e Lidos

Avaliação dos Artigos Selecionados

Apenas um artigo foi extraído, o próprio já mencionado como artigo de controle, cujos dados extraídos estão na tabela a seguir:

Título	An Inspection Approach for Conceptual Models on Notations Derived from UML: A Case Study
Autores	Özgür Tanriöver, Semih Bilgen
Fonte	SCOPUS
Tipo de Artigo	Aplicação Industrial
Nome da Abordagem	Não informado
Modelo inspecionado	Diagrama de Atividades KAMA
Inspeção individual?	Sim
Inspeção horizontal?	Sim, Diagrama de Ontologia KAMA
Inspeção vertical?	Sim, Caso de Uso -> DA KAMA -> DSubA KAMA
Tipo de Abordagem	Técnica de Leitura
Foco da Abordagem	Ambos (Sintática e Semântica)
Referência com a descrição	O Próprio
Abordagem já aplicada?	Sim
Tipo de Projeto	Não informado
Tipo de Organização	Não informado

Quanto a avaliação da qualidade do artigo, a pontuação total obtida foi de 3 pontos para um máximo de 4 pontos:

Questão	Conceito
1. Onde se localiza a descrição da abordagem de inspeção?	1
2. O artigo utiliza a terminologia adequada?	0
3. O artigo explicita as restrições e as condições de aplicação da abordagem de inspeção?	1
4. O artigo apresenta algum estudo sobre a abordagem de inspeção?	0,5
5. Os resultados do estudo sobre a abordagem de inspeção são descritos?	0,5

Revisão da Seleção

Os arquivos com as referências e este protocolo foram encaminhados para revisão pelo segundo pesquisador em 22/12/2009 e foram devidamente revisados.

Ameaças à validade do Estudo

Importante observar que a especificidade do tema representa um risco para a pesquisa via *abstract*, uma vez que as abordagens de inspeção, podem ser constituídas por um conjunto de técnicas de inspeção, como no caso de OORTs (Travassos *et al.*, 1999), para a inspeção de artefatos de projeto Orientado a Objetos e no caso de *checklists* como o apresentado em (Sabaliuskaite *et al.*, 2003), que mescla diversos modelos UML. Desta forma, expressões de pesquisa (*search strings*) como “diagrama de atividades”, podem, por exemplo, não constar no resumo de um artigo que trate de uma abordagem de inspeção mais ampla que apenas este diagrama. Uma alternativa seria pesquisar tais *strings* em todo o artigo ao invés de somente no título, *abstract* e palavras-chave, mas observamos que a aplicação de *search strings* para todo conteúdo do artigo é limitada pelas ferramentas de busca devido a existência de artigos que precisam ser comprados para serem lidos, sendo que seu conteúdo completo não fica disponível para pesquisa.

Referências Bibliográficas

(Bock, 2003a) Bock, C. **UML 2 Activity and Action Models- Part 2- Actions**. Journal of Object Technology, Vol. 2, No 4. 2003

(Bock, 2003b) Bock, C. **UML 2 Activity and Action Models**. Journal of Object Technology, Vol. 2, No 4. 2003

(Conradi, 2003) Conradi, R., Mohagheghi, P., Arif, T., Hegde, L.C., Bunde, G.A., Pedersen, A. **Object-Oriented Reading Techniques for Inspection of UML Models – An Industrial Experiment.** LNCS (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2743, pp. 483-500. Springer- Verlag, 2003

(Conte *et al.*, 2005) Conte, T. U., Mendes, E., Travassos, G.H., **Processos de Desenvolvimento para Aplicações Web: Uma Revisão Sistemática**, Proc. XI Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia), Poços de Caldas, Brasil, 2005.

(Elsevier, 2009) **Scopus**, disponível em http://www.scopus.com. Último acesso: maio de 2009

(Gamma *et al.*, 1995) Gamma, E., Helm, H., Johnson, R., Vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, Reading, Massachusetts, 1995.

(IEEE, 1990) IEEE**. IEEE Standard glossary of software engineering terminology, Standard 610.12**. IEEE Press, 1990.

(IEEE, 1998) IEEE. **IEEE Standard 1028- IEEE Standard for Software Review.** Institute of Electrical and Electronics Engineers Inc., New York, 1998

(Machado *et al*, 2005) Machado, R. J., Ramos, I., Fernandes, J. M., “Specification of Requirements Models”. Engineering and Managing Software Requirements, pp 47-68, Springer-Berlin Heidelberg, 2005

(Massollar, 2008) Massollar, J. **Uma abordagem baseada na engenharia para desenvolvimento garantia da qualidade de aplicações web**. Exame de Qualificação. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2008.

(OMG, 2009) OMG. **Business Process Modeling Notation- version 1.2** Disponível em http://www.omg.org/docs/formal/09-01-03.pdf. Último acesso: maio de 2009

(OMG, 2009b) OMG. **OMG Unified Modeling Language (OMG UML), Superstructure- Version 2.2** . Disponível em http://www.omg.org/spec/UML/2.2/Superstructure

(Petersson, 2002) Petersson, H. **Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis**. Technical report 147, Department of Communication Systems, Lund Institute of Technology. Lund University, 2002

(Riehle. D.; Zullighoven, H.,1996) Riehle, D., Zullighoven, H. **Understanding and Using Patterns in Software Development.** Theory and Practice of Object Systems, 2(1):3-13, 1996

(Rocha, 2001) Rocha, A. R. C., Maldonado, J. C., Weber, K. C. **Qualidade de Software- Teoria e Prática**. São Paulo, 1ed., Prentice Hall, 2001

(Sabaliauskaite et al., 2003) Sabaliauskaite, G., Matsukawa, F., Kusumoto, S., Inoue, K. **Further investigations of reading techniques for object-oriented design inspection.** Information and Software Technology 45, pp 571-585. ElSevier Science B. V., 2003

(SEI, 2006) SEI. **CMMI for Development, version 1.2**. Carnegie Mellon University, 2006

(SOFTEX, 2006) Softex. **MPS-BR- Melhoria de Processo de Software Brasileiro- Guia Geral- versão 1.1**. Maio de 2006

(Shull et al., 2003) Shull, F., Carver, J., Travassos, G. H., Maldonado, J. C., Conradi, R., Basili, V. **Replicated Studies: Building a Body of Knowledge about Software Reading Techniques**. Lecture Notes on Empirical Software Engineering, pp 39-84,

World Scientific Publishing Co., Inc., USA, NJ, 2003

(Störrle, 2004) Störrle, H. **Semantics of Control-Flow in UML 2.0 Activities.** Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC'04). IEEE, 2004

(Tanriöver e Bilgen, 2007) Tanriöver, Ö., Bilgen, S. **An Inspection Approach for Conceptual Models in Notations Derived from UML: A Case Study**. ISCIS 2007: 22nd International Symposium on Computer and Information Sciences, 7-9, pp 1-6, 2007

(Travassos et al., 1999) Travassos, G. H., Shull, F., Fredericks, M, Basili, V.R. **Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Increase Software Quality.** Proc. Int'l Conf. -Oriented Programming Systems, Languages & Applications, 1999.

(Travassos et al., 2002) Travassos, G. H., Shull, F., Carver, J., Basili, V. **Reading Techniques for OO Design Inspections**. Technical Report CS-TR-4353, University of Maryland Computer Science Department, 2002. Disponível em: http://www.cs.umd.edu/Library/TRs. Último acesso: fevereiro de 2009

(Travassos, 2008) Travassos, G. H. **Notas de Aula- Disciplina de Engenharia de Software Orientada a Objetos**. Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2008.

(Van Der Aalst, 1999) Van Der Aalst, W. M. P. **Formalization and verification of event-driven process chains.** Information as Software Technology 41, pp 639-650. Elsevier, 1999

(Van Der Aalst et al., 2003) Van Der Aalst, W. M. P., Stoffele, M., Wamelink, J. W. F. **Workflow Patterns**. Distributed and Parallel Databases vol. 14, n.1, pp. 5-51. 2003. Springer Netherlands

(Van Der Aalst; Hofstede, 2005) Van Der Aalst, W.M.P., Ter Hofstede, A.H.M. **YAWL: Yet Another Workflow Language.** Information Systems vol. 30 Issue 4, pp 245-275. Elsevier, 2005.

(Wohed et al., 2005) Wohed, P., Van Der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M., Russel, N. **Pattern-Based Analysis of the Control-Flow Perspective of UML Activity Diagrams**. LNCS 3716, pp. 63-78. Springer- Verlag Berlin Heidelberg, 2005.

(Wohed et al., 2006) Wohed, P., Van Der Aalst, W.M.P., Dumas, M., Ter Hofstede, A.H.M., Russell, N. **On the suitability of BPMN for Business Process Modeling**, LNCS 4102 pp. 161-176. Srpinger-Verlag Berlin Heidelberg, 2006

(Wong, 2006) Wong, Y. K. **Modern Software Review- Techniques and Technologies.** IRM Press, 2006

(Wynn et al., 2009) Wynn, M. T., Verbeek, H. M. W., Van Der Aalst, W. M. P, ter Hofstede, A. H. M., Edmond, D. **Business process verification- finally a reality!** Business Process Management Journal, vol. 15, no. 1, pp. 74-92. Emerald, 2009

(Yourdon, 1992) Yourdon, E. **Análise Estruturada Moderna.** 3 ed. Editora Campus, 1992

APÊNDICE B- CASOS DISCREPANTES

A. Nós de Ação e Fluxo de Ações

Omissão

A01. O rótulo da ação está incompleto, faltando a ação ou o objeto da ação. Ex.: Ações em paralelo: “Recebe conta de água”, “Recebe conta de luz”, “Recebe conta de telefone” → “Verifica contas em atraso” → “Efetua pagamento” (de que?)

A02. Alguma pré ou pós condição deixou de ser foi informada

A03. Alguma ação foi omitida numa sequência de ações

Fato Incorreto

A04. O rótulo da ação refere-se a algum fato que não condiz com o especificado na especificação ou com o conhecimento do domínio. Ex.: “(Bancário) Informa a senha do cliente”, Evento de tempo “Data= 05/09” para ação de “Alertar feriado no dia seguinte”

A05. Alguma pré ou pós condição da ação não condiz com a especificação ou com o conhecimento do domínio. Ex.: “Ter mais de 21 anos” para ação “Votar”

A06. O tempo especificado no aceite de evento (recebimento de sinal) não condiz com o especificado no oráculo ou o conhecimento do domínio

A07. Uma sequência de ações não condiz com a especificação ou com o conhecimento do domínio.

Fato Incorreto, Inconsistência

A08. Algum recebimento de sinal está sendo especificado desnecessariamente, pois nunca será acionado

A09. Existe uma pré ou pós condição que nunca poderá ser alcançada

Inconsistência

A10. O tempo especificado para um evento diverge do especificado em outra parte do modelo

A11. Existe uma sequência cujas ações estão sendo executadas concorrentemente em outra parte do modelo

Ambiguidade

A12. O rótulo da ação e/ou pré/pós condições não estão claras, sendo utilizadas expressões/verbos genéricos ou de duplo sentido. Ex.: “Administra acesso”; “Ser maior de idade”

A13. A relação entre envio/recepção de algum sinal não está clara, devido à descrição ambígua, podendo ser relacionado dois sinais diferentes para um mesmo evento

A14. Uma sequência de ações dá margem a interpretações distintas. Ex.: Recebe ocorrências abertas na data atual → identifica ocorrências vincendas em 7 dias (quais? Todas ou somente as abertas no dia?)

Informação Estranha

A15. O rótulo da ação contém elementos desnecessários, que não dizem respeito à própria ação. Ex.: “Recebe pagamento *após confirmação da operação*”

A16. Uma Pré ou pós condição apresenta justificativas ou detalhes desnecessários. Ex.: “Ter mais de 18 anos*, ou seja, maior de idade*”

A17. Existe uma pré ou pós condição desnecessária, redundante, que já está representada no fluxo das ações

B. Nós de Controle- Início e Fim de Atividade e Fim de Fluxo

Omissão

B01. Um nó de fim de fluxo ou de fim de atividade deixou de ser informado

Fato Incorreto, Inconsistência

B02. Um fim de atividade em determinado momento não condiz com o oráculo ou com o domínio da aplicação. Ex.: Fim de atividade após “Recebe pagamento em dinheiro” (não trata o troco, conforme descrito no oráculo)

B03. Um fim do fluxo leva a uma sequencia de ações absurdas. Ex.: Rescinde contrato → Paga multa rescisória → Fim de Fluxo → Renova contrato

Inconsistência

B04. Um fim de fluxo ou fim de atividade inserido num fluxo contradiz o comportamento de outro fluxo. Ex.: Ação “Solicita Cancelamento da Compra” → “Confirma Cancelamento da Compra” → Fim de atividade; Ação “Solicita Cancelamento da Venda” → Fim de fluxo

Ambiguidade

B05. Um fim de fluxo leva a ações que podem ter dupla interpretação. Ex.: Rescinde contrato → Paga multa rescisória → Fim de Fluxo → Avalia empresa (a empresa cujo contrato foi rescindido ou uma nova empresa?)

C. Nós de Controle- Nós de Bifurcação e Sincronização

Omissão

C01. Alguma ação ou sequência de ações não foi inserida no nó de Bifurcação

C02. Faltou informar *"JoinSpec"*, com a condição necessária para a Sincronização ser concretizado, caso existam condições

C03. Para uma escolha múltipla, falta representar alguma condição de guarda

Fato Incorreto

C04. Uma Sincronização ou Bifurcação especificada no modelo acontece em um momento diferente do especificado no oráculo

C05. O *JoinSpec* não condiz com o oráculo ou com o conhecimento do domínio

C06. As condições de guarda da Bifurcação não condizem com o especificado no oráculo.

C07. Existem sequências em paralelo que não podem ser executadas em paralelo, seja por dependências entre ações ou pela manipulação de objetos em comum

Fato Incorreto, Inconsistência

C09. Faltou conectar alguma ação ou sequência de ações a um nó de sincronização

C08. Existe alguma condição de guarda em uma bifurcação que nunca será alcançada

C10. A escolha tratada como múltipla (Bifurcação) está especificada no oráculo como uma escolha exclusiva ou entende-se pelo conhecimento do domínio que esta escolha deveria ser exclusiva. Ex.: Ao contrário do especificado no modelo, ou o cliente informa CPF (pessoa física) ou informa CNPJ (pessoa jurídica).

Inconsistência

C11. Ações que normalmente são sequenciais em parte do modelo são representadas concorrentemente em outra parte do modelo

Ambiguidade

C12. Um Join*Spec* não descreve claramente a restrição. Ex.: {JoinSpec= desconto <= 0,03} (pode ser interpretado como é 3% ou 0,03% ou R$0,03

C13. Falta clareza em alguma condição de guarda de um Bifurcação

D. Nós de Controle- Nos de Decisão e de *Merge*

Omissão

D01. As condições de guarda não contemplam todas as possibilidades; está faltando alguma condição ou o “senão”

Fato Incorreto, Inconsistência

D02. Alguma condição de guarda contraria a especificação

D03. Existe uma condição de guarda que nunca poderá ser atendida

D04. Falta alguma ação ou sequência de ações ser conectada ao nó de *merge*

D05. Um nó de decisão representa uma escolha que não deveria ser tratada como exclusiva. Ex.: Um “Funcionário” pode possuir tanto atributo “ValorValeRefeição” >0 quanto “ValorValeAlimentação” >0, necessitando receber os dois benefícios

Inconsistência

D06. Não deveria haver decisão/ *merge* em determinada sequência de ações ou o local da decisão/ *merge* não é o adequado

D07. Uma mesma condição de guarda ou semelhante é adotada em decisões distintas com ações divergentes. Ex.: Em uma escolha: Se idade>19, pode entrar; senão “Entrada Proibida”. Em outra escolha: Se idade > 18, pode entrar; senão, solicita autorização de responsável.

Ambiguidade

D08. Falta clareza na condição de guarda. Ex.: “Tempo de trabalho > 5 anos”- tempo de trabalho na empresa ou de carteira assinada?

Informação Estranha

D09. Existe “senão” sem necessidade, em redundância com as demais condições de guarda.

E. Regiões de Interrupção, Regiões de Expansão, Nós de Loop, Nós Condicionais e Exceções

Omissão

E01. Falta alguma ação ser inserida na região de expansão ou de interrupção

E02. Falta algum ação ser inserida no nó protegido

E03. Não foi informada a quantidade de repetições das ações de um nó de loop, embora a quantidade esteja especificada.

Fato Incorreto

E04. Alguma exceção ou interrupção contraria a especificação ou o conhecimento do domínio

E05. O tratamento de uma exceção ou de uma interrupção não condiz com o oráculo ou com o conhecimento do domínio. Ex.: Exceção: valor<0; ação: calcula raiz quadrada do valor.

E06. A quantidade de repetições especificadas num nó de loop não condiz com o oráculo ou com o conhecimento do domínio

E07. A sequência de ações de um nó de loop não está em conformidade com a especificação ou com o conhecimento do domínio.

E08. A sequência de ações de uma região de expansão não está em conformidade com a especificação ou com o conhecimento do domínio.

E09. Um fluxo descrito dentro de um nó condicional não está em conformidade com a especificação ou com o conhecimento do domínio.

Inconsistência

E10. A quantidade de repetições de uma ação diverge em regiões de expansão ou em nós de loop semelhantes.

Ambiguidade

E11. O rótulo de uma exceção não está claro, podendo ser confundida com outra exceção.

E12. O critério para repetição de um nó de loop não está descrito com clareza.

Informação Estranha

E13. O rótulo de uma exceção apresenta informações desnecessárias

F. Nós de Objeto e Fluxo de Objetos

Omissão

F01. Falta especificar algum parâmetro de entrada ou de saída de alguma atividade empacotada

Fato Incorreto, Inconsistência

F02. O objeto deve ser tratado como persistente e não está sendo e vice-versa

F03. O estado de um objeto em determinado momento não é adequado

F04. O limite superior informado para o objeto é inadequado

F05. O critério de ordenação ou de seleção informado para um objeto é inadequado.

F06. A transformação de objetos entre *pins* de entrada e de saída é inadequada

F07. O objeto passado em uma ação não condiz com a necessidade da ação, atividade ou região de expansão envolvida

Inconsistência

F08. Alguma propriedade de um objeto não está sendo respeitada na atividade. Ex.: O objeto utilizado numa região de expansão não pode ser instanciado mais de uma vez ou possui limitação de instanciação pré-definida

F09. O objeto recebe tratamento diferente (seleção, limites ou estados) em fluxos semelhantes. Ex.: Ordenação= LIFO, objeto “Bola Sorteada”, Ação “Apresenta bolas sorteadas”; Ordenação= FIFO, objeto “Bola Sorteada”, Ação “Reapresenta bolas sorteadas”

F10. Um objeto repassado entre ações, atividades ou numa região de expansão contradiz outra situação de utilização deste objeto descrita no modelo. Ex.: “Informa dados do Sócio”-> Sócio.Codigo-> “Informa situação do título”; “Informa dados do Sócio”-> Sócio.Nome-> “Informa situação da mensalidade”.

Ambiguidade, Inconsistência

F11. O nome da classe ou do atributo não está claro. Ex.: “Cadastro”

F12. Os estados utilizados para o objeto são imprecisos ou redundantes. Ex.: “permitido” e “autorizado”

F13. Não está explícita a forma de ordenação de objetos, mas o fluxo sinaliza que a ordenação faz diferença no resultado da atividade. Ex.: Seleciona o milésimo número da lista para ação de premiar

F14. O critério de seleção do objeto não está claro. Ex.: idade > idade permitida

F15. A transformação de um objeto não está clara. Ex.: Usuário.telefone → Linha

Informação Estranha

F16. O rótulo de um objeto apresenta descrições desnecessárias e/ou redundantes sobre o objeto. Ex.: “Casa *selecionada*”; “Requerente *(ou solicitante)*”

F17. Há informações desnecessárias e/ou redundantes numa associação (ordenação, seleção ou limite inferior/ superior). Ex.: <<selection>> empregado.cargo=”analista” AND empregado.cargo <> gerente (segundo o oráculo, o empregado só pode ter um cargo; logo, o segundo critério da seleção é desnecessário)

G. Raias

Fato Incorreto

G01. A raia ou sub-raia da qual faz parte determinado nó de ação contradizem o especificado ou o conhecimento do domínio. Ex.: “(Cliente) Autoriza Desconto”

Inconsistência

G02. Uma Ação ou grupo de ações semelhantes/ idênticas fazem parte de partições distintas. Ex.: “(Médico) Realiza diagnóstico” e “(Auxiliar) Realiza diagnóstico”

Ambiguidade

G03. O rótulo de uma raia ou sub-raia não está claro, não identificando as ações relacionadas. Ex.: “Pessoa”

Informação Estranha

G04. O rótulo de uma raia ou sub-raia contém informações desnecessárias. Ex.: “Professor- *auxiliar ou assistente*”

H. Rastreabilidade Entre Atividades

Omissão

H01. Uma ação da atividade consiste em outra atividade do projeto, mas isto não foi explicitado.

H02. Uma ação foi omitida em um dos modelos que descrevem uma mesma atividade em níveis de detalhe distintos. Ex.: Acorda->*Escova os dentes*->*Toma banho* X Acorda-> Abre os olhos-> Levanta-se-> Abre a porta do banheiro-> *Pega a escova*-> *Toma banho*

Informação Estranha

H03. Alguma atividade referenciada por uma ação não existe no projeto

I. Estereótipos para Descrição de Casos de Uso

Omissão

I01. Alguma ação referente a comportamento interno do sistema não foi referenciada através do estereótipo <<system action>>

I02. Uma regra de negócio especificada, relevante para uma ação, deixou de ser referenciada na atividade.

I03. Uma ação em que o ator interage com o sistema não foi representada através do estereótipo <<actor action>>

I04. Uma ação em que o sistema interage com o ator não foi representada através do estereótipo <<system response>>

I05. Uma ação do ator <<actor action>> não define as informações fornecidas ao sistema

I06. Uma ação que representa uma inclusão não foi representada através do estereótipo <<include>>

I07. Uma ação que representa uma extensão não foi representada através do estereótipo <<extend>>

I08. Uma resposta do sistema <<system response>> não define as informações que são apresentadas ao ator

Fato Incorreto, Inconsistência

I09. Uma condição de guarda imediatamente de uma decisão imediatamente posterior à uma ação do ator (<<ator action>>) não está relacionada à esta ação. Ex.: Ação: “Seleciona forma de pagamento”, condição de guarda: “A1- relatório mensal”

I10. Uma condição de guarda de uma decisão imediatamente posterior a uma ação do sistema (<<system action>>) não está relacionada com o resultado gerado pela ação. Ex.: Ação: “Verifica situação do cliente”, condição de guarda: “A2- crédito aprovado”

I11. Uma condição de guarda imediatamente posterior a uma resposta do sistema (<<system response>>) não está relacionada a uma ação do ator. Ex.: Ação “Gera Relatório Mensal”, condição de guarda: “crédito reprovado”.

I12. Uma ação imediatamente posterior a uma resposta do sistema (<<system response>>) não está relacionada a uma ação do ator. Ex.: Ação “Gera Relatório Mensal”, ação imediatamente posterior: “calcula taxa de juros”.

Inconsistência

I13. Algum estereótipo foi utilizado incorretamente em alguma ação. Ex.: <<Input>> no

lugar de <<output>>, <<extend>> no lugar de <<include>>

I14. Alguma regra de negócio foi referenciada indevidamente na ação. Ex.: Regra de negócio RN05: “O plano de saúde só pode abater até 10% do salário integral do funcionário”, utilizada na ação “Calcula INSS”.

I15. Alguma regra de negócio referenciada na atividade não existe na especificação

I16. Uma ação da atividade está agrupando mais de um passo do caso de uso

Informação Estranha

I17. Uma ação da atividade descreve a implementação do sistema, o que é inadequado para a descrição de um caso de uso

APÊNDICE C- ARTEFATOS DE ACTCHECK (VERSÃO 1)

C.1 Checklist A

#	Item de Avaliação	Resposta					
1	Todos os rótulos das ações da atividade informam qual é a ação e qual é o objeto de cada ação?	Sim		Não		N.A.	
2	Todos os nós de loop esclarecem os critérios para sua repetição?	Sim		Não		N.A.	
3	As ações e condições da atividade estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
4	As ações temporais (sinal baseado em condição temporal) da atividade estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
5	Os critérios para repetição dos nós de loop estão claramente descritos?	Sim		Não		N.A.	
6	As ações e condições (pré e pós condições locais, condições de guarda, JoinSpecs, exceções, interrupções, critérios de teste dos nós condicionais) da atividade foram descritas com clareza?	Sim		Não		N.A.	
7	Os critérios para repetição dos nós de loop estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
8	Alguma ação ou condição da atividade contém informação desnecessária?	Sim		Não		N.A.	
9	Alguma ação ou condição deixou de ser inserida na atividade, numa região ou num nó protegido?	Sim		Não		N.A.	
10	Algum nó de fim de atividade ou de fim de fluxo deixou de ser informado?	Sim		Não		N.A.	
11	As regiões de expansão, de interrupção, nós de loop, nós condicionais e as regiões protegidas (exceções) descrevem suas ações de maneira coerente com a especificação?	Sim		Não		N.A.	
12	Cada fluxo da atividade está em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
13	Existe alguma decisão (escolha exclusiva) que deve ser tratada como escolha múltipla (bifurcação) para representar corretamente a descrição dos requisitos?	Sim		Não		N.A.	
14	Existe alguma escolha múltipla (bifurcação) que deve ser representada como uma escolha exclusiva (decisão) para representar corretamente a descrição dos requisitos?	Sim		Não		N.A.	
15	Cada fluxo da atividade permite interpretar claramente a atividade?	Sim		Não		N.A.	

#	Item de Avaliação	Resposta					
16	Algum objeto externo utilizado/ retornado por um atividade empacotada ou região não está sendo informado como parâmetro desta atividade ou região?	Sim		Não		N.A.	
17	Todas as transformações entre objetos estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
18	Os objetos de entrada e de saída de cada ação, região de expansão ou subatividade estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
19	Os objetos da atividade e suas respectivas propriedades estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
20	Os objetos da atividade e suas respectivas propriedades (critérios de ordenação, estados, critérios de seleção, limites, persistência, isMultiCast, isMultiReceive, isControlType) estão claramente descritos?	Sim		Não		N.A.	
21	Todas as transformações entre objetos estão claramente descritas?	Sim		Não		N.A.	
22	Algum objeto da atividade, propriedade de objeto ou transformação de objeto apresenta informação desnecessária/irrelevante para o contexto da atividade?	Sim		Não		N.A.	
23	Os fluxos agrupados em cada raia/ sub-raia da atividade estão em conformidade com o especificado nos requisitos?	Sim		Não		N.A.	
24	As raias e sub-raias da atividade estão claramente descritas?	Sim		Não		N.A.	
25	Alguma raia ou sub-raia da atividade apresenta alguma informação desnecessária/irrelevante ao contexto da atividade?	Sim		Não		N.A.	
26	Alguma regra de negócio relevante para o contexto de uma determinada ação da atividade e especificada nos requisitos deixou de ser referenciada na atividade?	Sim		Não		N.A.	
27	Os relacionamentos de inclusão e extensão estão devidamente referenciados no modelo através dos estereótipos <<include>> e <<extend>>?	Sim		Não		N.A.	
28	As ações referentes ao comportamento interno do sistema estão devidamente representadas através do estereótipo <<system action>>?	Sim		Não		N.A.	
29	Todas as ações em que o ator interage com o sistema e o sistema com o ator estão representadas, respectivamente, através dos estereótipos <<actor action>> e <<system response>>?	Sim		Não		N.A.	
30	Nas ações do ator (<<actor action>>) estão definidas as informações que este fornece ao sistema?	Sim		Não		N.A.	
31	Nas respostas do sistema (<<system response>>), estão definidas as informações que são apresentadas ao ator?	Sim		Não		N.A.	
32	As condições de guarda que precedem uma ação do ator (<<actor action>>) estão relacionadas com as informações fornecidas nessa ação?	Sim		Não		N.A.	

#	Item de Avaliação	Resposta					
33	As condições de guarda que precedem uma ação do sistema (<<system action>>) estão relacionadas com os resultados gerados nessa ação?	Sim		Não		N.A.	
34	Cada condição de guarda ou a ação que precede uma resposta do sistema <<system response>> está relacionada a uma ação do ator?	Sim		Não		N.A.	
35	Toda ação da atividade está descrita num nível de abstração adequado para a descrição de um caso de uso?	Sim		Não		N.A.	
36	Alguma ação adota uma regra de negócio que não tem a ver com seu escopo?	Sim		Não		N.A.	
37	As regras de negócio referenciadas na atividade existem na especificação?	Sim		Não		N.A.	
38	Algum estereótipo está sendo indevidamente utilizado ao longo da atividade?	Sim		Não		N.A.	
39	Alguma ação da atividade detalha parte da solução computacional (como é feito)?	Sim		Não		N.A.	

C.2 Checklist B

#	Item de Avaliação	Resposta					
1	As ações e condições da atividade estão consistentes entre si?	Sim		Não		N.A.	
2	Os fluxos da atividade estão consistentes entre si?	Sim		Não		N.A.	
3	Existem ações sendo executadas concorrentemente, mas que são executadas sequencialmente em outras partes do modelo?	Sim		Não		N.A.	
4	Existem ações semelhantes/ idênticas sendo executadas por raias ou sub-raias distintas?	Sim		Não		N.A.	
5	Os critérios das regiões de expansão da atividade e dos nós de loop estão consistentes entre si?	Sim		Não		N.A.	
6	A manipulação dos objetos está consistente ao longo da atividade?	Sim		Não		N.A.	
7	As propriedades dos objetos estão sendo respeitadas ao longo da atividade?	Sim		Não		N.A.	
8	As relações de envio x recebimento de sinal da atividade estão representadas com clareza?	Sim		Não		N.A.	
9	De acordo com o seu conhecimento do domínio, todo fluxo em paralelo da atividade é viável?	Sim		Não		N.A.	
10	De acordo com o seu conhecimento do domínio, todas as condições da Atividade são viáveis?	Sim		Não		N.A.	
12	Alguma ação corresponde a outra atividade do projeto, mas esta referência não está explícita no Diagrama de Atividades?	Sim		Não		N.A.	
13	É possível identificar cada fluxo da atividade em outro Diagrama de Atividades do projeto representando a mesma atividade num nível maior de detalhamento?	Sim		Não		N.A.	
14	Todas as outras atividades referenciadas nas ações do Diagrama de Atividades existem no projeto?	Sim		Não		N.A.	

C.3 Questionário de Caracterização da Aplicação

ActCheck

Questionário de Caracterização da Aplicação

versão 3

Data: ___/___/____ Projeto:_____

Nome do Desenvolvedor: _____

1) Quais dos seguintes recursos do Diagrama de Atividades estão previstos na técnica de modelagem adotada na especificação de requisitos?

( ) I- Nó de decisão/ merge
( ) II- Nó de bifurcação/ sincronização
( ) III- Raias (*swimlanes*)
( ) IV- Nós de objeto
( ) V- Grupos de Atividades
( ) VI- Interrupções e Exceções
( ) VII- Regiões de Expansão, Nós de Loop e Nós Condicionais
( ) VIII- Ações de Envio e Recebimento de Sinal
( ) IX- Chamadas de atividade
( ) X- Estereótipos para descrição de casos de uso (Abordagem para desenvolvimento de Aplicações Web)
( ) Outros: _____

2) Existe mais de um diagrama de atividades na especificação de requisitos?

( ) Sim
( ) Não

3) A técnica de modelagem adotada no projeto permite que os diagramas de atividades da especificação de requisitos representem uma mesma atividade em diferentes níveis de detalhamento ?

( ) Sim
( ) Não

4) Algum inspetor possuirá conhecimento do domínio do problema?

( ) Sim
( ) Não

C.4 Tabela de Configuração- Checklist A

#	Resposta Esperada	Casos Discrepantes Relacionados	Classes de Defeito Relacionadas	Item do Questionário de Caracterização da Aplicação
1	**Sim**	**A01**	**Omissão**	**-**
2	Sim	E03	Omissão	1) VII
3	**Sim**	**A04, A05, C05, C06, C07, D02, E04**	**Fato Incorreto**	**-**
4	Sim	A06	Fato Incorreto	1) VIII
5	Sim	E12	Ambiguidade	1) VII
6	**Sim**	**A12, C12, C13, D08, E11**	**Ambiguidade**	**-**
7	Sim	E06	Fato Incorreto	1) VII
8	**Não**	**A15, A16, A17, D09, E13**	**Informação Estranha**	**-**
9	**Não**	**A02, A03, C01, C02, C03, D01, E01, E02**	**Omissão**	**-**
10	**Não**	**B01**	**Omissão**	**-**
11	Sim	E05, E07, E08, E09	Fato Incorreto	1) VI, VII
12	**Sim**	**A07, B02, B03, C04, C09, D04, D06, E05**	**Fato Incorreto, Inconsistência**	**-**
13	Não	D05	Fato Incorreto, Inconsistência	1) I e II
14	Não	C08	Fato Incorreto, Inconsistência	1) I e II
15	**Sim**	**A14, B05**	**Ambiguidade**	**-**
16	Não	F01	Omissão	1) IV e (V ou VI ou VII)
17	Sim	F06	Fato Incorreto, Inconsistência	1) IV
18	Sim	F07	Fato Incorreto, Inconsistência	1) IV

#	Resposta Esperada	Casos Discrepantes Relacionados	Classes de Defeito Relacionadas	Item do Questionário de Caracterização da Aplicação
19	Sim	F02, F03, F04, F05	Fato Incorreto, Inconsistência	1) IV
20	Sim	F11, F12, F13, F14	Ambiguidade, Inconsistência	1) IV
21	Sim	F15	Ambiguidade, Inconsistência	1) IV
22	Não	F16, F17	Informação Estranha	1) IV
23	Sim	G01	Fato Incorreto	1) III
24	Sim	G03	Ambiguidade	1) III
25	Não	G04	Informação Estranha	1) III
26	Não	I02	Omissão	1) X
27	Sim	I06, I07	Omissão	1) X
28	Sim	I01	Omissão	1) X
29	Sim	I03, I04	Omissão	1) X
30	Sim	I05	Omissão	1) X
31	Sim	I08	Omissão	1) X
32	Sim	I09	Fato Incorreto, Inconsistência	1) X
33	Sim	I10	Fato Incorreto, Inconsistência	1) X
34	Sim	I11, I12	Fato Incorreto, Inconsistência	1) X
35	Sim	I16	Inconsistência	1) X
36	Sim	I14	Inconsistência	1) X
37	Sim	I15	Inconsistência	1) X
38	Não	I13	Inconsistência	1) X
39	Não	I17	Informação Estranha	1) X

C.5 Tabela de Configuração- Checklist B

#	Resposta Esperada	Casos Discrepantes Relacionados	Classes de Defeito Relacionadas	Item do Questionário de Caracterização da Aplicação
1	**Sim**	**A10, D07**	**Inconsistência**	-
2	**Sim**	**A11, B04**	**Inconsistência**	-
3	Não	A11, C11	Inconsistência	1) II
4	Não	G02	Inconsistência	1) III
5	Sim	E10	Inconsistência	1) VII
6	Sim	F09, F10	Inconsistência	1) IV
7	Sim	F08	Inconsistência	1) IV
8	Sim	A13	Ambiguidade	1) VIII
9	Sim	C10	Fato Incorreto, Inconsistência	1) II e 4) Sim
10	Sim	A09, D03	Fato Incorreto, Inconsistência	4) Sim
11	Sim	A08	Fato Incorreto, Inconsistência	1) VIII e 4) Sim
12	Não	H01	Omissão	1) IX e 2) Sim
13	Sim	H02	Omissão	1) IX e 2) Sim
14	Sim	H03	Informação Estranha	2) Sim e 3) Sim

C.6 Relatório de Discrepâncias

ActCheck- Relatório de Discrepâncias

Data: ____/____/____ **Tempo total dedicado à inspeção: ____ minutos**

Nome do Inspetor: _____

Projeto: _____ Atividade:_____

#	# Item de Avaliação Relacionado	Tipo de Defeito	Descrição	Localização	Locais onde se repete
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

APÊNDICE D- PLANEJAMENTO DO ESTUDO EXPERIMENTAL

D.1 Definição

D.1.1 Identificação

- Título: Estudo de viabilidade de ActCheck para apoiar a inspeção de Diagramas de Atividades na especificação de requisitos
- Tema: Inspeção de Software
- Área Técnica: Engenharia de Software
- Autores: Rafael Maiani de Mello, Jobson Massollar e Guilherme Horta Travassos
- Afiliação: Programa de Engenharia de Sistemas de Computação COPPE/UFRJ
- Local: COPPE/UFRJ
- Data: Novembro-Dezembro/2010

D.1.2 Caracterização

- Tipo: *quasi*-experimento
- Domínio: Inspeção de Software
- Linguagem: português
- Parceiros: não há
- Links: http://lens-ese.cos.ufrj.br/ese/
- Realização estimada: identificar a viabilidade da técnica ActCheck como técnica para inspeção de diagramas de atividades na especificação de requisitos de projetos de aplicações reais.

D.2 Planejamento

D.2.1 Definição do Estudo Experimental

Objeto de Estudo

Casos de uso extraídos de um projeto real de uma aplicação financeira.

Objetivo Geral do Estudo

Avaliar os efeitos da aplicação de ActCheck em um projeto de aplicação real, comparando os resultados com os de inspeções *ad hoc*

Metas Específicas

- Analisar: os efeitos da aplicação de ActCheck
- Com o propósito de: caracterizar
- Em relação à: sua viabilidade (opinião dos participantes, eficácia e eficiência na identificação de defeitos)
- Do ponto de vista de: estudantes de mestrado e doutorado em Engenharia de Software
- No contexto de: um projeto de aplicação real

Foco em Qualidade

- Cada participante do experimento (inspetor) deverá assinar um termo de consentimento;
- Os inspetores serão organizados em grupos conforme o perfil identificado através do formulário de caracterização do inspetor;
- Para terem sua participação validada, todos os inspetores deverão comparecer aos treinamentos a que forem convocados e entregar as tarefas dentro dos prazos previamente estipulados.

Contexto

Um projeto de aplicação real (Sistema MFT), cuja especificação de requisitos contém diagramas de atividades que descrevem um caso de uso cada, conforme técnica descrita em [Massollar, 2008]

Questões e Métricas

- ActCheck é viável como técnica de inspeção?
  Métricas: *quantidade de defeitos.*
- A aplicação de ActCheck auxilia na redução de falsos positivos?
  Métricas: *quantidade de defeitos, quantidade de discrepâncias, defeitos/discrepâncias***.**
- O tempo utilizado na inspeção com ActCheck é bem aplicado?
  Métricas: *tempo dedicado à inspeção, eficiência da inspeção (defeitos/hora).*
- Uma inspeção aplicando ActCheck é mais eficaz do que uma inspeção *ad hoc*?
  Métricas: *quantidade de defeitos, quantidade total de defeitos conhecidos, taxa de defeitos identificados (defeitos/defeitos conhecidos)*

Questões que não poderão ser respondidas pelo Estudo Experimental

- ActCheck é viável para projetos em contextos diferentes do contexto em que o estudo foi aplicado?

Questões abertas

- ActCheck estimula os inspetores à sua aplicação?

D.2.2 Planejamento Detalhado

Hipóteses

- H01: Não há diferença entre a eficiência de inspeções de diagramas de atividades que aplicam ActCheck e a eficiência de inspeções *ad hoc*.
- HA1: A eficiência de inspeções de diagramas de atividades que aplicam ActCheck é maior que a eficiência de inspeções *ad hoc.*

- H02: Não há diferença entre a eficácia de inspeções de diagramas de atividades que aplicam ActCheck e a eficácia de inspeções *ad hoc*.
- HA2: A eficácia de inspeções de diagramas de atividades que aplicam ActCheck é maior que a eficácia de inspeções *ad hoc.*

Variáveis

- Dependentes: Quantidade de defeitos, quantidade de falsos positivos, tempo dedicado a uma inspeção, eficiência, eficácia, compreensibilidade e aplicabilidade percebida na aplicação de ActCheck, aplicabilidade percebida na aplicação de ActCheck para detecção de defeitos

- Independentes: Tipo de especificação de requisitos- descrição de casos de uso através de diagramas de atividades; Experiência dos inspetores; Conhecimento prévio pelos inspetores do domínio do problema

Seleção dos participantes

- Critérios para seleção: alunos de pós-graduação (mestrado e doutorado) da disciplina de VV&T (Verificação, Validação e Testes) do Programa de Engenhara em Sistemas da Computação da COPPE/UFRJ
- Critérios para seleção dos grupos:
    - Quantidade mínima de dois participantes por grupo;
    - Quantidade mínima de 4 grupos, ou múltiplos de 4;
    - Cada grupo comportará os participantes com níveis de experiência mais próximos entre si. Ex.: (Grupo A: participantes mais experientes, Grupo D: participantes menos experientes)
    - A experiência dos participantes será identificada conforme as suas respostas aos itens do questionário de caracterização do inspetor.

Projeto Experimental

O experimento contou com 8 participantes, que foram distribuídos entre os seguintes grupos:

- Grupo A- os dois participantes mais experientes;
- Grupo B- os dois participantes com experiência intermediária mais próxima

ao grupo A;

- Grupo C- os dois participantes com experiência intermediária mais distante do grupo A;
- Grupo D- os dois participantes com menos experiência dentre os demais participantes.

Cada participante deveria realizar duas inspeções *ad hoc* e duas inspeções utilizando ActCheck

i. *Treatment*: Aplicação de ActCheck em projetos reais de Engenharia de Software
ii. Objeto: Projeto de aplicação real- sistema MFI
iii. Métricas:
    - FP: Quantidade de falsos positivos de uma inspeção individual (inteiro)
    - D: Quantidade de defeitos detectados numa inspeção individual (inteiro)
    - DC: Defeitos conhecidos de um artefato (inteiro)
    - T: Tempo dedicado a uma inspeção individual (inteiro-minutos)
    - E: Eficiência Individual= D/T (racional- defeitos por minuto)
    - Ef: Eficácia Individual= D/DC (racional)

Recursos

- Termo de consentimento
- Formulário de caracterização do inspetor
- Apresentações: Categorias de defeitos, ActCheck (com exemplos), Técnica para descrição de casos de uso através de diagramas de atividades

Instrumentação

Cada participante realizará a inspeção de quatro casos de uso do projeto MFI, sendo que duas inspeções serão realizadas de modo *ad hoc*, numa primeira rodada, e duas inspeções serão realizadas aplicando-se ActCheck (segunda rodada), conforme a seguinte tabela:

Rodada	Grupo A	Grupo B	Grupo C	Grupo D
Rodada 1	*ad hoc*	*ad hoc*	*ad hoc*	*ad hoc*
Rodada 2	ActCheck	ActCheck	ActCheck	ActCheck

Para as rodadas de inspeção foram selecionados 4 casos de uso do projeto MFI (UC006, UC015, UC036 e UC040), descritos através de duas versões cada (UC006C, UC006S, UC015C, UC015S, UC036C, UC036S, UC040C e UC40S), totalizando 8 diagramas de atividades. Para equilibrar a distribuição dos artefatos entre os grupos de participantes, cada caso de uso foi classificado conforme a sua complexidade em duas categorias (complexidade baixa e complexidade alta). Para os grupos B, C e D, os artefatos foram distribuídos de modo que cada participante inspecionasse um caso de uso mais complexo e outro menos complexo a cada rodada. No grupo A, entretanto, cada participante recebeu dois artefatos de menor complexidade numa rodada e dois artefatos de maior complexidade na outra rodada. A distribuição dos artefatos entre as rodadas pode ser visualizada na tabela a seguir.

Grupo	Participante	Primeira Rodada (*ad hoc*)		Segunda Rodada (ActCheck)	
A	P1	UC015S	UC040C	UC006S	UC036C
	P2*	UC006S	UC036C	UC015S	UC040C
B	P3	UC036S	UC040C	UC006S	UC015C
	P4*	UC006S	UC015C	UC036S	UC040C
C	P5	UC036S	UC006C	UC040S	UC015C
	P6*	UC040S	UC015C	UC036S	UC006C
D	P7*	UC015S	UC006C	UC040S	UC036C
	P8	UC040S	UC036C	UC015S	UC006C

Para a segunda rodada, como ActCheck é composto por dois checklists configuráveis, primeiramente foram ajustados os checklists que seriam encaminhados para preenchimentos dos participantes. Como foi identificado que cada grupo possuía um participante com conhecimento ao menos parcial do domínio do problema (marcados com asterisco), foram consideradas os itens de Avaliação de ActCheck dependentes do conhecimento do domínio do problema (Checklist B).

Como se tratam de inspeções, os relatos de discrepâncias dos participantes em cada rodada serão analisados por dois pesquisadores, extraindo-se as discrepâncias. Posteriormente, estes dois inspetores reclassificarão cada discrepância em um defeito ou em um falso positivo. Os pesquisadores identificarão também:

- A categorização mais adequada para cada defeito;

- Os defeitos repetidos detectados pelos participantes para um mesmo artefato numa mesma rodada, obtendo a quantidade de defeitos distintos detectados para cada artefato numa rodada;
- Os defeitos identificados para cada rodada somente nas inspeções *ad hoc* e somente nas inspeções com ActCheck e, consequentemente os defeitos encontrados nas duas rodadas.

A partir de uma entrevista com os participantes, serão coletadas as suas impressões sobre a dificuldade percebida na aplicação de ActCheck, as contribuições que a aplicação técnica trouxe para os participantes no seu aprendizado e na detecção de defeitos, além de também serem coletadas sugestões de melhoria.

Mecanismos de Análise

- Comparação dos resultados totais e por grupo das inspeções *ad hoc* com ActCheck: defeitos, falsos positivos, eficácia, eficiência, tempo.
- Comparação do desempenho de cada participante ao longo das duas rodadas
- Cálculo da variância e do desvio padrão dos defeitos e tempo da inspeção para cada tipo de inspeção aplicado;
- Análise qualitativa da opinião dos participantes sobre ActCheck, orientada à identificação de sua viabilidade como técnica para detecção de defeitos.
- Identificação de vantagens e oportunidades de melhoria de ActCheck, conforme os relatos dos participantes através de uma entrevista conduzida pelos pesquisadores;

Validade dos Resultados

Teste de Hipótese através da Distribuição de Student: média da população e diferenças entre médias (p-value), com alfa=10%

D.2.3 Treinamento

Serão realizados dois treinamentos em datas /fases distintas ao longo da execução do experimento, abrangendo as ementas abaixo:

- T1- diagrama de atividades UML 2.2; técnica para descrição de casos de uso através de diagramas de atividades (Massollar, 2008); categorias de

defeito. Duração: 1 hora;

- T2- ActCheck, com exemplos de defeitos. Duração: 1 hora.

D.2.4 Procedimento de Execução

A execução do Estudo buscará cumprir o cronograma a seguir:

Encontro	Data	Descrição
A	04/11	• Distribuição e recolhimento de: 1. Termo de Consentimento 2. Formulário de Caracterização do Inspetor
Distribuição dos grupos conforme documentação recolhida		
B	11/11	• T1 • Encaminhamento via e-mail de pacote contendo os artefatos para inspeção *ad hoc*
Primeira rodada de inspeção		
-	17/11	• Deadline para recebimento dos relatórios de discrepâncias
C	18/11	• T2 • Encaminhamento via e-mail de pacote contendo os artefatos para inspeção *ad hoc*
Segunda rodada de inspeção		
-	24/11	• Deadline para recebimento dos checklists e relatórios de discrepâncias preenchidos
Análise dos resultados		
D	01/12	• Entrevista com os participantes
Análise dos resultados		

D.2.5 Planejamento de Custos

Por tratar-se de um estudo envolvendo alunos da pós-graduação no contexto de uma disciplina do curso, não estão previstos dispêndios financeiros.

APÊNDICE E- CHECKLIST DE ACTCHECK (VERSÃO 2)

Observe cada ação da atividade individualmente e responda os itens a seguir:

1	Cada ação está compreensível e foi descrita corretamente?	Sim		Não		N. A.	
2	Existe alguma ação que deveria conter mais informações?	Sim		Não		N. A.	
3	Cada ação contém apenas as informações necessárias e suficientes para sua compreensão? Existe alguma informação desnecessária?	Sim		Não		N. A.	
4	Alguma ação deveria ser desmembrada em mais de uma para se adequar ao nível de abstração da atividade?	Sim		Não		N. A.	
5	Alguma ação está descrita num nível de abstração adequado para a descrição de um caso de uso?	Sim		Não		N. A.	
6	Alguma ação descreve parte da solução computacional, ou seja, como é feito?	Sim		Não		N. A.	
7	As ações referentes ao comportamento interno do sistema estão devidamente representadas através do estereótipo <<system action>>?	Sim		Não		N. A.	
8	Todas as ações em que o ator interage com o sistema e o sistema com o ator estão representadas, respectivamente, através dos estereótipos <<actor action>> e <<system response>>?	Sim		Não		N. A.	
9	Nas ações do ator (<<actor action>>) estão definidas as informações que este fornece ao sistema?	Sim		Não		N. A.	
10	Nas respostas do sistema (<<system response>>) estão definidas as informações que são apresentadas ao ator?	Sim		Não		N. A.	
11	As regras de negócio referenciadas nas ações da atividade são compreensíveis e estão corretas?	Sim		Não		N. A.	
12	A ação adota alguma regra de negócio que não tem a ver com seu escopo?	Sim		Não		N. A.	
13	Alguma regra de negócio relevante para o contexto de uma determinada ação da atividade deixou de ser referenciada ou mesmo deveria ser criada?	Sim		Não		N. A.	
14	Todas as pré e pós-condições locais presentes em cada ação estão descritas corretamente?	Sim		Não		N. A.	
15	Existe alguma pré ou pós-condição local que, mesmo estando correta, é impossível de ser atendida?	Sim		Não		N. A.	
16	Todas as pré e pós-condições locais necessárias para a execução de cada ação foram informadas?	Sim		Não		N. A.	
17	Toda ação definida como ação de envio/recebimento de sinal de fato realiza este papel?	Sim		Não		N. A.	

18	É possível identificar a origem de cada sinal recebido pela atividade?	Sim		Não		N. A.	
19	Alguma ação corresponde a uma chamada de outra atividade do projeto, mas a referência não está explícita na atividade? Alguma outra atividade deixou de ser chamada?	Sim		Não		N. A.	
20	Todas os diagramas de atividade referenciados por ações da atividade podem ser identificados?	Sim		Não		N. A.	
21	Cada raia foi descrita com clareza e não contém informações estranhas?	Sim		Não		N. A.	
22	Cada ação da atividade está relacionada às raias adequadas?	Sim		Não		N. A.	

Agora, observe o fluxo de controle do diagrama de atividades, e responda os itens a seguir:

23	Cada sequência entre duas ações permite interpretar claramente a atividade?	Sim		Não		N. A.	
24	Existe alguma sequência de ações de entendimento confuso?	Sim		Não		N. A.	
25	Alguma ação da atividade está sendo executada em momento inadequado?	Sim		Não		N. A.	
26	Alguma ação do sistema <<system action>> deveria estar seguida por uma resposta do sistema <<system response>>?	Sim		Não		N. A.	
27	Toda ação do actor <<actor action>> está sendo seguida de uma ação do sistema <<system action>>?	Sim		Não		N. A.	
28	As condições de guarda de uma decisão imediatamente posterior a uma ação do ator (<<actor action>>) estão relacionadas com as informações fornecidas nessa ação?	Sim		Não		N. A.	
29	As condições de guarda de uma decisão imediatamente posterior a uma ação do sistema (<<system action>>) estão relacionadas com os resultados gerados nessa ação?	Sim		Não		N. A.	
30	Cada condição de guarda ou ação imediatamente posterior a uma resposta do sistema <<system response>> está relacionada a uma ação do ator?	Sim		Não		N. A.	
31	Alguma ação está fora do escopo da atividade ou mesmo deveria fazer parte de outra atividade?	Sim		Não		N. A.	
32	Está faltando inserir alguma ação na atividade?	Sim		Não		N. A.	
33	Algum desvio do tipo decisão (escolha exclusiva) deveria ser tratado como uma escolha múltipla (bifurcação) ou vice-versa?	Sim		Não		N. A.	
34	Existem ações sendo executadas concorrentemente, mas que são executadas sequencialmente em outras partes do modelo?	Sim		Não		N. A.	
35	As condições de guarda, critérios de teste e restrições para junção (joinspecs) da atividade estão corretamente descritas e são realmente necessárias?	Sim		Não		N. A.	

36	Existe alguma condição de guarda, critério de teste ou restrição para junção que, mesmo estando correta, é impossível de ser atendida?	Sim		Não		N. A.	
37	Alguma condição de guarda, critério de teste ou restrição para junção deixou de ser informada?	Sim		Não		N. A.	
38	Considerando-se as condições e desvios de fluxo existentes na atividade, todas as possibilidades de execução da atividade do seu nó inicial até cada nó final estão corretas?	Sim		Não		N. A.	
39	Algum nó de desvio deixou de ser informado? Algum fluxo de desvio deixou de ser inserido na atividade?	Sim		Não		N. A.	
40	Os critérios para repetição dos nós de loop e critérios de regiões de expansão estão claramente descritos e estão corretos?	Sim		Não		N. A.	
41	As exceções e interrupções descritas estão relacionadas a situações condizentes com o escopo da atividade?	Sim		Não		N. A.	
42	As ações da atividade inseridas em regiões de expansão, de interrupção, nós de loop ou nós protegidos deveriam estar realmente inseridas nestes locais?	Sim		Não		N. A.	
43	Faltou inserir alguma ação dentro de uma região de expansão, de interrupção ou nó protegido?	Sim		Não		N. A.	
44	Existem inconsistências entre os componentes da atividade? Por exemplo, alguma sequência de ações contraria outra sequência de ações da atividade ou uma condição de guarda da atividade contraria outra?	Sim		Não		N. A.	
45	Existem ações semelhantes/ idênticas sendo executadas por raias ou sub-raias distintas?	Sim		Não		N. A.	
46	É possível identificar cada fluxo da atividade em outro diagrama de atividades do projeto que representa a mesma atividade em um nível de detalhe maior?	Sim		Não		N. A.	

Agora, observe o fluxo de dados (nós de objeto) da atividade e responda os itens a seguir:

47	Os nós de objeto repassados entre duas ações contribuem para o entendimento da atividade? Todos os objetos descritos podem ser identificados na especificação?	Sim		Não		N. A.	
48	Existe algum objeto na atividade que não aparenta fazer parte do contexto da atividade?	Sim		Não		N. A.	
49	É possível compreender claramente cada propriedade descrita para os objetos da atividade? Estas propriedades estão corretas?	Sim		Não		N. A.	
50	As propriedades dos objetos estão sendo respeitadas ao longo da atividade?	Sim		Não		N. A.	
51	Todas as transformações entre objetos da atividade estão corretas?	Sim		Não		N. A.	