PREVIA: UMA ABORDAGEM PARA A VISUALIZAÇÃO DA EVOLUÇÃO DE MODELOS DE SOFTWARE

Marcelo Schots de Oliveira

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 Engenharia de Sistemas e Computação.

Orientadores: Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

Rio de Janeiro
Fevereiro de 2011

PREVIA: UMA ABORDAGEM PARA A VISUALIZAÇÃO DA EVOLUÇÃO DE MODELOS DE SOFTWARE

Marcelo Schots de Oliveira

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.ª Cláudia Maria Lima Werner, D. Sc.

_____
Prof. Leonardo Gresta Paulino Murta, D. Sc.

_____
Prof. Guilherme Horta Travassos, D. Sc.

_____
Prof. Manoel Gomes de Mendonça Neto, Ph. D.

RIO DE JANEIRO, RJ - BRASIL
FEVEREIRO DE 2011

Oliveira, Marcelo Schots

PREViA: Uma Abordagem para a Visualização da Evolução de Modelos de Software/ Marcelo Schots de Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2011.

XVII, 185 p.: il.; 29,7 cm.

Orientadores: Cláudia Maria Lima Werner

Leonardo Gresta Paulino Murta

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

Referências Bibliográficas: p. 110-123.

1. Evolução de Software. 2. Arquitetura de Software. 3. Visualização de Software. 4. Verificação. I. Werner, Cláudia Maria Lima *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

A meus pais e à minha esposa Natália.

AGRADECIMENTOS

A Deus, pelas graças a mim concedidas durante toda a minha vida, por me permitir fazer de cada contrariedade uma oportunidade de aprendizado, e por ter colocado tantas pessoas especiais em minha vida. À Nossa Senhora, por sempre interceder por mim, e a meu anjo da guarda, pela constante proteção.

À Natália, uma esposa-amiga-mulher fantástica que me apoia em cada passo e que não mede esforços em me auxiliar no que for preciso. Seu amor e compreensão fizeram toda a diferença neste trabalho. Obrigado por me fazer, todo e cada dia, o homem mais feliz do mundo.

A meus pais, Weliton e Maria Helena, que desde sempre me apoiam e incentivam. Obrigado pela educação e formação que recebi de vocês, sem a qual nada disto seria possível. A todos os meus familiares, pela torcida, pelas orações e pelo carinho. A meus amigos, por acreditarem no meu potencial e pelo incentivo.

À minha orientadora Cláudia Werner, pelas oportunidades que me ofereceu durante o mestrado, pela enorme dedicação, pela competência, pelas orientações e principalmente pela paciência. Obrigado por tudo.

Ao meu orientador, Leonardo Murta, pelos inúmeros conselhos, pelas discussões e questionamentos que me levaram (e levam) a fazer um trabalho cada vez melhor, pela experiência compartilhada e pela paciência. Você é um dos principais responsáveis por eu querer me superar a cada dia. Obrigado.

À professora Ana Regina Rocha, pelos ensinamentos em qualidade de software, pelas oportunidades de consultoria que me ajudaram muito a crescer profissionalmente e pela competência e seriedade com que conduz seu trabalho.

Ao professor Guilherme Travassos, pelos valiosos ensinamentos na área de experimentação e de engenharia de software em geral e por ter aceitado o convite para participar desta banca. Ao professor Manoel Mendonça, por ter aceitado o convite para participar desta banca. A ambos, pelas contribuições a este trabalho durante a defesa.

À professora Aline Vasconcelos, por ter participado com contribuições na proposta desta dissertação. Ao colega e amigo Jobson, pelas dicas que foram fundamentais na execução do estudo. Aos professores e amigos Arilo Dias Neto e Márcio Barros, pelas dicas na análise dos dados da avaliação da abordagem.

Aos alunos e colegas que se disponibilizaram a participar da avaliação da abordagem desta dissertação, pelas contribuições que deram a este trabalho.

Ao amigo e meu coorientado Marlon Silva, que trabalhou junto comigo na elaboração e construção dos componentes que contribuíram com o desenvolvimento da abordagem desta dissertação. Obrigado pelo esforço e pela paciência.

À "mãezona" do grupo, Cláudia Susie, pelo apoio e empenho na condução do estudo no contexto de educação em engenharia de software, pela amizade, pelas risadas e pelo carinho. Obrigado por ajudar a me manter perseverante em meus objetivos.

A todos os demais colegas do grupo de reutilização da COPPE/UFRJ: Anderson, Danny, Hua Lin, Rafael Cepêda, Wagner, Rodrigo, João Gustavo, Paula Cibele, Ana Maria, Chessman, Andréa, Vanessa, Marco Eugênio, Henrique e Caio. Ao Nicolas, da UCI/Irvine, pelo apoio e incentivo. Obrigado por terem feito parte desta jornada, cada um de uma forma, e por terem contribuído para meu crescimento como pesquisador.

Aos demais colegas da linha de engenharia de software, em especial ao Wallace, Sílvia, Rafael Maiani, Rafael Espírito Santo, Jobson, Breno, Tayana, Ana Cândida, Marcelo Mello, Elaine, Andrea, Gleison e Mariano. Conviver com vocês foi e é um privilégio.

Às funcionárias do PESC, Taísa, Solange, Mercedes, Sônia, Cláudia e Nathália, pela colaboração nos procedimentos administrativos e pela amizade. À equipe do departamento de registro da COPPE, pela presteza e atenção dedicada.

À professora Mariluci Portes, por ter aberto as portas da UERJ para meus primeiros passos como professor, pela confiança e pelos preciosos conselhos e orientações. Aos professores Vera Werneck e Alexandre Rojas, pelo apoio. A meus alunos, pelo constante feedback e por me ensinarem a ser um professor cada vez melhor. Ao Wagner, pelo incentivo sem o qual nada disto teria acontecido.

Aos revisores anônimos (dos artigos submetidos sobre a abordagem) que contribuíram com críticas construtivas e apontaram direcionamentos importantes para o andamento desta dissertação.

Aos amigos do Centro Universitário da Tijuca e do Centro de Formação da Marquês de Valença, pelos conselhos, amizade e constantes orações.

À CAPES, pelo apoio financeiro durante o mestrado; ao PESC, ao CNPq e à FAPERJ, pelo auxílio financeiro para a apresentação de trabalhos em eventos.

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

PREVIA: UMA ABORDAGEM PARA A VISUALIZAÇÃO DA EVOLUÇÃO DE MODELOS DE SOFTWARE

Marcelo Schots de Oliveira

Fevereiro/2011

Orientadores: Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

Programa: Engenharia de Sistemas e Computação

A complexidade de um sistema de software está, em grande parte, determinada por sua estrutura (i.e., sua arquitetura). No entanto, existe uma deficiência em reconhecer a distinção entre a arquitetura planejada (conceitual) e a implementada (emergente). Desta forma, a falta de percepção (*awareness*) e compreensão dos desvios arquiteturais e da evolução do software pode comprometer tanto o processo de desenvolvimento quanto o produto desenvolvido, prejudicando o equilíbrio entre custo, tempo e qualidade, gerando a insatisfação dos envolvidos.

Este trabalho apresenta a abordagem PREViA (Procedimento para Representar a Evolução por meio da Visualização de Arquiteturas), que visa prover uma melhor percepção da aderência entre o que está sendo implementado com relação ao que foi projetado, bem como prover uma melhor percepção sobre a evolução do projeto e da implementação. Para auxiliar estas atividades, faz-se uso de conceitos e técnicas de visualização de software para fornecer uma melhor representação das informações obtidas. A abordagem PREViA foi avaliada a partir da execução de dois estudos, cujos resultados fornecem evidências positivas quanto ao uso da abordagem em projetos de software, em termos de eficiência, precisão e eficácia.

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

PREVIA: AN APPROACH FOR VISUALIZING THE EVOLUTION OF SOFTWARE MODELS

Marcelo Schots de Oliveira

February/2011

Advisors: Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

Department: Systems and Computing Engineering

The complexity of a software system is largely determined by its structure (i.e., its architecture). However, there is a failure in recognizing the distinction between the planned architecture (conceptual) and implemented one (emerging). Thus, the lack of awareness and understanding of architectural drifts and software evolution may affect both the development process and the developed product, impairing the balance between cost, time and quality, leading to the dissatisfaction of stakeholders.

This work presents the PREViA approach (Procedure for Representing Evolution through Visualization of Architectures), which aims to provide a better understanding of the adherence between what is being implemented with respect to what was designed, as well as providing a better perception of the evolution of design and implementation. In order to assist these activities, concepts and techniques of software visualization are used for providing a better representation of the obtained information. The PREViA approach was evaluated in two studies, which results provide positive evidences for the use of the approach in software projects, in terms of efficiency, accuracy and efficacy.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

CAPÍTULO 1 - INTRODUÇÃO

1.1 Motivação

A teoria e a prática em engenharia de software afirmam que (i) o software evolui continuamente, e precisa ser constantemente modificado em resposta a novos requisitos do usuário ou modificações no ambiente, e (ii) quando o software é modificado, ele se torna mais complexo e complicado (LEHMAN, 1996). Em consequência disto, a manutenibilidade dos sistemas de software decresce no decorrer do tempo, tornando-se cada vez mais difícil realizar qualquer modificação no mesmo (GRAAF, 2007a, 2007b), bem como compreender modificações previamente realizadas.

A complexidade de um sistema de software está, em grande parte, determinada por sua estrutura (i.e., sua arquitetura) (GRAAF, 2007a) e, por isto, esta deve ser bem definida, documentada e atualizada, permitindo o entendimento dos módulos do sistema. No entanto, de acordo com TAYLOR *et al.* (2009), existe uma deficiência em reconhecer a distinção entre a arquitetura planejada e a implementada, fazendo com que os *stakeholders* (envolvidos, ou interessados) se enganem quanto ao que eles acreditam que têm (isto é, sem a percepção do que eles realmente têm), fazendo com que qualquer estratégia de desenvolvimento ou evolução que se baseie na arquitetura documentada (porém imprecisa) esteja fadada ao fracasso.

A falta de percepção (*awareness*) e compreensão dos desvios arquiteturais e da evolução do software podem comprometer tanto o processo de desenvolvimento quanto o produto desenvolvido, prejudicando o equilíbrio entre custo, tempo e qualidade, gerando a insatisfação dos envolvidos. Neste sentido, conceitos e técnicas de visualização de software podem auxiliar estas atividades que envolvem o raciocínio humano, fornecendo uma melhor representação das informações obtidas (BALL & EICK, 1996).

Para que a compreensão da evolução do software e a verificação da aderência entre a arquitetura planejada (conceitual) e a arquitetura implementada (emergente) sejam efetivas, é necessário observar algumas características que são importantes no contexto destas tarefas, conforme observado em trabalhos recentes como (CASERTA & ZENDRA, 2010) e (TELEA *et al.*, 2010). No entanto, as abordagens identificadas na

literatura não atendem simultaneamente a todas estas características, por possuírem focos de visualização diferentes ou por limitações técnicas.

Neste contexto, com o presente trabalho, pretende-se compreender como auxiliar equipes de desenvolvimento em tarefas de verificação de aderência estrutural e compreensão da evolução, permitindo maior interação com o usuário e explorando melhor a dimensão do tempo.

Desta forma, acredita-se que uma abordagem que trate da percepção da evolução de modelos arquiteturais de software (i.e., auxiliando na verificação da aderência e no acompanhamento da evolução) pode beneficiar equipes de desenvolvimento na execução destas tarefas.

1.2 Objetivo

O objetivo desta dissertação é definir uma abordagem que possa (i) prover uma melhor percepção da aderência entre o que está sendo implementado (arquitetura emergente) com relação ao que foi projetado (arquitetura conceitual), (ii) prover uma melhor percepção sobre a evolução da implementação (arquitetura emergente), e (iii) prover uma melhor percepção sobre a evolução do projeto (*design*) do software (arquitetura conceitual).

Para atender a estes objetivos, pode-se fazer uso de conceitos e técnicas de visualização de software, que proporcionam uma forma simples e intuitiva de compreensão dos dados representados (BALL & EICK, 1996), tornando "visíveis" os artefatos e o comportamento do software por meio de representações visuais. Segundo MUKHERJEA & FOLEY (1996), a visualização destes dados é particularmente importante por permitir que as pessoas utilizem o raciocínio perceptivo (em vez do raciocínio cognitivo) na realização das tarefas, o que facilita a detecção de padrões que revelam a estrutura implícita nos dados.

Com base em trabalhos da literatura sobre a visualização da evolução de software (em especial de arquiteturas de software), foram identificadas algumas características importantes neste domínio para auxiliar o desenvolvimento de software. Tais características guiaram a definição da abordagem proposta nesta dissertação, denominada PREViA (Procedimento para Representar a Evolução por meio da Visualização de Arquiteturas).

A abordagem PREViA consiste da composição de mecanismos de comparação de modelos, extração de métricas e visualização de software de forma integrada,

permitindo identificar diferenças entre estes modelos e exibi-las conforme o foco de visualização aplicável à tarefa de desenvolvimento sendo executada. Além da definição da abordagem, foi desenvolvido um protótipo para automatizar parte das tarefas às quais a PREViA pretende dar apoio.

A abordagem proposta nesta dissertação foi avaliada a partir da execução de dois estudos, sendo um deles voltado para a utilização da PREViA no contexto de educação em engenharia de software (visando verificar a aplicabilidade da abordagem neste cenário), e o outro voltado para projetos de software, alinhado ao cenário discutido nesta dissertação. Os resultados do estudo em projetos de software fornecem indícios de que o uso da abordagem PREViA permite executar algumas tarefas comuns no cenário de desenvolvimento de software de forma mais eficiente, precisa e eficaz, quando comparado à execução das mesmas tarefas sem o uso da abordagem.

1.3 Estrutura da Dissertação

Esta dissertação está organizada em seis capítulos. O presente capítulo apresentou a motivação para o desenvolvimento deste trabalho, bem como os objetivos da pesquisa.

O Capítulo 2 trata de arquiteturas, visões arquiteturais e modelos de software, incluindo definições sobre arquitetura conceitual e arquitetura emergente. Alguns fatores que causam e/ou impactam o processo de evolução arquitetural são apresentados, e algumas definições e as principais técnicas utilizadas para comparação de modelos são discutidas.

O Capítulo 3 apresenta alguns conceitos e técnicas de visualização de software, discutindo sua utilização no contexto da evolução arquitetural. Além disso, alguns trabalhos relacionados são analisados.

No Capítulo 4, propõe-se uma abordagem que visa atender aos requisitos identificados no Capítulo 3, voltada para a visualização da evolução de arquiteturas de software com os focos de percepção da aderência e compreensão da evolução. Assim, é apresentada uma visão geral da solução proposta, denominada PREViA (Procedimento para Representar a Evolução por meio da Visualização de Arquiteturas), junto a um exemplo de sua utilização e alguns detalhes de implementação do protótipo da abordagem.

O Capítulo 5 descreve as etapas de definição, planejamento, execução e análise de dois estudos realizados para avaliar a abordagem PREViA, sendo um deles alinhado

ao cenário discutido nesta dissertação, e o outro voltado para a utilização da PREViA no contexto de educação em engenharia de software.

Por fim, o Capítulo 6 contém as considerações finais deste trabalho, bem como as contribuições da dissertação, algumas limitações identificadas e as perspectivas de trabalhos futuros.

CAPÍTULO 2 - ARQUITETURA E MODELOS DE SOFTWARE

2.1 Introdução

O surgimento do conceito de arquitetura de software está relacionado ao desenvolvimento de técnicas de abstração em ciência da computação. De acordo com GARLAN & SHAW (1993), a utilização de símbolos para descrever códigos de máquina, endereços de memória e sequências de instruções caracteriza a forma mais primitiva de abstração. Posteriormente, com o advento de linguagens de programação de alto nível, foi possível a separação entre a especificação e a implementação de módulos do sistema. A criação de tipos abstratos de dados, por sua vez, permitiu que partes do sistema fossem desenvolvidas a partir de um vocabulário de tipos de dados.

Quando os sistemas passaram a ser construídos em larga escala, a organização destes sistemas (em termos de seus diversos componentes interconectados) passou a ser o foco principal da fase de projeto (*design*), o que resultou nos primeiros estudos sobre a arquitetura do software (GARLAN & SHAW, 1993).

A arquitetura de um software envolve a descrição dos elementos a partir dos quais um sistema é construído, as interações entre esses elementos, os padrões que direcionam sua composição e as restrições sobre estes padrões (SHAW & GARLAN, 1996). Em outras palavras, é a estrutura ou organização de um sistema expressa através de seus componentes, os relacionamentos destes componentes (uns com os outros e com o ambiente externo), propriedades externamente visíveis de ambos (componentes e relacionamentos) e os princípios que guiam seu projeto e evolução (BASS *et al.*, 2003; ISO/IEC, 2007; CLEMENTS *et al.*, 2010).

Há diversos relatos na literatura sobre a utilização de arquiteturas de software em diferentes cenários (DRONGOWSKI, 1993; SONI *et al.*, 1995). EELES (2006) relata alguns dos benefícios obtidos a partir desta utilização:

- Tratamento da qualidade do sistema;
- Guia para o consenso;
- Apoio ao processo de planejamento;
- Preservação da integridade dos módulos do sistema;
- Auxílio à gerência da complexidade;

- Base para a reutilização;
- Redução dos custos de manutenção; e
- Apoio à análise de impacto.

Visando descrever a arquitetura do software, é comum a utilização de modelos (também conhecidos por modelos arquiteturais) para representar diferentes visões da arquitetura. Estes modelos podem representar um subconjunto de elementos arquiteturais (decomposição em módulos) ou uma visão específica que contemple apenas determinado tipo de elemento (voltada, por exemplo, para a estrutura ou para o comportamento do software).

Além desta seção introdutória, este capítulo está estruturado em cinco seções. Na Seção 2.2, é apresentado o conceito de visões arquiteturais de software; definições sobre arquitetura conceitual e arquitetura emergente são detalhadas na Seção 2.3. A Seção 2.4 descreve alguns fatores que causam e/ou impactam o processo de evolução arquitetural. Algumas definições e as principais técnicas utilizadas para comparação de modelos são discutidas na Seção 2.5. Por fim, na Seção 2.6, são relatadas as considerações finais deste capítulo.

2.2 Visões Arquiteturais

A necessidade de organizar a descrição de uma arquitetura de software utilizando diversas visões deu origem ao conceito de visões arquiteturais, cujo objetivo é fornecer diferentes representações arquiteturais de um sistema, cada uma sob uma determinada perspectiva, isto é, relacionada a um conjunto específico de interesses (KRUCHTEN, 1995). Um exemplo clássico na engenharia de software é o modelo proposto pelo mesmo autor, que identifica cinco visões arquiteturais básicas (que podem ser descritas em diferentes níveis de abstração):

- A visão lógica descreve o modelo de objetos do projeto do sistema;
- A visão de processos trata do agrupamento de tarefas, envolvendo aspectos de concorrência e sincronização no projeto;
- A visão física representa o mapeamento entre entidades de software e hardware, refletindo os aspectos de distribuição do sistema;
- A visão do desenvolvimento diz respeito à organização estática do software no ambiente de desenvolvimento, isto é, possui foco na organização efetiva ou concreta dos módulos de software; e

- A visão dos cenários consiste de composições de elementos das quatro visões anteriores, selecionados para ilustrar um conjunto de cenários relevantes em que as visões trabalham em conjunto, servindo para dois propósitos principais: (i) descobrir, durante o projeto arquitetural, os elementos que irão compor a arquitetura, e (ii) auxiliar na validação e na representação da arquitetura, após a finalização do projeto desta.

A forma como estas visões se relacionam e o escopo tratado por cada visão estão representados na Figura 2.1.

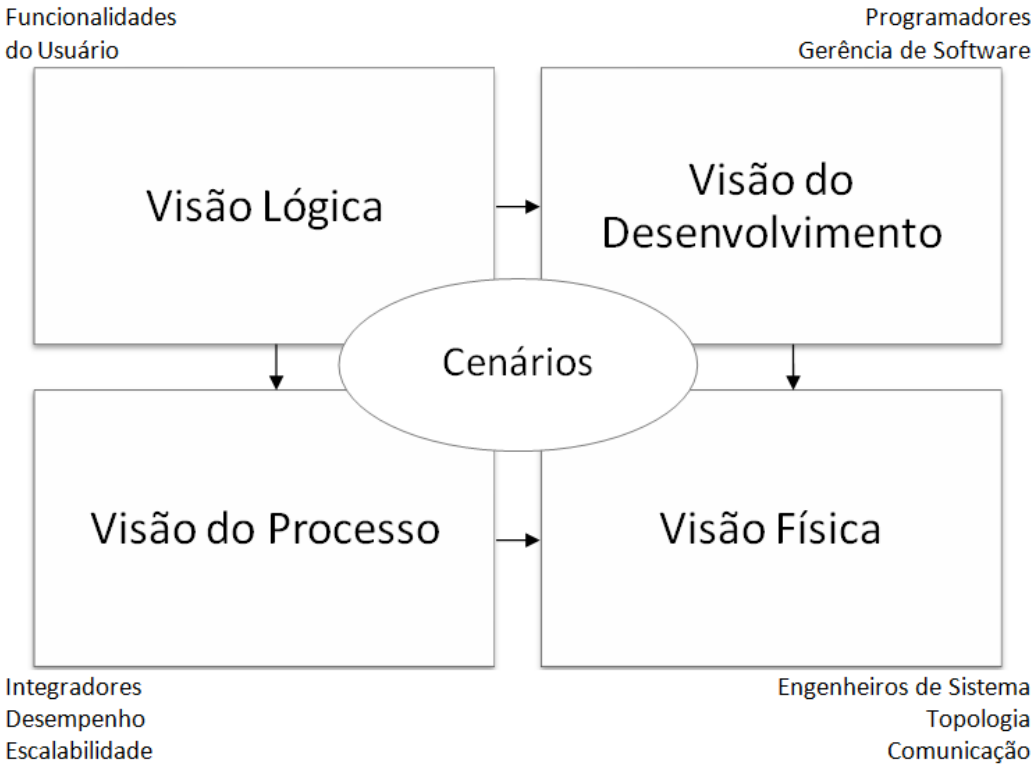


Figura 2.1 – Visão geral do modelo arquitetural "4+1" (KRUCHTEN, 1995)

Outro exemplo de decomposição do sistema em visões arquiteturais é apresentado por MERSON (2005), que prevê que um arquiteto pode considerar o sistema de cinco formas (com a ressalva de que nem todas as visões são relevantes a todos os sistemas):

- A visão de módulos descreve a forma como o sistema está estruturado, em termos de unidades de código;
- A visão de tempo de execução também trata de estrutura do sistema, porém em termos de um conjunto de elementos disponíveis em tempo de execução;

- A visão de implantação consiste da forma como o sistema é implantado em hardware;
- A visão de implementação diz respeito à maneira como os artefatos estão organizados no sistema de arquivos; e
- A visão do modelo de dados representa a estrutura do repositório de dados.

Ao analisar abordagens voltadas para a identificação e definição de visões arquiteturais, NAAB *et al.* (2005) observaram um consenso nos aspectos mais importantes a serem descritos: uma visão conceitual (ou lógica), uma visão dos módulos estáticos e uma visão dinâmica relacionada à execução do sistema.

Embora haja diversas abordagens para a descrição de arquiteturas, tais como as listadas em (SHAW & GARLAN, 1996) e (MEDVIDOVIC & TAYLOR, 2000), muitos trabalhos na literatura fazem uso da notação UML (*Unified Modeling Language*) para a representação arquitetural de sistemas orientados a objetos (AVGERIOU *et al.*, 2005; KÜMMEL, 2007).

O objetivo da UML é fornecer a arquitetos de sistemas, engenheiros de software e desenvolvedores de software ferramentas para análise, projeto e implementação de sistemas baseados em software (OMG, 2009). Por meio desta notação, é possível descrever o projeto arquitetural sob diferentes perspectivas, bem como em diferentes níveis de detalhamento. Além disso, por se tratar de um modelo extensível, é possível incluir elementos adicionais aos modelos por meio da criação de perfis e estereótipos.

A UML proporciona uma forma para a criação, visualização e documentação de projetos arquiteturais de sistemas (ESTIVALETE, 2007), envolvendo aspectos conceituais (e.g., processos de negócio e funções de sistema) e itens concretos (e.g., classes e componentes reutilizáveis) (BOOCH *et al.*, 2005). Seguindo sua notação semiformal, é possível representar diferentes visões arquiteturais por meio de modelos e diagramas UML – um diagrama de classes, por exemplo, pode apoiar na descrição dos módulos de um sistema (MERSON, 2005). A Figura 2.2 exibe um exemplo deste uso.

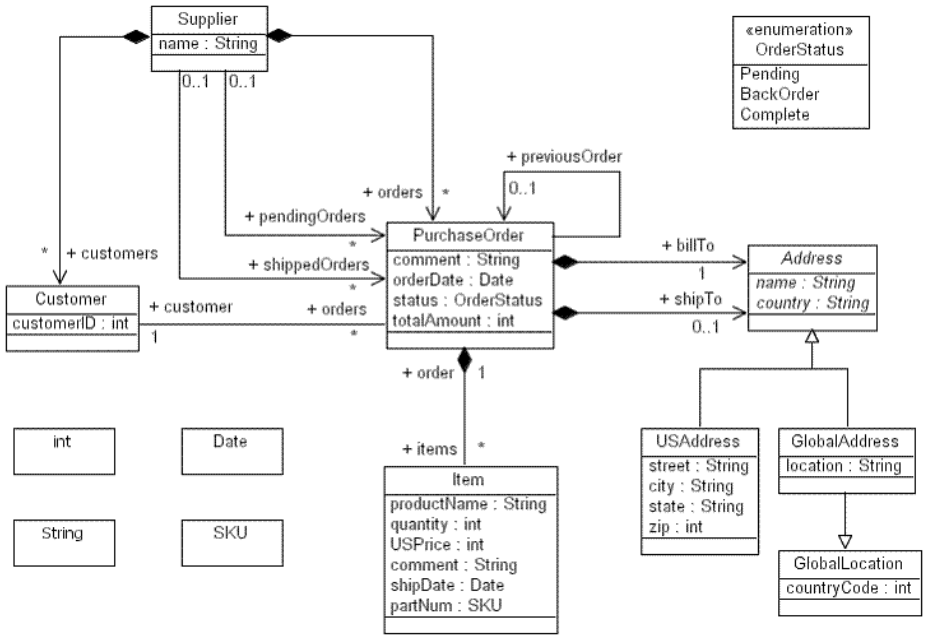


Figura 2.2 – Diagrama de classes: visão dos módulos de um sistema (IBM CORP., 2008)

A UML se tornou rapidamente um padrão utilizado na indústria de software. Seu uso em larga escala é um dos fatores que impulsionaram sua utilização para efetuar descrições arquiteturais. Dentre as razões para tal abrangência, pode-se citar a padronização (e constante manutenção) da infraestrutura da UML e a facilidade de acesso para um amplo grupo dos *stakeholders* (CULBERTSON, 2005).

Uma das deficiências apontadas por SHAW & CLEMENTS (2006) é a ausência de uma suíte robusta de ferramentas para tarefas como análise, verificação de consistência ou outros métodos para conectar, de forma automática, a informação expressa em modelos UML com o código fonte do sistema.

2.3 Arquitetura Conceitual e Arquitetura Emergente

Alguns trabalhos na literatura (como os apresentados a seguir) visam definir arquitetura conceitual e arquitetura emergente; no entanto, apesar de haver muitos aspectos semelhantes, existe uma variação no que diz respeito aos nomes dados a tais arquiteturas, bem como à definição dada pelos autores.

Uma das técnicas pioneiras no tratamento do desvio entre artefatos de projeto (*design*) e implementação é a de modelos de reflexão de software (*software reflexion model*), desenvolvida por MURPHY *et al.* (2001) com o objetivo de auxiliar

engenheiros de software na execução de suas tarefas, permitindo explorar (ao invés de simplesmente preencher) a lacuna existente entre modelos de alto nível e modelos de implementação extraídos do código fonte. A abordagem requer que sejam informados um modelo de alto nível e um modelo de implementação (obtido por meio de alguma ferramenta específica para este fim), solicitando também que o engenheiro de software defina manualmente o mapeamento entre estes modelos. A partir disto, é construído o modelo de reflexão, que exibe as convergências (ou seja, interações entre os modelos que ocorreram conforme o esperado), divergências (interações não esperadas) e ausências (interações esperadas, mas não encontradas) identificadas.

O trabalho de VÖLTER (2007) define a arquitetura conceitual (*conceptual architecture*) como um conjunto limitado de conceitos (tais como padrões e estilos arquiteturais) e os relacionamentos existentes entre eles; a instanciação desses conceitos utilizados para desenvolver uma aplicação em particular caracteriza a arquitetura de aplicação (*application architecture*).

Já ANDERSON *et al.* (2000) definem que a arquitetura conceitual (*conceptual architecture*) é destinada a capturar uma visão abstrata do sistema, cujos componentes correspondem a características da interface com o usuário ou a funcionalidades principais, enquanto a arquitetura da implementação (*implementation architecture*) é a concretização da arquitetura conceitual, na qual questões como distribuição, protocolos de rede, utilização de *cache*, replicação e controle de consistência deverão ser tratadas. Os autores ainda acrescentam que existem diversas arquiteturas de implementação possíveis para uma dada arquitetura conceitual.

Sob a perspectiva de TAYLOR *et al.* (2009), a arquitetura prescritiva (*prescriptive architecture*) captura as decisões de projeto (*design*) de software tomadas antes da construção do sistema, ou seja, é a arquitetura conforme foi concebida ou pretendida. Por sua vez, a arquitetura descritiva (*descriptive architecture*) mostra como o sistema foi construído, sendo referida como a arquitetura conforme foi implementada ou realizada.

No presente trabalho, assim como na definição em (TAYLOR *et al.*, 2009), objetiva-se diferenciar a arquitetura conceitual como uma arquitetura prospectiva – que determina como deverá ser o software – da arquitetura emergente como uma arquitetura retrospectiva – que representa como está o software. Neste sentido, a definição de KNODEL *et al.* (2006a) é a que melhor reflete estas ideias. Nela, a arquitetura planejada, ou pretendida, é descrita pelo modelo arquitetural (*architectural model*), e a

arquitetura real, ou implementada, é representada pelo modelo do código fonte (*source code model*). Esta definição é utilizada no decorrer deste trabalho para descrever, respectivamente, as arquiteturas conceitual e emergente.

A criação de arquiteturas conceituais permite uma avaliação antecipada do impacto das decisões de projeto na implementação (ANDERSON *et al.*, 2000). Decisões que são tomadas em nível arquitetural são consideradas como o primeiro conjunto de decisões de projeto relacionadas aos requisitos funcionais e de qualidade, podendo interferir (positiva ou negativamente) no atendimento a estes requisitos e aos objetivos de negócio (BABAR *et al.*, 2004; KNODEL *et al.*, 2006a).

A arquitetura conceitual de um software não deve conter qualquer detalhe de implementação, e deve permanecer independente de decisões específicas de tecnologia, utilizando apenas especificações dos componentes (ou, de forma mais genérica, dos elementos arquiteturais) para representar os módulos e serviços existentes na arquitetura (D'SOUZA & WILLS, 1999; KÜMMEL, 2007). Isto permite sua reutilização em contextos semelhantes ao qual tal arquitetura foi criada.

2.4 Evolução de Arquiteturas de Software

Na medida em que mudanças ocorrem no ambiente e surgem demandas por novas funcionalidades ao longo do ciclo de vida do software, tornam-se necessárias alterações nos requisitos do software; tais alterações, por sua vez, implicam muitas vezes em mudanças nas descrições arquiteturais que representam estes requisitos (KÜMMEL, 2007).

Em um cenário ideal, ao evoluir um sistema, as modificações são feitas primeiramente sobre a arquitetura elaborada na fase de projeto. Neste cenário, em qualquer instante do tempo de desenvolvimento, todos os elementos estruturais presentes em uma versão da arquitetura encontram-se implementados na versão correspondente do código fonte. Além disso, os elementos implementados não contêm conexões que não estejam descritas na arquitetura, e são poucos os elementos que não possuem correspondência a um elemento arquitetural (TAYLOR *et al.*, 2009).

Sabe-se, no entanto, que esta visão é bastante restritiva e pouco realista. A aplicação implementada tem uma arquitetura própria, que, ao contrário da que está documentada, é latente. Na prática, o sistema costuma ser modificado diretamente, devido a fatores como (TAYLOR *et al.*, 2009):

- Negligência por parte dos desenvolvedores;

- Prazos curtos que impedem o planejamento e a execução da documentação;
- Falta de documentação da arquitetura conceitual;
- Necessidade ou vontade de efetuar otimizações no código; e
- Técnicas e ferramentas de apoio inadequadas.

Situações como estas podem fazer com que a arquitetura conceitual, ou seja, a arquitetura planejada na fase de projeto do software, comece a divergir da arquitetura emergente, que é a arquitetura que representa o que está implementado em código fonte (obtida a partir de artefatos de implementação, como o código fonte ou algum estágio intermediário entre o código fonte e a aplicação, como o *bytecode*). Essa divergência pode representar, por exemplo, elementos da arquitetura conceitual que ainda não foram implementados, ou mesmo elementos implementados – presentes na arquitetura emergente – que não foram descritos na arquitetura conceitual.

No entanto, sabe-se que é importante manter a consistência entre as aplicações desenvolvidas e sua arquitetura conceitual em qualquer instante do tempo (VÖLTER, 2007). Neste sentido, a verificação da aderência entre as arquiteturas pode evitar posteriores impactos negativos e auxiliar no processo de tomada de decisões em nível de projeto e implementação, especialmente no que tange a processos de manutenção e reengenharia. Nestes dois processos, a arquitetura conceitual desempenha um papel crucial, visto que ela possui uma descrição em alto nível do sistema implementado, podendo assim ser utilizada para reconstruir o software (em outra linguagem de programação, por exemplo) e também para permitir a compreensão do software passível de manutenção.

Algumas normas – tais como a ISO/IEC 12207 (ISO/IEC, 2008) – e modelos de maturidade enfatizam a necessidade de verificação de aderência. Por exemplo, o processo Solução Técnica (*Technical Solution – TS*) do nível 3 de maturidade do CMMI-DEV (CMMI PRODUCT TEAM, 2006) inclui um Objetivo Específico (*Specific Goal - SG*) que diz respeito à implementação do projeto (*design*) (SG 3), apoiada por duas Práticas Específicas (*Specific Practices - SP*) no que tange a manter consistentes e atualizados os produtos de trabalho (a saber: SP 3.1 e SP 3.2). Já no nível D do MR-MPS (SOFTEX, 2009), o processo Projeto e Construção do Produto (PCP) possui dois resultados esperados (PCP 6 e PCP 8) que também requerem que o produto seja implementado e verificado de acordo com o que foi especificado na etapa de projeto (*design*), mantendo esta especificação atualizada para garantir a consistência. Em ambos

os processos (do CMMI-DEV e do MR-MPS), existe uma interseção com o processo de Verificação (*Verification*) (VER), também presente nos respectivos níveis (3 e D). Estes modelos corroboram com a necessidade de se manter a consistência entre os artefatos gerados em diferentes fases do processo de desenvolvimento.

Existe uma deficiência em reconhecer a distinção entre a arquitetura planejada e a implementada. De acordo com TAYLOR *et al.* (2009), isso resulta na incapacidade de raciocinar a respeito da arquitetura do software no futuro, de forma que os *stakeholders* se enganem quanto ao que eles acreditam que têm (isto é, sem a percepção do que eles realmente têm), fazendo com que qualquer estratégia de desenvolvimento ou evolução que se baseie na arquitetura documentada (porém imprecisa) esteja fadada ao fracasso.

Neste cenário, surgem conceitos e mecanismos de gerência de configuração de software (ESTUBLIER, 2000), que permitem que os artefatos de software evoluam de maneira controlada, e que podem ser utilizados também para a comparação e o versionamento de modelos arquiteturais de software.

2.5 Comparação de Modelos de Software

Na engenharia de software, o propósito de um modelo é representar um aspecto particular de um sistema de software, podendo ser usado de forma descritiva, para determinar propriedades de um sistema, ou prescritiva, como uma especificação de um sistema a ser construído (SEIDEWITZ, 2003). A atividade de modelagem deve resultar em um modelo que seja mais simples (de usar e de criar) do que o sistema real, porém, sendo completo o suficiente para que seja considerado útil (GRAAF, 2007a). A arquitetura de um software é comumente descrita por meio de modelos, também conhecidos como modelos arquiteturais, que podem representar diferentes visões da arquitetura.

A comparação de modelos pode ser utilizada para diversas finalidades, algumas delas citadas em (VASCONCELOS, 2007). Para o foco deste trabalho, cabe ressaltar as seguintes:

- Obter a percepção da evolução do software: quando há versões de um mesmo modelo, resultante de uma determinada fase do processo de desenvolvimento; por exemplo, é possível comparar duas versões de um determinado modelo da arquitetura conceitual – também denominado "modelo conceitual" –, ou duas versões de modelos obtidos a partir de engenharia reversa do código fonte; e

- Verificação de conformidade: quando os modelos comparados são resultantes de diferentes fases no processo de desenvolvimento; por exemplo, pode ser feita a comparação de uma versão de um modelo conceitual com uma versão de um modelo obtido a partir do código fonte, para a detecção de desvios.

A compreensão de arquiteturas e modelos de software não é uma atividade trivial, pois consome tempo e esforço, devido a fatores tais como (KNODEL *et al.*, 2008): (i) a alta complexidade dos sistemas, (ii) a grande quantidade de dados, e (iii) o número elevado de dependências entre os seus elementos arquiteturais.

O uso de técnicas tradicionais de controle de versão não permite um controle adequado à granularidade dos elementos que compõem um modelo arquitetural, isto é, não são levados em conta aspectos importantes como a sintaxe, estrutura ou semântica dos elementos envolvidos, realizando uma comparação linha a linha dos documentos (VASCONCELOS, 2007). Ferramentas tradicionais visam ser independentes de linguagem, sendo voltadas para operar sobre arquivos de texto. No entanto, mesmo que tais arquivos de texto contemplem descrições arquiteturais, o resultado da comparação não possui a natureza arquitetural esperada (WESTHUIZEN & HOEK, 2002).

Adicionalmente, as diferenças arquiteturais obtidas por mecanismos de *diff* não são muito intuitivas (algumas vezes restringindo-se a representações textuais) e podem demandar muito tempo e esforço para que sejam compreendidas (LINTERN *et al.*, 2003). No caso especial de modelos bidimensionais (como diagramas), não é apropriada a exibição das diferenças de maneira tradicional (i.e., em duas colunas lineares de forma que elementos correspondentes se situem em frente um do outro) (OHST *et al.*, 2003; SCHIPPER *et al.*, 2009).

Atualmente, existem diversos trabalhos na literatura voltados para o versionamento e/ou a comparação de modelos arquiteturais. Alguns deles estão brevemente apresentados a seguir.

A abordagem UMLDiff (XING & STROULIA, 2005) consta de um algoritmo que detecta automaticamente mudanças estruturais entre modelos (obtidos por engenharia reversa) de versões subsequentes de produtos de software orientado a objetos. Este algoritmo faz uso de uma métrica de similaridade, cujo valor é uma ponderação da similaridade entre os nomes e da similaridade estrutural dos elementos.

ABI-ANTOUN *et al.* (2007) apresentam um algoritmo denominado MDIR (*Moves-Deletes-Inserts-Renames*) baseado em informações estruturais, generalizando

um algoritmo de correção (denominado "*tree-to-tree correction*") para efetuar comparações, classificar mudanças (conforme o tipo de mudança, detectando elementos inseridos, removidos, renomeados e movidos) e permitir operações de junção (*merge*) sobre visões arquiteturais hierárquicas, representadas por árvores. O algoritmo considera uma dada árvore e calcula o custo de operações (aplicadas às subárvores e aos nós folha) para transformá-la em outra árvore.

A abordagem ArchToDDSA (KÜMMEL, 2007) visa a comparação de arquiteturas de aplicações, detectando similaridades, diferenças e variabilidades para que estas informações sirvam de apoio na criação de uma Arquitetura de Software Específica de Domínio (*Domain Specific Software Architecture – DSSA*). A ferramenta desenvolvida efetua comparações de arquivos XMI (*XML Metadata Interchange*), uma especificação para possibilitar o compartilhamento de modelos UML provenientes de ferramentas CASE distintas (OMG, 2007).

TREUDE *et al.* (2007) elaboraram o framework SiDiff (*Similarity-based Difference*), que provê um algoritmo de *diff* configurável, voltado para a comparação de modelos em larga escala. O algoritmo detecta as similaridades sem depender de identificadores ou de nomes únicos (chaves primárias) de elementos, baseando-se apenas nos valores de atributos e na vizinhança (elementos referenciados) de cada elemento.

A abordagem ArchTrace (MURTA *et al.*, 2007) visa atualizar continuamente os rastros entre versões de elementos arquiteturais e versões de elementos de código fonte, na medida em que novas versões da arquitetura e do código fonte são produzidas. A ferramenta possibilita a exploração dos links basicamente de duas maneiras: é possível visualizar todos os links que apontam para um dado elemento arquitetural ou selecionar um arquivo de forma a descobrir os elementos arquiteturais aos quais este arquivo pertence. Além disso, desenvolvedores podem escolher, a partir de um conjunto de políticas de gerência de rastros, aquelas que melhor correspondam às suas necessidades situacionais e/ou estilos de trabalho. Além de a arquitetura emergente possuir granularidade grossa (arquivo), a abordagem não faz uso de métricas que permitam obter uma visão geral do grau de divergência.

O trabalho de UHRIG (2008) propõe um algoritmo que, de forma semelhante à utilizada na abordagem de ABI-ANTOUN *et al.* (2007), calcula o custo estimado para que um dado elemento de um diagrama de classes seja transformado em outro, levando em consideração operações de inserção e remoção de elementos e mudanças de nome,

tipo e valor (no caso de cardinalidade de associações) dos elementos. Os custos estimados são um limite inferior dos custos reais, visto que o cálculo destes teria um custo computacional alto.

A abordagem Odyssey-MEC (CORRÊA *et al.*, 2008) oferece suporte à transformação, sincronização e versionamento de modelos para manter a consistência durante a evolução de modelos inter-relacionados, por meio da utilização de mecanismos do Odyssey-VCS (MURTA *et al.*, 2008), um sistema de controle de versões voltado para elementos UML de granularidade fina que trata de questões de controle de transação e políticas de concorrência, dentre outras funcionalidades. A abordagem Odyssey-VCS não provê uma interface visual para a comparação de modelos, ficando a cargo da aplicação cliente a implementação de tal funcionalidade; o Odyssey-MEC, por sua vez, mostra os modelos em abas separadas, e também não possui a visualização das diferenças.

Por fim, a ferramenta dclcheck (*Dependency Constraint Language Checker*) (VILLELA, 2009) utiliza uma linguagem de restrição de dependências denominada DCL (*Dependency Constraint Language*), proposta pelo mesmo autor, por meio da qual arquitetos de software podem definir dependências aceitáveis e inaceitáveis, de acordo com a arquitetura planejada dos seus sistemas. A dclcheck verifica se o código fonte (isto é, a arquitetura concreta de um sistema) respeita as restrições de dependência definidas em DCL. Não é utilizada nenhuma métrica que permita detectar o grau de erosão arquitetural.

Nenhuma das abordagens descritas trata simultaneamente do controle de versões e da comparação de modelos com foco no aspecto visual da evolução, de forma a auxiliar a percepção e compreensão do andamento do projeto de software como um todo. Algumas propostas neste sentido que foram encontradas na literatura são discutidas no Capítulo 2.

2.6 Considerações Finais

Neste capítulo, foi apresentada uma visão geral sobre arquitetura de software (incluindo visões arquiteturais) e comparação de modelos, por meio de uma breve caracterização destes tópicos, incluindo os benefícios na utilização de seus conceitos. Dois tipos especiais de arquitetura (conceitual e emergente) foram tratados. A forma como os modelos arquiteturais evoluem também foi abordada, junto à necessidade de verificação da conformidade durante o desenvolvimento, prática também recomendada

por modelos e normas de maturidade. Por fim, foram retratados conceitos e ferramentas de gerência de configuração (em especial o controle de versões e a comparação de modelos) como apoio para o controle e o entendimento desta evolução.

SHAW & CLEMENTS (2006) identificaram pontos em aberto, áreas promissoras e oportunidades de contribuição no que diz respeito à pesquisa em arquiteturas de software. Um destes pontos, que foi posteriormente reafirmado em (CLEMENTS & SHAW, 2009), indica a necessidade de estabelecer maneiras de garantir a conformidade entre a arquitetura e o código, dado que a falta de conformidade traz irrelevância a uma arquitetura, uma vez que o código passa a estabelecer sua própria trajetória independente. Os mesmos autores afirmam que as soluções existentes para a questão da conformidade são incompletas, requerendo intervenção humana para fazer sugestões (ou palpites) sobre construções arquiteturais que possam estar escondidas no código.

Considerando que o conhecimento, a habilidade de raciocínio e de percepção do ser humano são requeridos neste e em outros cenários, torna-se necessária a utilização de técnicas que facilitem e agilizem sua participação no processo de desenvolvimento. Neste sentido, a visualização de software pode auxiliar a compreender e lidar com as informações geradas durante tal processo.

No próximo capítulo, são apresentados alguns conceitos, métodos e técnicas de visualização de software, em especial no contexto do acompanhamento da evolução de software, que é o foco principal deste trabalho.

CAPÍTULO 3 - VISUALIZAÇÃO DE SOFTWARE

3.1 Introdução

Produtos de software são criados por pessoas e voltados para pessoas. Não obstante, o entendimento do software, englobando sua estrutura, comportamento e evolução, é uma atividade complexa, que requer recursos específicos que facilitem o entendimento e a percepção do software e do seu processo de desenvolvimento (DIEHL, 2007). Neste sentido, a utilização de técnicas de visualização de software pode auxiliar estas atividades que envolvem o raciocínio humano, fornecendo uma melhor representação das informações obtidas.

A visualização de software pode ser definida como a utilização de técnicas de visualização da informação (CHEN, 2006) aplicadas a software. Os dados sobre o desenvolvimento são abstratos e, por esta razão, não possuem estrutura física associada. Neste sentido, o mapeamento de entidades (do domínio de sistemas de software) para representações gráficas visa prover suporte à compreensão e ao desenvolvimento (GALLAGHER *et al.*, 2008).

A visualização de software proporciona uma forma simples e intuitiva de compreensão dos dados representados (BALL & EICK, 1996), tornando "visíveis" os artefatos e o comportamento do software por meio de representações visuais. Segundo MUKHERJEA & FOLEY (1996), a visualização destes dados é particularmente importante por permitir que as pessoas utilizem o raciocínio perceptivo (em vez do raciocínio cognitivo) na realização das tarefas, o que facilita a detecção de padrões que revelam a estrutura implícita nos dados.

De acordo com DIEHL (2007), o objetivo da visualização de software não é produzir imagens agradáveis ao olho humano, mas imagens que evocam abstrações mentais, que permitam memorizar conceitos e explorar analogias para compreender melhor o software e suas estruturas ou funções. Assim, a descoberta e o desenvolvimento de novas metáforas não apenas produzem melhores visualizações, mas também melhoram a percepção das pessoas a respeito de sistemas.

A visualização é amplamente utilizada nas áreas de engenharia mecânica, química, física e medicina, graças a sistemas sofisticados desenvolvidos para estas disciplinas por cientistas da computação (DIEHL, 2007). O mesmo autor afirma que,

para que a visualização seja efetiva em seu objetivo, é importante ter em mente que as representações e metáforas visuais devem ser adaptadas às habilidades de percepção dos interessados, e não o contrário.

Além desta seção introdutória, este capítulo apresenta algumas das principais técnicas de visualização de software na Seção 3.2 e suas subseções. A visualização aplicada no domínio de evolução de arquiteturas de software é discutida na Seção 3.3. A Seção 3.4 e suas subseções apresentam alguns trabalhos relacionados. Por fim, na Seção 3.5, as considerações finais deste capítulo são relatadas.

3.2 Princípios e Técnicas de Visualização de Software

A sobrecarga de informação (*information overloading*) é um problema conhecido, que se deve ao crescimento exponencial de informações amplamente acessíveis na sociedade moderna. Para contornar este problema, é necessária a aplicação de mecanismos eficientes de tratamento de dados e informações (e.g., classificação e filtragem), bem como facilidades de compartilhamento (CHEN, 2006). O seguinte processo representa o fluxo de tratamento destes dados (a informação produzida em um estágio serve de entrada para o estágio seguinte) (DIEHL, 2007):

- Aquisição de dados: Extração de dados (a partir de uma fonte de dados) acerca do sistema de software, tais como código fonte, modelos de projeto (*design*), resultados de teste etc.;
- Análise: Tratamento dos dados relevantes (por meio de filtragens, agrupamentos etc.) para reduzir a quantidade de informações a serem apresentadas, focando apenas no que é importante; e
- Visualização: Mapeamento dos dados resultantes em um ou mais modelos visuais, que consistem de uma série de símbolos gráficos a serem convertidos numa representação visual e apresentados ao usuário.

Os princípios e técnicas de visualização são aplicados às abstrações visuais para permitir a interação e a manipulação por parte do usuário. As subseções a seguir apresentam algumas das técnicas de visualização de software disponíveis, seguidas de uma breve discussão sobre sua utilização.

3.2.1 Zoom

A técnica de zoom consiste, basicamente, em permitir que o tamanho de um elemento visual possa ser ampliado ou reduzido dinamicamente, conforme a interação do usuário (DIEHL, 2007). Esta técnica se mostra particularmente útil quando o objeto visualizado está em escala muito pequena, dificultando a exploração de detalhes.

Uma interface de usuário com zoom aplicável (*Zoomable User Interface – ZUI*) possui como requisito essencial permitir ao usuário efetuar operações de ampliação (*zoom in*) e redução (*zoom out*) de escala livremente e com facilidade, em vários níveis de detalhe conforme o foco de interesse (CHEN, 2006). Segundo este autor, o conceito de ZUI é relativamente novo, embora técnicas similares já sejam utilizadas há anos.

É possível distinguir dois tipos básicos de zoom: zoom geométrico e zoom semântico. Enquanto o zoom geométrico (abordagem mais comum) amplia e reduz os elementos visuais, o zoom semântico possui natureza mais complexa, pois diferentes representações visuais podem ser dadas aos objetos de informação, dependendo da quantidade de espaço disponível (BUERING *et al.*, 2006). Um exemplo de zoom semântico pode ser visto na Figura 3.1, que demonstra uma transição da visão mais geral ao detalhe por meio da interação com o usuário.

Figura 3.1 – Zoom semântico (SU, 2010)

Outra forma (complementar ao zoom) de interagir com objetos é através de movimentações em escala constante (ou *panning*), normalmente acionada por recursos

de “arrastar e soltar” (*drag & drop*) em visões panorâmicas, permitindo a movimentação de objetos para dentro e para fora da área de foco.

3.2.2 Filtragem

Ao lidar com um grande volume de informação, torna-se conveniente (e por vezes necessário) exibir apenas o que é relevante no contexto de um determinado raciocínio ou tarefa de compreensão. A esta técnica de realce/inclusão dos itens mais relevantes ou da suavização/ocultação/remoção dos itens menos relevantes em um dado contexto, dá-se o nome de filtragem de informações. A Figura 3.2 mostra um exemplo de utilização desta técnica em uma visão de dependência entre classes, que pode ser utilizada para se obter informações de acoplamento (CARNEIRO *et al.*, 2010); a parte superior apresenta os nós (que representam classes e pacotes) e as arestas (que representam as ligações de dependência entre as classes e pacotes), enquanto na parte inferior as arestas foram filtradas, exibindo apenas os nós.

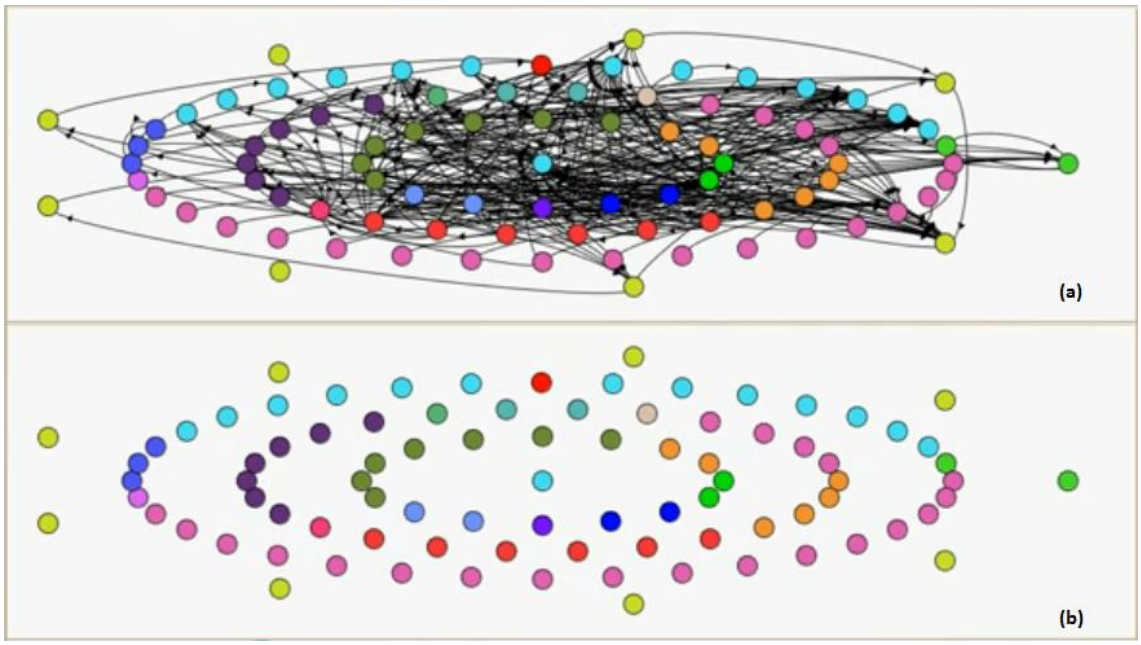


Figura 3.2 – Filtragem aplicada à visão de dependência entre classes (CARNEIRO *et al.*, 2010)

A filtragem pode ocorrer por meio de um pré-processamento (desde que seja possível saber de antemão o que deve ser filtrado) ou sob demanda (em tempo real, normalmente utilizada no contexto de tarefas específicas). Um exemplo de filtragem no pré-processamento aplicada à visualização da informação é a baseada no conceito de

perfis de usuários, que contêm estimativas da relevância de cada tipo de informação para cada pessoa (CHEN, 2006).

3.2.3 Agrupamento (*Clustering*)

O objetivo da análise de agrupamentos é dividir um grande conjunto de dados em um número de subconjuntos (grupos), de acordo com determinadas medidas de similaridade (CHEN, 2006).

No desenho de grafos, por exemplo, o destaque de agrupamentos e valores atípicos (*outliers*) nas visualizações podem proporcionar descobertas interessantes a respeito das relações entre os nós. Um princípio de visualização para grafos mais complexos sugere que (i) as partes que estão fortemente interconectadas (e que podem ser agrupadas) devem ser desenhadas separadamente de quaisquer outras partes, e que (ii) arestas traçadas dentro de um agrupamento devem ser mais curtas do que as que conectam agrupamentos distintos (DIEHL, 2007).

3.2.4 Hierarquias

Hierarquias (ou árvores) são abstrações de estruturas de informação comumente utilizadas, não apenas por desempenharem um papel importante, mas também por permitirem a representação de estruturas mais complexas (CHEN, 2006). Alguns exemplos de hierarquias são a estrutura organizacional de um sistema de arquivos, a estrutura de um sistema de classificação e a classificação biológica de todos os animais.

Uma forma clássica de representar informações hierárquicas é por meio de *treemaps* (JOHNSON & SHNEIDERMAN, 1991), cuja ideia básica é dividir uma área recursivamente em subáreas que não se sobrepõem, permitindo a navegação entre áreas. Em muitas aplicações, o tamanho de cada subárea no *treemap* (calculado por um algoritmo de preenchimento de espaço) é proporcional a alguma medida (utilizada como peso). No caso mais simples, este peso é a soma de todos os elementos da subárvore correspondente (DIEHL, 2007).

3.2.5 Detalhamento (*Drill-Down*)

O ciclo cognitivo ideal envolvendo um computador está relacionado a obter a informação de forma precisa e no momento em que ela se faz necessária (ou seja, ter acesso apenas às informações relevantes em um determinado instante). Isto também resulta na redução do custo de obtenção de informações, quando estas estão

relacionadas a algo já descoberto (WARE, 2008). Está técnica de revelar mais detalhes conforme a necessidade cognitiva do usuário é conhecida como detalhamento (*drill-down*), que difere do zoom semântico em função de estar associada a uma hierarquia inerente.

A Figura 3.3 apresenta um exemplo de uso da técnica de *drill-down* sobre um *treemap*, detalhando camadas sucessivas de uma hierarquia. Ao selecionar o nó pai de uma subárvore em que pode residir o nó de interesse, esta subárvore passa a ocupar todo o espaço de apresentação e o usuário pode selecionar subárvores recursivamente até atingir a localização ou o nó desejado. No exemplo da figura, são necessárias quatro operações de detalhamento de forma a examinar o nó realçado (SHI *et al.*, 2005).

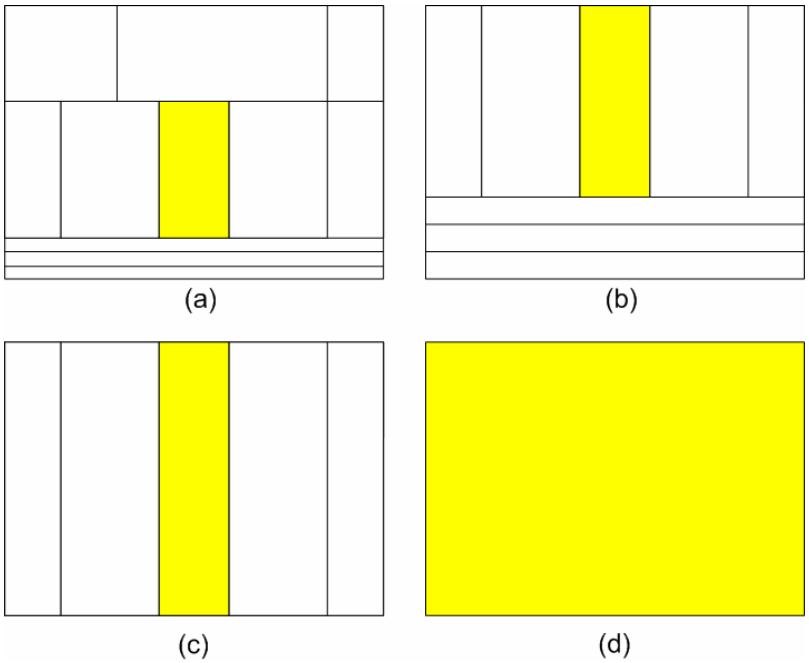


Figura 3.3 – Técnica de *drill-down* aplicada a um *treemap* (SHI *et al.*, 2005)

3.2.6 Foco + Contexto

A técnica de foco + contexto (*focus + context*) busca eliminar a separação temporal e espacial, por meio da exibição do foco dentro do contexto, em uma única visão contínua (COCKBURN *et al.*, 2008). Esta técnica pode ser implementada com uma variedade de métodos que propiciem a redução seletiva da informação apresentada, tais como filtragem de informações irrelevantes, agregação seletiva de informações relacionadas, micro e macroleitura, realce (*highlighting*) e distorção (CARD *et al.*, 1999).

Um exemplo de utilização desta técnica é o uso de visões "olho de peixe" (*fisheye views*). Estas visões se baseiam na distorção de uma lente grande-angular

(*wide-angle lens*) que apresenta o foco em detalhe (zoom), enquanto as regiões periféricas são exibidas em detalhe progressivamente menor, conforme a distância do foco de interesse. A Figura 3.4 apresenta a técnica aplicada ao painel de ícones do sistema operacional Mac OS.

Figura 3.4 – Visão "olho de peixe" no painel X Dock (COCKBURN *et al.*, 2008)

Em comparação com ZUIs (conceito mencionado na Subseção 3.2.1), o "olho de peixe" tem a vantagem de que tanto o detalhe quanto o contexto podem ser integrados em uma única visão ao mesmo tempo (BUERING *et al.*, 2006). No entanto, STASKO (2005) aponta as seguintes desvantagens desta técnica: (i) a distorção pode ser irritante, (ii) pode ser muito difícil e complicada de implementar, e (iii) qualquer alteração no ponto focal irá potencialmente requerer um recálculo do grau de interesse para todos os objetos e, portanto, reprocessamento de todos os objetos, o que é computacionalmente custoso.

3.2.7 Visão Geral + Detalhe

Uma interface que utilize esta técnica é caracterizada pela exibição simultânea de uma visão geral (*overview*) e uma visão detalhada (*detail*) de um mesmo espaço de informação, sendo que cada visão ocupa um espaço de apresentação distinto. Em função da separação física entre estas visões, os usuários interagem com cada uma delas separadamente; contudo, muitas vezes as ações em uma das visões são refletidas imediatamente na outra (COCKBURN *et al.*, 2008).

Os mesmos autores explicam que a principal diferença desta técnica com relação à técnica foco + contexto é que, nesta última, a apresentação das informações se concentra em um único espaço de apresentação, onde todas as partes são simultaneamente visíveis.

Um exemplo de visão geral + detalhe é o *minimap* (ou *miniview*), que busca abranger, em uma visão compacta, todo o conteúdo visível no momento da exibição da visão, além de partes significativas das zonas circundantes. Em função da escala reduzida, o *minimap* pode apresentar o conteúdo com um layout original ligeiramente

modificado, dando ênfase apenas a itens cuja identificação no contexto do *minimap* seja relevante; é comum, por exemplo, a utilização de uma figura geométrica em *minimaps* para indicar a posição corrente do detalhe em um dado momento (ROTO *et al.*, 2006).

A aplicação de *minimaps* é observada com frequência em jogos com cenários complexos, de forma a permitir que os jogadores se orientarem em função de locais importantes, inimigos e aliados (SILVA, 2010). Em software, é comum a utilização de *minimaps* nas ferramentas em que a escala dos artefatos visualizados ultrapassa o espaço disponível para sua exibição. Um exemplo pode ser visto na Figura 3.5.

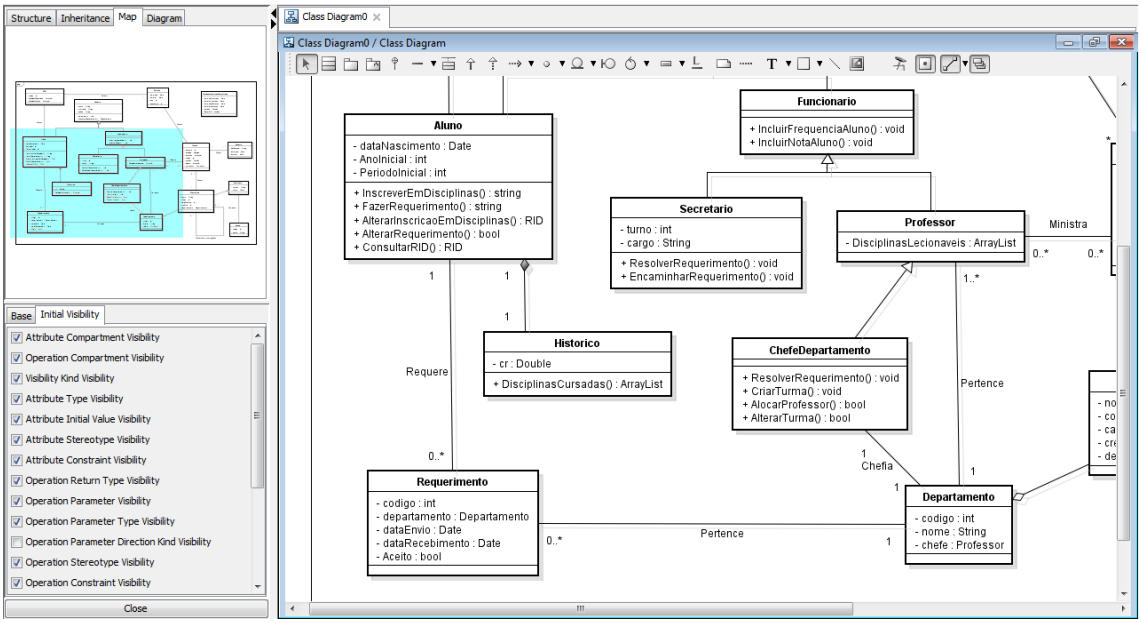


Figura 3.5 – *Minimap* da ferramenta astah* Community (CHANGEVISION, 2010)

3.2.8 Discussão

De acordo com CHEN (2006), as implicações práticas na escolha de uma técnica de visualização em especial devem ser analisadas tendo em conta os princípios de projeto (*design*) que deram origem à técnica em questão. Adicionalmente, o autor afirma que o poder da visualização só passa a ser plenamente compreendido quando a visualização se torna parte integrante das atividades dos usuários.

STOREY *et al.* (2005) afirmam que uma visualização só é útil se os dados atendem a um determinado propósito, tendo em vista que muitas das questões provenientes do desenvolvimento de software podem ser respondidas por derivação e agregação de fatos provenientes de várias fontes de dados.

Por vezes, além das técnicas de visualização apresentadas, pode ser necessária a utilização de visões múltiplas para evitar (ou ao menos controlar) a sobrecarga cognitiva

(*cognitive overloading*). Algumas características desejáveis para sistemas desta categoria são (BALDONADO *et al.*, 2000):

- Seleção: a identificação de um conjunto de visões a ser utilizado de forma coordenada em prol de uma determinada tarefa;
- Apresentação: as visões podem ser apresentadas em sequência (neste caso, o usuário pode fazer uso de um *menu* para alternar entre diferentes visões), simultaneamente ou em qualquer outra configuração possível na tela; e
- Interação: cada visão pode ter *affordances*[1] independentes (e.g., recursos de seleção ou a funcionalidade de navegação, tais como *pan* e *zoom*) que podem ser vinculados, de modo que as ações efetuadas em uma visão possam ter um efeito na outra visão.

3.3 Visualização Aplicada à Evolução de Arquiteturas de Software

CASERTA & ZENDRA (2010) consideram que a visualização da evolução da arquitetura do software é um dos tópicos mais importantes no domínio da visualização da evolução. É importante ter uma visão global da evolução do sistema para que seja possível explicar e documentar o estado atual do projeto de software.

TELEA *et al.* (2010) observaram que a adoção efetiva de novas ferramentas de visualização de arquiteturas se correlaciona fortemente com três tipos diferentes de *stakeholders* e suas percepções de valor:

- Usuários técnicos (desenvolvedores, *designers*, testadores e arquitetos): foco na criação de um produto de software;
- Gerentes: foco na execução integrada do projeto durante longos períodos de tempo; e
- Consultores: foco em períodos curtos de trabalho para auxiliar na tomada de decisões estratégicas.

Os mesmos autores acrescentam que alguns dos valores chave, do ponto de vista de gerentes de projeto ou arquitetos, são a identificação de problemas na evolução do software e a monitoração da execução do projeto (equipe, qualidade, seguimento aos

[1] *Affordance* indica uma ação que um indivíduo pode, potencialmente, vir a realizar em seu ambiente.

planos). De forma a atender à demanda destes *stakeholders*, os autores também afirmam que os principais requisitos para uma ferramenta de visualização de software são: (i) extração automática de dados a partir de repositórios, (ii) geração automática da visualização, e (iii) alta escalabilidade (de processamento e visualização), em termos dos dados a serem tratados.

De uma maneira geral, visualizar a evolução do software não é uma tarefa fácil, pois a adição da dimensão tempo pressupõe a inclusão de dados extras (CASERTA & ZENDRA, 2010). No que tange a evolução de modelos arquiteturais, conforme mencionado na Seção 2.5, as diferenças entre modelos obtidas por mecanismos de *diff* não são muito intuitivas, podendo demandar muito tempo e esforço para que sejam compreendidas (LINTERN *et al.*, 2003). Para contornar estes obstáculos, a comparação destes modelos (seja entre versões de um mesmo modelo ou entre modelos distintos) pode ser auxiliada pelas técnicas de visualização de software construídas especificamente para este propósito.

Segundo TELEA *et al.* (2010), uma boa ferramenta de visualização de arquiteturas deve ser capaz de prover suporte a diversos tipos de dados e proporcionar meios para comparar, correlacionar e examinar dados e visões, além de permitir integração com ferramentas existentes e ser flexível, customizável e escalável. Estes autores também afirmam que as principais limitações técnicas para a análise de arquiteturas incluem (i) a falta de soluções confiáveis e facilmente reutilizáveis para extração automatizada da arquitetura, (ii) a comparação (visual ou não) de arquiteturas, e (iii) a detecção de padrões arquiteturais, apesar de existirem pesquisas em andamento nestas áreas.

3.4 Trabalhos Relacionados

A visualização de software tem sido aplicada na engenharia de software para diversos propósitos. As subseções seguintes apresentam alguns trabalhos e ferramentas que, de alguma forma, possuem relação com o tema de evolução de arquiteturas e modelos de software.

3.4.1 Visualizador de Diferenças em Diagramas UML

O trabalho desenvolvido por OHST *et al.* (2003) visa à detecção e visualização de diferenças existentes entre versões de diagramas UML. A abordagem proposta por estes autores representa, em um único diagrama, elementos comuns a ambas as versões

e elementos específicos de cada versão, sendo estes últimos destacados para o usuário, conforme pode ser visto na Figura 3.6.

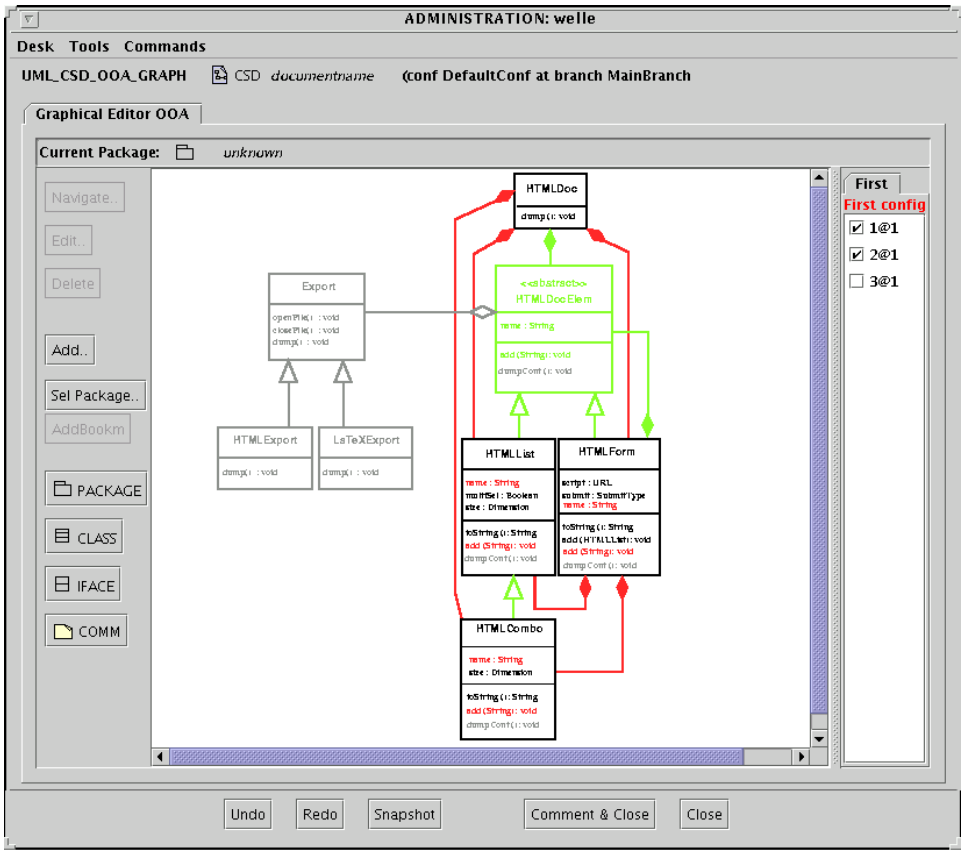


Figura 3.6 – Visualização das diferenças em um editor gráfico (OHST *et al.*, 2003)

OHST *et al.* (2003), porém, não tratam o conceito de comparação entre arquiteturas conceitual e emergente. Além disto, a abordagem não prevê a exibição da sequência temporal das versões com suas diferenças capturadas.

3.4.2 SAVE

A ferramenta SAVE (*Software Architecture Visualization and Evaluation*) (KNODEL *et al.*, 2006a) tem como propósito a avaliação e visualização de arquiteturas estáticas, com foco em verificações de conformidade entre os elementos arquiteturais a partir da comparação efetuada entre a arquitetura conceitual e a emergente (sendo esta obtida através de engenharia reversa, utilizando-se de estratégias de transformação do código para modelo). A Figura 3.7 exibe um exemplo de utilização da ferramenta na avaliação do protótipo de um sistema voltado para o domínio aeroespacial.

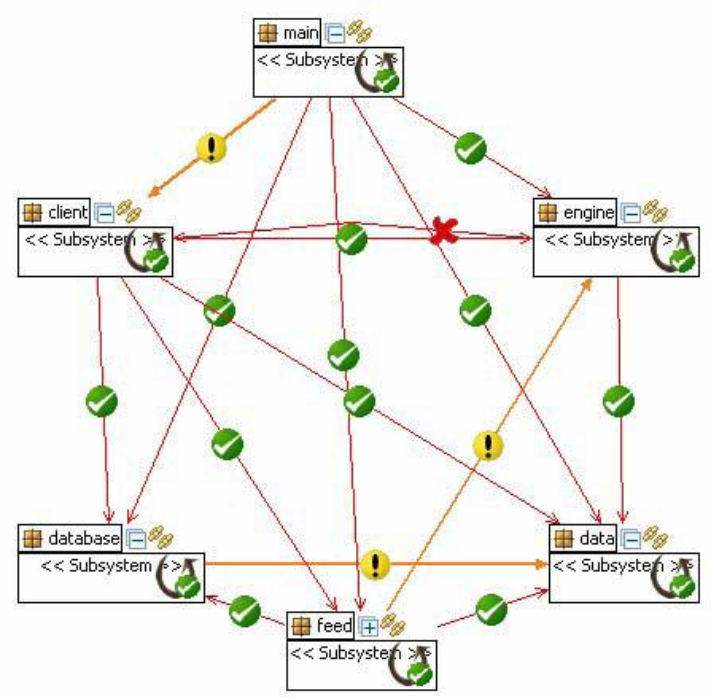


Figura 3.7 – Verificação de conformidade de um sistema, utilizando a ferramenta SAVE (KNODEL & POPESCU, 2007)

A ferramenta também possui um *plug-in* (LAMERSDORF & KNODEL, 2006) que adiciona a funcionalidade de visualização de métricas que podem ser customizadas e estendidas, como pode ser visto na Figura 3.8.

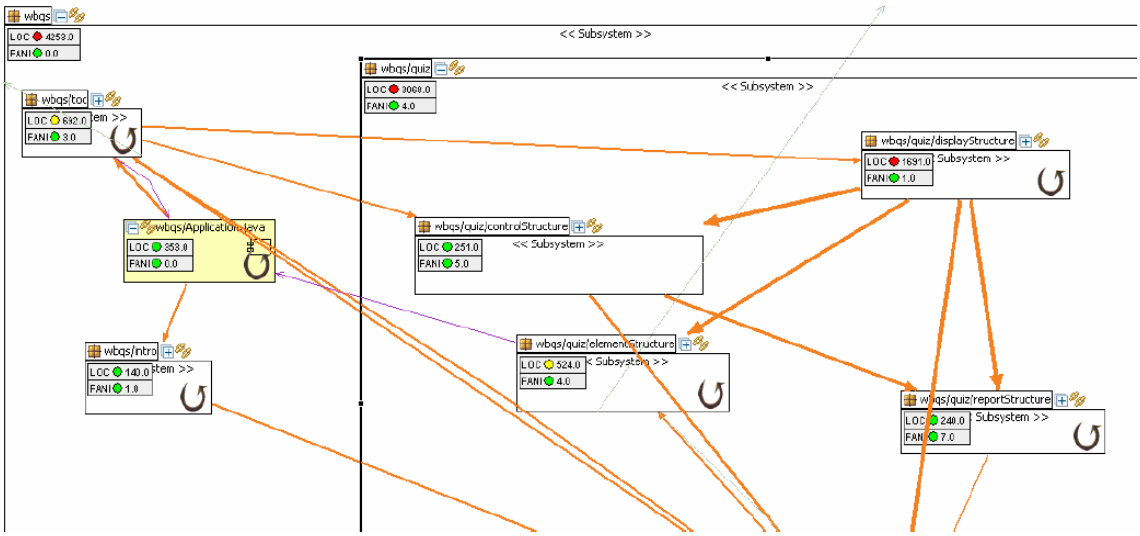


Figura 3.8 – Visualização de métricas com a ferramenta SAVE (LAMERSDORF & KNODEL, 2006)

Um aspecto não tratado pela ferramenta é a captura e o armazenamento dos modelos obtidos em diferentes instantes do tempo. Desta forma, faz-se necessária a importação manual de cada versão, não sendo possível a navegação entre versões dos

modelos arquiteturais e suas diferenças; embora o foco de visualização da ferramenta SAVE não seja a evolução das arquiteturas, tal funcionalidade poderia ser útil em tarefas de verificação da conformidade ao longo do tempo. Em função da ausência da dimensão tempo, a variação da similaridade entre as arquiteturas também não é explorada.

3.4.3 FAME

WENZEL (2008) propõe uma abordagem denominada FAME (*Fine-grained Analysis of Model Evolution*), baseada em visões polimétricas (*polymetric views*) (LANZA & DUCASSE, 2003) para a apresentação de diferenças entre modelos. Tais visões, no contexto deste trabalho, representam as métricas de diferenças como funções que mapeiam para valores numéricos as alterações que ocorrem em elementos específicos do modelo. Segundo a autora, estas métricas permitem a contagem, agregação ou classificação das alterações de acordo com sua relevância.

Foi implementado um protótipo da abordagem, voltado para o ambiente de desenvolvimento Eclipse, conforme é mostrado na Figura 3.9. Os modelos selecionados pelo usuário para comparação (aba "*Navigator*") são transformados em estruturas baseadas em grafos e, após a comparação, as métricas são computadas (com base na diferença resultante) e apresentadas ao usuário.

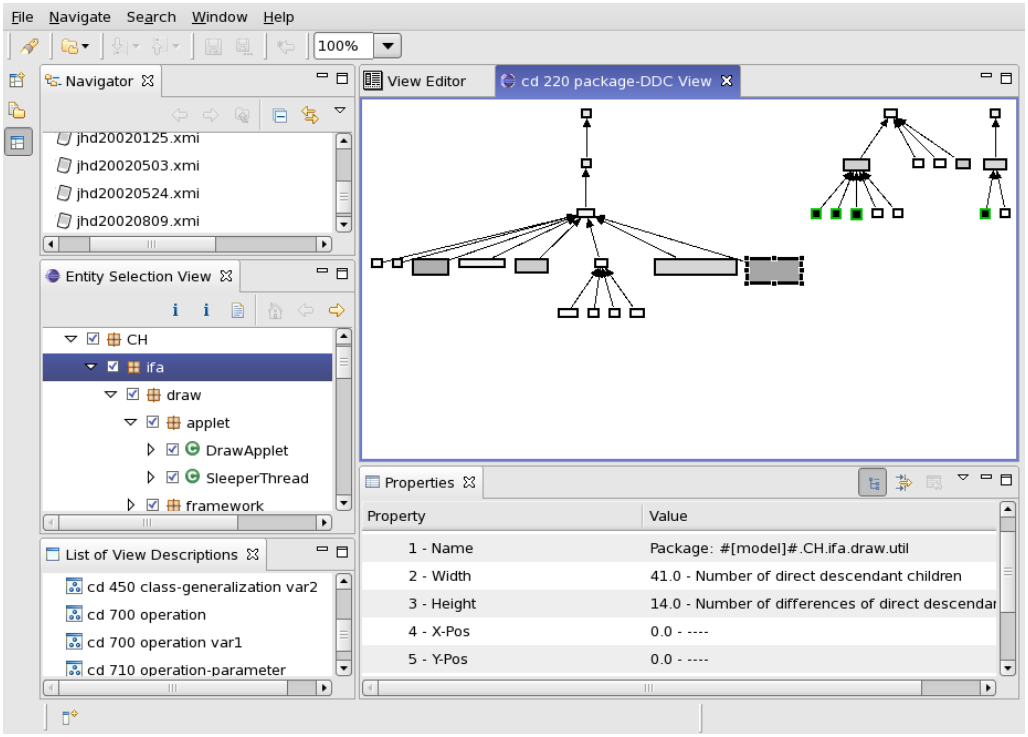


Figura 3.9 – FAME: Visualização de métricas de diferenças (WENZEL, 2008)

Para o cálculo das diferenças entre modelos, é utilizado o framework SiDiff (TREUDE *et al.*, 2007), que provê um algoritmo de *diff* configurável, baseado na identificação da similaridade entre os elementos (já mapeados para grafos).

Segundo WENZEL & KELTER (2008), a ferramenta é capaz de importar modelos do Subversion (embora não tenha sido possível verificar se tal importação ocorre de forma automatizada). No entanto, o foco da ferramenta se restringe à visualização das diferenças e dos rastros entre modelos, sem considerar a aderência do código fonte ao modelo (isto, provavelmente, só seria possível se um modelo fosse criado manualmente para cada versão do código fonte, já que a ferramenta não faz uso de engenharia reversa). Além disso, o uso de visões polimétricas para compreender modificações entre versões não é muito intuitivo, visto que o a percepção sobre o significado dos detalhes das diferenças não é facilmente obtida usando somente estas visões (BRAND *et al.*, 2010).

3.4.4 KIELER

KIELER (*Kiel Integrated Environment for Layout for the Eclipse Rich Client Platform*) (SCHIPPER *et al.*, 2009) é um framework de modelagem construído com o objetivo de investigar o projeto (*design*) de sistemas complexos, baseado em modelos gráficos. Um exemplo de comparação de diagramas de fluxo de dados é exibido na Figura 3.10.

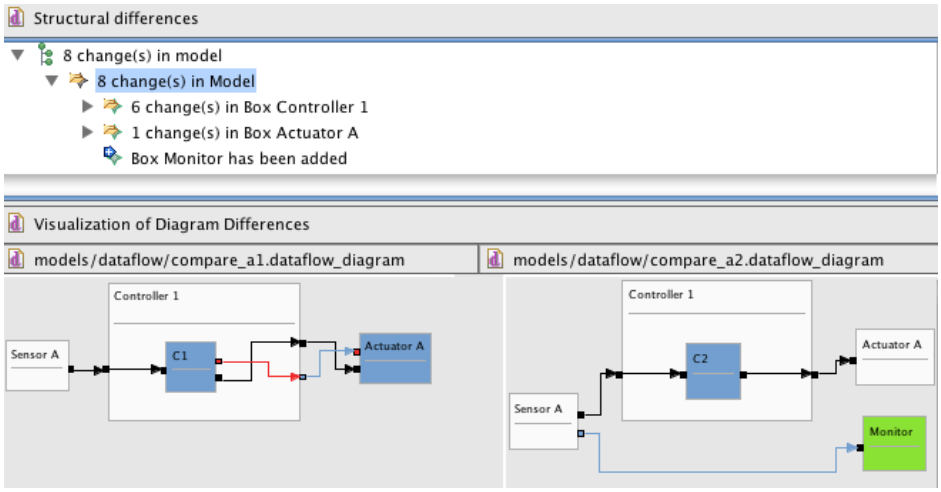


Figura 3.10 – Exibição das diferenças na ferramenta KIELER (SCHIPPER *et al.*, 2009)

A ferramenta KIELER utiliza-se de cores em diagramas para representar as diferenças. Foi incorporada à visão original do framework EMF Compare uma visão adicional, permitindo visualizar graficamente as diferenças. Isto faz com que, assim

como no EMF Compare, seja necessário selecionar manualmente os modelos a serem comparados a cada etapa de comparação, o que pode se tornar uma tarefa cansativa e propensa a erros em tarefas de compreensão da evolução de modelos.

3.4.5 SourceMiner

SourceMiner (CARNEIRO *et al.*, 2010) é uma ferramenta baseada em visões múltiplas para melhorar as atividades de compreensão do software, fazendo uso de técnicas interativas (tais como filtragem e navegação) de forma a apoiar os programadores na construção de modelos mentais sobre um sistema de software.

A ferramenta foi projetada e implementada como um *plug-in* do Eclipse, e fornece recursos visuais integrados a este ambiente (incluindo filtragem, zoom etc.). A Figura 3.11 mostra a visualização da dependência entre classes, uma das visões da ferramenta. Além desta, a ferramenta também possui visões polimétricas, *treemaps*, entre outras.

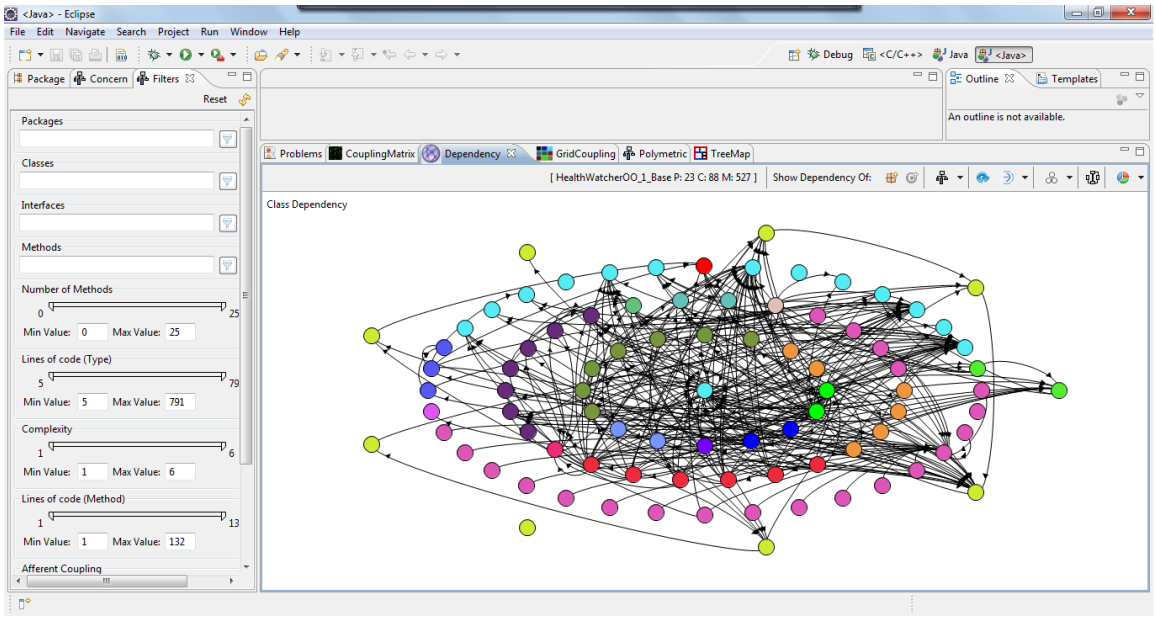


Figura 3.11 – Visão de dependência entre classes (CARNEIRO *et al.*, 2010)

Embora o SourceMiner exiba os dados presentes na IDE em um dado instante, a ferramenta não permite visualizar o histórico do projeto ao longo do tempo. Em função disto, também não foi possível identificar a funcionalidade de comparação entre versões, visando verificar mudanças nos valores de métricas. Segundo os autores, encontra-se em andamento um trabalho que utiliza a ferramenta para apoiar a análise de evolução de software (CARNEIRO *et al.*, 2010).

3.5 Considerações Finais

Neste capítulo, foram apresentados alguns conceitos, princípios e técnicas de visualização de software, junto a alguns contextos de utilização e exemplo. Foi discutida a visualização de software no contexto de evolução de arquiteturas de software. Por fim, alguns trabalhos relacionados foram apresentados, junto a suas características.

Com base nos trabalhos de CASERTA & ZENDRA (2010) e TELEA *et al.* (2010), juntamente com a discussão apresentada na Seção 3.3, foi observado que uma abordagem voltada para a visualização da evolução arquitetural deve contemplar as seguintes características (aqui denominadas como requisitos):

R1. Extração automática de dados a partir de repositórios;
R2. Geração automática da visualização;
R3. Alta escalabilidade (de processamento e visualização), em termos dos dados a serem tratados;
R4. Capacidade de dar suporte a diversos tipos de dados;
R5. Meios para comparar, correlacionar e examinar dados e visões;
R6. Integração com ferramentas existentes;
R7. Flexibilidade e capacidade de customização;
R8. Auxílio na identificação de problemas na evolução do software;
R9. Facilidades na monitoração da execução do projeto (planejado × realizado); e
R10. Tratamento da dimensão de tempo (i.e., dados sensíveis ao tempo).

A Tabela 3.1 apresenta um quadro comparativo das abordagens descritas nesta seção, no que diz respeito aos requisitos mencionados. Cabe ressaltar que as abordagens comparadas podem diferir em diversos aspectos (foco principal de visualização, tipo de tarefa para a qual provê suporte etc.), o que pode resultar em alguns dos requisitos não serem atendidos.

Como se pode observar, nenhuma das soluções atende a todos os requisitos identificados (mesmo que de forma parcial). Isto motiva o desenvolvimento de uma nova abordagem para a visualização da evolução arquitetural.

Tabela 3.1 – Quadro comparativo das abordagens

Requisitos	**Abordagens**				
	OHST *et al.* (2003)	KNODEL *et al.* (2006a)	WENZEL (2008)	SCHIPPER *et al.* (2009)	CARNEIRO *et al.* (2010)
R1	+	–	?	–	–
R2	?	+	+	+	+
R3	?	+	+	?	?
R4	–	+	+	+	+
R5	+	+	+	+	+/–
R6	+	–	+	+	+
R7	–	+	+	?	+
R8	–	+/–	+	–	–
R9	–	–	–	–	–
R10	+/ –	–	+	–	–

\+ Requisito atendido
+/– Requisito parcialmente atendido
– Requisito não atendido
? Não foi possível avaliar (fontes de informações não encontradas)

Considerando os estudos apresentados sobre arquitetura e visualização de software, acredita-se que uma abordagem que trate da percepção da evolução de modelos arquiteturais de software pode auxiliar equipes de desenvolvimento em tarefas de compreensão, permitindo maior interação com o usuário e explorando melhor a dimensão tempo. Assim, foi criada a abordagem PREViA, que é apresentada no próximo capítulo.

CAPÍTULO 4 - ABORDAGEM PROPOSTA

4.1 Introdução

A partir da revisão da literatura descrita no Capítulo 3 deste trabalho, foi possível observar a importância e os ganhos obtidos com o uso da visualização de software nas tarefas de desenvolvimento e, particularmente, no que tange à evolução, ou seja, ao andamento do processo. De acordo com CASERTA & ZENDRA (2010), a visualização gráfica do software tem potencial para resultar em um entendimento melhor e mais rápido do seu projeto (*design*) e suas funcionalidades, poupando tempo e fornecendo informações valiosas para melhorar a sua qualidade.

Neste sentido, a abordagem proposta nesta dissertação, denominada PREViA (Procedimento para Representar a Evolução por meio da Visualização de Arquiteturas) visa apoiar atividades de desenvolvimento que envolvam a percepção e a compreensão da evolução do software (mais especificamente, de sua arquitetura).

Além desta seção introdutória, este capítulo está organizado da seguinte forma: A Seção 4.2 traz uma visão geral da solução proposta. A Seção 4.3 introduz as métricas de precisão e cobertura, adaptadas para o contexto da abordagem PREViA de forma a apoiar a avaliação do grau de divergência entre arquiteturas. Dois exemplos de utilização da PREViA são descritos na Seção 4.4. A implementação da abordagem e os recursos de visualização utilizados são detalhados na Seção 4.5. A Seção 4.6 apresenta uma discussão sobre como a abordagem atende aos requisitos identificados no Capítulo 3. Uma análise comparativa entre a PREViA e outras abordagens é feita na Seção 4.7. Por fim, na Seção 4.8, as considerações finais deste capítulo são apresentadas.

4.2 Visão Geral da Abordagem

A abordagem PREViA consiste da composição de mecanismos de comparação de modelos, extração de métricas e visualização de software de forma integrada, permitindo identificar diferenças entre estes modelos e exibi-las conforme o foco de visualização aplicável à tarefa de desenvolvimento sendo executada.

A abordagem PREViA tem como objetivo: (i) prover uma melhor percepção da aderência entre o que está sendo implementado (arquitetura emergente) com relação ao que foi projetado (arquitetura conceitual), (ii) prover uma melhor percepção sobre a

evolução da implementação (arquitetura emergente), e (iii) prover uma melhor percepção sobre a evolução do projeto (*design*) do software (arquitetura conceitual). Para atender a estes objetivos, a abordagem faz uso de conceitos e técnicas de visualização de software.

De maneira indireta, espera-se também que, a partir das informações de aderência (descritas no primeiro objetivo), seja possível identificar possíveis problemas no processo de elaboração das arquiteturas conceituais e/ou na implementação destas arquiteturas. É importante ressaltar que a abordagem PREViA não visa indicar a fonte do problema, deixando esta atividade a cargo dos envolvidos no processo de desenvolvimento. Além disso, a abordagem PREViA limita-se, até o presente momento, ao aspecto estrutural das arquiteturas e modelos.

As informações sobre a evolução são obtidas a partir de sucessivas comparações de versões armazenadas em fontes de dados versionados (e.g., repositórios de gerência de configuração). Para isto, tanto o modelo conceitual quanto o código fonte devem estar sob controle de versão. No caso do modelo emergente, a abordagem prevê o uso de técnicas de engenharia reversa para a transformação das versões de código fonte em modelos.

O processo de comparação dos elementos que compõem os modelos considera métricas de similaridades de nome, de tipo (com relação ao metamodelo a que pertence), de valor (com relação aos atributos do elemento) e de relacionamentos. Tais comparações podem ser feitas no decorrer do tempo sob uma das perspectivas a seguir:

- Entre duas versões do modelo conceitual;
- Entre duas versões do modelo emergente; e
- Entre uma versão do modelo emergente e uma versão do modelo conceitual.

A comparação entre versões de um mesmo modelo pode ser chamada de comparação horizontal, enquanto a comparação entre modelos produzidos por fases distintas do processo de desenvolvimento (e.g., projeto e implementação) pode ser denominada comparação vertical (CONRADI *et al.*, 2003). A Figura 4.1 busca ilustrar estas perspectivas de comparação ao longo do tempo.

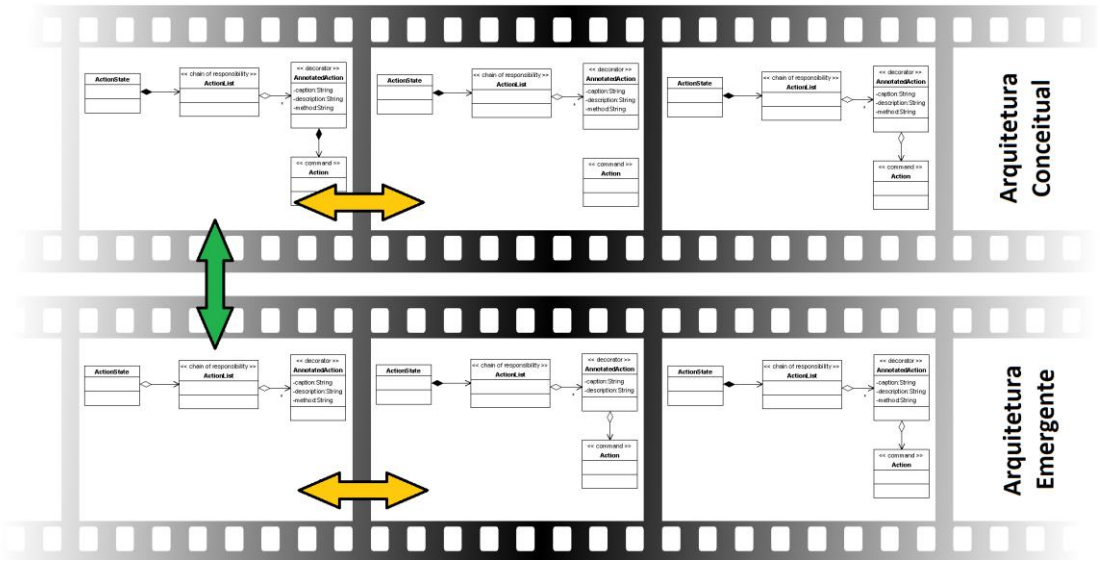


Figura 4.1 – Perspectivas de comparação da abordagem PREViA

É necessário informar à abordagem o foco de visualização desejado, o que determina a perspectiva de comparação a ser efetuada sobre os modelos. A abordagem PREViA provê dois focos de visualização: a aderência da arquitetura emergente à arquitetura conceitual no decorrer do tempo e as diferenças encontradas entre duas versões distintas em um mesmo nível de abstração no processo de evolução. É importante observar que, em ambos os focos, o que se deseja obter é a percepção das diferenças existentes entre as arquiteturas, mas sob uma ótica distinta em cada caso.

De acordo com o foco selecionado, é exibida uma determinada perspectiva de comparação, com base em dados como, por exemplo, a versão do modelo conceitual e o endereço do repositório (ou outra fonte de dados) que contém as versões do código fonte a partir das quais os modelos emergentes serão extraídos.

Em seguida, são feitas sucessivas comparações de modelos, de forma a detectar os elementos comuns a ambas e as diferenças existentes. No caso do foco de aderência, é feita uma comparação de uma versão do modelo conceitual (que pode ser modificada durante o processo) com cada versão do modelo emergente selecionada para comparação, a fim de identificar possíveis divergências estruturais. Já no caso do foco de evolução, as versões selecionadas são comparadas duas a duas, sendo que cada versão é comparada primeiramente com sua antecessora (caso haja) e, na iteração seguinte, com sua sucessora (caso haja).

Com base nestas diferenças, é gerado um modelo resultante que contém todos os elementos analisados de ambos os modelos, de forma que possam ser exibidos de acordo com o resultado das comparações efetuadas.

A fim de auxiliar a compreensão e a percepção da aderência e da evolução dos modelos, alguns dos conceitos e técnicas de visualização de software vistos no Capítulo 3 são utilizados na etapa de visualização do modelo resultante.

No que tange o foco de aderência, pretende-se transmitir a ideia de "sobreposição" ou "preenchimento" da arquitetura emergente à arquitetura conceitual no decorrer do tempo. Para isto, a arquitetura conceitual é representada na PREViA possuindo certo nível de transparência (configurável, sujeito a restrições de visibilidade), como uma marca d'água. Assim, na medida em que é identificada uma correspondência entre elementos das arquiteturas emergente e conceitual, esta última é "preenchida" pela primeira.

Os elementos que são considerados divergentes pela abordagem ganham coloração diferente, de forma que tal realce seja percebido pelo usuário. Um exemplo é mostrado na Seção 4.4 (Figura 4.4) e outro no Apêndice B (Figura B.3).

No que tange o foco de evolução, voltado para as diferenças entre versões de uma mesma arquitetura (conceitual ou emergente), diferentes cores indicam quais elementos foram adicionados, removidos ou modificados. Com isto, espera-se obter melhor identificação dos elementos modificados e, indiretamente, permitir melhor compreensão do significado da alteração realizada. A Seção 4.4 (Figura 4.7) e o Apêndice B (Figura B.6) contêm um exemplo neste contexto.

STOREY *et al.* (2005) afirmam que não é indicado que uma única ferramenta forneça todos os recursos de visualização possíveis, pois isto pode levar a uma ferramenta muito complexa e difícil de usar, não atendendo à finalidade para a qual ela foi desenvolvida. Para o desenvolvimento da abordagem PREViA, foram selecionadas as técnicas que acredita-se serem úteis (conforme suas características, apresentadas na Seção 3.2) na execução das tarefas para as quais a abordagem PREViA pretende dar suporte. Alguns detalhes da implementação destes recursos são tratados na Seção 4.5.

É esperado que o modelo emergente – extraído do código fonte – contenha elementos que são específicos da implementação – e.g., classes provedoras de acesso e persistência dos dados da aplicação (ou *Data Access Objects*), ou elementos relacionados a tecnologias específicas. Além disso, sabe-se que, idealmente, uma arquitetura conceitual não deve conter detalhes de implementação e deve ser independente de decisões específicas de tecnologia. Desta forma, quando se faz uso do foco de visualização da aderência, é importante ter em mente que a existência de

elementos específicos de implementação não representa divergências com relação à arquitetura conceitual.

A separação dos elementos específicos de implementação de forma automatizada é bastante suscetível a erros, o que prejudicaria a efetividade da comparação entre modelos arquiteturais. Por outro lado, a separação de elementos de implementação dos elementos conceituais em um modelo emergente pode não ser trivial e, consequentemente, demandar bastante tempo e esforço, o que torna inviável o processo de identificação manual.

Assim, faz-se necessária uma identificação semiautomática. É possível automatizar o processo de identificação da similaridade entre os elementos de cada modelo arquitetural. No entanto, associações entre elementos conceituais e emergentes podem ser estabelecidas ou removidas manualmente. Isto é necessário porque (i) elementos considerados como divergência podem ser, na verdade, elementos específicos de implementação, e (ii) elementos específicos de implementação do modelo emergente podem ter sido identificados incorretamente como pertencentes ao modelo conceitual.

4.3 Precisão e Cobertura

Com o intuito de quantificar a adequação entre as arquiteturas conceitual e emergente e apoiar a avaliação do grau de divergência entre arquiteturas, são utilizados os conceitos de **precisão** (*precision*) e **cobertura** (*recall*), provenientes da área de recuperação de informação (BAEZA-YATES & RIBEIRO-NETO, 1999). No contexto deste trabalho, estas medidas podem indicar, respectivamente, a exatidão e a completude de um determinado instante do desenvolvimento (ou seja, de uma dada versão do software) com relação a outro (no caso do foco de aderência, seria uma versão do modelo que foi planejado na arquitetura conceitual). Os cálculos são efetuados da seguinte forma:

$$\text{Precisão} = \frac{n_{ci}}{n_i}, \text{ onde } n_i = n_{ti} - n_{ei} \quad \text{(i)} \qquad\qquad \text{Cobertura} = \frac{n_{ci}}{n_c} \quad \text{(ii)}$$

Nas fórmulas (i) e (ii), n_{ci} é o número de elementos conceituais implementados, n_c é o número de elementos conceituais e n_i é o número de elementos implementados, resultante da subtração entre o número total de elementos implementados (n_{ti}) e o número de elementos implementados identificados como específicos de implementação (n_{ei}), isto é, que devem ser subtraídos de forma a não interferir no cálculo da precisão.

Estas medidas são propagadas dos níveis mais baixos de abstração aos mais altos, desde que o modelo utilizado para representar a arquitetura esteja hierarquicamente definido/configurado – considerando, por exemplo, uma classe UML, o cálculo é baseado em suas propriedades, operações e relacionamentos (associações e heranças) presentes nesta classe. A Figura 4.2 mostra esquematicamente (por meio de um diagrama de Venn) como as métricas de precisão e cobertura estão relacionadas.

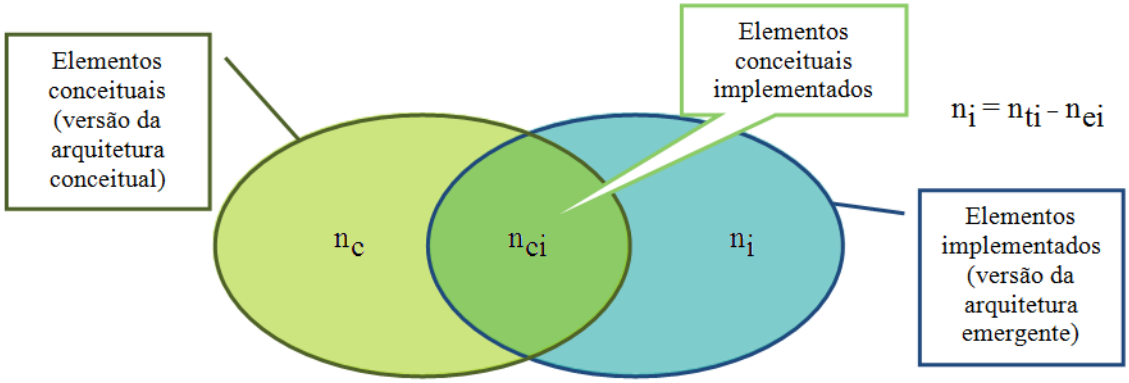


Figura 4.2 – Relação entre precisão e cobertura, no contexto da abordagem

O número total de elementos implementados é o número de elementos presentes em uma determinada versão do código fonte (i.e., o número de elementos conceituais implementados somado ao número de elementos implementados que não constam no modelo conceitual). Já o número total de elementos conceituais é o número de elementos que pertencem a uma determinada versão do modelo conceitual (i.e., o número de elementos conceituais implementados somado ao número de elementos conceituais que não foram implementados).

Os valores de precisão e de cobertura variam entre 0 e 1. Um valor de precisão igual a 1 indica que todos os elementos implementados estão no modelo conceitual (mas não necessariamente expressa que todos os elementos conceituais foram implementados); já um valor de cobertura igual a 1 indica que todos os elementos conceituais foram implementados (mas não fornece nenhuma informação a respeito de elementos que não constam no modelo conceitual e que podem também ter sido implementados). Quanto mais distantes de 1 os valores de ambas as métricas estiverem, menor é a relação de pertinência dos elementos de cada modelo no outro e, consequentemente, maior é o grau de divergência entre os modelos conceitual e emergente.

O Apêndice A mostra, em linguagem de pseudocódigo, o algoritmo criado para o cálculo de precisão e cobertura.

Durante a visualização, no contexto da abordagem PREViA, é possível especificar manualmente os elementos que são identificados pelo usuário como específicos de implementação. Visto que tais elementos impactam nos valores obtidos para a precisão, retorna-se ao cálculo desta métrica, de forma a atualizar as marcações do modelo resultante. Uma vez que um elemento é identificado desta forma, não é necessário identificá-lo novamente nas demais versões que o contenham, pois o rastro é mantido (é possível, no entanto, desfazer esta marcação a qualquer momento).

Estas métricas foram projetadas para o foco de aderência, em que há um gabarito disponível (i.e., o modelo conceitual). No entanto, o cálculo de precisão e cobertura também pode ser útil no contexto de evolução, de forma a quantificar a diferença de uma dada versão do modelo com relação à outra. Neste caso, $n_i = n_{ti}$, visto que elementos específicos de implementação não precisam ser desconsiderados, pois ambos os modelos comparados seriam da mesma natureza (i.e., conceitual ou emergente).

Um exemplo de aplicação seria no cenário em que um desenvolvedor retorna de suas férias e deseja verificar o quanto a última versão em que ele trabalhou (que faz o papel de seu "modelo conceitual") diverge da versão mais recente (emergente) do repositório, e também o quanto esta versão mais recente diverge da última versão do desenvolvedor em questão. Tais informações podem ser obtidas por meio do valor do complemento[2] às métricas de precisão e cobertura, respectivamente – ou seja, (*1 - precisão*) e (*1 - cobertura*).

4.4 Exemplos de Utilização

Nesta seção, são apresentados dois exemplos de aplicação da abordagem PREViA: um deles possui foco de visualização na aderência da implementação (representada pela arquitetura emergente) à arquitetura conceitual, enquanto o outro possui foco na visualização da evolução de um sistema em termos da implementação. Os cenários apresentados retratam situações específicas e são apenas exemplos; acredita-se que haja outros cenários em que o uso da abordagem PREViA também pode ser útil.

[2] Na Matemática, o complemento de um subconjunto A' (pertencente a um conjunto A) é o conjunto de todos os elementos de A que não pertencem a A'.

É importante ressaltar que, para fins didáticos e de ilustração, estes exemplos apresentam cenários relativamente simples. Sabe-se, no entanto que, em cenários reais, a verificação da aderência e a percepção da evolução são tarefas difíceis e complexas, em função do tamanho do sistema e do número de modificações e manutenções feitas durante o tempo, dentre outros fatores.

4.4.1 Exemplo 1 – Foco de Visualização da Aderência

O primeiro exemplo faz uso de um projeto fictício, denominado *HotelSystem*, adaptado de PRUDÊNCIO (2008). A equipe do projeto é formada por um engenheiro de software (responsável por gerenciar todo o processo de desenvolvimento), um arquiteto de software e três desenvolvedores. Toda a equipe utiliza um sistema de controle de versões para gerenciar a evolução dos artefatos gerados em suas atividades no processo.

Com base nos documentos de especificação dos requisitos, o arquiteto de software elabora um modelo conceitual que, em sendo inspecionado e aprovado, é repassado para os desenvolvedores, que iniciam a etapa de implementação. O modelo conceitual, elaborado na ferramenta UML2Tools (ECLIPSE FOUNDATION, 2010a), é exibido na Figura 4.3.

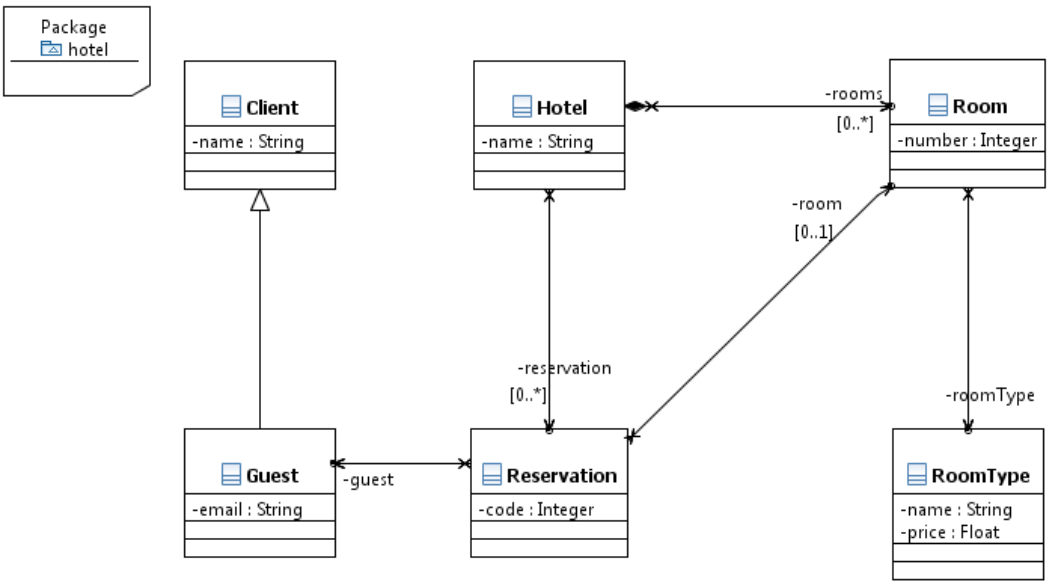
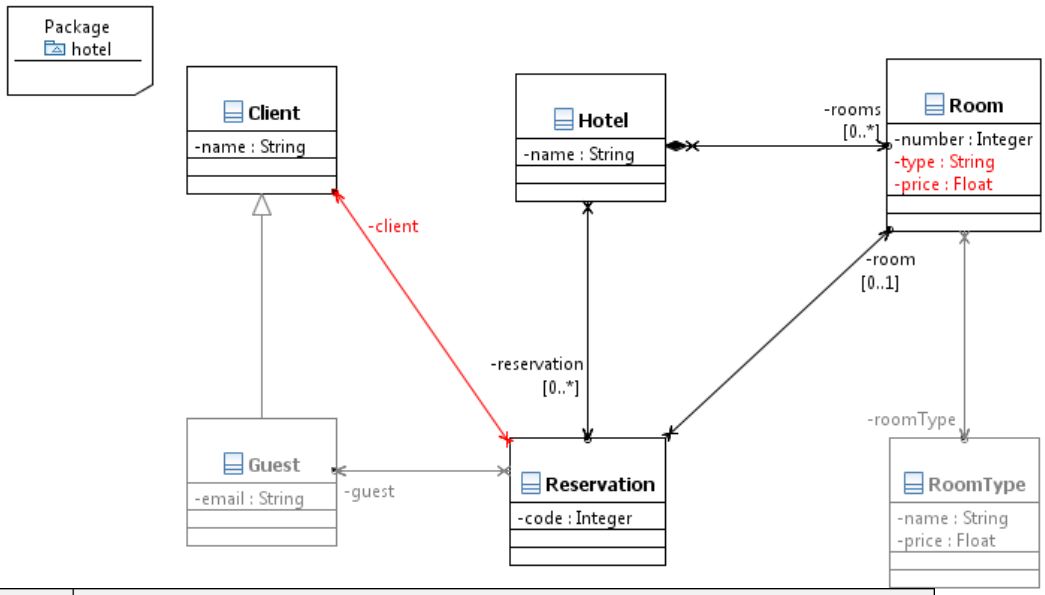


Figura 4.3 – Modelo conceitual do projeto *HotelSystem*

Em um determinado momento do desenvolvimento, um engenheiro de software deseja examinar o andamento do processo de desenvolvimento. Para isto, é efetuado o *check-out* do código fonte do projeto. Em seguida, de posse de uma versão do modelo conceitual (considere-se que este não foi modificado até o momento) e das versões do

código fonte, o engenheiro de software verifica a aderência da versão mais recente da implementação ao que foi inicialmente planejado no modelo conceitual. Ao utilizar a abordagem PREViA, é exibido o modelo representado na Figura 4.4.



Cor	Significado
Vermelha	O elemento consta no modelo emergente, mas **não** consta no conceitual (divergência).
Cinza	O elemento consta no modelo conceitual, mas **não** consta no emergente (ausência).
Preta	O elemento consta tanto no modelo conceitual quanto no emergente.

Figura 4.4 – Visualização de aderência do projeto *HotelSystem*

A partir desta visão, é possível observar que a classe `Guest` não foi implementada (e foi criado um atributo `client`, do tipo `Client`, na classe `Reservation`). Além disso, os atributos da classe `RoomType` (que também não foi implementada) parecem ter sido implementados diretamente na classe `Room`.

Percebe-se, nesta situação, que o modelo conceitual já não representa mais o que está sendo implementado no sistema. Cabe ao engenheiro de software decidir se (i) o modelo conceitual deverá ser atualizado com as informações modificadas pelos desenvolvedores, ou (ii) a equipe de desenvolvimento deverá ser alertada da detecção do desvio do planejamento inicial, efetuando ações corretivas nas próximas versões.

A Figura 4.5 exibe o valor de precisão e cobertura calculado para cada elemento do modelo, segundo o algoritmo descrito no Apêndice A.

Elemento			Resultado da comparação	Precisão	Cobertura	n_{ci}	n_c	n_i	n_{ii}
Pacote "hotel"			Em ambos os modelos	0,7	0,53846154				
	Classe "Guest"		Apenas no conceitual	1	0				
		Herança de "Cliente"	Apenas no conceitual	1	0	0	1	0	0
		Atributo "email"	Apenas no conceitual	1	0	0	1	0	0
	Classe "RoomType"		Apenas no conceitual	1	0				
		Atributo "name"	Apenas no conceitual	1	0	0	1	0	0
		Atributo "price"	Apenas no conceitual	1	0	0	1	0	0
	Classe "Room"		Em ambos os modelos	0,33333333	0,5				
		Atributo "number"	Em ambos os modelos	1	1	1	1	1	0
		Atributo "type"	Apenas no emergente	0	1	0	0	1	0
		Atributo "price"	Apenas no emergente	0	1	0	0	1	0
		Atributo "roomType"	Apenas no conceitual	1	0	0	1	0	0
	Classe "Hotel"		Em ambos os modelos	1	1				
		Atributo "name"	Em ambos os modelos	1	1	1	1	1	0
		Atributo "rooms"	Em ambos os modelos	1	1	1	1	1	0
		Atributo "reservation"	Em ambos os modelos	1	1	1	1	1	0
	Classe "Reservation"		Em ambos os modelos	0,66666667	0,66666667				
		Atributo "room"	Em ambos os modelos	1	1	1	1	1	0
		Atributo "code"	Em ambos os modelos	1	1	1	1	1	0
		Atributo "guest"	Apenas no conceitual	1	0	0	1	0	0
		Atributo "client"	Apenas no emergente	0	1	0	0	1	0
	Classe "Client"		Em ambos os modelos	1	1				
		Atributo "name"	Em ambos os modelos	1	1	1	1	1	0

Figura 4.5 – Valores de precisão e cobertura para elementos do projeto *HotelSystem*

4.4.2 Exemplo 2 – Foco de Visualização da Evolução

O segundo exemplo foi baseado no projeto *Floggy* (FLOGGY OPEN SOURCE GROUP, 2010), um framework de persistência de dados J2ME disponível em código aberto. Os modelos utilizados no exemplo são derivados do código fonte deste projeto, mas os dados de caracterização do cenário são fictícios.

O projeto é desenvolvido por uma equipe cujos membros estão geograficamente distantes, e é composta de um gerente de projetos, dois arquitetos de software e dez desenvolvedores. Toda a equipe utiliza um sistema de controle de versões para gerenciar a evolução dos artefatos gerados em suas atividades no processo.

Um dos membros da equipe (desenvolvedor) viajou de férias por 15 dias e, ao retornar, deseja verificar a situação ou estado atual de um módulo do projeto em questão. De posse da abordagem PREViA, o desenvolvedor seleciona o intervalo de versões que deseja visualizar (no caso, o intervalo compreende desde a última versão na qual trabalhou até a versão mais atual disponível no repositório). Em seguida, a abordagem PREViA exibe um diagrama representando a primeira versão do intervalo (ou seja, a versão na qual ele trabalhou antes de sair de férias), representada na Figura 4.6.

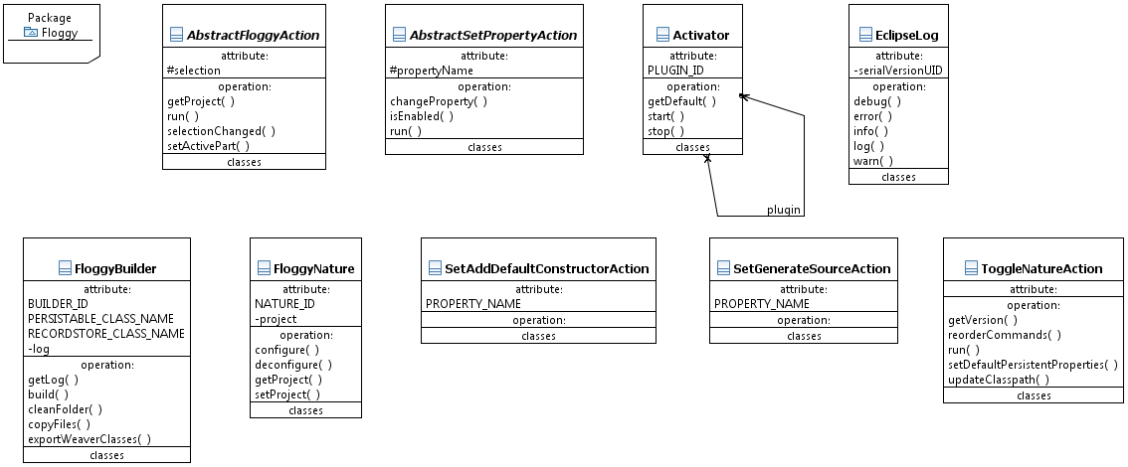
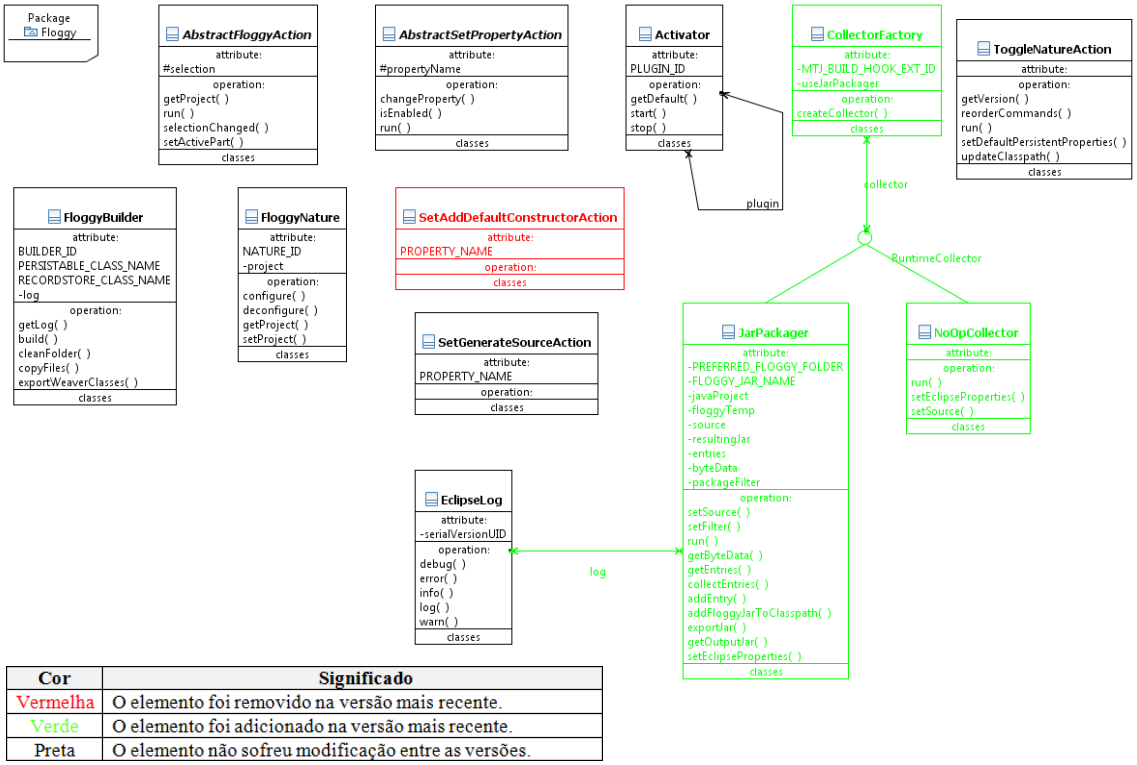


Figura 4.6 – Primeira versão do intervalo selecionado do projeto *Floggy*

Em seguida, por meio de uma linha do tempo, o desenvolvedor visualiza o andamento do projeto, e chega ao instante representado pelo diagrama exibido na Figura 4.7.



Cor	Significado
Vermelha	O elemento foi removido na versão mais recente.
Verde	O elemento foi adicionado na versão mais recente.
Preta	O elemento não sofreu modificação entre as versões.

Figura 4.7 – Visualização de evolução do projeto *Floggy*

A partir desta visão, é possível notar que, desde a última versão na qual o desenvolvedor trabalhou, surgiram três classes (`CollectorFactory`, `JarPackager` e `NoOpCollector`), sendo que duas destas classes implementam uma interface recém-criada (`RuntimeCollector`). Além disso, a classe

`SetAddDefaultConstructorAction` foi removida, e foi criado um atributo `log`, do tipo `EclipseLog`, na classe `JarPackager`. Como neste cenário o desenvolvedor já estava anteriormente alocado ao projeto, é possível que, em função do conhecimento prévio, ele consiga avaliar quais foram as possíveis razões para tal evolução.

Caso o desenvolvedor necessite de mais informações, ele pode procurar o(s) autor(es) das modificações e direcionar questões mais específicas, opcionalmente fazendo uso da visão gerada pela abordagem PREViA, por se tratar de uma visão em mais alto nível quando comparada ao código fonte, e cujo foco está voltado para as diferenças existentes.

Os valores de precisão e cobertura entre estes modelos são 0,522 e 0,973, respectivamente. Como o projeto possui muitos elementos, o cálculo detalhado se encontra na Figura A.1 do Apêndice A.

4.5 Implementação da Abordagem

No que tange à implementação da abordagem PREViA, optou-se por estender a ferramenta EvolTrack (CEPÊDA *et al.*, 2010), um mecanismo de extração e visualização do ciclo de evolução de projetos de software implementado como um *plug-in* do ambiente de desenvolvimento Eclipse (ECLIPSE FOUNDATION, 2010b). Tal escolha se deve ao fato de ser uma ferramenta desenvolvida no contexto do grupo de reutilização da COPPE/UFRJ e pelo fato de os componentes da arquitetura do EvolTrack serem desacoplados, comunicando-se com o *kernel* da ferramenta por meio de interfaces bem definidas (o que permite maior grau de reutilização das funcionalidades da ferramenta). O EvolTrack consiste de um *kernel* e três tipos de componentes: conectores de fontes de dados, transformadores de modelos e conectores de visualização.

O conector de fonte de dados é responsável por obter, em um dado momento do tempo, informações sobre um projeto a partir de uma determinada fonte de dados (sendo que cada tipo de fonte de dados requer um conector apropriado), mapeando-as para um modelo. Este modelo é tratado pelo transformador de modelos, um conector opcional que pode agregar informações adicionais ao modelo (e.g., obtenção e cálculo de medidas para métricas). O conector de visualização tem por objetivo criar uma representação visual dos dados obtidos pelo conector de fonte de dados e transformadores, apresentados em sequência temporal. Por fim, o *kernel* visa orquestrar o fluxo de informações entre conectores e manter o ciclo de vida do projeto.

A Figura 4.8 mostra a arquitetura do mecanismo EvolTrack e os componentes que foram criados/adaptados/utilizados no contexto da abordagem PREViA.

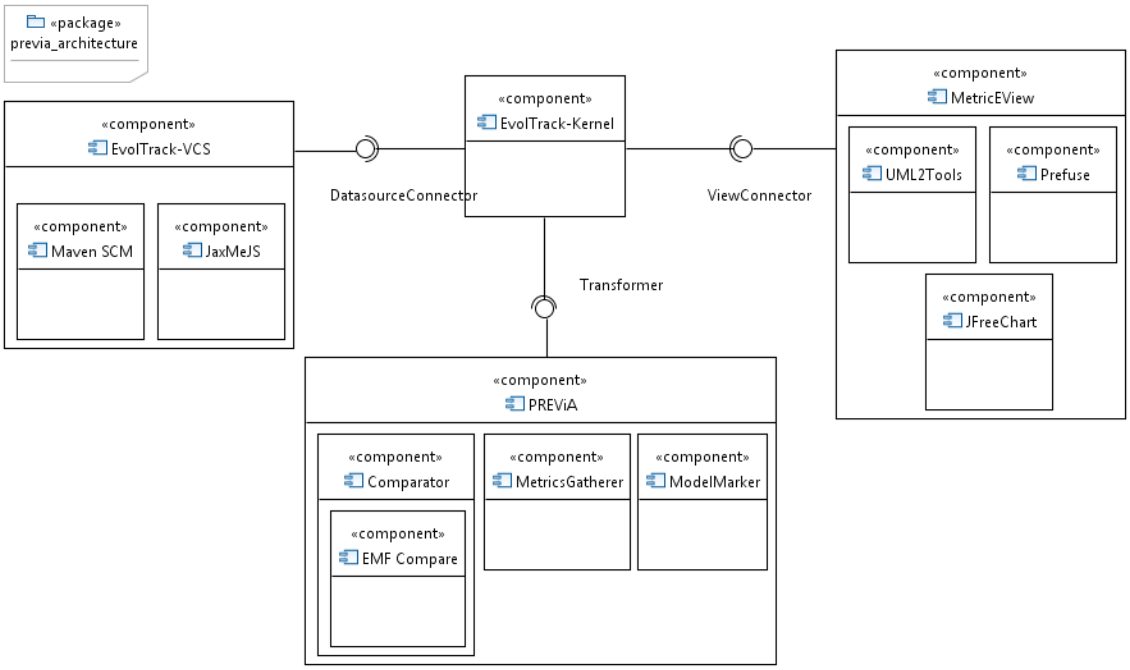


Figura 4.8 – Arquitetura do EvolTrack, incluindo a abordagem PREViA

4.5.1 Obtenção dos Dados

Um dos conectores de fontes de dados do EvolTrack era voltado para repositórios de controle de versões Subversion (COLLINS-SUSSMAN *et al.*, 2010), o que limitava o universo de projetos que poderiam ser analisados com o EvolTrack. Isto motivou a construção de um novo conector que ampliasse as funcionalidades do conector anterior (incluindo manutenções corretivas na engenharia reversa do código fonte) e não fosse acoplado a um sistema de controle de versões específico (SCHOTS *et al.*, 2010b).

Assim, foi criado para o EvolTrack o conector EvolTrack-VCS (SILVA, 2010), que utiliza-se da API Maven SCM (APACHE SOFTWARE FOUNDATION, 2010a) para a comunicação com diversos tipos de repositório. O Maven SCM é um software livre que provê mecanismos para o uso de ferramentas de gerência de configuração, fazendo uso de interfaces genéricas.

A Figura 4.9 mostra as opções de configuração do EvolTrack-VCS (à esquerda), incluindo a identificação do repositório e uma lista de seleção das versões a serem obtidas pelo conector; na configuração da PREViA (à direita), os dados sobre os modelos arquiteturais e o foco de visualização são preenchidos.

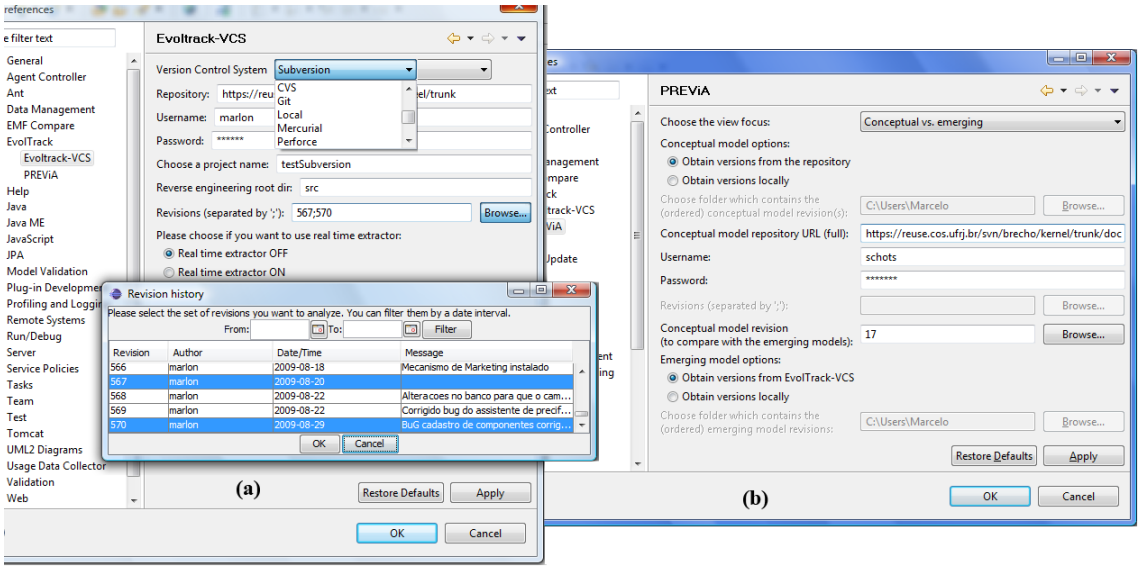


Figura 4.9 – Configurações do EvolTrack-VCS (a) e PREViA (b)

O conector EvolTrack-VCS foi projetado de forma a ser extensível, isto é, existe a possibilidade de se estender estas interfaces para outros sistemas de controle de versões. Assim como no conector anterior, a engenharia reversa fica a cargo do framework JaxMeJS (APACHE SOFTWARE FOUNDATION, 2010b).

Ainda no que diz respeito à obtenção dos dados, a implementação da abordagem PREViA (em função do coletor de métricas e do marcador de modelos) requer que os modelos conceitual e emergente estejam no formato XMI 2.2 (*XML Metadata Interchange*) (OMG, 2007), por se tratar de um padrão utilizado para a troca de informações de metadados. Diversas ferramentas de modelagem UML exportam seus modelos e diagramas no formato XMI, o que facilita o intercâmbio de informações entre elas.

4.5.2 Extração e Transformação das Informações

O transformador de modelos (composto de três módulos, conforme apresentado na Figura 4.8) foi desenvolvido especificamente para o contexto da PREViA, e contempla as principais funcionalidades da abordagem.

O comparador de modelos (*Comparator*) faz uso do framework EMF Compare (ECLIPSE FOUNDATION, 2010c) para obter as semelhanças e diferenças entre modelos e para mapear os dois modelos em um único modelo com as informações das diferenças. Isto é possível por meio dos mecanismos de *match*, *diff* e *merge* presentes no framework (BRUN & PIERANTONIO, 2008).

O coletor de métricas (*MetricsGatherer*), por sua vez, calcula os valores de precisão e cobertura para cada elemento do modelo, fazendo uso do algoritmo apresentado no Apêndice A.

Por fim, o marcador de modelos (*ModelMarker*) atribui estereótipos aos elementos, com os valores de métricas (obtidos pelo coletor de métricas) e com a informação de *diff* (obtida pelo comparador de modelos), caso aplicável.

O framework EMF Compare funciona da seguinte forma: um mecanismo de *match* tenta encontrar correspondências entre os elementos das diferentes versões, com base em métricas de similaridades de nome, de tipo (com relação ao metamodelo a que pertence), de valor (com relação aos atributos do elemento) e de relacionamentos (ECLIPSE FOUNDATION, 2010d).

Em seguida, é gerado um modelo de correspondências (*match model*) que é, essencialmente, uma união dos modelos comparados. Depois disso, um construtor *diff* extrai as diferenças detectadas e gera um modelo de diferenças (*diff model*), que consiste basicamente em adições, exclusões e alterações. Os algoritmos de correspondência e de diferenciação foram inspirados pelo trabalho de XING & STROULIA (2005). A Figura 4.10 mostra a exibição das diferenças entre modelos utilizando a interface gráfica do EMF Compare.

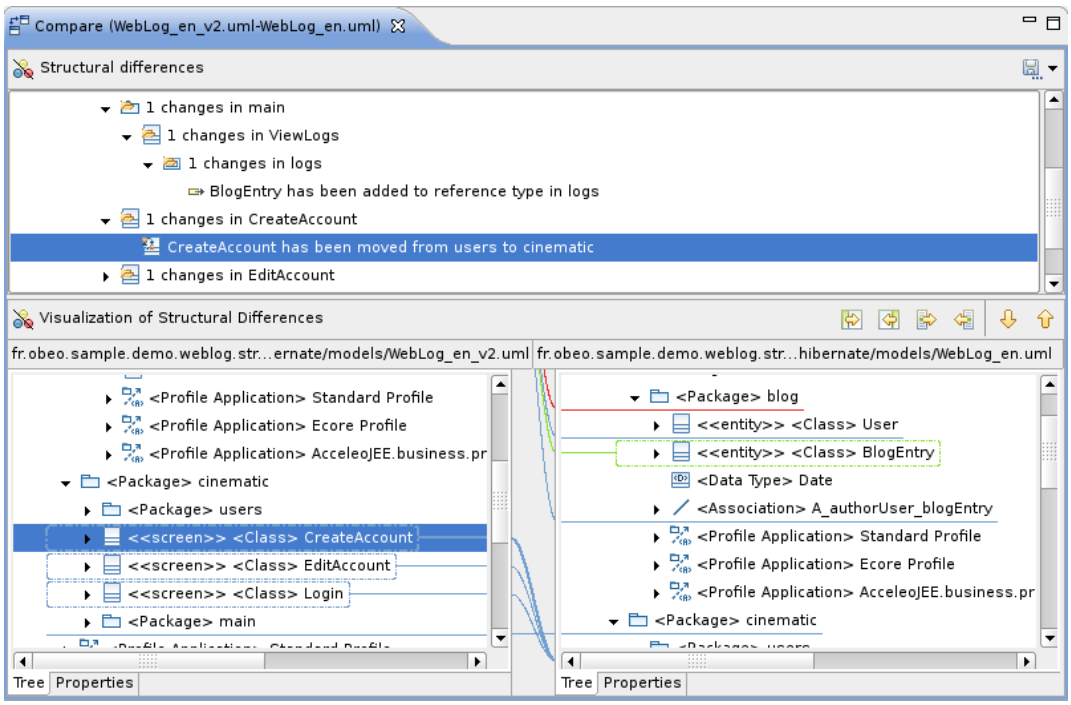


Figura 4.10 – Interface gráfica do EMF Compare

Uma limitação do EMF Compare é a exibição das mudanças na forma de dados estruturados, em uma visão de árvore (*tree-view*), sem outras facilidades gráficas. No entanto, pode-se utilizar o framework para computar as diferenças e repassar a um conector de visualização a tarefa de representar graficamente as diferenças.

4.5.3 Recursos de Visualização

Embora a abordagem PREViA tenha sido projetada para ser aplicável a qualquer modelo ou linguagem de descrição/representação de arquiteturas, a implementação atual da PREViA utiliza-se da versão 2.2 da UML para representar as arquiteturas, como prova de conceito. Para a marcação dos elementos (em termos das diferenças identificadas, das métricas de precisão e cobertura e dos elementos específicos de implementação), são utilizados perfis e estereótipos UML. Isto se deve ao fato de o *kernel* do EvolTrack também fazer uso do metamodelo da UML e do framework UML2Tools, um framework e editor gráfico para a manipulação de modelos UML.

Foi criado para o EvolTrack um conector de visualização denominado MetricEView (SILVA, 2010), com o objetivo de prover a percepção da evolução de métricas marcadas nos modelos. Este conector baseia-se no conceito de visualização multiperspectiva (WU & STOREY, 2000), no qual o usuário faz uso de uma visão principal (adaptada do conector de visualização original do EvolTrack) para tarefas de compreensão em geral e, em determinados contextos ou tarefas, pode lançar mão de visões acessórias para tarefas de exploração adicionais. Este conceito é derivado do conceito de visões múltiplas (BALDONADO *et al.*, 2000), que foi discutido na Seção 3.2.8.

No contexto da abordagem PREViA, as visões do conector são utilizadas para representar, conforme o foco de visualização (aderência ou evolução), as diferenças entre versões dos modelos arquiteturais, bem como a evolução das métricas de precisão e cobertura providas pela PREViA.

Vale ressaltar que, em função da utilização da infraestrutura do EvolTrack estendida pelo conector de visualização MetricEView (SILVA, 2010), as cores utilizadas para representar cada foco de visualização são configuráveis a partir de um conjunto de escalas de cores disponíveis. Em função disto, no entanto, é possível que alguns conjuntos de cores selecionados pelo usuário prejudiquem a percepção das diferenças, devido à possível similaridade entre tais cores.

Outra vantagem trazida por esta infraestrutura é que, com o agrupamento, é possível saber desde os níveis mais altos de abstração quais grupos (no caso, pacotes) possuem elementos ou subgrupos que tenham sido modificados. A Figura 4.11 mostra duas instâncias da visão instantânea (uma das visões do conector MetricEView que acompanha a evolução das métricas no decorrer do tempo), representando, respectivamente, as métricas de precisão e cobertura em um determinado instante.

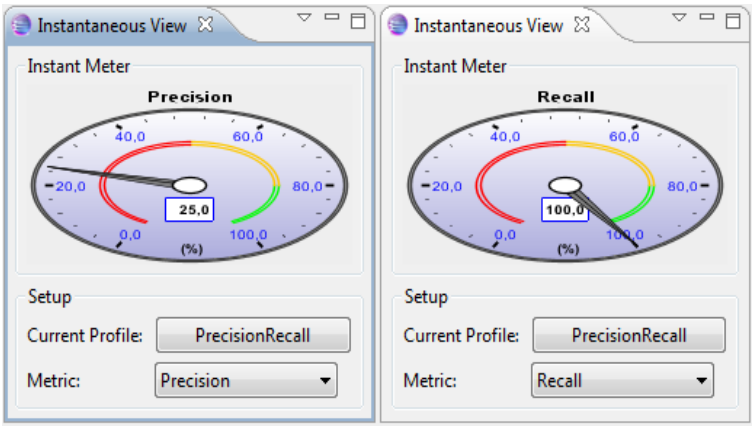


Figura 4.11 – Precisão e cobertura, na visão instantânea do MetricEView

A Figura 4.12 mostra um *screenshot* da ferramenta PREViA (que implementa a abordagem) com alguns de seus recursos de visualização integrados, utilizando o foco de evolução para verificar as alterações do projeto *Floggy* (descrito na Seção 4.4.2).

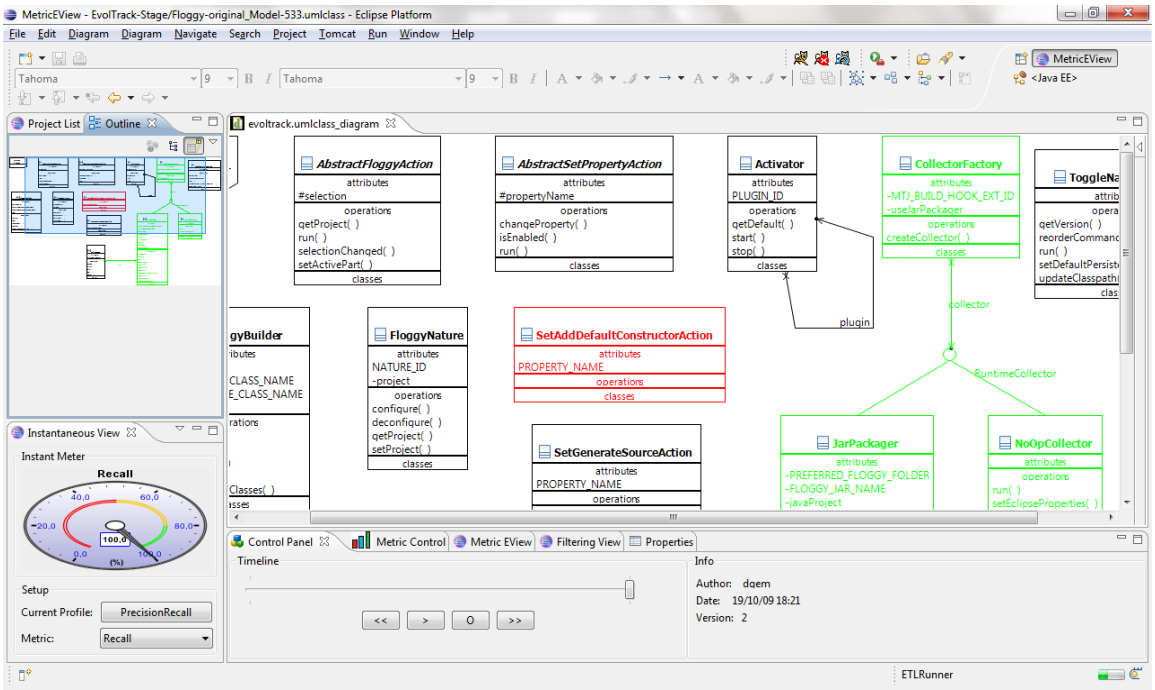


Figura 4.12 – *Screenshot* da ferramenta PREViA: foco de evolução

Na parte superior esquerda é possível visualizar o *minimap*. A visão instantânea de métricas, localizada na parte inferior esquerda, exibe o valor da métrica de cobertura em um dado instante, cujo valor é 100% (equivalente a 1). A parte central contém o diagrama com a visualização das marcações da abordagem PREViA, indicando elementos adicionados e removidos. Por fim, a parte inferior exibe a linha do tempo da ferramenta EvolTrack, que é atualizada dinamicamente a cada nova versão encontrada o repositório (caso a opção de tempo real tenha sido selecionada nas configurações exibidas na Figura 4.9). É possível navegar por entre as versões do modelo por meio desta linha do tempo.

Analogamente, a Figura 4.13 exibe a ferramenta utilizando o foco de aderência. Nesta figura, é apresentada (na parte inferior central) uma visão hierárquica do modelo, com os valores de precisão e cobertura para cada elemento, incluindo a opção de marcar um elemento como específico de implementação. Com exceção desta visão, as demais visões auxiliares utilizadas para este foco podem também ser utilizadas para o foco de evolução. No canto superior esquerdo, é possível visualizar a lista de projetos sendo executados, bem como os diagramas resultantes das comparações realizadas entre versões do projeto.

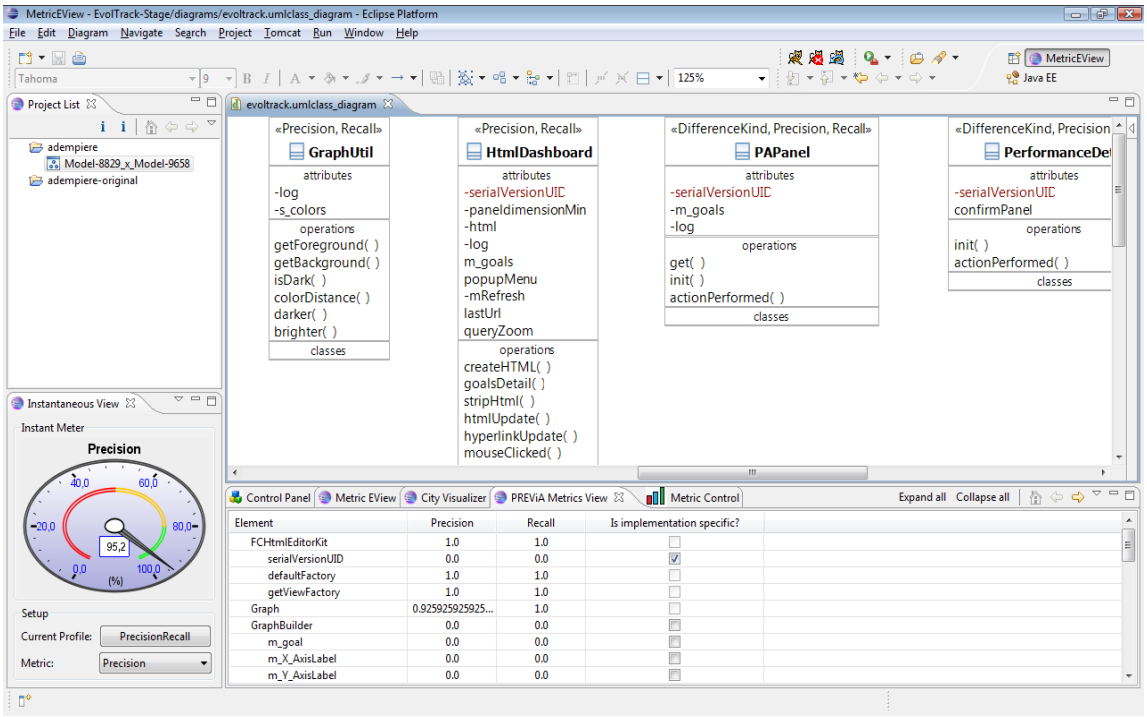


Figura 4.13 – *Screenshot* da ferramenta PREViA: foco de aderência

A Tabela 4.1 enumera as técnicas de visualização apresentadas na Seção 3.2, indicando sua relação com a implementação atual da abordagem PREViA. Cabe

ressaltar que, conforme discutido na Seção 4.2, foram implementadas/reutilizadas as técnicas que acredita-se serem úteis na execução das tarefas para as quais a abordagem PREViA pretende dar suporte.

Tabela 4.1 – Técnicas de visualização, no contexto da abordagem PREViA

Técnica	Aplicação na abordagem PREViA
Zoom	O framework UML2Tools provê o zoom geométrico, permitindo a ampliação e a redução do diagrama visualizado
Filtragem	A filtragem, implementada no conector MetricEView, é baseada em valores de métricas (ou seja, conforme o valor da métrica, o elemento é ou não exibido para o usuário). Além disso, é possível exibir o diagrama em nível de implementação (todos os detalhes são mostrados), análise (são exibidos apenas as literais, as operações e os atributos que são públicos), e uma visão de ocultação de detalhes (todos os compartimentos de classes e interfaces são ocultos).
Agrupamento (*Clustering*)	Atualmente, o agrupamento é feito apenas por pacotes, permitindo a navegação. Esta melhoria foi incorporada ao EvolTrack por meio do conector de fonte de dados EvolTrack-VCS (que contém um algoritmo para este agrupamento), sendo visualizado no conector MetricEView. SHARIF & MALETIC (2009) afirmam que layouts agrupados demonstram melhoria significativa em precisão e rapidez na resolução dos problemas em tarefas de compreensão.
Hierarquias	Diagramas UML possuem uma hierarquia implícita. Não foi feita nenhuma implementação específica neste sentido.
Detalhamento (*Drill-Down*)	Atualmente, o detalhamento existe apenas em nível de expansão de pacotes. O *plug-in* UML2Tools provê esta funcionalidade em dois níveis: é possível gerar um novo diagrama a partir de um pacote ou visualizar o conteúdo do pacote expandido-o, sem sair do diagrama que contém o pacote.
Foco + Contexto	Sobre técnicas de foco + contexto, comumente orientadas à distorção, LEUNG & APPERLEY (1994) afirmam que um tempo de resposta excessivamente longo pode tornar uma interface inutilizável. Segundo estes autores, tal problema pode ser superado pela utilização de hardware dedicado e sistemas de gerenciamento de memória para apoiar a implementação de tais técnicas. No caso do EvolTrack/PREViA, optou-se por não implementar esta técnica, em função de problemas de escalabilidade de visualização já existentes em decorrência do *plug-in* UML2Tools. Ao invés disso, faz-se uso do recurso de visão geral + detalhe.
Visão Geral + Detalhe	O framework UML2Tools possui uma visão de *minimap* navegável, que apresenta uma visão geral para navegação pelos diagramas.

4.6 Discussão

A partir das considerações feitas na Seção 3.3, foram identificados e delineados alguns requisitos considerados relevantes neste contexto (apresentados na Seção 3.5). A seguir, é apresentada uma discussão a respeito da abordagem PREViA quanto ao atendimento a estes requisitos.

R1. A abordagem PREViA faz uso do conector EvolTrack-VCS, que efetua a extração automática de dados a partir de repositórios diversos, com a opção de efetuar esta extração a cada nova versão (modo de execução em tempo real); a comparação entre modelos também é automatizada;

R2. Devido à arquitetura do EvolTrack, cujo *kernel* gerencia as chamadas aos conectores de fontes de dados e de visualização, a geração da visualização ocorre de maneira automática, após cada nova coleta de dados;

R3. Por se tratar de um protótipo acadêmico, o aspecto da escalabilidade não foi o principal foco de implementação; embora não tenha sido feita uma análise quantitativa e integrada da escalabilidade, os componentes utilizados pela abordagem PREViA permitem realizar algumas afirmações a este respeito:

- No que diz respeito à escalabilidade de processamento, testes realizados com o EMF Compare, que é utilizado pela PREViA, indicaram que o framework efetua a comparação de modelos compostos de 256.000 elementos em menos de vinte segundos (BRUN, 2010); comparado ao tempo de execução dos algoritmos deste framework, o tempo gasto pela PREViA na execução de outras instruções (relacionadas à coleta e marcação de valores de métricas) não é significativo; um gargalo identificado no EvolTrack-VCS é o tempo gasto para a conexão e obtenção de dados de alguns repositórios;
- Em função dos novos recursos de visualização recentemente adicionados ao EvolTrack (SILVA, 2010), algumas deficiências na escalabilidade de visualização existentes apontadas em (CEPÊDA *et al.*, 2010) foram parcialmente contornadas; no que diz respeito à quantidade de informações exibidas, foram efetuados testes com até 60 classes em um único pacote/diagrama no EvolTrack com o conector MetricEView, e o diagrama foi exibido corretamente (pode-se fazer uso do *minimap* disponível para navegar pelo diagrama); o agrupamento em pacotes (que

é específico de cada modelo) e o recurso de *drill-down* ajudam a diminuir a poluição visual e a trazer mais semântica à análise, permitindo visualizar apenas o que é relevante em um dado contexto; no entanto, o framework UML2Tools se torna lento e instável (apresentando erros de memória indisponível para processamento) quando há uma grande quantidade de modelos e quando os modelos são muito grandes;

R4. O EvolTrack possui a capacidade de dar suporte a diversos tipos de dados; além dos obtidos pelo conector de fontes de dados, as atuais visualizações permitem lidar com modelos hierárquicos, dados numéricos (como as métricas de precisão e cobertura) e enumerações (como o tipo de diferença identificada entre elementos); além disso, o framework utilizado para efetuar as comparações é genérico, abrangendo diversos tipos de dados;

R5. A abordagem PREViA permite comparar, correlacionar e examinar dados e visões, por meio dos seguintes recursos: (i) utilização do framework genérico EMF Compare para comparação; (ii) uso de visões múltiplas do EvolTrack; (iii) uso do conector MetricEView, que correlaciona visualmente a evolução de duas ou mais métricas;

R6. A abordagem PREViA possui integração com ferramentas existentes por construção, visto que foi construída como um *plug-in* do EvolTrack e se comunica com outros *plug-ins* deste mecanismo. Por também ser um *plug-in* da IDE Eclipse e por atender ao requisito R7, acredita-se que seja possível, sem muito esforço, efetuar a integração (total ou parcial) com novas ferramentas (a integração parcial é possível devido ao baixo acoplamento entre os componentes; por exemplo, uma determinada ferramenta pode requerer apenas a utilização do cálculo de precisão e cobertura da abordagem PREViA);

R7. Os componentes da abordagem PREViA possuem baixo acoplamento devido a suas interfaces bem definidas, o que indica a flexibilidade e capacidade de customização da ferramenta; pode-se, por exemplo, substituir o componente responsável pelo cálculo das métricas por um que dê pesos diferentes a determinados elementos do metamodelo utilizado (neste caso, o conector deve apenas implementar as interfaces requeridas pela abordagem PREViA); a arquitetura flexível e extensível do EvolTrack também permite a substituição dos conectores de fontes de dados e de visualização;

R8. Por meio dos focos de visualização (aderência e evolução) e das métricas de precisão e cobertura integradas aos recursos visuais, a abordagem PREViA visa prover auxílio na identificação de problemas na evolução do software; a avaliação da abordagem descrita na Seção 5.2 provê alguma evidência neste sentido;

R9. No que diz respeito a facilidades na monitoração da execução do projeto (planejado × realizado), a abordagem PREViA possui o foco de visualização da aderência e provê métricas de precisão e cobertura;

R10. Por fim, a integração com o EvolTrack permite o tratamento da dimensão de tempo; a cada nova versão identificada pela fonte de dados sendo utilizada, a abordagem PREViA (caso ativada) inicia o processo de comparação, coleta e marcação dos modelos, que são associados à linha do tempo do EvolTrack, permitindo a navegação por entre diferentes instantes no tempo.

No que diz respeito às características desejáveis a sistemas de visões múltiplas (BALDONADO *et al.*, 2000) apresentadas na Subseção 3.2.8, a abordagem PREViA é analisada a seguir.

- Seleção: para as tarefas de verificação de aderência e compreensão da evolução, são disponibilizadas visões para (i) a identificação das diferenças detectadas entre os modelos, (ii) a seleção de elementos específicos de implementação (que, opcionalmente, também pode ser feita a partir da primeira visão), e (iii) o acompanhamento das mudanças nos valores das métricas;
- Apresentação: as visões associadas à abordagem PREViA são apresentadas simultaneamente (conforme pode ser visto na Figura 4.12), e podem também ser alternadas por meio de um *menu* do Eclipse, caso estas não estejam visíveis em função de alguma necessidade do usuário; é possível também alterar a disposição dos elementos na tela (no entanto, o redimensionamento está sujeito ao tamanho mínimo necessário para cada visão); e
- Interação: as ações efetuadas em uma visão são propagadas para as outras visões, como demonstram os seguintes exemplos: (i) ao navegar pela visão de linha do tempo do EvolTrack, a visão de exibição do diagrama da PREViA e as visões das métricas de precisão e cobertura (vide Figura 4.11) são automaticamente atualizadas; (ii) ao selecionar um elemento específico de implementação, o cálculo da métrica de precisão é refeito para todos os

elementos envolvidos, e a visão que exibe esta métrica é atualizada para refletir a alteração no valor. Em ambos os exemplos, isto é possível por meio da implementação do padrão de projeto Observador (*Observer design pattern*) (GAMMA *et al.*, 1994).

4.7 Análise Comparativa

Na Seção 3.4, foram comparadas e analisadas algumas abordagens que proveem algum apoio à visualização e compreensão da evolução arquitetural. Nesta seção, é feita uma análise comparativa entre estas abordagens e a abordagem PREViA.

No que tange à representação visual, a abordagem de OHST *et al.* (2003) se assemelha à abordagem PREViA, pois também é utilizado um único diagrama para representar elementos comuns a ambas as versões e elementos específicos de cada versão. Em ambas as abordagens, estas diferentes versões têm um significado no tempo, indicando elementos que foram adicionados, modificados ou removidos do modelo (no foco de visualização da evolução). No caso da PREViA, os elementos considerados específicos de cada versão indicam ausências e divergências (no foco de visualização da aderência, não previsto na abordagem de OHST *et al.* (2003).

Em contraste com a ferramenta SAVE, a abordagem PREViA fornece a representação visual da variação da similaridade entre as arquiteturas, utilizando as métricas de precisão e cobertura calculadas para cada par de versões durante o tempo.

Assim como a ferramenta KIELER, a PREViA utiliza-se do framework EMF Compare, sendo que, no caso do KIELER, uma visão adicional foi incorporada à visão original do framework, permitindo visualizar as diferenças nos diagramas (no caso da PREViA, um único diagrama exibe as diferenças. A principal diferença está em a PREViA obter os modelos automaticamente (devido à integração com o EvolTrack) e marcar as diferenças, automatizando o processo de obtenção e comparação dos modelos arquiteturais.

Como não foi possível ter acesso à ferramenta FAME, não se pode afirmar como ocorre a extração dos dados do repositório. A PREViA, integrada ao conector EvolTrack-VCS, permite obter novos dados do repositório em um curto intervalo de tempo (o conector verifica o repositório em intervalos de tempo pré-determinado em busca de novas versões). Adicionalmente, a ferramenta FAME não possui facilidades na monitoração da execução do projeto com relação ao que foi planejado.

A versão atual da ferramenta SourceMiner lida apenas com os dados disponíveis em cada espaço de trabalho do ambiente Eclipse, produzindo apenas resultados locais e individuais, enquanto o EvolTrack (e, consequentemente a abordagem PREViA) permite a obtenção de resultados globais e individuais, por utilizar uma fonte de dados centralizada (mesmo que os dados sejam recuperados a partir de diversos espaços de trabalho individuais) (CEPÊDA *et al.*, 2010). Sendo assim, não é possível até o momento utilizar o SourceMiner para a compreensão da evolução. Além disto, a ferramenta não possui nenhuma visão que permita explicitamente a comparação entre o que foi planejado e o que foi realizado.

Por fim, conforme indicado na Seção 3.5, a maioria das abordagens apresentadas não exibe os modelos em sequência temporal, impedindo a navegação entre eles. Tal funcionalidade é provida por meio da integração entre o EvolTrack e a PREViA.

A Tabela 4.2 sumariza a discussão apresentada na seção anterior e inclui a abordagem PREViA integrada aos demais *plug-ins* do EvolTrack na análise comparativa apresentada na Tabela 3.1. As células realçadas indicam que houve alguma contribuição da abordagem PREViA para o requisito em questão; as células destacadas em pontilhado representam requisitos que não eram contemplados na versão original do EvolTrack e que são contribuições diretas da abordagem PREViA.

Novamente, cabe ressaltar que as abordagens comparadas se relacionam de alguma forma à abordagem PREViA, mas podem diferir em diversos aspectos (foco principal de visualização, tipo de tarefa para a qual provê suporte etc.), o que pode resultar em alguns dos requisitos não serem atendidos.

Tabela 4.2 – Quadro comparativo das abordagens, incluindo a abordagem PREViA

Requisitos	**Abordagens**					
	OHST *et al.* (2003)	KNODEL *et al.* (2006a)	WENZEL (2008)	SCHIPPER *et al.* (2009)	CARNEIRO *et al.* (2010)	PREViA (SCHOTS *et al.*, 2010b)
R1	+	–	?	–	–	+
R2	?	+	+	+	+	+
R3	?	+	+	?	?	+/–
R4	–	+	+	+	+	+
R5	+	+	+	+	+/–	+

Requisitos	Abordagens					
	OHST *et al.* (2003)	KNODEL *et al.* (2006a)	WENZEL (2008)	SCHIPPER *et al.* (2009)	CARNEIRO *et al.* (2010)	PREViA (SCHOTS *et al.*, 2010b)
R6	+	–	+	+	+	+
R7	–	+	+	?	+	+/–
R8	–	+/–	+	–	–	+
R9	–	–	–	–	–	+
R10	+/ –	–	+	–	–	+

+ Requisito atendido
+/– Requisito parcialmente atendido
– Requisito não atendido
? Não foi possível avaliar (fontes de informações não encontradas)

4.8 Considerações Finais

Neste capítulo, foi introduzida a abordagem PREViA, proposta nesta dissertação, junto a seus conceitos e técnicas, como a utilização das métricas de precisão e cobertura, e exemplos de utilização. A infraestrutura na qual o protótipo da abordagem está inserido foi detalhada em paralelo a alguns detalhes da implementação, com ênfase nos recursos de visualização. Os trabalhos relacionados foram discutidos, e suas diferenças com relação à abordagem proposta nesta dissertação foram apresentadas.

O próximo capítulo trata da avaliação da abordagem proposta. Nele, são apresentados os detalhes do planejamento e execução dos estudos conduzidos, bem como os resultados obtidos e as ameaças à validade.

CAPÍTULO 5 - AVALIAÇÃO DA ABORDAGEM

5.1 Introdução

De acordo com MAFRA & TRAVASSOS (2006), a engenharia de software baseada em evidências (ou engenharia de software experimental) permite a caracterização de uma tecnologia em uso, sendo possível determinar, em certas circunstâncias, o seu funcionamento.

Seguindo estes princípios, a abordagem PREViA foi avaliada por meio da condução de dois estudos, sendo um deles alinhado ao cenário discutido nesta dissertação, e o outro voltado para a utilização da PREViA no contexto de educação em engenharia de software (visando verificar a aplicabilidade da abordagem neste cenário).

Este capítulo apresenta, além desta seção introdutória, os detalhes dos estudos sobre a utilização da abordagem PREViA, tanto no contexto para o qual a abordagem foi originalmente projetada (Seção 5.2) quanto no contexto de educação em engenharia de software (Seção 5.3). Estas duas seções contêm em suas subseções, para cada estudo, o planejamento, a execução, a análise dos dados e algumas ameaças à validade identificadas. Por fim, a Seção 5.4 conclui este capítulo com as considerações finais.

5.2 Avaliação da Abordagem no Contexto de Projetos de Software

Este estudo consiste em analisar a abordagem PREViA no que tange à percepção das divergências/diferenças entre modelos durante a execução de tarefas de desenvolvimento de software. Para isto, as atividades devem ser efetuadas (i) sem informações visuais e (ii) com a utilização de alguns recursos visuais da abordagem PREViA. Em suma, pretende-se verificar se o aspecto da visualização foi diferencial durante a análise das informações de aderência e evolução, permitindo obter melhor identificação dos elementos divergentes/diferentes e, indiretamente, permitindo melhor compreensão do significado da alteração realizada, no contexto das tarefas de desenvolvimento.

A partir dos dados de execução deste estudo, é feita uma análise quantitativa dos resultados e, em seguida, uma análise qualitativa, na qual os participantes são solicitados a descrever aspectos positivos e negativos da abordagem (caso haja), bem

como dar sugestões de melhoria e apontar possíveis problemas na execução das atividades.

A identificação dos objetivos deste estudo foi feita segundo a abordagem GQM (BASILI *et al.*, 1994), conforme pode ser visto na Tabela 5.1.

Tabela 5.1 – Identificação dos objetivos do estudo

Perguntas	Respostas
Objeto do estudo (o que será analisado?)	a abordagem PREViA
Propósito (por que / para quê o objeto será analisado?)	caracterizar/avaliar
Foco de qualidade (que propriedades do objeto serão analisadas?)	precisão, cobertura e eficiência, comparadas à execução das mesmas atividades sem a aplicação da abordagem
Ponto de vista (quem utilizará os dados coletados?)	engenheiros de software
Ambiente (em que ambiente será realizada a análise?)	projetos de desenvolvimento de software

Assim, este estudo pode ser definido da seguinte forma:

Analisar	a abordagem PREViA
Com o propósito de	avaliar
Com respeito a	precisão, cobertura e eficiência, em comparação à execução das mesmas atividades sem a aplicação da abordagem
Do ponto de vista de	engenheiros de software
No contexto de	projetos de desenvolvimento de software

A precisão, que é um indicador de exatidão, se refere ao número de respostas corretas dadas às atividades, com relação ao número de respostas dadas. Seu valor é fornecido pela seguinte fórmula:

$$Precisão = \frac{número\ de\ respostas\ corretas\ dadas}{número\ de\ respostas\ dadas}$$

Já a cobertura, que é um indicador de eficácia e completude, se refere ao número de respostas dadas às atividades, com relação ao número de respostas corretas esperadas. Seu valor é fornecido pela seguinte fórmula:

$$Cobertura = \frac{número\ de\ respostas\ corretas\ dadas}{número\ de\ respostas\ corretas}$$

Por fim, no contexto deste estudo, o termo eficiência se refere ao esforço gasto durante a execução das atividades do estudo, e é fornecido pela seguinte fórmula:

$$Eficiência = \frac{número\ de\ respostas\ dadas}{tempo\ gasto}$$

Para analisar o indicador eficiência, é necessário considerar todas as respostas (incluindo-se também as incorretas e duplicatas) encontradas (GOMES *et al.*, 2009). O oráculo foi gerado com base nas respostas de dois especialistas no domínio, e foi validado por meio das respostas dadas pelos participantes.

Optou-se por não obter o tempo gasto por tarefa (hora de início e de término), pois, além de o esforço de execução do estudo ser maior, a unidade de tempo não é adequada. É possível que uma tarefa dure menos que um minuto, e a utilização de segundos como unidade de comparação pode comprometer o estudo, em função da dificuldade de obter com precisão a quantidade de segundos gasta em uma tarefa. Devido a isto, é solicitado do participante o tempo de início e de término para cada grupo de tarefas (tais grupos estão descritos na Subseção 5.2.1).

Conforme sugerido por MAFRA & TRAVASSOS (2006), o planejamento e os instrumentos utilizados no estudo foram revisados por dois pesquisadores que não possuem interesse direto nos resultados do estudo, visando minimizar a presença de viés. Além disso, um terceiro pesquisador executou o estudo com base no planejamento.

A seleção destes pesquisadores foi baseada em conveniência (disponibilidade para realizar a revisão e conhecimento prático nas áreas envolvidas do estudo). Dos dois pesquisadores que revisaram o planejamento e os instrumentos utilizados, um deles possui conhecimento em arquitetura de software, enquanto o outro possui experiência prática em experimentação. O terceiro pesquisador (que executou o estudo) possui experiência em atividades de verificação. Durante a revisão, foram apresentadas críticas e sugestões de melhoria que contribuíram para ajustes no planejamento e para aperfeiçoamento dos instrumentos.

5.2.1 Planejamento do Estudo

Neste estudo, pretende-se avaliar a abordagem PREViA por meio de seus focos de visualização (a saber: aderência e evolução). Na avaliação do foco de aderência, os participantes recebem uma versão W do modelo conceitual e uma versão X do modelo emergente (sendo W e X identificadores de versão). Já na avaliação do foco de evolução, os participantes recebem uma versão Y do modelo emergente e uma versão Z

deste mesmo modelo (sendo Y e Z identificadores de versão, onde Z é uma versão mais recente que Y).

De forma a facilitar a imersão do participante no contexto do estudo, é criada uma situação fictícia, na qual o participante havia sido contratado como engenheiro de software por uma empresa de desenvolvimento de software, sendo alocado inicialmente em atividades de inspeção. Seus objetivos são (i) verificar nos projetos a aderência do código fonte (desenvolvido por programadores) ao modelo elaborado por projetistas/arquitetos de software, com o intuito de manter a consistência entre estes artefatos, e (ii) analisar a evolução do código implementado pela equipe de desenvolvimento em um dado intervalo de tempo.

Os participantes devem preencher um formulário de consentimento em participar do estudo (Seção B.1 do Apêndice B), o que compreende uma etapa preliminar do estudo. Nesta etapa, também é solicitado que os participantes preencham um questionário de caracterização (Seção B.2 do Apêndice B).

As respostas ao questionário de caracterização servem para avaliar se a divisão dos participantes resultou em grupos homogêneos. Esta divisão é feita de forma "cega", ou seja, não se sabe quais participantes estão alocados a determinado grupo.

Para isto, é solicitado que cada participante escreva seu nome em frente a um dos números disponíveis em uma lista à parte, sendo que este número passa a ser sua identificação durante todas as etapas do estudo. Isto impede o favorecimento de um determinado cenário da avaliação por meio da alocação dos participantes, bem como evita que a correção das atividades leve em conta o participante que as realizou. A lista que identifica os participantes só é consultada durante a etapa de análise, caso necessário.

As duas etapas seguintes são compostas pelos modelos descritos no início desta seção e por formulários que consistem de um conjunto de tarefas a serem realizadas, visando avaliar os focos de aderência (Seção B.3 do Apêndice B) e evolução (Seção B.4 do Apêndice B). Estas tarefas são divididas em três grupos, baseados no trabalho de KNODEL *et al.* (2006b), a saber:

- **Tarefas de Filtragem (TF)**: Este grupo consiste de tarefas mais simples, de forma que, se algum participante não estiver apto a realizá-las, o mesmo deve ser desconsiderado na análise dos resultados (por indicar possíveis problemas no entendimento da abordagem PREViA ou das tarefas do experimento). Exemplos destas tarefas são:

    - Quais classes compõem o modelo conceitual? E o modelo emergente? Escreva o nome de cada uma delas. **[Foco: aderência]**
    - Qual(ais) classe(s) possui(em) o menor número de métodos na versão 345? E na versão 378? **[Foco: evolução]**
- **Tarefas Básicas (TB)**: São tarefas que podem ser realizadas por meio de fatos extraídos da visualização. Neste grupo de tarefas, o aspecto da percepção é tratado. Exemplos destas tarefas são:
    - Qual(ais) classe(s) e/ou relacionamento(s) foi(foram) implementado(s) – isto é, está(ão) no modelo emergente –, mas não foi(foram) planejado(s) no modelo conceitual? **[Foco: aderência]**
    - Qual(ais) classe(s) e/ou relacionamento(s) surgiu(iram) na versão 378, comparada à versão 345? **[Foco: evolução]**
- **Tarefas de Assimilação (TA)**: Este grupo consiste de tarefas mais difíceis e complexas, exigindo que o participante utilize o conhecimento prévio e o raciocínio de forma a obter o entendimento e interpretar as informações contidas nos instrumentos da avaliação. Exemplos destas tarefas são:
    - Considerando o requisito RF02 – "O sistema deve permitir a inclusão dos sintomas durante cada anamnese" –, você acredita que ele tenha sido implementado de alguma forma, com base nas classes presentes no modelo emergente? Em caso positivo, qual(ais) classe(s) está(ão) envolvida(s) na implementação do requisito? **[Foco: aderência]**
    - Analise todas as modificações realizadas entre as versões 345 e 378. Em sua opinião, qual(ais) mudança(s) nos requisitos pode(m) ter levado a tais modificações? **[Foco: evolução]**

Os participantes não foram comunicados da existência e do objetivo desta divisão de tarefas em grupos, de forma a não influenciar a execução das atividades.

Por fim, na última etapa, é dado um questionário *follow-up* (Seção B.5 do Apêndice B) para a obtenção de informações qualitativas acerca do estudo, incluindo a percepção e considerações dos participantes sobre a aplicação da abordagem PREViA.

Sabe-se que a ordem de execução das atividades (aderência e evolução) e o apoio para a execução das mesmas (com PREViA e sem PREViA) podem exercer influência nos resultados. Visando reduzir o impacto, foi feito o produto cartesiano[3] entre as atividades e o apoio a ser utilizado, resultando em 8 possibilidades, reunidas em 4 grupos com 2 subgrupos cada, de modo que cada grupo faz uso dos mesmos artefatos, variando em seus subgrupos apenas a ordem de execução das atividades.

Adicionalmente, a distribuição em grupos e subgrupos, adaptada do trabalho de DIAS NETO (2009), permite efetuar as seguintes análises:

- Os grupos 2 e 3 permitem analisar o efeito da utilização da abordagem PREViA;
- Os grupos 1 e 4 são adequados para avaliar o efeito de aprendizagem na avaliação (se a segunda atividade é melhor que a primeira); e
- O grupo 4 é apropriado para avaliar se a utilização da abordagem PREViA traz benefícios, sem o viés da comparação do resultado de uma atividade sem uso da abordagem com outra atividade que faz uso da PREViA.

A distribuição das atividades realizadas por etapa é apresentada na Tabela 5.2.

Tabela 5.2 – Distribuição das tarefas e artefatos entre os participantes

Grupo	Subgrupo	Etapa preliminar	1ª etapa	2ª etapa	Etapa final
1	A	Questionário de caracterização	Aderência (sem PREViA)	Evolução (sem PREViA)	Questionário *follow-up*
	B		Evolução (sem PREViA)	Aderência (sem PREViA)	
2	C		Aderência (sem PREViA)	Evolução (com PREViA)	
	D		Evolução (com PREViA)	Aderência (sem PREViA)	
3	E		Aderência (com PREViA)	Evolução (sem PREViA)	
	F		Evolução (sem PREViA)	Aderência (com PREViA)	
4	G		Aderência (com PREViA)	Evolução (com PREViA)	
	H		Evolução (com PREViA)	Aderência (com PREViA)	

[3] Na Matemática, dados dois conjuntos X e Y, o produto cartesiano (ou produto direto) dos dois conjuntos (denotado por X × Y) é o conjunto de todos os pares ordenados cujo primeiro elemento pertence a X e o segundo pertence a Y.

5.2.2 Execução do Estudo

Os participantes do estudo, selecionados por conveniência, são alunos da Universidade do Estado do Rio de Janeiro (UERJ) no nível de graduação que cursaram a disciplina Engenharia de Software. Alguns deles também cursaram a disciplina Modelagem de Sistemas na mesma universidade. Não houve nenhum tipo de compensação para os participantes.

No total, 35 alunos participaram do estudo. Um benefício de se ter este número de participantes é que, segundo JURISTO & MORENO (2001), trata-se de um número grande de observações ($n \geq 30$) para um estudo em engenharia de software, o que permite realizar análises estatísticas mais aprimoradas.

A Tabela 5.3 apresenta alguns dados de caracterização dos participantes deste estudo. Estes dados foram obtidos por meio do formulário apresentado na Seção B.2 do Apêndice B.

Tabela 5.3 – Caracterização dos participantes

Grupo	ID	Grau de Experiência			
		Modelagem de Sistemas	UML	Inspeção de Software	Java
1	1	4	2	1	2
	2	2	2	4	3
	5	1	2	1	2
	6	1	1	4	1
	14	2	4	1	2
	15	4	4	1	2
	17	2	2	2	2
	18	3	3	1	3
	24	2	2	3	2
	25	2	2	1	2
2	3	0	1	0	4
	7	4	2	2	2
	9	1	1	0	2
	16	2	3	1	2
	20	4	2	1	2
	30	3	1	2	4
	31	1	1	0	4
	33	2	2	1	1
3	4	2	2	2	2
	11	2	4	2	4
	12	2	1	1	2
	19	3	4	4	4
	21	2	2	4	1
	28	0	0	0	0
	29	2	1	0	1
	32	1	1	1	1

Grupo	ID	Grau de Experiência			
		Modelagem de Sistemas	UML	Inspeção de Software	Java
4	8	1	4	0	1
	10	2	4	0	1
	13	4	4	1	4
	22	2	4	4	2
	23	1	0	2	1
	26	1	2	1	3
	27	4	4	4	4
	34	1	3	0	2
	35	3	3	2	3

Os números nesta tabela indicam o grau de experiência do participante em cada área, de acordo com a seguinte escala: 0 = Nenhum; 1 = Estudado em aula ou em livro; 2 = Praticado em projetos em sala de aula; 3 = Utilizado em projetos pessoais; 4 = Utilizado em projetos na indústria.

A Figura 5.1 indica o nível de experiência dos participantes com o paradigma de orientação a objetos (OO). Por se tratar de uma questão de múltipla escolha, a soma dos valores ultrapassa 100%, visto que um mesmo participante pode se encaixar em mais de uma situação.

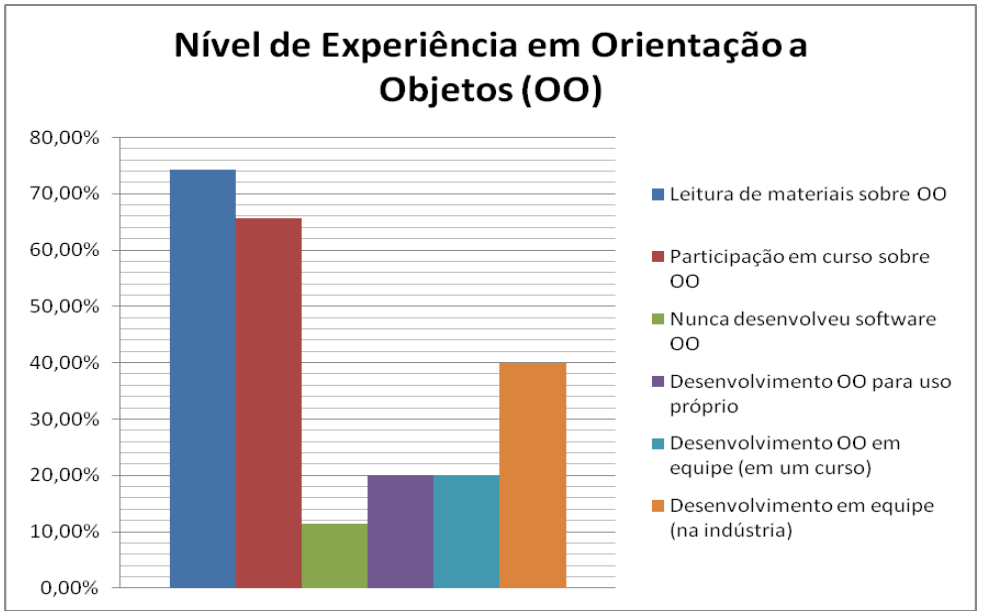


Figura 5.1 – Nível de experiência dos participantes em orientação a objetos

Vale notar que nem todos os participantes fizeram curso de orientação a objetos. No entanto, conceitos teóricos e práticos de orientação a objetos foram explorados no decorrer das disciplinas de modelagem de sistemas e engenharia de software. Outro ponto que merece atenção é que 40% dos participantes possui experiência em

desenvolvimento OO na indústria, embora por um período de tempo de aproximadamente um ano e meio de duração, em média.

Além disso, apenas quatro participantes afirmaram estar familiarizados com o domínio médico-hospitalar, que foi utilizado no estudo (um destes participantes afirmou ser muito familiar com este domínio). No entanto, observou-se que estes participantes estavam alocados em grupos distintos entre si, isto é, tal circunstância não resultou em desbalanceamento entre os grupos.

A Tabela 5.4 mostra a alocação dos alunos nos subgrupos (as letras A e E representam, respectivamente, as etapas de Aderência e Evolução, enquanto as letras SP e CP representam se a etapa é realizada Com PREViA ou Sem PREViA).

Tabela 5.4 – Alocação dos alunos nos subgrupos

Subgrupo	1ª etapa	2ª etapa	Identificadores dos participantes					
A	**ASP**	**ESP**	1	5	6	14		
B	**ESP**	**ASP**	2	15	17	18	24	25
C	**ASP**	**ECP**	3	7	16	30		
D	**ECP**	**ASP**	9	20	31	33		
E	**ACP**	**ESP**	4	11	21	28		
F	**ESP**	**ACP**	12	19	29	32		
G	**ACP**	**ECP**	22	23	26	34		
H	**ECP**	**ACP**	8	10	13	27	35	

Como o estudo contou com 35 participantes, dois subgrupos (B e H) contêm participantes a mais. Além disso, em decorrência de um equívoco na distribuição dos formulários, o subgrupo B possui maior quantidade de participantes.

Antes da realização do estudo, foi ministrado para os participantes um breve treinamento sobre modelos conceituais, modelos emergentes e divergências entre modelos arquiteturais, de forma a familiarizá-los com os conceitos envolvidos no estudo e na abordagem. É importante ressaltar que, durante as disciplinas, os alunos tiveram a oportunidade de trabalhar com a elaboração de diagramas e com tarefas de verificação de consistência, o que fez com que este treinamento fosse suficiente para o estudo.

5.2.3 Análise dos Dados

Após a coleta e contabilização dos dados de execução do estudo, é feita uma análise quantitativa (por meio dos valores das variáveis indicadas na Seção 5.2) e uma análise qualitativa (por meio das respostas ao questionário *follow-up*).

5.2.3.1 Análise Quantitativa

De forma a avaliar os focos de visualização da aderência e da evolução, são utilizadas as respostas dos formulários B.3 e B.4 do Apêndice B, respectivamente. No contexto deste estudo, é utilizado o software JMP® 9.0.0 (SAS INSTITUTE INC., 2011) para apoiar na análise dos dados estatísticos. O nível de significância utilizado para todas as análises é igual a 0,05 (5%).

Foram extraídos a média aritmética e o desvio padrão de cada variável, a fim de averiguar a percepção dos participantes acerca dos focos de visualização.

A análise sobre a distribuição normal dos dados foi realizada utilizando o método Levene. Quando os dados possuem distribuição normal, deve ser utilizado um método paramétrico para análise de variância – no contexto desta análise, é empregado o modelo estatístico ANOVA (*ANalysis Of VAriance*, ou Análise de Variância). Caso os dados não possuam distribuição normal, um método não paramétrico é utilizado – neste caso, o teste de Wilcoxon. Estes métodos visam determinar se a diferença entre as médias dos valores das variáveis nos diferentes grupos é estatisticamente significante. Os testes de normalidade e de análise de variância encontram-se no Apêndice D.

Por meio dos resultados das tarefas de filtragem, alguns participantes foram desconsiderados na análise dos dados para cada foco de visualização. Como o número de participantes por grupo é pequeno, optou-se por filtrar os participantes que não atingissem 40% dos valores de precisão ou de cobertura, o que equivale a dois participantes no foco de aderência e três participantes no foco de evolução. A diferença mínima do resultado destes participantes com relação aos demais é superior a 10%. Os participantes em questão também tiveram um baixo desempenho na maior parte das tarefas, o que corrobora com o objetivo das tarefas de filtragem.

Também é feita uma análise de *outliers*[4] para cada variável apresentada na Seção 5.2, por meio de gráficos *box plot*. Nestes casos, os participantes considerados *outliers* para uma determinada variável são desconsiderados apenas da análise da variável em questão. Em todos os casos, o processo de eliminação de *outliers* foi repetido até que

[4] Em Estatística, um *outlier* é um ponto/dado numericamente distante quando comparado aos demais pontos/dados de um conjunto observado.

não houvesse mais *outliers*; para fins de simplificação, apenas os gráficos inicial e final deste processo serão mostrados.

Precisão

Ao efetuar a análise da variável de precisão, observou-se que inicialmente três participantes são considerados *outliers* na avaliação da aderência, conforme representado na Figura 5.2. No total, oito participantes foram considerados *outliers* na avaliação da aderência. Analogamente, na avaliação da evolução, apenas um participante é considerado *outlier* (também exibido na Figura 5.2).

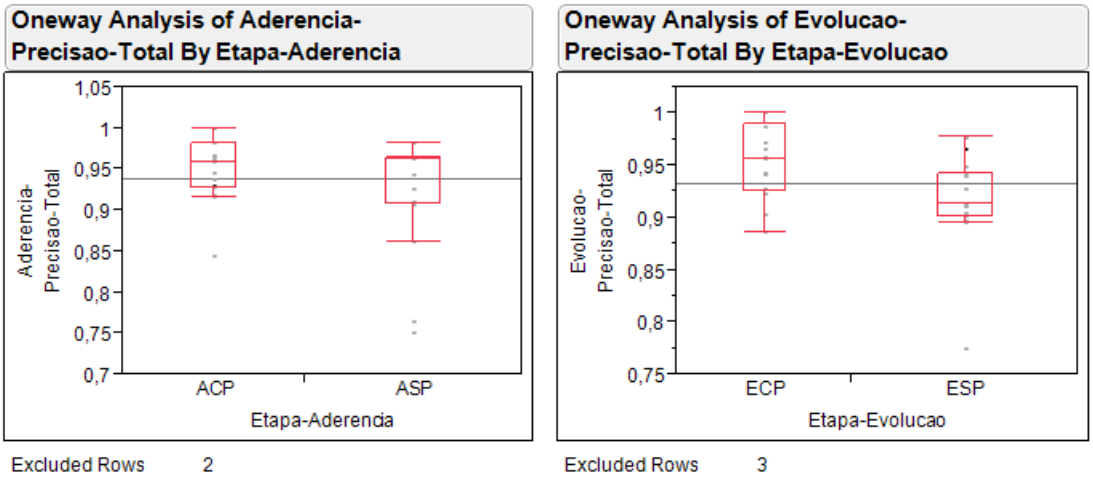


Figura 5.2 – *Box plot* da variável precisão, focos de aderência e evolução (antes da eliminação dos *outliers*)

Os *outliers* foram removidos da análise dos respectivos focos para a variável de precisão. A Figura 5.3 mostra os gráficos *box plot* após a remoção dos *outliers*.

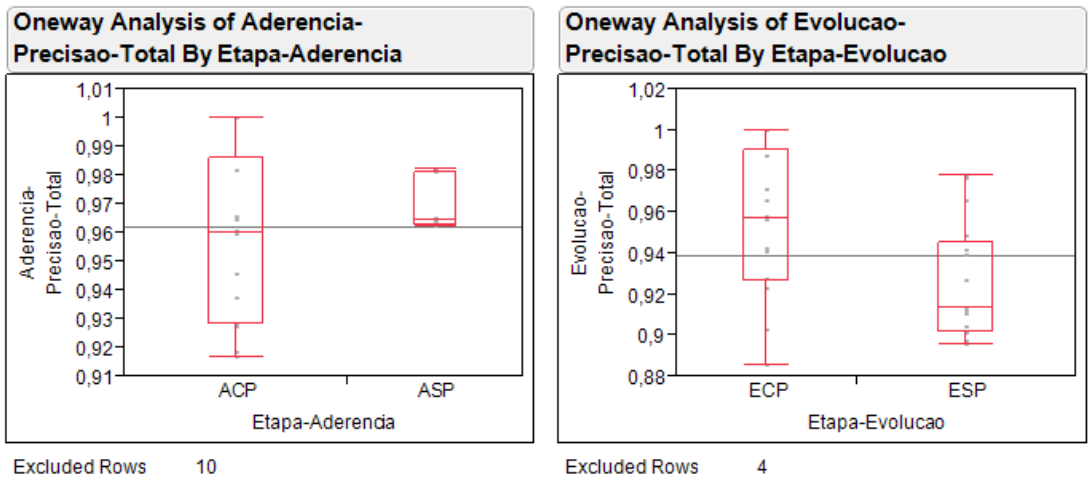


Figura 5.3 – *Box plot* da variável precisão, focos de aderência e evolução (após a eliminação dos *outliers*)

A Tabela 5.5 e a Figura 5.4 exibem os valores da média e do desvio padrão para cada foco de visualização e alternativa.

Tabela 5.5 – Média e desvio padrão da variável precisão (geral)

Foco	Alternativa	Média	Desvio Padrão
Aderência	Sem PREViA	96,87%	0,85%
	Com PREViA	95,75%	2,99%
Evolução	Sem PREViA	92,67%	2,80%
	Com PREViA	95,42%	3,63%

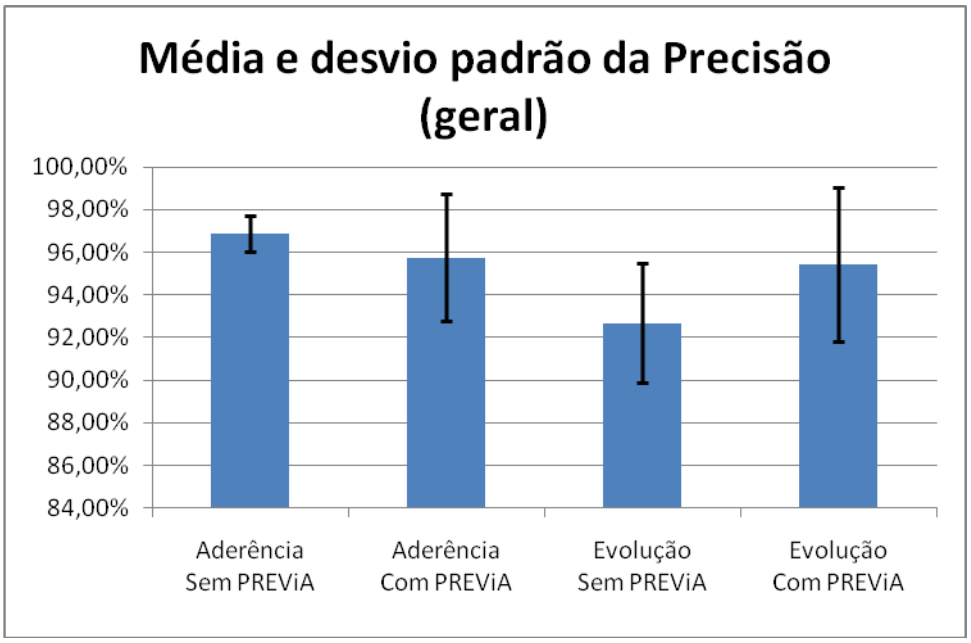


Figura 5.4 – Média e desvio padrão da variável precisão (geral)

No foco de aderência, é possível observar que, de modo geral, o nível de precisão na avaliação da aderência sem a utilização da PREViA é ligeiramente maior que o nível de precisão fazendo uso da abordagem. Já no foco de evolução, a diferença entre as médias aumenta, indicando que a PREViA pode possibilitar maior precisão na execução de atividades com este foco.

Visando compreender melhor este cenário, no contexto da variável de precisão, é efetuada a análise dos dados por grupos de tarefas (de filtragem, básicas e de assimilação). Na avaliação da aderência, 8 participantes são considerados *outliers*, sendo 4 nas tarefas de filtragem e 4 nas tarefas básicas, enquanto na avaliação da evolução, apenas 1 participante é considerado *outlier*, nas tarefas básicas. Os resultados dos *outliers* foram removidos da análise dos respectivos focos e grupos de tarefas para a variável de precisão. Os gráficos utilizados na exclusão dos *outliers* são apresentados no Apêndice D (Figura D.1, Figura D.2 e Figura D.3).

As figuras seguintes (Figura 5.5, Figura 5.6 e Figura 5.7) exibem os gráficos finais após a eliminação dos *outliers*.

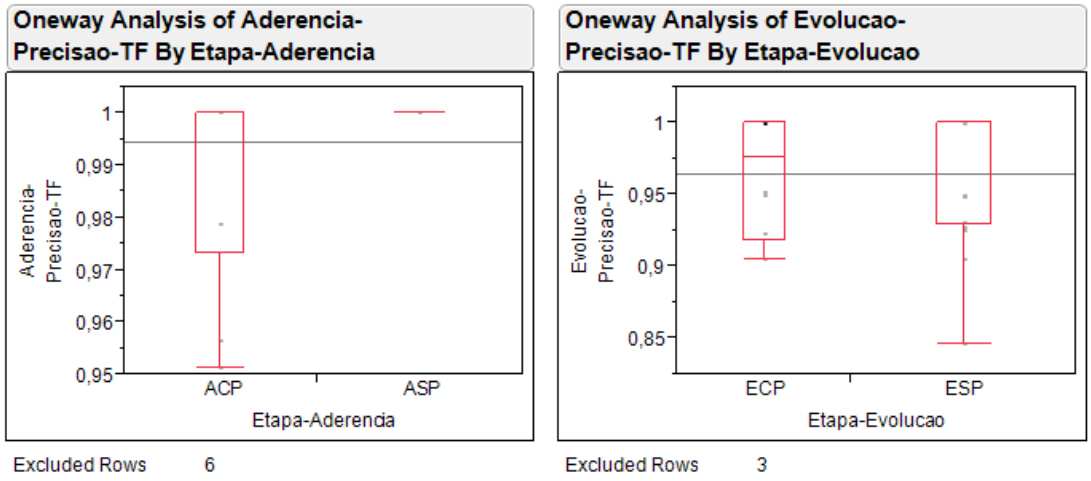


Figura 5.5 – *Box plot* da variável precisão para as tarefas de filtragem, focos de aderência e evolução (após a eliminação dos *outliers*)

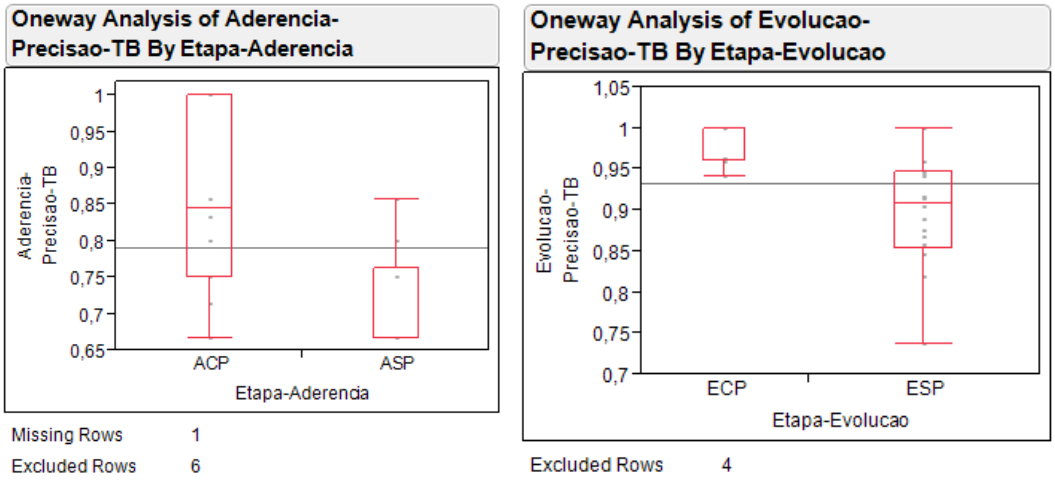


Figura 5.6 – *Box plot* da variável precisão para as tarefas básicas, focos de aderência e evolução (após a eliminação dos *outliers*)

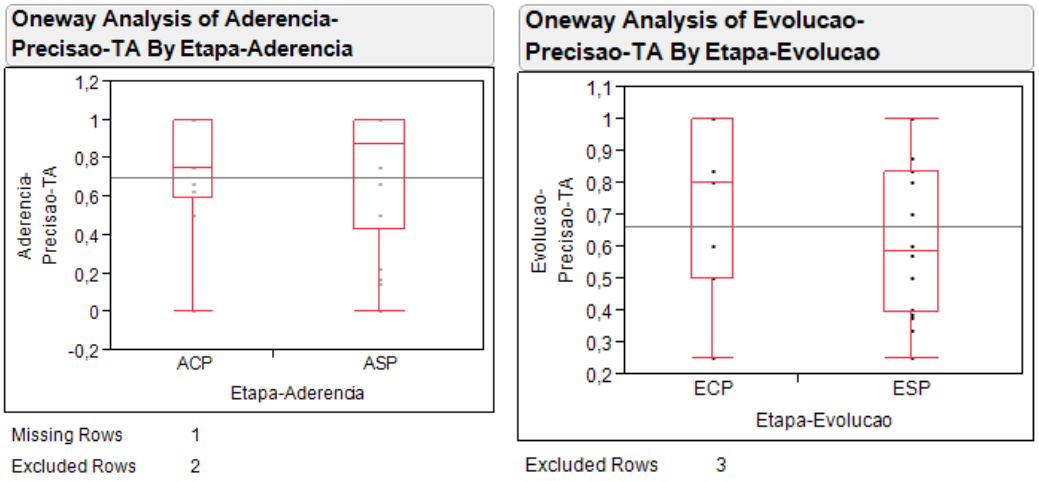


Figura 5.7 – *Box plot* da variável precisão para as tarefas de assimilação, focos de aderência e evolução (após a eliminação dos *outliers*)

As médias (Tabela 5.6 e Figura 5.8) também foram analisadas separadamente.

Tabela 5.6 – Média e desvio padrão da variável precisão (por grupo de tarefas)

Foco	Alternativa	Tarefas de Filtragem		Tarefas Básicas		Tarefas de Assimilação	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
Aderência	Sem PREViA	100,00%	0,00%	71,72%	6,62%	71,94%	35,69%
	Com PREViA	98,88%	1,93%	86,58%	12,90%	67,56%	32,69%
Evolução	Sem PREViA	96,57%	4,54%	89,54%	7,82%	60,87%	5,56%
	Com PREViA	96,35%	4,10%	98,61%	2,22%	74,17%	24,90%

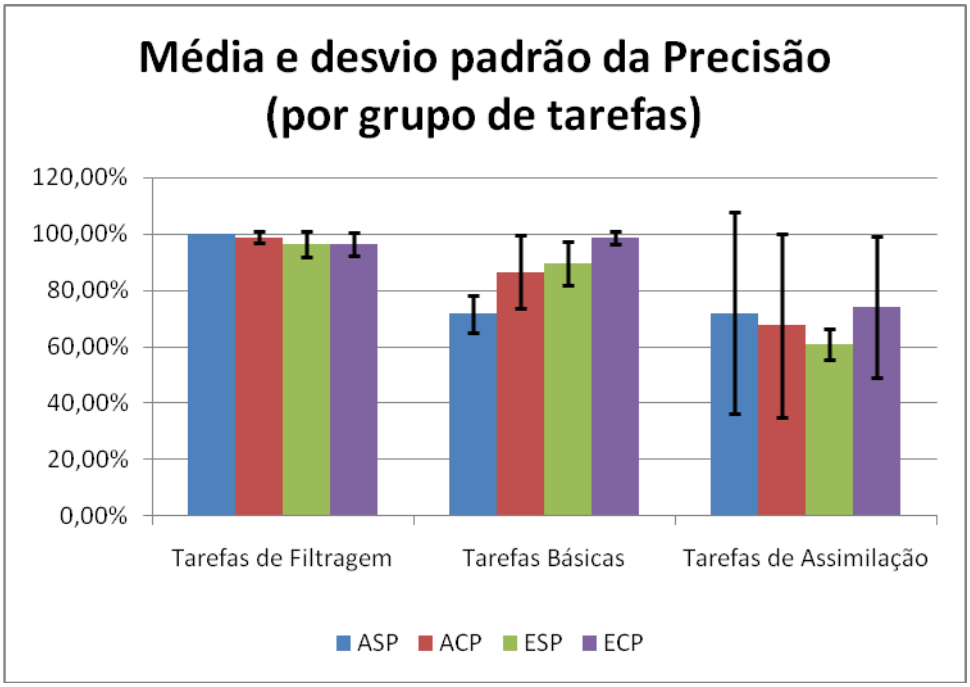


Figura 5.8 – Média e desvio padrão da variável precisão (por grupo de tarefas)

Por meio dos gráficos e dos dados da tabela, é possível notar que o uso da PREViA resultou em um valor de precisão inferior para as tarefas de filtragem (embora a média não desvie muito da obtida sem a PREViA no foco de aderência). Tal situação pode ser explicada pelo caráter das questões de filtragem: a abordagem PREViA exibe uma combinação de duas versões em um mesmo diagrama, o que requer compreensão e atenção para a identificação correta dos elementos de contexto (isto é, nos quais não ocorreu diferença). Outro ponto de possível dificuldade é a assimilação de que, quando um elemento não está destacado, isto indica que ele está presente nas duas versões sendo comparadas.

A análise de variância (Figura D.22 do Apêndice D) indica que existe diferença estatística entre o uso e o não uso da abordagem no foco de aderência (*p-value* = 0,0317), mas não no foco de evolução (*p-value* = 0,8892).

Já no que diz respeito às tarefas básicas, observa-se que o uso da PREViA obteve vantagem nos resultados obtidos em termos de precisão, em ambos os focos de visualização. Como as tarefas básicas dizem respeito ao cenário para o qual a PREViA foi desenvolvida, estes resultados são indícios positivos de que a PREViA provê maior precisão na execução de tarefas que envolvem a percepção.

Adicionalmente, a análise de variância (Figura D.23 do Apêndice D) indica que existe diferença estatística significativa entre o uso e o não uso da abordagem, tanto no foco de aderência quanto no de evolução (*p-value* = 0,0023 e 0,0003, respectivamente).

Por fim, é possível observar que, para as tarefas de assimilação, o uso da PREViA permitiu melhor precisão (em média) no foco de evolução, mas não no foco de aderência. As tarefas de assimilação pressupõem que os participantes tenham noções básicas do domínio do estudo e um conhecimento intermediário da linguagem Java (vide Seção B.3 do Apêndice B).

Ao analisar as médias dos dados de caracterização dos participantes, nota-se que, na etapa de aderência, os participantes que utilizaram a PREViA possuem menor experiência em Java e no domínio do estudo (2,118 e 0,118, respectivamente) do que a dos participantes que não utilizaram a PREViA (2,333 e 0,167, respectivamente), sendo esta uma possível causa para tais resultados.

Adicionalmente, a análise de variância (Figura D.24 do Apêndice D) indica que a diferença estatística entre o uso e o não uso da abordagem não é significativa em nenhum dos dois focos (*p-value* = 0,7238 para o foco de aderência e *p-value* = 0,1328 para o foco de evolução).

Cobertura

Ao efetuar a análise da variável de cobertura, observou-se que três participantes são considerados *outliers* na avaliação da aderência (conforme mostra a Figura 5.9). Seus resultados foram removidos da análise deste foco para a variável de cobertura. Na avaliação da evolução, nenhum participante é considerado *outlier*. A Figura 5.10 mostra os gráficos *box plot* dos focos de aderência e de evolução.

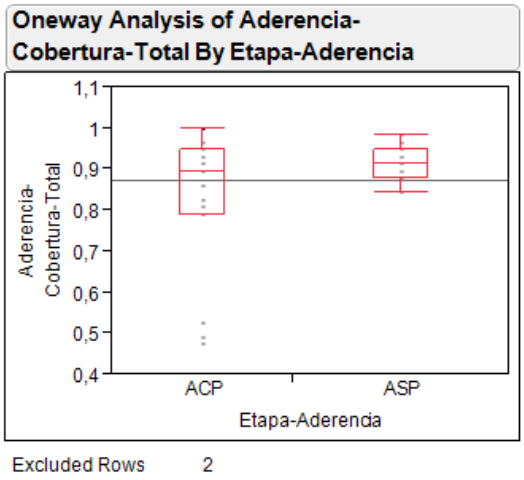


Figura 5.9 – *Box plot* da variável cobertura, foco de aderência (antes da eliminação dos *outliers*)

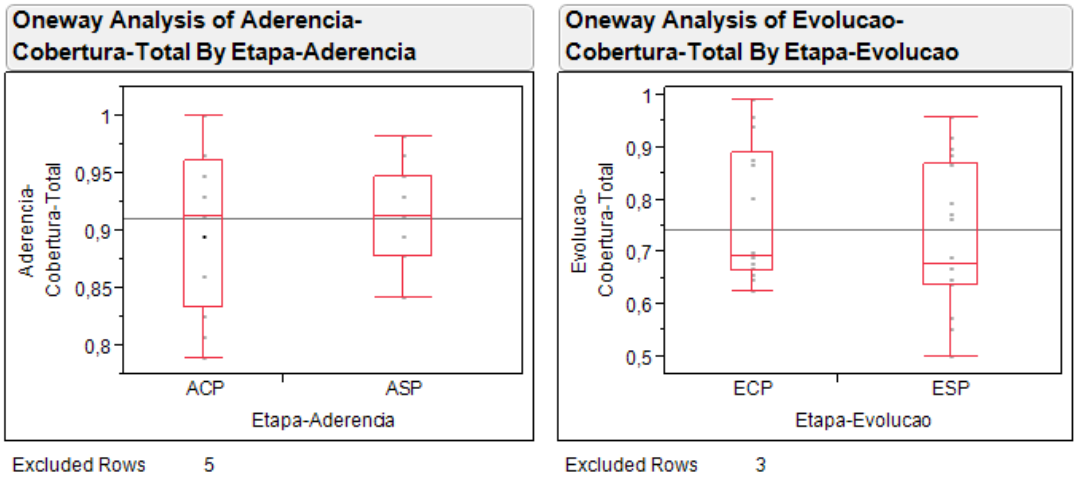


Figura 5.10 – *Box plot* da variável cobertura, focos de aderência (após a eliminação dos *outliers*) e evolução

A Tabela 5.7 e a Figura 5.11 exibem os valores da média e do desvio padrão para cada foco de visualização e alternativa.

Tabela 5.7 – Média e desvio padrão da variável cobertura (geral)

Foco	Alternativa	Média	Desvio Padrão
Aderência	Sem PREViA	91,52%	3,82%
	Com PREViA	90,35%	7,12%
Evolução	Sem PREViA	72,63%	13,52%
	Com PREViA	76,86%	13,07%

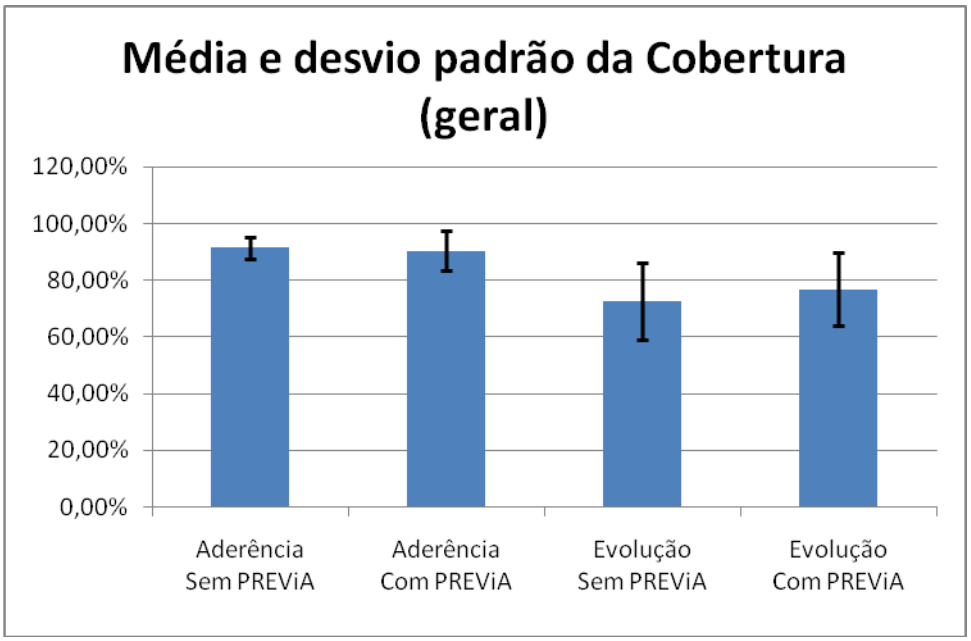


Figura 5.11 – Média e desvio padrão da variável cobertura (geral)

No foco de aderência, é possível observar que, embora os participantes sem PREViA tenham obtido melhor cobertura, os valores das médias são bem próximos, sendo que o desvio padrão aumenta com o uso da PREViA. Já no foco de evolução, a distância entre as médias aumenta, favorecendo o uso da PREViA.

Visando compreender melhor este cenário, no contexto da variável de cobertura, é efetuada a análise dos dados por grupos de tarefas (de filtragem, básicas e de assimilação). Na avaliação da aderência, 7 participantes são considerados *outliers*, todos nas tarefas de filtragem, enquanto na avaliação da evolução, 3 participantes são considerados *outliers*, todos nas tarefas básicas. Os resultados dos *outliers* foram removidos da análise dos respectivos focos e grupos de tarefas para a variável de cobertura. Os gráficos utilizados na exclusão dos *outliers* são apresentados no Apêndice D (Figura D.4 e Figura D.5).

As figuras seguintes (Figura 5.12, Figura 5.13 e Figura 5.14) exibem os gráficos finais após a eliminação dos *outliers*.

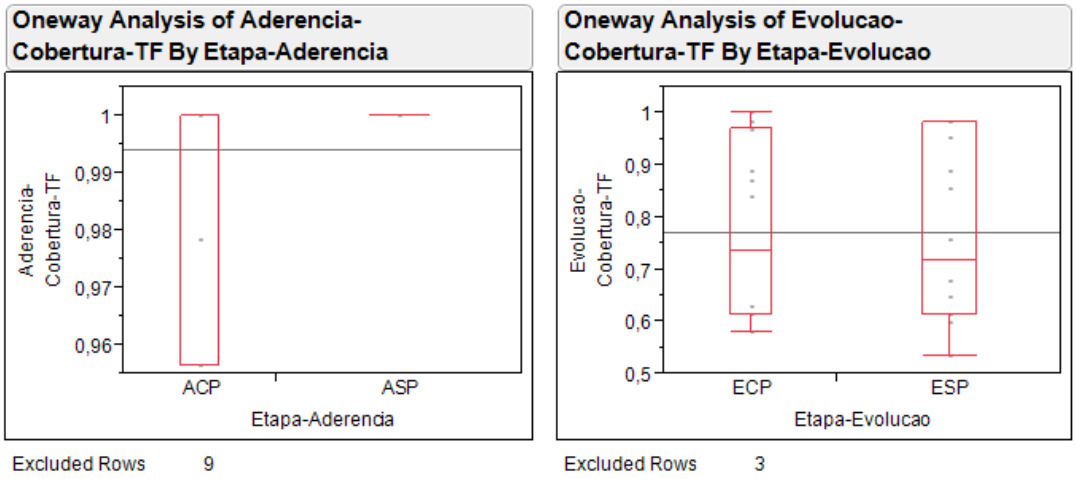


Figura 5.12 – *Box plot* da variável cobertura para as tarefas de filtragem, focos de aderência e evolução (após a eliminação dos *outliers*)

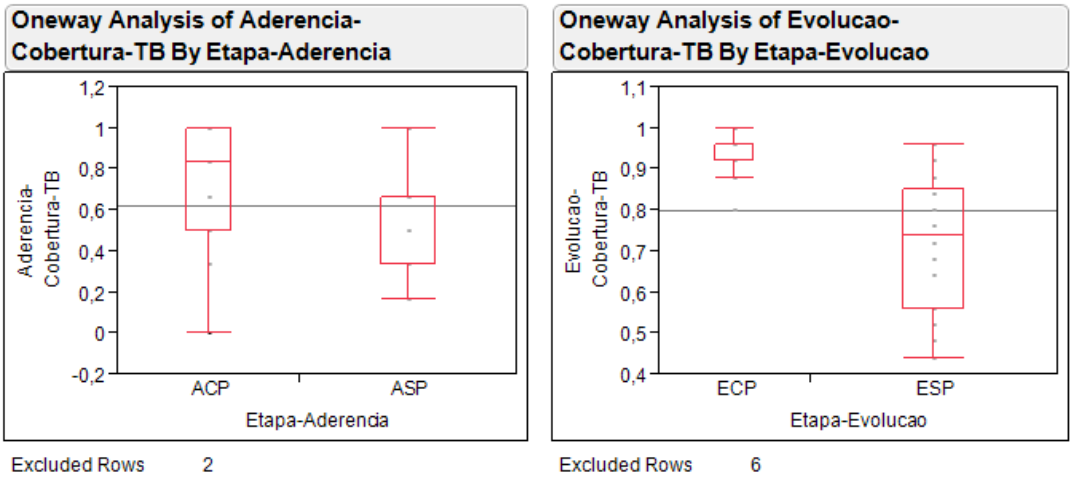


Figura 5.13 – *Box plot* da variável cobertura para as tarefas básicas, focos de aderência e evolução (após a eliminação dos *outliers*)

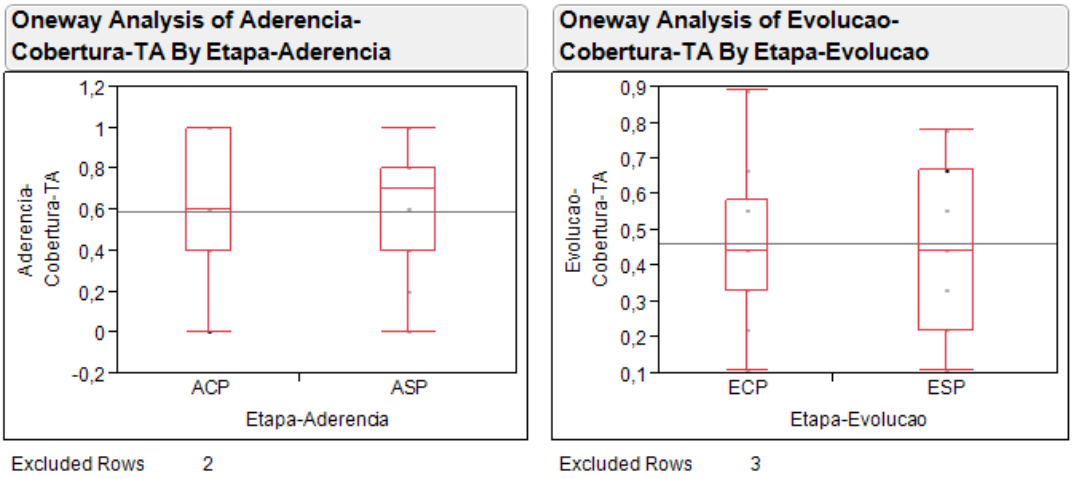


Figura 5.14 – *Box plot* da variável cobertura para as tarefas de assimilação, focos de aderência e evolução (após a eliminação dos *outliers*)

As médias (Tabela 5.8 e Figura 5.15) também foram analisadas separadamente.

Tabela 5.8 – Média e desvio padrão da variável cobertura (por grupo de tarefas)

Foco	Alternativa	Tarefas de Filtragem		Tarefas Básicas		Tarefas de Assimilação	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
Aderência	Sem PREViA	100,00%	0,00%	55,56%	19,80%	63,33%	26,79%
	Com PREViA	98,48%	2,06%	70,00%	30,34%	54,67%	35,83%
Evolução	Sem PREViA	77,06%	17,53%	71,56%	16,80%	45,06%	22,70%
	Com PREViA	77,30%	17,50%	93,45%	5,73%	48,41%	22,48%

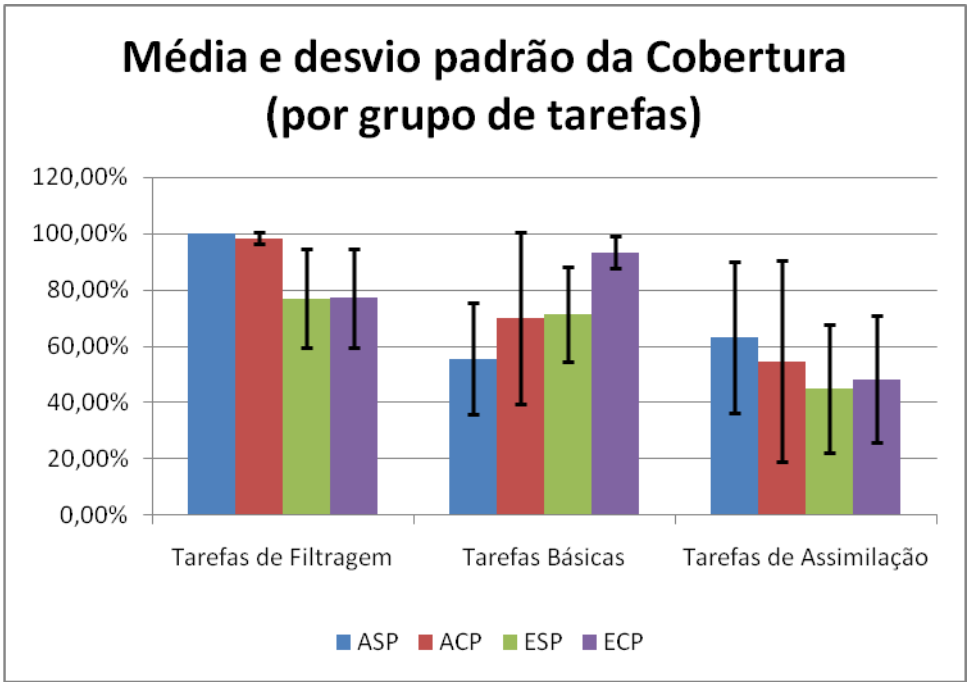


Figura 5.15 – Média e desvio padrão da variável cobertura (por grupo de tarefas)

Por meio dos gráficos e dos dados da tabela, é possível notar que, para as tarefas de filtragem, o uso da PREViA resultou em um valor de cobertura inferior no foco de aderência e superior no foco de evolução (em ambos os casos, a média não desvia muito da obtida sem a PREViA). Conforme explicado para a variável de precisão, não se trata de um resultado imprevisível. Além disso, o fato de os participantes que utilizaram a PREViA terem obtido resultados próximos aos dos que não a utilizaram pode indicar que houve um nível de compreensão razoável das informações do diagrama gerado pela abordagem.

A análise de variância (Figura D.26 do Apêndice D) indica que existe diferença estatística entre o uso e o não uso da abordagem no foco de aderência (*p-value* = 0,0081), mas não no foco de evolução (*p-value* = 0,9692).

Já no que diz respeito às tarefas básicas, há uma diferença considerável entre as médias dos resultados com e sem o uso da PREViA, tanto no foco de aderência quanto no de evolução, o que novamente traz indícios positivos, indicando que a PREViA provê maior cobertura na execução destas tarefas.

A análise de variância (Figura D.27 do Apêndice D) indica que não há diferença estatística no foco de aderência (*p-value* = 0,1101), mas há diferença significativa no foco de evolução (*p-value* = 0,0005).

Por fim, para as tarefas de assimilação, a média de cobertura obtida com o uso da PREViA foi mais baixa no foco de aderência e mais alta no foco de evolução. Cabe ressaltar que, em ambos os casos, os valores de desvio padrão são altos. Novamente, a possível causa para este resultado (apresentada para a variável de precisão) é o conhecimento envolvido para a resolução das tarefas e os dados de caracterização dos participantes.

Adicionalmente, a análise de variância (Figura D.28 do Apêndice D) indica que a diferença estatística entre o uso e o não uso da abordagem não é significativa em nenhum dos dois focos (*p-value* = 0,4329 para o foco de aderência e *p-value* = 0,6804 para o foco de evolução).

Eficiência

Ao efetuar a análise da variável de eficiência, observou-se que um participante é considerado *outlier* na avaliação da evolução (conforme mostra a Figura 5.16). Seus resultados foram removidos da análise deste foco para a variável de eficiência. Na

avaliação da aderência, nenhum participante é considerado *outlier*. A Figura 5.17 mostra os gráficos *box plot* dos focos de aderência e de evolução.

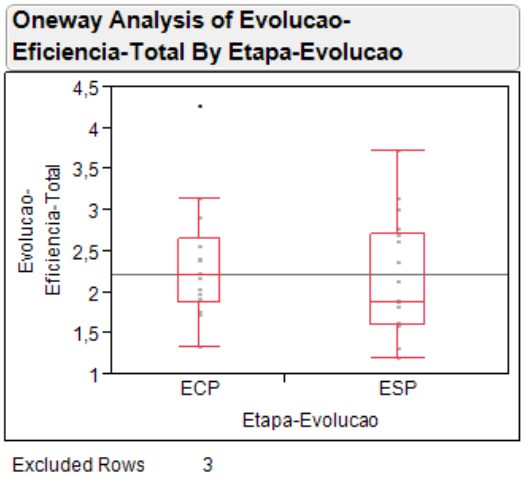


Figura 5.16 – *Box plot* da variável eficiência, foco de evolução (antes da eliminação dos *outliers*)

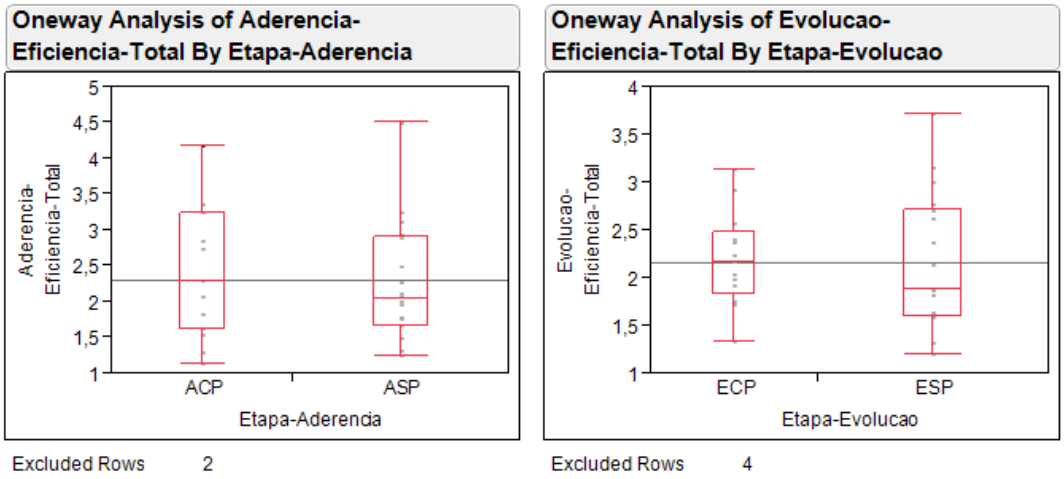


Figura 5.17 – *Box plot* da variável eficiência, focos de aderência e evolução (após a eliminação dos *outliers*)

A Tabela 5.9 e a Figura 5.18 exibem os valores da média e do desvio padrão para cada foco de visualização e alternativa.

Tabela 5.9 – Média e desvio padrão da variável eficiência (geral)

Foco	Alternativa	Média	Desvio Padrão
Aderência	Sem PREViA	2,25	0,82
	Com PREViA	2,41	0,89
Evolução	Sem PREViA	2,14	0,71
	Com PREViA	2,19	0,49

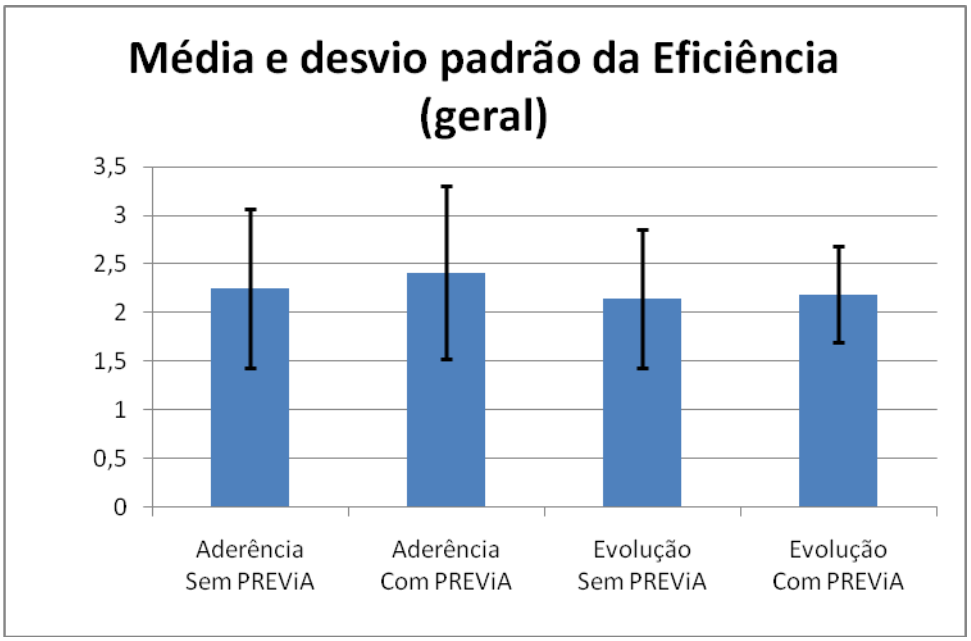


Figura 5.18 – Média e desvio padrão da variável eficiência (geral)

Tanto no foco de aderência quanto no de evolução, é possível observar que as médias de eficiência obtidas com o uso da PREViA foram superiores, embora os valores sejam muito próximos e possuam um desvio padrão consideravelmente alto.

Visando compreender melhor este cenário, no contexto da variável de eficiência, é efetuada a análise dos dados por grupos de tarefas (de filtragem, básicas e de assimilação).

Na avaliação da aderência, 2 participantes são considerados *outliers*, sendo 1 nas tarefas de filtragem e 1 nas tarefas básicas, enquanto na avaliação da evolução, 2 participantes são considerados *outliers*, todos nas tarefas de assimilação. Os resultados dos *outliers* foram removidos da análise dos respectivos focos e grupos de tarefas para a variável de eficiência. Os gráficos utilizados na exclusão dos *outliers* são apresentados no Apêndice D (Figura D.6, Figura D.7 e Figura D.8).

As figuras seguintes (Figura 5.19, Figura 5.20 e Figura 5.21) exibem os gráficos finais após a eliminação dos *outliers*.

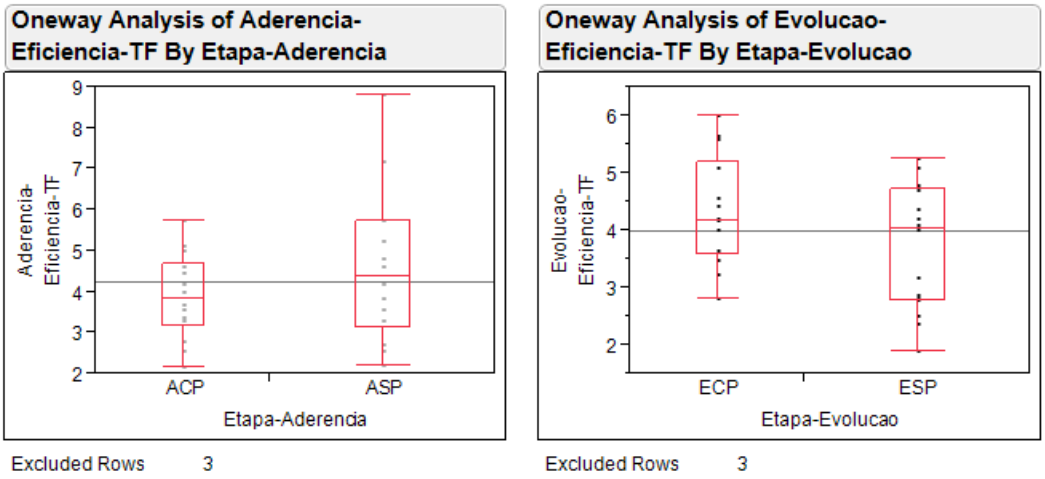


Figura 5.19 – *Box plot* da variável eficiência para as tarefas de filtragem, focos de aderência e evolução (após a eliminação dos *outliers*)

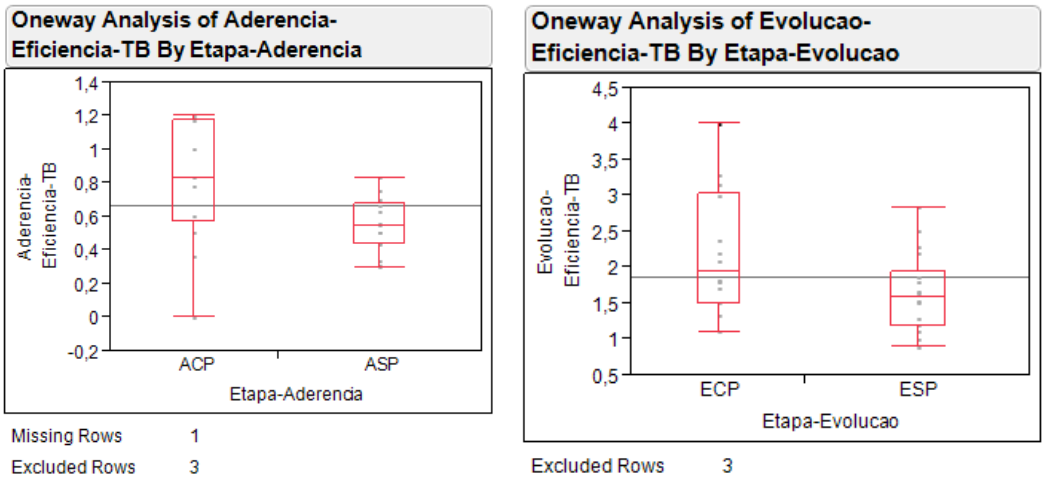


Figura 5.20 – *Box plot* da variável eficiência para as tarefas básicas, focos de aderência e evolução (após a eliminação dos *outliers*)

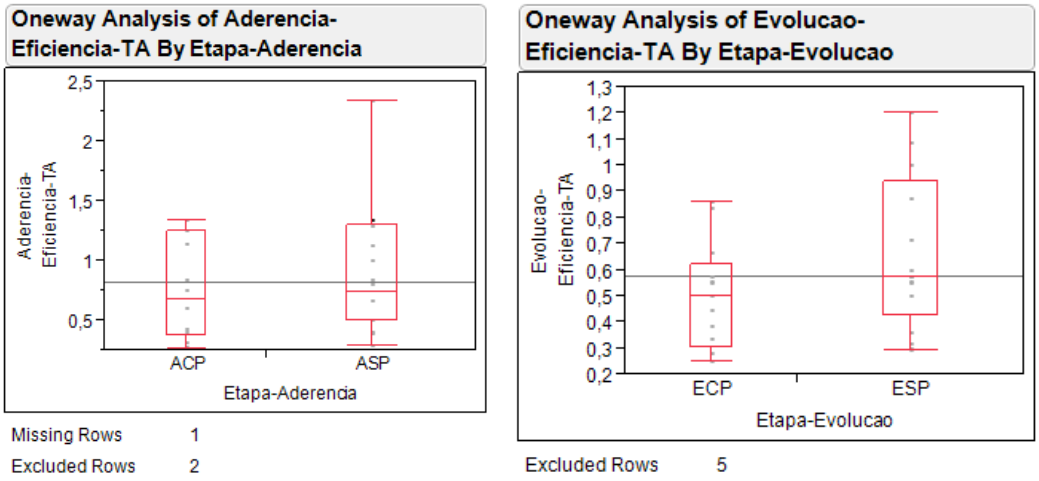


Figura 5.21 – *Box plot* da variável eficiência para as tarefas de assimilação, focos de aderência e evolução (após a eliminação dos *outliers*)

As médias (Tabela 5.10 e Figura 5.22) também foram analisadas separadamente.

Tabela 5.10 – Média e desvio padrão da variável eficiência (por grupo de tarefas)

Foco	Alternativa	Tarefas de Filtragem		Tarefas Básicas		Tarefas de Assimilação	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
Aderência	Sem PREViA	4,56	1,75	0,56	0,16	0,88	0,51
	Com PREViA	3,89	1,04	0,81	0,36	0,76	0,43
Evolução	Sem PREViA	3,74	1,09	1,64	0,54	0,65	0,29
	Com PREViA	4,34	0,95	2,20	0,86	0,50	0,20

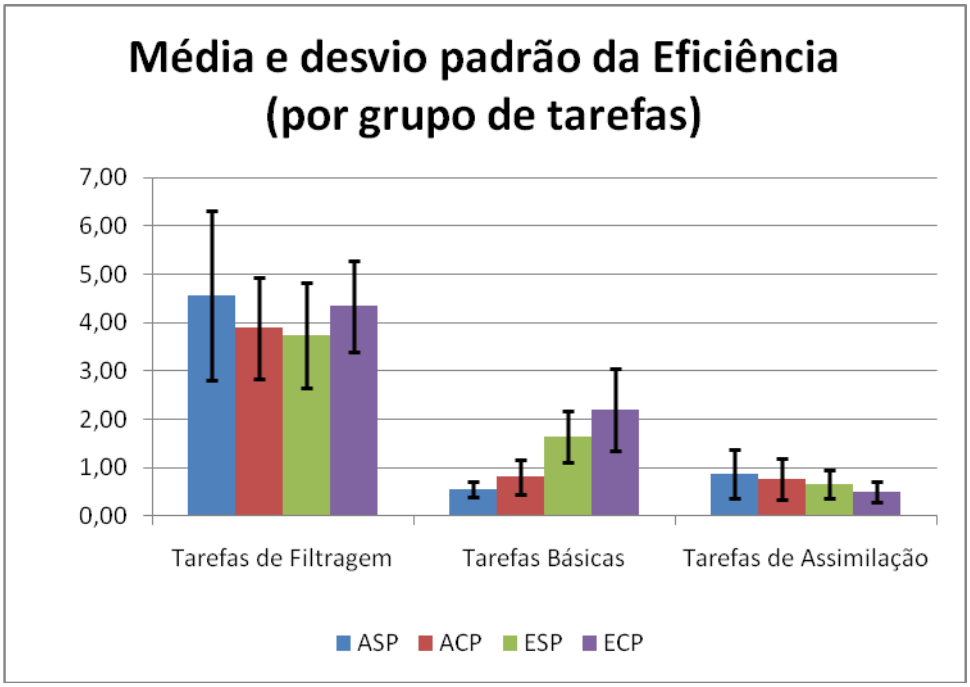


Figura 5.22 – Média e desvio padrão da variável eficiência (por grupo de tarefas)

Nas tarefas de filtragem, a eficiência com a abordagem PREViA foi inferior no foco de aderência e superior no foco de evolução, com desvio padrão razoável. Não foi possível identificar nenhum padrão de comportamento nos dados que justificasse a diferença entre estes resultados.

A análise de variância (Figura D.30 do Apêndice D) indica que não existe diferença estatística entre o uso e o não uso da abordagem, tanto no foco de aderência quanto no de evolução (*p-value* = 0,2100 e 0,1137, respectivamente).

Com relação às tarefas básicas, novamente a abordagem PREViA resultou em médias maiores para os focos de aderência e evolução, corroborando com o propósito para o qual a abordagem foi desenvolvida. Com isto, há indícios de que a abordagem PREViA fornece valores mais altos de precisão e cobertura para tarefas desta categoria, permitindo também maior eficiência na execução de tais tarefas (cabe lembrar que o poder de conclusão está limitado à significância estatística dos resultados).

A análise de variância (Figura D.31 do Apêndice D) indica que existe diferença estatística significativa entre o uso e o não uso da abordagem em ambos os focos de visualização (*p-value* = 0,0161 no foco de aderência e *p-value* = 0,0315 no foco de evolução).

Por fim, quanto às tarefas de assimilação, a eficiência no uso da abordagem PREViA é menor em ambos os focos de visualização. Algumas possíveis causas para isto podem ser: (i) a insegurança no uso da abordagem para as tarefas deste tipo, (ii) a dificuldade de interpretação das tarefas, e (iii) a dificuldade de interpretação dos elementos visuais como apoio para a execução das atividades. Apesar de os resultados com o uso da PREViA não terem sido melhores, o desvio padrão chega a ser superior à metade da média no foco de aderência, com e sem o uso da PREViA.

De forma complementar, a análise de variância (Figura D.32 do Apêndice D) indica que a diferença estatística entre o uso e o não uso da abordagem não é significativa em nenhum dos dois focos (*p-value* = 0,4863 para o foco de aderência e *p-value* = 0,1240 para o foco de evolução).

5.2.3.2 Análise Qualitativa

Com base nas respostas do questionário *follow-up* (apresentado na Seção B.5 do Apêndice B) e em algumas informações obtidas dos dados quantitativos, foi feita uma análise qualitativa sobre a utilização da abordagem PREViA. Foi solicitado, dentre

outros tópicos, que os participantes indicassem pontos positivos e negativos, bem como sugestões de melhoria para a abordagem.

A Figura 5.23 mostra as respostas dos participantes quanto à facilidade de execução das etapas do estudo.

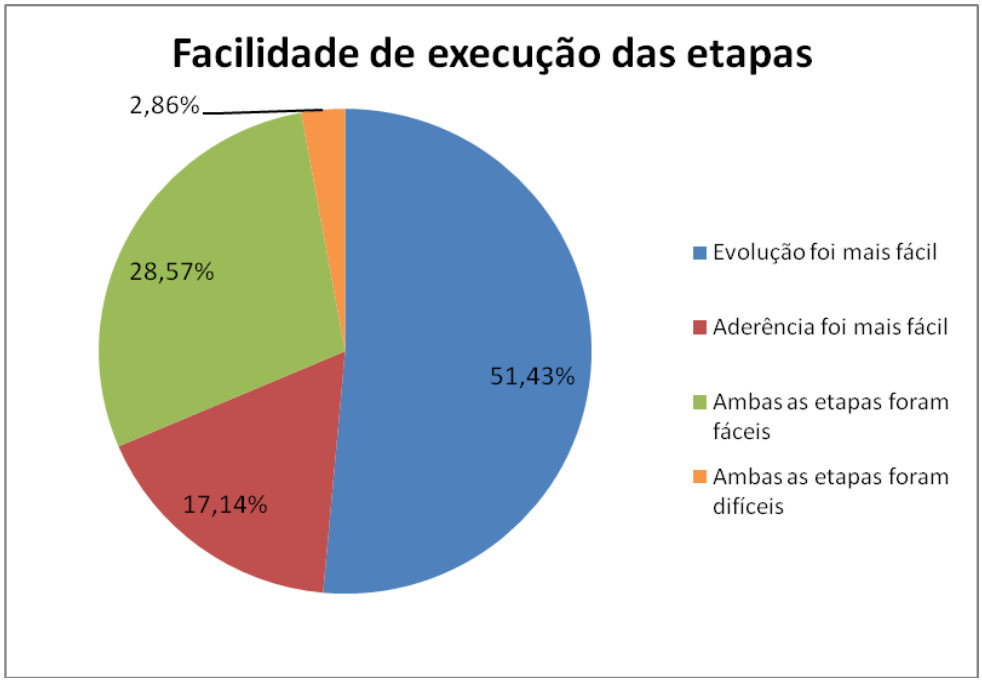


Figura 5.23 – Respostas dos participantes quanto à facilidade de execução das tarefas do estudo

Mais da metade dos participantes considerou a etapa de evolução mais fácil. Destes, 55,56% executaram esta etapa com a abordagem PREViA e os outros 44,44% sem a abordagem. Desta parcela (que considerou a etapa de evolução mais fácil e que não fez uso da PREViA nesta etapa), 22,22% utilizaram PREViA na etapa de aderência, o que significa que 22,22% dos que não consideraram a etapa de evolução mais fácil também não teve contato com a abordagem na etapa de aderência.

Analisando os participantes que consideraram a etapa de aderência mais fácil, 50% executaram tal etapa fazendo uso da abordagem PREViA, enquanto os outros 50% não a utilizaram. Desta parcela (que considerou a etapa de aderência mais fácil e que não fez uso da PREViA nesta etapa), apenas 16,67% utilizaram PREViA na etapa de evolução, o que significa que 33,33% dos que não consideraram a etapa de aderência mais fácil também não teve contato com a abordagem na etapa de evolução.

Quando questionados sobre as dificuldades sentidas na realização das tarefas, os participantes mencionaram basicamente os seguintes pontos: (i) as atividades foram cansativas; (ii) algumas questões soaram ambíguas; (iii) pouco conhecimento e

familiaridade com os modelos e conceitos do domínio tratado; e (iv) dificuldades no entendimento e realização das tarefas devido a uma prova ocorrida no intervalo de tempo anterior ao estudo.

Foi solicitada aos participantes que tiveram contato com a abordagem PREViA em alguma etapa (ao todo, 25 participantes) sua opinião quanto ao impacto das informações visuais na realização das tarefas. A Figura 5.24 mostra os resultados obtidos.

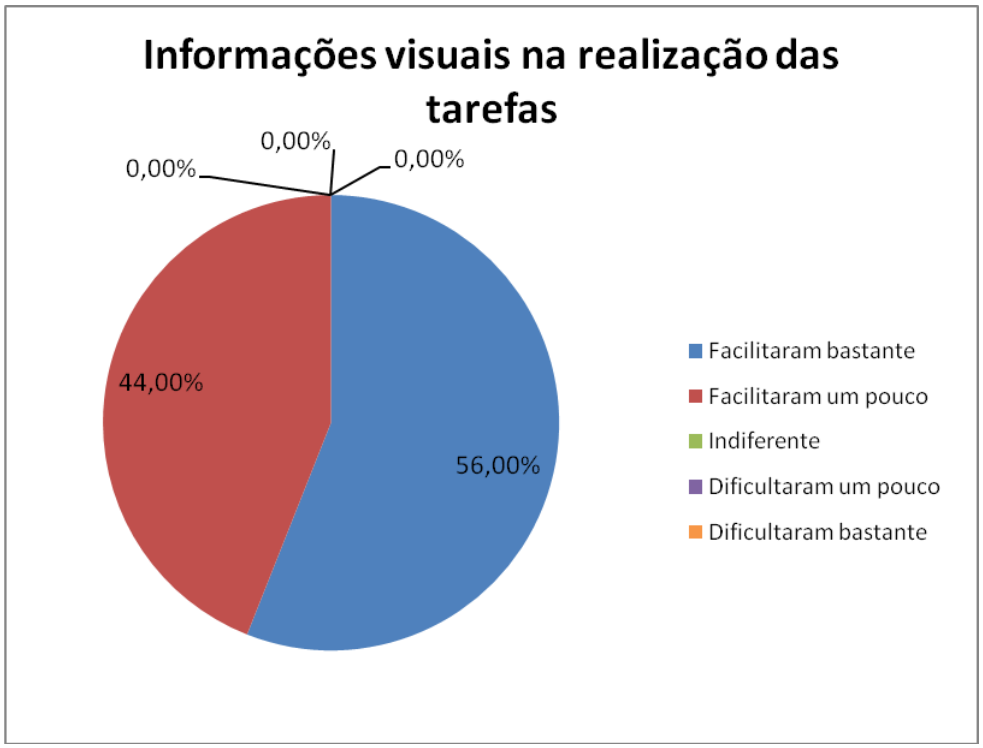


Figura 5.24 – Respostas dos participantes quanto à facilidade de execução das tarefas do estudo

Pelo gráfico, é possível notar que todos os participantes do estudo consideraram que as informações visuais da abordagem PREViA facilitaram na realização das tarefas. Nenhum participante considerou que as informações dificultaram ou que são indiferentes.

Analisando o efeito de aprendizagem durante a execução deste estudo, dentre os 34 participantes que executaram o estudo por completo (etapas de aderência e evolução), 23 obtiveram tempo inferior de execução na segunda etapa com relação à primeira. A diferença máxima de tempo chega a 58 minutos, enquanto a diferença mínima é de 0 minutos (i.e., não houve diferença de tempo neste caso). Dentre os 11 participantes que demoraram mais na segunda seleção, os valores em cada seleção

variaram bastante: a diferença máxima foi de 34 minutos (o mesmo valor da mediana), sendo a média das diferenças destes participantes igual a 34,73 com desvio padrão igual a 11,12.

É interessante notar que 10 destes 11 participantes iniciaram o estudo com a etapa de verificação de aderência (igualmente distribuídos com relação ao uso e não uso da abordagem PREViA), o que indica que pode existir um comportamento padrão para um melhor tempo na primeira seleção com relação ao tipo de atividade executado.

Alguns dos pontos positivos mencionados pelos participantes são:

- "Facilidade e rapidez de perceber a alteração" (7 participantes);
- "O aspecto visual provê melhor percepção das mudanças e diferenças" (5 participantes);
- "Clareza e entendimento das informações" (2 participantes);
- "Não há repetição de informações visuais para elementos que não foram alterados entre as revisões" (2 participantes);
- "Não necessita fazer múltiplas leituras de diferentes diagramas para encontrar diferenças, pois elas já são explicitadas";
- "É possível verificar de forma clara a evolução de um projeto através do seu diagrama e confrontar possíveis alterações no que foi projetado para o que foi implementado de fato, facilitando a verificação e questionamento das divergências"; e
- "Permite verificar, no mesmo diagrama, as mudanças de uma revisão para outra".

Como se pode observar, alguns benefícios esperados com a abordagem foram confirmados pelos participantes, tais como a facilidade de identificação das divergências/diferenças e a clareza das informações.

Os participantes também apontaram os seguintes pontos negativos:

- "Comparação de apenas dois modelos por vez" / "Só dá pra comparar duas versões" (2 participantes);
- "Dificuldade inicial de entendimento da abordagem PREViA" (2 participantes);
- "Muita informação" / "Informações ficam confusas com os modelos na mesma folha" (2 participantes);
- "Ocasionalmente, pode-se confundir os modelos" / "Requer atenção" (2 participantes);

- “Pode acabar tornando o diagrama muito complexo” (2 participantes); e
- “Quando houver muitas alterações entre versões ou diferenças entre modelos, o diagrama deve ficar com muita informação, podendo ficar confuso”;
- “Dificuldade de notar a mudança de um relacionamento de uma classe para outra”;
- “A abordagem apenas identifica as divergências, não oferecendo qualquer informação sobre por que ocorreram”;
- “Uma quantidade grande de diagramas precisa ser gerada num ciclo comum de desenvolvimento de software”.

No que diz respeito à quantidade de modelos comparada, pode haver sobrecarga cognitiva ao comparar mais de dois modelos, devido à necessidade de novas representações visuais para informar as diferenças/divergências existentes. Quanto à quantidade de diagramas a ser gerada, isto é automatizado pelo apoio ferramental da abordagem PREViA.

Um ponto interessante (que foi mencionado por dois participantes) é a dificuldade inicial de entendimento da abordagem PREViA, o que pode, de fato, causar confusão e dificultar a execução das atividades num primeiro momento. No entanto, acredita-se que, assim como ocorre com a maioria das ferramentas utilizadas no dia-a-dia, a frequência de uso pode trazer mais clareza nas informações contidas nas representações visuais.

Com relação ao comentário sobre haver muitas alterações entre versões (ou diferenças entre modelos), os recursos de filtragem do apoio ferramental podem, até certo ponto, contornar tal deficiência.

Por fim, as seguintes sugestões de melhoria foram mencionadas:

- “Quando gerar muita informação, possibilitar ao usuário escolher as classes das quais ele quer ver as informações”;
- [Permitir a] “comparação de mais de duas versões por diagrama, para comparar versões mais distantes”; e
- “Precisa-se verificar outra forma de rastrear a mudança de um relacionamento entre as classes”.

Quanto à primeira sugestão, o recurso de filtragem presente na ferramenta pode auxiliar; quanto à segunda, com a ferramenta é possível selecionar quais versões devem

ser comparadas, e em que ordem; para que a terceira sugestão seja atendida, é necessário verificar uma forma de representar movimentações sem aumentar a carga de informações visuais da abordagem.

5.2.4 Ameaças à Validade

É comum que haja questões que possam impactar ou limitar a validade dos resultados dos estudos. Estas questões são denominadas ameaças à validade (WOHLIN *et al.*, 2000). Durante o planejamento deste estudo, buscou-se evitar tais ameaças; no entanto, não é possível garantir que tais ameaças não tenham afetado de alguma forma os resultados. Desta forma, esta subseção apresenta as ameaças identificadas, categorizando-as em validade interna, validade externa, validade de conclusão e validade de construção, conforme proposto por WOHLIN *et al.* (2000).

As ameaças à validade interna são influências que podem prejudicar o conhecimento sobre o relacionamento de causa e efeito entre o tratamento e o resultado. Por se tratarem de eventos não controlados pelo pesquisador, podem produzir distorções no resultado esperado. Neste estudo, foram identificadas as seguintes ameaças nesta categoria:

- O estudo foi executado após uma prova de outra disciplina (evento externo), o que pode ter prejudicado os participantes no preenchimento dos formulários do estudo, conforme apontado por alguns alunos no questionário *follow-up*. Além disso, o estudo não foi executado em um único dia por todos os participantes; isto também pode ter influenciado os resultados, já que não é possível confirmar que as circunstâncias eram as mesmas nas ocasiões que cada participante participou do estudo;
- Conforme descrito na seção de planejamento, o estudo consistiu de duas etapas (aderência e evolução, não necessariamente nesta ordem). Embora o estudo tenha sido projetado para evitar o efeito de aprendizado dos participantes (fornecendo modelos diferentes a cada fase), não é possível confirmar que este efeito tenha sido totalmente eliminado;
- Para realizar algumas justificativas na análise dos dados, as informações de caracterização que cada participante forneceu de si mesmo foram utilizadas. Não é possível confirmar que tais informações fornecidas estejam corretas;
- A disposição dos elementos (layout) dos diagramas pode exercer influência sobre os resultados (SUN & WONG, 2005; KNODEL *et al.*, 2006b; SHARIF &

MALETIC, 2009). No contexto deste estudo, os diagramas foram reorganizados de forma a não ultrapassarem as margens de impressão e não haver sobreposição de elementos;

- O entendimento dos participantes sobre as questões dos formulários é diretamente influenciado pela forma como as questões foram elaboradas; se a questão tiver sido mal formulada, o estudo pode ser afetado negativamente (WOHLIN *et al.*, 2000). A análise dos instrumentos utilizados no estudo (inclusive os formulários) por especialistas visou reduzir esta interferência. No entanto, a partir das análises quantitativa e qualitativa, pode-se observar que algumas perguntas foram interpretadas de maneiras diferentes pelos alunos, enquanto outras foram consideradas ambíguas;
- Embora os participantes tenham sido monitorados durante o estudo, não é possível garantir que os mesmos não tenham copiado as respostas de outros. De forma a minimizar este problema, foi explicado textual (vide Seção B.1 do Apêndice B) e oralmente durante a execução do estudo que o foco da avaliação era a abordagem, e não os alunos de maneira individual;
- Como não se trata de um estudo de observação (inviável pela quantidade de participantes), assumiu-se que os alunos seguiram as instruções e a ordem das atividades; adicionalmente, não é possível confirmar se o tempo informado pelos participantes durante as tarefas de execução do estudo está correto; além disso, a unidade de tempo utilizada na análise (minutos) possui uma margem de erro inerente;
- Assim como no estudo descrito em (DIAS NETO, 2009), sabe-se que os participantes não possuem a mesma habilidade para resolução de problemas. Para tentar minimizar este efeito, os participantes foram alocados aos grupos de forma aleatória.

As ameaças à validade externa limitam a generalização dos resultados do estudo para outros contextos fora do ambiente avaliado. Neste estudo, existem as seguintes ameaças à validade externa:

- Os participantes deste estudo são alunos de graduação, o que limita a representatividade das pessoas que poderiam se beneficiar da abordagem; no entanto, parte dos alunos possui experiência na indústria (por exemplo, 40% do total já trabalharam com OO na indústria), então isto diminui tal ameaça;

- Normalmente, ambientes acadêmicos não simulam totalmente as condições existentes em um ambiente de desenvolvimento de software (GOMES *et al.*, 2009). Em contrapartida, CARVER *et al.* (2003) apontam que a realização de estudos experimentais tendo alunos como participantes permite obter uma série de benefícios para os pesquisadores (e.g., obtenção de evidências preliminares para confirmar ou rejeitar hipóteses e treinamento de jovens investigadores no campo da pesquisa empírica), os estudantes (e.g., atividades práticas e a educação em tópicos que são alvo de pesquisas recentes) e os professores (e.g., a utilização de formas menos convencionais de ensino e a introdução de engenharia de software empírica como parte do ensino de engenharia de software);
- Foi utilizado um único projeto (Prontuário Eletrônico) durante o estudo, o que limita consideravelmente a generalização dos resultados. Experimentos com outros tipos de projetos devem ser executados;
- Não é possível representar todas as situações possíveis de um projeto de software. As tarefas/questões do estudo foram elaboradas com base em situações comuns no contexto em que estão inseridas; no entanto, é necessário verificar se os objetivos da abordagem PREViA são atingidos em outras circunstâncias.

As ameaças à validade de conclusão referem-se a fatos que prejudicam o estabelecimento de relacionamentos estatísticos entre o tratamento e o resultado. Nesta categoria, as seguintes ameaças foram identificadas:

- Na configuração do estudo, os participantes foram distribuídos nos grupos de forma aleatória. Os grupos resultantes apresentaram certo grau de heterogeneidade quanto ao grau de experiência (embora em áreas distintas), o que pode contribuir para que a variação devido às diferenças individuais dos participantes seja maior que a variação produzida pela aplicação da abordagem PREViA;
- O tamanho da amostra é limitado, o que não é ideal do ponto de vista estatístico. Desta forma, os resultados do estudo não são conclusivos: somente fornecem indícios. No entanto, considerando experimentos em engenharia de software, o número de observações é considerado alto, o que aumenta o poder estatístico das conclusões obtidas;

- Na análise dos dados, o critério de remoção de *outliers* por meio das tarefas de filtragem é subjetivo. No entanto, ao analisar os dados destes participantes, os resultados também foram ruins para as tarefas básicas e de assimilação, o que, de certa forma, corrobora com o objetivo da filtragem. De acordo com os dados do questionário de caracterização, observou-se que (i) três destes *outliers* só possuem experiência teórica em orientação a objetos, (ii) a experiência que todos estes *outliers* possuem em modelagem, UML e inspeção é, no máximo, resultado de práticas em sala de aula.

Por fim, as ameaças relacionadas à validade de construção dizem respeito a eventos que podem impedir que a configuração do experimento reflita adequadamente a construção do relacionamento de causa e efeito (entre o tratamento e o resultado) em estudo. Em outras palavras, são eventos que podem prejudicar a medição correta no estudo. Foram identificadas as seguintes ameaças à validade de construção:

- Há uma hierarquia entre os participantes do estudo (alunos) e o responsável pela execução do estudo (professor). A fim de minimizar os efeitos desta ameaça, tanto nos instrumentos utilizados no estudo como no treinamento inicial no início do estudo, buscou-se deixar claro que os resultados do estudo não iriam influenciar o resultado do desempenho dos participantes na disciplina em questão. No entanto, não foi possível confirmar se os participantes executaram o estudo da mesma forma como o teriam executado em outro cenário;
- O agrupamento das tarefas por tipo auxilia a análise dos dados. No entanto, embora algumas destas tarefas possam ter grau de dificuldade maior do que o de outras tarefas, o mesmo peso foi atribuído a todas as tarefas. Isto pode influenciar os resultados. Devido à subjetividade na avaliação do grau de dificuldade (o que introduziria viés na análise dos dados), optou-se por manter esta configuração.

5.3 Avaliação da Abordagem no Contexto de Educação em Engenharia de Software

De acordo com LETHBRIDGE *et al.* (2007), a educação dos princípios e práticas de engenharia de software tem recebido atenção especial da comunidade, de forma a aprimorá-la e identificar os problemas, soluções e desafios. Alguns desafios apontados por estes autores são: (i) tornar os programas atraentes para os alunos, (ii)

fazer com que o ensino de engenharia de software seja mais baseado em evidências, e (iii) aumentar o reconhecimento e a qualidade de pesquisas de engenharia de software voltadas para a educação.

Neste cenário, o estudo dos benefícios alcançados com a utilização de arquiteturas de software e o entendimento das consequências de sua aplicação fizeram com que o tópico se tornasse uma disciplina, tornando-se um dos principais tópicos de engenharia de software (GARLAN & SHAW, 1994; LAGO & VLIET, 2005). Uma revisão sistemática da literatura sobre as iniciativas realizadas no ensino de arquitetura de software (RODRIGUES & WERNER, 2009) reafirmou a necessidade de um programa de educação que permita uma experiência rica e prática e vise aprofundar a compreensão das habilidades necessárias à formação de um bom arquiteto de software.

Na dinâmica da condução de uma disciplina de arquitetura de software, a detecção de divergências entre modelos é uma atividade importante em termos de ensino e aprendizagem. Do ponto de vista dos alunos, a comparação entre o seu modelo e o modelo do professor permite a interpretação dos acertos e erros cometidos durante um exercício de modelagem (o conhecimento dos erros conduz à aprendizagem). No entanto, o entendimento das divergências entre modelos não é uma atividade trivial. Da mesma forma, o professor pode ter dificuldades em corrigir exercícios de modelagem, especialmente considerando que elementos que diferem do gabarito podem significar diferentes alternativas de projeto. Estes exercícios podem conter vários elementos, e a comparação é geralmente realizada utilizando modelos feitos à mão.

A comunicação visual é um fator chave no processo de ensino e aprendizagem de arquitetura de software e modelagem de sistemas (PERRY & WOLF, 1992). Neste sentido, as técnicas de visualização de software podem ser utilizadas pelos alunos para facilitar a compreensão e a aprendizagem de novos conceitos no desenvolvimento de software, além de apoiar professores nas tarefas de ensino.

Desta forma, acredita-se que a abordagem PREViA possa fornecer apoio a atividades de comparação e compreensão de modelos, especialmente quando tais atividades ocorrem repetidamente (o que pode fazer com que se tornem cansativas e ineficientes). O uso deste tipo de abordagem no cenário de ensino e aprendizagem decorre da importância de se discutir e investigar os aspectos da construção do conhecimento no ensino das disciplinas relacionadas.

Esta seção apresenta um estudo de viabilidade sobre o uso da abordagem PREViA para apoiar atividades relacionadas à educação em engenharia de software

(mais especificamente, na educação em modelagem de sistemas) (SCHOTS *et al.*, 2010a). A realização deste estudo pretende avaliar se a aplicação da abordagem no contexto de educação em engenharia de software é viável, no que diz respeito à facilidade de interpretar e compreender as divergências entre modelos.

Segundo SHULL *et al.* (2001), um estudo de viabilidade tem como objetivo verificar se uma tecnologia ou processo proposto atinge um determinado objetivo para o qual foi planejado, com um retorno de investimento razoável. Este tipo de estudo provê informações sobre a viabilidade de continuar com o desenvolvimento da tecnologia ou processo proposto. Considerando o contexto de educação em engenharia de software, pretende-se também explorar quais adaptações são necessárias a fim de adequar a abordagem a este cenário.

Neste estudo, a abordagem PREViA é utilizada na avaliação de modelos em alto nível de abstração criados por estudantes e corrigidos pelo professor. O principal objetivo deste estudo é caracterizar como professores avaliam as divergências entre modelos. Assim como o estudo apresentado na Seção 5.2, o planejamento do estudo (apresentado a seguir) foi revisado por dois pesquisadores que não possuem interesse direto nos resultados do estudo, visando minimizar a presença de viés.

5.3.1 Planejamento do Estudo

Conforme discutido no Capítulo 4, a abordagem PREViA possui originalmente dois focos de visualização: (i) a aderência da arquitetura emergente à arquitetura conceitual no decorrer do tempo, e (ii) a compreensão do processo de evolução por meio das diferenças entre as versões. No entanto, como o objetivo deste estudo não envolve a evolução, a dimensão tempo não foi levada em consideração, e somente o foco de aderência foi adotado, fazendo com que o gabarito (i.e., o modelo esperado como resultado do exercício) seja o modelo conceitual, e cada modelo desenvolvido pelos alunos seja considerado como um modelo emergente distinto (que não compõe um ciclo evolutivo).

De forma análoga ao que ocorre no foco de aderência original da abordagem PREViA, os modelos elaborados por cada aluno são sobrepostos ao gabarito, e o último é apresentado na forma de uma marca d'água, que é preenchida quando ocorre uma correspondência (*match*) entre elementos dos dois modelos. Assim, para este uso, os destaques feitos com a cor vermelha indicam elementos que foram inseridos na resposta, mas não existem no gabarito; a cor cinza indica elementos que aparecem no

modelo, mas não foram incluídos na resposta. Elementos que foram encontrados em ambos os modelos estão representados na cor preta.

A identificação dos objetivos deste estudo foi feita segundo a abordagem GQM (BASILI *et al.*, 1994), conforme pode ser visto na Tabela 5.11.

Tabela 5.11 – Identificação dos objetivos do estudo

Perguntas	Respostas
Objeto do estudo (o que será analisado?)	a abordagem PREViA
Propósito (por que / para quê o objeto será analisado?)	caracterizar
Foco de qualidade (que propriedades do objeto serão analisadas?)	facilidade de interpretação e compreensão das divergências entre modelos
Ponto de vista (quem utilizará os dados coletados?)	professores e instrutores
Ambiente (em que ambiente será realizada a análise?)	disciplinas de arquitetura de software e modelagem de sistemas

Assim, este estudo pode ser definido da seguinte forma:

Analisar	a abordagem PREViA
Com o propósito de	caracterizar
Com respeito a	facilidade de interpretação e compreensão das divergências entre modelos
Do ponto de vista de	professores e instrutores
No contexto de	disciplinas de arquitetura de software e modelagem de sistemas

Para auxiliar na verificação da facilidade de interpretação e compreensão das divergências entre modelos, são utilizadas algumas respostas dos participantes e as variáveis de precisão, cobertura (ou eficácia) e eficiência.

A precisão, que é um indicador de exatidão, se refere ao número de divergências corretas (válidas) identificadas, com relação ao número total de divergências identificadas. Seu valor é fornecido pela seguinte fórmula:

$$Precisão = \frac{número\ de\ divergências\ corretas\ encontradas}{número\ total\ de\ divergências\ encontradas}$$

Já a cobertura, que é um indicador de eficácia e completude, se refere ao número divergências corretas (válidas) identificadas, com relação ao número de divergências corretas esperadas (ou seja, o número total de divergências do gabarito). Seu valor é fornecido pela seguinte fórmula:

$$Cobertura = \frac{número\ de\ divergências\ corretas\ encontradas}{número\ de\ divergências\ corretas\ existentes}$$

Por fim, no contexto deste estudo, o termo eficiência se refere ao esforço gasto durante a identificação das divergências, e é fornecido pela seguinte fórmula:

$$Eficiência\ = \frac{número\ de\ divergências\ encontradas}{tempo\ gasto}$$

Para analisar o indicador eficiência, é necessário considerar todas as respostas (incluindo-se também as incorretas e duplicatas) encontradas (GOMES *et al.*, 2009). O oráculo foi validado por meio das respostas dadas pelos participantes.

Dois exercícios (X e Y) são utilizados para a geração dos modelos com a abordagem PREViA. Vale ressaltar que estes modelos foram elaborados por alunos no contexto da disciplina de Modelagem e Projeto de Sistemas de uma universidade, no nível de graduação. Este exercício foi focado especificamente em *design* no nível mais alto da etapa de análise, ou seja, atributos e métodos não são contemplados. Os tópicos utilizados para os exercícios (domínio escolar e empresarial) são simples e de conhecimento geral.

Para a seleção dos participantes, os seguintes critérios são utilizados: (i) experiência prévia no ensino de modelagem de sistemas e/ou de arquitetura de software, (ii) formação acadêmica (graduação completa), e (iii) disponibilidade para participar no estudo.

5.3.2 Execução do Estudo

A amostra do estudo foi selecionada por conveniência. Para a execução deste estudo foram selecionados quatro participantes, a partir dos critérios apresentados. Quanto à formação acadêmica, todos os participantes possuem, no mínimo, pós-graduação em andamento: um estudante de mestrado, dois alunos de doutorado e um professor doutor, todos com experiência no ensino de modelagem. Todo o material necessário para a execução do estudo foi enviado via e-mail. Não houve apoio presencial para os participantes.

Cabe ressaltar que, devido ao caráter exploratório do estudo em termos de como professores lidam com divergências entre modelos, não houve treinamento prévio em verificação de modelos ou inspeção, para não haver viés.

O estudo foi dividido em duas etapas. Na primeira etapa, os participantes recebem um formulário (Seção C.1 do Apêndice C) com o gabarito de um exercício X (isto é, um modelo que era esperado como resultado do exercício X) e respostas dadas por dois alunos (ou seja, os modelos gerados pelos alunos durante o exercício de modelagem), e devem comparar os modelos com o gabarito sem a aplicação da abordagem PREViA. A Figura 5.25 mostra um exemplo de um modelo e uma resposta de um aluno sem a aplicação da abordagem.

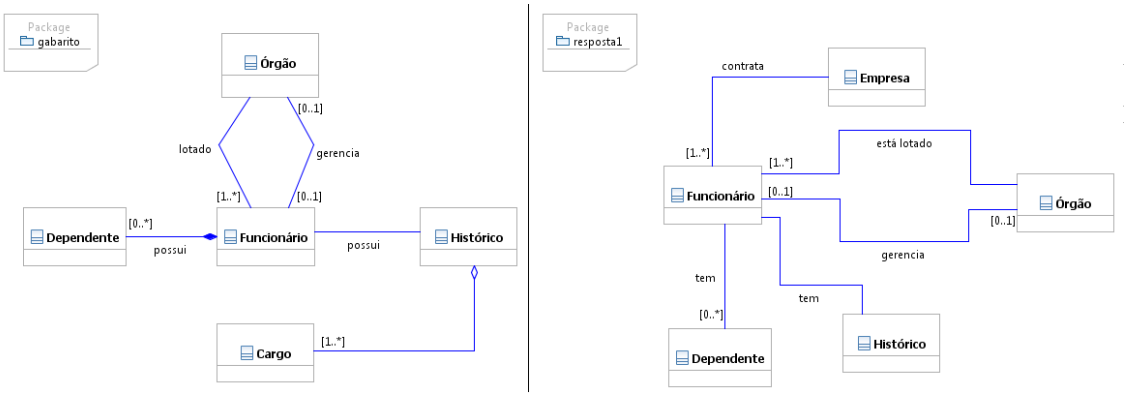


Figura 5.25 – Gabarito (à esquerda) e resposta (à direita) ao exercício, sem a aplicação da abordagem PREViA

Na segunda etapa, os participantes recebem outro formulário (Seção C.2 do Apêndice C) com o gabarito de um exercício Y e respostas dadas por outros dois alunos, com a aplicação da abordagem PREViA. A fim de diminuir o viés de aprendizagem, o domínio tratado é alterado entre as etapas (os participantes que fazem uso do domínio escolar na primeira etapa passam a utilizar o empresarial na segunda, e vice-versa). A Figura 5.26 mostra o diagrama resultante da aplicação da abordagem PREViA sobre a resposta de um aluno (os modelos que serviram de entrada para este resultado são os apresentados na Figura 5.25; a legenda utilizada encontra-se na Tabela C.1 do Apêndice C).

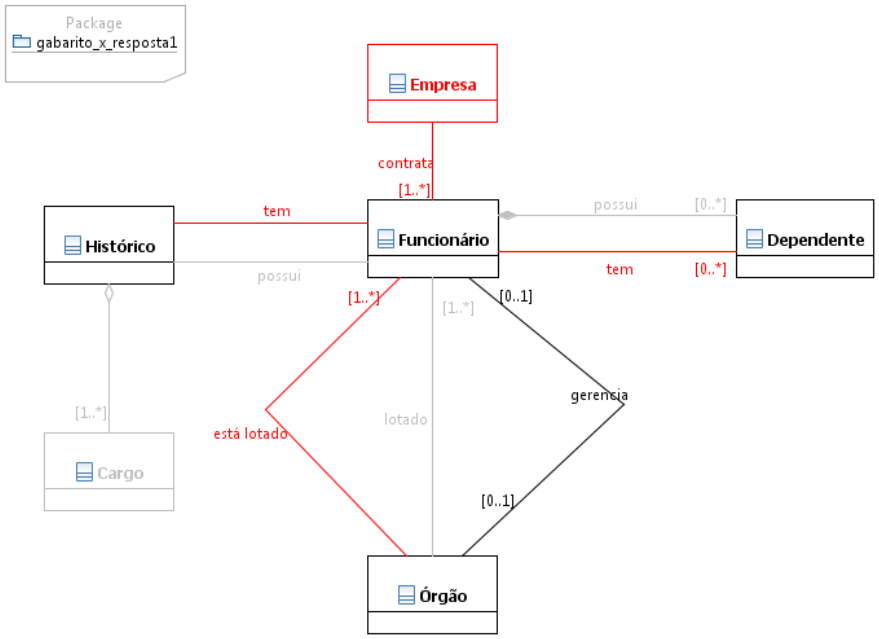


Figura 5.26 – Gabarito sobreposto por uma resposta ao exercício, com a aplicação da abordagem PREViA

Nesta figura, a classe `Cargo` representa um elemento esperado (tal como está definido no gabarito), mas não foi encontrada na resposta do aluno. Além disso, a classe `Empresa` foi encontrada na resposta do aluno, mas não era prevista, de acordo com o gabarito. Outro exemplo é o relacionamento entre as classes `Funcionário` e `Dependente`: além da diferença de nomes, uma associação de composição era esperada, mas a resposta do aluno contém uma associação simples.

Para estes exercícios, há sete tipos de divergência possíveis:

- classe ausente (e.g., uma classe que era esperada, mas que não está presente na resposta);
- classe não prevista (i.e., uma classe que não era esperada, mas que está presente na resposta);
- multiplicidades de associação diferentes (e.g., `[0..*]` em vez de `[1..*]`);
- nomes de associação diferentes (e.g., `tem` em vez de `possui`);
- associação ausente (i.e., uma associação que era esperada, mas que não está presente na resposta);
- associação não prevista (i.e., uma associação que não era esperada, mas que está presente na resposta); e

- tipos de associação diferentes (e.g., uma associação de agregação em vez de uma associação de composição).

Em ambas as etapas, foi solicitado que os participantes preenchessem um formulário com as seguintes informações: (i) o tempo gasto no exercício, (ii) os elementos identificados como divergências, e (iii) uma nota (pontuação) dada ao exercício. A nota foi utilizada como meio de verificar quão correto o exercício está (na opinião do participante) e se a corretude pode, de alguma forma, influenciar na avaliação do exercício. As informações restantes foram utilizadas para analisar o nível de detalhe em cada descrição das divergências e para calcular os valores de precisão, cobertura e eficiência.

É importante observar que a abordagem PREViA fornece apenas o realce das diferenças entre os modelos, deixando a cargo do usuário a tarefa de verificar se cada diferença caracteriza um erro ou uma alternativa de projeto (*design*). Esta informação foi compartilhada com os participantes durante o estudo. Um exemplo de tal situação pode ser encontrado na Figura 5.26 (as palavras “possui” e “tem” podem ser consideradas sinônimos).

Ao final da execução do estudo, foi distribuído um questionário *follow-up* para coletar alguns dados qualitativos acerca do estudo e da abordagem PREViA, de forma a obter dos participantes suas opiniões, benefícios percebidos, dificuldades observadas e oportunidades de melhoria. Este formulário se encontra na Seção C.3 do Apêndice C.

5.3.3 Análise dos Dados

Os resultados deste estudo foram analisados com base (i) nas divergências identificadas pelos participantes, (ii) na duração da atividade (informada durante a execução) e, principalmente, (iii) a partir das considerações feitas pelos participantes no questionário *follow-up* quanto à execução da abordagem.

5.3.3.1 Análise Quantitativa

Os dados quantitativos são utilizados para verificar se o uso de PREViA foi útil para que os participantes encontrassem as divergências.

Quanto às divergências possíveis que podem ser encontradas neste estudo, observou-se que as classes ausentes, as classes não previstas, e os tipos de associação diferentes foram os mais facilmente identificados pelos participantes (100% das

ocorrências no exercício foram encontradas), seguidos pelas associações ausentes (67% das ocorrências foram encontradas) e as multiplicidades de associação diferentes (59% das ocorrências foram encontradas). A Tabela 5.12 exibe a média e o desvio padrão dos resultados dos participantes.

Tabela 5.12 – Resultados quantitativos dos participantes

Variável	Sem PREViA		Com PREViA	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Número de divergências	6,50	2,67	7,00	5,24
Número de divergências corretas	6,63	3,07	7,25	5,95
Tempo gasto (minutos)	10,88	7,53	8,38	5,53
Eficiência (número de divergências por minuto)	0,75	0,36	0,89	0,33
Precisão	100,00%	0,00%	100,00%	0,00%
Cobertura	62,80%	24,08%	64,72%	28,82%

Como pode ser observado, os participantes encontraram uma quantidade similar de divergências corretas (válidas), com ou sem a abordagem PREViA. Isto pode ser devido a os exercícios selecionados para o estudo serem considerados simples. No entanto, ao considerar a média, menos tempo foi gasto quando se utiliza a abordagem PREViA (uma diferença de 2,5 minutos, em média). Adicionalmente, com a abordagem PREViA a média da eficiência foi maior. Em termos de precisão e cobertura, não foi observada nenhuma diferença significativa.

Com exceção do valor da precisão, em todos os casos o desvio padrão indica que estes resultados não apresentam robustez, pois se afastam razoavelmente do valor médio. No entanto, tal comportamento pode ser devido à baixa quantidade de participantes do estudo.

Ao investigar os resultados, observou-se que, para um dos participantes, alguns nomes de associação diferente não foram considerados como divergências. Outro ponto interessante é que um participante se destacou e alcançou os valores máximos de precisão e cobertura em todos os cenários. Embora ele tenha gasto um longo período de tempo na execução das atividades (13,25 minutos em média), ele obteve a maior taxa de divergências corretas (i.e., válidas). Isto pode ser explicado por seu tempo de experiência em inspeções: devido a este perfil, ele provavelmente teve um cuidado especial em descrever as diferenças, e pode ter sido meticuloso (isto é, com atenção consciente para detalhes).

Ao analisar a nota dada pelos participantes para cada resposta do exercício, um ponto interessante é que os exercícios com notas mais baixas tiveram os piores resultados usando a PREViA. A abordagem também não trouxe melhorias em termos de eficiência e eficácia (cobertura) em tais casos, visto que alguns dos participantes utilizando a PREViA apresentaram valores mais baixos destas variáveis quando comparados aos participantes sem a PREViA. Uma hipótese é que os exercícios com mais erros podem ser mais difíceis de avaliar; além disso, o significado de divergência não foi formalizado para os participantes do estudo. Apesar de cada participante ter sua própria estratégia para pontuar e classificar os exercícios, isto é uma questão que deve ser analisada posteriormente.

Na maioria dos casos, o uso da PREViA permitiu que os participantes obtivessem melhores resultados. No entanto, não é possível obter informações conclusivas devido ao número reduzido de participantes.

5.3.3.2 Análise Qualitativa

Todos os participantes consideraram a abordagem PREViA fácil de utilizar para a comparação de modelos e interpretação dos dados. Além disso, nenhum deles teve dificuldades na interpretação das informações marcadas pela PREViA. Isto traz indícios de um dos objetivos almejados pela utilização da abordagem (i.e., apoio às atividades de comparação e compreensão, facilitando este tipo de interpretação).

Alguns deles mencionaram que seria difícil realizar as atividades de comparação em modelos de larga escala, e indicaram a necessidade de se obter automaticamente informações sobre os tipos de divergência entre os modelos (por exemplo, por meio de uma lista). Isto pode ser obtido por meio da ferramenta PREViA, que utiliza o framework EMF Compare a fim de obter as diferenças entre os modelos.

A maioria dos participantes indicou que um aspecto positivo da utilização da abordagem PREViA é a facilidade na identificação das diferenças entre os modelos: “não precisa ficar olhando a toda hora para o gabarito”; “o avaliador não perde tempo com o que não é relevante”; “facilitou muito a verificação das divergências nas multiplicidades dos relacionamentos, que não é tão simples de enxergar sem esse recurso”.

Um ponto negativo (que foi unânime quanto à utilização da PREViA na correção da modelagem) é que os elementos que possuem a mesma semântica, mas que têm nomes diferentes, são considerados como divergências pela abordagem. Este ponto

fica mais crítico quando o número de elementos aumenta. No entanto, a abordagem não se propõe a fazer uma verificação semântica, restringindo-se ao aspecto sintático (uma análise semântica envolveria o processamento de linguagem natural, que não é o foco principal da abordagem, mas pode ser desenvolvido como trabalho futuro).

Também foi possível observar neste cenário que o uso da PREViA pode dificultar a correção quando o número de divergências é bastante superior ao número de similaridades, pois é gerada uma poluição visual muito grande e a percepção que a abordagem visa fornecer é fortemente prejudicada. Um comentário feito por um dos participantes corrobora com tal afirmação. Tal limitação também foi identificada no estudo descrito na Seção 5.2.

Outro ponto negativo foi a indicação das divergências entre relacionamentos. Na versão atual, a ferramenta que implementa a abordagem não é capaz de destacar apenas a parte divergente do relacionamento (isto é, apenas a divergência de nome, tipo ou cardinalidade). Por exemplo, a única diferença entre os relacionamentos `está lotado` e `lotado` (Figura 5.26) é o nome – o tipo e a multiplicidade estão de acordo com o gabarito. Esta é uma limitação do *plug-in* utilizado para o protótipo[5]. Este é um ponto que merece atenção e que deve ser contornado nas próximas versões da ferramenta.

Por fim, um dos participantes considerou que o uso da cor cinza dificultou um pouco a leitura do diagrama; isso pode ser contornado a partir do uso da ferramenta, que permite a customização das cores conforme o que se quer representar. Com isto, foi possível observar que, em termos de tarefas de correção de modelos, o uso de outras cores pode trazer uma semântica mais rica. Por exemplo, se os acertos forem representados na cor verde, o professor pode compreender mais rapidamente o que o aluno errou.

Como sugestões de melhoria, dois participantes indicaram que uma lista com todas as divergências poderia ser gerada de forma automática, deixando que o avaliador decida se tais divergências representam ou não um erro. Isto pode facilitar o trabalho de avaliação final do modelo. Esta lista já está incorporada na ferramenta PREViA.

[5] http://dev.eclipse.org/mhonarc/lists/mdt-uml2tools.dev/msg00031.html

5.3.4 Ameaças à Validade

Neste estudo, foram identificadas algumas ameaças à validade, que são apresentadas segundo a classificação descrita na Seção 5.2.4, proposta por WOHLIN *et al.* (2000). Estas ameaças são descritas a seguir.

Ameaças à validade interna (podem prejudicar o conhecimento sobre o relacionamento de causa e efeito entre o tratamento e o resultado):

- Não é possível confirmar se o tempo foi informado corretamente pelos participantes, o que compromete a medição do tempo utilizado para executar cada atividade; além disso, a unidade de tempo utilizada na análise (minutos) possui uma margem de erro inerente; e
- Os participantes podem ter se esforçado para que seu desempenho no estudo seja diferente do que seria em outras condições (visto que há proximidade entre os participantes e os pesquisadores).

Ameaças à validade externa (limitam a generalização dos resultados do estudo para outros contextos fora do ambiente avaliado):

- A representatividade dos participantes é limitada, o que se deve, em parte, aos critérios de seleção da amostra; e
- O número de exercícios utilizados é pequeno; novos estudos devem ser executados incluindo um número maior de exercícios.

Ameaças à validade de conclusão (fatos que prejudicam o estabelecimento de relacionamentos estatísticos entre o tratamento e o resultado):

- Neste estudo de viabilidade, não foram utilizados testes estatísticos para analisar os dados, pois o tamanho da amostra é bastante limitado. Desta forma, os resultados do estudo não são conclusivos: somente fornecem alguns indícios.

Ameaças à validade de construção (eventos que podem impedir que a configuração do experimento reflita adequadamente a construção do relacionamento entre o tratamento e o resultado):

- A variação da complexidade entre os domínios utilizados no estudo pode ter influenciado a identificação das divergências e a nota dada (o domínio escolar, por exemplo, possui um número maior de entidades);

- A qualidade da resposta do aluno pode exercer influência, isto é, uma resposta mais correta pode resultar em maior facilidade de correção e de identificação das divergências; e
- A seleção dos exercícios que compõem o estudo ocorreu de forma não aleatória; para cada exercício, foram selecionadas duas respostas, sendo uma com maior número de erros e uma com menor número.

5.3.5 Considerações Finais do Estudo

Este estudo analisou a aplicação da abordagem PREViA no contexto de educação em engenharia de software, mais especificamente no cenário de educação em modelagem, no que diz respeito à facilidade de interpretar e compreender as divergências entre modelos. O principal benefício esperado com a abordagem PREViA é facilitar a interpretação e compreensão das divergências entre modelos; tal benefício foi percebido por meio dos resultados do estudo, que forneceram indícios de que o uso da PREViA pode contribuir nestas atividades de ensino de engenharia de software. No entanto, esta afirmação só poderá ser confirmada após a condução de novos estudos, inclusive em cenários distintos de aplicação.

A abordagem PREViA foi originalmente planejada para lidar com cenários evolutivos, nos quais os modelos e elementos geralmente não diferem muito da versão anterior (ou, pelo menos, planeja-se diminuir o grau de divergência na medida em que o software é codificado). Já no cenário educacional, cada modelo comparado é resultante de um processo de *design* criativo, em que cada projetista (aluno) pode alcançar uma solução de uma maneira distinta (mas não necessariamente incorreta). Conforme afirmado em (WERNER *et al.*, 2010), este tipo de processo é geralmente conduzido por pessoas com formação acadêmica e profissional diferentes, usando metodologias diferentes (cada um com um determinado nível de detalhe) e que podem produzir vários tipos de resultados.

Este estudo forneceu indícios positivas do uso da abordagem no contexto da avaliação de modelos, representando um passo importante para a avaliação da PREViA tanto no contexto da educação quanto no seu propósito original. As respostas, os indicadores e os comentários foram muito interessantes e úteis, mostrando-se essenciais para verificar a viabilidade da abordagem, e foram considerados para melhorias na abordagem PREViA e nos estudos relacionados a ela.

Desta forma, este estudo de viabilidade se mostrou válido, uma vez que possibilitou a análise da abordagem em um contexto distinto do qual ela foi projetada, permitindo a identificação de diversos aperfeiçoamentos possíveis para torná-la mais adequada a este contexto. Esta avaliação teve um caráter exploratório e, como tal, levantou algumas questões interessantes e indicou o potencial de utilização da abordagem PREViA no contexto educacional. Pretende-se realizar estudos futuros com uma amostra mais representativa de participantes, com a finalidade de fornecer evidências mais conclusivas.

5.4 Considerações Finais

Neste capítulo, foi apresentada a avaliação da abordagem PREViA em dois contextos distintos, visando analisar os resultados de sua aplicação.

A partir da execução destas avaliações, foi possível observar alguns pontos fracos da abordagem, bem como identificar oportunidades de melhoria e trabalhos futuros. A avaliação também forneceu indícios de que a abordagem é capaz de auxiliar algumas tarefas de verificação de aderência e acompanhamento da evolução no contexto de projetos de software, bem como pode ser útil quando aplicada ao contexto de educação em engenharia de software.

O próximo capítulo apresenta as considerações finais deste trabalho, descrevendo algumas contribuições, limitações e possíveis trabalhos futuros.

CAPÍTULO 6 - CONCLUSÃO

6.1 Epílogo

Esta dissertação apresentou a importância da definição, documentação e atualização da arquitetura do sistema de forma a permitir o entendimento dos seus módulos, bem como a deficiência em reconhecer a distinção entre a arquitetura planejada e a implementada. Foi discutido o uso de técnicas de visualização de software como apoio a atividades de compreensão da evolução do software ao longo do tempo, e percebeu-se que as ferramentas analisadas não proviam todo o apoio necessário a estas tarefas de compreensão da evolução da arquitetura.

Neste contexto, a abordagem PREViA, descrita neste trabalho, foi proposta com o objetivo de (i) prover uma melhor percepção da aderência entre o que está sendo implementado (arquitetura emergente) com relação ao que foi projetado (arquitetura conceitual), (ii) prover uma melhor percepção sobre a evolução da implementação (arquitetura emergente), e (iii) prover uma melhor percepção sobre a evolução do projeto (*design*) do software (arquitetura conceitual), utilizando-se de conceitos e técnicas de visualização de software para este fim.

Para embasar a elaboração da abordagem PREViA, a revisão da literatura abrangeu as áreas de arquitetura e modelos de software (Capítulo 2) e visualização de software (Capítulo 3). A partir desta revisão da literatura, foram identificados os requisitos para a construção da abordagem. Um protótipo foi implementado visando prover apoio à execução da abordagem.

Dois estudos foram conduzidos no contexto desta dissertação, sendo um deles com a finalidade de avaliar se a abordagem atinge seu objetivo (ou seja, prover uma melhor percepção da evolução do projeto e da aderência entre o que foi implementado e o que foi planejado), e o outro com a finalidade de verificar a viabilidade da utilização da abordagem no contexto de educação em engenharia de software (mais especificamente em modelagem de sistemas). A partir da execução destes estudos, verificou-se que há indícios de que a abordagem auxilia na execução de tarefas que envolvem a percepção da aderência e da evolução (especialmente no que diz respeito a este segundo foco, segundo os resultados do estudo), e que algumas adaptações seriam necessárias para sua utilização no contexto de educação em engenharia de software.

6.2 Contribuições

As principais contribuições desta dissertação são:

- Identificação de características relevantes para abordagens voltadas para a visualização da evolução arquitetural;
- Abordagem PREViA, que visa prover uma melhor percepção da aderência entre o que está sendo implementado (arquitetura emergente) com relação ao que foi projetado (arquitetura conceitual), bem como prover uma melhor percepção sobre a evolução do projeto e da implementação separadamente, fazendo uso de conceitos e técnicas de visualização de software;
- Definição de métricas de precisão e cobertura voltadas para modelos arquiteturais e aplicáveis a ambos os focos de visualização (percepção da aderência e compreensão da evolução), permitindo quantificar a divergência de uma dada versão de um modelo com relação à outra;
- Um protótipo integrado ao mecanismo EvolTrack, que implementa os conceitos da abordagem e reutiliza frameworks e componentes de apoio;
- Elaboração de um pacote (incluindo o plano e os formulários) para possíveis replicações dos estudos executados, que pode servir também como base para estudos em contextos similares.

6.3 Limitações

A abordagem PREViA não é capaz de identificar se a implementação do modelo está completa e correta em termos funcionais. Desta forma, sua utilização em projetos que seguem a metodologia MDE (*Model Driven Engineering*) pode indicar níveis de precisão e cobertura não condizentes com a realidade destes projetos; por exemplo, ao gerar o código a partir do modelo, os valores destas métricas serão altos, mesmo que o código não seja funcional.

Operações de movimentação (como, por exemplo, quando uma classe pertencente a um pacote é movida para outro pacote) não são exibidas adequadamente em abordagens que fazem uso de uma visão de diagrama unificado (WENZEL, 2008), como é o caso da PREViA. Embora o framework utilizado (EMF Compare) detecte este tipo de operação, a abordagem PREViA trata estas movimentações como uma remoção seguida de uma adição.

WENZEL (2008) também indica que tais abordagens (que utilizam uma visão de diagrama unificado) não são visualmente escaláveis, isto é, não lidam bem com modelos grandes e complexos, devido a limitações de espaço de exibição. No caso da abordagem PREViA, este problema se aplica quando existem agrupamentos "naturais" no modelo que não podem ser subdivididos em outros agrupamentos. Um exemplo desta situação ocorre em pacotes que possuem um número muito grande de classes, sendo que estas não estão subdivididas em outros pacotes. No caso da abordagem PREViA, um diagrama extenso tende a dificultar a percepção, mesmo por meio da utilização do *minimap*; uma alternativa seria lançar mão dos recursos de filtragem, embora isto não resolva de maneira efetiva o problema apontado.

A granularidade da representação visual depende tanto do nível de modularização do sistema quanto da frequência do número de operações de *check-in* do projeto em análise. Caso o número de elementos por pacote seja muito grande ou haja um grande volume de dados armazenado a cada *check-in*, a percepção da evolução pode ser prejudicada (o que também pode ser parcialmente tratado por técnicas de filtragem). Um problema similar é retratado no trabalho de OHST *et al.* (2003). Já nos casos em que há um número muito grande de operações de *check-in* e a diferença entre versões é muito pequena, tal situação é contornada pelo uso do conector de fonte de dados EvolTrack-VCS, que permite a seleção das versões a partir das quais os modelos serão extraídos (SILVA, 2010).

A seleção de elementos específicos de implementação é uma atividade subjetiva e fortemente dependente do conhecimento prévio do engenheiro de software, seja na tecnologia em uso, seja no domínio para o qual o sistema foi desenvolvido; tal situação se agrava durante as atividades de manutenção e reengenharia de software caso não haja nenhum membro da equipe de projeto que conheça detalhes do desenvolvimento do software, impactando diretamente no resultado da visualização e no cálculo das métricas propostas.

Por fim, sabe-se que é possível encontrar elementos do modelo conceitual que não possuem correspondência com nenhum elemento do modelo emergente (e.g., quando o modelo conceitual é uma arquitetura de referência ou um modelo de domínio). A abordagem PREViA precisaria ser adaptada/estendida para cobrir as particularidades de cenários como este.

6.4 Perspectivas Futuras

No estudo apresentado na Seção 5.3, os participantes indicaram que os elementos que possuem a mesma semântica, mas que têm nomes diferentes, são considerados como divergências. A versão atual da abordagem PREViA restringe-se à verificação sintática dos elementos. A fim de efetuar uma análise semântica, mecanismos de processamento de linguagem natural podem ser úteis. No entanto, acredita-se que a intervenção do usuário continua sendo necessária para confirmar se os elementos considerados similares o são de fato.

No que diz respeito à implementação atual da abordagem, um dos gargalos relacionados à escalabilidade é a utilização do framework UML2Tools, utilizado pelo EvolTrack. Pretende-se analisar a possibilidade de substituição deste framework (reutilizado da implementação atual do EvolTrack) por outro que atenda a esta demanda e que permita também a utilização de outros recursos visuais integrados.

A criação de uma base de dados, composta de elementos de tecnologia já conhecidos como específicos de implementação (por exemplo, o atributo `serialVersionUID`, em Java), pode ser útil para minimizar a subjetividade da atividade de seleção destes elementos.

No que diz respeito aos rastros entre as arquiteturas, a abordagem PREViA poderia ser integrada à ferramenta ArchTrace (apresentada na Seção 2.5), a fim de que a detecção de rastros entre elementos seja mais rica em termos das políticas de gerência de rastros disponibilizadas por esta ferramenta.

Por fim, pretende-se executar um estudo de observação no contexto de projetos reais (cujos modelos arquiteturais e código fonte estejam disponíveis e sob controle de versão) visando avaliar o protótipo implementado, junto aos recursos de visualização e às métricas de precisão e cobertura adaptadas para a abordagem. De posse da ferramenta e dos dados dos projetos, os participantes do estudo (alunos de pós-graduação e profissionais da indústria) devem identificar informações consideradas relevantes no que diz respeito à aderência e evolução, tais como:

- Em que versões ocorreram desvios arquiteturais mais frequentes (e qual pode ser a causa relacionada a isto);
- Em que versões o desenvolvimento seguiu com mais rigor o que estava planejado na arquitetura conceitual;

- Em que momentos houve maior grau de evolução da arquitetura, e o que as modificações entre versões podem representar no contexto do projeto.

A partir destes dados, é possível obter informações que podem ser úteis para o gerenciamento de projetos (e.g., qual é o desenvolvedor que mais desvia da arquitetura planejada, ou em quais dias/horários os desenvolvedores desviam do planejamento com maior frequência).

REFERÊNCIAS BIBLIOGRÁFICAS

ABI-ANTOUN, M.; ALDRICH, J.; NAHAS, N.; SCHMERL, B.; GARLAN, D., 2007, "Differencing and Merging of Architectural Views", *Automated Software Engineering*, v. 15, n. 1 (Dez.), pp. 35-74.

ANDERSON, G.; GRAHAM, T.; WRIGHT, T., 2000, "Dragonfly: Linking Conceptual and Implementation Architectures of Multiuser Interactive Systems". In: *Proceedings of the 22nd International Conference on Software Engineering (ICSE'00)*, pp. 252-261, Limerick, Ireland.

APACHE SOFTWARE FOUNDATION, 2010a. Maven SCM API. *Maven SCM API - About*. Disponível em: http://maven.apache.org/scm/maven-scm-api/. Acesso em: 12 Dez 2010.

APACHE SOFTWARE FOUNDATION, 2010b. The JaxMe JavaSource Framework. *The JaxMe JavaSource framework*. Disponível em: http://ws.apache.org/jaxme/js/index.html. Acesso em: 12 Dez 2010.

AVGERIOU, P.; GUELFI, N.; MEDVIDOVIC, N., 2005, "Software Architecture Description and UML", *UML Modeling Languages and Applications*, Springer Berlin / Heidelberg, pp. 23-32.

BABAR, M.; ZHU, L.; JEFFERY, R., 2004, "A Framework for Classifying and Comparing Software Architecture Evaluation Methods". In: *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)*, pp. 309-318, Melbourne, Australia.

BAEZA-YATES, R.; RIBEIRO-NETO, B., 1999, *Modern Information Retrieval*. 1 ed. Addison Wesley.

BALDONADO, M. Q. W.; WOODRUFF, A.; KUCHINSKY, A., 2000, "Guidelines for

Using Multiple Views in Information Visualization". In: *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '00)*, pp. 110–119, Palermo, Italy.

BALL, T.; EICK, S. G., 1996, "Software Visualization in the Large", *Computer*, v. 29, n. 4, pp. 33-43.

BASILI, V.; CALDIERA, G.; ROMBACH, H., 1994, "Goal Question Metric Paradigm", *Encyclopedia of Software Engineering*, v. 1, n. edited by John J. Marciniak, John Wiley & Sons, pp. 528-532.

BASS, L.; CLEMENTS, P.; KAZMAN, R., 2003, *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I., 2005, *The Unified Modeling Language User Guide*. 2 ed. Addison-Wesley Professional.

BRAND, M. V. D.; PROTIĆ, Z.; VERHOEFF, T., 2010, "Generic Tool for Visualization of Model Differences". In: *Proceedings of the 1st International Workshop on Model Comparison in Practice, International Conference on Model Transformation (ICMT'10)*, pp. 66-75, Malaga, Spain.

BRUN, C., 2010. Compare/Helios - Every Second Counts. *Model Driven Blogging: Compare/Helios - Every Second Counts*. Disponível em: http://model-driven-blogging.blogspot.com/2010/04/comparehelios-every-second-counts.html. Acesso em: 16 Jan 2011.

BRUN, C.; PIERANTONIO, A., 2008, "Model Differences in the Eclipse Modelling Framework", *The European Journal for the Informatics Professional (UPGRADE)*, v. 9, n. 2 (Abr.), pp. 29-34.

BUERING, T.; GERKEN, J.; REITERER, H., 2006, "User Interaction with Scatterplots on Small Screens - A Comparative Evaluation of Geometric-Semantic Zoom and Fisheye Distortion", *IEEE Transactions on Visualization and Computer Graphics*, v. 12, n. 5, pp. 829-836.

CARD, S. K.; MACKINLAY, J.; SHNEIDERMAN, B., 1999, *Readings in Information Visualization: Using Vision to Think*. 1 ed. Morgan Kaufmann.

CARNEIRO, G.; ROBERTO JÚNIOR, P.; NUNES, A.; MENDONÇA, M., 2010, "An Eclipse-Based Multi-Perspective Environment to Visualize Software Coupling". In: *Anais da Sessão de Ferramentas, I Congresso Brasileiro de Software (CBSoft'10)*, p. 6, Salvador, Brasil.

CARVER, J.; JACCHERI, L.; MORASCA, S.; SHULL, F., 2003, "Issues in Using Students in Empirical Studies in Software Engineering Education". In: *Proceedings of the 9th International Symposium on Software Metrics (METRICS'03)*, pp. 239-249, Sydney, Australia.

CASERTA, P.; ZENDRA, O., 2010, "Visualization of the Static Aspects of Software: A Survey", *IEEE Transactions on Visualization and Computer Graphics*, pp. 1-20.

CEPÊDA, R. D. S. V.; MAGDALENO, A. M.; MURTA, L. G. P.; WERNER, C. M. L., 2010, "EvolTrack: Improving Design Evolution Awareness in Software Development", *Journal of the Brazilian Computer Society*, v. 16, n. 2 (Ago.), pp. 117-131.

CHANGEVISION, 2010. astah* community. *astah* community - FREE UML Modeling Tool*. Disponível em: http://astah.change-vision.com/en/product/astah-community.html. Acesso em: 18 Dez 2010.

CHEN, C., 2006, *Information Visualization: Beyond the Horizon*. 2 ed. Springer.

CLEMENTS, P.; SHAW, M., 2009, ""The Golden Age of Software Architecture" Revisited", *IEEE Software*, v. 26, n. 4, pp. 70-72.

CLEMENTS, P.; BACHMANN, F.; BASS, L.; GARLAN, D.; IVERS, J.; LITTLE, R.; MERSON, P.; NORD, R.; STAFFORD, J., 2010, *Documenting Software Architectures: Views and Beyond*. 2 ed. Addison-Wesley Professional.

CMMI PRODUCT TEAM, 2006, "CMMI® for Development, Version 1.2", n. CMU/SEI-2006-TR-008 (Ago.), p. 561.

COCKBURN, A.; KARLSON, A.; BEDERSON, B. B., 2008, "A Review of Overview+Detail, Zooming, and Focus+Context Interfaces", *ACM Computing Surveys*, v. 41, n. 1, pp. 1-31.

COLLINS-SUSSMAN, B.; FITZPATRICK, B. W.; PILATO, C. M., 2010. Version Control with Subversion. *Version Control with Subversion*. Disponível em: http://svnbook.red-bean.com/. Acesso em: 12 Dez 2010.

CONRADI, R.; MOHAGHEGHI, P.; ARIF, T.; HEGDE, L.; BUNDE, G.; PEDERSEN, A., 2003, "Object-Oriented Reading Techniques for Inspection of UML Models - An Industrial Experiment", *ECOOP 2003 - Object-Oriented Programming*, Springer Berlin / Heidelberg, pp. 69-81.

CORRÊA, C.; MURTA, L.; WERNER, C., 2008, "Odyssey-MEC: Uma Ferramenta para o Controle de Evolução no Contexto do Desenvolvimento Dirigido por Modelos". In: *Sessão de Ferramentas, XXII Simpósio Brasileiro de Engenharia de Software (SBES'08)*, pp. 1-6, Campinas, Brasil.

CULBERTSON, A., 2005, "An Experience Report on Using UML 2.0 to Document Software Architectures" (Tutorial), SEI Software Architecture Technology User Network, Pittsburgh, USA. Disponível em: http://www.sei.cmu.edu/library/assets/UML-culbertson-saturn2005.pdf. Acesso em: 31 Ago 2010.

DIAS NETO, A. C., 2009, *Seleção de Técnicas de Teste Baseado em Modelos*. Tese de D.Sc., Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, Brasil. Disponível em: http://www.cos.ufrj.br/uploadfiles/1259750753.pdf.

DIEHL, S., 2007, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. 1 ed. Springer.

DRONGOWSKI, P., 1993, "Software Architecture in Realtime Systems". In: *Proceedings of the IEEE Workshop on Real-Time Applications*, *IEEE Workshop on Real-Time Applications*, pp. 198-203, New York, USA.

D'SOUZA, D. F.; WILLS, A. C., 1999, *Objects, Components, and Frameworks with UML: the Catalysis Approach*. Addison-Wesley Longman Publishing Co., Inc.

ECLIPSE FOUNDATION, 2010a. Eclipse Model Development Tools (MDT) - UML2Tools. *Eclipse Modeling - MDT - Home*. Disponível em: http://www.eclipse.org/modeling/mdt/?project=uml2tools. Acesso em: 23 Dez 2010.

ECLIPSE FOUNDATION, 2010b. Eclipse. *Eclipse - The Eclipse Foundation open source community website*. Disponível em: http://www.eclipse.org/. Acesso em: 12 Dez 2010.

ECLIPSE FOUNDATION, 2010c. Eclipse Modeling Framework Project (EMF) - Compare. *Eclipse Modeling - EMF - Home*. Disponível em: http://www.eclipse.org/modeling/emf/?project=compare. Acesso em: 12 Dez 2010.

ECLIPSE FOUNDATION, 2010d. EMF Compare FAQ. *EMF Compare/FAQ - Eclipsepedia*. Disponível em: http://wiki.eclipse.org/EMF_Compare/FAQ. Acesso em: 20 Dez 2010.

EELES, P., 2006. The Benefits of Software Architecting. Disponível em: http://www.ibm.com/developerworks/rational/library/may06/eeles/index.html. Acesso em: 12 Set 2010.

ESTIVALETE, P. B., 2007, *Documentação da Arquitetura de Sistemas e Frameworks para o Processamento e Análise de Imagens: Uma Abordagem Baseada em Visões da UML e Padrões*. Dissertação de M.Sc., UFSM, Santa Maria, RS, Brasil.

ESTUBLIER, J., 2000, "Software Configuration Management: a Roadmap". In: *Proceedings of the Conference on The Future of Software Engineering*, *22nd*

International Conference on Software Engineering (ICSE'00), pp. 279-289, Limerick, Ireland.

FLOGGY OPEN SOURCE GROUP, 2010. Floggy - J2ME Persistence Framework. *Floggy - J2ME persistence framework - Welcome to Floggy*. Disponível em: http://floggy.sourceforge.net/. Acesso em: 19 Jan 2011.

GALLAGHER, K.; HATCH, A.; MUNRO, M., 2008, "Software Architecture Visualization: An Evaluation Framework and Its Application", *IEEE Transactions on Software Engineering*, v. 34, n. 2 (Mar.), pp. 260-270.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. M., 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*. 1 ed. Addison-Wesley Professional.

GARLAN, D.; SHAW, M., 1993, "An Introduction to Software Architecture". , *Advances in Software Engineering and Knowledge Engineering*, pp. 1-39, Singapore.

GARLAN, D.; SHAW, M., 1994, *An Introduction to Software Architecture*, Technical Report CMU-CS-94-166, Carnegie Mellon University. Disponível em: http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf.

GOMES, M.; VIANA, D.; CHAVES, L.; CASTRO, A.; VAZ, V. T.; SOARES, A.; TRAVASSOS, G. H.; CONTE, T., 2009, "WDP-RT: Uma técnica de leitura para inspeção de usabilidade de aplicações Web". In: *Proceedings of the 6th Experimental Software Engineering Latin American Workshop (ESELAW 2009)*, pp. 124-133, São Carlos, Brasil.

GRAAF, B., 2007a, "Model-Driven Evolution of Software Architectures". In: *Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR '07)*, pp. 357-360, Amsterdam, The Netherlands.

GRAAF, B., 2007b, *Model-Driven Evolution of Software Architectures*. Ph.D. Thesis, Delft University of Technology, Delft, The Hetherlands.

IBM CORP., 2008. Eclipse UML2 Tutorial. *Getting Started with UML2*. Disponível em: http://www.eclipse.org/modeling/mdt/uml2/docs/articles/Getting_Started_with_UML2/article.html. Acesso em: 18 Set 2010.

ISO/IEC, 2007, *ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE Computer Society.

ISO/IEC, 2008, *ISO/IEC 12207: System and Software Engineering - Software Life Cycle Processes*. The International Organization for the Standardization and the International Electrotechnical Commission.

JOHNSON, B.; SHNEIDERMAN, B., 1991, "Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures". In: *Proceeding of the IEEE Conference on Visualization*, pp. 284-291, San Diego, USA.

JURISTO, N.; MORENO, A. M., 2001, *Basics of Software Engineering Experimentation*. 1 ed. Kluwer Academic Publishers.

KNODEL, J.; LINDVALL, M.; MUTHIG, D.; NAAB, M., 2006a, "Static Evaluation of Software Architectures". In: *Proceedings of the 10th European Conference on Software Maintenance and Reengineering (CSMR 2006)*, pp. 279-294, Bari, Italy.

KNODEL, J.; MUTHIG, D.; NAAB, M., 2006b, "Understanding Software Architectures by Visualization - An Experiment with Graphical Elements". In: *Proceedings of the 13th Working Conference on Reverse Engineering (WCRE '06)*, pp. 39-50, Benevento, Italy.

KNODEL, J.; MUTHIG, D.; NAAB, M., 2008, "An Experiment on the Role of Graphical Elements in Architecture Visualization", *Empirical Software Engineering*, v. 13 (Dez.), pp. 693–726.

KNODEL, J.; POPESCU, D., 2007, "A Comparison of Static Architecture Compliance

Checking Approaches". In: *Proceedings of the 2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*, p. 12, Mumbai, India.

KRUCHTEN, P., 1995, "Architectural Blueprints: The 4+1 View Model of Architecture", *IEEE Software*, v. 12, n. 6, pp. 42-50.

KÜMMEL, G. Z., 2007, *Uma Abordagem para a Criação de Arquiteturas de Referência de Domínio a partir da Comparação de Modelos Arquiteturais de Aplicações*. Dissertação de M.Sc., Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, Brasil. Disponível em: http://reuse.cos.ufrj.br/files/publicacoes/mestrado/Mes_GuilhermeKummel.pdf.

LAGO, P.; VLIET, H. V., 2005, "Teaching a Course on Software Architecture". In: *Proceedings of the 18th Conference on Software Engineering Education & Training (CSEE&T)*, pp. 35-42, Ottawa, Canada.

LAMERSDORF, A.; KNODEL, J., 2006, *Design and Implementation of a Customizable Metrics Plug-in in Eclipse* IESE-Report 091.06/E. Kaiserslautern, Fraunhofer Institute for Experimental Software Engineering (IESE). Disponível em: http://publica.fraunhofer.de/documents/N-48222.html.

LANZA, M.; DUCASSE, S., 2003, "Polymetric Views - A Lightweight Visual Approach to Reverse Engineering", *IEEE Transactions on Software Engineering*, v. 29, n. 9, pp. 782-795.

LEHMAN, M. M., 1996, "Laws of Software Evolution Revisited", *Software Process Technology*, Springer Berlin / Heidelberg, pp. 108-124.

LETHBRIDGE, T. C.; DIAZ-HERRERA, J.; LEBLANC, R. J. J.; THOMPSON, J. B., 2007, "Improving Software Practice through Education: Challenges and Future Trends". In: *Proceedings of the Workshop on the Future of Software Engineering (FOSE'07), 29th International Conference on Software Engineering (ICSE'07)*, pp. 12–28, Minneapolis, MN, USA.

LEUNG, Y. K.; APPERLEY, M. D., 1994, "A Review and Taxonomy of Distortion-Oriented Presentation Techniques", *ACM Transactions on Computer-Human Interaction (TOCHI)*, v. 1, n. 2 (Jun.), pp. 126–160.

LINTERN, R.; MICHAUD, J.; STOREY, M.; WU, X., 2003, "Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse". In: *Proceedings of the 2003 ACM Symposium on Software Visualization*, pp. 47-ff, San Diego, California, USA.

MAFRA, S. N.; TRAVASSOS, G. H., 2006, *Estudos Primários e Secundários Apoiando a Busca por Evidência em Engenharia de Software*, Relatório Técnico ES 687/06, COPPE/UFRJ. Disponível em: http://lens.cos.ufrj.br:8080/ESEWEB/materials/Mafra_Travassos_RT68706.pdf.

MEDVIDOVIC, N.; TAYLOR, R. N., 2000, "A Classification and Comparison Framework for Software Architecture Description Languages", *IEEE Transactions on Software Engineering*, v. 26, n. 1, pp. 70-93.

MERSON, P., 2005, "Como Documentar Arquitetura de Software" (Minicurso), XIX Simpósio Brasileiro de Engenharia de Software (SBES'05), Uberlândia, Brasil. Disponível em: http://www.camposmf.eti.br/UML/PMsbbd_sbes2005.pdf. Acesso em: 30 Ago 2010.

MUKHERJEA, S.; FOLEY, J., 1996, "Requirements and Architecture of an Information Visualization Tool", In: Wierse, A., Grinstein, G., Lang, U. [orgs.] (eds), *Database Issues for Data Visualization*, Springer Berlin / Heidelberg, pp. 57-75.

MURPHY, G.; NOTKIN, D.; SULLIVAN, K., 2001, "Software Reflexion Models: Bridging the Gap between Design and Implementation", *IEEE Transactions on Software Engineering*, v. 27, n. 4 (Abr.), pp. 364-380.

MURTA, L.; CORRÊA, C.; PRUDÊNCIO, J. G.; WERNER, C., 2008, "Towards Odyssey-VCS 2: Improvements over a UML-Based Version Control System". In: *Proceedings of the 2008 International Workshop on Comparison and Versioning of*

Software Models (CVSM '08), 30th International Conference on Software Engineering (ICSE'08), pp. 25-30, Leipzig, Germany.

MURTA, L. G. P.; VAN DER HOEK, A.; WERNER, C. M. L., 2007, "Continuous and Automated Evolution of Architecture-to-Implementation Traceability Links", *Automated Software Engineering*, v. 15, n. 1 (Nov.), pp. 75-107.

NAAB, M.; MUTHIG, D.; KNODEL, J.; FORSTER, T., 2005, *Evaluation of Graphical Elements and their Adequacy for the Visualization of Software Architectures* IESE-Report 078.05/E, Fraunhofer IESE.

OHST, D.; WELLE, M.; KELTER, U., 2003, "Differences Between Versions of UML Diagrams", *SIGSOFT Softw. Eng. Notes*, v. 28, n. 5, pp. 227-236.

OMG, 2007, *XMI 2.2.1 Specification*, XMI specification formal/2007-12-01, Object Management Group.

OMG, 2009, *Unified Modeling Language (UML) 2.2, Superstructure*, Superstructure specification formal/2009-02-02, Object Management Group. Disponível em: http://www.omg.org/spec/UML/2.2/Superstructure/PDF/.

PERRY, D. E.; WOLF, A. L., 1992, "Foundations for the Study of Software Architecture", *ACM SIGSOFT Software Engineering Notes*, v. 17, n. 4 (Out.), pp. 40–52.

PRUDÊNCIO, J. G. G., 2008, *Orion: Uma Abordagem para Seleção de Políticas de Controle de Concorrência*. Dissertação de M.Sc., Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, Brasil. Disponível em: http://reuse.cos.ufrj.br/files/publicacoes/mestrado/Mes_JoaoPrudencio.pdf.

RODRIGUES, C. S. C.; WERNER, C. M. L., 2009, "Software Architecture Teaching: A Systematic Review". In: *Proceedings of the 9th IFIP World Conference on Computers in Education (WCCE'09)*, pp. 1-10, Bento Gonçalves, Brasil.

ROTO, V.; POPESCU, A.; KOIVISTO, A.; VARTIAINEN, E., 2006, "Minimap: a Web Page Visualization Method for Mobile Phones". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*, pp. 35-44, Montreal, Canada.

SAS INSTITUTE INC., 2011. JMP® Statistical Discovery Software. *JMP | Software*. Disponível em: http://www.jmp.com/software/. Acesso em: 10 Jan 2011.

SCHIPPER, A.; FUHRMANN, H.; VON HANXLEDEN, R., 2009, "Visual Comparison of Graphical Models". In: *Proceedings of the 14th IEEE International Conference on Engineering of Complex Computer Systems*, pp. 335-340, Potsdam, Germany.

SCHOTS, M.; RODRIGUES, C. S.; WERNER, C.; MURTA, L., 2010a, "A Study on the Application of the PREViA Approach in Modeling Education". In: *Proceedings of the XXIX International Conference of the Chilean Computer Society*, pp. 1-6, Antofagasta, Chile.

SCHOTS, M.; SILVA, M.; MURTA, L.; WERNER, C., 2010b, "Verificação de Aderência entre Arquiteturas Conceituais e Emergentes Utilizando a Abordagem PREViA". In: *VII Workshop de Manutenção de Software Moderna (WMSWM), IX Simpósio Brasileiro de Qualidade de Software (SBQS'09)*, pp. 1-8, Belém, Brasil.

SEIDEWITZ, E., 2003, "What Models Mean", *IEEE Software*, v. 20, n. 5 (Set.), pp. 26-32.

SHARIF, B.; MALETIC, J. I., 2009, "The Effect of Layout on the Comprehension of UML Class Diagrams: A Controlled Experiment". In: *Proceedings of the 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT'09), 25th International Conference on Software Maintenance (ICSM'09)*, pp. 11-18, Edmonton, Canada.

SHAW, M.; CLEMENTS, P., 2006, "The Golden Age of Software Architecture", *IEEE Software*, v. 23, n. 2, pp. 31-39.

SHAW, M.; GARLAN, D., 1996, *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, Inc.

SHI, K.; IRANI, P.; LI, B., 2005, "An Evaluation of Content Browsing Techniques for Hierarchical Space-Filling Visualizations", *IEEE Symposium on Information Visualization (INFOVIS 2005)* (Out.), pp. 81-88.

SHULL, F.; CARVER, J.; TRAVASSOS, G. H., 2001, "An Empirical Methodology for Introducing Software Processes", *ACM SIGSOFT Software Engineering Notes*, v. 26, n. 5 (Set.), pp. 288–296.

SILVA, M. A., 2010, *IAVEMS: Infraestrutura de Apoio à Visualização da Evolução de Métricas de Software*. Projeto Final de Curso, UFRJ, Rio de Janeiro, Brasil.

SOFTEX, 2009, "Guia de Implementação – Parte 4: Fundamentação para Implementação do Nível D do MR-MPS", *MPS.BR – Melhoria de Processo do Software Brasileiro*, n. 2009 (Ago.), p. 49.

SONI, D.; NORD, R. L.; HOFMEISTER, C., 1995, "Software Architecture in Industrial Applications". In: *Proceedings of the 17th International Conference on Software Engineering (ICSE)*, pp. 196-207, Seattle, Washington, USA.

STASKO, J., 2005, "Focus + Context" (Lecture Slides), Georgia Tech College of Computing, Atlanta, Georgia, USA. Disponível em: http://www.cc.gatech.edu/classes/AY2005/cs7450_spring/Talks/11-focuscontext.pdf. Acesso em: 13 Jan 2011.

STOREY, M. D.; ČUBRANIĆ, D.; GERMAN, D. M., 2005, "On the Use of Visualization to Support Awareness of Human Activities in Software Development". In: *Proceedings of the 2005 ACM Symposium on Software Visualization (SoftVis'05)*, pp. 193-216, St. Louis, Missouri, USA.

SU, S., 2010, "Interacting with Visualizations" (Lecture Slides), Tufts University,

Medford, Massachusetts, USA. Disponível em: http://www.cs.tufts.edu/comp/150VIZ/pdf/comp150-11-Interaction.pdf. Acesso em: 12 Jan 2011.

SUN, D.; WONG, K., 2005, "On Evaluating the Layout of UML Class Diagrams for Program Comprehension". In: *Proceedings of the 13th International Workshop on Program Comprehension (IWPC'05)*, pp. 317-326, St. Louis, Missouri, USA.

TAYLOR, R. N.; MEDVIDOVIC, N.; DASHOFY, E. M., 2009, *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing.

TELEA, A.; VOINEA, L.; SASSENBURG, H., 2010, "Visual Tools for Software Architecture Understanding: A Stakeholder Perspective", *IEEE Software*, v. 27, n. 6, pp. 46-53.

TREUDE, C.; BERLIK, S.; WENZEL, S.; KELTER, U., 2007, "Difference Computation of Large Models". In: *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE '07)*, pp. 295-304, Dubrovnik, Croatia.

UHRIG, S., 2008, "Matching Class Diagrams: With Estimated Costs Towards the Exact Solution?". In: *Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models (CVSM '08), 30th International Conference on Software Engineering (ICSE'08)*, pp. 7-12, Leipzig, Germany.

VASCONCELOS, A. P. V., 2007, *Uma Abordagem de Apoio à Criação de Arquiteturas de Referência de Domínio Baseada na Análise de Sistemas Legados*. Tese de D.Sc., Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, Brasil. Disponível em: http://reuse.cos.ufrj.br/files/publicacoes/doutorado/Dou_Aline.pdf.

VILLELA, R. T. N. B., 2009, *Conformação Arquitetural utilizando Restrições de Dependência entre Módulos*. Dissertação de M.Sc., Pontifícia Universidade Católica

de Minas Gerais (PUC-MG), Belo Horizonte, Brasil. Disponível em: http://homepages.dcc.ufmg.br/~mtov/diss/2009_terra.pdf.

VÖLTER, M., 2007, "Software Architecture Documentation in the Real World" (Tutorial), 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Montreal, Canada. Disponível em: http://www.voelter.de/data/presentations/SoftwareArchitectureDocumentation.pdf. Acesso em: 1 Set 2010.

WARE, C., 2008, *Visual Thinking: for Design*. Morgan Kaufmann.

WENZEL, S., 2008, "Scalable Visualization of Model Differences". In: *Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models (CVSM '08), 30th International Conference on Software Engineering (ICSE'08)*, pp. 41-46, Leipzig, Germany.

WENZEL, S.; KELTER, U., 2008, "Analyzing Model Evolution". In: *Proceedings of the 30th International Conference on Software Engineering (ICSE '08)*, pp. 831-834, Leipzig, Germany.

WERNER, C.; CEPÊDA, R.; SCHOTS, M.; MURTA, L., 2010, "How Design Style Relates to the Representational Power of Design Outcomes". , *NSF-Sponsored Workshop on Studying Professional Software Design*, pp. 1-10, Irvine, CA, USA.

WESTHUIZEN, C. V. D.; HOEK, A. V. D., 2002, "Understanding and Propagating Architectural Changes". In: *Proceedings of the Working IFIP Conference on Software Architecture (WICSA'02)*, pp. 95-109, Deventer, The Netherlands.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A., 2000, *Experimentation in Software Engineering: An Introduction*. 1 ed. Norwell, USA, Kluwer Academic Publishers.

WU, J.; STOREY, M. D., 2000, "A Multi-Perspective Software Visualization

Environment". In: *Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research*, pp. 1-10, Mississauga, Ontario, Canada.

XING, Z.; STROULIA, E., 2005, "UMLDiff: An Algorithm for Object-Oriented Design Differencing". In: *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'05)*, pp. 54-65, Long Beach, California, USA.

APÊNDICE A - ALGORITMO DE CÁLCULO DE PRECISÃO E COBERTURA

Este apêndice contém o algoritmo utilizado pela abordagem PREViA para fornecer os valores de precisão e cobertura, seguido de um exemplo prático de cálculo destas métricas no contexto de modelos do projeto *Floggy*, descrito na Subseção 4.4.2.

```
int nci <- 0, nei <- 0, ni <- 0, nc <- 0;

// Calcula recursivamente os valores de precisão e cobertura
procedimento calcula(Elemento elemento) {
   se (elemento.obterSubelementos() = NULO) {
      // efetua os cálculos apenas para o elemento
      Estereótipo diferença <- elemento.obterDiferença();
      se (diferença = NULO) {
         // elemento está em ambos os modelos conceitual e emergente
         nci <- nci + 1; nc <- nc + 1; ni <- ni + 1;
      } senão {
         String tipoDeDiferença <- diferença.obterAtributo("tipo");
         se (tipoDeDiferença = "remoção") {
            nc <- nc + 1; // elemento está apenas no modelo conceitual
         } senão se (tipoDeDiferença = "adição") {
            ni <- ni + 1; // elemento está apenas no modelo emergente
            se (elemento.éEspecíficoDeImplementação()) {
               nei <- nei + 1; // não impacta o valor de precisão
            }
         } senão { // o tipo de diferença é uma modificação
            // elemento está em ambos os modelos conceitual e emergente
            nci <- nci + 1; nc <- nc + 1; ni <- ni + 1;
         }
      }
   } senão {
      // efetua os cálculos para subelementos, somando ao elemento pai
      para i de 0 até elemento.obterSubelementos().tamanho() faça {
         Elemento subelemento <- elemento.obterSubelemento(i);
         calcula(subelemento); // chamada recursiva
         nci <- nci + subelemento.nci;
         nc <- nc + subelemento.nc;
         ni <- ni + subelemento.ni;
         nei <- nei + subelemento.nei;
      }
   }
   se ((ni = 0) OU (ni - nei = 0)) {
      // todos os elementos implementados (0) foram previstos
      elemento.precisão <- 1;
   } senão {
      elemento.precisão <- nci / (ni - nei);
   }
   se (nc = 0) {
      // todos os elementos previstos (0) foram implementados
      elemento.cobertura <- 1;
   } senão {
      elemento.cobertura <- nci / nc;
   }
}
```

Elemento	Resultado da comparação	Precisão	Cobertura	n_{ci}	n_c	n_i	n_{ii}
Pacote "Floggy"	Em ambos os modelos	0,52173913	0,97297297				
Classe "AbstractFloggyAction"	Em ambos os modelos	1	1				
Atributo "selection"	Em ambos os modelos	1	1	1	1	1	0
Método "getProject()"	Em ambos os modelos	1	1	1	1	1	0
Método "run()"	Em ambos os modelos	1	1	1	1	1	0
Método "selectionChanged()"	Em ambos os modelos	1	1	1	1	1	0
Método "setActivePart()"	Em ambos os modelos	1	1	1	1	1	0
Classe "AbstractSetPropertyAction"	Em ambos os modelos	1	1				
Atributo "propertyName"	Em ambos os modelos	1	1	1	1	1	0
Método "changeProperty()"	Em ambos os modelos	1	1	1	1	1	0
Método "isEnabled()"	Em ambos os modelos	1	1	1	1	1	0
Método "run()"	Em ambos os modelos	1	1	1	1	1	0
Classe "Activator"	Em ambos os modelos	1	1				
Atributo "PLUGIN_ID"	Em ambos os modelos	1	1	1	1	1	0
Atributo "plugin"	Em ambos os modelos	1	1	1	1	1	0
Método "getDefault()"	Em ambos os modelos	1	1	1	1	1	0
Método "start()"	Em ambos os modelos	1	1	1	1	1	0
Método "stop()"	Em ambos os modelos	1	1	1	1	1	0
Classe "CollectorFactory"	Apenas na versão mais recente	0	1				
Atributo "MTJ_BUILD_HOOK_EXT_ID"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "useJarPackager"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "collector"	Apenas na versão mais recente	0	1	0	0	1	0
Método "createCollector()"	Apenas na versão mais recente	0	1	0	0	1	0
Classe "EclipseLog"	Em ambos os modelos	1	1				
Atributo "serialVersionUID"	Em ambos os modelos	1	1	1	1	1	0
Método "debug()"	Em ambos os modelos	1	1	1	1	1	0
Método "error()"	Em ambos os modelos	1	1	1	1	1	0
Método "info()"	Em ambos os modelos	1	1	1	1	1	0
Método "log()"	Em ambos os modelos	1	1	1	1	1	0
Método "warn()"	Em ambos os modelos	1	1	1	1	1	0
Classe "FloggyBuilder"	Em ambos os modelos	1	1				
Atributo "BUILDER_ID"	Em ambos os modelos	1	1	1	1	1	0
Atributo "PERSISTABLE_CLASS_NAME"	Em ambos os modelos	1	1	1	1	1	0
Atributo "RECORDSTORE_CLASS_NAME"	Em ambos os modelos	1	1	1	1	1	0
Método "log()"	Em ambos os modelos	1	1	1	1	1	0
Método "getLog()"	Em ambos os modelos	1	1	1	1	1	0
Método "build()"	Em ambos os modelos	1	1	1	1	1	0
Método "cleanFolder()"	Em ambos os modelos	1	1	1	1	1	0
Método "copyFiles()"	Em ambos os modelos	1	1	1	1	1	0
Método "exportWeaverClasses()"	Em ambos os modelos	1	1	1	1	1	0
Classe "FloggyNature"	Em ambos os modelos	1	1				
Atributo "NATURE_ID"	Em ambos os modelos	1	1	1	1	1	0
Atributo "project"	Em ambos os modelos	1	1	1	1	1	0
Método "configure()"	Em ambos os modelos	1	1	1	1	1	0
Método "deconfigure()"	Em ambos os modelos	1	1	1	1	1	0
Método "getProject()"	Em ambos os modelos	1	1	1	1	1	0
Método "setProject()"	Em ambos os modelos	1	1	1	1	1	0
Classe "JarPackager"	Apenas na versão mais recente	0	1				
Implementação da interface "RuntimeCollector"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "PREFERRED_FLOGGY_FOLDER"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "FLOGGY_JAR_NAME"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "javaProject"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "floggyTemp"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "source"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "resultingJar"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "entries"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "byteData"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "packageFilter"	Apenas na versão mais recente	0	1	0	0	1	0
Atributo "log"	Apenas na versão mais recente	0	1	0	0	1	0
Método "setSource()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "setFilter ()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "run()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "getByteData()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "getEntries()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "collectEntries()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "addEntry()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "addFloggyJarToClasspath()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "exportJar()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "getOutputJar()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "setEclipseProperties()"	Apenas na versão mais recente	0	1	0	0	1	0
Classe "NoOpCollector"	Apenas na versão mais recente	0	1				
Implementação da interface "RuntimeCollector"	Apenas na versão mais recente	0	1	0	0	1	0
Método "run()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "setEclipseProperties()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "setSource()"	Apenas na versão mais recente	0	1	0	0	1	0
Interface "RuntimeCollector"	Apenas na versão mais recente	0	1				
Método "setSource()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "run()"	Apenas na versão mais recente	0	1	0	0	1	0
Método "setEclipseProperties()"	Apenas na versão mais recente	0	1	0	0	1	0
Classe "SetAddDefaultConstructorAction"	Apenas na versão menos recente	1	0				
Atributo "PROPERTY_NAME"	Apenas na versão menos recente	1	0	0	1	0	0
Classe "SetGenerateSourceAction"	Em ambos os modelos	1	1				
Atributo "PROPERTY_NAME"	Em ambos os modelos	1	1	1	1	1	0
Classe "ToggleNatureAction"	Em ambos os modelos	1	1				
Método "getVersion()"	Em ambos os modelos	1	1	1	1	1	0
Método "reorderCommands()"	Em ambos os modelos	1	1	1	1	1	0
Método "run()"	Em ambos os modelos	1	1	1	1	1	0
Método "setDefaultPersistentProperties()"	Em ambos os modelos	1	1	1	1	1	0
Método "updateClasspath()"	Em ambos os modelos	1	1	1	1	1	0

Figura A.1 – Valores de precisão e cobertura para os elementos do projeto *Floggy*

APÊNDICE B - INSTRUMENTOS UTILIZADOS NA AVALIAÇÃO DA ABORDAGEM NO CONTEXTO DE PROJETOS DE SOFTWARE

Este apêndice apresenta os instrumentos utilizados na avaliação da abordagem PREViA no contexto de projetos de software. Para economia de espaço, os trechos em branco reservados para respostas são omitidos.

B.1. Formulário de Consentimento

O formulário de consentimento foi entregue antes do início do estudo, de forma a obter o consentimento do participante. Para dar respaldo a tal consentimento, o formulário provê algumas informações acerca do estudo, tais como objetivo e formato.

Avaliação da Abordagem PREViA
Formulário de Consentimento

O estudo a ser conduzido visa avaliar a viabilidade da abordagem PREViA no apoio à verificação de aderência e evolução entre modelos no desenvolvimento de software. O estudo consiste em tarefas referentes à análise de alguns diagramas de classe UML.

Eu declaro ter mais de 18 anos de idade e concordar em participar do estudo conduzido por Marcelo Schots de Oliveira na Universidade do Estado do Rio de Janeiro. Eu entendo que os trabalhos que eu desenvolver serão estudados, visando entender atributos de qualidade dos procedimentos e técnicas propostos.

Eu estou ciente de que meu nome não será divulgado em hipótese alguma. Também estou ciente de que os dados obtidos por meio deste estudo serão mantidos sob confidencialidade, e os resultados serão posteriormente apresentados de forma agregada, de modo que um participante não seja associado a um dado específico.

Da mesma forma, comprometo-me a não comunicar os meus resultados enquanto o estudo não for concluído, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e apresentado. Também entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Por fim, declaro participar de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

Pesquisador responsável

Marcelo Schots de Oliveira

Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ

Professores responsáveis

Profa. Claudia Maria Lima Werner

Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ

Prof. Leonardo Paulino Gresta Murta

Instituto de Computação – Universidade Federal Fluminense (UFF)

Nome (em letra de forma):_____

Assinatura:_____ Data:_____

B.2. Questionário de Caracterização

O questionário de caracterização foi entregue antes do início do estudo, com o objetivo de caracterizar o perfil de cada participante para auxiliar na análise dos dados obtidos por meio do estudo.

Avaliação da Abordagem PREViA **Questionário de Caracterização do Participante**	
Código do participante:	

Prezado(a) participante,

Este formulário contém algumas perguntas sobre sua experiência acadêmica e profissional.

1. Formação acadêmica
   ( ) Doutorado concluído
   ( ) Doutorado em andamento
   ( ) Mestrado concluído
   ( ) Mestrado em andamento
   ( ) Graduação concluída
   ( ) Graduação em andamento

Instituição acadêmica: _____

Ano de ingresso: _____ Ano de conclusão / Ano previsto de conclusão: _____

2. Formação geral
   a) Qual é sua experiência com desenvolvimento de software orientado a objetos (OO)? (marque todos os itens que se aplicam)

   ( ) Já li material sobre desenvolvimento de software OO.
   ( ) Já participei de um curso sobre desenvolvimento de software OO.
   ( ) Nunca desenvolvi software OO.
   ( ) Tenho desenvolvido software OO para uso próprio.
   ( ) Tenho desenvolvido software OO como parte de uma equipe, relacionado a um curso.
   ( ) Tenho desenvolvido software OO como parte de uma equipe, na indústria.

   b) Por favor, detalhe sua resposta. Inclua o número de semestres ou número de anos de experiência relevante em desenvolvimento de software. (por exemplo, "Eu trabalhei por 2 anos como programador de software na indústria")

_____

_____

_____

c) Por favor, indique o grau de sua experiência nas áreas a seguir, com base na escala abaixo:

0 = Nenhum

1 = Estudei em aula ou em livro

2 = Pratiquei em projetos em sala de aula

3 = Utilizei em projetos pessoais

4 = Utilizei em projetos na indústria

Área de conhecimento	**Grau de experiência**				
Engenharia de software	0	1	2	3	4
Modelagem de sistemas de informação	0	1	2	3	4
UML (*Unified Modeling Language*)	0	1	2	3	4
Inspeção de software	0	1	2	3	4
Desenvolvimento de software em Java	0	1	2	3	4

3. Experiência em domínios

Esta seção será utilizada para compreender quão familiar você está com o domínio que será utilizado para as atividades durante o experimento (o que **não** implica já ter desenvolvido sistemas para o domínio em questão). Por favor, indique o grau de experiência com base na escala abaixo:

0 = Eu não tenho familiaridade com este domínio.

1 = Eu utilizo isto algumas vezes, mas não sou um especialista.

2 = Eu sou muito familiar com este domínio.

Domínio	**Grau de experiência**		
Domínio médico-hospitalar	0	1	2

Desde já, agradecemos a sua colaboração.

Marcelo Schots de Oliveira

Cláudia Maria Lima Werner

Leonardo Gresta Paulino Murta

B.3. Instrumentos da Etapa de Verificação de Aderência

O formulário desta etapa tem como objetivo verificar a aderência de uma determinada versão do código fonte (representado por uma arquitetura emergente) com relação a uma determinada versão do modelo conceitual.

O mesmo tipo de formulário foi utilizado para ambos os contextos (com e sem a aplicação da abordagem PREViA), variando apenas o nome da etapa (ASP = Aderência Sem PREViA, e ACP = Aderência Com PREViA) em função dos diagramas de apoio. Nesta dissertação, as partes que variaram entre um formulário e outro estão diferenciadas para melhor identificação: as partes específicas da etapa ASP estão sublinhadas em linha tracejada (como neste exemplo), enquanto as partes específicas da etapa ACP estão sublinhadas em linha pontilhada (como neste exemplo).

Avaliação da Abordagem PREViA **Etapa ASP/ACP**	
Código do participante:	

Prezado(a) participante,

Esta é uma das etapas de um estudo para a avaliação da abordagem PREViA. As tarefas relacionadas a esta etapa são referentes à contextualização a seguir.

Contextualização

Você foi contratado(a) como engenheiro de software pela empresa de desenvolvimento de software MANHUMIRIM S/A, sendo alocado inicialmente em atividades de inspeção. Todos os projetos da empresa utilizam um sistema de controle de versões e empregam a notação UML em seus modelos.

Sua tarefa atual é verificar, no projeto "Prontuário Eletrônico do Paciente", a aderência de uma dada versão do código fonte (desenvolvido pelos implementadores da empresa) com relação a uma versão do modelo conceitual, elaborado por projetistas/arquitetos de software, visando garantir a consistência entre estes artefatos. Esta verificação é feita com base em um questionário.

Para a execução desta atividade, verifique se os seguintes instrumentos foram entregues:

- Etapa ASP
    - Modelo conceitual
    - Modelo emergente, que representa uma versão do código fonte
    - Conjunto de tarefas (questões A1 a A10)
- Etapa ACP

- Modelo conceitual x emergente (resultante da comparação entre o modelo conceitual e uma versão do código fonte), com a aplicação da abordagem PREViA
- Conjunto de tarefas (questões A1 a A10)

IMPORTANTE:

- Para esta atividade, sempre que for mencionado o termo "relacionamento", incluem-se **todos os tipos de relacionamento**, tais como **generalização (herança)**, **associação simples**, **agregação**, **composição** etc.
- Para esta atividade, sempre que o valor da cardinalidade de um relacionamento estiver ausente, tal valor deve ser considerado [1..1]. Vale lembrar que associações do tipo generalização (herança) não possuem cardinalidade.
- Resolva as tarefas do questionário **na ordem em que elas são apresentadas**.
- Registre o **horário de início** e o **horário de fim** de cada atividade sempre que solicitado. Se for gasto algum tempo no entendimento do modelo **antes das atividades**, este tempo **não deve ser contabilizado**.

Etapa ASP/ACP – Questões 1, 2 e 3	
Código do participante:	

Escreva agora o horário de início desta atividade: _____

Com base no modelo conceitual e no modelo emergente, responda às seguintes questões:

A1) Quais classes compõem o modelo **conceitual**? E o modelo **emergente**? Escreva o nome de cada uma delas.

A2) Considerando apenas o modelo **conceitual**, quais classes se relacionam com a classe "*AtividadeDeProntuario*"? E se for considerado apenas o modelo **emergente**, quais classes se relacionam com esta classe?

A3) Considerando apenas o modelo **conceitual**, qual é a classe que possui **maior** número de **métodos**? E se for considerado apenas o modelo **emergente**?

Escreva o horário de finalização desta atividade: _____

Etapa ASP/ACP – Questões 4, 5, 6 e 7	
Código do participante:	

Escreva agora o horário de início desta atividade: _____

Com base no modelo conceitual e no modelo emergente, responda às seguintes questões:

A4) **Quantos atributos** representam uma **divergência** com relação ao que foi projetado (isto é, **não foram previstos** no modelo conceitual)?

A5) Qual(ais) **classe(s) e/ou relacionamento(s)** foi(foram) planejado(s) – isto é, está(ão) no modelo conceitual –, mas **não foi(foram) implementado(s)** no modelo emergente?

A6) Qual(ais) **classe(s) e/ou relacionamento(s)** foi(foram) implementado(s) – isto é, está(ão) no modelo emergente –, mas **não foi(foram) planejado(s)** no modelo conceitual?

A7) Comparando os modelos conceitual e emergente, qual(ais) **método(s)** teve(tiveram) sua(s) **assinatura(s) modificada(s)**?

Escreva o horário de finalização desta atividade: _____

Etapa ASP/ACP – Questões 8, 9 e 10	
Código do participante:	

Escreva agora o horário de início desta atividade: _____

Com base no modelo conceitual e no modelo emergente, responda às seguintes questões:

A8) Considerando o requisito RF02 – "O sistema deve permitir a inclusão dos sintomas durante cada anamnese" –, você acredita que ele tenha sido implementado de alguma forma, com base nas classes presentes no modelo **conceitual**? Em caso positivo, **qual(ais) classe(s) está(ão) envolvida(s)** na implementação do requisito?

A9) Considerando o requisito RF02 – "O sistema deve permitir a inclusão dos sintomas durante cada anamnese" –, você acredita que ele tenha sido implementado de alguma forma, com base nas classes presentes no modelo **emergente**? Em caso positivo, **qual(ais) classe(s) está(ão) envolvida(s)** na implementação do requisito?

A10) Sabendo que elementos específicos de implementação não devem fazer parte do modelo conceitual, há algum elemento no modelo **emergente** que se encaixe nesta situação (isto é, **realmente não deveria estar no modelo conceitual** e, portanto, não caracteriza uma divergência)? Em caso positivo, **qual(ais) é(são)** esse(s) elemento(s)?

Escreva o horário de finalização desta atividade: _____

Obrigado pela sua colaboração.

Marcelo Schots de Oliveira
Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

Modelo Conceitual **(Etapa ASP)**

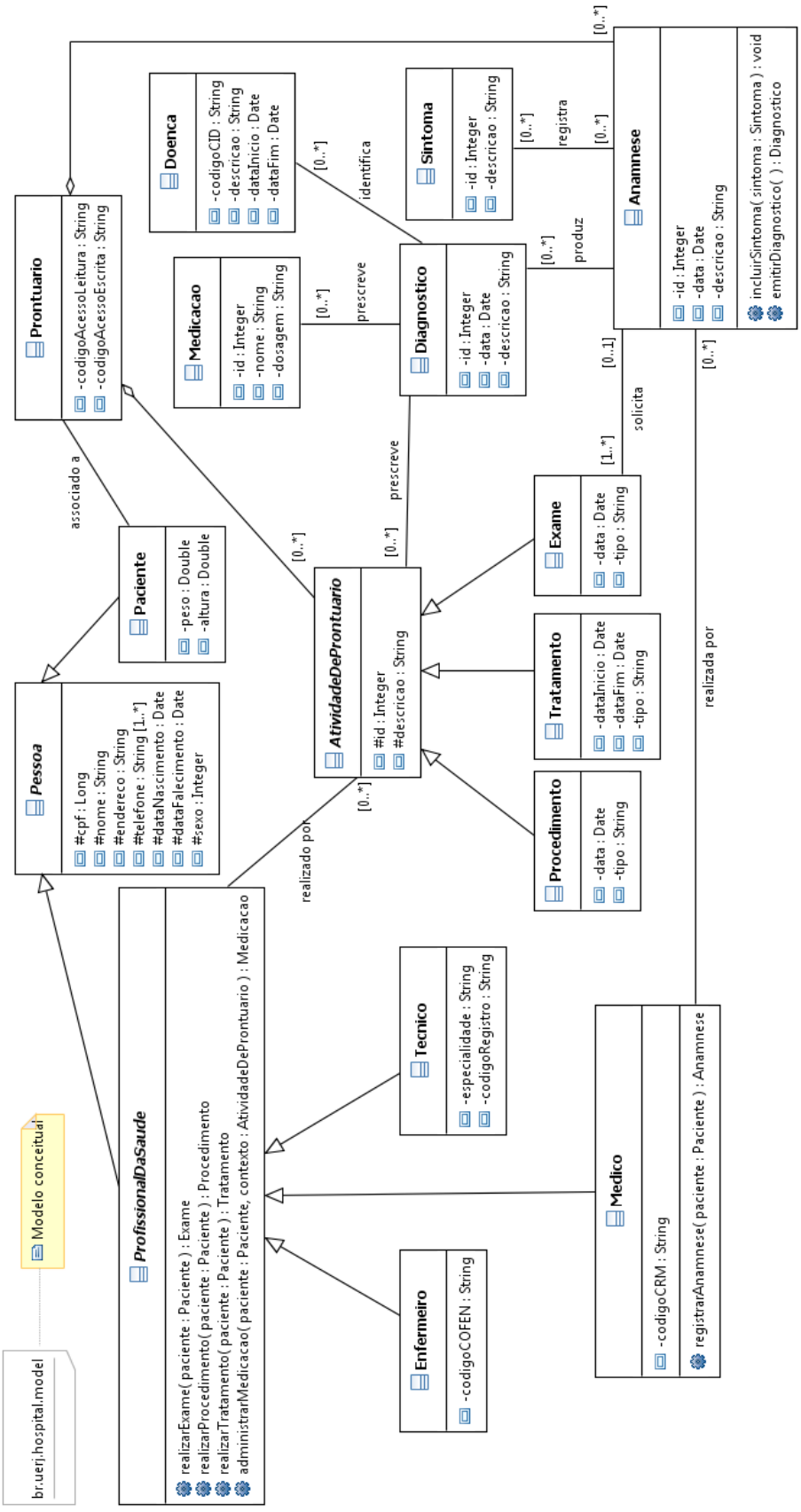


Figura B.1 – Modelo conceitual, utilizado na avaliação da abordagem

Modelo Emergente **(Etapa ASP)**

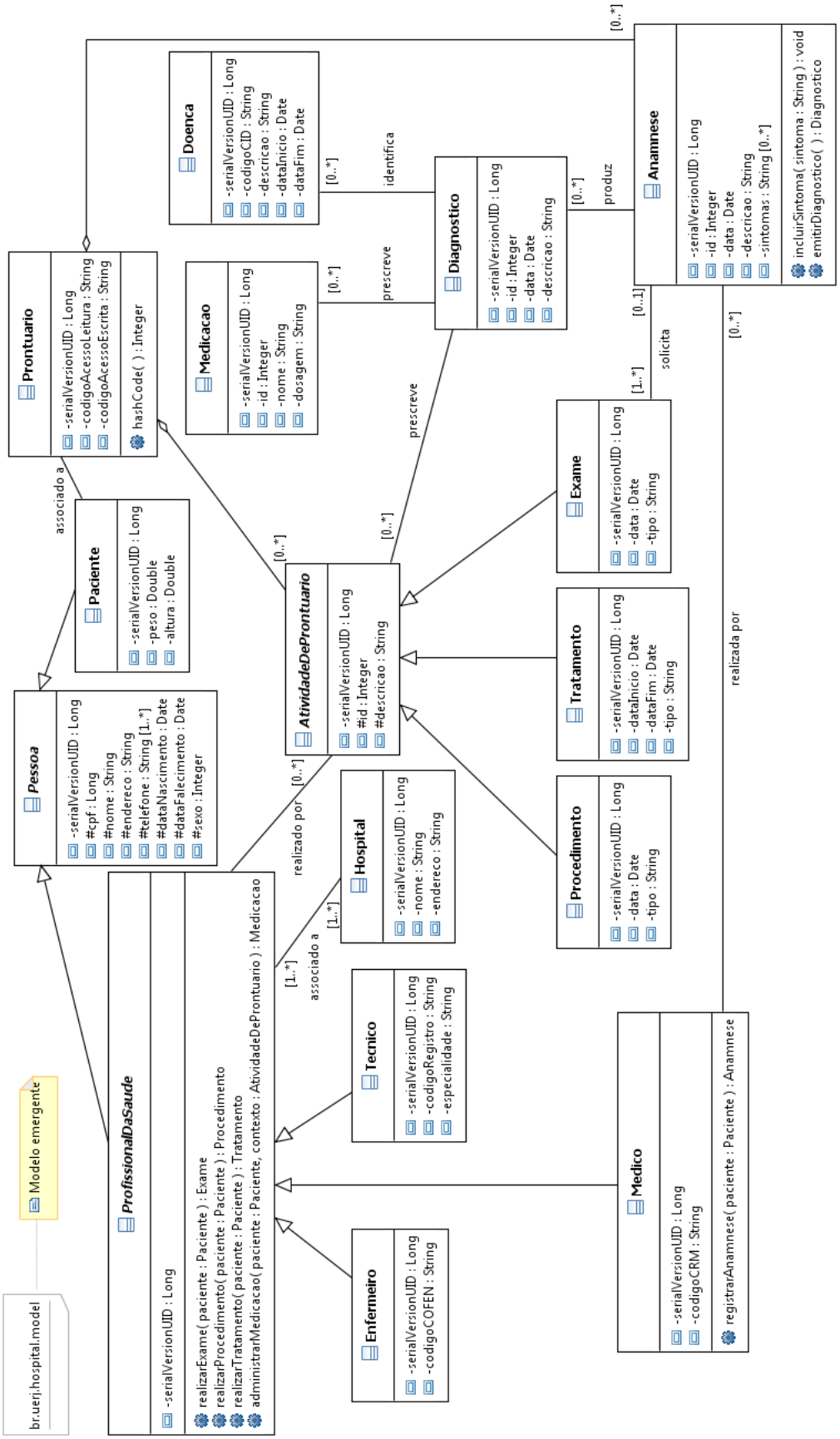


Figura B.2 – Modelo emergente, utilizado na avaliação da abordagem

Modelo Conceitual x Emergente **(Etapa ACP)**

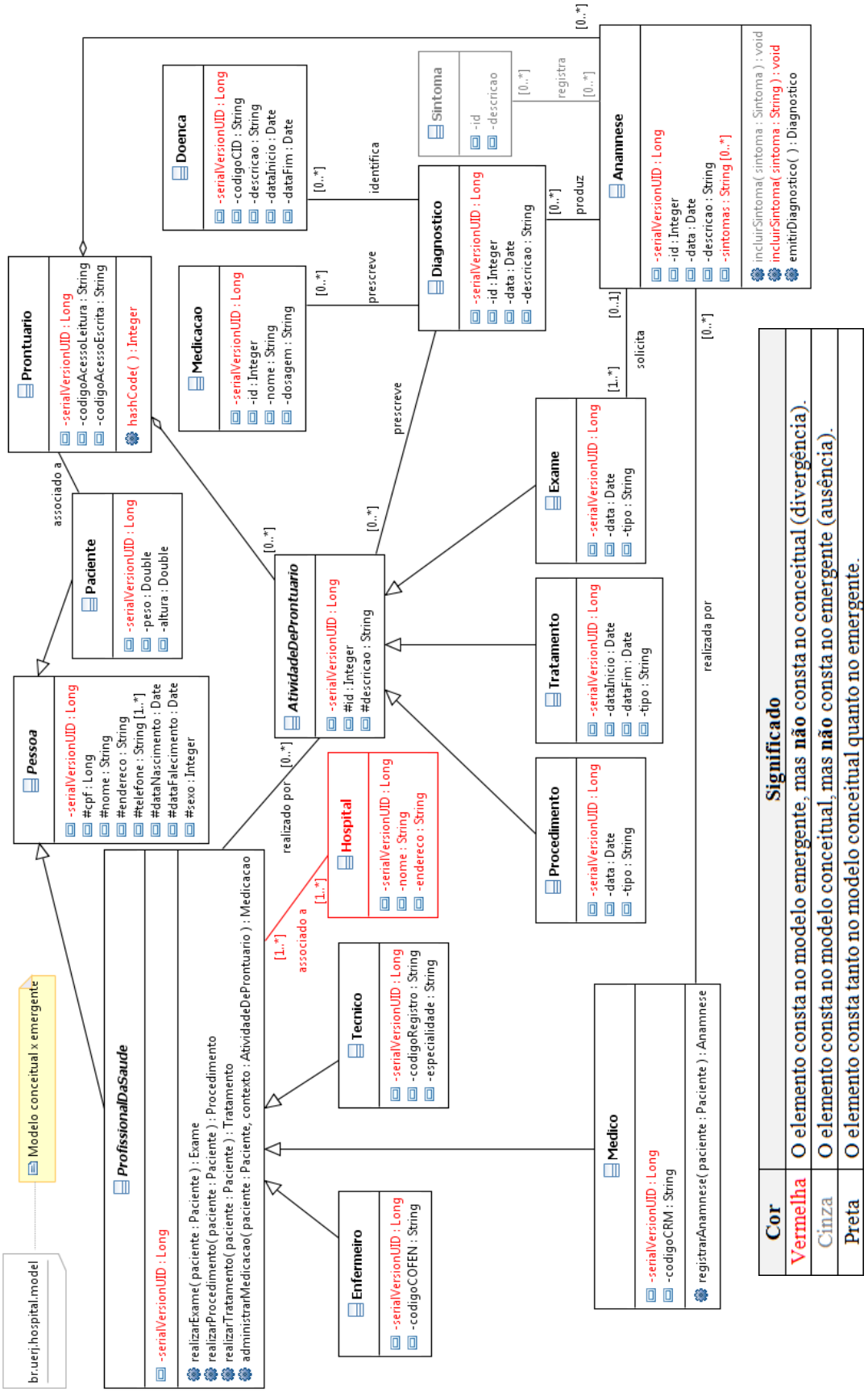


Figura B.3 – Modelo conceitual *versus* emergente, utilizado na avaliação da abordagem

B.4. Instrumentos da Etapa de Verificação da Evolução

O formulário desta etapa tem como objetivo verificar a evolução ocorrida no intervalo entre duas versões do código fonte (sendo que cada versão é representada por uma arquitetura emergente correspondente).

O mesmo tipo de formulário foi utilizado para ambos os contextos (com e sem a aplicação da abordagem PREViA), variando apenas o nome da etapa (ESP = Evolução Sem PREViA, e ECP = Evolução Com PREViA) em função dos diagramas de apoio. Nesta dissertação, as partes que variaram entre um formulário e outro estão diferenciadas para melhor identificação: as partes específicas da etapa ESP estão sublinhadas em linha tracejada (como neste exemplo), enquanto as partes específicas da etapa ECP estão sublinhadas em linha pontilhada (como neste exemplo).

Avaliação da Abordagem PREViA **Etapa ESP/ECP**	
Código do participante:	

Prezado(a) participante,

Esta é uma das etapas de um estudo para a avaliação da abordagem PREViA. As tarefas relacionadas a esta etapa são referentes à contextualização a seguir.

Contextualização

Você foi contratado(a) como engenheiro de software pela empresa de desenvolvimento de software MANHUMIRIM S/A, e está atualmente alocado em atividades de inspeção. Todos os projetos da empresa utilizam um sistema de controle de versões e empregam a notação UML em seus modelos.

Devido a uma demanda da gerência quanto ao andamento da equipe de desenvolvimento, você deve analisar o projeto "Prontuário Eletrônico do Paciente" visando compreender a evolução das atividades de implementação em um dado intervalo de tempo, reportando suas descobertas à gerência com base em um questionário.

Para a execução desta atividade, verifique se os seguintes instrumentos foram entregues:

- Etapa ESP
    - Modelo emergente (versão 345), que representa a versão 345 do código fonte
    - Modelo emergente (versão 378), que representa a versão 378 do código fonte
    - Conjunto de tarefas (questões E1 a E10)
- Etapa ECP

- Modelo emergente, resultante da comparação entre as versões 345 e 378 do código fonte, com a aplicação da abordagem PREViA
- Conjunto de tarefas (questões E1 a E10)

IMPORTANTE:

- Para esta atividade, sempre que for mencionado o termo "relacionamento", incluem-se **todos os tipos de relacionamento**, tais como **generalização (herança)**, **associação simples**, **agregação**, **composição** etc.
- Para esta atividade, sempre que o valor da cardinalidade de um relacionamento estiver ausente, tal valor deve ser considerado [1..1]. Vale lembrar que associações do tipo generalização (herança) não possuem cardinalidade.
- Resolva as tarefas do questionário **na ordem em que elas são apresentadas**.
- Registre o **horário de início** e o **horário de fim** de cada atividade sempre que solicitado. Se for gasto algum tempo no entendimento do modelo **antes das atividades**, este tempo **não deve ser contabilizado**.

Etapa ESP/ECP – Questões 1, 2 e 3	
Código do participante:	

Escreva agora o horário de início desta atividade: _____

Com base nas duas versões do modelo emergente, responda às seguintes questões:

E1) Quais classes compõem a versão 345? E a versão 378? Escreva o nome de cada uma delas.

E2) Considerando apenas a versão 345, quais classes se relacionam com a classe "Prontuario"? E se for considerada apenas a versão 378, quais classes se relacionam com esta classe?

E3) Qual(ais) classe(s) possui(em) o **menor** número de **métodos** na versão 345? E na versão 378?

Escreva o horário de finalização desta atividade: _____

Etapa ESP/ECP – Questões 4, 5, 6 e 7	
Código do participante:	

Escreva agora o horário de início desta atividade: _____

Com base nas duas versões do modelo emergente, responda às seguintes questões:

E4) Qual(ais) **classe(s)** não sofreu(eram) **nenhuma modificação** entre as versões 345 e 378?

E5) Qual(ais) **classe(s) e/ou relacionamento(s) surgiu(iram)** na versão 378, comparada à versão 345?

E6) Qual(ais) **classe(s) e/ou relacionamento(s)** foi(foram) **removido(s)** na versão 378, comparada à versão 345?

E7) Comparando as versões 345 e 378, qual(ais) **relacionamento(s)** passou(aram) a **apontar para outra classe**?

Escreva o horário de finalização desta atividade: _____

Etapa ESP/ECP – Questões 8, 9 e 10	
Código do participante:	

Escreva agora o horário de início desta atividade: _____

Com base nas duas versões do modelo emergente, responda às seguintes questões:

E8) Considerando a regra de negócio RN15 – "Todas as atividades de prontuário devem estar associadas a um profissional de saúde responsável por executá-las" –, você acredita que ela tenha sido implementada de alguma forma, com base nas classes presentes na **versão 345**? Em caso positivo, **qual(ais) classe(s) está(ão) envolvida(s)** na implementação da regra de negócio?

E9) Analise **todas as modificações** realizadas entre as versões 345 e 378. Em sua opinião, **qual(ais) mudança(s) nos requisitos** pode(m) ter levado a tais modificações?

E10) Com base nos relacionamentos presentes em cada versão, qual das versões você acha que possui **maior grau de acoplamento entre as classes**? Por quê?

Escreva o horário de finalização desta atividade: _____

Obrigado pela sua colaboração.

Marcelo Schots de Oliveira
Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

Modelo Emergente (versão 345) **(Etapa ESP)**

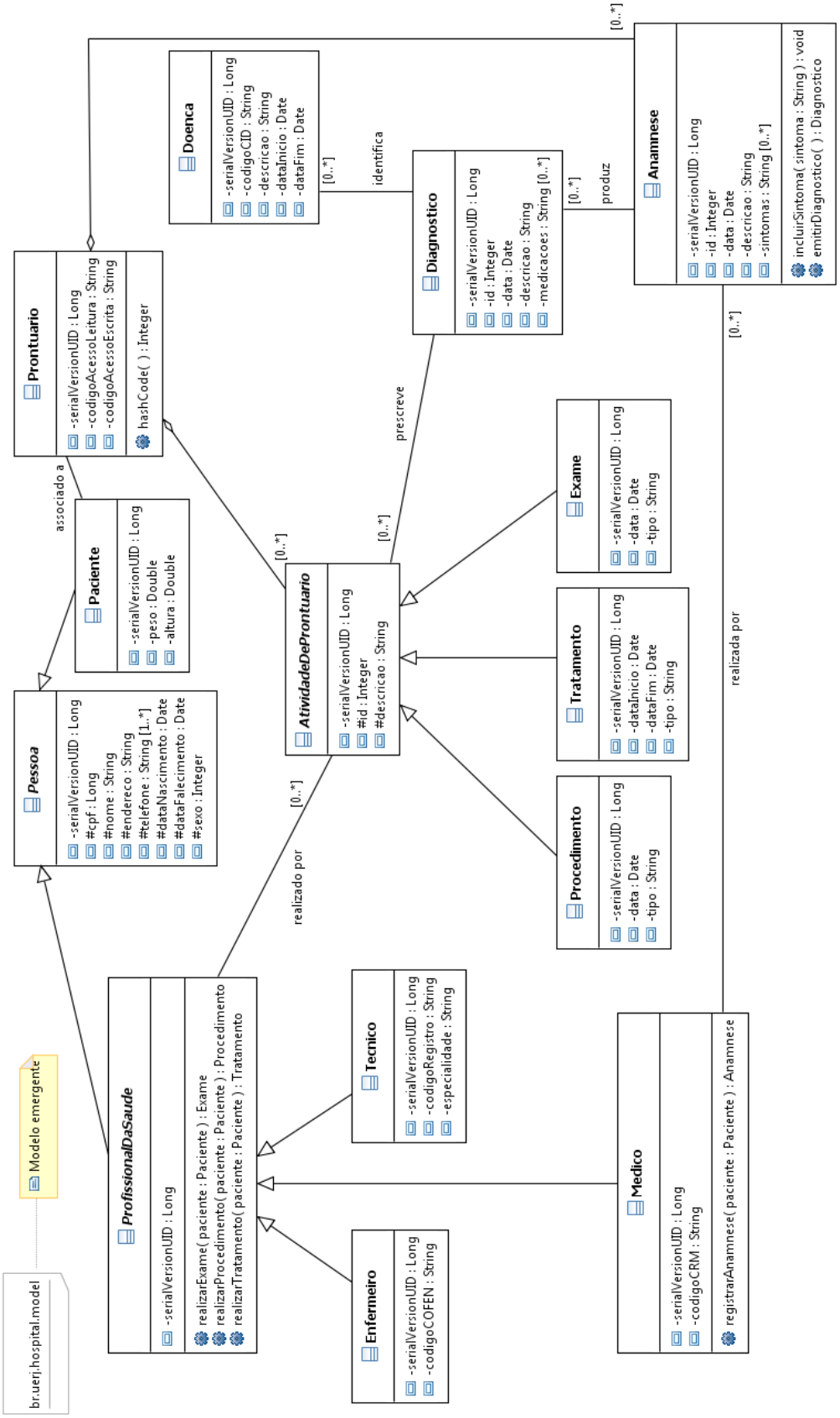


Figura B.4 – Versão 345 do modelo emergente, utilizada na avaliação da abordagem

Modelo Emergente (versão 378) **(Etapa ESP)**

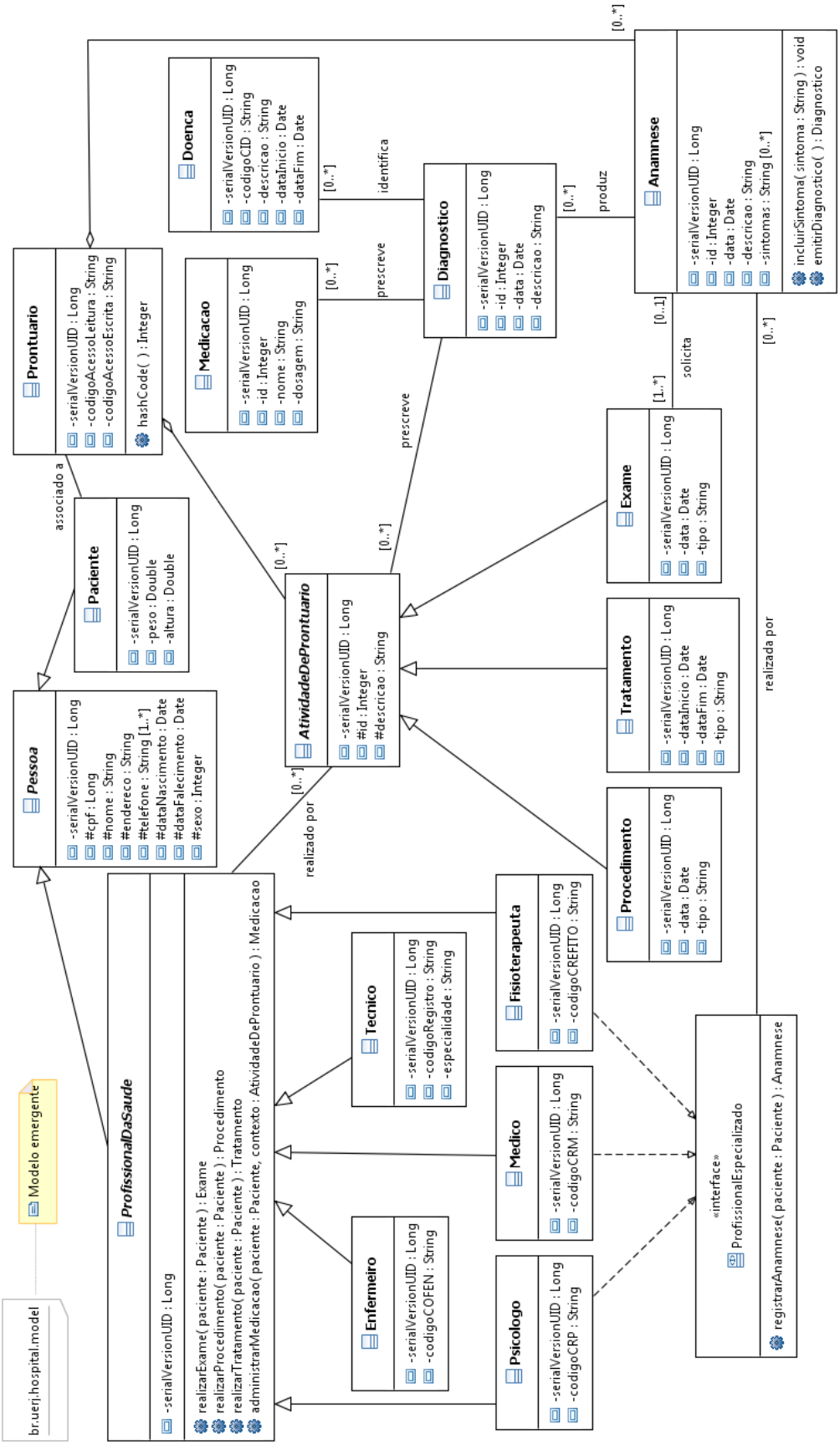


Figura B.5 – Versão 378 do modelo conceitual, utilizada na avaliação da abordagem

Evolução de Modelos (versões 345 x 378) **(Etapa ECP)**

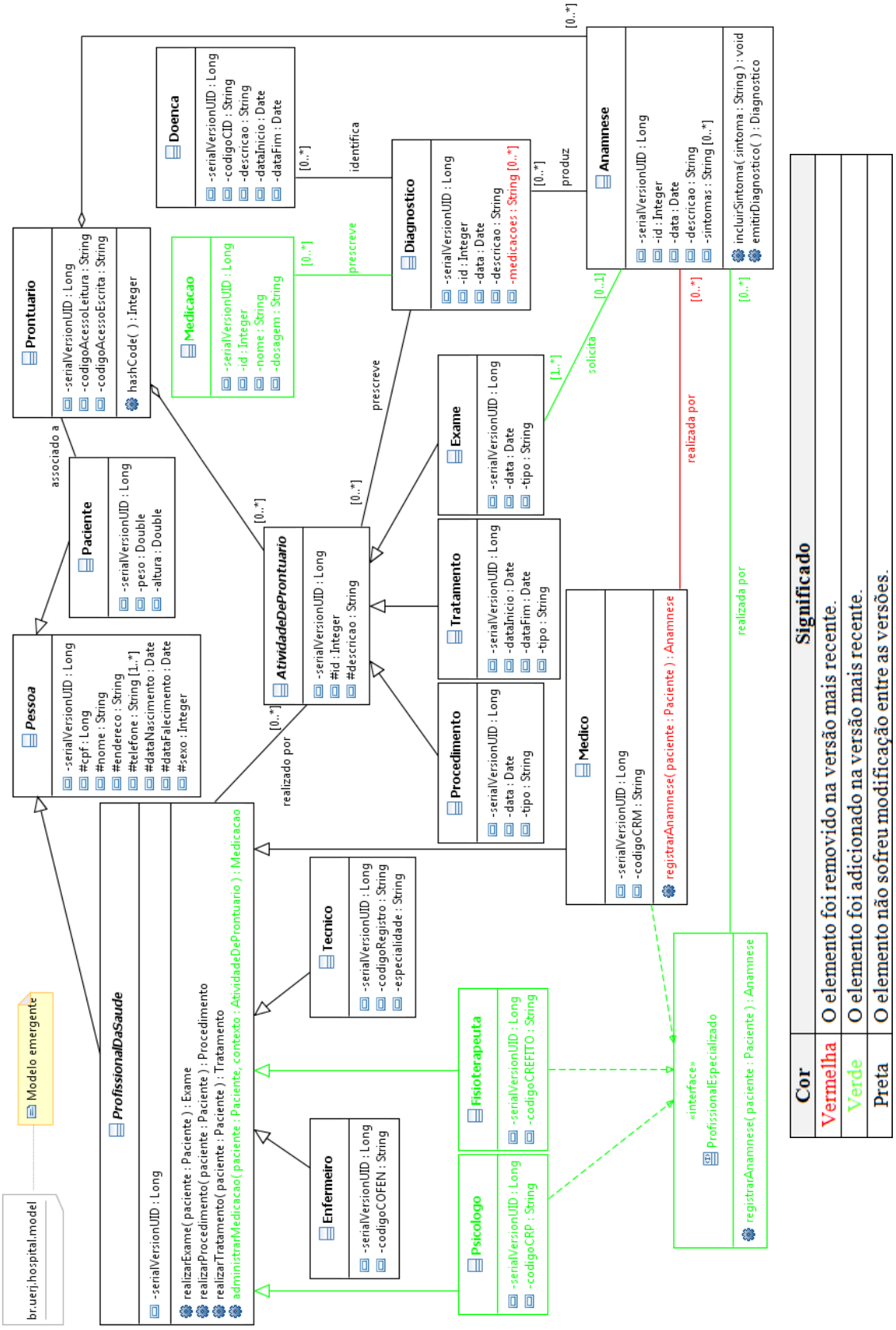


Figura B.6 – Versão 345 *versus* versão 378 do emergente, utilizada na avaliação da abordagem

B.5. Questionário *Follow-Up*

O questionário *follow-up* foi entregue na última parte do estudo, e tem como objetivo coletar as percepções e considerações sobre a execução da atividade de identificação de divergências nos modelos arquiteturais com e sem a aplicação da abordagem PREViA.

Avaliação da Abordagem PREViA **Questionário *Follow-Up***	
Número do participante:	

Prezado(a) participante,

Esta é a última parte de um estudo de utilização da abordagem PREViA. O objetivo da atividade a ser realizada é obter informações adicionais para a avaliação da abordagem aplicada no estudo, a partir das respostas às questões listadas a seguir:

1. Você sentiu alguma **dificuldade na realização das tarefas**? **Em caso positivo**, por favor, especifique-a(s).

   ( ) Sim ( ) Não

2. Considerando a **facilidade de execução** das etapas do estudo (verificação de aderência e compreensão da evolução):
   ( ) a verificação da aderência foi mais fácil
   ( ) a compreensão da evolução foi mais fácil
   ( ) ambas as etapas foram fáceis
   ( ) ambas as etapas foram difíceis

[ATENÇÃO: As questões 3 a 7 só se aplicam às etapas ACP e ECP; se você não executou nenhuma destas etapas, pule para a questão 8]

3. Com relação à **realização das tarefas**, as informações visuais da abordagem PREViA:
   ( ) facilitaram bastante
   ( ) facilitaram um pouco
   ( ) não tiveram muito impacto (indiferente / não facilitaram em nada)
   ( ) dificultaram um pouco
   ( ) dificultaram bastante

4. Você sentiu alguma **dificuldade na interpretação das informações** apresentadas pela abordagem PREViA? **Em caso positivo**, por favor, indique-a(s).

   ( ) Sim ( ) Não

5. De acordo com sua opinião, liste os **aspectos positivos** da utilização da abordagem PREViA.

6. De acordo com sua opinião, liste os **aspectos negativos** da utilização da abordagem PREViA.

7. Você possui alguma **sugestão para melhoria** da abordagem PREViA? **Em caso positivo**, por favor, especifique-a(s).

( ) Sim ( ) Não

8. Este espaço é reservado para quaisquer comentários adicionais (dificuldades, críticas e/ou sugestões) a respeito do estudo executado. Contamos com sua contribuição para que o trabalho seja aprimorado.

Novamente, gostaríamos de agradecer pela sua disponibilidade e participação neste estudo.

Marcelo Schots de Oliveira
Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

APÊNDICE C - INSTRUMENTOS UTILIZADOS NA AVALIAÇÃO DA ABORDAGEM NO CONTEXTO DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

Este apêndice apresenta os instrumentos utilizados na avaliação da abordagem PREViA no contexto de educação em engenharia de software. Para economia de espaço, os trechos em branco reservados para respostas são omitidos.

C.1. Formulários da Etapa Sem o Uso da Abordagem PREViA

Os formulários desta etapa têm como objetivo a avaliação de modelos arquiteturais elaborados por alunos, a partir da identificação de divergências entre um gabarito, sem a utilização da abordagem PREViA.

Os mesmos formulários (A, B, C e D) foram utilizados para ambos os grupos, variando apenas o Formulário B, em termos do enunciado do exercício e dos modelos utilizados, conforme pode ser visto no formulário em questão (embora os exercícios estejam apresentados sequencialmente, cada formulário continha apenas um exercício, relacionado a apenas um domínio). O formulário D é uma cópia do formulário C para as divergências entre a resposta 2 e o gabarito, e por isso não será representado.

Execução do Estudo – Formulário A	
Nome do participante:	

Prezado participante,

Por favor, leia atentamente as instruções a seguir.

Esta é a primeira parte de um estudo de viabilidade da aplicação da abordagem PREViA no contexto de Educação de Engenharia de Software. O objetivo da atividade a ser realizada é a avaliação de modelos arquiteturais elaborados por alunos, a partir da identificação de divergências entre um gabarito. Os modelos são reais e foram elaborados no contexto de uma disciplina de modelagem e projeto de sistemas.

Este formulário está dividido da seguinte forma: esta primeira parte (formulário A) contém o objetivo do estudo e as instruções. A parte B contém alguns dados a serem preenchidos, bem como a descrição do exercício, o gabarito e as respostas a serem corrigidas. A parte C diz respeito às divergências detectadas entre a primeira resposta e o gabarito. Por fim, a parte D diz respeito às divergências detectadas entre a segunda resposta e o gabarito.

IMPORTANTE: Para a correção dos exercícios, sempre que o valor da cardinalidade de uma associação estiver ausente, tal valor deve ser considerado [1..1].

Segue a ordem dos passos para a realização deste exercício:

1. Leia o exercício e o gabarito (o tempo deste passo não deve ser registrado). Não corrija o gabarito, utilize-o para a correção das respostas mesmo que discorde da solução do gabarito.
2. Registrar o horário de início da correção do exercício 1.
3. Comparar a resposta 1 com o gabarito, registrando textualmente as diferenças entre ambos na parte C do formulário.
4. Dar uma nota para o exercício 1, tomando como base o gabarito. Note que, para pontuar o exercício, pode-se considerar que o mesmo pode ser uma alternativa de projeto, sem que divergências entre este e o gabarito signifiquem erro.
5. Registrar o horário de finalização da correção do exercício 1.
6. Registrar o horário de início da correção do exercício 2.
7. Comparar a resposta 2 com o gabarito, registrando textualmente as diferenças entre ambos na parte D do formulário.
8. Dar uma nota para o exercício 2, tomando como base o gabarito. Note que, para pontuar o exercício, pode-se considerar que o mesmo pode ser uma alternativa de projeto, sem que divergências entre este e o gabarito signifiquem erro.
9. Registrar o horário de finalização da correção do exercício 2.
10. Informar o tempo gasto com cada exercício.
11. Enviar este formulário para marceloschots@gmail.com e para claudiasusie@gmail.com

Desde já, agradecemos a sua disponibilidade.

Marcelo Schots de Oliveira

Cláudia Susie Camargo Rodrigues

Cláudia Maria Lima Werner

Leonardo Gresta Paulino Murta

Execução do Estudo – Formulário B	
Nome do participante:	
Respostas	
Horário em que você começou a corrigir a resposta 1 (hora:minuto):	
Nota dada para a resposta 1:	
Horário em que você terminou de corrigir a resposta 1 (hora:minuto):	
Horário em que você começou a corrigir a resposta 2 (hora:minuto):	
Nota dada para a resposta 2:	
Horário em que você terminou de corrigir a resposta 2 (hora:minuto):	
Tempo gasto (em minutos) para a correção da resposta 1:	
Tempo gasto (em minutos) para a correção da resposta 2:	

Exercício **(domínio empresarial)**: A hierarquia de uma empresa é composta de órgãos que são subordinados. Um funcionário, que pode ter dependentes, tem que estar lotado em um

órgão e ter um histórico de cargos desde a sua contratação pela empresa. Um órgão pode ter ou não um gerente, que tem que ser um funcionário contratado da empresa.

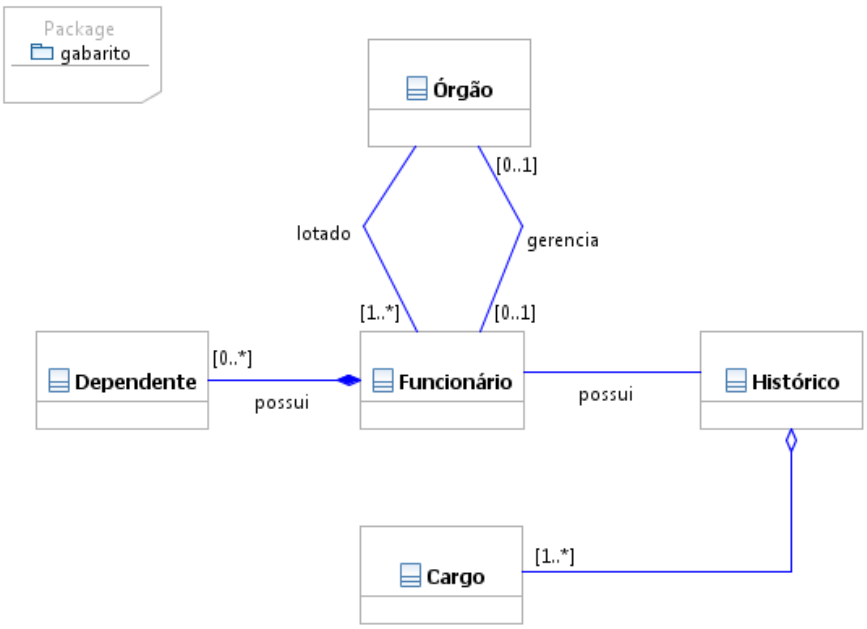


Figura C.1 – Gabarito (domínio empresarial)

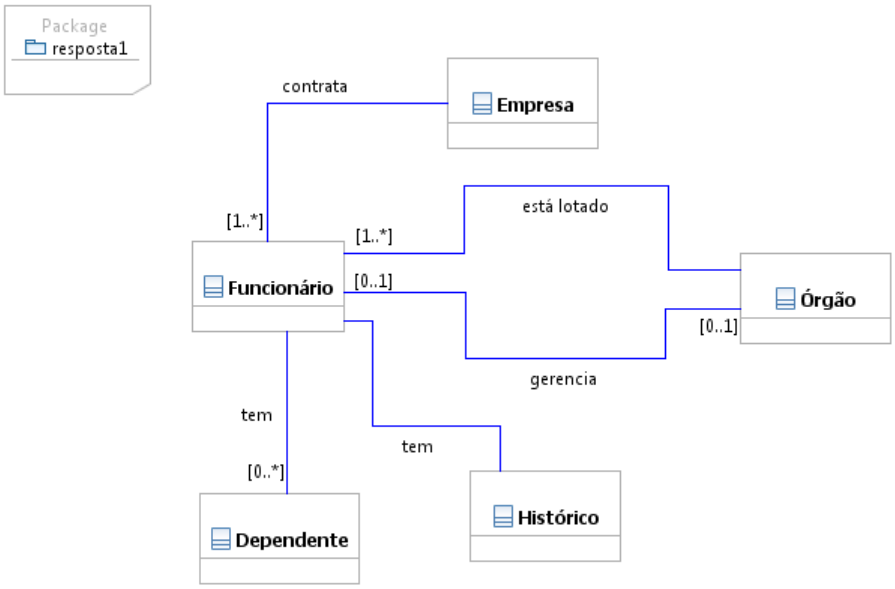


Figura C.2 – Resposta 1 (domínio empresarial)

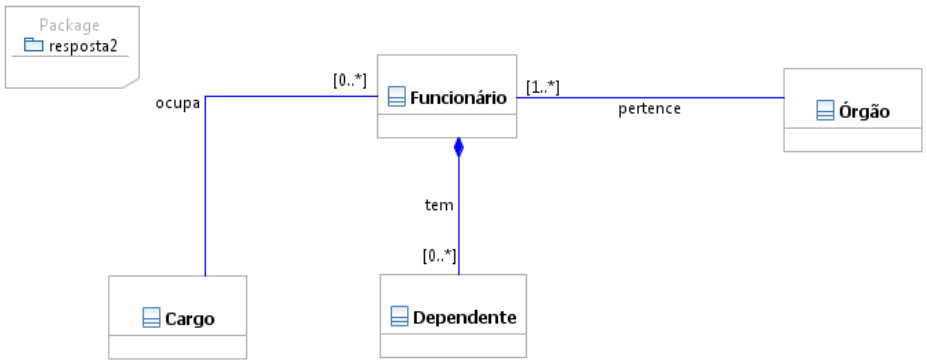


Figura C.3 – Resposta 2 (domínio empresarial)

Exercício **(domínio escolar)**: Um aluno poder estar matriculado em um ou mais cursos. Cada curso tem na sua grade uma série de disciplinas. Disciplinas são oferecidas por departamentos, que estão subordinados às diretorias acadêmicas, e são lecionadas por professores que podem pertencer ao departamento que as oferece ou não. Cada curso está associado a um único departamento. É importante poder recuperar o histórico dos alunos com todas as disciplinas cursadas e respectivas notas, mesmo para os alunos que já saíram da universidade (ex-alunos). "Alunos" são somente aqueles matriculados.

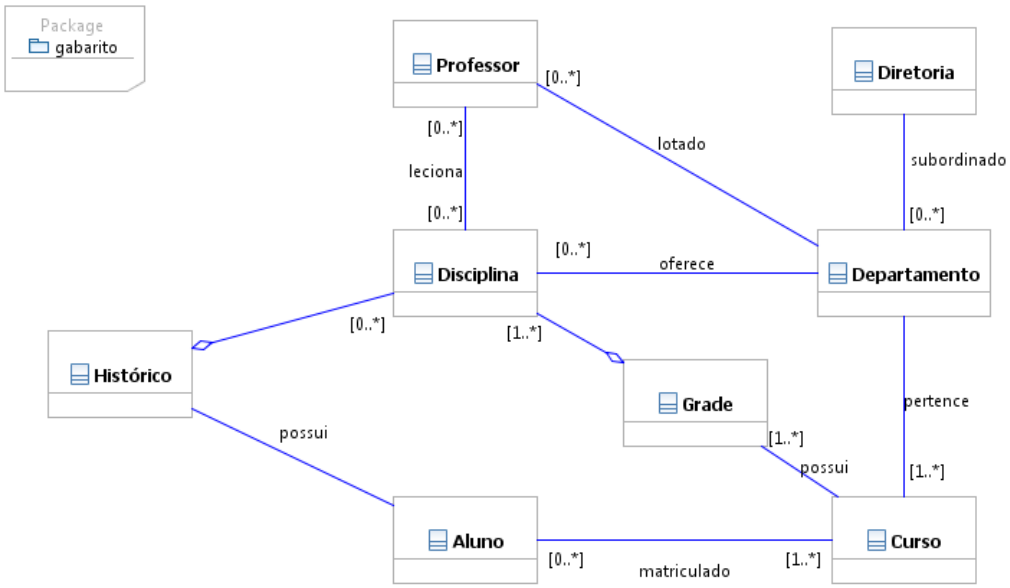


Figura C.4 – Gabarito (domínio escolar)

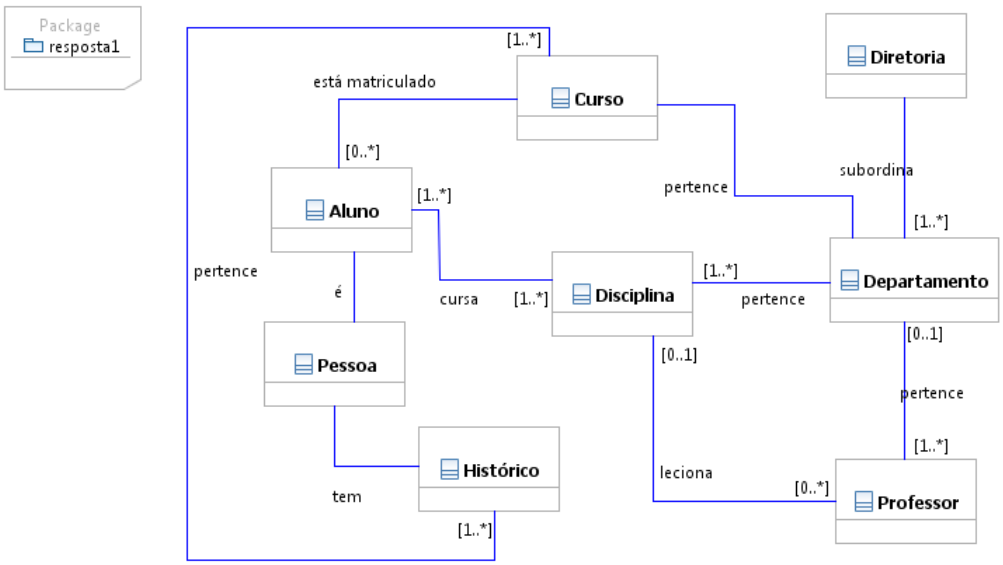


Figura C.5 – Resposta 1 (domínio escolar)

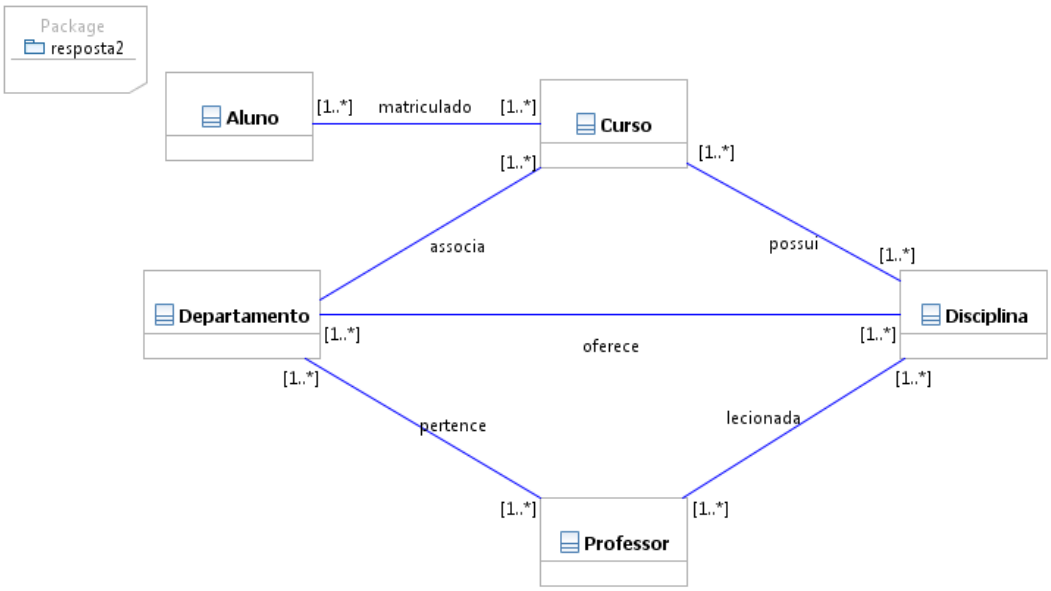


Figura C.6 – Resposta 2 (domínio escolar)

Execução do Estudo – Formulário C	
Nome do participante:	
Divergências detectadas (resposta 1)	
Elemento (ex: *Classe Cliente*, *Associação "compra" entre Cliente e Produto*)	**Descrição da divergência**
	...

C.2. Formulário da Etapa Com o Uso da Abordagem PREViA

Os formulários desta etapa têm como objetivo a avaliação de modelos arquiteturais elaborados por alunos, a partir da identificação de divergências entre um gabarito, utilizando a abordagem PREViA.

Os mesmos formulários (E, F, G e H) foram utilizados para ambos os grupos, variando apenas o Formulário B, em termos do enunciado do exercício e dos modelos utilizados, conforme pode ser visto no formulário em questão (embora os exercícios estejam apresentados sequencialmente, cada formulário continha apenas um exercício, relacionado a apenas um domínio).

Como o conteúdo do formulário B (incluindo seus enunciados e gabaritos) são os mesmos descritos na Seção C.1, serão exibidos apenas os modelos com a aplicação da abordagem PREViA (é importante ressaltar que o grupo que utilizou um domínio sem a PREViA trabalhou com outro domínio com a PREViA. Além disso, os formulário G e H são cópias do formulário C para as divergências entre as respostas 1 e 2 e o gabarito neste cenário, e por isso não serão representados.

Execução do Estudo – Formulário E	
Nome do participante:	

Prezado participante,

Esta é a segunda parte de um estudo de viabilidade da aplicação da abordagem PREViA no contexto de Educação de Engenharia de Software. O objetivo da atividade a ser realizada é a avaliação de modelos arquiteturais elaborados por alunos, a partir da identificação de divergências entre um gabarito. Os modelos são reais e foram elaborados no contexto de uma disciplina de modelagem e projeto de sistemas. **Nesta segunda parte, a representação visual destes modelos é resultante da aplicação da abordagem PREViA**; após cada aplicação, é exibida uma legenda com o significado das representações.

Este formulário está dividido da seguinte forma: esta parte (formulário E) contém o objetivo do estudo e as instruções. A parte F contém alguns dados a serem preenchidos, bem como a descrição do exercício, o gabarito e as respostas a serem corrigidas. A parte G diz respeito às divergências detectadas entre a primeira resposta e o gabarito. Por fim, a parte H diz respeito às divergências detectadas entre a segunda resposta e o gabarito.

IMPORTANTE: Para a correção dos exercícios, sempre que o valor da cardinalidade de uma associação estiver ausente, tal valor deve ser considerado [1..1]. Segue novamente a ordem dos passos para a realização deste exercício:

1. Leia o exercício e o gabarito (o tempo deste passo não deve ser registrado). Não corrija o gabarito, utilize-o para a correção das respostas mesmo que discorde da solução do gabarito.
2. Registrar o horário de início da correção do exercício 1.
3. Comparar a resposta 1 com o gabarito, registrando textualmente as diferenças entre ambos na parte G do formulário.
4. Dar uma nota para o exercício 1, tomando como base o gabarito. Note que, para pontuar o exercício, pode-se considerar que o mesmo pode ser uma alternativa de projeto, sem que divergências entre este e o gabarito signifiquem erro.
5. Registrar o horário de finalização da correção do exercício 1.
6. Registrar o horário de início da correção do exercício 2.
7. Comparar a resposta 2 com o gabarito, registrando textualmente as diferenças entre ambos na parte H do formulário.
8. Dar uma nota para o exercício 2, tomando como base o gabarito. Note que, para pontuar o exercício, pode-se considerar que o mesmo pode ser uma alternativa de projeto, sem que divergências entre este e o gabarito signifiquem erro.
9. Registrar o horário de finalização da correção do exercício 2.
10. Informar o tempo gasto com cada exercício.
11. Enviar este formulário para marceloschots@gmail.com e para claudiasusie@gmail.com

Desde já, agradecemos a sua disponibilidade.

Marcelo Schots de Oliveira
Cláudia Susie Camargo Rodrigues
Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

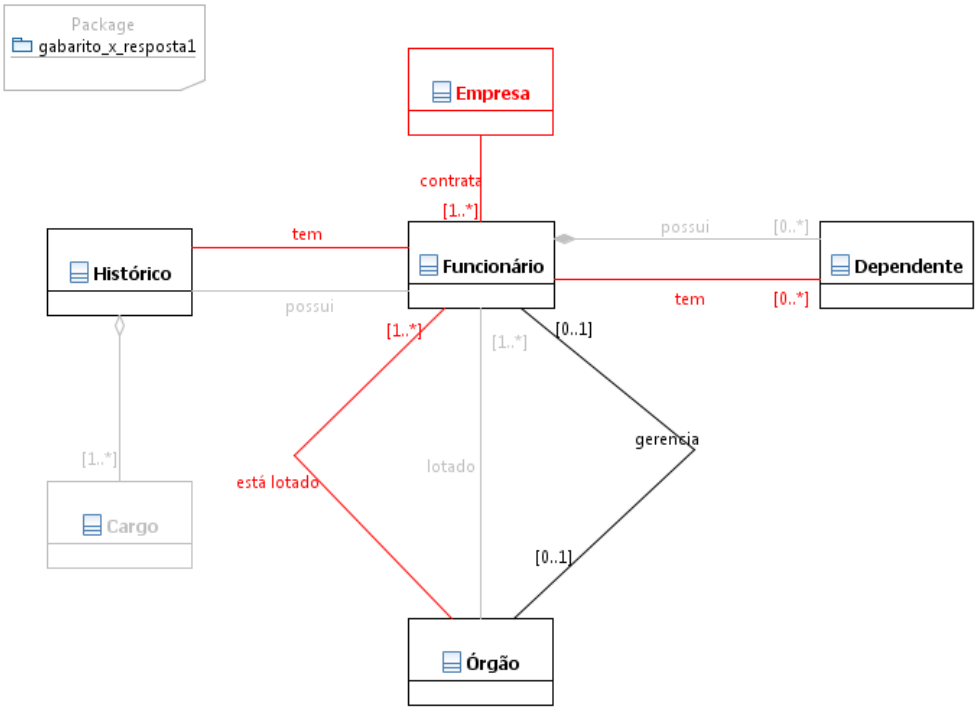


Figura C.7 – Resposta 1, com PREViA (domínio empresarial)

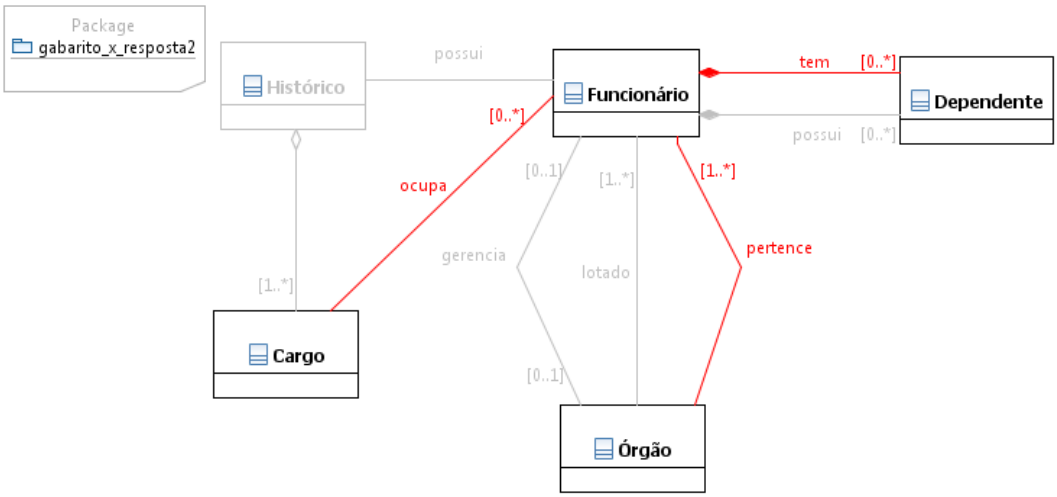


Figura C.8 – Resposta 2, com PREViA (domínio empresarial)

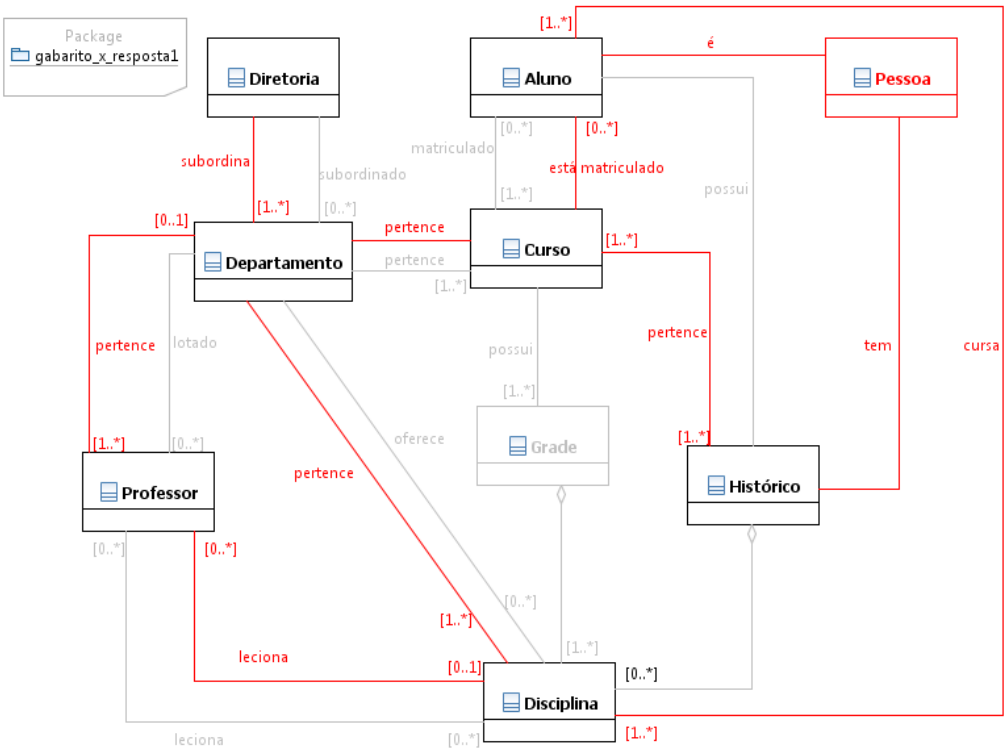


Figura C.9 – Resposta 1, com PREViA (domínio escolar)

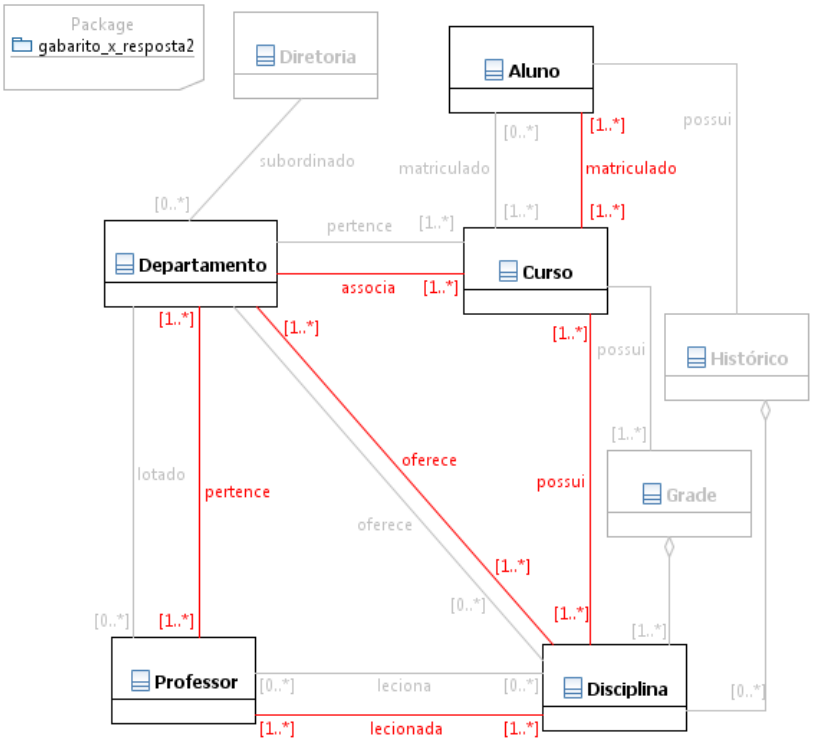


Figura C.10 – Resposta 2, com PREViA (domínio escolar)

Tabela C.1 – Legenda utilizada para os exercícios

Legenda	
Cor	**Significado**
Vermelha	O elemento consta na resposta do aluno, mas **não** consta no gabarito.
Cinza	O elemento consta no gabarito, mas **não** consta na resposta do aluno.
Preta	O elemento consta tanto no gabarito quanto na resposta do aluno.

C.3. Questionário *Follow-Up*

O questionário *follow-up* foi entregue na última parte do estudo, e tem como objetivo coletar as percepções e considerações sobre a execução da atividade de análise e correção das divergências nos modelos arquiteturais com e sem a aplicação da abordagem PREViA.

Execução do Estudo – Formulário I	
Nome do participante:	

Prezado participante,

Esta é a terceira parte de um estudo de viabilidade da aplicação da abordagem PREViA no contexto de Educação de Engenharia de Software. O objetivo da atividade a ser realizada é a avaliação da abordagem aplicada no estudo a partir do preenchimento das questões listadas abaixo.

Este formulário está dividido em 8 questões. Após respondê-las envie este formulário para marceloschots@gmail.com e para claudiasusie@gmail.com

Desde já, agradecemos a sua disponibilidade.

1. Comparando a facilidade na correção das duas respostas da primeira etapa (ou seja, sem a utilizar a abordagem PREViA),
   ( ) a primeira resposta foi mais fácil de corrigir
   ( ) a segunda resposta foi mais fácil de corrigir
   ( ) ambas as respostas foram fáceis de corrigir
   ( ) ambas as respostas foram difíceis de corrigir

2. Comparando a facilidade na correção das duas respostas da segunda etapa (ou seja, utilizando a abordagem PREViA),
   ( ) a primeira resposta foi mais fácil de corrigir
   ( ) a segunda resposta foi mais fácil de corrigir
   ( ) ambas as respostas foram fáceis de corrigir
   ( ) ambas as respostas foram difíceis de corrigir

3. Em relação à facilidade de correção, as marcações apontadas pela abordagem PREViA:
   ( ) dificultaram muito a correção
   ( ) dificultaram a correção
   ( ) foram indiferentes
   ( ) facilitaram a correção
   ( ) facilitaram muito a correção

4. Você sentiu alguma dificuldade na interpretação das informações apresentadas pela abordagem PREViA? Se sim, por favor, indique-a.
   ( ) Sim ( ) Não

5. De acordo com sua opinião, liste os aspectos positivos da utilização da abordagem PREViA:

6. De acordo com sua opinião, liste os aspectos negativos da utilização da abordagem PREViA:

7. Você possui alguma sugestão para melhoria da abordagem PREViA? Em caso positivo, por favor, especifique-a.
   ( ) Sim ( ) Não

8. Este espaço é reservado para quaisquer comentários adicionais (dificuldades, críticas e/ou sugestões) a respeito do estudo executado. Contamos com sua contribuição para que o trabalho seja aprimorado.

Obrigado pela sua colaboração!

Marcelo Schots de Oliveira
Cláudia Susie Camargo Rodrigues
Cláudia Maria Lima Werner
Leonardo Gresta Paulino Murta

APÊNDICE D - GRÁFICOS COMPLEMENTARES À ANÁLISE DOS DADOS DA AVALIAÇÃO NO CONTEXTO DE PROJETOS DE SOFTWARE

Este apêndice contém alguns gráficos de remoção de *outliers* e os gráficos resultantes da análise de variância dos dados da avaliação da abordagem PREViA no contexto de projetos de software.

D.1. Precisão – Análise de *Outliers*

Na análise das tarefas de filtragem (foco de aderência), foram identificados 4 *outliers*. Na iteração seguinte, não houve mais *outliers*.

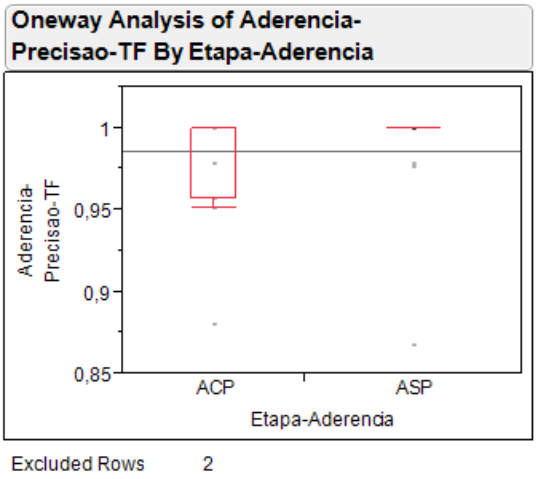


Figura D.1 – *Outliers* identificados durante a análise da precisão (tarefas de filtragem, foco de aderência)

Na análise das tarefas básicas (foco de aderência), foram identificados 4 *outliers* em 2 iterações. Na iteração seguinte, não houve mais *outliers*.

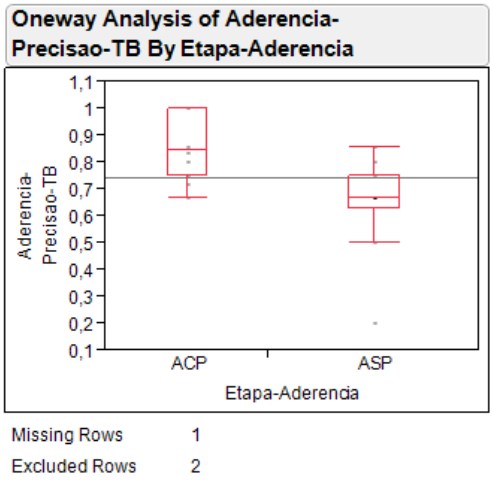


Figura D.2 – *Outliers* identificados durante a análise da precisão (tarefas básicas, foco de aderência)

Na análise das tarefas básicas (foco de evolução), foi identificado 1 *outlier*. Na iteração seguinte, não houve mais *outliers*.

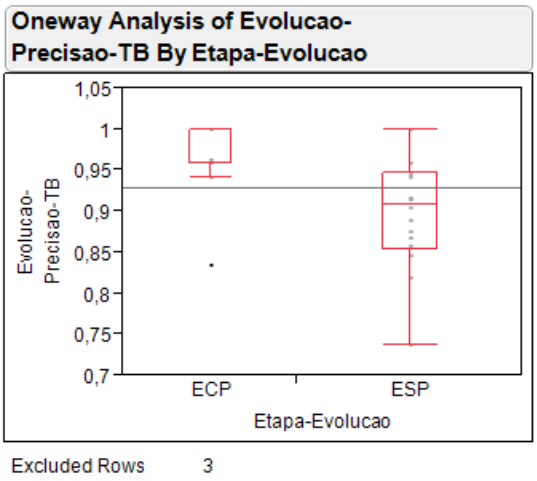


Figura D.3 – *Outliers* identificados durante a análise da precisão (tarefas básicas, foco de evolução)

D.2. Cobertura – Análise de *Outliers*

Na análise das tarefas de filtragem (foco de aderência), foram identificados 7 *outliers* em 2 iterações. Na iteração seguinte, não houve mais *outliers*.

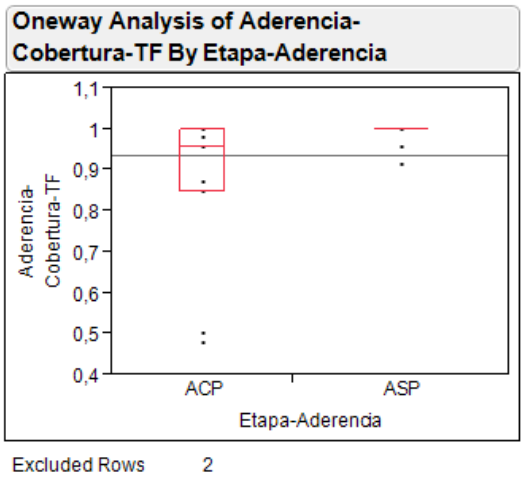


Figura D.4 – *Outliers* identificados durante a análise da cobertura (tarefas de filtragem, foco de aderência)

Na análise das tarefas básicas (foco de evolução), foram identificados 3 *outliers* em 3 iterações. Na iteração seguinte, não houve mais *outliers*.

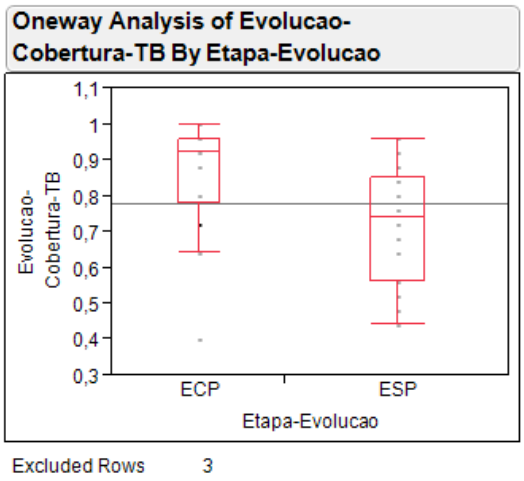


Figura D.5 – *Outliers* identificados durante a análise da cobertura (tarefas básicas, foco de evolução)

D.3. Eficiência – Análise de *Outliers*

Na análise das tarefas de filtragem (foco de aderência), foi identificado 1 *outlier*. Na iteração seguinte, não houve mais *outliers*.

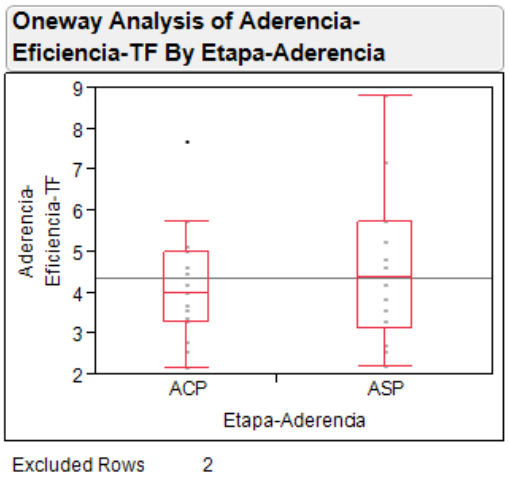


Figura D.6 – *Outliers* identificados durante a análise da eficiência (tarefas de filtragem, foco de aderência)

Na análise das tarefas básicas (foco de aderência), foi identificado 1 *outlier*. Na iteração seguinte, não houve mais *outliers*.

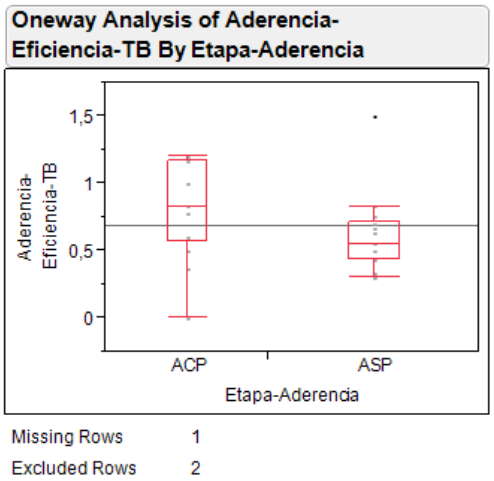


Figura D.7 – *Outliers* identificados durante a análise da eficiência (tarefas básicas, foco de aderência)

Na análise das tarefas de assimilação (foco de evolução), foram identificados 2 *outliers*. Na iteração seguinte, não houve mais *outliers*.

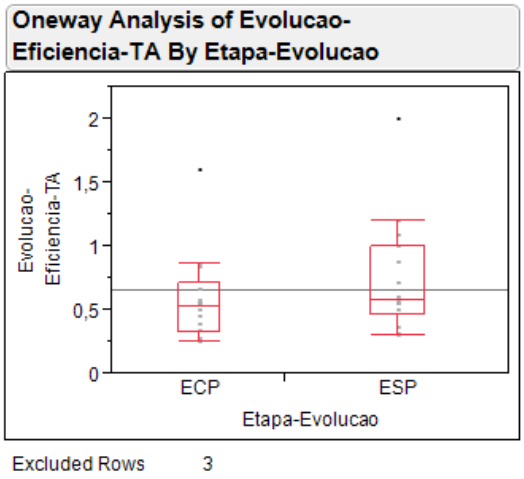


Figura D.8 – *Outliers* identificados durante a análise da eficiência (tarefas de assimilação, foco de evolução)

D.4. Precisão – Análise da Distribuição dos Dados

Ao aplicar o teste de Levene para a métrica precisão (focos de aderência e evolução), observa-se que todas as distribuições são consideradas normais, uma vez que os *p-values* obtidos em todos os casos são maiores do que 0,05. Desta forma, é necessário utilizar um método paramétrico na análise de variância.

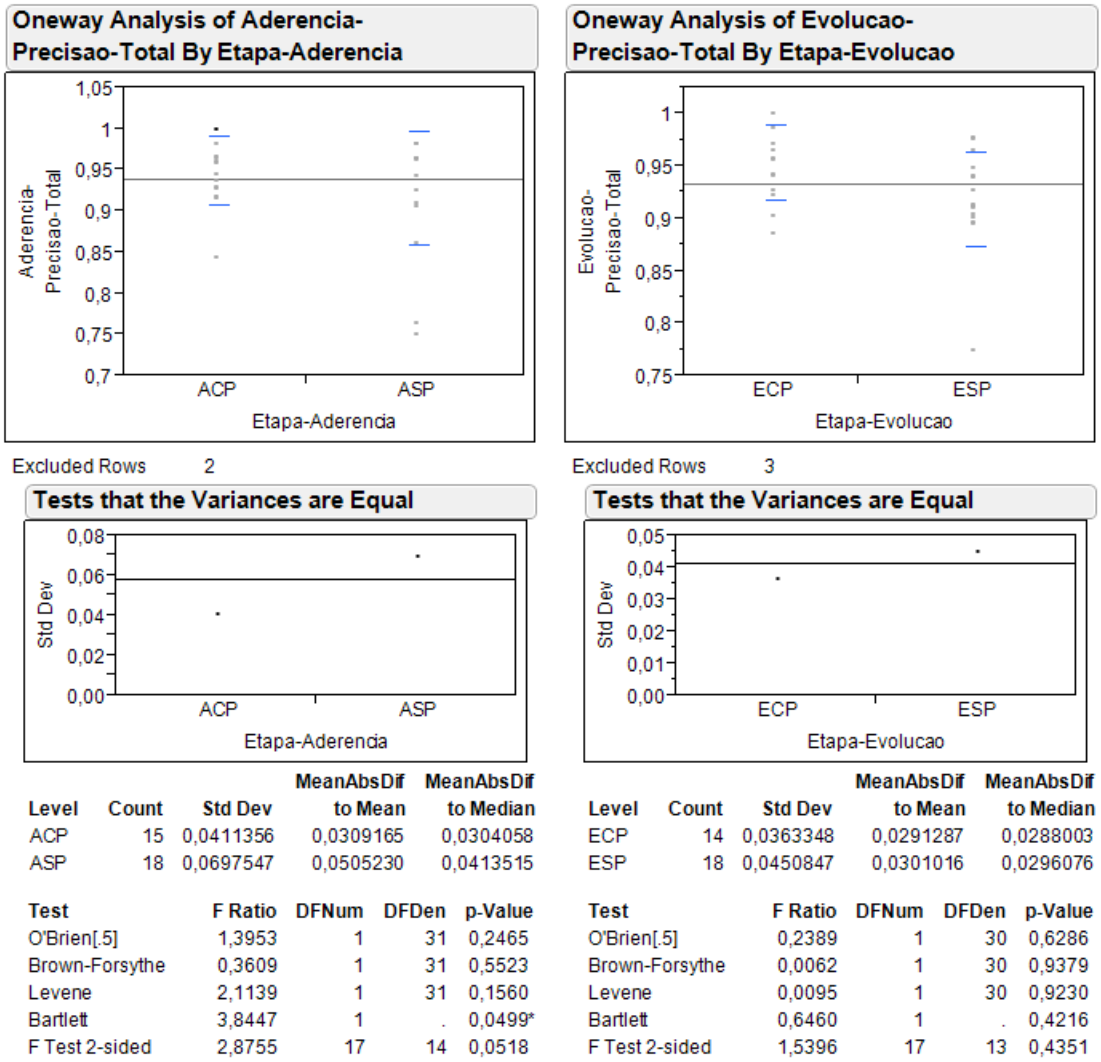


Figura D.9 – Teste de normalidade para a métrica de precisão (todas as tarefas), focos de aderência e evolução

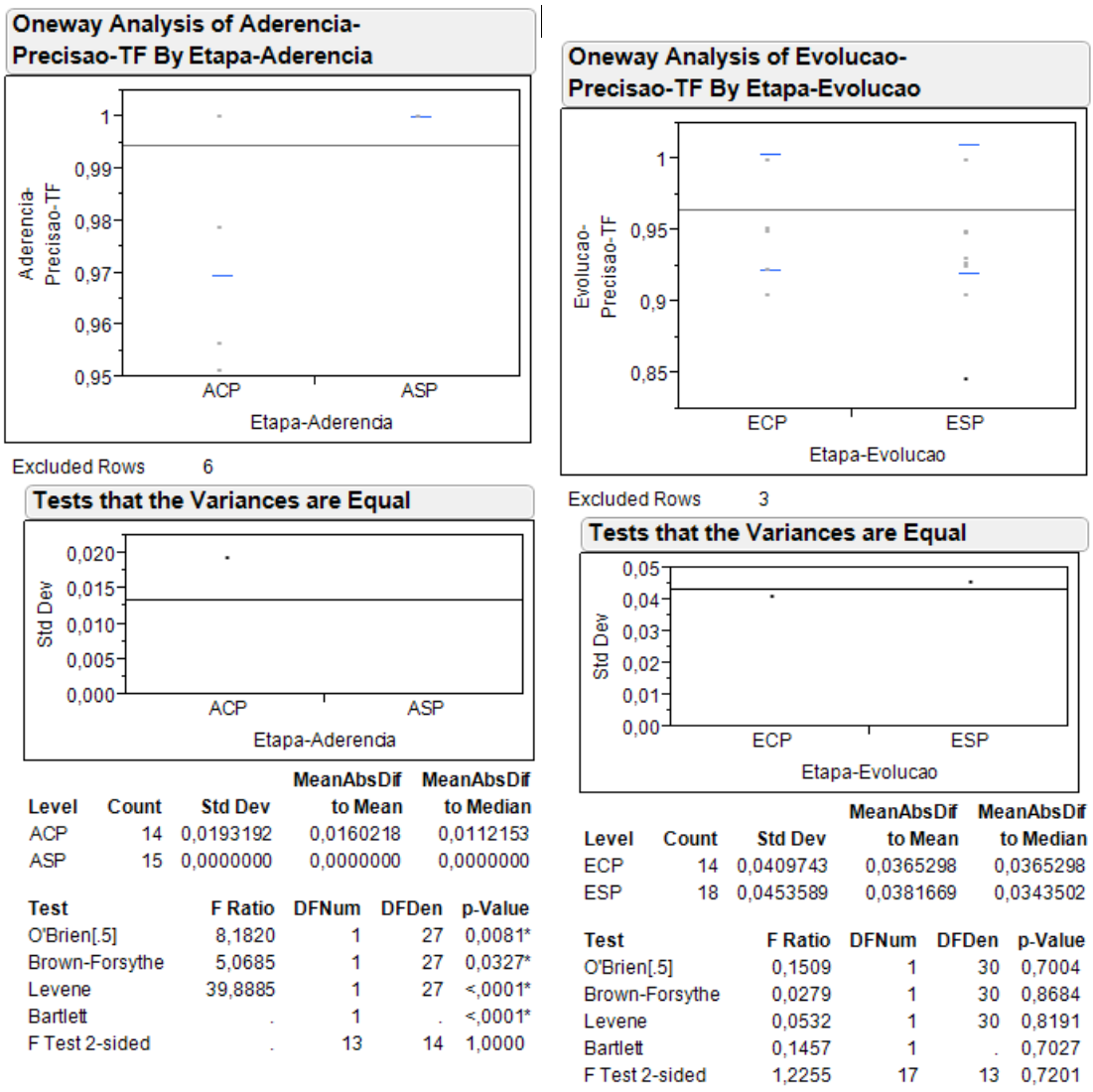


Figura D.10 – Teste de normalidade para a métrica de precisão (tarefas de filtragem), focos de aderência e evolução

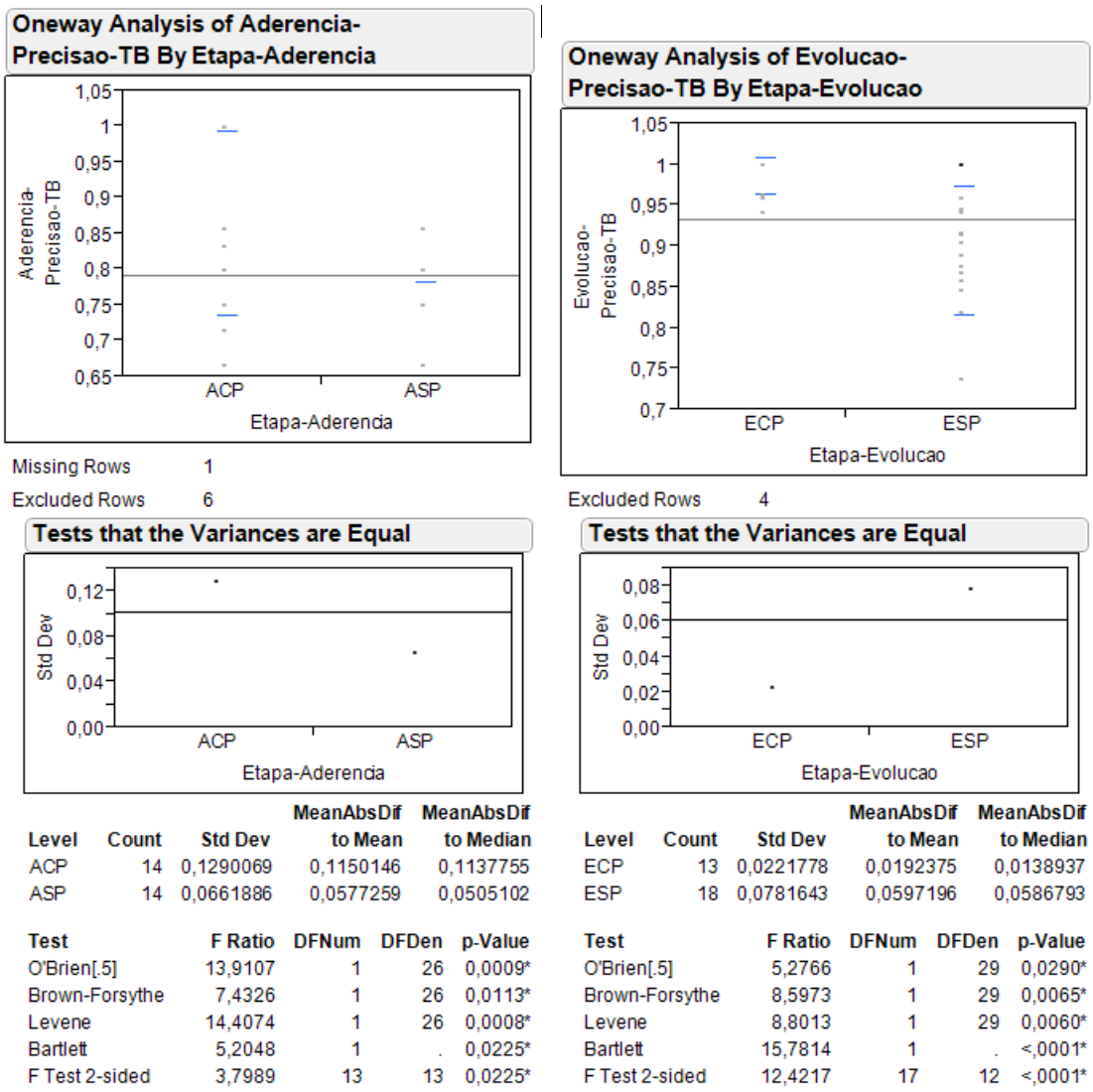


Figura D.11 – Teste de normalidade para a métrica de precisão (tarefas básicas), focos de aderência e evolução

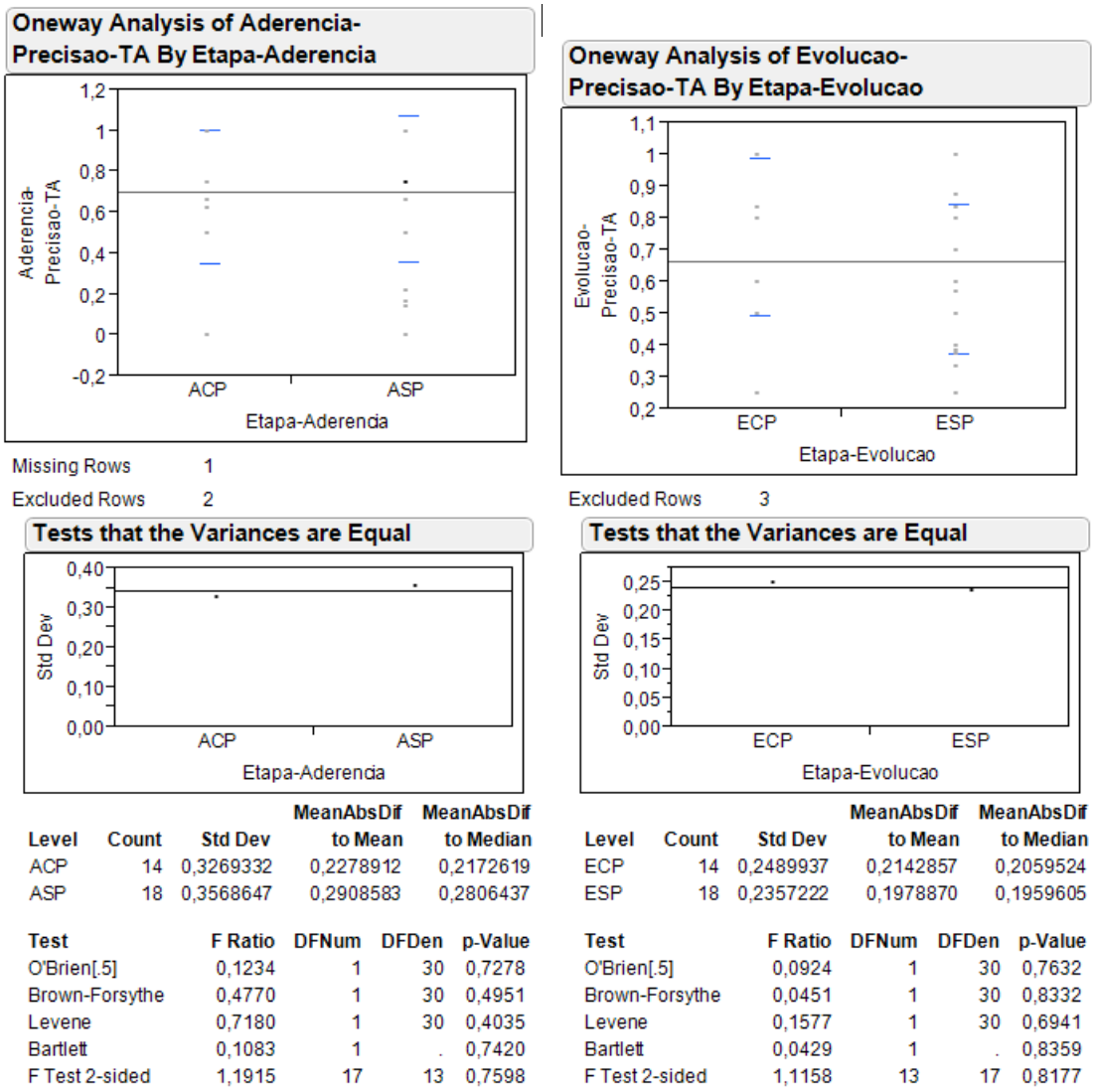
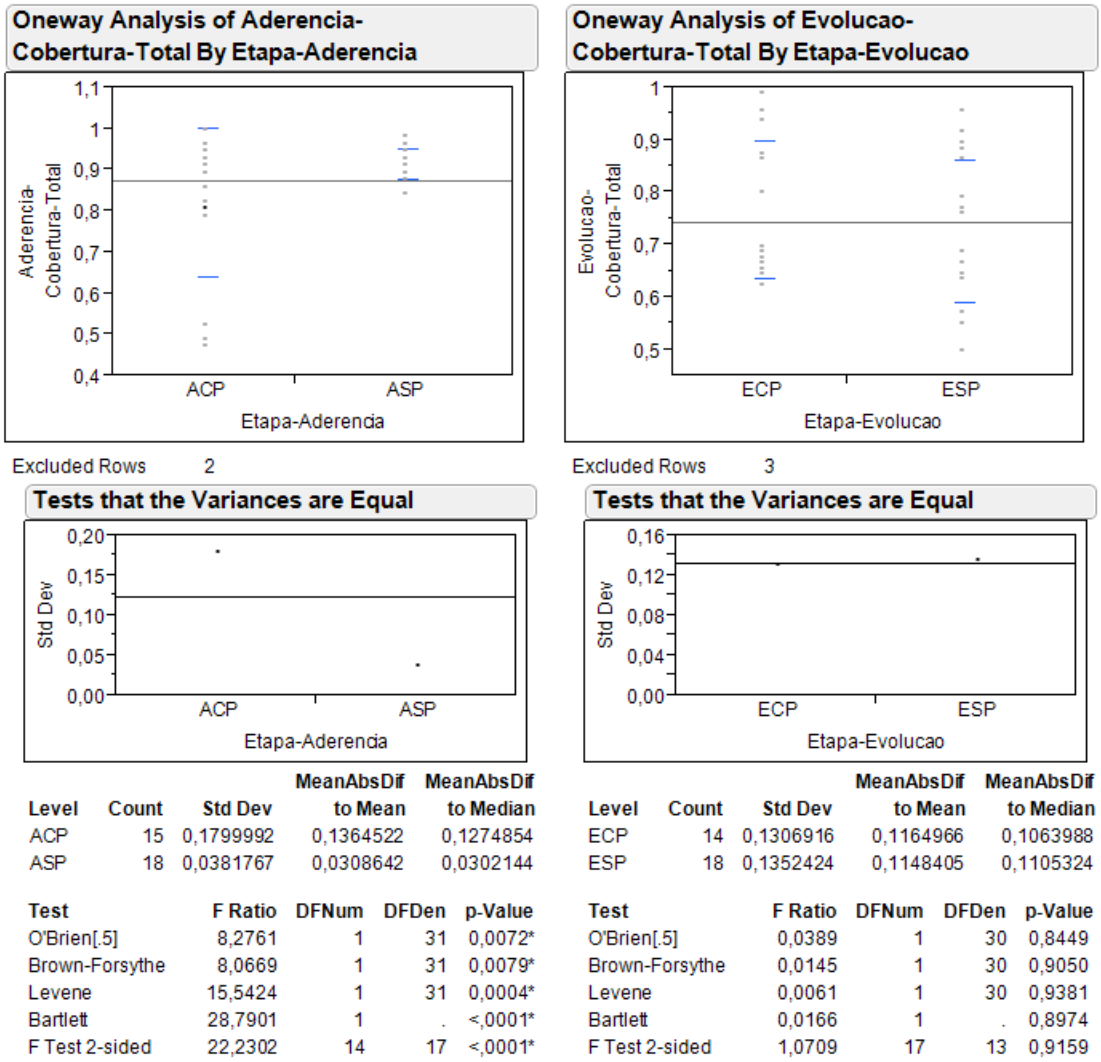


Figura D.12 – Teste de normalidade para a métrica de precisão (tarefas de assimilação), focos de aderência e evolução

D.5. Cobertura – Análise da Distribuição dos Dados

Ao aplicar o teste de Levene para a métrica cobertura (focos de aderência e evolução), observa-se que, para a análise geral e por tarefas de filtragem, a distribuição não é normal, pois os *p-values* obtidos são menores do que 0,05; estes casos requerem o uso de um método não paramétrico. As demais distribuições são consideradas normais, uma vez que os *p-values* obtidos em todos os casos são maiores do que 0,05; para estas, é necessário utilizar um método paramétrico na análise de variância.



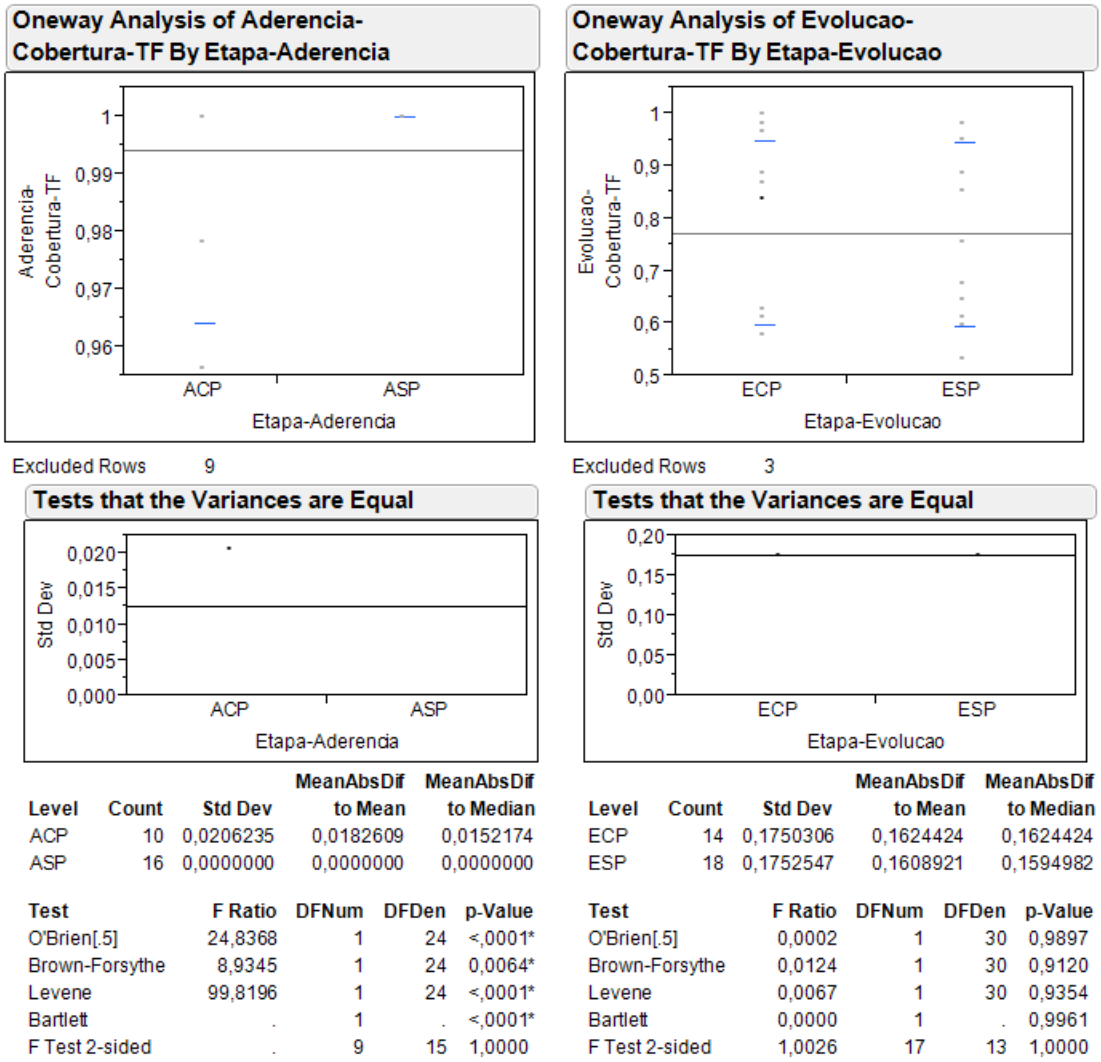
Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
ACP	15	0,1799992	0,1364522	0,1274854
ASP	18	0,0381767	0,0308642	0,0302144

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	8,2761	1	31	0,0072*
Brown-Forsythe	8,0669	1	31	0,0079*
Levene	15,5424	1	31	0,0004*
Bartlett	28,7901	1	.	<,0001*
F Test 2-sided	22,2302	14	17	<,0001*

Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
ECP	14	0,1306916	0,1164966	0,1063988
ESP	18	0,1352424	0,1148405	0,1105324

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	0,0389	1	30	0,8449
Brown-Forsythe	0,0145	1	30	0,9050
Levene	0,0061	1	30	0,9381
Bartlett	0,0166	1	.	0,8974
F Test 2-sided	1,0709	17	13	0,9159

Figura D.13 – Teste de normalidade para a métrica de cobertura (todas as tarefas), focos de aderência e evolução



Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
ACP	10	0,0206235	0,0182609	0,0152174
ASP	16	0,0000000	0,0000000	0,0000000

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	24,8368	1	24	<,0001*
Brown-Forsythe	8,9345	1	24	0,0064*
Levene	99,8196	1	24	<,0001*
Bartlett	.	1	.	<,0001*
F Test 2-sided	.	9	15	1,0000

Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
ECP	14	0,1750306	0,1624424	0,1624424
ESP	18	0,1752547	0,1608921	0,1594982

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	0,0002	1	30	0,9897
Brown-Forsythe	0,0124	1	30	0,9120
Levene	0,0067	1	30	0,9354
Bartlett	0,0000	1	.	0,9961
F Test 2-sided	1,0026	17	13	1,0000

Figura D.14 – Teste de normalidade para a métrica de cobertura (tarefas de filtragem), focos de aderência e evolução

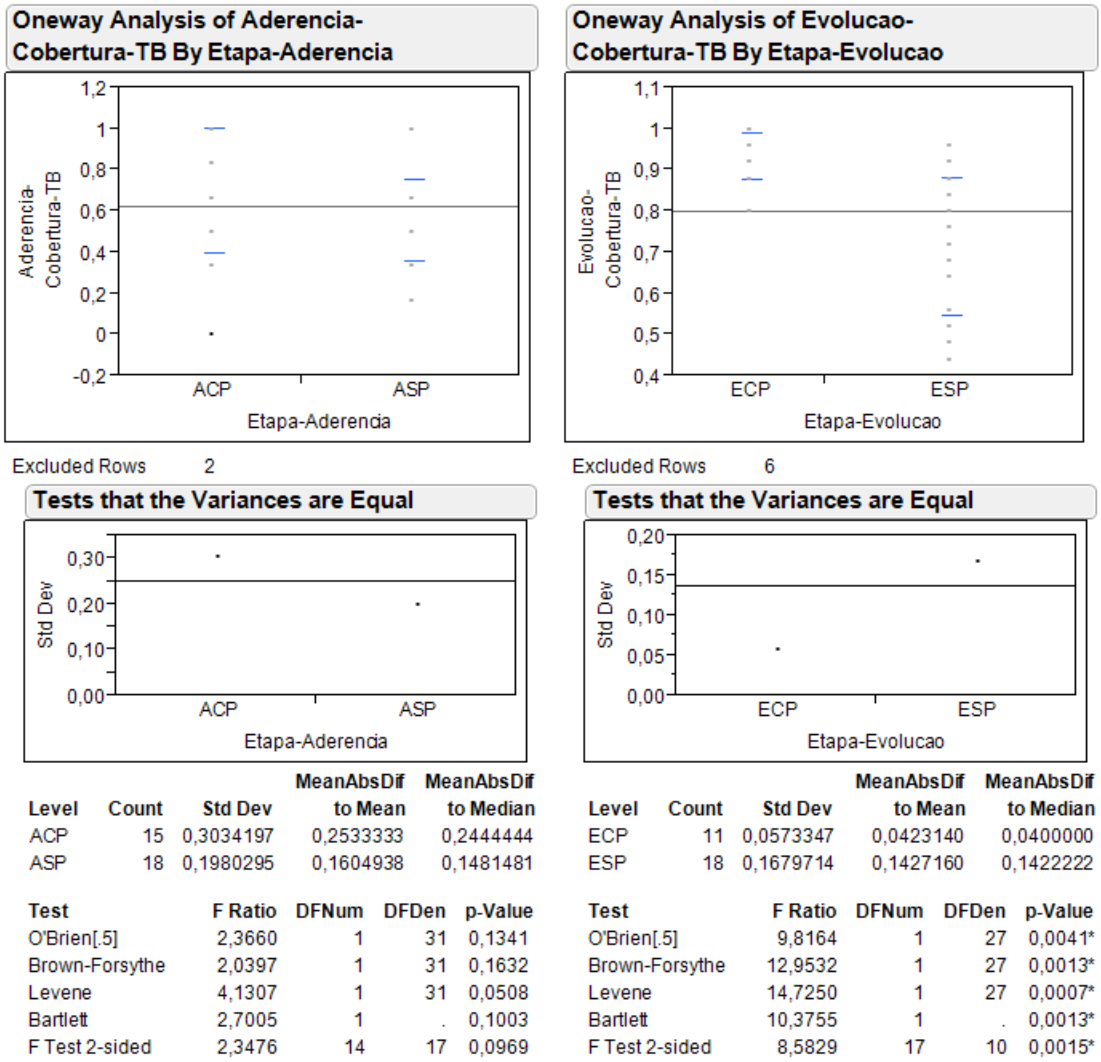


Figura D.15 – Teste de normalidade para a métrica de cobertura (tarefas básicas), focos de aderência e evolução

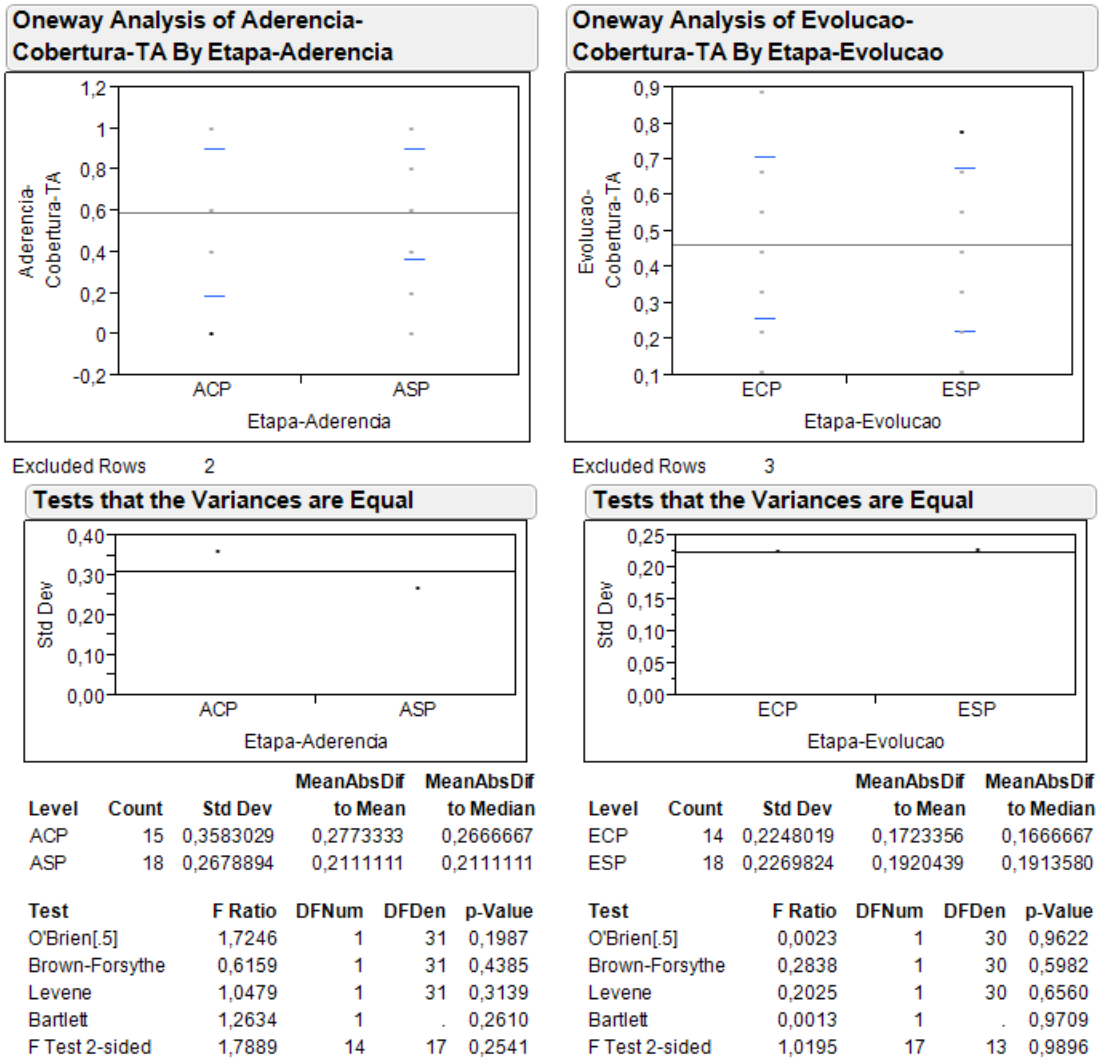


Figura D.16 – Teste de normalidade para a métrica de cobertura (tarefas de assimilação), focos de aderência e evolução

D.6. Eficiência – Análise da Distribuição dos Dados

Ao aplicar o teste de Levene para a métrica eficiência (focos de aderência e evolução), observa-se que todas as distribuições são consideradas normais, uma vez que os *p-values* obtidos em todos os casos são maiores do que 0,05. Desta forma, é necessário utilizar um método paramétrico na análise de variância.

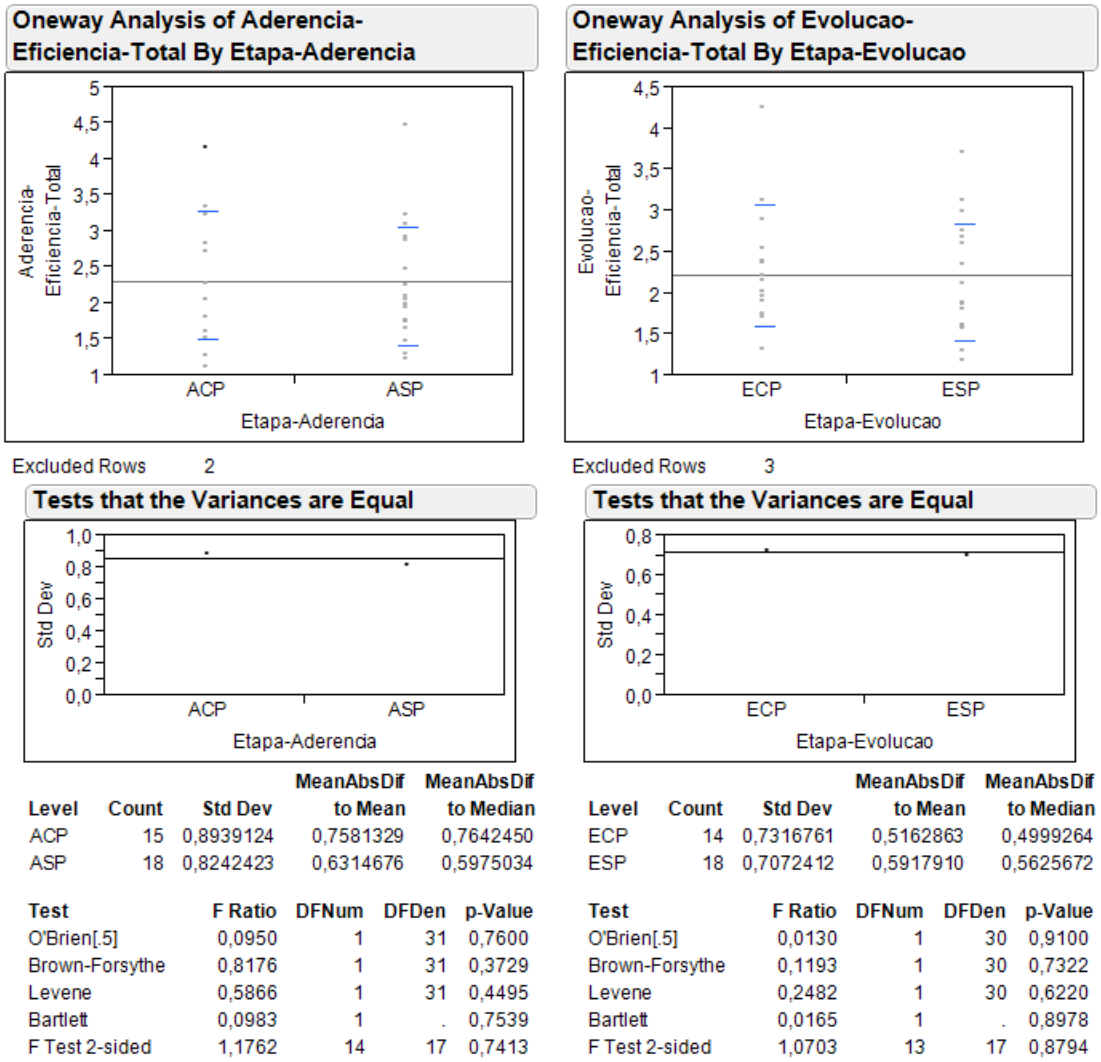


Figura D.17 – Teste de normalidade para a métrica de eficiência (todas as tarefas), focos de aderência e evolução

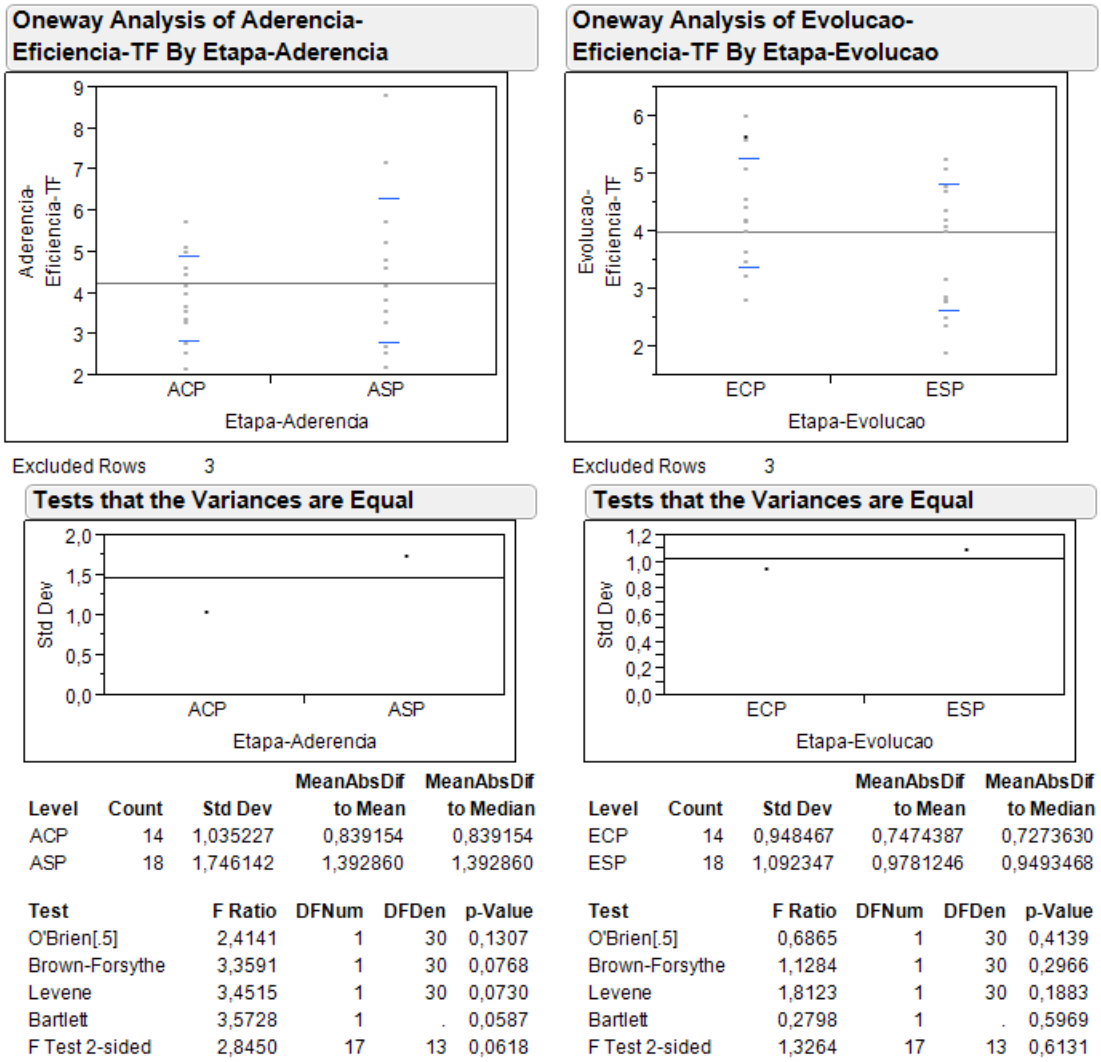


Figura D.18 – Teste de normalidade para a métrica de eficiência (tarefas de filtragem), focos de aderência e evolução

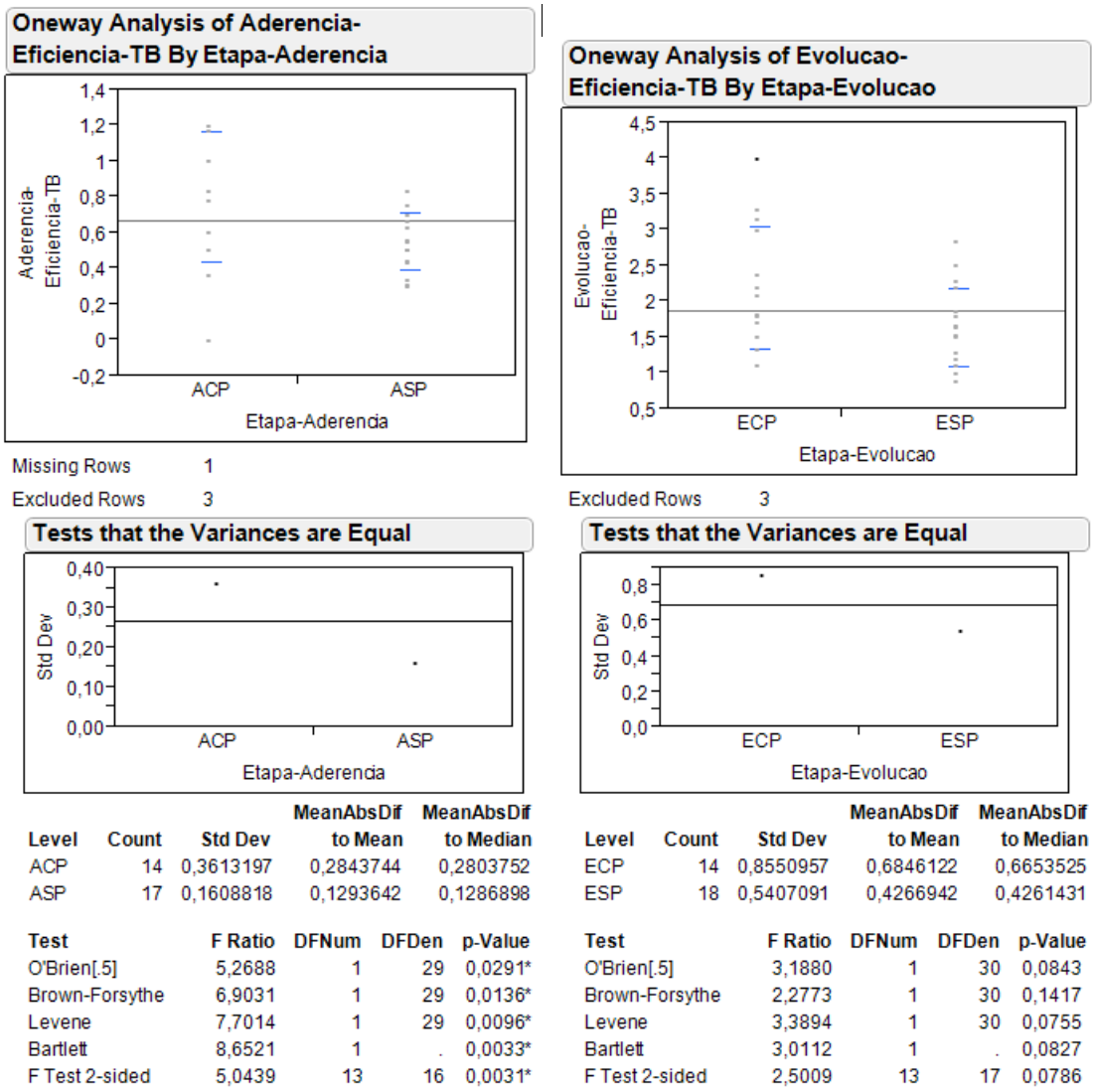
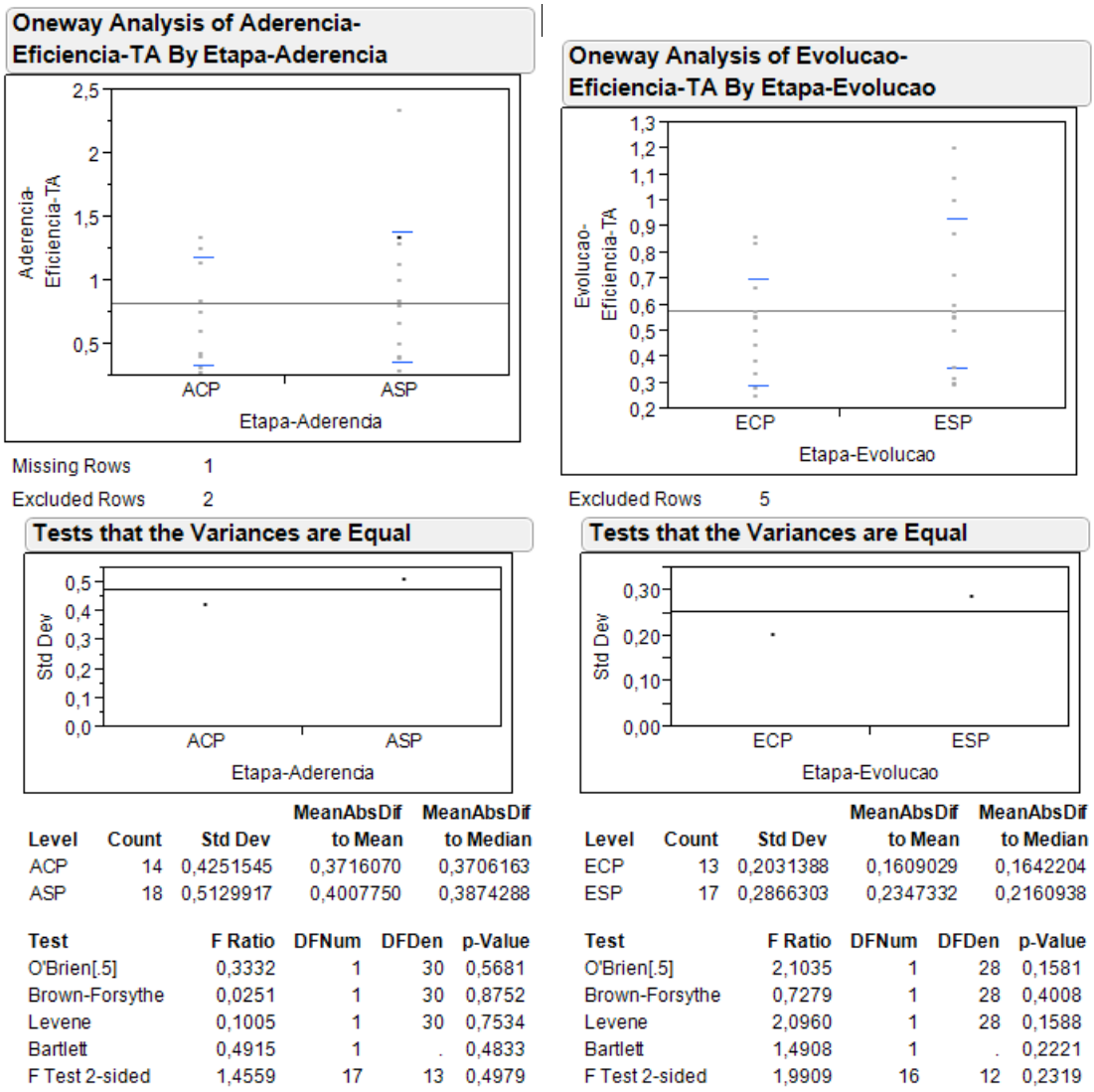


Figura D.19 – Teste de normalidade para a métrica de eficiência (tarefas básicas), focos de aderência e evolução



Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
ACP	14	0,4251545	0,3716070	0,3706163
ASP	18	0,5129917	0,4007750	0,3874288

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	0,3332	1	30	0,5681
Brown-Forsythe	0,0251	1	30	0,8752
Levene	0,1005	1	30	0,7534
Bartlett	0,4915	1	.	0,4833
F Test 2-sided	1,4559	17	13	0,4979

Level	Count	Std Dev	MeanAbsDif to Mean	MeanAbsDif to Median
ECP	13	0,2031388	0,1609029	0,1642204
ESP	17	0,2866303	0,2347332	0,2160938

Test	F Ratio	DFNum	DFDen	p-Value
O'Brien[.5]	2,1035	1	28	0,1581
Brown-Forsythe	0,7279	1	28	0,4008
Levene	2,0960	1	28	0,1588
Bartlett	1,4908	1	.	0,2221
F Test 2-sided	1,9909	16	12	0,2319

Figura D.20 – Teste de normalidade para a métrica de eficiência (tarefas de assimilação), focos de aderência e evolução

D.7. Precisão – Resultados dos Testes de Análise de Variância

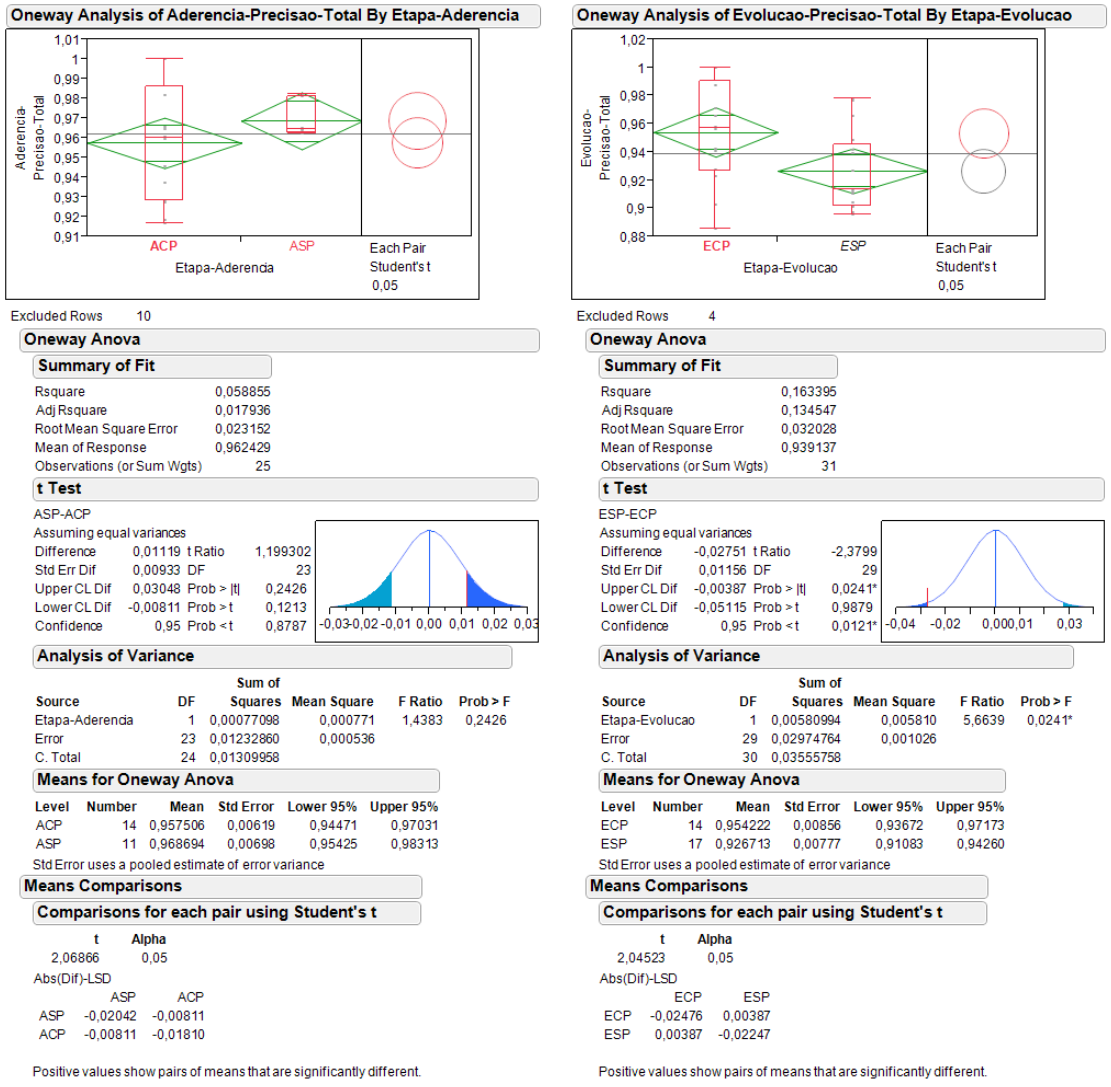


Figura D.21 – Resultados para a análise da precisão (total), etapas de aderência e evolução

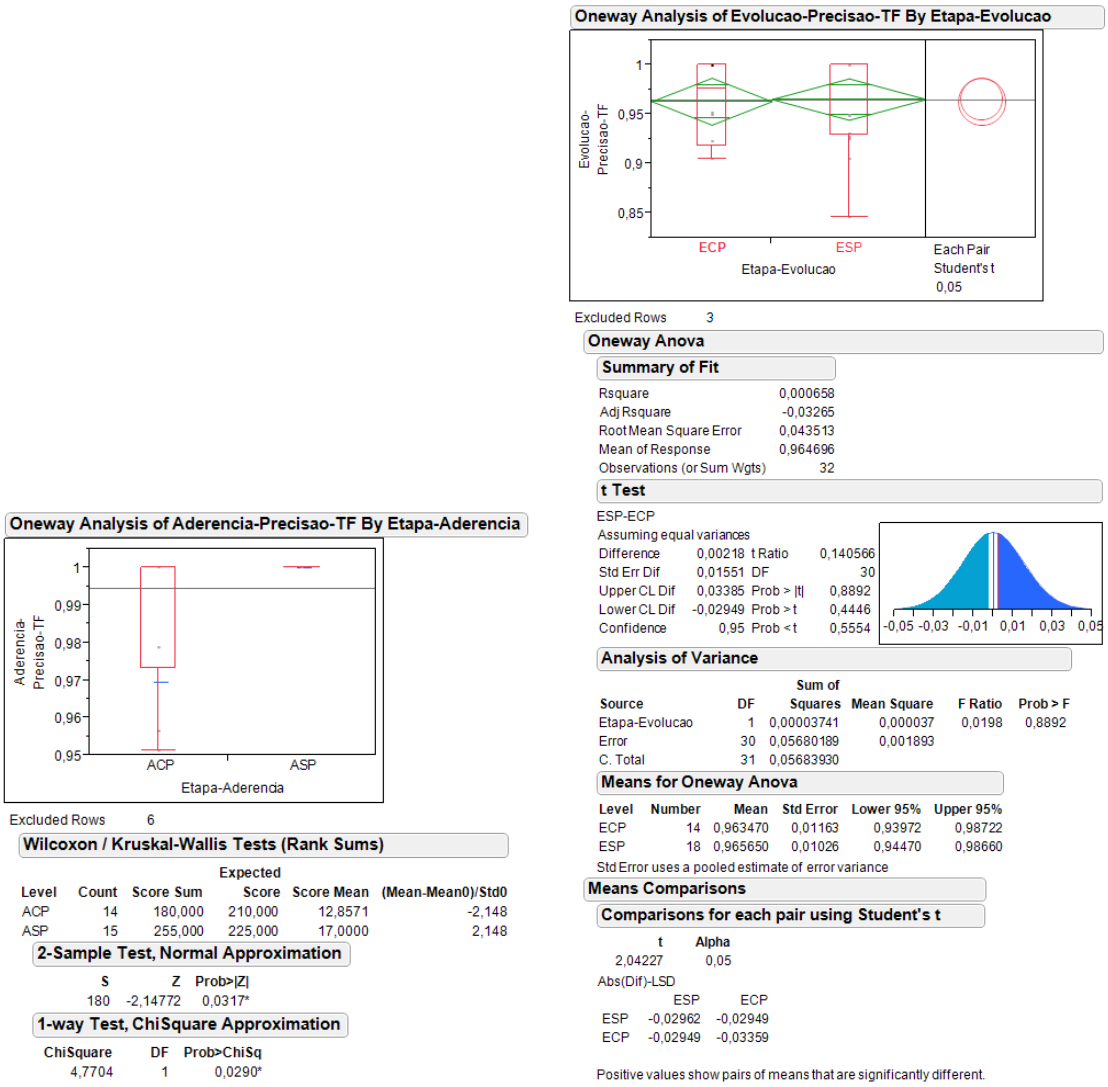


Figura D.22 – Resultados para a análise da precisão (tarefas de filtragem), etapas de aderência e evolução

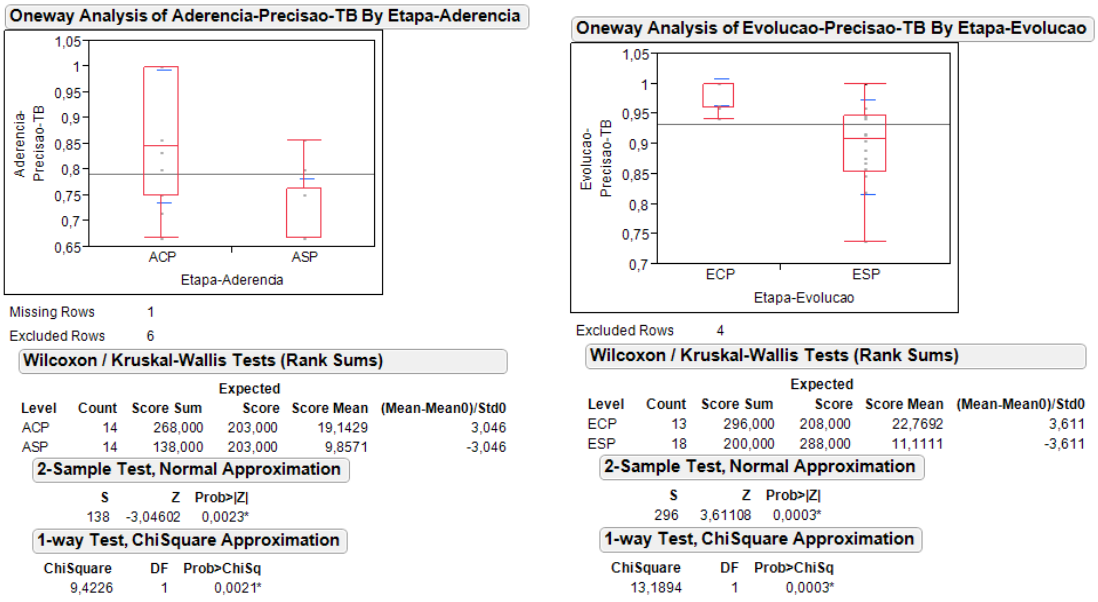


Figura D.23 – Resultados para a análise da precisão (tarefas básicas), etapas de aderência e evolução

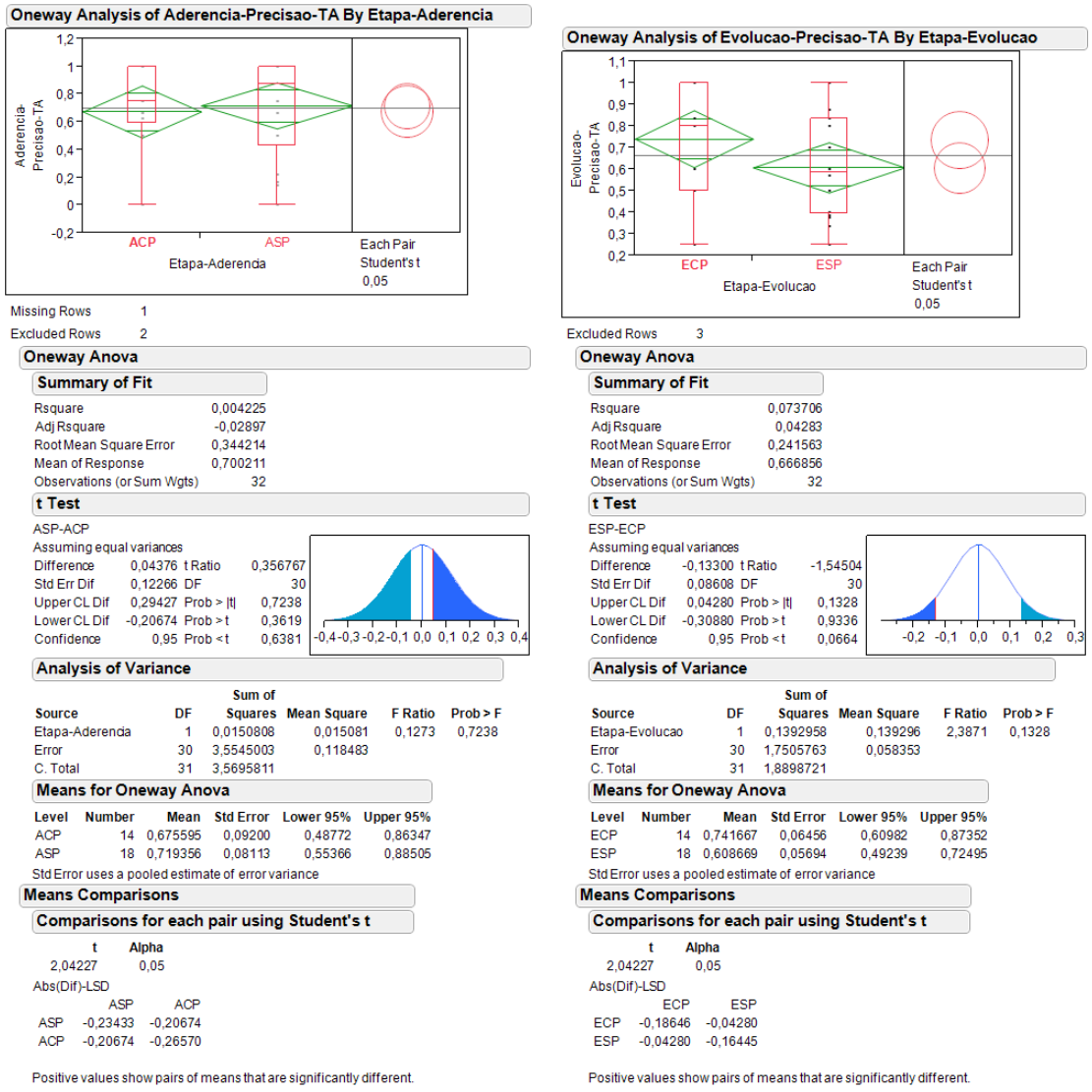


Figura D.24 – Resultados para a análise da precisão (tarefas de assimilação), etapas de aderência e evolução

D.8. Cobertura – Resultados dos Testes de Análise de Variância

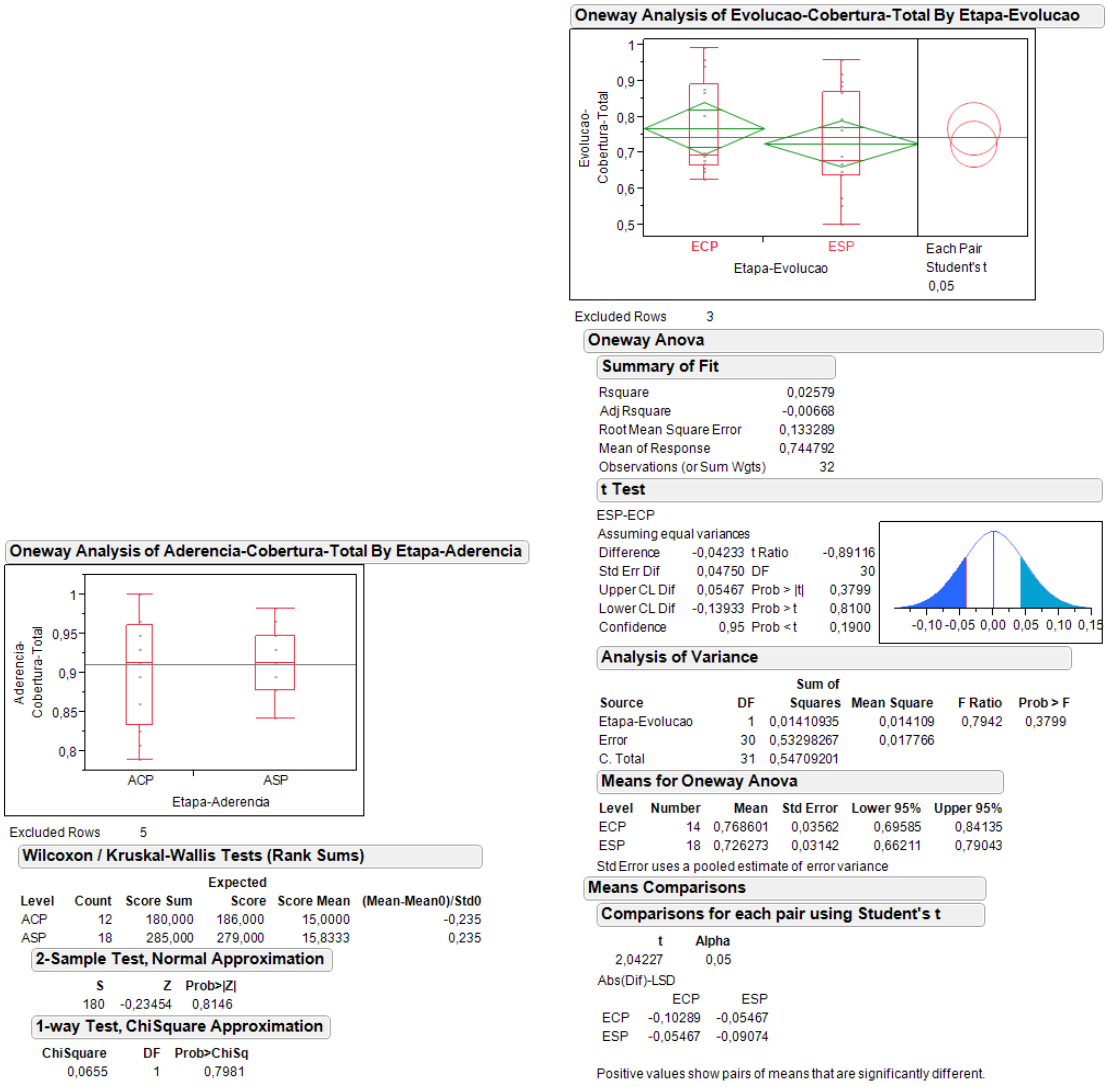


Figura D.25 – Resultados para a análise da cobertura (total), etapas de aderência e evolução

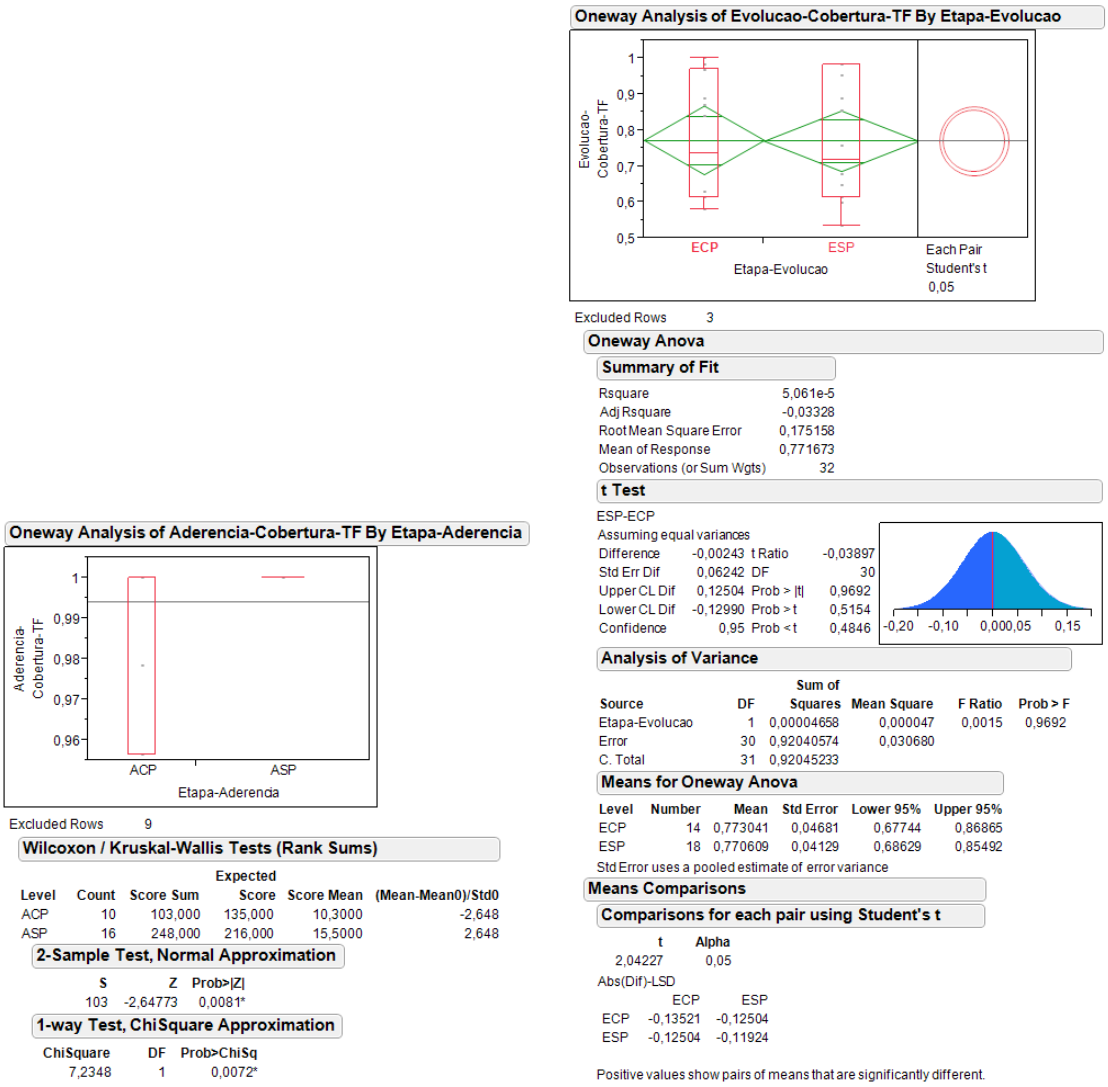


Figura D.26 – Resultados para a análise da cobertura (tarefas de filtragem), etapas de aderência e evolução

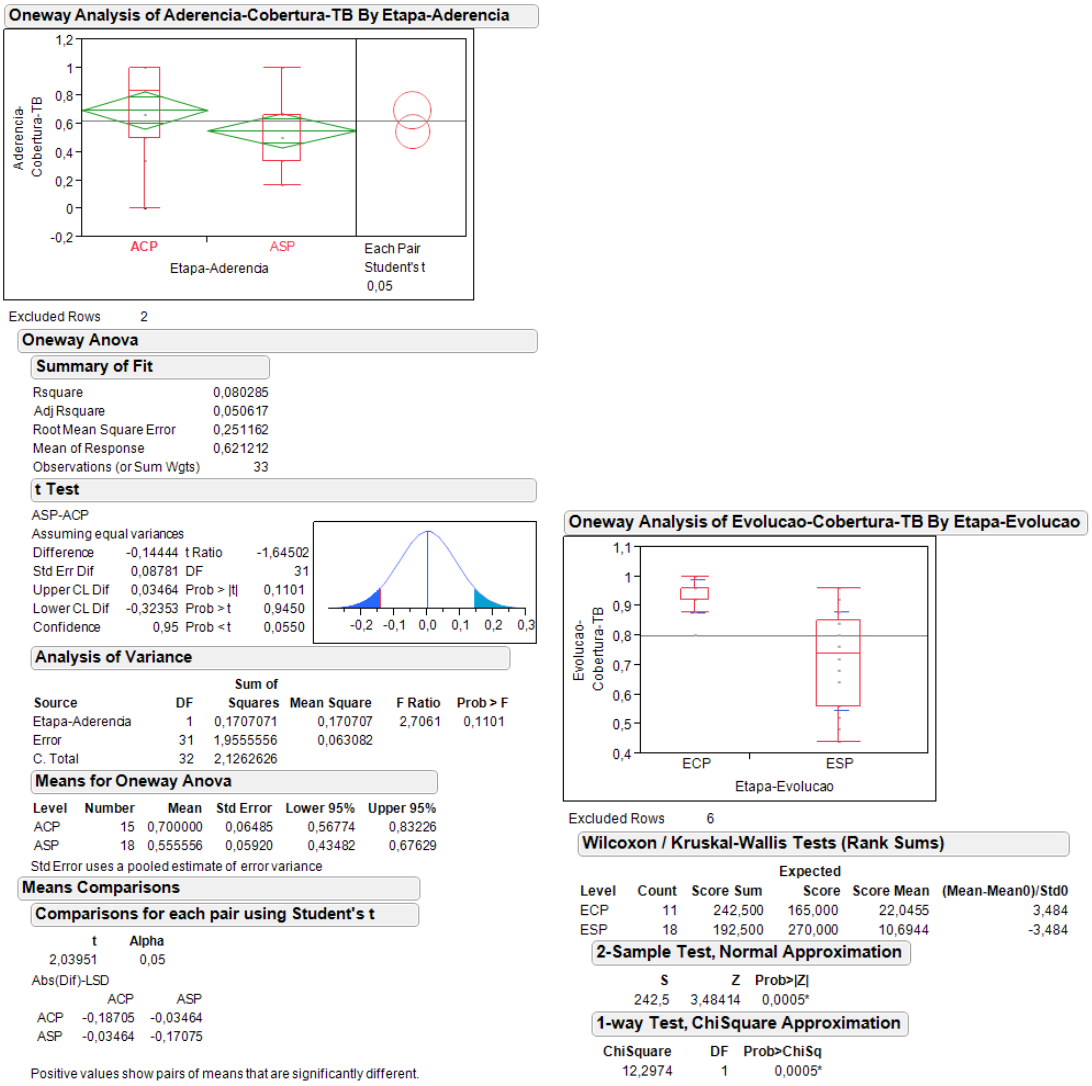


Figura D.27 – Resultados para a análise da cobertura (tarefas básicas), etapas de aderência e evolução

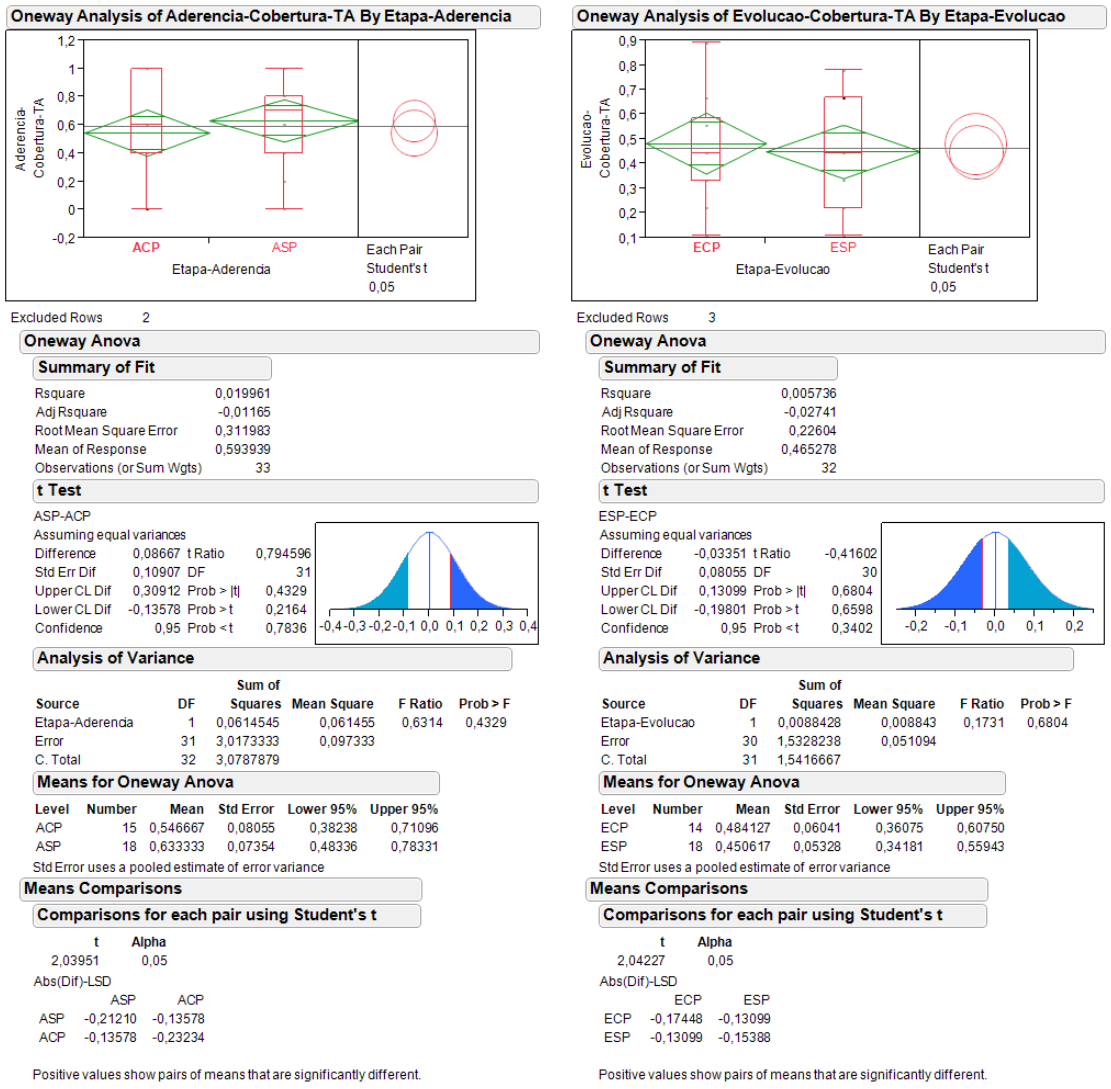


Figura D.28 – Resultados para a análise da cobertura (tarefas de assimilação), etapas de aderência e evolução

D.9. Eficiência – Resultados dos Testes de Análise de Variância

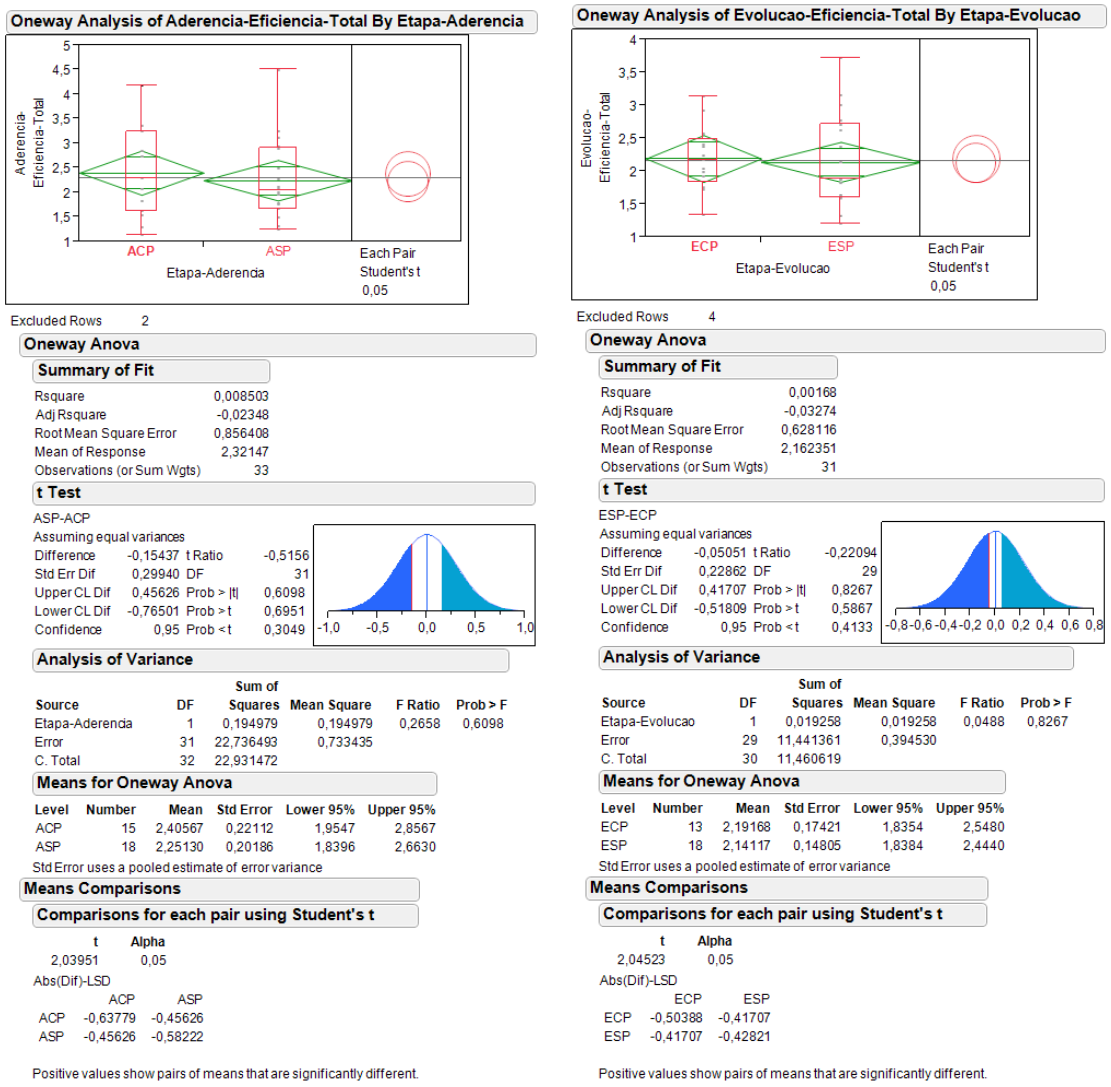


Figura D.29 – Resultados para a análise da eficiência (total), etapas de aderência e evolução

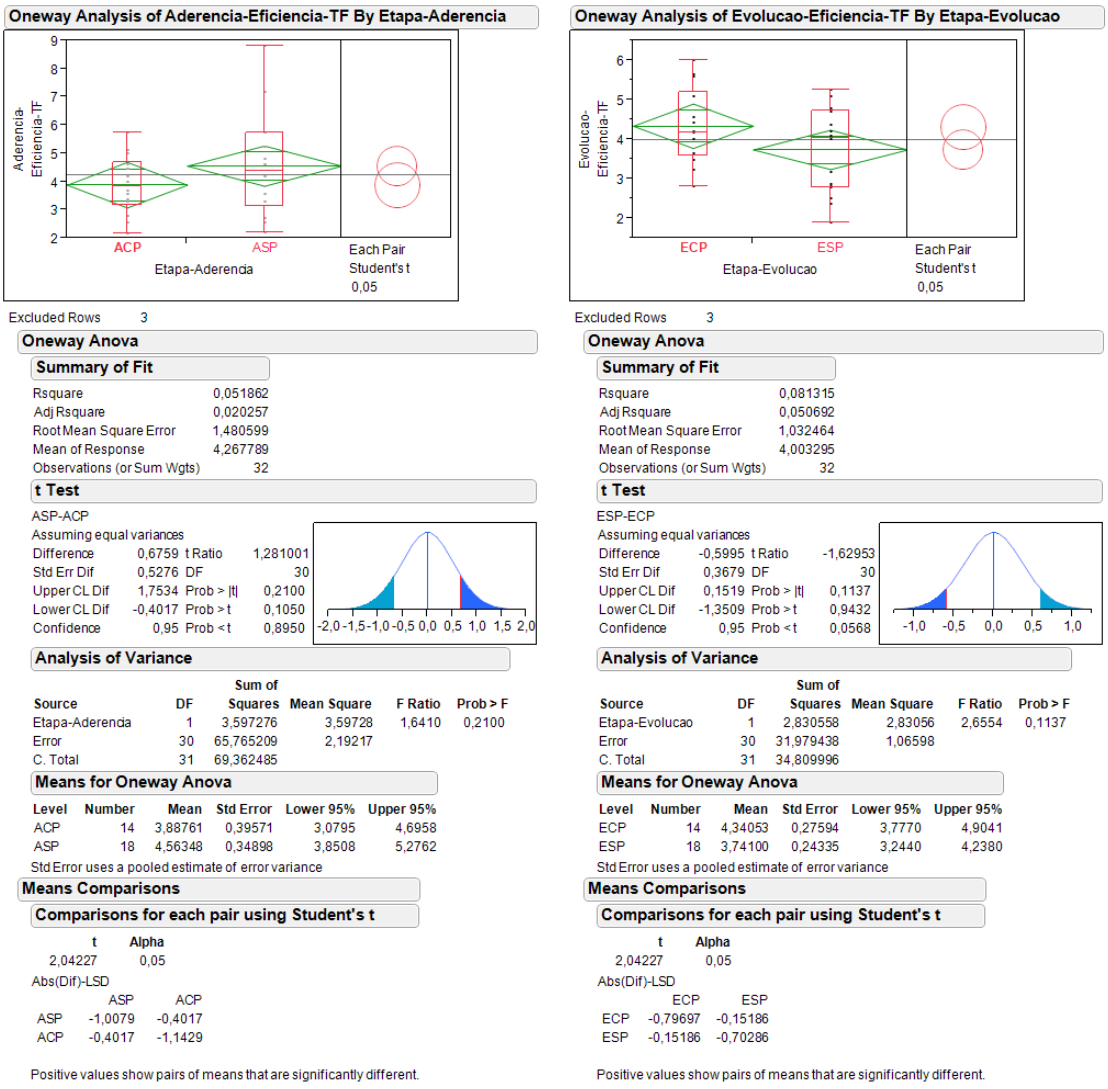


Figura D.30 – Resultados para a análise da eficiência (tarefas de filtragem), etapas de aderência e evolução

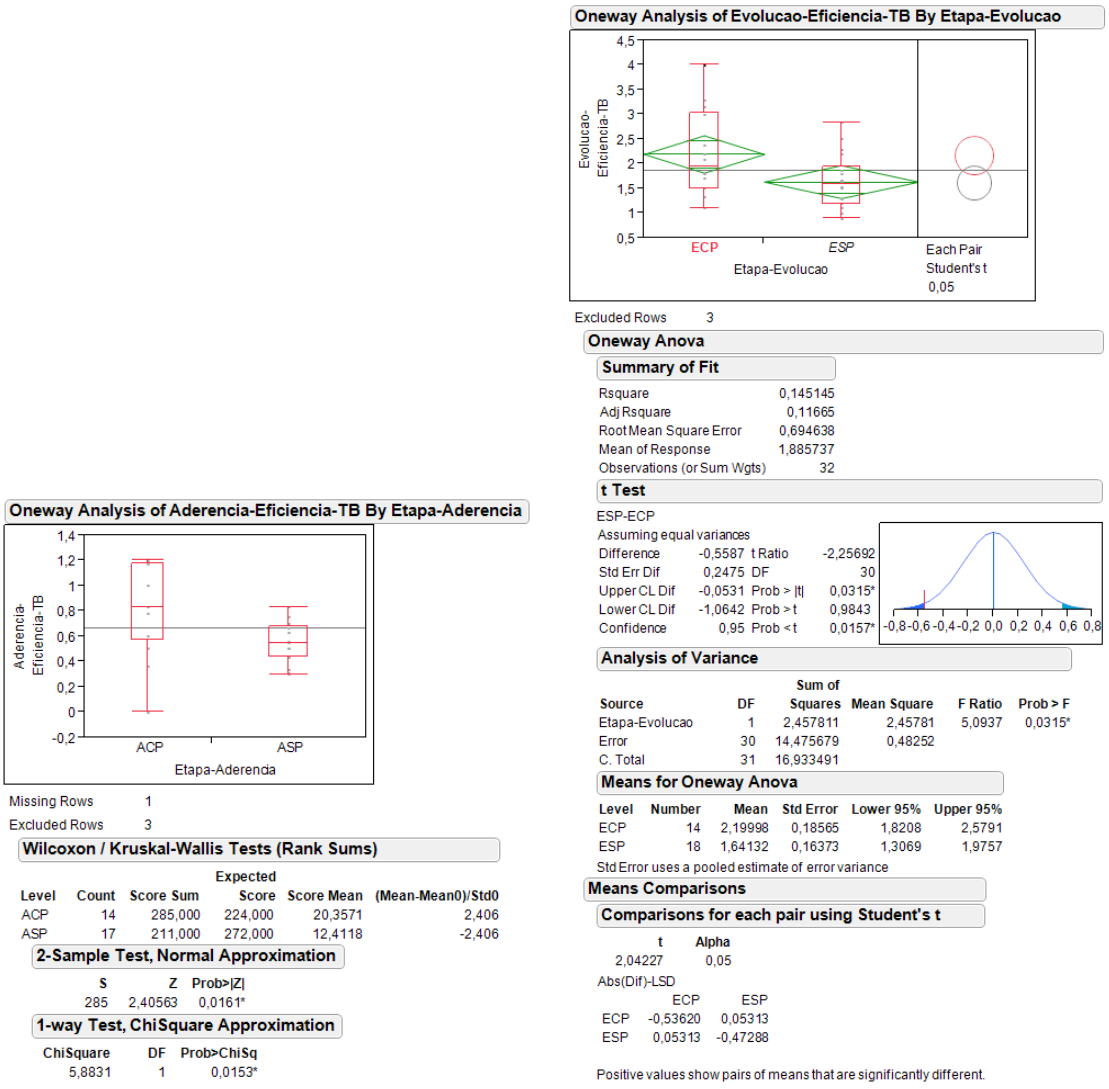


Figura D.31 – Resultados para a análise da eficiência (tarefas básicas), etapas de aderência e evolução

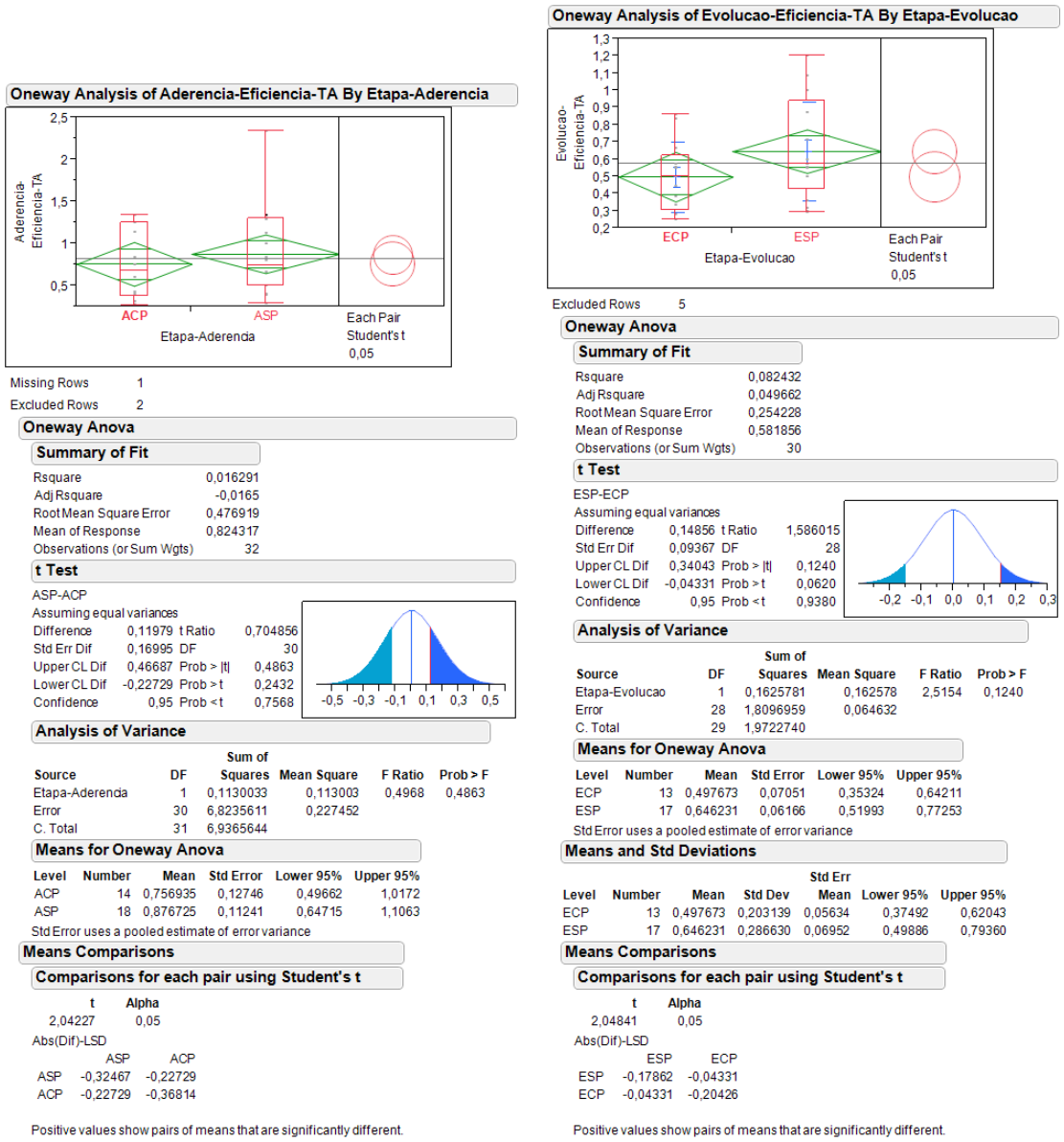


Figura D.32 – Resultados para a análise da eficiência (tarefas de assimilação), etapas de aderência e evolução