UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# O IMPACTO DO USO DE RASTROS DE EXECUÇÃO EM ATIVIDADES DE LOCALIZAÇÃO DE CARACTERÍSTICAS DE SOFTWARE: UM EXPERIMENTO CONTROLADO

RAQUEL FIALHO DE QUEIROZ LAFETÁ

Uberlândia - Minas Gerais
2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RAQUEL FIALHO DE QUEIROZ LAFETÁ

# O IMPACTO DO USO DE RASTROS DE EXECUÇÃO EM ATIVIDADES DE LOCALIZAÇÃO DE CARACTERÍSTICAS DE SOFTWARE: UM EXPERIMENTO CONTROLADO

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador:
Prof. Dr. Marcelo de Almeida Maia

Uberlândia, Minas Gerais
2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**O Impacto do Uso de Rastros de Execução em Atividades de Localização de Características de Software: Um Experimento Controlado**" por **Raquel Fialho de Queiroz Lafetá** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 18 de Fevereiro de 2011

Orientador: _____

Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Stéphane Julia
Universidade Federal de Uberlândia

_____

Prof. Dr. Eduardo Magno Lages Figueiredo
Universidade Federal de Minas Gerais

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Fevereiro de 2011

| | |
|---|---|
| Autor: | **Raquel Fialho de Queiroz Lafetá** |
| Título: | **O Impacto do Uso de Rastros de Execução em Atividades de Localização de Características de Software: Um Experimento Controlado** |
| Faculdade: | **Faculdade de Computação** |
| Grau: | **Mestrado** |

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

# Dedicatória

*Aos meus pais José Antônio de Queiroz Lafetá e Regina Coeli Fialho e meus irmãos Thiago e José Antônio Jr..*

*Ao meu namorado Tiago Alves Leonel.*

# Agradecimentos

Agradeço...

A Deus em primeiro lugar, por tudo que tem proporcionado em minha vida e por ser a maior herança que tenho. Minha fortaleza, nos momentos mais difíceis.

A meus pais José Antônio e Regina e meus irmão Thiago e Junior pelo amor, carinho e apoio nesta caminhada. Por ser a grande base de minha vida, pessoas que sei que sempre poderei contar e que me amam incondicionalmente.

Ao meu namorado Tiago Leonel pela paciência que teve durante esta etapa, pelo carinho e amor que sempre dedicou a mim, me fortalecendo nos momentos difíceis e proporcionando grandes alegrias.

Aos meus amigos e colegas por terem não só me apoiado, mas ajudado efetivamente nos resultados aqui apresentados. Sabia que podia contar com vocês, não imaginei que teria tanto apoio. Obrigada a todos:

Allysson Costa e Silva, Ana Carolina Pereira Rocha, André Luiz Bernardo Ramos, Carlos Antônio Martins, Clenio da Rocha Ferreira, Daniel Oliveira Ferraz, Danilo Cesar de Oliveira, Eberth Duarte Melgaço, Ederson Machado de Lima, Eduardo Braga Rabelo, Felipe Anuar Miziara, Gabriel Coutinho, Guilermo Pelizer Alves Pereira, João Paulo de Araújo Rodrigues, Joaquim Costa Ribeiro Neto, Lucas Pinhal Landim, Luciana Silva, Luiz Fernando Signorelli, Luiz Carlos dos Santos, Marcel Olivier, Marcos José Pires, Mariana Tannús Spirandelli, Matheus Pacheco de Resende, Paula Caroline de Sousa Dias, Rafael Antônio Ribeiro Gomes, Renato Souza, Robson de Carvalho Soares, Rodrigo Reis Pereira, Thalys Alexandre de Santana, Tarcísio Abadio de Magalhães Junior, Thiago Fialho de Queiroz Lafetá, Thiago Rodrigues de Souza e Silva, Thiago Henrique Welte de Almeida.

A meus amigos Luciana do Espírito Santo, Nathália Rodrigues, Mariana Mendonça, Bruna Vallinoto, Andressa Salviano, Denise Nakamura, Ana Sofia Costa, Aparecida Maria Fialho de Souza Almeida, Niara Almeida, Tathiany Boaventura, Cinara Medeiros, Laziane Rennó, Lígia Maria Soares, Pedro Henrique Barbosa, Ricardo Soares de Brito, Ângela Maria Soares, Gelcimar Soares dos Reis, Franz Victor Ramos, Leonardo Marques, Rafael Gouvea, Paulo Marcos Rodrigues, Carlos Henrique, Maria Abadia Arantes, Darlene Dutra, Niselma Borges, Marilene Mazi, Vanessa Ricarty pelo incentivo e carinho que sempre me deram fazendo meus dias mais felizes.

Aos funcionários do laboratório de informática da UFU, Cynthia, Ademir, Marcelo e Geraldo, pela paciência e apoio no uso dos laboratórios.

Agradecimento especial ao meu irmão Thiago Fialho e amigos Felipe Miziara e Daniel Vieira de Souza por terem me ajudado durante a concepção deste trabalho, vocês sempre poderão contar comigo.

E com grande gratidão agradeço ao professor Marcelo Maia pelo companheirismo, paciência, apoio, dedicação e orientação que superou as minhas expectativas, pelo grande profissional que é. Sentirei falta de nossas reuniões e conversas intermináveis que tanto contribuíram para o meu aprendizado.

*"Se um dia tiver que escolher entre o mundo e o amor... Lembre-se. Se escolher o mundo ficará sem o amor, mas se escolher o amor com ele você conquistará o mundo.¨*

*(Albert Einstein)*

# Resumo

Um dos problemas mais frequentes enfrentados em manutenção de software é a localização do código para características específicas, as quais são importantes para a compreensão de requisitos de software. Descobrir onde uma característica está localizada é uma tarefa custosa porque, em geral, estas tendem a estar espalhadas ou entrelaçadas pelo código. Considerando este problema, foi desenvolvida uma abordagem para localização de características utilizando análise dinâmica, que apresenta visões geradas a partir dos rastros de execução. O objetivo desta abordagem é auxiliar na compreensão das características, ao tornar mais rápida a sua localização, com informações que direcionam a compreensão e propiciam maior taxa de acerto nas atividades de manutenção. Para avaliação desta abordagem e verificar se o objetivo é alcançado, foi realizado um estudo controlado com sujeitos humanos, executando atividades reais de manutenção em sistemas de diferentes portes. O estudo pretende contribuir com a avaliação do impacto do uso de informações de rastros de execução em atividades de manutenção de software. Este estudo mostrou os benefícios do uso sistemático de informação de rastros de execução na diminuição do tempo de execução e no aumento da taxa de acerto em atividades de manutenção de software para o problema apresentado. As visões da abordagem proposta foram úteis através da localização das características de interesse e redução do espaço de busca inicial, levando ao direcionamento na busca pelas informações, acarretando em um menor nível de dificuldade percebida pelos participantes que utilizaram a abordagem nas manutenções realizadas nos experimentos. Além do mais, constatou-se que os experimentos onde ocorreram as mais expressivas reduções no espaço de busca inicial e melhor qualidade neste, foram os que apresentaram os melhores desempenhos, em tempo e taxa de acerto. Entretanto, este estudo também revela alguns desafios para aplicação desta abordagem em larga escala, uma vez que as visões podem apresentar elementos falsos positivos e falsos negativos que podem impactar na compreensão de sistemas a partir das informações fornecidas. Para um uso em larga escala seria necessária a introdução de outras visões, mais robustas, para viabilizar análises mais aprofundadas. Finalmente, outra conclusão foi que a abordagem não se faz útil em atividades de compreensão onde o problema do espalhamento e entrelaçamento não ocorre.

**Palavras chave:** localização de características, análise dinâmica, rastros de execução e compreensão de sistemas.

# Abstract

One of the most frequent problems faced by software maintainers is to find the location of the code related with specific features, which are important for understanding the software requirements. To find where a certain feature is located is a costly task because, in general, it tends to be scattered or tangled into the code. Considering this problem, an approach for locating features using dynamic analysis has been developed, which presents views generated from execution traces. The objective of this work is to help understanding the features, enabling faster location of features and providing higher accuracy rate in maintenance activities, achieved the information that leads directly to the understanding. To evaluate this approach and check whether the goal was achieved, a controlled study was performed with the participation of human beings, when they were performing actual maintenance activities on systems of different sizes. This study contributes with an appraisal about the impact of using information from execution traces on software maintenance. This study shows the benefits related with a systematic use of information from execution traces by reducing the execution time and increasing the rate of correction in searches for information during software maintenance. The views were useful for locating the features of interest and reducing the initial search space, providing better precision in the search for information. This feature resulted in a lower level of difficulty perceived by study participants during the proposed maintenance activities. Besides that, this study has indicated that the size and quality of the search spaces (presented in the views) can affect the rate of correction and execution time on maintenance activities using the presented approach. However, this study also reveals that there are some challenges when it comes to implement this approach on a large scale, system since the views may reveal many false positives and false negatives. This situations can impact the understanding of the systems based on the provided information. On the large-scale use, of the approach, it would be interesting to introduce more robust views, enabling more robust analysis. Finally, another conclusion is that this approach is not useful in comprehension activities where the problem of scattering and entanglement does not occur.

**Keywords:** feature location, dynamic analysis, execution traces and understanding systems.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

É reconhecido que a compreensão de programas requer um esforço significativo durante a execução de tarefas de manutenção. Diversas alternativas têm sido propostas para amenizar este esforço. Entretanto, ainda não existe nenhuma alternativa que seja amplamente reconhecida para ser usada em um contexto geral.

Nos anos 70, Lehman estabeleceu leis, ou hipóteses empíricas, que sugerem que qualquer programa que seja usado regularmente deve passar por modificações contínuas para satisfazer seus usuários [Lehman 1980]. De fato, até os dias atuais é amplamente reconhecido que a modificação é uma necessidade inerente à grande parte do programa existente. Os desenvolvedores responsáveis por modificações sejam elas, corretivas, adaptativas, evolutivas ou preventivas tem a necessidade de compreender o código, o qual possivelmente não foi escrito por eles. Assim, antes de executar uma modificação, desenvolvedores devem explorar o código fonte do sistema, achar e entender um subconjunto de estruturas e o comportamento do programa que é relevante à modificação proposta. Quanto maior é o sistema em avaliação e maiores são as pressões normais do ambiente de desenvolvimento, tanto maior será o nível de dificuldade colocado para os desenvolvedores [Robillard e Murphy 2007].

Um problema frequentemente enfrentado durante a manutenção de sistemas é a localização do código que implementa características de interesse, as quais são importantes para o entendimento de requisitos de programa [Wilde et al. 2003]. Características encapsulam o domínio do conhecimento, sendo fontes valiosas de informações para tarefas de engenharia reversa. Existem diferentes definições para características [1] na comunidade de engenharia de software, e para o trabalho aqui apresentado adotou-se a seguinte definição: características são entidades definidas como incrementos funcionais de programas [Batory 2006]. Por exemplo, um editor de textos oferece características como edição, carregamento e atualização de arquivos, correção ortográfica, etc.

[1] do inglês, *Feature Location*

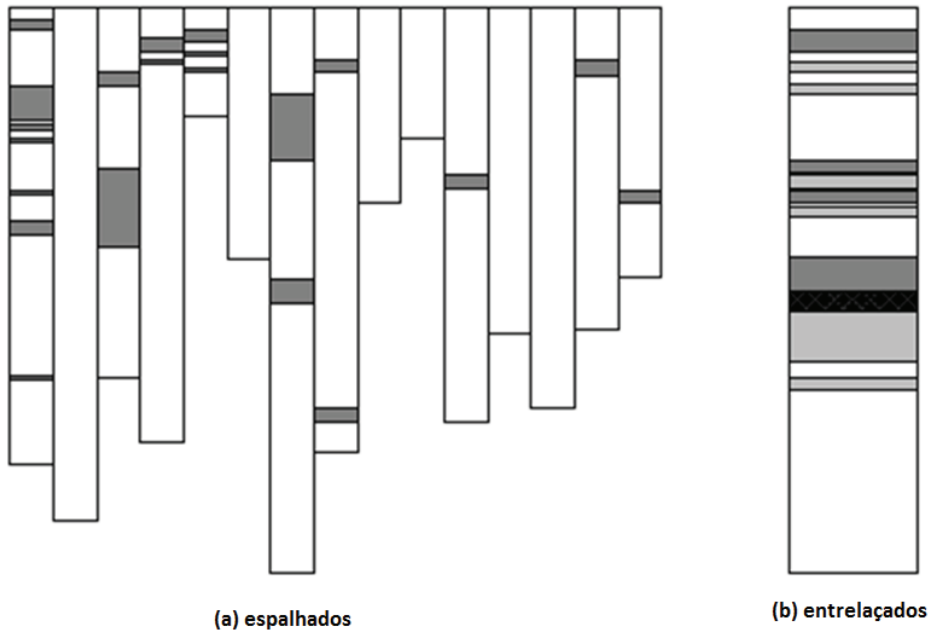



Figura 1.1: Interesses espalhados (a) e entrelaçados (b).

A idéia de considerar características separadamente na implementação de um sistema é original de Dijkstra [Dijkstra 2002, Dijkstra 1997] e Parnas [Parnas 1979]. Estes primeiros trabalhos definiram o que seria característica de maneira flexível, sendo algo que um desenvolvedor queira considerar como uma unidade conceitual de um programa [Robillard 2003]. Idealmente, características deveriam ser organizadas, encapsuladas dentro de um módulo. Por exemplo, um programa desenvolvido na linguagem C que possui a funcionalidade de classificação de sequências de números inteiros, "classificação"pode ser encapsulado em uma função. Modificação da implementação do algoritmo de "classificação"não exigirá atualizar as demais partes do código que chamam a respectiva função. Porém, na prática as características que um desenvolvedor deve considerar durante a evolução de um programa, normalmente, se encontram espalhadas no código fonte, em módulos diferentes e ao mesmo tempo entrelaçadas dentro de módulos [Tarr et al. 2002]. A Figura 1.1 [Robillard 2003] ilustra como as características podem se encontrar dispersas e entrelaçadas em um código fonte. Foi utilizado o sistema *SeeSoft* [Eick et al. 1992] para representar os entrelaçamentos e dispersões, onde retângulos brancos representam o código fonte para um módulo (por exemplo, um arquivo de C ou uma classe Java). Para representar características dispersas, retângulos em cinza foram utilizados para indicar o código fonte relevante a uma característica. Este código fonte pode estar espalhado em múltiplos módulos. Na representação do entrelaçamento (Figura 1.1.b), mostra-se um único módulo. Este módulo implementa dois interesses, indicados pelos retângulos em tons diferentes de cinza. Estes dois interesses, também, possuem códigos que se sobrepõe, representados pelos retângulos em preto. Em outras palavras, entrelaçamento significa a presença de código que implementa interesses diferentes dentro de um módulo.

```
public void save( )
{
    if (initialized)
        _save( );
}
```

Figura 1.2: Método *AbstractOptionPane.save()* do *Jedit*.

Para melhor visualização do problema de localizar, entender e documentar características de interesse durante uma manutenção no sistema, será apresentado um exemplo de evolução de programa que envolve características de interesse fragmentados, este exemplo foi retirado de [Robillard 2003]. O sistema utilizado neste exemplo é o editor de texto *Jedit*, que é um programa com código aberto, escrito em Java e com aproximadamente 65.000 linhas de código e mais de 301 classes distribuídas em 20 pacotes. O *Jedit* possibilita que os usuários visualizem e editem arquivos de texto, executem procuras por expressões regulares, etc. O exemplo apresentado foca na função de *autosave*. Na versão 4.6 - pre6 do *Jedit*, qualquer mudança em texto não salva é arquivada no *buffer* e é gerado um arquivo de segurança em intervalos regulares. Esta frequência pode ser alterada pelo usuário por meio de uma funcionalidade disponível no "Menu"principal da aplicação. Se o *Jedit* for interrompido por algum motivo, e o texto não tiver sido salvo pelo usuário, esta função é executada e o texto será recuperado a partir do *backup* da função de *autosave*. O usuário pode desabilitar o *autosave*, atribuindo o valor zero para a frequência. Porém, esta opção não é documentada, e somente pode ser descoberta, se o código fonte for analisado.

Suponha que seja realizada uma solicitação de manutenção no *Jedit*, em que a aplicação seja modificada para que de maneira explícita se desabilite a função de *autosave*. Para executar esta manutenção, seria necessário entender e localizar várias características de interesse. Será apresentada a análise de uma destas características: "Salvar Estado do Componente". A análise pode se iniciar a partir do método *save()* presente no código do menu de opções, que tem como função salvar o estado do componente. Porém, a compreensão desta característica não será obtida pela simples análise do método *save()*, pois esta característica se encontra espalhada no código fonte. Ao tentar recuperar a informação sobre quem chama o método *save()*, percebe-se que este é chamado em um único ponto, dentro do método *save()* da classe *AbstractOptionPane*, na superclasse *LoadSaveOptionPane*, conforme mostrado na Figura 1.2.

Longe de esclarecer as circunstâncias em que o *save()* é chamado, a identificação desta chamada revela uma complexidade adicional para a "Salvar Estado do Componente". Inicialmente, deve-se determinar as circunstâncias em que o *save()* da classe *AbstractOption-*

*Pane* é chamado. Como *LoadSaveOptionPane* é uma subclasse de *AbstractOptionPane*, então, talvez uma investigação adicional seja necessária para determinar se os métodos de *LoadSaveOptionPane* podem afetar o estado do campo inicializado. Evoluindo nesta investigação, é possível observar o código fonte do método *OptionGroup.save()*, que é chamado por *AbstractOptionPane.save()* (Figura 1.3). Nota-se que a chamada ao método *save()* (Figura 1.3, linha 12) está embutida em algum código transversal. A extensão desta investigação iria mostrar que a avaliação das chamadas do método *save()* se prolonga por pelo menos mais nove métodos e seis classes diferentes.

A localização das características é prejudicada pela forma como o código se encontra disperso e entrelaçado. Ao utilizar um ambiente integrado de desenvolvimento para realizar as solicitações, o desenvolvedor deverá entender e argumentar sobre estas implementações, navegando por múltiplas janelas do editor, e cada uma destas fornecerá apenas um fragmento da informação necessária para compreender a característica. Devido à implementação da característica estar entrelaçada, cada pedaço do código implementado é confuso, com detalhes que não pertencem à característica. Tal como pode ser visualizado no código da Figura 1.3, com estruturas transversais e tratamento de exceções em meio às características.

Outro ponto interessante neste exemplo é que para compreender a característica "Salvar Estado do Componente", deve se compreender a característica "Gerência de Objetos de Propriedade". A presença de código da "Gerência de Objetos de Propriedade"dentro de "Salvar Estado do Componente"ilustra um ponto importante: as características não existem isoladamente. As características estão integradas numa base de código, interagem com outras características e os limites entre estas diferentes características no código fonte não é claramente definido. Aliás, estudos mostram que as interações entre características, frequentemente, são obstáculos durante as atividades de evolução de programas [Robillard 2003, Baniassad et al. 2002].

Uma atividade de evolução de programa, como a apresentada sobre a função *autosave* no *Jedit*, exige que se considere diferentes características do sistema, tal como aquela que salva as informações contidas nos componentes de interface e a que administra as propriedades globais da aplicação. As características até podem ser conceitos simples e óbvios em um nível abstrato, porém, suas implementações frequentemente são espalhadas e entrelaçadas, tornando difícil a atividade de localização e compreensão da estrutura e do comportamento. O espalhamento e entrelaçamento de características em um código fonte é consequência de quatro causas principais: projeto insuficiente, limitações das linguagens de programação, surgimento de novos requisitos durante a evolução do programa e a deterioração do código durante as manutenções mal estruturadas [Robillard 2003]. Adicionalmente, os limites das características não são claramente definidos, e frequentemente interagem com outras características, fazendo com que seja difícil focar em uma única característica durante a investigação e tornando a atividade de localização de ca-

```
1.  public void save ( )
2.  {
3.      Enumeration enum = members.elements ( );
4.
5.      while  (enum.hasMoreElements ( ) )
6.      {
7.          Object elem =  enum.nextElement ( );
8.          try
9.          {
10.             if (elem instanceof OptionPane)
11.             {
12.                 ((OptionPane) elem) .save ( );
13.             }
14.             else if (elem instanceof OpitionGroup)
15.             {
16.                 ((OptionGroup) elem) .save ( );
17.             }
18.         }
19.         catch (Throwable t)
20.         {
21.             Log.log (Log.ERROR, elem,
22.                     "Error saving option pane");
23.             Log.log (Log.ERROR, elem, t);
24.         }
25.     }
26. }
```

Figura 1.3: Método *OptionGroup.save()* do *Jedit*.

racterísticas uma tarefa custosa e desafiadora quando o desenvolvedor ainda não está familiarizado com o sistema [Robillard 2003]. O entendimento incompleto de uma característica dispersa pode levar a modificações incorretas, ineficientes ou não respeitando o projeto existente.

A dificuldade de localizar e entender o código relevante a uma mudança, a ausência de projetos e documentações, a falta de técnicas adequadas para determinar o impacto de uma modificação e a pressão sobre os desenvolvedores contribuem para o mau entendimento do código fonte [Arnold 1996]. Como consequência, limitações do projeto são transgredidas e relações adicionais entre módulos são introduzidos ao código, resultando frequentemente em mais fragmentação e entrelaçamento das características em código fonte.

Esta dificuldade de localização e entendimento do código fonte de uma determinada característica caracteriza o primeiro problema motivador do trabalho descrito nesta dissertação. Diante desta dificuldade apresentada, é interessante capturar, mesmo que parcialmente, informações sobre a implementação das características apoiando uma atividade de investigação mais sistemática do código fonte de interesse. Adicionalmente, se estas informações forem arquivadas, documentadas, os demais desenvolvedores teriam acesso à estas e isto poderia os auxiliar em atividades de manutenção que envolvem estas características, aproveitando assim o esforço prévio [Robillard 2003].

Em [Sefika et al. 1996], estima-se que até 90% do custo de desenvolvimento de sistemas é gasto em atividades de manutenção e evolução. Ao executar uma tarefa de

manutenção, é necessário identificar e localizar a porção de código que precisa ser alterada. Localizar esta parte do código que precisa ser alterada é relativamente fácil quando o sistema está documentado corretamente e é possível rastrear dos documentos de alto nível para o código fonte. Em um mundo ideal, características são localizadas utilizando a rastreabilidade documentada [Wilde et al. 2003]. As ligações de rastreabilidade entre áreas específicas do código e os respectivos requisitos funcionais ajudam na compreensão da motivação e fundamentação daquele trecho de código analisado, e delimitam a área de código diretamente ligada a qualquer tipo de manutenção ou alteração em determinada funcionalidade. Esta rastreabilidade é importante tanto para realizar as manutenções como para compreender as manutenções e evoluções realizadas nos sistemas, além de limitar o espaço de busca em muitos casos. A Rastreabilidade de Requisitos é definida por Gotel em [Gotel e Finkelstein 1994] como sendo a habilidade de descrever e seguir a vida de um requisito nos dois sentidos de abstração. A simplicidade dessa definição esconde a complexidade do problema. A garantia da conformidade do programa com os seus requisitos é uma exigência básica da indústria de desenvolvimento de programas, mas nem sempre é alcançada. Com o objetivo de resolver esse problema, surgiram as práticas de Rastreabilidade de Requisitos. Pesquisadores e a indústria de sistemas desenvolveram diferentes métodos e ferramentas para aperfeiçoar a gerência de requisitos, que, até então, infelizmente falharam em alcançar uma adoção generalizada [Neumuller e Grunbacher 2006].

Apesar da rastreabilidade de requisitos ser utilizada na Engenharia de Software há alguns anos, códigos fonte frequentemente evoluem sem a atualização da documentação [Ramesh et al. 1997]. Infelizmente, dois problemas surgem com esta prática: em primeiro lugar, com a evolução do programa, as características relevantes para a manutenção podem não estar mapeadas (rastreadas) corretamente para os requisitos, tal como inicialmente concebidas. Em segundo lugar, as documentações de rastreabilidade em geral são de difícil atualização e, muitas vezes, não são mantidas devido ao tempo e pressões do trabalho em curso. Assim, é bastante raro encontrar boa documentação de rastreabilidade para sistemas legados [Wilde et al. 2003], pois manter a consistência da rastreabilidade é uma atividade custosa e consome um tempo considerável [Antoniol et al. 2005]. Além do mais, na rastreabilidade entre requisitos e código é possível que os destinos não sejam unidades, mas estejam fragmentados e diluídos em todo o código, como pode ser notado na Figura 1.1, caracterizando uma forma de ligação diferente. O requisito "Salvar o Estado do Componente"é um exemplo, pois este não se liga a uma área bem definida e delimitada do código, e a implementação que resolve o requisito pode estar espalhada por todo o código.

O segundo problema trabalhado nesta dissertação está relacionado a esta necessidade de documentação da rastreabilidade entre as características e os elementos de código que as implementa. Entende-se que esta rastreabilidade deveria ser de fácil atualização

para poder garantir o uso e consistência desta prática, contribuindo para a melhoria da compreensão dos sistemas.

Informações extraídas de rastros de execução vêm sendo bastante utilizadas como alternativa para facilitar a compreensão de características dos programas. Este estudo analisará o impacto do uso sistemático de informação de rastros de execução no tempo de execução e na taxa de acerto na procura de informação no código fonte durante atividades de manutenção de programas. Espera-se que este estudo revele possíveis desafios para aplicação desta abordagem em larga escala.

## 1.1 Objetivo

Atividades de compreensão normalmente, quando solicitadas, são apresentadas no domínio dos usuários, ou seja, como características do sistema que devem ser compreendidas para alguma atividade. Diante de uma atividade que envolve a compreensão do sistema, como correção de erros, reuso de código, implementar nova funcionalidade, verificação de evolução do sistema etc., o analista precisa localizar a característica de interesse no código fonte para compreender e então realizar a respectiva atividade. Portanto, a localização e documentação dos elementos de código que implementam as características de interesse e seus comportamentos é importante para a compreensão de sistemas. Entretanto, mesmo para sistemas de porte médio esta tarefa pode se tornar extremamente laboriosa, pois o número de elementos de código e chamadas (interações) pode ser muito grande. A compreensão do código fonte a partir de seus requisitos e propósitos originais necessita de mecanismos efetivos para prover a localização (mapeamento) e documentação de características, especialmente na análise de sistemas desconhecidos.

Uma possível solução para obtenção do mapeamento de característica em código fonte é a utilização de análise dinâmica, pois é possível associar a uma característica os trechos de código fonte utilizados durante a execução da mesma. Diversos trabalhos relacionados que serão discutidos na Capítulo 5 estudaram esta alternativa e avaliaram aspectos positivos e negativos da mesma. Apesar de ser intuitiva, a abordagem de análise dinâmica impõe algumas restrições, tais como: dependência do cenário escolhido para a execução e excesso de informação provida pela coleta de rastros de execução. O fato é que o uso de informação dinâmica como mecanismo de localização de característica ainda é pouco difundido como prática abrangente entre os desenvolvedores, bastando verificar a escassez de ferramentas amplamente difundidas em IDEs para este propósito.

Este trabalho propõe como contribuição uma abordagem que pretende ajudar o desenvolvedor na compreensão dos códigos das características de programa, utilizando análise dinâmica e estática para obter informações necessárias aplicada a sistemas orientados a objeto desenvolvidos em Java. Método utilizado deve lidar com um grande volume de informação, por isto é necessário o uso de visões e filtragens, e ser integrado ao IDE Eclipse.

Esta abordagem pode ser desenvolvida para avaliação de sistemas em outras linguagens de programação e integrados à outros IDEs. A decisão de se trabalhar com análise de características partiu da necessidade e importância de se investigar o sistema a partir de suas funcionalidades.

O objetivo desejado é tornar mais rápida a localização da característica de interesse, com informações que direcionam a compreensão do sistema e propiciam maior taxa de acerto nas atividades de manutenção.

Com o objetivo de avaliar os benefícios e limitações da abordagem foi conduzido um estudo do uso da mesma em tarefas de manutenção de programa. O estudo propõe a realização de experimentos controlados para verificar as contribuições desta abordagem e se o objetivo apresentado foi alcançado. Estes experimentos serão realizados por grupos de analistas desenvolvedores, através da execução de atividades reais de manutenção sobre sistemas alvo com diferentes proporções, onde perguntas serão respondidas a fim de qualificar e quantificar a contribuição desta abordagem para o campo de compreensão de sistemas. Conforme afirmado em [Quante 2008, Cornelissen et al. 2008] e , experimentos como este são raramente executados devida a dificuldade de execução dos mesmos. Dentre os trabalhos relacionados em [Cornelissen et al. 2008], apenas 6 dentre 176 artigos analisados realizaram experimentos utilizando sujeitos humanos na sua validação. Dentre estes 6, apenas 1, [Quante 2008], realizou os experimentos seguindo uma metodologia muito próxima àquela proposta nesta dissertação. Porém, [Quante 2008] avalia uma abordagem totalmente diferente da aqui apresentada. Segundo [Quante 2008, Cornelissen et al. 2008], a participação de seres humanos nas avaliações é importante para o campo compreensão de sistemas e constitue uma importante contribuição. Devido a metodologia adotada para as avaliações, será possível reponder perguntas ligadas a contribuição da abordagem para as atividades que envolvem compreensão dos sistemas.

## 1.2 Resumo da Dissertação

Em resumo, este trabalho apresenta uma proposta de abordagem para localização de características que utiliza análise dinâmica para obtenção de visões que pretendem auxiliar no problema da compreensão de sistemas com códigos de características espalhados ou entrelaçados. É proposto um estudo controlado com sujeitos humanos, executando atividades reais de manutenção em sistemas de diferentes portes para verificar se o objetivo desta abordagem foi alcançado. No próximo capítulo será apresentada a abordagem de análise dinâmica proposta. No Capítulo 3 será apresentada a metodologia de avaliação adotada e os experimentos realizados para avaliação. O Capítulo 4 apresenta os resultados e discussões. O Capítulo 5 apresenta os trabalhos relacionados. Finalmente, no Capítulo 6 são apresentadas as conclusões deste trabalho.

# Capítulo 2

# Abordagem Proposta

Neste capítulo será apresentada a abordagem proposta por este trabalho, esta utiliza os conceitos de localização de características devida a necessidade e importância de se investigar o sistema a partir de suas funcionalidades ou conjunto de funcionalidades.

Para obter a localização das características, a abordagem proposta utiliza análise dinâmica. Originária da disciplina de depuração, a utilização da análise dinâmica para fins de compreensão de sistemas ganhou grande interesse e tornou-se mais popular desde o trabalho de Wilde e Scully [Wilde e Scully 1995], conforme apresentado em [Cornelissen et al. 2008]. Em relação à análise estática, a análise dinâmica fornece uma imagem parcial do sistema, ou seja, os resultados obtidos são válidos para os cenários que foram executados durante a análise. Em [Cornelissen et al. 2008], foi realizada uma análise de 176 artigos, criteriosamente selecionados, relacionados a análise dinâmica aplicada a compreensão de sistema, e verificou-se que a análise de características constitui o terceiro maior grupo de atividades com 30 representações no conjunto 176 artigos, contando com contribuições importantes para a literatura correlata.

Conforme comprovado por Eisenbarth [Eisenbarth et al. 2003], o refinamento da análise dinâmica utilizando a análise estática traz melhores resultados para a compreensão do sistema. Neste sentido, esta abordagem propõe o uso de dados estáticos para refinar as informações geradas pela análise dinâmica, e assim, ter informações suficientes para a compreensão do sistema. A análise estática é possibilitada pelas ferramentas do IDE Eclipse, com o qual as ferramentas da análise dinâmica podem ser integradas.

A análise dinâmica gera grandes quantidades de informações para os desenvolvedores, logo, o uso de visões e filtros é adotado nesta abordagem para apresentar as informações de maneira clara e diminuir a poluição visual causada pelo grande volume de informações geradas. Em [Cornelissen et al. 2008], observou-se que o uso de visualizações constitui o maior grupo de métodos utilizado em trabalhos relacionados a este, e o uso de filtros constitui o segundo maior grupo. Isso pode ser causado por diversas razões, entre as quais, a acessibilidade às ferramentas padrão para gráficos, diagramas de sequência e assim por diante, e a necessidade em diminuir a confusão visual causada pelo grande volume de

informações. Estas visões podem variar desde o nível de detalhamento das classes ao alto nível de visões da arquitetura [Richner e Ducasse 1999, Schmerl et al. 2006, Walker et al. 1998]. Na abordagem aqui apresentada, as quatro visões obtidas a partir dos rastros de execução das características e montadas a fim de auxiliar na compreensão das características são: o mapeamento, grafo de chamada, classificação das classes e o mapeamento com classificação.

A abordagem proposta foi estruturada para atuar sobre sistemas orientados a objeto (OO). A opção por trabalhar com OO foi por ser um paradigma amplamente utilizado e pela facilidade de instrumentação. Além do mais, a seleção de uma linguagem amplamente conhecida foi importante pra viabilizar a avaliação realizada por dois motivos: a participação de sujeitos que deveriam conhecer a linguagem Java e a facilidade de se encontrar sistemas de código aberto desenvolvidos com o paradigma de orientação a objeto em Java que atendessem os requisitos para serem avaliados nos estudos de caso. Em [Cornelissen et al. 2008], os autores apontam que trabalhos que atuam sobre sistemas orientados a objeto são o alvo mais comum no conjunto de artigos selecionados, 79 de 114 artigos foram desenvolvidos em linguagem Java ou Smalltalk. Não se tem certeza da razão, mas provavelmente está relacionada à facilidade de instrumentação, a adequação de certas visualizações comportamentais (por exemplo, diagramas de sequência UML) para sistemas OO, a complexidade (percepção) de aplicações OO requerendo análise dinâmica, ou simplesmente o fato de que para muitos pesquisadores a compreensão do sistema têm um forte interesse em orientação a objetos.

Logo, a abordagem apresentada neste trabalho pretende ajudar o desenvolvedor na localização de características de programas, utilizando análise dinâmica e estática para obter informações necessárias aplicada a sistemas orientados a objeto desenvolvidos em Java, e para tanto o método utilizado envolve visões e filtragens.

Simmons e colegas [Simmons et al. 2006], propõem três passos para localização de características, sendo eles a análise dinâmica, diferenciação dos rastros e análise estática. Esta abordagem foi dividida em quatro passos, descritos nas próximas subseções, que se assemelham ao proposto por Simmons, apresentados nas quatro raias da Figura 2.1: o planejamento dos roteiros de execução das características, extração dos rastros de execução, geração das visões e, finalmente, análise das visões geradas e análise estática. Também serão introduzidas as ferramentas: *TraceExtractor*, *TraceToConcern*, *TraceToVisions*, *ConcernMapper* e *Graphviz*, utilizadas na abordagem. Estas ferramentas, exceto a *Graphviz*, são executadas dentro do Eclipse, a fim de atender a necessidade do analista de interagir com o código fonte dentro de um IDE e de obter informações estáticas.

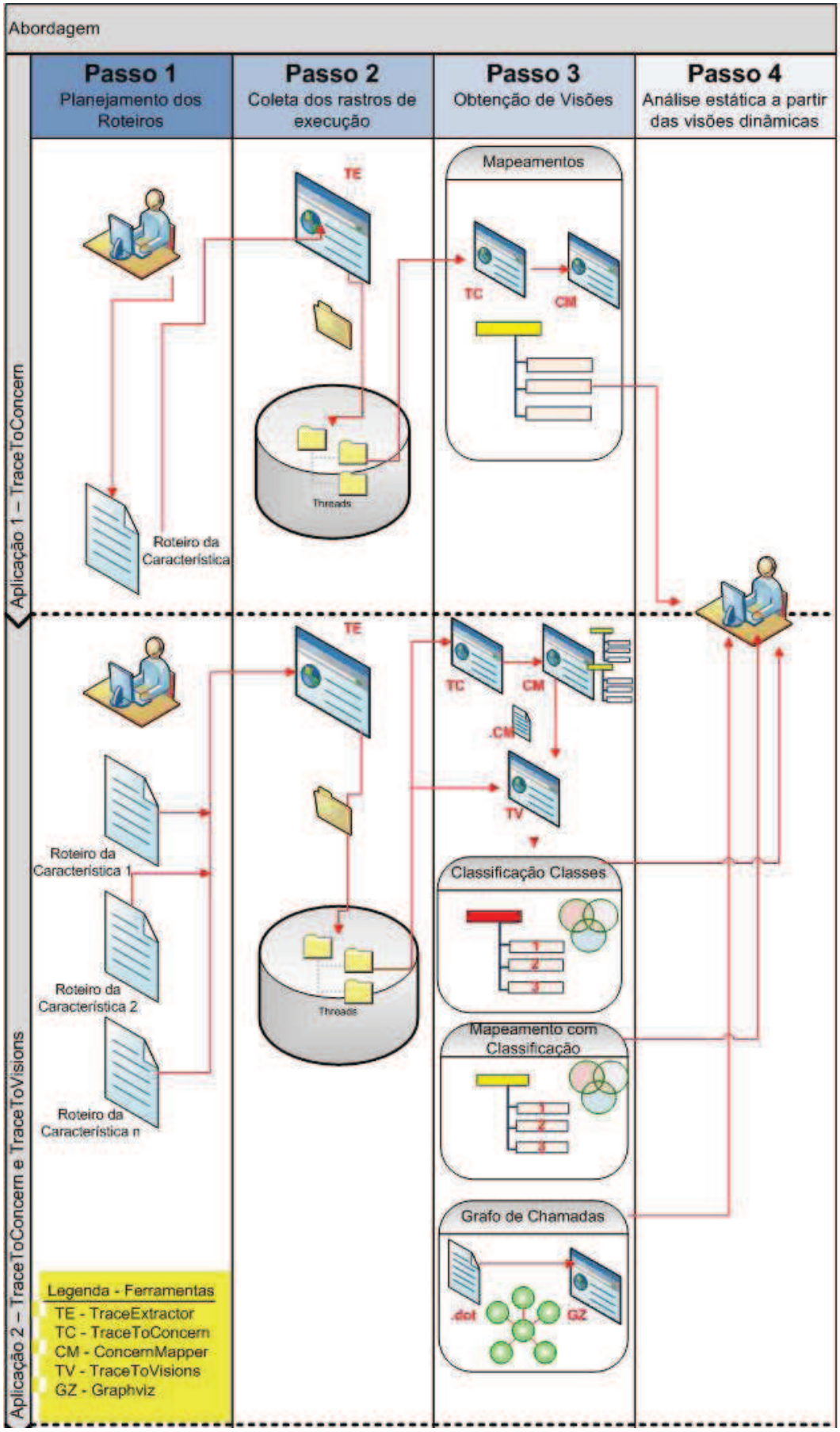



Figura 2.1: Visão geral da abordagem.

## 2.1 Passo 1 - Planejamento dos Roteiros

Como esta abordagem utiliza análise dinâmica, que consiste na análise de propriedades de um programa em execução, existem duas propriedades da análise dinâmica que devem ser cuidadosamente contempladas, sendo estas a precisão das informações coletadas e a dependência das entradas no programa. A precisão das informações diz respeito à capacidade dos mecanismos de extração das informações em ajustar a informação exata a ser coletada conforme as necessidades do analista. A dependência das entradas refere-se à sensibilidade as variações nas entradas ou interações com o programa durante a execução. Estas propriedades são importantes por permitirem limitar o escopo da análise e relacionar as variações na entrada às mudanças internas no comportamento dos programas [Sobreira e Maia 2008]. Em função da natureza da análise dinâmica, sensível às entradas do programa e sua execução, deve-se fazer um planejamento prévio para as execuções (Passo 1 da Figura 2.1) , e assim poder utilizar esta análise de maneira efetiva e satisfazer os objetivos estabelecidos. Este planejamento engloba a definição adequada de roteiros de execução e a execução destes de maneira fidedigna.

A definição dos roteiros dependem da finalidade da análise e devem especificar os passos a serem seguidos para a execução do sistema em análise. O analista antes de iniciar o processo de análise do programas deve definir bem as suas características. No contexto específico da localização de características, os roteiros devem estar ligados as características de interesse e exercitá-las seguindo as recomendações da abordagem proposta. O analista deve garantir que o roteiro execute apenas a característica ou poderá trazer informações erradas. O planejamento dos roteiros de execução é uma atividade alheia ao uso de ferramenta nesta abordagem, dependendo unicamente do analista desenvolvedor, e seu conhecimento sobre o sistema e o domínio da aplicação. Para isso a consulta à documentação ou a usuários do sistema pode ser necessária.

## 2.2 Passo 2 - Coleta dos Rastros de Execução

Toda abordagem baseada em análise dinâmica precisa de alguns mecanismos para a extração dos dados durante a execução do programa. Para isso, o sistema alvo deve passar por um processo de instrumentação para a captura de eventos gerados durante sua execução, seja esta feita diretamente no sistema, ou seja, pelo uso de algum recurso do ambiente de execução que forneça funcionalidade similar, por exemplo, o uso da plataforma de depuração da maquina virtual em sistemas Java - JPDA. Existem vários métodos, que variam de métodos intrusivos no código fonte até métodos transparentes que podem ser habilitados e desabilitados com facilidade como a instrumentação do código fonte, instrumentação de *bytecodes*, instrumentação da máquina virtual (JPDA) e a instrumentação baseada em aspectos [Sobreira e Maia 2008].

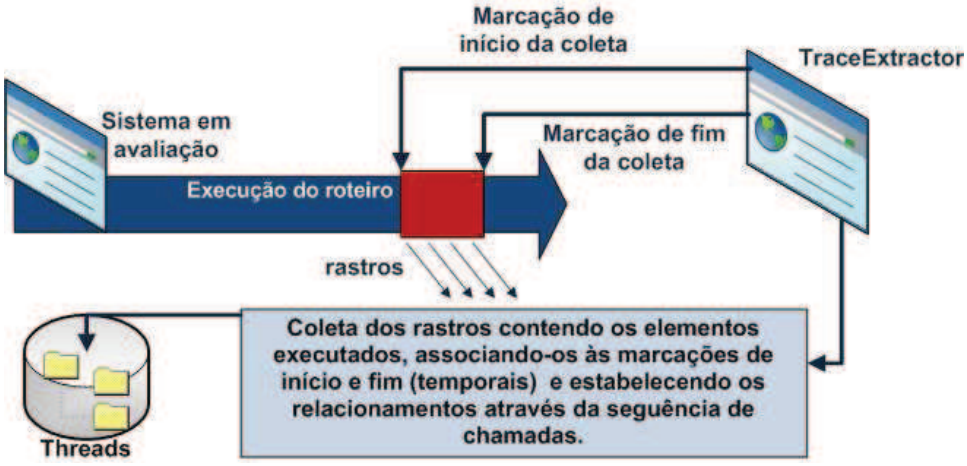



Figura 2.2: Processo de coleta e arquivamento dos rastros de execução.

A captura dos rastros de execução para esta abordagem foi implementada usando orientação a aspectos por meio de AspectJ. Dentre as razões para a escolha estão a flexibilidade, facilidade e simplicidade da implementação. A integração entre o aspecto responsável pela captura das informações e o sistema alvo é feita de forma transparente, podendo ser habilitada e desabilitada com facilidade. O controle sobre as partes analisadas pode ser feitos em nível de instruções, acesso a variáveis, métodos, classes ou pacotes. A ferramenta foi originalmente montada por Sobreira e Maia [Sobreira e Maia 2008], e foi adaptada a *TraceExtractor* para este trabalho.

Para a coleta dos rastros, o usuário da abordagem irá compilar o sistema alvo, com o *TraceExtractor* sendo executada em paralelo e integrada a este sistema, e irá executar a(s) característica(s) desejada(s) conforme o roteiro predefinido, realizando a marcação do início e fim da coleta do rastro de cada característica. A *TraceExtractor*, durante a execução das características, realiza a intercepção e registro dos elementos de código que foram executados, e arquiva esta informação separada por *threads* de execução em uma pasta selecionada pelo usuário, conforme processo descrito na Figura 2.2.

Após a coleta dos rastros de todas as características de interesse, dois tipos de arquivos serão gerados: arquivos com o rastro da execução para cada *thread* iniciada pelo sistema alvo, onde cada linha do arquivo corresponde a uma chamada de método devidamente qualificada e anotada com uma marca de tempo; e, um arquivo de marcação dos instantes de início e fim de cada característica executada. O arquivo de rastro contempla o nome qualificado da classe, nome do método, o *timestamp* (representado por um inteiro), nível de aninhamento do método na pilha de chamadas e um código de identificação do objeto ou classe sendo chamada.

A maneira como os rastros são coletados, notificando dos pontos de início e fim da execução das características, possibilita a diferenciação dos rastros destas. Por este motivo, a aplicabilidade da abordagem é restrita as características controláveis e observáveis pelo usuário. Um detalhe importante desta marcação, é que quanto mais específico for

a execução, delimitando bem a característica, maior será a redução no espaço de busca pelo(s) elemento(s) de código que o implementa, pois mais especifica será a rastreabilidade gerada.

Quanto ao número de características executadas para cada coleta dos rastros, há abordagens que lidam com um único rastro para todas as características de interesse, por exemplo, como nos trabalhos de [Liu et al. 2007, Rohatgi et al. 2008, Salah et al. 2006], enquanto outras abordagens utilizam um rastro para cada característica [Wilde e Scully 1995, Eisenbarth et al. 2003, Eisenberg e Volder 2005]. Assim, a escolha depende dos princípios de cada abordagem [Sobreira e Maia 2008]. Na abordagem aqui apresentada, o analista poderá utilizar as duas formas de coleta. A escolha da melhor maneira de se coletar o rastro irá depender unicamente da necessidade de análise da(s) característica(s) de interesse. Como esta abordagem é integrada ao IDE Eclipse, a extração dos rastros e geração das visões pode ser realizada durante a análise do sistema, podendo ser executada quantas vezes for necessária para compreensão do sistema.

Uma coleta mal definida e/ou mal executada pode ocasionar problemas com elementos de código que não deveriam ser coletados e foram (falso positivos - FP) e elementos não cobertos (falsos negativos - FN) pelos roteiros de execução escolhido. Este tipo de problema é natural em técnicas que utilizam análise dinâmica, sendo citadas por diversos autores de trabalhos relacionados a este [Wilde e Scully 1995,Lukoit et al. 2000,Eisenbarth et al. 2003,Eisenberg e Volder 2005]. Mesmo com roteiros bem definidos e executados, os rastros de execução poderão conter ruídos (FP e FN) gerados pela ocorrência de eventos invisíveis a observação do usuário.

## 2.3 Passo 3 - Obtenção de Visões

Nesta seção serão apresentadas as visões desenvolvidas para esta abordagem, visando apresentar as informações presentes nos rastros de execução de maneira clara e dinâmica.

### 2.3.1 Visão 1 - Mapeamento

A visão Mapeamento apresenta a rastreabilidade das características para os elementos de código (classes e métodos) executados durante a coleta. A ferramenta proposta pela abordagem para gerar esta visão é a *TraceToConcern*, a qual utiliza os rastros de execução e marcações obtidos no Passo 2 da Figura 2.1 para gerar um arquivo de saída contendo o mapeamento das características de interesse, que visa auxiliar na localização destas. Este mapeamento é apresentado no formato de leitura da ferramenta *ConcernMapper*, que viabiliza a estrutura em forma de árvore e integração com o código fonte. O processo de geração desta visão é representado na Figura 2.3, e será detalhado a seguir.

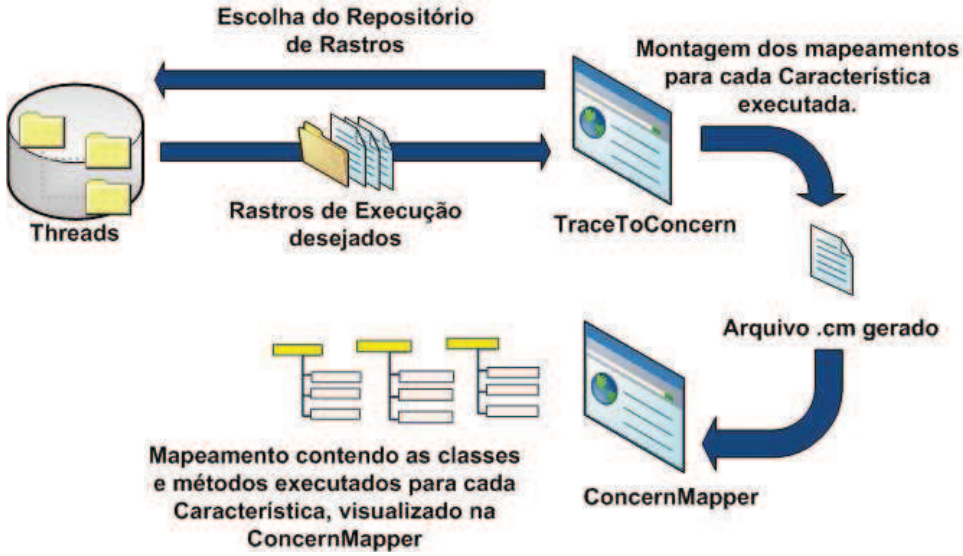



Figura 2.3: Processo de geração dos mapeamentos.

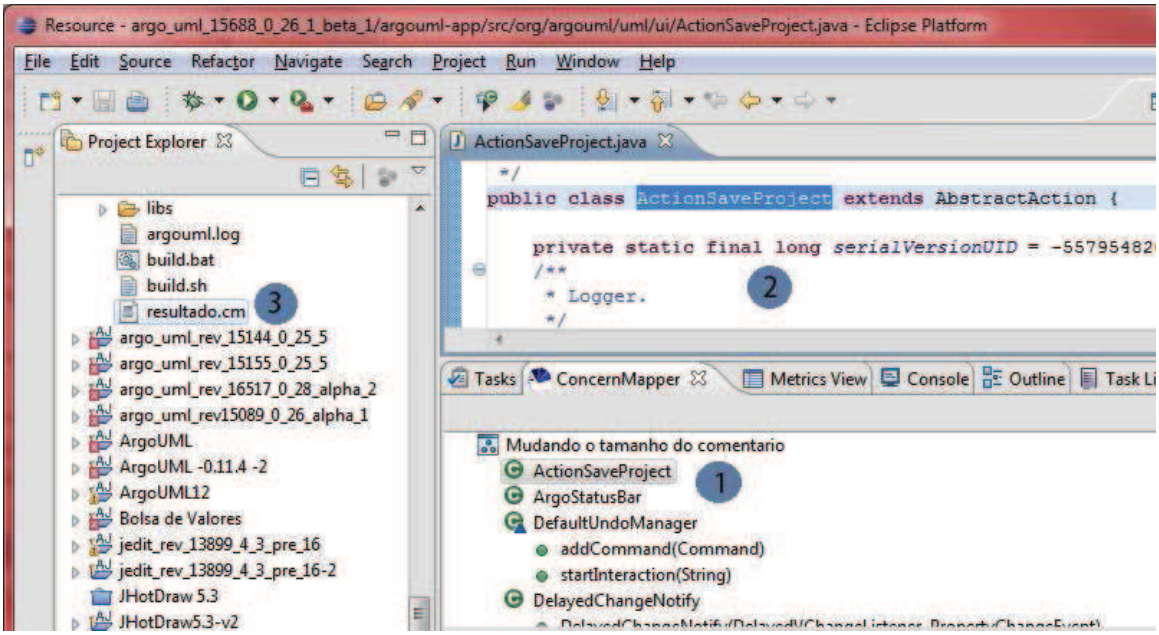



Figura 2.4: Exemplo de mapeamento gerado.

A *TraceToConcern* realiza filtragem de unidades de código de bibliotecas e pacotes no mapeamento, basta informar o nome do pacote. Este filtro possibilita a redução do espaço de busca, ao eliminar do mapeamento elementos de código de pastas que já se sabe não serem interessantes para a atividade em questão.

A geração deste mapeamento se baseia nos rastros e marcadores que delimitaram as características. Para cada classe e método que foi executado no intervalo de tempo delimitado, se verifica os níveis de aninhamento destes para associar os métodos às classes, e estas à característica. Em seguida, são eliminadas as repetições e estabelecidos os vínculos de relacionamento entre os elementos de código e as características, montando assim a rastreabilidade.

O mapeamento gerado pela *TraceToConcern* é um arquivo texto de extensão .cm de leitura pela ferramenta *ConcernMapper* (Figura 2.4, número 3 e 1). Este arquivo contém a estrutura do mapeamento montada, associando as características aos elementos de código (classes e métodos) que as implementam, com algumas informações adicionais necessárias para que esta estabeleça a navegação entre dos elementos listados e o código fonte do sistema [Robillard e Weigand-Warr 2005]. Para cada característica é criado um nó na árvore de interesse com o respectivo nome da característica (primeiro nível da árvore) e associada a esta todas as classes (segundo nível da árvore) e métodos (terceiro nível da árvore) que foram executados durante a coleta dos rastros desta característica. Desta maneira, os elementos de código, que antes se encontravam espalhados e entrelaçados pelo código fonte, se apresentam de maneira organizada ao desenvolvedor. A navegação destes elementos listados para o código fonte é estabelecida, facilitando assim a atividade de localização dos elementos no código fonte, análise e manutenção. A abordagem permite que o programador tenha acesso às unidades de código do mapeamento diretamente no ambiente de codificação (IDE - Eclipse), para não haver interrupção no trabalho de programação. A Figura 2.4, número 2, mostra o resultado da seleção no mapeamento do elemento de código *applyAttributes()*, o código do mesmo é acessado no IDE imediatamente. E, este arquivo .cm pode ser alterado e arquivado, possibilitando o refinamento do mapeamento, o que inclui a eliminação dos falsos positivos e a inclusão dos elementos faltantes (FN), e o uso deste arquivo para documentação da rastreabilidade entre as características os elementos de código.

### 2.3.2 Visão 2 - Grafo de Chamada dos Métodos

A segunda visão da abordagem apresenta o grafo de chamada dos métodos que implementam as características, visando prover ao analista em sua atividade de compreensão do código informações sobre o relacionamento entre os métodos. Serão utilizadas imagens gráficas para esta visão, pois imagens tendem a ser mais fáceis de assimilar do que textos. Os grafos de chamada são gerados utilizando a ferramenta *TraceToVisions*, que com base nos rastros de execução gera uma estrutura com as chamadas dos métodos no formato de grafos do tipo .dot de leitura da ferramenta *Graphviz* [1], seguindo os procedimentos desenhados na Figura 2.5. *Graphviz* é uma ferramenta de código aberto para visualização de grafos.

A *TraceToVisions* foi desenvolvida, com a finalidade de gerar as visões: classificação das classes, mapeamento com classificação e os grafos de chamadas dos métodos que implementam as características. A classificação das classes e o mapeamento com classificação são geradas a partir do arquivo (.cm) gerado pela *TraceToConcern*, e os grafos de chamadas são gerados a partir dos rastros de execução montados pela *TraceExtractor*.

[1] http://www.graphviz.org

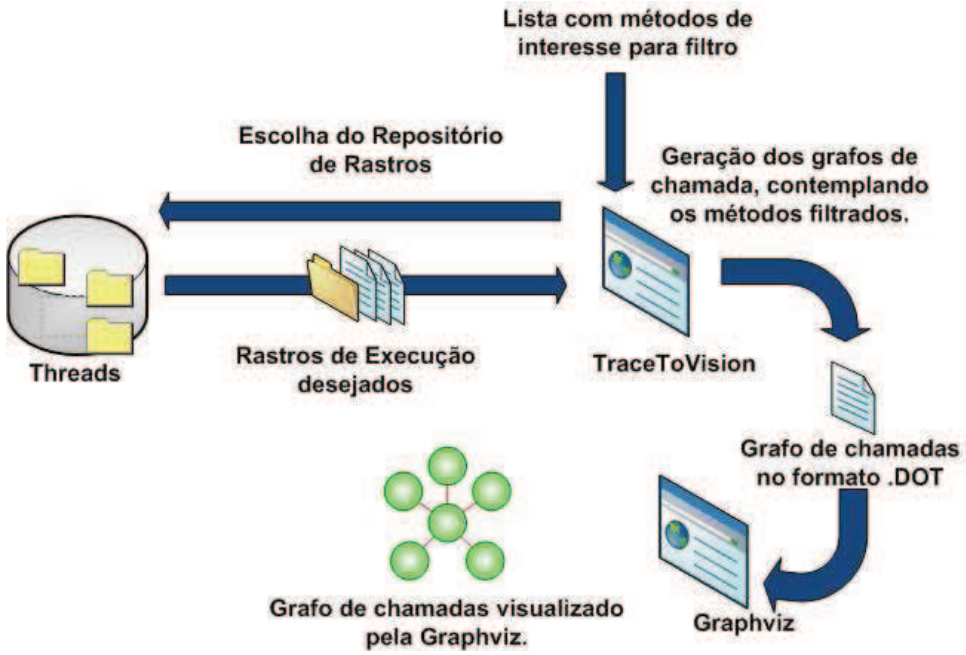



Figura 2.5: Processo de geração dos grafos de chamadas dos métodos.

Portando, esta ferramenta possui como entradas o caminho para o diretório com os rastros de execução, o caminho para o arquivo (.cm) do mapeamento, um filtro que define quais métodos de interesse serão contemplados nos grafos de chamadas e o diretório de saída dos resultados gerados pela ferramenta.

Para gerar os grafos de chamadas, a *TraceToVisions* utiliza os marcadores para delimitar as características e os níveis de aninhamento definidos nos rastros para determinar quais são as chamadas dos métodos e montar os grafos, seguindo a seguinte regra: se o nível de aninhamento N for igual à 1, então este é o primeiro método na cadeia de chamada e nenhum outro método o chamou. Pode-se ter vários métodos com nível de aninhamento igual à 1. Se o nível de aninhamento N for diferente de 1, então o método foi chamado pelo elemento no rastro anterior a ele (olhando em ordem cronológica crescente de execução), que possui o nível de aninhamento igual a N -1. É possível a existência de métodos que não chamam outros métodos, logo a existência do elemento com nível de aninhamento N não implica na existência do elemento com o nível N+1.

Um grafo de chamada com todos os métodos que implementam uma característica simples, pode implicar em um grafo muito grande e de difícil visualização, que em nada acrescentaria a compreensão da característica. Por este motivo, a *TraceToVisions* possui como entrada um filtro, onde a partir das informações presentes na visão Mapeamento (gerado separadamente) pode-se definir os métodos de interesse como parâmetro de entrada. As visualizações são geradas separadas por *thread* de execução. A Figura 2.6 mostra uma visualização gerada.

Acredita-se que os grafo de chamadas dos métodos de interesse que implementam as características possa ser uma visão útil na análise dos elementos de código que sofrem


Figura 2.6: Exemplo de Grafo de Chamadas para o método *AbstractApplication.run* (sistema *JhotDraw* 7.1).

impacto com a manutenção, na compreensão dos relacionamentos entre estes métodos, ajude a averiguar porque este é relevante para uma característica (como citado por Eisenberg em [31]) e, com isso, ajude no direcionamento da localização das características no código fonte.

### 2.3.3 Visão 3 - Classificação de Classes

Assim como nas abordagens de localização de características de Eisenberg [Eisenberg e Volder 2005] e Eisenbarth [Eisenbarth et al. 2003], este trabalho propõem classificar as classes que implementam as características baseando-se nos níveis de participação das mesmas na implementação do conjunto de características em questão. A ferramenta *TraceToVisions* gera esta classificação a partir dos arquivos .cm gerados pela *TraceToConcern*, aplicando os critérios para classificação das classes que foram definidos com base nos critérios apresentados em [Greevy et al. 2006]. Como resultado são apresentadas todas as classes que foram executas durante a coleta do rastro de alguma característica e a sua classificação. A Figura 2.7 apresenta este processo.

São distinguidos quatro níveis de participação das classes nas características em análise, calculados a partir do número de características avaliadas e do número de características que a classe implementa, onde NOF é o número de características presentes na avaliação e NOFC é o número de características implementadas pela classe em avaliação.

- **Única Característica (UC)** é uma classe que participa em apenas uma característica da análise, cuja fórmula para cálculo é (NOFC = 1).

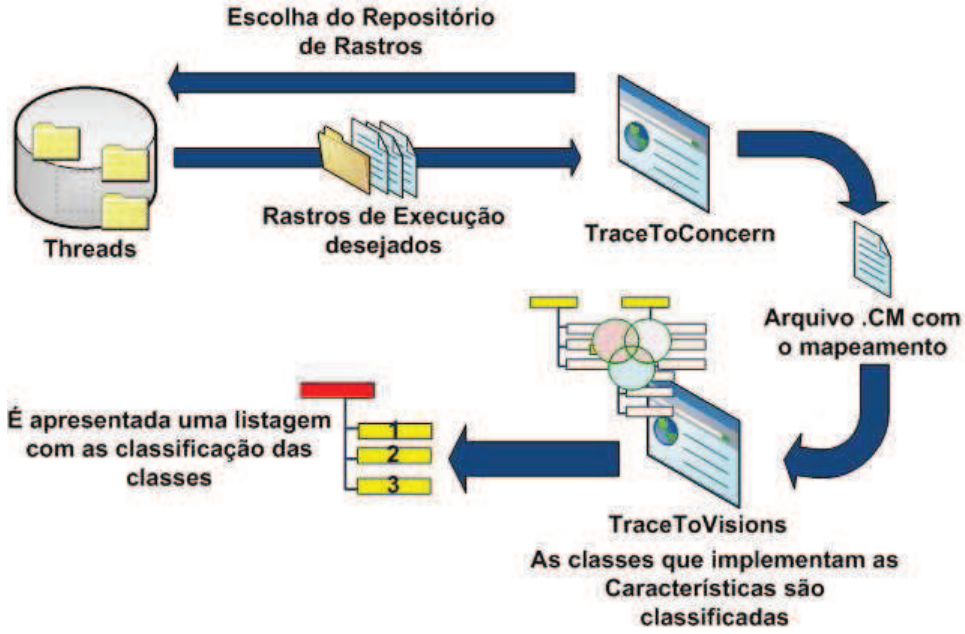



Figura 2.7: Processo de geração da classificação das classes

- **Baixo Número de Características (BNC)** é uma classe que participa em mais de uma característica, mas em menos da metade das características em análise, cuja fórmula para cálculo é (NOFC> 1) ∧ (NOFC <NOF / 2).
- **Alto Número de Características (ANC)** é uma classe que participa na metade ou mais das características em avaliação, mas não em todas. A fórmula para cálculo é (NOFC> 1) ∧ (NOFC> = NOF / 2).
- **Presente em Todas (PT)** é uma classe que participa de todas as características em análise, cuja fórmula para cálculo é (NOFC = NOF).

Como esta classificação é dada em relação ao conjunto de características, se deve utilizar a extração de um único rastro com várias características, o que permite comparar a participação das classes no conjunto. Ao coletar uma única característica por rastro, esta visão irá apresentar a classificação destas como "Presente em Todas", pois implementam todas as características (única) do conjunto e esta informação não teria utilidade.

A Figura 2.8 apresenta um exemplo da visão classificação das classes. Utilizou-se filtros nesta visão para reduzir a confusão visual, devido a possibilidade de se ter um número elevado de classes listadas e para direcionar o interesse pela classificação.

Esta visão pretende apontar o grau de participação das classes no conjunto de características executadas e com isso permitir ao desenvolvedor avaliar: (i) quais classes podem causar maior impacto ao sistema quando alteradas, (ii) quais são classes de infraestrutura do sistema, (iii) quais por serem específicas devem apresentar funcionalidades típicas de uma característica. Acredita-se que esta visão poderá auxiliar o desenvolvedor em atividades de reuso de código, melhorias envolvendo modularização ou programação orientada a aspectos, dentre outras atividades nas quais se necessita saber este tipo de informação.

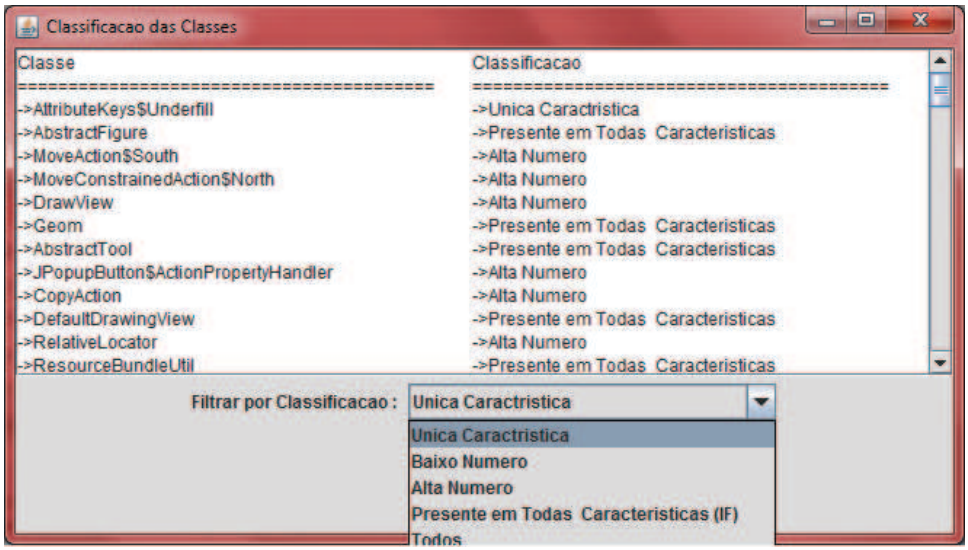



Figura 2.8: Exemplo de classificação das classes, com filtro por classificação.

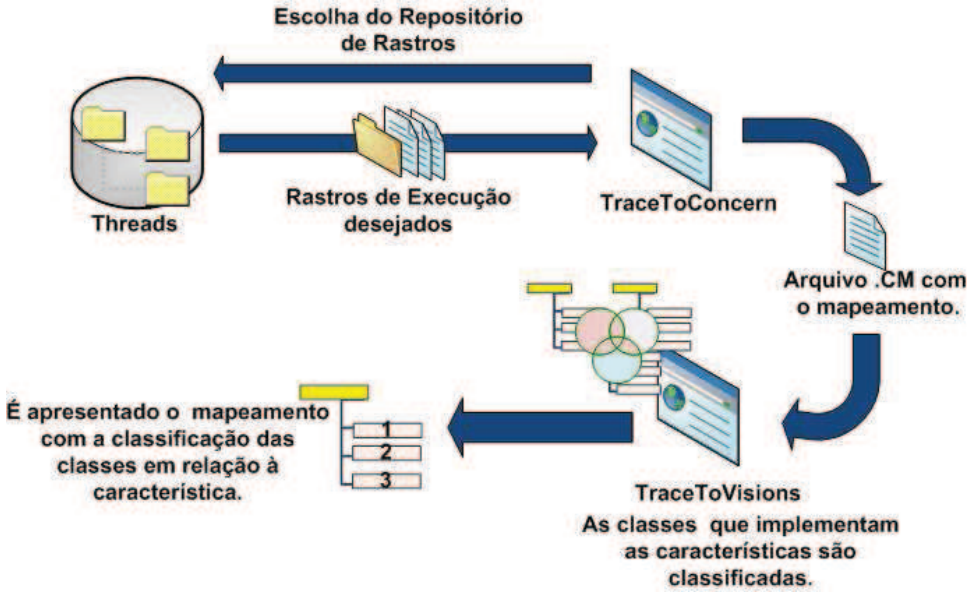



Figura 2.9: Processo de geração do Mapeamento com Classificação.

### 2.3.4 Visão 4 - Mapeamento com Classificação

Enquanto na Visão 3 tem-se uma classificação das classes em relação ao conjunto de características, na Visão 4 tem-se a apresentação organizada por características. Assim como a Classificação das Classes, o Mapeamento com Classificação é gerado a partir do arquivo contendo o mapeamento gerado pela *TraceToConcern*. A ferramenta *TraceToVisions* com base neste arquivo, aplica os critérios definidos para classificação e monta a visão, contendo cada característica em análise e as classes que a implementam devidamente classificadas. A Figura 2.9 esboça este processo.

Com esta visão, permite-se a análise da relação entre as características executadas e esta relação é obtida através dos elementos de código que participam da implementação destas. Os critérios de classificação definidos para esta visão são:

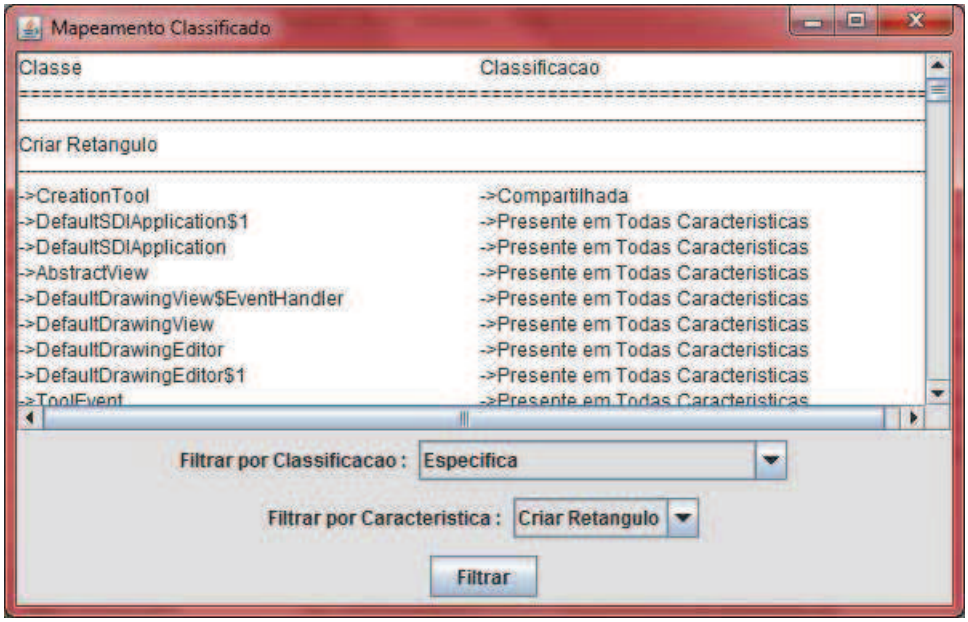



Figura 2.10: Exemplo de Mapeamento com Classificação, com filtro por classificação das classes.

- **Específica:** a classe somente implementa a característica em apresentada, para o conjunto avaliado;
- **Compartilhada:** a classe implementa a característica e implementa outra(s) característica(s) analisadas, mas não todas;
- **Presente em Todas:** a classe implementa todas as características em análise.

Acredita-se que esta visão poderá auxiliar da mesma forma que a Visão 3, estas se complementam, pois uma apresenta a classificação sobre a participação das classes que implementam a característica de interesse e a outra a classificação das classes que implementam o conjunto de características. A Figura 2.10 apresenta um exemplo da Visão 4, e também, utilizou-se de filtros para reduzir a confusão visual.

## 2.4 Passo 4 - Análise Estática em Visões Dinâmicas

O resultado obtido ao final da aplicação da abordagem é o mapeamento das características interessadas para os respectivos elementos de código (classes e métodos) que as implementam, os grafos de chamadas dos métodos de interesse, a classificação das classes conforme a sua participação na implementação do conjunto de características e um mapeamento com as classes classificadas por participação nas características. Conforme descrito anteriormente, estas visões foram montadas visando auxiliar o desenvolvedor na compreensão do código, ampliando a visão do analista sobre o funcionamento do sistema e reduzindo o espaço de busca pelos itens de interesse.

Para entender completamente o interesse, o desenvolvedor poderá precisar de mais informações sobre o sistema. Neste caso, a integração com o IDE Eclipse permite uma

busca por informação adicional guiada pelas visões obtidas. Os analistas como base nas informações (visões) geradas dinamicamente podem refinar as mesmas e até ampliar a informação através da análise estática, a partir do código fonte integrado ao mapeamento e de outras informações fornecidas pelas ferramentas disponíveis no IDE. Trabalhos relacionados [Simmons et al. 2006, Antoniol e Gueheneuc 2005, Eisenbarth et al. 2003] relataram a importância do uso da análise dinâmica conjuntamente com a análise estática para uma melhoria da compreensão e abrangência da informação. Eisenbarth e colegas realizaram estudos de caso [Eisenbarth et al. 2003, Eisenbarth et al. 2001] e concluíram que a combinação de análise dinâmica refinada pela análise estática reduz o espaço de busca drasticamente.

## 2.5 Resumo da Abordagem

Resumindo, neste cápitulo foi apresentada a abordagem proposta, os passos necessários para execução desta e as visões geradas a partir dos rastros de execução, assim como as pretenções com o uso desta. A autora desta dissertação avaliou vários trabalhos onde foram apresentadas abordagens para localização de características, utilizando análise dinâmica e apresentando visões que auxiliassem na compreensão de sistemas, e constatou que a abordagem proposta apresenta diversos pontos em comum com as propostas em trabalhos relacionados, que serão discutidos no Capítulo 5. Porém, não foi encontrada nenhuma abordagem com o mesmo conjunto de visões aqui apresentado, constituindo assim um dos diferenciais deste trabalho. Para verificar a aplicabilidade da abordagem e se esta atinge o objetivo apresentado na Seção 1.2 , serão realizados estudos de caso e experimentos controlados, que serão descritos no próximo capítulo.

# Capítulo 3

# Descrição do Estudo

A avaliação que será apresentada nas próximas seções foi elaborada e executada a fim de verificar se a abordagem atingiu os objetivos apresentados no Capítulo 1. Foi realizado um estudo preliminar e quatro experimentos envolvendo manutenções em diferentes sistemas, estes estudos foram realizados por grupos de sujeitos desenvolvedores, não envolvidos com o projeto, que participaram voluntariamente e foram acompanhados, treinados e questionados pela experimentadora deste trabalho. Dados quantitativos e qualitativos foram coletados para avaliação dos resultados que serão apresentados.

Para melhor compreensão de como foi realizado este estudo, as próximas seções irão apresentar detalhes sobre a metodologia de avaliação adotada, as perguntas e hipóteses que fundamentam este trabalho, os estudos preliminares, os experimentos realizados e resultados.

## 3.1 Metodologia de Avaliação

Diante dos objetivos deste estudo e a variedade de fatores que afetam os resultados avaliados durante as atividades de manutenção de programa, considerou-se experimentos controlados, como um método de pesquisa mais adequado para a explicação de um fenômeno que envolve um grande número de fatores dos quais apenas uma quantidade limitada pode ser controlada. Por exemplo, o sucesso de uma atividade de manutenção pode ser influenciado pela habilidade e capacidade de um desenvolvedor, pela competência em técnicas específicas, ou até mesmo, pela motivação do desenvolvedor para ter sucesso na atividade, a hora do dia quando a tarefa é realizada, o número de pausas tomadas, a presença ou ausência de distrações do ambiente, etc. Todos estes fatores podem influenciar nos resultados da análise e compreensão de um sistema e podem impactar nos resultados da validação, entretanto os mesmos são difíceis de serem controlados.

Considerando estes fatores, decidiu-se realizar um estudo de caso para verificar se a abordagem possui a precisão e acurácia minimamente adequados para a realização dos experimentos controlados, por meio de duas replicações. Em seguida, foram realizados

experimentos envolvendo diferentes sistemas de código aberto (Tabela 3.3) e atividades, que serão executados por grupos de desenvolvedores nivelados por conhecimento. Para comparar o desempenho da abordagem, foram definidos três grupos, um grupo *Controle*, que não utiliza a abordagem, e dois grupos que irão utilizar a abordagem *Completa* e *Parcial*, detalhes sobre estes grupos serão apresentados em seções futuras.

Sobre experimentos com grupos de sujeitos humanos, Cornelissen e colegas, em [Cornelissen et al. 2008], avaliaram diversos artigos que tratam sobre a análise dinâmica para compreensão de sistemas e selecionarem 176 trabalhos para uma análise aprofundada. Neste trabalho, eles constataram que, no contexto de trabalhos relacionados ao desta dissertação, estudos de caso (geralmente, através de sistemas de código aberto) são as mais comuns na literatura e o envolvimento de seres humanos (externos ao grupo de pesquisa) são pouco frequentes. Conforme apresentado em [Cornelissen et al. 2008], no domínio da compreensão do programa, uma avaliação que envolve seres humanos normalmente procura medir aspectos como a utilidade e a usabilidade de uma ferramenta ou técnica na prática. A participação de seres humanos nas avaliações é importante para o campo de compreensão de sistemas, pois este campo tem a tarefa de transmitir informação para o homem. Apesar de sua importância, este tipo de avaliação foi utilizada em não mais que 6 artigos dos 176 avaliados. Em uma nota positiva, o fato de 3 dos 6 experimentos mencionados acima terem sidos realizados em 2008 pode sugerir que este tipo de avaliação vem se tornando necessária. Logo, estudos envolvendo sujeitos humanos são importantes para a avaliação de abordagens como esta que se propõe a tratar o problema da compreensão de sistemas. Por este motivo, mesmo com as dificuldades naturais deste tipo de avaliação, optou-se por realizar os experimentos deste tipo, supondo que estes podem trazer informações mais precisas e observações valiosas para a pesquisa do uso de abordagens dinâmicas para localização de características, que visam auxiliar na compreensão de sistemas.

Os estudos e experimentos foram elaborados com a finalidade de verificar os objetivos deste trabalho e pretendem responder às perguntas e hipóteses que serão apresentadas nas próximas seções. As perguntas de pesquisa respondem às proposições teóricas concebidas a partir de hipóteses. Os resultados dos experimentos devem fornecer dados para responder às perguntas. Assim, foram definidas algumas premissas e variáveis independentes e dependentes que devem fornecer os dados adequados. Para apresentação deste trabalho dentro da metodologia proposta, serão apresentadas as hipóteses e perguntas, descritos os cenário do estudo de caso e experimentos, seus resultados individuais, os resultados gerais, discutidos os fatores relevantes que afetam a validade dos experimentos e, finalmente, resumidos os resultados sob a forma de resposta para cada pergunta de pesquisa.

## 3.2 Hipóteses

Abaixo seguem as hipóteses que deverão ser verificadas com os resultados dos experimentos.

H1) Sujeitos que utilizam a abordagem apresentam melhor tempo de execução em atividades de manutenção sobre sistemas desconhecidos do que sujeitos utilizando uma abordagem tradicional.

*Argumentação:* Quando a abordagem é utilizada sobre sistemas desconhecidos, a localização do interesse potencialmente seria mais rápida do que quando não utilizada e, consequentemente, a resposta a uma atividade ligada à compreensão do sistema que requeresse a localização do interesse, também, seria mais rápida. A localização de características utilizando análise dinâmica reduz e organiza o espaço de busca do sujeito pela característica de interesse, que geralmente se encontra dispersa no código fonte.

H2) Sujeitos que utilizam a abordagem apresentam maior correção nas respostas durante a execução de atividades de manutenção em sistemas desconhecidos do que sujeitos que utilizam a abordagem tradicional.

*Argumentação:* As visões desta abordagem auxiliam o desenvolvedor, pois provêem informações dinâmicas para a compreensão do sistema, tais como: as classes e métodos executados para uma característica de interesse, a presença de elementos de código em comum entre características de interesse e as chamadas entre os métodos executados para as características. A não utilização da abordagem implica em navegação em informações estáticas, que apesar de ser potencialmente útil, ocorre sem necessariamente a aplicação de nenhum filtro. Porém, com o uso da abordagem apresentada, ocorre um direcionamento para informações das características de interesse potencialmente induzindo respostas mais precisas.

Estas hipóteses foram formuladas com a intenção de responder ao objetivo de auxiliar na compreensão de sistemas. Para verificar estas hipóteses, serão formuladas perguntas cujas respostas serão obtidas por informações qualitativas e quantitativas dos resultados apresentados nos experimentos.

Seguindo a metodologia definida, para verificar se os objetivos deste trabalho foram atingidos e verificar a validade das hipóteses anteriores, foram definidas as perguntas que devem ser respondidas com os dados experimentais:

- **Sobre o tempo de execução de uma atividade:**

  P1) O uso da abordagem torna a resposta à compreensão de sistema desconhecidos mais rápida, ao se comparar com um procedimento tradicional de compreensão do sistema?

- **Sobre a taxa de acerto na execução de uma atividade:**

P2) Utilizando a abordagem, as respostas dadas pelo sujeito para uma atividade de manutenção em um sistema desconhecido são menos erradas?

- **Sobre a utilidade das informações fornecidas pela abordagem:**

  P3) As informações fornecidas nas visões da abordagem foram úteis para completar as atividades solicitadas nos experimentos?

- **Sobre o grau de dificuldade percebido na execução da atividade:**

  P4) Os grupos apresentaram variação expressiva quanto ao grau de dificuldade atribuído à atividade? Qual grupo atribuiu maior grau de dificuldade às atividades?

- **Sobre a influência do espaço de busca:**

  P5) Quando ocorre redução do espaço de busca inicial, também, ocorre redução no tempo de resposta e aumento na taxa de acerto do sujeito em atividades de manutenção em sistemas desconhecidos?

  P6) A presença de elementos de código falsos positivos e a ausência de elementos de interesse (falsos negativos) nas visões interferem no desempenho, taxa de acerto e tempo de resposta, do sujeito que utiliza a abordagem?

## 3.3 Estudo Preliminar

Um estudo preliminar foi definido para avaliação da precisão e acurácia da abordagem e coleta de *feedback* qualitativo diante a satisfação do usuário. Estes dados avaliados possuem a finalidade de proporcionar maior segurança para os experimentos controlados fornecendo informações sobre possíveis ajustes na abordagem, que poderiam ser feitos antes da execução dos mesmos. As seguintes perguntas foram definidas para verificar o objetivo deste estudo:

- a abordagem apresentou a precisão e acurácia esperada para os estudos?
- esta deve ser alterada para melhoria da precisão e acurácia?
- esta deve ser alterada para melhoria da satisfação do usuário?

Para este estudo foram realizadas duas replicações, os sistemas avaliados foram: *Jogos de Tabuleiro* e *Cotação de Bolsa*, estes são sistemas pequenos e desenvolvidos pelos sujeitos que avaliaram os resultados, que serão apresentados a seguir.

Para análise dos resultados foram definidas variáveis controladas, sendo uma variável independente e duas variáveis dependentes, que serão base para os dados apresentados pelo estudo preliminar. Porém, estudos como este, que envolvem sujeitos humanos e sistemas desconhecidos, apresentam variáveis não controladas pelo experimentador que

podem ocorrer e afetar os resultados controlados. Algumas destas variáveis são dificilmente percebidas ou medidas, mas devem ser consideradas. A seguir serão apresentadas estas variáveis.

Como variável independente deste estudo, tem-se:

- Os relacionamentos entre os elementos de código fonte e as características apresentadas nas visões da abordagem. O controle desta variável será dado pelo próprio usuário após a execução da abordagem. Esta variável será medida para responder as perguntas que este estudo pretende responder.

As variáveis dependentes sofrem os efeitos da variável independente (controlada) e deve ajudar a responder as perguntas da pesquisa. Para este experimento, as variáveis dependentes são:

- precisão e acurácia;
- a satisfação subjetiva do usuário.

As primeiras variáveis, precisão e acurácia, foram medidas após a execução do experimento, onde o sujeito que desenvolveu o sistema alvo apontou os elementos FP, FN, VP e VN (ver Tabela 3.1), e com base nestes dados foi realizado o cálculo destas variáveis.

Tabela 3.1: Definição dos tipos de elementos de código segundo a veracidade da sua presença nas visões.

| Veracidade do Elemento | Definição |
|---|---|
| VP = verdadeiro positivo | elementos de código que aparecem nas visões, e que, segundo o sujeito avaliador, este implementa a característica. |
| VN = verdadeiro negativo | elementos de código que não aparecem na abordagem, e que, segundo o sujeito avaliador, este não implementa a característica. |
| FN = falso negativo | elementos de código que não aparecem na abordagem, e que, segundo o sujeito avaliador, este implementa a característica. |
| FP = falso positivo | elementos de código que aparecem na abordagem, e que, segundo o sujeito avaliador, este não implementa a característica. |

A variável satisfação foi medida através de um questionário aplicado após a atividade, visando coletar a satisfação subjetiva do sujeito que participou do experimento. Esta satisfação é percebida por meio das respostas dadas pelos sujeitos para as perguntas referentes ao uso da abordagem, que foram úteis para a definição das melhorias no ferramental, ambiente e processo que contribuíram para os experimentos futuros.

Várias variáveis não puderam ser controladas, tais como:

- familiaridade do participante com o ferramental da abordagem e ambiente utilizado, uma vez que foi realizado um breve treinamento sobre ferramental e contato com o ambiente utilizado, onde o aprendizado e adaptação pode variar de participante para participante e interferir nos resultados;
- efeitos experimentador, uma vez que o experimentador está envolvido com a proposta e tem interesse que esta seja bem sucedida, este pode influenciar nos experimentos ao instruir os participantes, escolher as perguntas realizadas, selecionar os experimentos etc.;
- disposição do participante para o experimento, já que o desempenho do participante durante uma atividade é influenciada pela sua disposição no momento. Este pode estar disposto e animado ou cansado e chateado e isto pode interferir em sua concentração e capacidade de compreensão;
- ruídos gerados por roteiros mal definidos ou por ocorrência de eventos invisíveis a observação do usuário. Os elementos falsos positivos e negativos são naturais na análise dinâmica para localização de características. A existência de elementos não observáveis ao usuário depende unicamente do sistema, e a perfeita definição dos roteiros depende da experiência do sujeito com o uso da abordagem, sistema alvo e como desenvolvedor;
- subjetividade da definição sobre quais elementos de código são relevantes para a implementação das características de interesse, sob o ponto de vista do participante. Uma vez que os participantes podem considerar diferentes elementos relevantes para a característica, variando conforme a solução e compreensão apresentada sobre o sistema alvo.

### 3.3.1 Replicação 1 - Jogos de Tabuleiro

A primeira replicação deste estudo foi realizada sobre o Sistema *Jogos de Tabuleiro* (Figura 3.1) que é um sistema pequeno e simples, possui 2 pacotes, 16 classes e 72 métodos, cujas características principais são os dois jogos: *Jogo da Velha* e *Jogo Connect*. Ele possui características auxiliares como a escolha da quantidade de jogadores (*single player* ou *multi player*), contagem de jogadas, contagem de jogadas ganhas por usuário, retornar ao menu principal e selecionar nova partida. Este sistema foi desenvolvido por um aluno que participou do programa de iniciação científica da Faculdade de Computação desta universidade, que foi classificado como sujeito iniciante, segundo os critérios da Tabela 3.4. A possibilidade de contar com o sujeito que desenvolveu o sistema na avaliação garante uma verificação confiável das variáveis independentes definidas para o estudo.

Para a execução desta replicação, o desenvolvedor do sistema utilizou a abordagem com todas as visões disponíveis sobre as características de interesse escolhidas:


Figura 3.1: Tela inicial do sistema *Jogos de Tabuleiro*.

- iniciar jogo da velha com único jogador;
- iniciar jogo da velha com dois jogadores;
- iniciar jogo connect com único jogador;
- iniciar jogo connect com dois jogadores;
- selecionar novo jogo;
- executar uma jogada no jogo da velha com dois jogadores;
- voltar para o menu principal;
- contar vitórias por jogador;
- contar partidas completadas.

O sujeito participante montou o roteiro de cada característica e utilizou o *TraceExtractor* para obter os rastros de execução a partir destes roteiros. Logo depois, utilizou a *TraceToConcern*, *ConcernMapper*, *TraceToVisions* e *Graphviz* para gerar as visões: Mapeamento, Classificação das Classes, Mapeamento com Classificação e o Grafo de Chamadas, como apresentado Capítulo 2. Para a geração do grafo, foi selecionado apenas um método de interesse: *jogosDeTabuleiro.Tabuleiro.getElementoNaPosição*. Após obter os resultados, o desenvolvedor analisou os mesmos verificando a precisão dos resultados gerados pela abordagem, ou seja, os valores de VP, VN, FN e FP. Após esta análise, o desenvolvedor foi instruído a responder um questionário com perguntas sobre os resultados gerados pela abordagem e o uso desta.

### 3.3.2 Replicação 2 - Cotação de Bolsa

A segunda replicação do estudo de precisão foi realizada sobre o sistema *Cotação de Bolsa*, que é um sistema de pequeno porte desenvolvido para um trabalho de uma disciplina de graduação por um dos sujeitos colaboradores. Este sujeito foi classificado como Avançado, segundo critérios da Tabela 3.4. Este sistema possui 8 pacotes, 22 classes e 114 métodos. Este sistema cadastra as siglas das ações da bolsa de valores e realiza consultas na Bovespa para cotar o valor destas em tempo real. Este, também, possibilita o cadastro dos investimentos e a consulta dos valores de venda em tempo real.

As características do sistema analisadas neste experimento foram:

- abrir ajuda: abre uma breve descrição do sistema;
- adicionar nova ação: adiciona uma sigla de ação para consulta em tempo real;
- adicionar novo investimento: adiciona um novo investimento do usuário;
- carregar ações: apresenta as ações cadastradas no banco de dados;
- carregar investimentos: apresenta os investimentos cadastrados no banco de dados;
- carregar cotações: apresenta os valores das ações, estes valores são obtidos em tempo real através de conexão com a Bovespa;
- excluir ação: exclui uma ação previamente cadastrada no banco de dados;
- fechar app: fecha a aplicação finalizando o banco de dados e as conexões.

Nesta replicação foram realizados os mesmos procedimentos da replicação anterior. Esta foi realizada porque diante do impacto das variáveis não controladas duas replicações fornecem resultados mais confiáveis do que apenas uma. Isso possibilita a observação de dois sistemas distintos e sujeitos com diferentes níveis de conhecimento.

### 3.3.3 Resultados

Os resultados do estudo preliminar são apresentados, contendo os resultados das variáveis dependentes definidas: precisão e acurácia. A Tabela 3.2 apresenta os resultados destas variáveis, obedecendo às seguintes fórmulas:

- Precisão: $\frac{VP}{VP+FP}$
- Acurácia: $\frac{VP+VN}{P+N}$

onde, $\mathbf{P}=\mathbf{VP}+\mathbf{FN}$, elementos apresentados corretamente na abordagem e $\mathbf{N}=\mathbf{VN}+\mathbf{FP}$, elementos apresentados incorretamente na abordagem.

Conforme se pode notar os resultados para o sistema *Cotação de Bolsa* foram consideravelmente melhores do que os resultados para o sistema *Jogos de Tabuleiro*. *Cotação de bolsa* apresentou máxima precisão para as classes e métodos listados e máxima acurácia

Tabela 3.2: Cálculo da precisão e acurácia da abordagem nos estudos preliminares.

| Fórmulas | Jogos de Tabuleiro | Cotação de Bolsa |
|---|---|---|
| Precisão Classes | 0.91 | 1 |
| Precisão Métodos | 0.72 | 1 |
| Acurácia Classes | 0.91 | 0.97 |
| Acurácia Métodos | 0.69 | 1 |

para os métodos. A acurácia para as classes foi 0.97, um valor muito bom, que não foi máximo devido a uma classe que não foi listada (FN). Esta classe não possui métodos, e pela característica da abordagem não havia como ser listada. Para o sistema *Jogos de Tabuleiro*, os resultados foram piores. A precisão e acurácia para as classes apresentaram um bom valor, porém para os métodos este resultado não foi satisfatório pelo valor elevado de FP, causado pelo roteiro definido e por elementos que foram executados durante a coleta e não foram considerados relevantes para a característica. Diante dos valores apresentados, os resultados foram considerados satisfatórios.

Os participantes também preencheram um questionário que indaga sobre os pontos positivos do uso da abordagem e apresentaram satisfação com o uso desta. Mas, durante o estudo, os participantes apresentaram dificuldades e sugestões, que deram origem a algumas melhorias sobre a usabilidade do ferramental. Não foi realizada nenhuma melhoria relacionada aos FN e FP, porque a literatura relacionada indica que a presença deste tipo de elemento é natural na análise dinâmica e que dependem da boa escolha dos roteiros e entradas [Robillard 2003]. É possível implementar técnicas para reduzir ou minimizar os impactos dos FP e FN, que podem ser propostas para trabalhos futuros de melhoria desta abordagem. A alternativa adotada para mitigar o risco de números elevados de falsos negativos e positivos foi destacar nos treinamentos a necessidade da boa definição dos roteiros e entradas.

## 3.4 Projeto Experimental

Nesta seção serão apresentados experimentos que pretendem responder às perguntas de avaliação deste trabalho. Para isto, foram realizados quatro experimentos. No primeiro, os grupos de sujeitos implementaram uma melhoria com reuso de código no sistema *Jogos de Tabuleiro*. No segundo experimento, os grupos realizaram uma correção na funcionalidade *Undo* e *Redo* do sistema *JHotDraw* 7.1 [1], um *framework* de médio porte para criação de figuras. No terceiro, realizou-se uma atividade de localização dos elementos de código envolvidos na evolução das características *Undo* e *Redo* do *JHotDraw* nas versões 7.1 e 5.3. No último experimento, foi avaliado um sistema de grande porte, o *ArgoUML* 0.26.1,

[1] www.jhotdraw.org

e realizada uma atividade de melhoria nesta ferramenta de modelagem UML, fazendo com que o tamanho do objeto "comentário"e texto inserido neste sejam fixos.

Assim como nos estudos preliminares, os resultados são limitados pelas mesmas variáveis independentes, dependentes e não controladas, dos quatro experimentos, permitindo verificar os mesmos dados em todos e obter um número maior de observações sobre as variáveis dependentes, tornando os resultados mais confiáveis.

Para responder as perguntas P1 à P4, a variável independente é:

- a disponibilidade das visões da abordagem, variando conforme o grupo. Para verificar os ganhos da abordagem, três tipos de grupos foram montados, conforme apresentado na Seção 3.4.1. O diferencial destes é a visão apresentada, um grupo não utiliza a abordagem (grupo *Controle*), um grupo usa somente a visão de Mapeamento e um utiliza todas as visões. O uso e não uso das visões permitirá comparar o desempenho da abordagem.

Para a pergunta P1 a variável dependente é:

- o tempo necessário para executar a atividade de manutenção.

Para a pergunta P2 a variável dependente é:

- a taxa de acerto das soluções.

Para a pergunta P3 a variável dependente é:

- resposta do sujeito em relação à utilidade das informações fornecidas pela abordagem.

Para a pergunta P4 a variável dependente é:

- resposta do sujeito em relação à percepção da dificuldade na execução das atividades por parte dos participantes.

Para responder a pergunta P5:

- a variável independente é a redução do espaço de busca;
- a variável dependente é o tempo de execução da atividade e taxa de acerto nesta.

Para responder a pergunta P6:

- a variável independente é a ausência de elementos de código relevantes nas visões, os chamados Falsos Negativos;
- a variável dependente é o tempo de execução da atividade e taxa de acerto nesta.

As variáveis não controladas são as mesmas dos estudos preliminares, adicionado as seguintes variáveis:

- instrumentação, a maneira como será coletada as informações sobre as variáveis dependentes pode interferir nos resultados. Por exemplo, se fosse realizada de maneira automática a coleta do tempo de execução poderia ser ainda mais precisa do que coletada através de anotação;
- o nível de conhecimento dos participantes em desenvolvimento na linguagem Java e o uso do IDE Eclipse. Cada participante possui conhecimentos variados sobre desenvolvimento Java que podem o auxiliar na atividade e depende unicamente de sua experiência.

Nestes experimentos, os impactos das variáveis não controladas foram minimizados com a atribuição aleatória de participantes e nivelamento dos grupos. As variáveis ligadas ao experimentador e a instrumentação serão discutidos nas próximas seções.

Estas variáveis irão responder ou influenciar nas respostas às perguntas ligadas ao objetivo deste trabalho, e para garantir o controle e confiabilidade sobre estas, medidas foram tomadas para seleção dos participantes, definição dos grupos, escolha das atividades e sistemas avaliados, treinamento e instrução dos participantes, preparação da infraestrutura e procedimentos para execução dos experimentos. Estas ações visam minimizar os impactos das variáveis não controladas e serão apresentadas nas próximas seções.

### 3.4.1 Atividades de Manutenção e Sistemas Alvo

Os experimentos foram realizados sobre os seguintes sistemas: *Jogos de Tabuleiro*, *JHotDraw* e *ArgoUML*, que foram selecionados conforme as premissas da abordagem que é aplicada em sistemas desenvolvidos em Java, de código aberto. Outros critérios é que fossem bem documentados e versionados para se ter as informações necessárias para verificação dos resultados e definição das manutenções dos experimentos. Também foi critério de seleção a necessidade de executar os experimentos com sistemas de diferentes portes para verificar a robustez da abordagem. Portanto, selecionou-se um sistema pequeno (*Jogos de Tabuleiro*), um médio (*JHotDraw*) e um grande (*ArgoUML*), conforme se pode verificar pelos dados apresentados na Tabela 3.3.

Tabela 3.3: Sistemas utilizados nos experimentos e dados sobre o número de classes e métodos destes e o experimento em que foram utilizados.

| Sistema | Número de Classes | Número de Métodos | Manutenção |
|---|---|---|---|
| Jogos de Tabuleiro 1.0 | 16 | 72 | Reuso |
| JHotDraw 7.1 | 466 | 4078 | Correção e Localização |
| JHotDraw 5.3 | 215 | 1793 | Localização |
| ArgoUML 0.26.1 | 2.022 | 14.038 | Melhoria |

Para cada experimento foi definida uma atividade de manutenção diferente sobre um dos sistemas selecionados, visando verificar os resultados da abordagem para diferentes tipos de atividades e sistemas. Para definir atividades que pudessem ser resolvidas no tempo predefinido de execução das atividades, a experimentadora pesquisou o uso dos sistemas em suas documentações, e no caso do sistema *Jogos de Tabuleiro*, a atividade foi definida junto ao desenvolvedor do sistema. Considerando atividades de rotina de um desenvolvedor, foram escolhidos diferentes tipos de atividades, sendo estas: (i) reuso de código para implementar novos requisitos no sistema, (ii) correção de um erro, (iii) localização dos elementos de código que implementam as mesmas características em diferentes versões do sistema e apontamento da evolução que ocorreu entre as versões para estas características, e (iv) uma melhoria em uma funcionalidade existente. A seguir serão apresentadas estas atividades com detalhes.

### 3.4.2 Sujeitos

As atividades de manutenção dos quatro experimentos foram executadas por grupos de sujeitos desenvolvedores, a fim de observar o comportamento destes grupos durante as atividades e obter respostas para as perguntas formuladas. A formação destes grupos está diretamente ligada à variável independente deste experimento, que pretende verificar os ganhos com o uso da abordagem e da visão Mapeamento em relação às demais visões, considerando o desempenho (tempo e taxa de acerto) destes grupos e relacionando este resultado ao uso ou não da abordagem. E para isso estes foram divididos da seguinte maneira:

- **Grupo Controle:** Este não utiliza a abordagem proposta, mas pôde utilizar todas as funcionalidades do IDE Eclipse, disponíveis para executar a atividade do estudo de caso, em condições semelhantes aos demais grupos;
- **Grupo Parcial:** Este utilizou a abordagem parcial, somente utilizou a visão Mapeamento e as funcionalidades do IDE Eclipse;
- **Grupo Completa:** Este utilizou a abordagem completa, as 4 visões apresentadas e as funcionalidades do IDE Eclipse.

A seleção dos sujeitos, que participaram dos estudos de caso, foi realizada sobre uma lista de contatos pessoais e profissionais da experimentadora. Um total de 27 sujeitos desenvolvedores que voluntariamente participaram dos experimentos, sendo que 96.29% destes sujeitos são profissionais que trabalham com análise de sistemas e estão inseridos no mercado de trabalho e um sujeito é aluno do curso de Mestrado em Ciência da Computação - UFU. Sobre a formação, 78% são formados e os 22% cursam os últimos períodos da faculdade de Ciência da Computação. Ao todo participaram sujeitos de 8 (oito) diferentes empresas de Uberlândia-MG e de uma universidade (UFU). Estes responderam um

questionário (Anexo A) contendo perguntas sobre sua formação acadêmica e profissional, perguntas técnicas relativas à experiência com o IDE Eclipse, análise de sistemas desenvolvidos na linguagem Java e uma questão prática para medir o grau de conhecimento em desenvolvimento e manutenção de sistemas. Os voluntários também foram entrevistados e questionados sobre os conhecimentos específicos. Os principais requisitos para seleção e formação dos grupos foram: conhecimento em desenvolvimento Java e experiência em manutenção de programas. Estes sujeitos foram classificados conforme o grau de conhecimento, como: iniciante, intermediário ou avançado, seguindo os quesitos da Tabela 3.4.

Tabela 3.4: Critério de classificação dos sujeitos.

| Conhecimento Técnico | Critério de Classificação |
|---|---|
| Iniciante | Sujeito quando questionado e no relatório de avaliação demonstrou ter pouca experiência em desenvolvimento na linguagem Java. |
| Intermediário | Sujeito quando questionado e no relatório de avaliação demonstrou ter um bom conhecimento e experiência em desenvolvimento na linguagem Java e uso do IDE Eclipse. |
| Avançado | Sujeito quando questionado e no relatório de avaliação demonstrou possuir larga experiência e conhecimento em desenvolvimento na linguagem Java e uso do IDE Eclipse. |

Cada grupo montado é formado por 3 sujeitos, sendo um iniciante, um intermediário e um avançado, ou seja, 9 sujeitos participaram de cada experimento. A homogeneidade de conhecimento dos grupos é um fator importante, pois pode influenciar nos resultados. Isto foi levado em consideração na seleção dos integrantes dos grupos (*Controle, Parcial* ou *Completa*), onde para cada experimento, esta seleção foi realizada de maneira aleatória, levando em consideração a disponibilidade do sujeito na data do experimento e sua classificação.

Ao final deste processo, obteve-se grupos qualificados e homogêneos para a execução dos quatro experimentos controlados, minimizando parte dos impactos gerados pelas variáveis ligadas aos sujeitos participantes como conhecimento, adaptação e disposição.

### Experimento 1 - Reuso

Foi avaliado neste experimento o sistema *Jogos de Tabuleiro*, cujas características foram descritas anteriormente nos estudos preliminares. A atividade que deve ser desempenhada envolve as características: Contar Partidas, Contar Vitórias, Menu Principal e Novo Jogo. Estas características estão presentes no menu lateral do *Jogo Connect*, como se pode ver na Figura 3.2, e devem ser reutilizadas para que o *Jogo da Velha* tenha este mesmo menu lateral. O *Jogo da Velha* originalmente possui o referido menu lateral, mas


Figura 3.2: Telas dos *Jogos Connect* e *Jogo da Velha* com o menu lateral.

para este experimento o menu lateral e as partes do código que remetiam ao reuso foram removidos. Os sujeitos foram instruídos a utilizar as melhores práticas de reuso de código para desenvolvimento do menu lateral com as mesmas características do menu presente no *Jogo Connect*, segue a descrição das características de interesse:

- *Contar Partidas:* Esta funcionalidade conta o número de partidas realizadas pelos jogadores e disponibiliza este valor no menu lateral do jogo;
- *Contar Vitórias:* conta o número de vitórias de cada jogador no conjunto de partidas realizadas, e disponibiliza este valor no menu lateral do jogo;
- *Menu Principal:* se caracteriza por um botão que possibilita voltar ao menu principal (tela inicial do sistema - Figura 3.2) dos *Jogos de Tabuleiro*;
- *Novo Jogo:* reinicia o jogo e atribui o valor zero para todos os contadores.

Esta atividade foi desempenhada pelos 3 grupos (*Controle, Parcial* e *Completa*) e seguiram procedimentos definidos na Seção 3.4.3, que apresenta os detalhes de execução dos experimentos. A seguir serão apresentadas as atividades de manutenção dos demais experimentos.

### Experimento 2 - Correção

O sistema *JHotDraw* é um *framework* em Java, com código aberto, que permite traçar gráficos bi-dimensionais (2D). Este foi baseado no *JHotDraw* de Erich Gamma, que foi registrado em 1996, 1997, pela IFA - *Informatik and Erich Gamma*. O projeto começou em outubro de 2000 e ainda está ativo. Este é um sistema considerado maduro, possui suporte e se encontra em constantes melhorias, com uma ampla rede de utilizadores. Por possuir boa documentação de apoio, repositório de versões[2] bem estruturado, suporte técnico ativo, lista de erros, e outras informações de fácil acesso, o *JHotDraw* foi considerado

[2]http://jhotdraw.svn.sourceforge.net

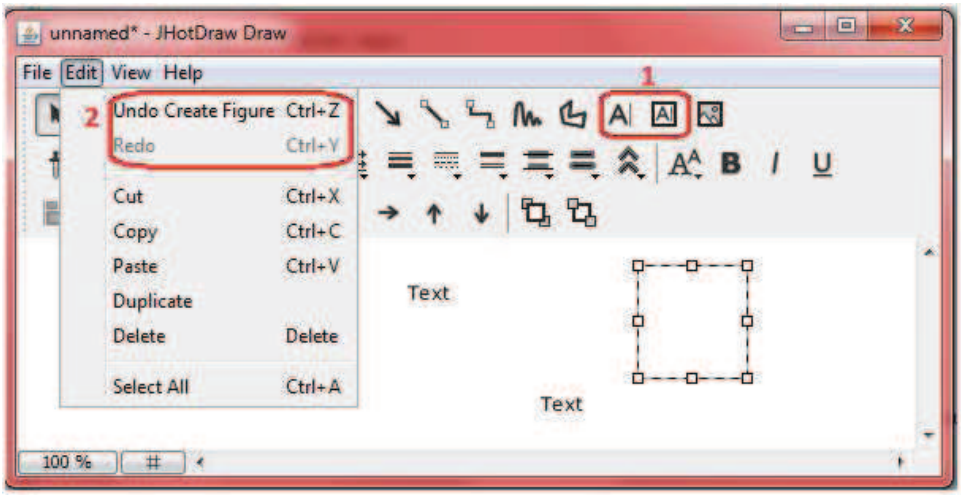



Figura 3.3: (1) Ícones para as características *TextTool* e (2) botões para as características *Undo* e *Redo* no *JHotDraw* 7.1.

um candidato ideal para os experimentos propostos. Estes dados facilitam a definição do experimento e verificação dos resultados para a atividade de manutenção proposta, por apresentarem informações importantes sobre o desenvolvimento do sistema.

Para este experimento foi selecionado um erro presente a versão 7.1, apresentado na lista de correção de erros no site oficial do *JHotDraw*, onde inclusive a correção foi postada. A reprodução deste erro é apresentada nesta lista e foi de grande utilidade para a definição do erro a ser escolhido, assim como a solução, pois a atividade não pode ultrapassar o tempo limite do experimento e deve ser de fácil reprodução, pois os sujeitos participantes desconhecem o sistema. O erro selecionado foi denominado *Text Area Tabs* e é referente ao uso das funcionalidades *TextTool* (número 1 da Figura 3.3), ferramentas de texto, que permite criar e modificar novos objetos do tipo textos dentro e fora de figuras, que quando executado apresenta erros nas características *Undo* e *Redo* (número 2 da Figura 3.3). O *Undo* é uma característica do *JHotDraw* que desfaz as alterações executadas sobre os objetos construídos, e a função do *Redo* é refazer o que foi desfeito pela funcionalidade *Undo*. O erro é que, para as ferramentas de texto, estas duas características não são disparadas.

Como atividade de manutenção para este experimento, foi solicitada a correção deste erro, fazendo com que os objetos do tipo texto do *JHotDraw* 7.1 possam sofrer as ações das características *Undo* e *Redo* perfeitamente, da mesma maneira que os demais objetos de desenho. A experimentadora deixou claro para os sujeitos participantes que esta é uma atividade de correção, onde elementos de código devem ser corrigidos e não constitui uma nova funcionalidade a ser implementada.

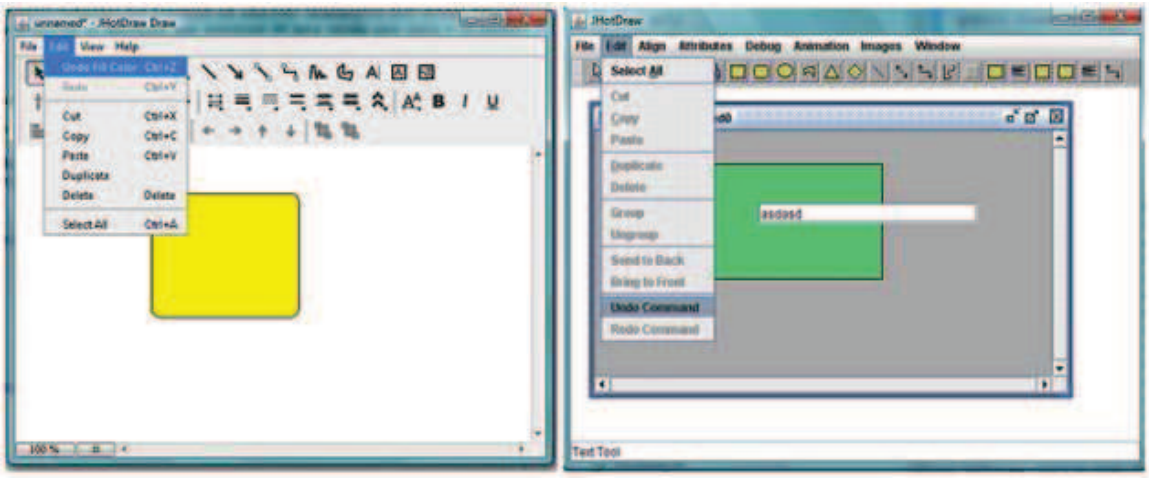


Figura 3.4: Exemplo de uso do *Undo* e *Redo* na versão 7.1 e na versão 5.3 do *JHotDraw*, respectivamente.

### Experimento 3 - Localização de Evolução

No terceiro experimento, o sistema alvo é novamente o *JHotDraw* nas versões 5.3 e 7.1. Este sistema sofreu substanciais alterações em sua estrutura na versão 7.1 [3]. Estas duas versões são visivelmente diferentes, ver Figura 3.4, e as características *Undo* e *Redo*, apresentadas na última seção, estão na lista das que sofreram maiores alterações, com novas classes e métodos incorporados, elementos de código excluídos e a adoção da biblioteca Swing para implementá-las.

Para este experimento foi solicitado aos participantes: 1) a análise das duas versões deste sistema para listagem de todas as classes e métodos envolvidos na implementação das características *Undo* e *Redo* em cada versão; 2) a descrição da evolução que estas características sofreram de uma versão para a outra.

### Experimento 4 - Melhoria

No quarto e último experimento, foi avaliado o sistema *ArgoUML*, uma ferramenta de modelagem UML que possui suporte para os diagramas no padrão 1.4. Este sistema é essencialmente escrita em Java, podendo assim ser rodada em qualquer plataforma compatível. Em suas versões mais recentes, está disponível em 10 línguas e dentre suas características estão funcionalidades de suporte cognitivo, tais como: geração de críticas de projeto, automação de correções, lista de tarefas/verificação e utilização de um modelo de usuário.

O projeto surgiu de uma iniciativa de Jason Robbins na Universidade da California em Irvine (UCI) em criar um ambiente de desenvolvimento com ênfase em fatores humanos. Como iniciativa de pesquisa, o sistema recebeu a colaboração de diversos pesquisadores e desenvolvedores. O *ArgoUML* possui extensa documentação de usuário e de projeto,

[3] estas alterações podem ser verificadas na pagina http://www.randelshofer.ch/oop/jhotdraw/

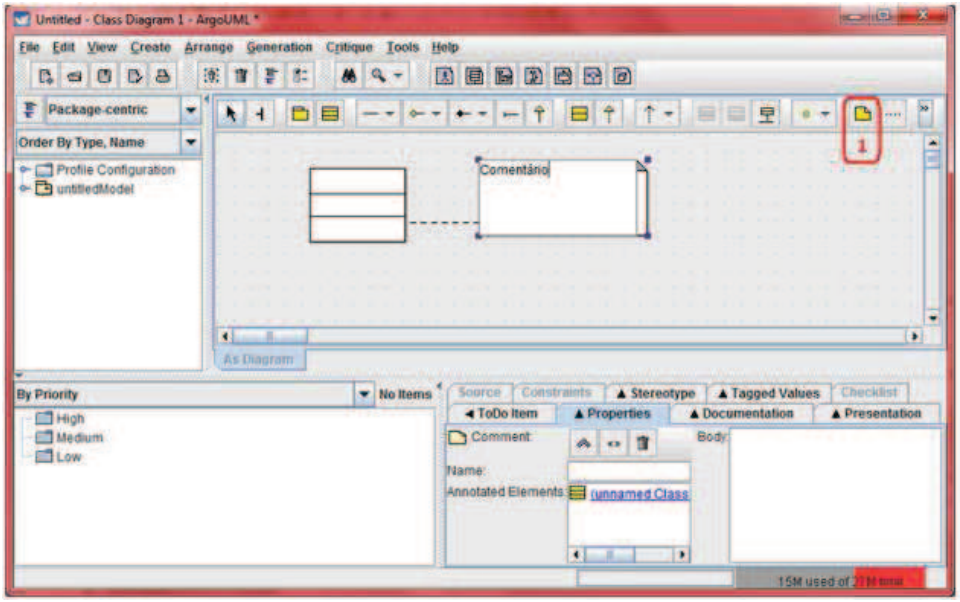



Figura 3.5: Ferramenta *ArgoUML*, exemplo de uso do objeto *Comentário* (1).

especialmente pela colaboração da comunidade e dos desenvolvedores. Além disso, o código fonte é aberto sobre a licença BSD. Atualmente, a principal fonte de acesso ao projeto é o *website* da *Tigris.org*, uma comunidade de código fonte aberto focada na produção de ferramentas para o desenvolvimento colaborativo de sistemas. Dessa maneira, a reprodução e validação dos experimentos realizados tornam-se mais fáceis para outros pesquisadores.

O *ArgoUML* atende aos pré-requisitos como sistema para este trabalho, e por ser um sistema relativamente de grande porte, foi apropriado para o último experimento. Além do mais, o tamanho do sistema e sua organização influenciam na compreensão do mesmo e será importante observar os resultados em um projeto deste porte.

Para a execução deste experimento, definiu-se uma atividade de manutenção que envolve o objeto *Comentário* (número 1 da Figura 3.5) dos diagramas UML. O *Comentário*, quando inserido em qualquer um dos diagramas UML desta ferramenta, apresenta um tamanho definido que aumenta em largura e altura conforme é inserido um texto neste. Este tamanho se adapta ao tamanho do texto, logo, se for inserido um texto sem quebras de linha e muito extenso, este pode ocupar uma grande área no diagrama. A ferramenta não permite que este objeto seja reduzido, enquanto o texto não for reorganizado. A melhoria solicitada é que o objeto *Comentário* tenha tamanho máximo fixo em altura e largura, e o texto inserido neste deve automaticamente adaptar-se ao tamanho do objeto e possuir um tamanho máximo de 160 caracteres.

Nesta seção foram apresentados os sistemas alvo e atividades de manutenção executadas durante os experimentos. Estes foram definidos visando a obtenção de uma visão ampla sobre a aplicabilidade da abordagem, sobre sistemas e atividades diferentes, que permitam a obtenção de dados quantitativos e qualitativos e respondam as perguntas

ligadas às hipóteses deste trabalho. Para execução controlada destas quatro atividades, foram definidos alguns procedimentos que serão descritos na próxima seção.

### 3.4.3 Procedimentos Experimentais

Para execução dos experimentos de maneira controlada e visando minimizar o impacto de algumas das varáveis não controladas, alguns procedimentos foram definidos. A experimentadora, neste processo, treinou e instruiu os participantes, preparou o laboratório, verificou as soluções e aplicou um questionário quantitativo e qualitativo, buscando garantir os mesmos procedimentos em todos os experimentos, estes serão descritos a seguir.

#### Treinar e Instruir os Participantes

Com o objetivo de preparar os participantes dos grupos *Parcial* e *Completa*, foram realizados treinamentos sobre a abordagem, para que estes fossem aptos a utilizar a abordagem de forma eficaz na investigação do sistema alvo. Os treinamentos foram realizados pela experimentadora, que instruiu os participantes até o aprendizado e uso correto da abordagem. O grupo *Controle* não recebeu este treinamento, pois não utilizou a abordagem. Neste treinamento, foi apresentado o ferramental, procedimentos, aplicabilidades e lógica da abordagem por meio de uma apresentação teórica, que durou em média uma hora. Também, foi apresentado um exemplo de uso da abordagem, com demonstração de utilização das ferramentas da abordagem e sua integração com o IDE Eclipse. Para garantir o máximo de uniformidade nos treinamentos, foi elaborado um material que foi seguido em todas as seções.

Em um segundo momento, antes de iniciar as atividades de manutenção dos experimentos, os participantes foram instruídos a utilizarem a abordagem para se adaptarem e sanarem dúvidas. Este momento durou em média quarenta minutos.

Em um terceiro momento, ainda antes da execução das atividades manutenção, os participantes foram instruídos sobre a atividade e sistema alvo e receberam um material descritivo sobre a atividade (ver Anexos B), que foi lido e explicado detalhadamente pela experimentadora. Esta instrução foi passada para os três grupos, visando garantir o máximo de normalidade entre estes, e durou em média 10 minutos. Os participantes puderam esclarecer dúvidas, porém, somente as que fossem pertinentes ao entendimento da atividade e ao uso da abordagem. Sobre o material, este continha um questionário sobre o sujeito participante, sobre a infraestrutura e ambiente de trabalho fornecido, descrição da atividade a ser executada e perguntas que irão quantificar e qualificar este trabalho. Para os grupos que utilizaram a abordagem foram propostos roteiros de execução para coleta dos rastros das características de interesse. Os participantes foram instruídos a utilizar roteiros predefinidos nos manuais, porém, com a liberdade de adotar ou não estes roteiros, podendo coletar os rastros da maneira que pudessem ajudar na compreensão.

Todos os grupos foram instruídos igualmente sobre a atividade a ser desempenhada e sobre o ambiente de desenvolvimento disponibilizado, visando diminuir os desvios, falhas e dúvidas durante a execução dos experimentos. Os treinamentos e materiais de instrução constituem uma etapa importante dos experimentos, pois garantem o mínimo de conhecimento necessário aos participantes. Esta etapa, se bem executada, minimiza muitos desvios que podem ocorrer pela falta de conhecimento sobre a abordagem e ambiente utilizados. Foram gastos, em média, uma hora e cinquenta minutos (1h 50min) por cada participante dos grupos *Parcial* e *Completa* para os treinamentos e instruções. Este foi considerado o tempo mínimo necessário para esta etapa. Porém, quanto mais robusto for este treinamento, menores seriam os impactos causados pela falta de conhecimento.

### Infraestrutura para os Experimentos

Sobre a infraestrutura montada, os experimentos foram realizados nos laboratórios da Faculdade de Computação da UFU - Universidade Federal de Uberlândia, onde as máquinas foram previamente configuradas para cada experimento, os sistemas utilizados foram instalados, a internet foi bloqueada e todas as máquinas foram testadas, garantindo assim que o ambiente disponibilizado estava em perfeito funcionamento e padronizado, visando diminuir os desvios nos resultados dos experimentos. Os 9 (nove) participantes dos 4 (quatro) experimentos, independentemente de grupo, utilizaram máquinas com esta mesma configuração.

As máquinas utilizadas são computadores de mesa, com processador Intel Celeron 420, 1G de memória RAM e 80Gb de memória em *HardDisk* (HD). Os sujeitos utilizaram o IDE Eclipse (*Ganymed)*, configurado igualmente para os três grupos, as funcionalidades deste IDE não foram bloqueadas e foi permitido o uso destas durante as atividades. Somente os grupos *Parcial* e *Completa* tiveram acesso ao ferramental da abordagem, que foi devidamente configurado e testado para cada experimento. As máquinas, IDE e ferramental da abordagem foram testados pela experimentadora e pelos sujeitos participantes, antes do experimento. No último experimento, realizado sobre a ferramenta *ArgoUML*, as máquinas do laboratório utilizado foram substituídas por equipamentos melhores. Computadores de mesa, processador Intel Quad Core, 2G de memória RAM e 320Gb de memória em HardDisk (HD).

O uso de máquinas iguais para cada experimento individual foi importante para minimizar a possível variação de desempenho dos sujeitos devido ao desempenho das máquinas utilizadas.

### Execução dos Experimentos

Os experimentos tiveram tempo fixo de duração, sendo que nos experimentos 1, 2 e 3 este foi de 4 horas e 50 minutos, utilizados com adaptação ao ambiente de trabalho,

utilizando o ferramental (Eclipse e ferramental da abordagem), e com as instruções sobre a atividade. Os participantes, independentemente do grupo, tiveram um prazo de três horas e dez minutos (3h 10min) para executar a atividade de manutenção para o qual foram selecionados. Para o experimento 4, devido a sua complexidade, o tempo dado para execução do experimento foi de três horas e trinta minutos (3h 30min) e o tempo para adaptar ao ambiente foi de 30 minutos. Este tempo de adaptação foi menor do que os demais experimentos devido ao tempo de 4 horas de experimento que foi definido e limitado pelo tempo de uso dos laboratórios. Porém, este tempo foi menor para todos os participantes deste experimento e, por isso, não gera impacto nos dados comparados. Os participantes, após instrução sobre a atividade a ser executada, tendo em mãos o material de instrução, iniciaram individualmente a análise dos sistemas alvo. Todos foram observados neste período, por isso, o número de sujeitos executando o experimento ao mesmo tempo foi limitado para 6 pessoas possibilitando um melhor acompanhamento por parte da experimentadora.

Os participantes, ao finalizar a atividade solicitada, apresentaram a solução para a experimentadora, que verificou se este atendeu aos requisitos da atividade. Como procedimento, foi adotado que quando a solução é considerada correta, a atividade é dada como concluída e o sujeito é instruído a responder o questionário presente no material. Quando a solução não é considerada correta, o sujeito retoma a atividade, caso o tempo não tenha se esgotado. Os tempos gastos com a execução da atividade e com o questionário foram registrados, separadamente. Cada participante do estudo de caso respondeu a um questionário, inclusive, os que não concluíram a atividade com sucesso. Os questionários do grupo *Controle* se diferem dos questionários dos grupos *Parcial* e *Completa*, devido às perguntas especificas, ligadas ao uso da abordagem e compreensão do código a partir desta. Estes questionários contemplaram perguntas sobre a compreensão do código, a solução dada para a atividade, dificuldades encontradas, o uso da abordagem e do IDE, a presença de elementos de código falsos negativos nas visões, usabilidade da abordagem, a integração com o IDE, sugestões e críticas. Os questionários são apresentadas no Anexo B.

Além dos questionários, a experimentadora acompanhou os grupos, verificou e analisou os resultados e anotou os comportamentos individuais e grupais, as dúvidas que surgiram, as dificuldades e facilidades sobre a compreensão do código e sobre o uso da abordagem. Todos estes pontos foram avaliados e são fontes de informação valiosa para as observações realizadas, que serão apresentadas para cada estudo no Capítulo 4.

Todos estes procedimentos adotados garantiram a execução controlada dos experimentos, minimizando os impactos causados por variáveis não controladas sobre as variáveis dependentes e independentes que quantificam e qualificam os resultados deste trabalho. Porém, mesmo com este controle, este tipo de experimento ainda sofre de ameaças a sua

validade que serão apresentadas a seguir, uma vez que estas medidas apenas minimizam os impactos ou possibilidade de ocorrência das ameaças.

### 3.4.4 Ameaças à Validade

Dada a configuração do experimento, é possível ter certeza de que qualquer efeito observado é causado apenas pela variação da variável independente, ou existem outras variáveis que podem ter influenciado o resultado? Diante desta questão, alguns fatores devem ser considerados e serão apresentados a seguir.

Alguns fatores internos podem influenciar nos resultados dos experimentos, sendo estes:

- diferenças individuais dos participantes: cada participante possui um conhecimento diferente em relação à atividade de programação e experiência em manutenção de programas. Considerando esta ameaça, foram atribuídos aleatoriamente os participantes dos grupos;
- número de participantes em cada grupo: cada grupo foi composto por 3 participantes, estes possuem habilidades distintas e pôde-se perceber que devido a este número pequeno de participantes por grupo as taxas de desvios nos resultados apresentados pelos participantes para alguns experimentos é alta, e isto pode impactar nos resultados individuais de cada experimento, porém, as conclusões serão tomadas sobre os resultados dos 4 experimentos, que fornece um dado mais confiável devido ao volume de 36 experimentos individuais, 12 experimentos individuais para o grupo *Controle* e 24 utilizando a abordagem, respondendo as mesmas perguntas;
- diferenças nos treinamentos e instruções: o treinamento e instruções que os diferentes grupos receberam podem ter sido diferentes em detalhes entre as sessões, embora tenha sido definido e seguido o mesmo procedimento e apresentação para todos os treinamentos;
- percepção dos participantes: sob o ponto de vista do participante existe uma subjetividade da definição sobre quais elementos de código são relevantes para a implementação das características de interesse, cada participante pode definir uma solução, que podem ser diferentes entre si, porém, corretas.

Há um grande número de fatores externos ao experimento que afetam a generalização dos resultados, como:

- representatividade do sistema alvo: a representatividade dos sistemas escolhidos para esta experiência é limitada. No entanto, os experimentos foram realizados sobre 4 sistemas distintos e de diferentes portes;

- familiaridade com o sistema: o envolvimento com um sistema completamente desconhecido não é a situação comum na manutenção de programas. Pois, normalmente, o sistema é bem conhecido para o mantenedor. Logo, os resultados não podem ser generalizados para tal situação;
- experiência com o uso da abordagem: a abordagem é um novo conceito para todos os participantes, eles tiveram que aprender e compreender este conceito dentro de um curto período de tempo antes de usá-los. Indivíduos com mais experiência no uso da abordagem podem apresentar um desempenho diferente;
- efeito do experimentador: o experimentador é a mesma pessoa que definiu a abordagem, e isso pode influenciar qualquer aspecto da experiência.

Todas estas ameaças devem ser consideradas na avaliação dos resultados, pois podem causar impactos sobre as variáveis dependentes deste trabalho. Considerando este conjunto de ameaças, que definiu-se os procedimentos de controle apresentados na Seção 3.4.3 e a execução de 4 experimentos controlados, contemplando um volume de 36 observações individuais. Decidiu-se que as conclusões serão apresentadas sobre o ponto de vista dos 4 experimentos para aumentar a confiabilidade dos resultados finais que serão apresentados.

## 3.5 Resumo da Descrição do Estudo

Neste capítulo foi apresentada a metodologia adotada para avaliação deste estudo, as perguntas e hipóteses que serão indagadas e discutidas para responder ao objetivo deste trabalho. Descreveu-se o estudo preliminar com seus resultados, e os experimentos realizados com detalhes sobre os sistemas utilizados, as variáveis trabalhadas para responder as perguntas, e os vários procedimentos tomados na execução experimental, visando garantir estudos controlados e resultados confiáveis. Foram apresentadas as ameaças a validade dos experimentos, que devem ser consideradas nos resultados. No próximo capítulo, os resultados dos experimentos controlados serão apresentados e discutidos.

# Capítulo 4

# Resultados e Discussões

Neste capítulo serão apresentados os resultados dos experimentos e as discussões realizadas sobre estes. A seguir serão apresentados os resultados de cada experimento, na forma de observações que apresentam os resultados obtidos. Estes serão utilizados na apresentação dos resultados gerais e conclusão.

## 4.1 Experimento 1 - Reuso

No primeiro experimento, cuja atividade a ser desempenhada pelos grupos já foi apresentada na Seção 3.4.2, tem-se como resultado que a maioria dos sujeitos desenvolvedores reutilizou os métodos do *Jogo Connect* referente às características de interesse, porém, as soluções variaram de sujeito para sujeito, estas puderam ser observadas pelas respostas do questionário. A experimentadora, ao validar as soluções de cada participante, tomou como corretas todas as soluções em que o requisito definido foi atingido, que seria reutilizar as classes e métodos do menu lateral do *Jogo Connect* para montar um menu lateral idêntico para o *Jogo da Velha*, em que todas as funcionalidades presentes fossem executadas conforme a definição apresentada. Segundo os procedimentos definidos, os participantes responderam aos questionários e foram observados, produzindo dados qualitativos e quantitativos que quando analisados levaram às observações descritas a seguir.

### 4.1.1 Observações sobre o Tempo

**Observação 1.1 -** *O desempenho em tempo dos grupos que utilizaram a abordagem neste experimento (Completa e Parcial) foi superior ao grupo Controle, que não utilizou a abordagem.*

Os grupos que utilizaram a abordagem tiveram uma média de tempo gasto inferior ao grupo *Controle* conforme se pode ver na Figura 4.1, o tempo médio gasto pelo grupo *Completa* foi de 95.66 minutos, do grupo *Parcial* foi 136 minutos e do grupo *Controle* foi 167 minutos. Este último teve variação em relação ao grupo que utilizou a abordagem

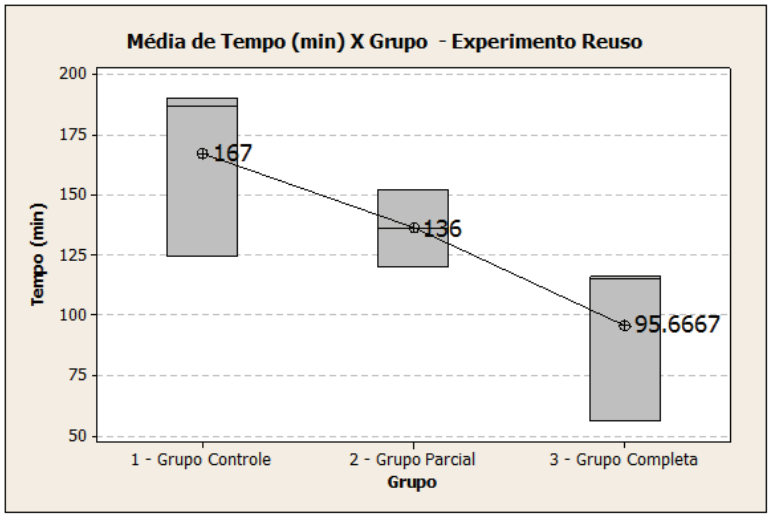



Figura 4.1: Gráfico com o tempo médio (em minutos) gastos pelos grupos no experimento 1 - Reuso.

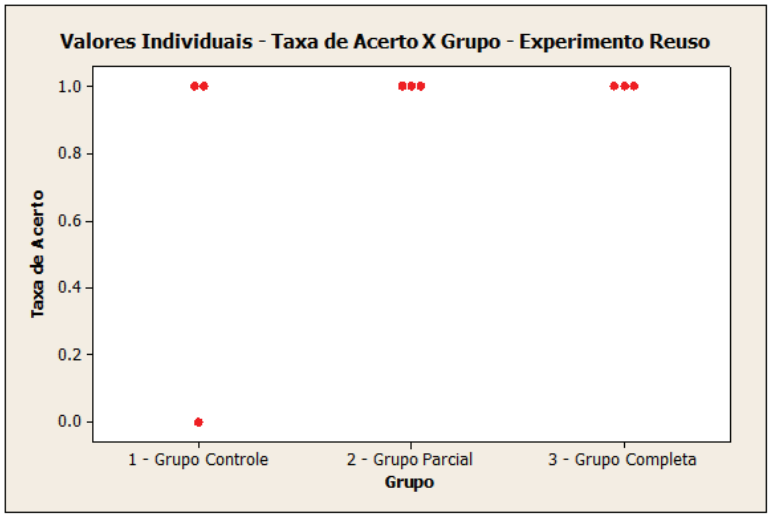



Figura 4.2: Gráfico com a taxa de acerto percentual por grupo no experimento 1 - Reuso.

completa de 71,34 minutos a mais na média, um valor considerável, 37,54% do tempo dado aos participantes para a atividade. Observe que o desvio para os grupos *Controle* e *Completa* nos permite concluir que o uso da abordagem completa foi significativamente superior.

### 4.1.2 Observações sobre a Taxa de Acerto

**Observação 2.1 -** *A taxa média de acerto dos grupos que utilizaram a abordagem neste experimento (Completa e Parcial) foi superior ao grupo Controle, que não utilizou a abordagem.*

Quanto à taxa de acerto, os resultados do primeiro experimento estão mostrados na Figura 4.2. Observa-se que todos os participantes dos grupos *Parcial* e *Completa* concluíram a atividade, e um dos três participantes do grupo *Controle* não conseguir concluir

a atividade com sucesso. Entretanto, dado o número relativamente pequeno de participantes no grupo *Controle*, seria possível que o participante que não concluiu a atividade significasse um ruído.

### 4.1.3 Observações sobre a Utilidade das Informações

**Observação 3.1 -** *As visões Mapeamento e Grafo de chamadas foram úteis na compreensão da característica.*

A visão Mapeamento foi considerada útil para esta atividade por todos os sujeitos que utilizaram a abordagem. A maioria dos participantes assim a consideram por que esta apresenta os elementos de código que implementam a característica de interesse, de maneira estruturada e integrada ao IDE, e ao avaliar esta visão, foi possível apontar os elementos que deveriam ser reutilizados.

Apenas o grupo *Completa* foi instruído para geração das demais visões (Grafo de Chamadas, Corte Funcional e Mapeamento com Classificação), e 66.66% dos participantes deste grupo apontaram que a visão Grafo de Chamadas como realmente útil na atividade. Segundo os participantes, o Grafo de Chamada foi útil, pois através deste foi possível descobrir as ligações entre os métodos que seriam alterados e visualizar as precedências de controle.

**Observação 3.2 -** *Os elementos de código (classes e métodos) apresentados nas visões foram relevantes para a compreensão da característica.*

Conforme o parecer de todos os participantes que utilizaram a abordagem, esta listou as classes e métodos que foram reutilizados na atividade e estes foram úteis para a compreensão das características de interesse, pois, auxiliaram na atividade, direcionando a investigação para os elementos de código ligados ao interesse.

**Observação 3.3 -** *Todos os participantes que utilizaram a abordagem apresentaram satisfação com o uso desta.*

Todos os participantes expressaram, no questionário, satisfação ao utilizar a abordagem e apontaram como esta os auxiliou na atividade e uma opinião positiva sobre a abordagem. Alguns sujeitos apontaram melhorias na abordagem que serão apresentadas na Seção 6.1, sobre trabalhos futuros.

### 4.1.4 Observações sobre a Dificuldade da Atividade

**Observação 4.1 -** *A atividade foi classificada como mediana a fácil e a maior dificuldade encontrada estava relacionada à reusabilidade do código fonte.*

A atividade apresentada não foi considerada difícil por participantes do experimento, 3 participantes de um total de 6 participantes (50%) dos grupos *Parcial* e *Completa* consideram a atividade mediana e 50% a considerou fácil. No grupo *Controle* 2 dos 3 participantes (66.66%) consideraram a atividade fácil e 1 participante (33.3%) a considerou

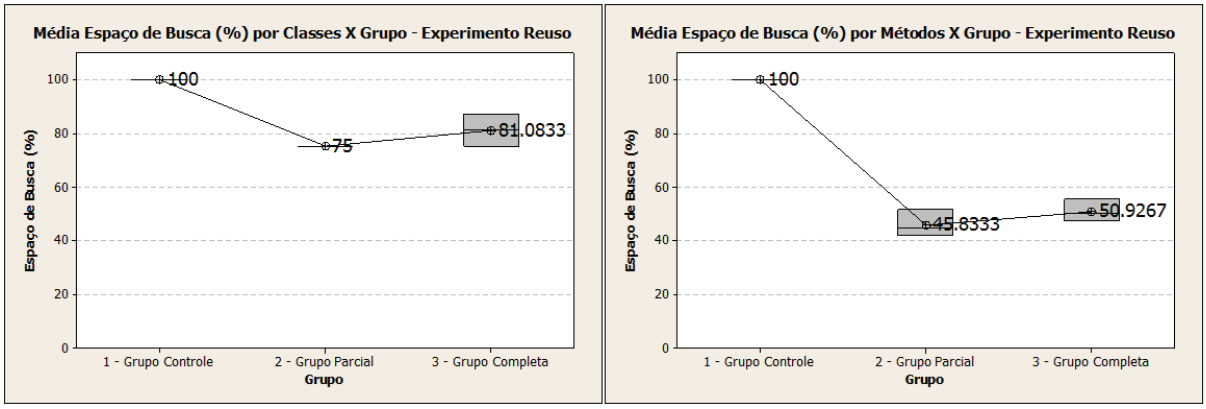



Figura 4.3: Gráficos das médias percentuais de redução do espaço de busca para os grupos que utilizaram a abordagem no experimento 1 - Reuso.

mediana. A complexidade do sistema era simples, de pequeno porte e bem estruturado, o que diminuía o grau de dificuldade da atividade. As dificuldades apresentadas pelos participantes de todos os grupos foram diversas, mas a mais citada (55.5% dos participantes) era relativa à atividade de reuso, compreender como alterar o código existente sem impactar no menu do *Jogo Connect* e proporcionando o mesmo menu para o *Jogo de Tabuleiro*.

### 4.1.5 Observações sobre o Espaço de Busca

**Observação 5.1 -** *Este experimento apresentou uma redução do espaço de busca inicial ao utilizar a abordagem, porém esta não foi expressiva em relação à redução do espaço de busca apresentado pelos experimentos de Correção e Melhoria.*

Neste experimento, conforme se pode notar na Figura 4.3, o grupo *Parcial* reduziu a média do espaço de busca inicial para 75% das classes do sistema e o grupo *Completa* reduziu para 81.08%. A redução do espaço de busca para os métodos foi de 45.83% para o grupo *Parcial* e 50.92% para o grupo *Completa*. A experimentadora considerou o espaço de busca inicial do grupo *Controle* como 100%, pois este grupo inicia sua análise sobre todos os elementos de código do sistema, enquanto os grupos *Parcial* e *Completa* iniciam suas buscar a partir da visão Mapeamento, onde o espaço de busca é reduzido. A redução do espaço de busca deste experimento, principalmente para o conjunto de classes envolvidas, não foi expressiva em relação à redução do espaço de busca apresentado pelos experimentos de Correção e Melhoria, que apresentaram em média 5.85% do espaço de busca original para os métodos dos sistemas alvo. Isto provavelmente ocorreu devido ao tamanho do sistema, por ser muito pequeno, e o fato de que as características envolvidas nesta manutenção correspondem a mais da metade das características do sistema. Isso faz com que boa parte dos elementos de código estejam realmente relacionados à manutenção solicitada e fossem listadas na coleta dos rastros destas características.

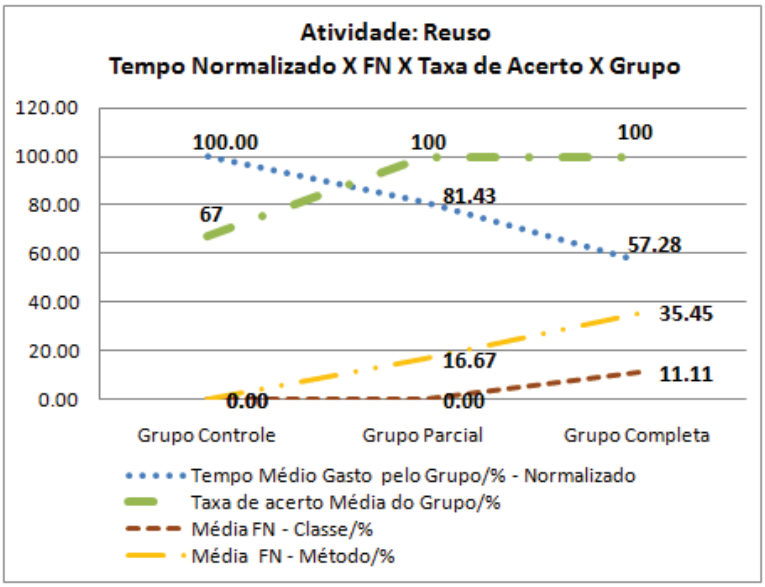



Figura 4.4: Gráfico de tempo normalizado versus o percentual de falsos negativos, a taxa de acerto e o grupo no experimento 1 - Reuso.

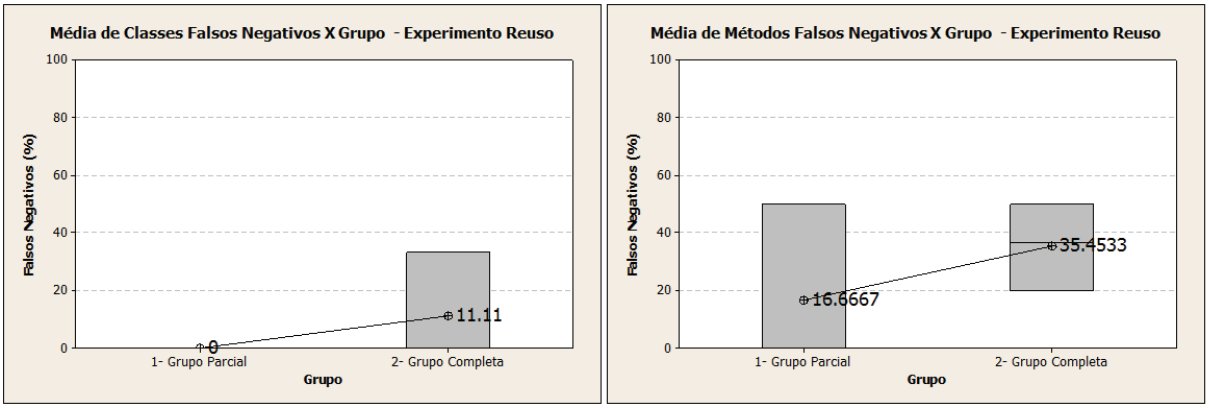



Figura 4.5: Gráficos do percentual médio de classes e métodos falsos negativos apresentados por grupo no experimento 1 - Reuso.

**Observação 5.2 -** *O número de falsos negativos não impactou a ponto de tornar a abordagem pior quando comparada ao grupo Controle.*

A existência de falsos negativos foi apontada por 80% dos participantes que utilizaram a abordagem, os outros 20% apontaram que todos os elementos de interesse foram listados. Os participantes listaram os falsos negativos, sendo em média 16.67% de métodos para o grupo *Parcial* e 35.45% para o grupo *Completa* nenhuma classe para o grupo *Parcial* e 11.11% das classes de interesse para o grupo *Completa*, conforme se pode notar nas Figuras 4.4 e 4.5. Cabe ressaltar que os elementos falsos negativos foram descobertos pelos sujeitos durante a análise estática a partir dos demais elementos apresentados nas visões dinâmicas. Mesmo com a presença dos elementos falsos negativos, os grupos *Completa* e *Parcial* apresentaram a média da taxa de acerto e tempo médio gasto com a atividade melhor do que o grupo *Controle*, o que leva a concluir a observação apresentada.

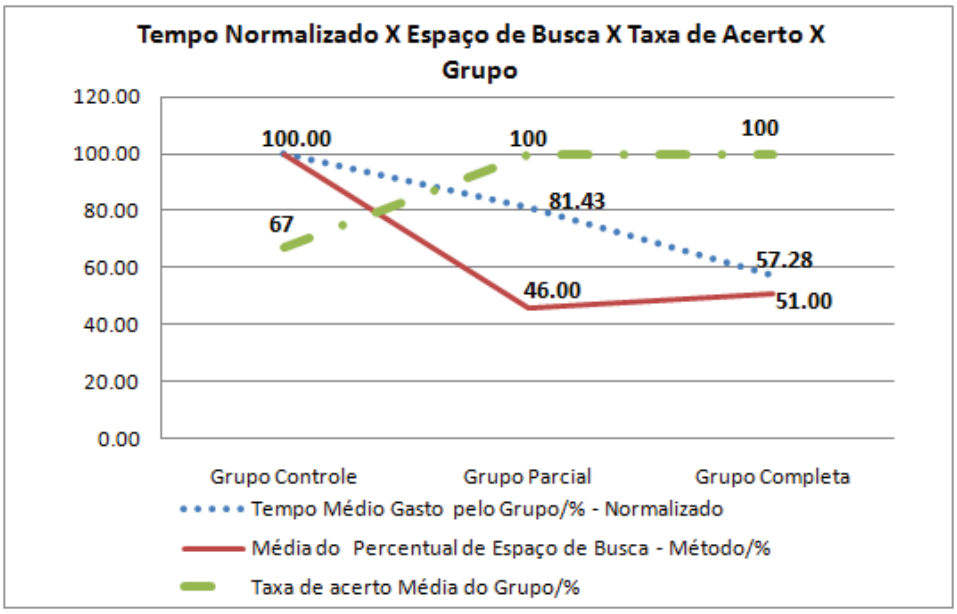



Figura 4.6: Gráfico de comportamento das curvas de tempo gasto com a atividade, espaço de busca inicial de métodos e taxa de acerto por grupos no experimento 1 - Reuso.

**Observação 5.3 -** *Ao analisar as curvas da taxa de acertos, redução do espaço de busca inicial e tempo gasto, nota-se que os grupos que utilizaram a abordagem apresentam maiores taxas de acerto, menores tempos de execução e menor espaço de busca inicial.*

Para melhor observar as relações entre o tempo gasto na atividade, espaço de busca e taxa de acerto por grupo, todos os valores destes dados foram transformados em percentuais e os valores da curva tempo foram normalizados, para poder montar um gráfico único que demonstrasse o comportamento das curvas e possibilitasse comparações. Lembrando que estas variáveis possuem desvios apresentados nos gráficos de tempo, taxa de acerto e espaço de busca. A Figura 4.6 demonstra a observação acima.

## 4.2 Experimento 2 - Correção

No segundo experimento, que envolve a atividade de correção de erro no sistema *JHotDraw*, as soluções tiveram uma variação menor. Todos os participantes apresentaram alterações parecidas, não iguais, envolvendo um conjunto de classes e métodos comuns, sendo as classes: *TextTool.java* e *CreationTool.java*, entretanto, somente a *TextTool.java* foi alterada, e, os métodos *endEdit()*, *getFieldsBounds()*, *mouseReleased()*, *creationFinished()* e o *fireToolDone()*, mas somente o *getFieldsBounds()* e *mouseReleased()* foram alterados.

A maioria dos sujeitos que utilizaram a abordagem, incluindo os que tiveram os melhores resultados em termos de tempo gasto com a atividade e sucesso no experimento, a utilizaram da seguinte maneira. Primeiro, eles coletaram os rastros de uma das ferramentas de texto. Depois, eles criaram um objeto do tipo texto na área da figura e

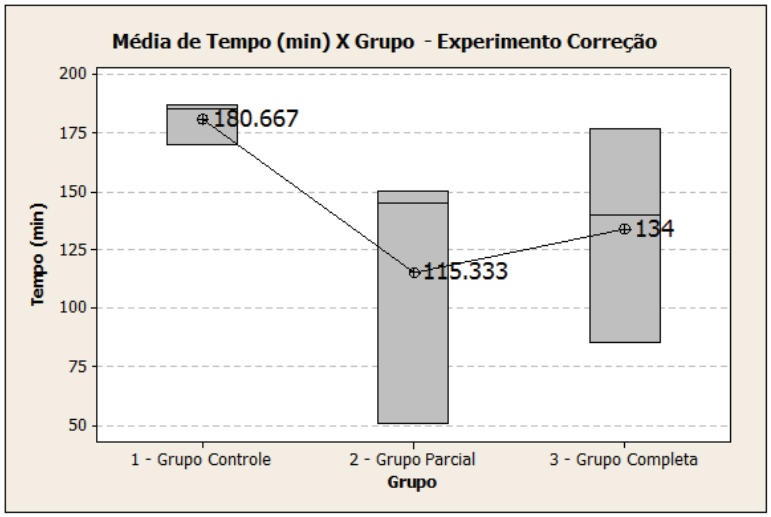



Figura 4.7: Gráfico de tempo médio (em minutos) gastos pelos grupos no experimento 2 - Correção.

separadamente coletaram o rastro de outra(s) ferramenta(s) de desenho (exemplo: desenhar figura quadrado), cuja funcionalidade *Undo* estava funcionando perfeitamente. Por último, coletaram os rastros da funcionalidade *Undo* para cada objeto gerado na figura. No momento da coleta do rastro do *Undo* para a ferramenta de texto, esta falhou apresentando o erro. A partir destes rastros, os sujeitos puderam analisar nas visões geradas o seguinte comportamento. Os roteiros de criação de um texto e o de criação de outro objeto, para o qual o *Undo* é executado perfeitamente, possuem elementos de código relacionados ao *Undo* e *Redo* em comum. Os participantes analisaram estas classes em comum listada na visão Mapeamento, buscando principalmente aquelas com nomes ligados ao interesse, verificaram os códigos fonte destas, e perceberam que a classe *CreationTool.java* era responsável por finalizar o *Undo*. Observaram diferenças nesta classe no mapeamento das características de criação do objeto texto e as de criação dos demais objetos, onde alguns métodos foram executados para a criação dos demais objetos e não foram executados para a ferramenta de texto. Neste experimento é importante destacar que a maneira como a abordagem foi utilizada influenciou no desempenho dos grupos, ajudando na localização das características.

A seguir serão apresentadas as observações realizadas sobre este experimento, que demonstrou resultados muito favoráveis ao uso da abordagem.

### 4.2.1 Observações sobre o Tempo

**Observação 1.1 -** *O desempenho em tempo de execução dos grupos que utilizaram a abordagem neste experimento (Completa e Parcial) foi superior ao grupo Controle, que não utilizou a abordagem.*

Os grupos que utilizaram a abordagem tiveram uma média de tempo gasto inferior ao grupo *Controle*, conforme se pode ver na Figura 4.7. Comparando o tempo do grupo

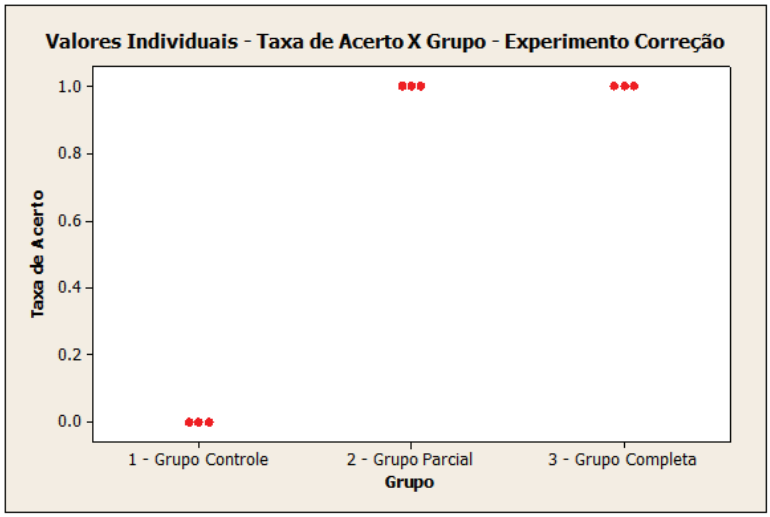



Figura 4.8: Gráfico da taxa de acerto percentual por grupo no experimento 2 - Correção

*Controle* com o do grupo *Parcial* que obteve o melhor desempenho em tempo, tem-se uma diferença de 65 minutos, o que corresponde a 34.21% do tempo para execução da atividade, um valor novamente considerável. Os desvios de tempo médio apresentados pelos grupos *Parcial* e *Completa* foram significativos e, portanto, tornam a comparação entre estes dois grupos menos confiável, indicando uma semelhança entre os mesmos. Entretanto, o desempenho da abordagem *Parcial* foi significativamente melhor do que da abordagem controle, mesmo considerando os desvios.

Provavelmente este desvio elevado nos dois grupos se deve ao fato destes terem que utilizar uma nova abordagem/ferramental, para o qual foram treinados, porém, não faz parte do cotidiano destes sujeitos. O nível de adaptação sobre a ferramenta foi observadamente variável, alguns sujeitos se familiarizaram com a abordagem melhor do que outros e isto interfere no tempo de execução. Além de outras variáveis não controladas como a disposição do participante para o experimento e nível de conhecimento em desenvolvimento na linguagem Java, já apresentadas, que podem influenciar nos experimentos, gerando tais desvios.

### 4.2.2 Observações sobre a Taxa de Acerto

**Observação 2.1 -** *A taxa de acerto dos grupos que utilizaram a abordagem neste experimento (Completa e Parcial) foi muito superior ao grupo* Controle*, que não utilizou a abordagem. Todos os participantes que utilizaram a abordagem tiverem sucesso na atividade e todos que não utilizaram não conseguiram concluir a atividade no tempo de experimento.*

Em relação ao segundo experimento, como se pode ver na Figura 4.8, nenhum participante do grupo *Controle* obteve sucesso na atividade e todos os participantes dos grupos que utilizaram a abordagem obtiveram sucesso. Obter sucesso nesta atividade significa corrigir o problema conforme requisitado e no tempo delimitado pelo experimento. Ao

término do tempo, todos os sujeitos do grupo *Controle* responderam as perguntas do questionário e apontaram corretamente os elementos de código que deveriam ser corrigidos, apesar de demonstrarem dúvida se estavam corretos, pois nenhum havia iniciado a atividade de correção em si. Talvez com mais tempo de experimento, estes tivessem executado a atividade com sucesso, porém em um tempo ainda maior de execução.

### 4.2.3 Observações sobre a Utilidade das Informações

**Observação 3.1 -** *As visões Mapeamento, Grafo de Chamadas e Mapeamento com Classificação foram úteis na compreensão da característica.*

A visão Mapeamento foi considerada útil para esta atividade por todos os sujeitos do grupo *Parcial*. Dos participantes do grupo *Completa*, que geraram todas as visões, 66.66% apontaram que a visão Mapeamento foi útil, 33.34% apontou que o Grafo de Chamada foi útil e 33.34% apontou o Mapeamento com Classificação como visão que o auxiliou na atividade.

**Observação 3.2 -** *Os elementos de código apresentados nas visões foram relevantes para a compreensão da característica.*

Conforme o parecer de todos os participantes que utilizaram a abordagem, esta listou todas as classes e métodos que foram reutilizados na atividade, e estes direcionaram a investigação para os elementos de código ligados à correção.

**Observação 3.3 -** *A maioria dos participantes que utilizaram a abordagem apresentou satisfação com o uso desta.*

A maioria dos participantes (83.33%) apresentou satisfação em utilizar a abordagem, apenas um participante (16.67%) não ficou satisfeito, este apresentou insatisfação com a usabilidade das ferramentas e resistência com a nova abordagem diante do bom domínio que possui sobre as ferramentas do IDE Eclipse. Este sujeito acredita que as funções do IDE o auxiliam perfeitamente na localização das características, porém, afirmou que a visão Grafo de Chamada o auxiliou na atividade e a achou interessante.

### 4.2.4 Observações sobre a Dificuldade da Atividade

**Observação 4.1 -** *A atividade foi classificada como mediana por 83.40% dos participantes dos grupos Parcial e Completa. E foi considerada difícil por 100% dos sujeitos do grupo Controle.*

A atividade apresentada não foi considerada difícil por nenhum participante dos grupos *Parcial* e *Completa*, 83.40% dos participantes consideram a atividade mediana e 16.60% a considerou fácil. No grupo *Controle* 100% dos participantes consideraram a atividade difícil. O sistema *JHotDraw* é de porte médio e os sujeitos do grupo *Controle* apresentaram grande dificuldade para localizar onde deveria ser realizada a correção, os participantes

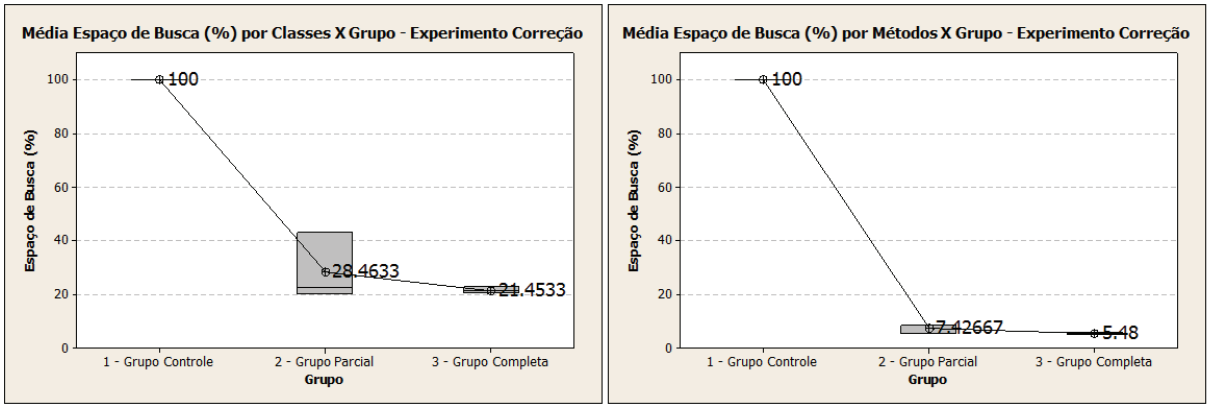



Figura 4.9: Gráficos com a média percentual da redução do espaço de busca para os grupos que utilizaram a abordagem no experimento 2 - Correção.

chegaram a localizar os elementos de código envolvidos na correção, mas já no final do tempo para execução da atividade e, portanto não a concluíram.

### 4.2.5 Observações sobre o Espaço de Busca

**Observação 5.1 -** *Este experimento apresentou expressiva redução do espaço de busca inicial ao utilizar a abordagem.*

O espaço de busca inicial médio do grupo *Parcial* foi de 28.47% das classes e 7.43% dos métodos e para o grupo *Completa* a média foi de 21.64% das classes e 5.48% dos métodos, conforme apontado na Figura 4.9. Diante de um sistema com 466 classes e 4078 métodos, esta redução é considerável. O grupo *Controle* apresenta 100% de espaço de busca inicial, pois este grupo inicia sua análise sobre todos os elementos de código do sistema, enquanto os demais grupos iniciam a busca sobre os elementos listados nas visões.

**Observação 5.2 -** *O número de falsos negativos foi nulo, portanto, não impactou os resultados.*

Todos os participantes que utilizaram a abordagem afirmaram que não ocorreram elementos de código falsos negativos nas visões, conforme apontado nos gráficos das Figuras 4.10 e 4.11. Todos os elementos envolvidos indiretamente ou diretamente com a correção realizada por eles foram listados. Logo, a inexistência deste tipo de elemento não pode impactar nos resultados.

**Observação 5.3 -** *Os grupos que utilizaram a abordagem apresentam maiores taxas de acerto, menores tempos de execução e menor espaço de busca inicial.*

A Figura 4.12 apresenta as curvas correspondentes ao tempo médio normalizado (percentual) gasto, a taxa de acerto percentual e o espaço de busca inicial dos métodos para cada grupo. É observável neste experimento, que quando houve uma redução do espaço de busca, houve uma taxa de acerto maior e um tempo médio menor para execução da atividade.

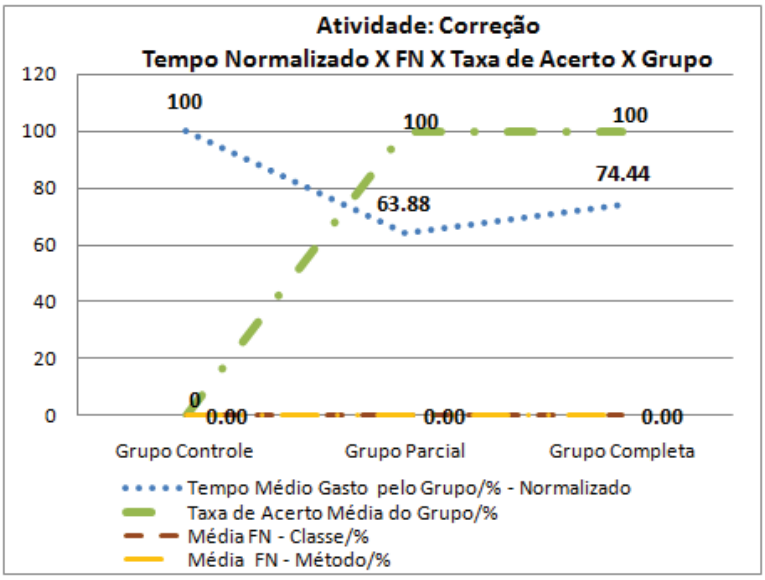



Figura 4.10: Gráfico com o tempo normalizado versus o percentual de falsos negativos, a taxa de acerto e o grupo no experimento 2 - Correção.

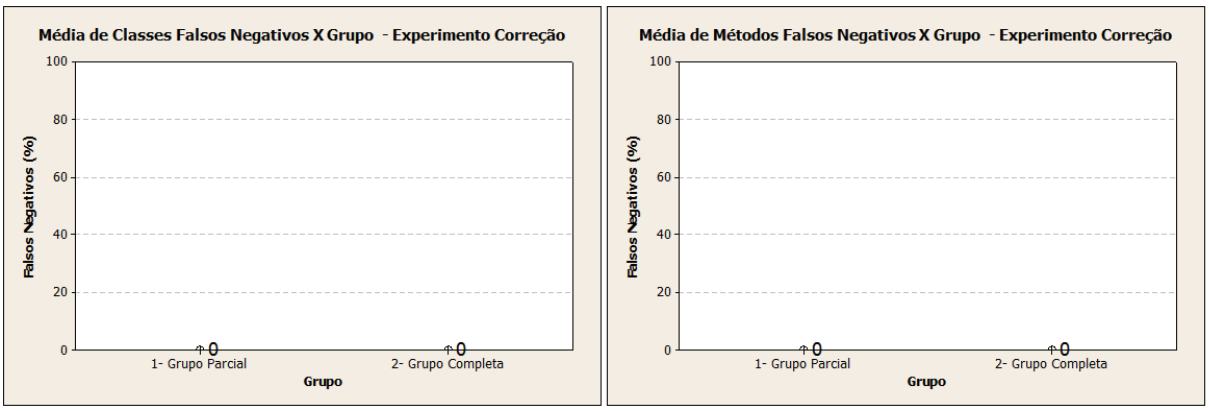



Figura 4.11: Gráficos com percentual médio de classes e métodos falsos negativos apresentados por grupo no experimento 2 - Correção.

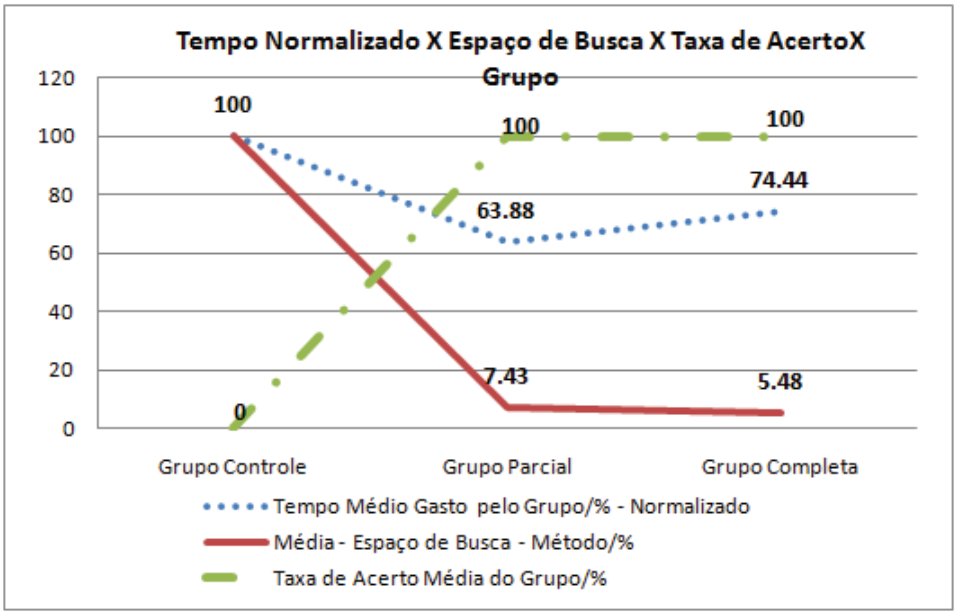



Figura 4.12: Gráfico com o comportamento das curvas de tempo gasto com a atividade, espaço de busca inicial de métodos e taxa de acerto por grupos no experimento 2 - Correção.

## 4.3 Experimento 3 - Localização

O terceiro experimento envolve a localização e compreensão dos elementos envolvidos na evolução de características, que foi detalhada na Seção 3.4.2. Neste, os grupos realizaram a atividade em um tempo menor do que os demais experimentos, pois não contemplou alterações no código fonte, apenas localização e compreensão das características. Os sujeitos analisaram o sistema *JHotDraw* e responderam a três perguntas relativas à localização do código que implementa as características *Undo* e *Redo* para as versões 7.1 e 5.3, e respoderam qual foi a evolução realizada nestas características de uma versão para outra. Como facilitador desta localização, o *JHotDraw* é um sistema bem estruturado que possui um pacote organizado com as classes e métodos que implementam estas características, e estes elementos de código foram nomeados de maneira intuitiva, contemplando prefixos ou sufixos com a palavra *Undo* e *Redo*, o que facilitou muito a atividade em questão.

### 4.3.1 Observações sobre o Tempo

**Observação 1.1 -** *O desempenho em tempo médio dos grupos que utilizaram a abordagem Completa neste experimento foi inferior ao grupo Controle.*

Os grupos que utilizaram a abordagem apresentaram uma média de tempo maior do que o grupo *Controle*, conforme se pode notar na Figura 4.13. Porém, o tempo médio de 70 minutos e 90 minutos dos grupos *Parcial* e *Completa* contemplam os minutos que estes grupos gastaram para executar a abordagem e analisar o código das características de interesse. Por outro lado, o tempo que os sujeitos do grupo *Controle* contempla apenas o tempo de análise do código. E, apesar da média do tempo grupo *Parcial* ter sido maior do que o grupo *Controle*, em função dos desvios não é possível afirmar que esta última tenha sido sistematicamente melhor. Entretanto, o grupo *Completa* teve um desempenho nitidamente pior.

Conforme relato de todos os participantes do grupo *Controle*, encontrar as classes que implementam a característica foi facilitada pela boa organização dos pacotes e pela busca do IDE Eclipse por classes que possuem as palavras *Undo* e *Redo*. Como a maioria (92.85%) das classes que implementam estas características possuem estas palavras como sufixo ou prefixo, a busca foi facilitada. O *JHotDraw* 7.1 apresenta um pacote chamado *org.jhotdraw.undo*, onde as cinco classes que implementam estas características se encontram organizadas. A versão 5.3 apresenta um pacote chamado *org.jhotdraw.util*, onde todas as nove classes estão listadas (entre outras classes) e todas possuem nomes com as palavras chaves como sufixo ou prefixo. A facilidade de localização dos elementos de código de interesse pode ter influenciado no desempenho em tempo dos grupos.

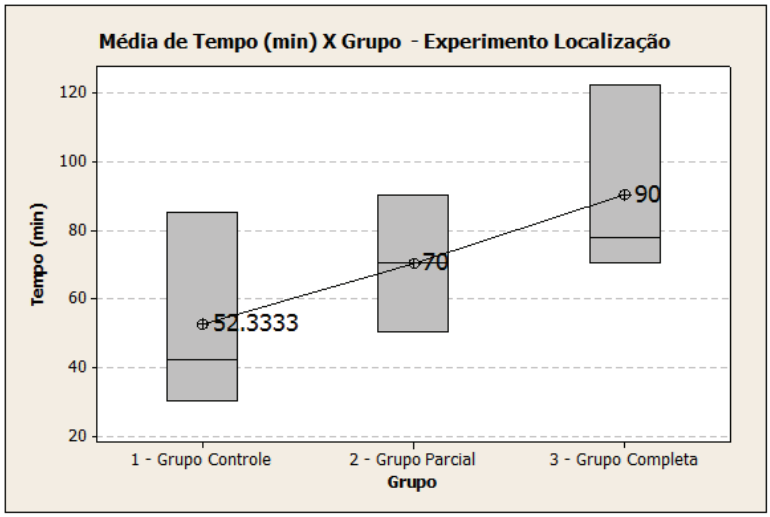



Figura 4.13: Gráfico com o tempo médio (em minutos) gastos pelos grupos no experimento 3 - Localização.

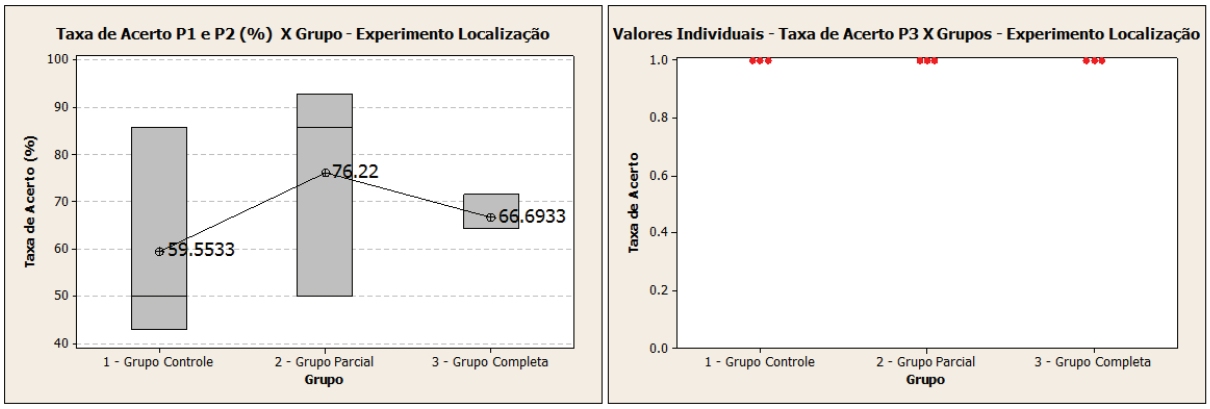



Figura 4.14: Gráficos com o percentual médio de acertos nas perguntas 1 e 2 do questionário e taxa de acerto percentual na pergunta 3 por grupo no experimento 3 - Localização.

### 4.3.2 Observações sobre a Taxa de Acerto

**Observação 2.1 -** *A média da taxa de acerto dos grupos que utilizaram a abordagem foi superior ao grupo Controle, porém com valores muito próximos, indicando uma semelhança nos mesmos.*

Como, neste experimento os sujeitos não realizaram alterações no código fonte e somente responderam às perguntas relativas às características de interesse, avaliou-se as respostas a estas perguntas para obter a taxa de acerto dos participantes e a média dos grupos. Seguem as perguntas avaliadas:

- Pergunta 1) Quais classes estão relacionadas ao *Undo* e *Redo* do *JHotDraw* 7.1?
- Pergunta 2) Quais classes estão relacionadas ao *Undo* e *Redo* do *JHotDraw* 5.3?
- Pergunta 3) Resuma qual foi a alteração que ocorreu nas características *Undo* e *Redo* da versão 5.3 para a Versão 7.1 do *JHotDraw*.

Como se pode ver na Figura 4.14, todos os grupos apresentaram 100% de acerto para a Pergunta 3 do questionário e um percentual de erros muito próximo para as Pergunta 1 e 2 do questionário, apesar de o grupo *Parcial* ter apresentado em média um resultado ligeiramente melhor. Os desvios dos grupos *Controle* e *Parcial* foram maiores do que o grupo *Completa*. Logo, neste estudo não se pode afirmar que existe uma diferença significativa entre os grupos.

### 4.3.3 Observações sobre a Utilidade das Informações

**Observação 3.1 -** *As visões Mapeamento e Classificação das Classes foram úteis na compreensão da característica.*

Todos os participantes do grupo *Parcial* consideram a visão Mapeamento útil para a atividade. Dos participantes do grupo *Completa*, que geraram todas as visões, 66.66% apontaram que a visão Mapeamento foi realmente útil para a atividade e 33.34% apontou a visão Classificação das Classes como útil.

**Observação 3.2 -** *Os elementos de código apresentados nas visões foram relevantes para a compreensão da característica.*

Conforme o parecer de todos os participantes que utilizaram a abordagem, esta listou elementos de código relacionados às características que foram importantes para direcionar a exploração por todas as classes e métodos que implementam as características *Undo* e *Redo* e compreensão da evolução ocorrida entre as versões.

**Observação 3.3 -** *Todos os participantes que utilizaram a abordagem apresentaram satisfação com o uso desta. Porém, os sujeitos que perceberam a existência de FN (33.33% dos 6 participantes) apresentaram este ponto como negativo para a abordagem.*

Esta satisfação foi apresentada no questionário ao demonstrar como a abordagem os auxiliaram. Porém, os sujeitos que perceberam a existência de falsos negativos, apresentaram este ponto com negativo para a abordagem. Alguns fizeram sugestões de melhorias que serão apresentadas na Seção 6.1.

### 4.3.4 Observações sobre a Dificuldade da Atividade

**Observação 4.1 -** *A atividade foi classificada pelos grupos Completa e Parcial como fácil por 50% dos participantes, 33.33% classificou como Mediana e 16.66% (1 pessoa) classificou como difícil. E, todos os participantes do grupo Controle a classificaram como mediana.*

Os grupos *Parcial* e *Completa* não apresentaram unanimidade na classificação desta atividade, mas a opinião mais apontada foi a de que esta é uma atividade fácil. Já no grupo *Controle*, todos os sujeitos classificaram esta atividade como mediana. Olhando este percentual diante da classificação dada pelos 9 participantes (3 grupos), tem-se 55.55% de sujeitos que a classificaram como mediana.

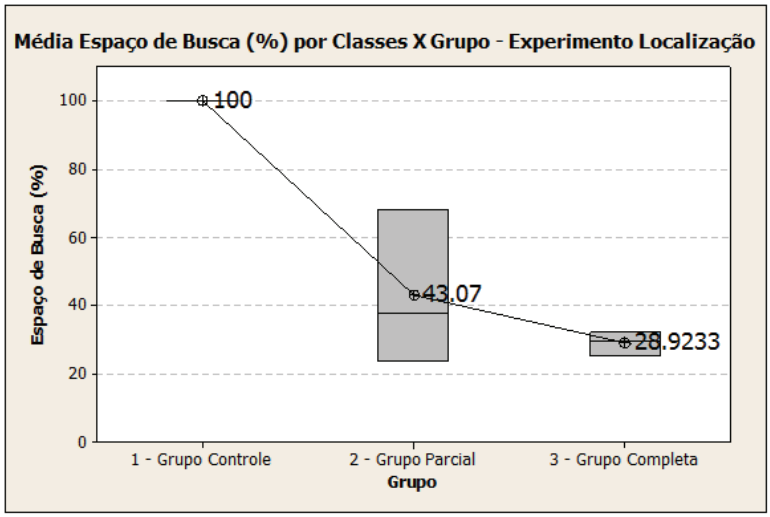



Figura 4.15: Gráfico com a média percentual da redução do espaço de busca por classes versus o grupo do experimento 3 - Localização.

### 4.3.5 Observações sobre o Espaço de Busca

**Observação 5.1 -** *Este experimento apresentou uma redução do espaço de busca inicial ao utilizar a abordagem, porém esta não foi expressiva em relação à redução do espaço de busca apresentado pelos experimentos de Correção e Melhoria.*

O espaço de busca inicial médio do grupo *Parcial* foi de 43.07% das classes e para o grupo *Completa* a média foi de 28.89% das classes, como se pode ver na Figura4.15. Estes apresentaram redução no espaço de busca inicial, porém a redução do grupo *Parcial* não foi expressiva em relação à redução do espaço de busca apresentado pelos experimentos de Correção e Melhoria, que apresentaram uma média de 24.13% do espaço de busca original para as classes dos sistemas alvo. Os sujeitos que utilizaram a abordagem para obter informações sobre o *Undo* e *Redo*, executaram muitas características, pois estas são executadas para cada tipo de desenho que se pode gerar no *JHotDraw* com diferenças no disparo, logo, um conjunto considerável de características foram executadas, o que impacta no número de elementos obtidos no rastro e consequentemente na redução do espaço de busca inicial.

**Observação 5.2 -** *O número de falsos negativos não impactou a ponto de tornar a abordagem pior, quando comparada ao grupo Controle.*

Como as perguntas destes experimentos apenas indagaram a localização das classes que implementam as versões 5.3 e 7.1 do *JHotDraw*, então os dado sobre os elementos falsos negativos, somente, são apresentados para os elementos do tipo classe. Como resultado, conforme se pode ver na Figura 4.17, o grupo *Parcial* apresentou em média 25.93% de FN para a versão 5.3 e 33.33% de FN para a versão 7.1. O grupo *Completa* apresentou média de 22.22% de FN para a versão 5.3 e 33.33% para a versão 7.1. Ao analisar os resultados, se percebe que apenas 33.33% dos sujeitos que utilizaram a abordagem descobriram os FN para a versão 5.3 e este mesmo valor se repete para a versão 7.1, com o detalhe

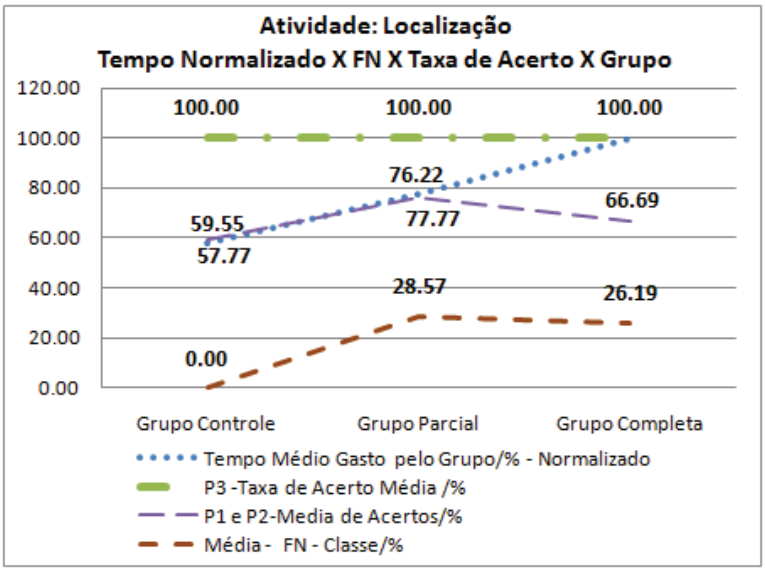



Figura 4.16: Gráfico com o tempo normalizado versus o percentual de falsos negativos, a taxa de acerto e o grupo no experimento 3 - Localização.

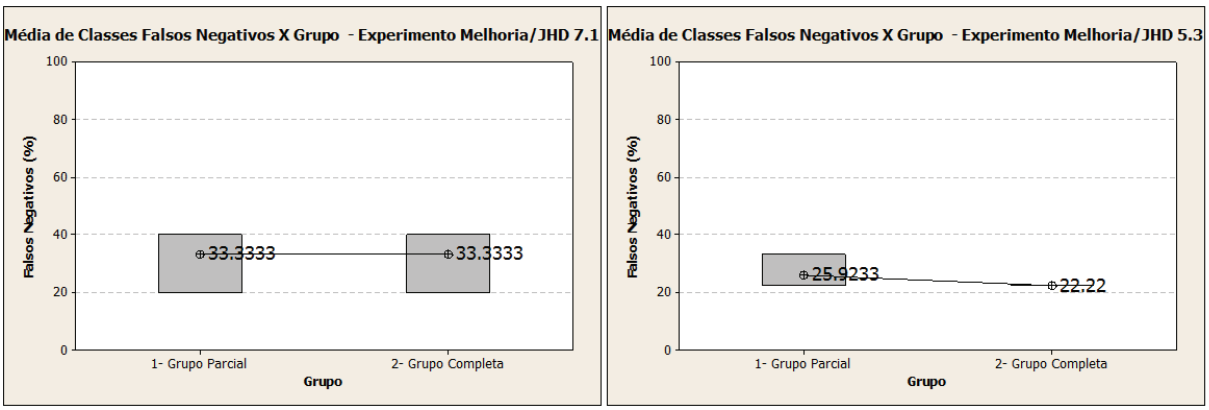



Figura 4.17: Gráficos com o percentual médio de classes e métodos falsos negativos apresentados por grupo no experimento 3 - Localização, nas versões 7.1 e 5.3 do *JHotDraw*, respectivamente.

de que não foram os mesmos sujeitos. Ao verificar as respostas sobre a localização dos elementos de código para as características, a experimentadora percebeu que a maioria dos participantes (66.66%) apenas listou os elementos verdadeiros positivos que foram apresentados pela abordagem.

Porém, ao comparar a taxa de acerto (percentual médio) na localização das classes de interesse (Figura 4.16) e a média de FN por grupo, temos que o grupo que obteve a maior média de FN foi o que obteve a melhor taxa de acerto. A média de FN variou minimamente entre o grupo *Completa* e *Parcial*, e as taxas de acerto destes grupos foram melhores do que a do grupo *Controle*, porém os resultados foram muito próximos. Logo, se pode concluir que o número de falsos negativos não impactou a ponto de tornar a abordagem pior, quando comparada ao grupo *Controle*.

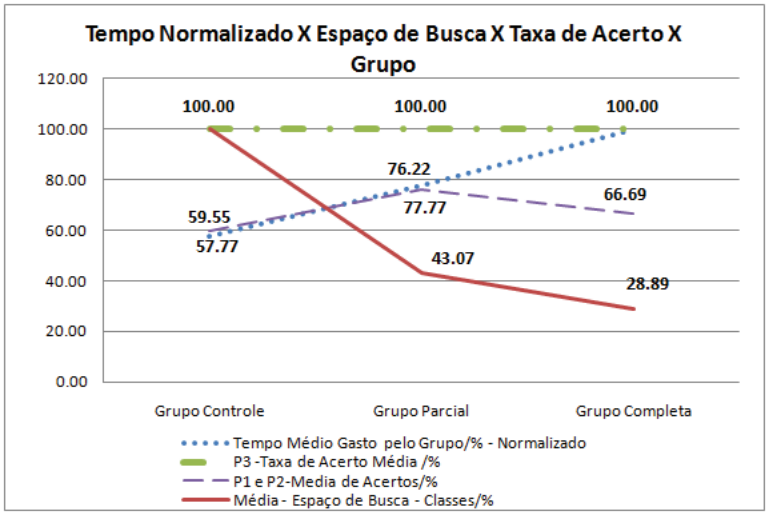



Figura 4.18: Gráfico com o comportamento das curvas de tempo gasto com a atividade, espaço de busca inicial de métodos e taxa de acerto por grupos no experimento 3 - Localização, respectivamente.

**Observação 5.3 -** *Ao contrário dos demais experimentos, o grupo com o maior espaço de busca inicial foi o que apresentou o menor tempo de execução, porém, assim como nos outros experimentos, a taxa de acerto foi maior para os grupos com menor espaço de busca inicial.*

Apesar do espaço de busca inicial do grupo *Controle* ter sido 100%, este apresentou o menor tempo de execução da atividade, conforme se pode ver na Figura 4.18, e isto provavelmente ocorreu por que o sistema *JHotDraw* é bem organizado e possui nomenclatura intuitiva nos elementos de código, o que facilitou a localização das classes de interesse utilizando-se de funcionalidades do Eclipse como o *Search* e *Package Explorer*, conforme explicado na anteriormente. Para este tipo de atividade e para um sistema com esta organização, esta localização de características é um problema tratado com facilidade utilizando-se somente as funcionalidades do IDE Eclipse. Além do mais, a abordagem proposta demanda tempo para ser executada, o que provavelmente impactou para que os grupos *Parcial* e *Completa* tivessem as maiores médias de tempo, tanto que, o grupo *Completa*, cujo número de visões para se analisar é maior, obteve a maior média de tempo.

Quanto à taxa de acerto, os grupos que apresentaram menor espaço de busca inicial, *Parcial* e *Completa* apresentaram melhor taxa de acerto, porém, com uma variação muito pequena em relação ao grupo *Controle* e com um tempo de execução maior.

## 4.4 Experimento 4 - Melhoria

No quarto e último experimento, que trata da melhoria no objeto *Comentário* do sistema *ArgoUML*, as soluções apresentadas pelos sujeitos participantes deste experimento variaram, porém envolvendo um grupo de elementos de código em comum. Devido ao

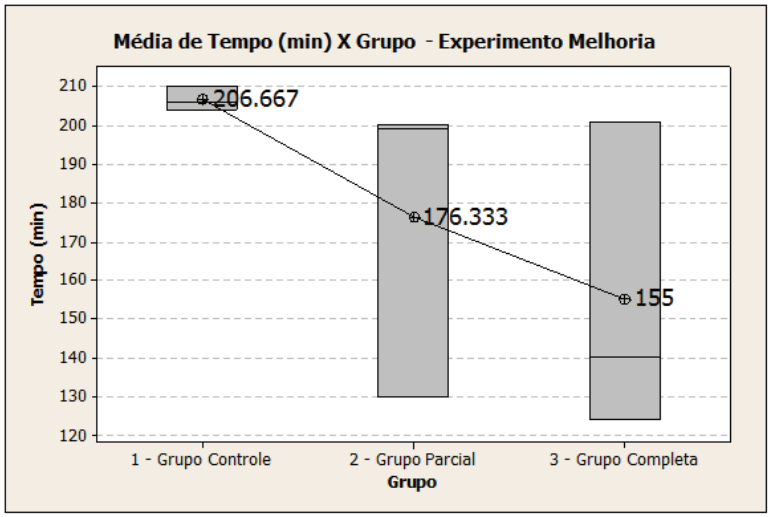



Figura 4.19: Gráfico com o tempo médios gastos pelos grupos no experimento 4 - Melhoria.

tamanho do sistema, esta atividade foi a que apresentou o maior nível de dificuldade, segundo os sujeitos. E como esta se constitui de diversas alterações no código fonte para cumprir o requisito, a experimentadora avaliou o cumprimento das atividades conforme a taxa de acerto nas alterações necessárias pra cumprir o requisito. Alguns sujeitos atenderam parcialmente o requisito, realizaram alterações que contribuiriam para a execução completa deste, mas não completaram a atividade no tempo limite. Para medir esta taxa de acerto parcial foi dada uma nota para cada alteração necessária para o cumprimento do requisito conforme o grau de dificuldade desta. Estes valores foram utilizados para pontuar a taxa de acerto de cada sujeito na atividade. Sujeitos que cumpriram todo o requisito receberam nota máxima. Todos os resultados foram testados pela pesquisadora para verificar e registrar o cumprimento dos requisitos. A taxa de acerto, os tempos de execução da atividade e de preenchimento dos questionários, e os resultados qualitativos apresentados nestes foram registrados e são fontes para as observações apresentadas abaixo.

### 4.4.1 Observações sobre o Tempo

**Observação 1.1 -** *O desempenho em tempo médio dos grupos que utilizaram a abordagem neste experimento foi superior ao grupo Controle.*

Os grupos que utilizaram a abordagem apresentaram uma média de tempo de execução da atividade inferior ao grupo *Controle* conforme se pode notar na Figura 4.19. Ao comparar o tempo médio do grupo *Completa* (155 minutos) com o grupo *Controle* (206.67 minutos) tem-se uma diferença de aproximadamente 51 minutos, o que corresponde um valor considerável de 24.28% do tempo dado para execução da atividade. Ainda mais quando esta diferença de tempo poderia ser ainda maior, já que todos os participantes do grupo *Controle* atingiram o limite do tempo de execução, sem concluírem a atividade

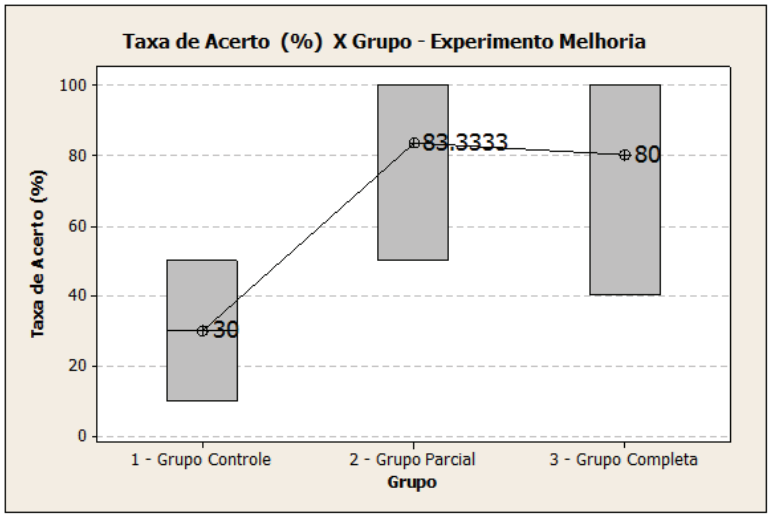



Figura 4.20: Gráfico com a taxa de Acerto percentual por grupo no experimento 4 - Melhoria.

solicitada. Logo, esta média de 206.67 minutos poderia ser maior, se lhes fosse dado mais tempo.

### 4.4.2 Observações sobre a Taxa de Acerto

**Observação 2.1 -** *A taxa de acerto média para os grupos que utilizaram a abordagem neste experimento foi superior ao grupo Controle.*

Sobre as taxas de acerto, apresentadas na Figura 4.20, nenhum membro do grupo *Controle* completou a atividade, porém realizou alterações necessárias para a conclusão do requisito que foram pontuadas conforme o grau de dificuldade e gerou o resultado de 30% de taxa de acerto para o grupo *Controle*. Nos grupos *Completa* e *Parcial* 66.66% dos participantes, de cada grupo, completaram a atividade com sucesso, cumprindo todo o requisito, e 33.33% não concluíram a atividade, mas executaram alterações que, também, foram pontuadas e compõem a taxa de acerto destes grupos, sendo que o grupo *Parcial* obteve 83% de taxa de acerto e grupo *Completa* obteve 80%.

### 4.4.3 Observações sobre a Utilidade das Informações

**Observação 3.1 -** *As visões Mapeamento, Classificação das Classes e Mapeamento com Classificação foram úteis na compreensão da característica.*

Todos os participantes que utilizaram a abordagem apontaram que a visão Mapeamento foi útil para esta atividade, ao avaliar esta visão foi possível apontar os elementos que deveriam ser alterados para a melhoria. Para o grupo *Completa*, em que os participantes foram instruídos a utilizar as demais visões, a Classificação das Classes e Mapeamento com Classificação foram consideradas úteis por 66.66% dos participantes que as geraram.

Nenhum participante apontou que utilizou a visão Grafo de Chamadas para esta atividade.

**Observação 3.2 -** *Os elementos de código apresentados nas visões foram relevantes para a compreensão da característica.*

Todos os participantes que utilizaram a abordagem apontaram que os elementos de código (classes e métodos) envolvidos na atividade foram apresentados nas visões da abordagem, e as visões foram úteis para a localização destes, direcionando a compreensão do código fonte de interesse.

**Observação 3.3 -** *Todos os participantes que utilizaram a abordagem apresentaram satisfação com o uso desta.*

Todos os participantes apresentaram satisfação ao utilizar a abordagem, conforme pareceres positivos apresentados nos questionários, onde apontaram como esta os auxiliou na atividade e a opinião sobre a abordagem. E, alguns sujeitos apontaram melhorias que serão apresentadas na Seção 6.1.

### 4.4.4 Observações sobre a Dificuldade da Atividade

**Observação 4.1 -** *Para os grupos que utilizaram a abordagem, a atividade foi classificada por 50% dos participantes como mediana, 33.33% classificaram como fácil e 16.66% como difícil. No grupo Controle, a atividade foi classificada como difícil por 66.66% dos participantes e como mediana por 33.33%.*

A atividade apresentada foi considerada pela metade (50%) dos sujeitos participantes dos experimentos como mediana, 33.33% classificaram como fácil e 16.66% como difícil. Quando apresentada a solicitação da atividade, esta foi considerada fácil pela maioria dos sujeitos, porém devido ao tamanho e complexidade do sistema *ArgoUML*, a maioria mudou a opinião. É importante observar que no grupo *Controle*, 66.66% apontaram a atividade de localizar onde deveria ser realizada a alteração como a maior dificuldade encontrada. Os grupos que utilizaram abordagem, também, apresentaram esta dificuldade, 66.66% apontaram que tiveram dificuldade em localizar dentre os métodos listados nas visões da abordagem quais deveria ser alterados, devido ao número elevado de falsos positivos, já discutidos. Contudo, mesmo com o número elevado de elementos listados que não faziam parte do interesse, os grupos *Parcial* e *Completa* tiveram maior facilidade para localizar as classes e métodos que deveriam ser alteradas, conforme se pode notar pela taxa de acerto dos grupos, na Figura 4.20.

### 4.4.5 Observações sobre o Espaço de Busca

**Observação 5.1 -** *Este experimento apresentou uma redução do espaço de busca inicial expressiva ao utilizar a abordagem.*

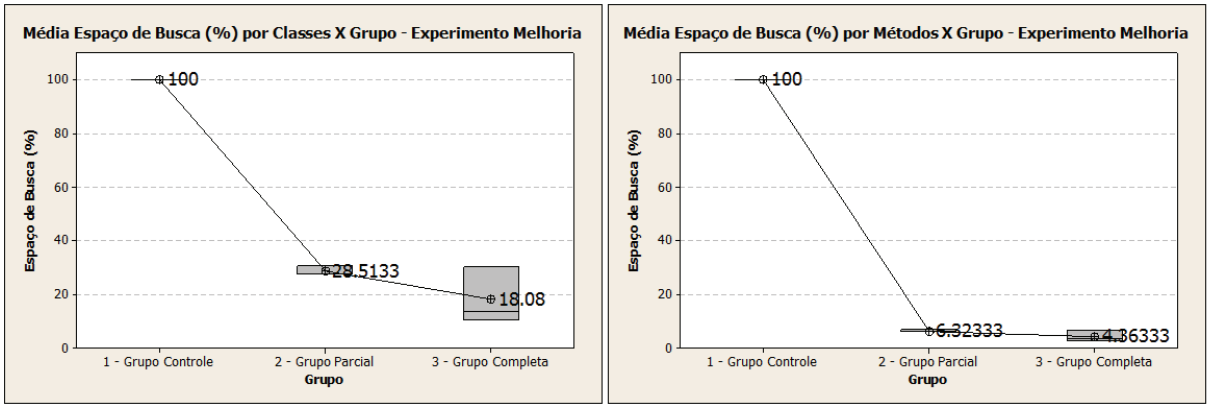



Figura 4.21: Gráficos com a média percentual da redução do espaço de busca para os grupos que utilizaram a abordagem no experimento 4 - Melhoria.

A redução do espaço de busca inicial apresentado pelos grupos *Parcial* e *Completa* foram consideráveis e diante de um sistema como o *ArgoUML* 0.26.1 que possui 14.038 métodos e 2.022 classes, provavelmente é de grande ajuda para o direcionamento da localização das características, conforme relatado pelos participantes dos experimentos. A média do espaço de busca inicial apresentado foi de 28.51% das classes e 6.32% dos métodos para o grupo *Parcial* e 18.08% das classes e 4.36% dos métodos para o grupo *Completa*, conforme apresentado na Figura 4.21. Notadamente, a redução do espaço de busca principalmente para o conjunto de métodos envolvidos foi muito expressiva. Porém, o espaço de busca poderia ter sido menor se não houvesse ocorrido um número elevado de falsos positivos. Estes falsos positivos foram apresentados nas visões devido às rotinas de críticas de projetos em UML que não são visíveis ao usuário final e são executadas enquanto o sistema estiver em uso. Para minimizar este número de elementos FP, os sujeitos poderiam ter utilizado o filtro da pasta onde se encontram estas rotinas de críticas, ao gerar o Mapeamento na *TraceToConcern*, e os elementos desta pasta não seriam listados. Porém, os sujeitos não foram instruídos a fazê-lo e não conheciam tão bem o sistema para perceber no momento da coleta a execução destas rotinas que não são visíveis ao usuário do *ArgoUML*.

**Observação 5.2 -** *O número de falsos negativos foi nulo, portanto, não impactou os resultados.*

Dos participantes que utilizaram a abordagem, 83.33% afirmaram que não ocorreram elementos de código falsos negativos nas visões, todos os elementos envolvidos indiretamente ou diretamente com a correção foram listados. Uma única pessoa (16.66%) afirmou que uma das classes, necessária para a solução dada por esta, não foi apresentada nas visões, constituindo então um FN. Porém, a experimentadora verificou a veracidade de todas as afirmações e percebeu que neste caso, o elemento foi listado e o sujeito não percebeu a presença deste nas visões. Logo, o percentual de falsos negativos é igual a 0% conforme se pode notar na Figura 4.22 e na Figura 4.23. Assim, não ocorreu impacto causado por elementos falsos negativos nas visões.

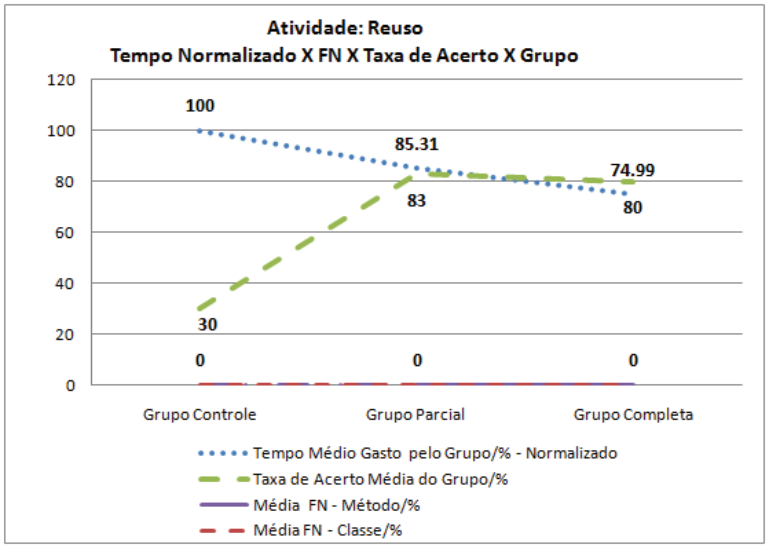



Figura 4.22: Gráfico com a tempo normalizado versus o percentual de falsos negativos, a taxa de acerto e o grupo no experimento 4 - Melhoria.

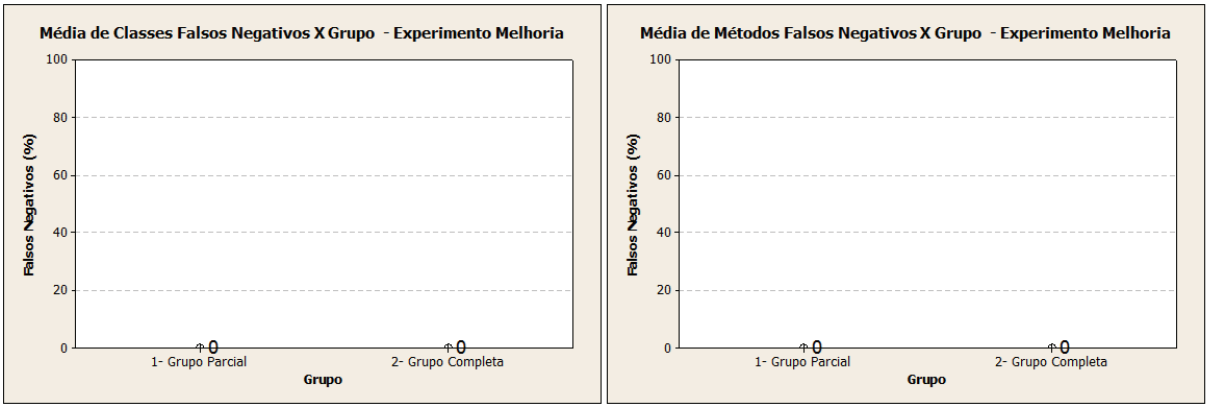



Figura 4.23: Gráficos com o Percentual Médio de classes e métodos Falsos Negativos apresentados por grupo no experimento 4 - Melhoria.

Conforme parecer dos participantes e percepção da experimentadora ao analisar todos os resultados gerados, o número de elementos falsos positivos foi elevado. Entretanto, a definição de quais são os elementos falsos positivos não foi possível e este valor não é apresentado, conforme já discutido anteriormente. Porém, para este experimento ficou nítido que o volume de elementos falsos positivos foi elevado devido às rotinas que são executadas em tempo integral pelo *ArgoUML* e que foram coletadas no rastro, sem fazerem parte das características de interesse. Este número elevado prejudicou o uso das visões por parte dos sujeitos, que apontaram a dificuldade em achar qual método deveria ser alterado dentre os métodos listados nas visões. O que provavelmente levou o sujeito citado a não perceber a presença de um elemento importante para a atividade na visão. Porém, mesmo com o volume elevando de falsos positivos, o desempenho dos grupos que utilizaram a abordagem foi superior.

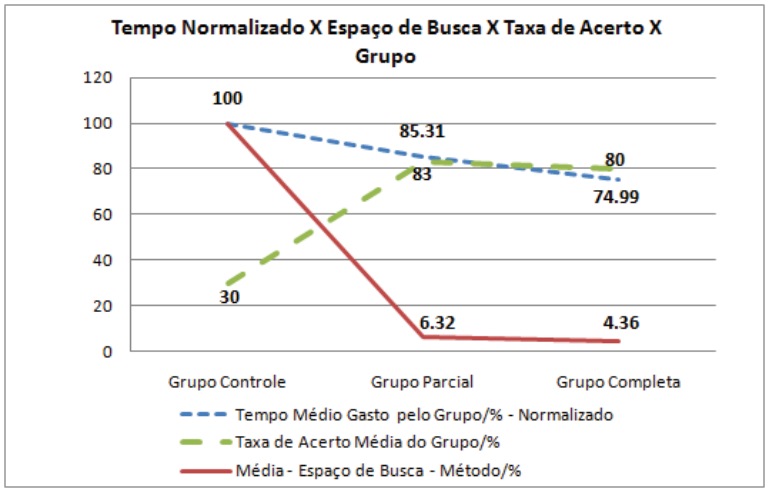



Figura 4.24: Gráfico com o comportamento das curvas de tempo gasto com a atividade, espaço de busca inicial de métodos e taxa de acerto por grupos no experimento 4 - Melhoria.

**Observação 5.3 -** *Os grupos que utilizaram a abordagem apresentam maiores taxas de acerto, menores tempos de execução e menor espaço de busca inicial.*

A Figura 4.24 apresenta as curvas correspondentes ao tempo médio normalizado (percentual) gasto, a taxa de acerto percentual e o espaço de busca inicial dos métodos para cada grupo. Conforme pode-se perceber pelas curvas os grupos *Parcial* e *Completa*, quando comparados ao grupo *Controle*, apresentam menor tempo médio de execução com maior taxa de acerto e menor espaço de busca. Este comportamento não obedece a uma tendência, especialmente quando comparamos o grupo *Controle* e *Parcial*. Entretanto se pode observar neste experimento que quando houve uma redução do espaço de busca, houve uma taxa de acerto maior e um tempo médio menor para execução da atividade, se considerar o grupo *Controle* comparado com os outros dois grupos.

Nesta seção foram apresentados os resultados dos 4 experimentos realizados, em forma de observações que trazem resultados para cada situação experimentada. Estes resultados, quando agrupados para uma avaliação geral, fornecem dados mais confiáveis diante de 36 observações individuais, levando à resultados mais robustos. E estes resultados serão apresentados e discutidos no próximo capítulo.

## 4.5 Discussão e Generalização dos Resultados

Nesta seção serão apresentados e discutidos os resultados gerais obtidos nos experimentos. Primeiramente serão apresentados e discutidos os resultados em relação ao tempo de execução, em seguida em relação à taxa de acerto, a utilidade das informações, o espaço de busca e dificuldade das atividades. Estas discussões são baseadas nas observações obtidas nos quatro experimentos e pretendem responder às perguntas de pesquisa deste trabalho, apresentadas na Seção 3.2.1.

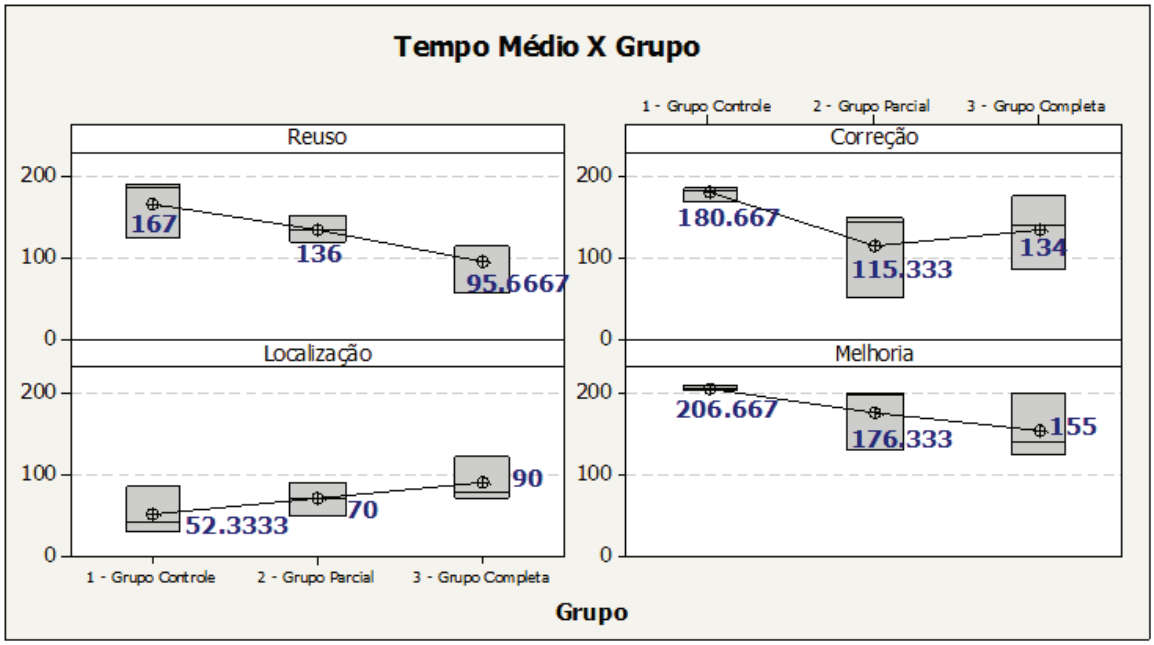



Figura 4.25: Gráficos com o tempo médio gasto pelos grupos nos experimentos.

### 4.5.1 Sobre o Tempo

Sobre a variável dependente tempo foi formulada a seguinte pergunta:

*P1) O uso da abordagem torna a resposta à compreensão do sistema desconhecidos mais rápida, ao se comparar com um procedimento tradicional de compreensão do sistema?*

Conforme apresentado nas observações referentes ao tempo dos experimentos na seção anterior e na Figura 4.25, os experimentos Reuso, Correção e Melhoria apresentaram uma média de tempo de execução menor para os grupos *Parcial* e *Completa* quando comparadas ao grupo *Controle*, que utilizou uma abordagem tradicional. Porém, no terceiro experimento, de Localização, este comportamento não se repetiu, observou-se que a média dos tempos dos grupos que utilizaram a abordagem neste experimento foi superior ao grupo *Controle*. Apesar da média do tempo grupo *Parcial* ter sido maior do que o grupo *Controle*, em função dos desvios não é possível afirmar que esta última tenha sido sistematicamente melhor. Entretanto, o grupo *Completa* teve um desempenho nitidamente pior em relação ao grupo *Controle* no experimento de Localização. Este comportamento diferente foi explicado nas observações da Seção 4.3.1 e se deve à localização solicitada ter sido facilitada pela estrutura do sistema. Com isto o grupo *Controle* apresentou o melhor tempo, agravado pelo fato que os tempos dos grupos *Parcial* e *Completa* contemplam os minutos gastos com a execução da abordagem. É importante observar que o *JHotDraw*, também, foi o sistema alvo da atividade de Correção e apesar se ser bem estruturado, os resultados não foram semelhantes ao do experimento de Localização, pois as classes e métodos onde deveria ocorrer a correção não eram de fácil localização e exigiam maior compreensão do sistema.

Em resumo, pode-se observar um ganho efetivo no uso da abordagem no primeiro (*Completa*), no segundo e quarto estudo. Entretanto, no terceiro estudo a abordagem não se mostrou melhor em função da natureza da manutenção requerida e da estruturação do sistema. E, neste caso, a sobrecarga do uso da abordagem tende a tornar a tarefa mais lenta desnecessariamente. E ao comparar o grupo *Parcial* e *Completa*, não foi possível concluir qual grupo apresenta melhor desempenho em relação ao tempo de execução, pois o grupo *Completa* obteve melhor resultado nos experimentos de Reuso e Melhoria, enquanto o grupo *Parcial* obteve melhores resultados nos experimentos de Correção e Localização. Além do mais, ocorreu uma pequena variação entre a maioria dos resultados sobre tempo destes grupos, não possibilitando aferir qual obteve melhores resultados.

### 4.5.2 Sobre a Taxa de Acerto

Sobre a variável dependente taxa de acerto foi formulada a seguinte pergunta:

*P2) Utilizando a abordagem, as respostas dadas pelo sujeito para uma atividade de manutenção em um sistema desconhecido são menos erradas?*

Em todos os experimentos, os grupos *Parcial* e *Completa* apresentaram maiores médias nas taxas de acerto do que os grupos *Controle*, porém, novamente, os experimentos de Reuso, Correção e Melhoria apresentaram um resultado mais expressivo, favorecendo o uso da abordagem. As taxas de acerto dos quatro experimentos são apresentadas na Figura 4.26, e observa-se que no primeiro experimento, Reuso, todos os participantes dos grupos *Parcial* e *Completa* concluíram a atividade, e um dos três participantes do grupo *Controle* não conseguiu concluir a atividade com sucesso. Em relação ao segundo experimento, Correção, nenhum participante do grupo *Controle* obteve sucesso na atividade e todos os participantes dos grupos que utilizaram a abordagem obtiveram sucesso. Este é um resultado bastante significativo. No quarto experimento (Melhoria), nenhum membro do grupo *Controle* completou a atividade, porém realizou alterações necessárias para a conclusão do requisito que foram pontuadas e apresentou uma taxa de acerto média de 30%. Neste mesmo experimento, nos grupos *Completa* e *Parcial*, 66.66% dos participantes completaram a atividade com sucesso e 33.33% não concluíram a atividade, mas executaram alterações que compõem a taxa de acerto destes grupos, o grupo *Parcial* obteve 83% de taxa de acerto e grupo *Completa* obteve 80%. No experimento de Localização, os grupos que utilizaram a abordagem apresentaram uma média de taxa de acerto superior, porém, com uma variação pequena entre os resultados do grupo *Controle* em comparação aos que utilizaram a abordagem. Devido aos desvios, neste experimento não se pode afirmar que existe uma diferença significativa entre os grupos. Como já foi explicado na pergunta anterior, a atividade de localização foi facilitada pela estrutura do sistema, levando à resultados que não foram expressivamente melhores para os grupos que utilizaram a abordagem.

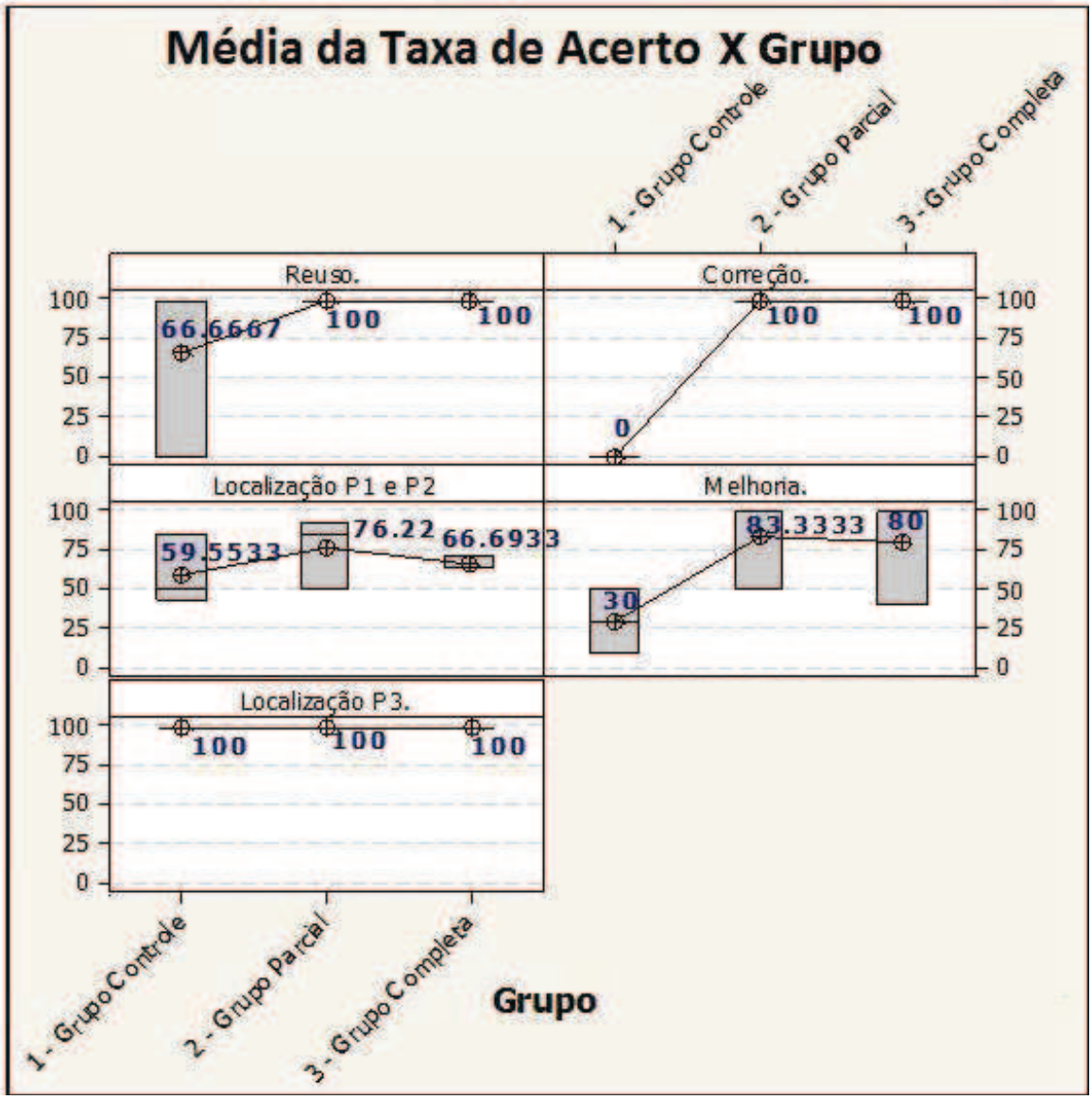



Figura 4.26: Gráficos com a taxa de acerto médio dos grupos nos experimentos.

É importante lembrar que foi estabelecido um tempo limite para execução das atividades, e que provavelmente com mais tempo de experimento, os sujeitos participantes que não executaram as atividades de manutenção com sucesso poderiam tê-lo feito, porém, com um tempo ainda maior.

Contudo, podemos concluir que utilizar a abordagem para uma atividade e manutenção em um sistema desconhecido pode levar a melhores taxas de acerto (soluções menos erradas). Nos quatro experimentos realizados, esta foi a conclusão, porém, as diferenças nesta taxa de acerto variam para diferentes tipos de atividades e sistemas, quando se compara o uso da abordagem com uma abordagem tradicional. Em alguns casos, como o do experimento de Localização em que a atividade é facilitada pela boa estrutura do código fonte, a diferença pode ser mínima, não justificando assim o uso da abordagem. Também foi observado que o uso das abordagens se mostra efetivo em relação à taxa de

acerto, especialmente se considerarmos a atividade de Correção e Melhoria. Em relação, às demais atividades não se observou uma melhoria efetiva.

### 4.5.3 Sobre a Utilidade das Informações

Diante da necessidade de se saber a utilidade das informações fornecidas pela abordagem, se estas realmente auxiliaram na compreensão de características de sistemas, algumas perguntas foram realizadas via questionários dos experimentos, todas visando responder a seguinte pergunta:

*P3) As informações fornecidas nas visões da abordagem foram úteis nas atividades solicitadas nos experimentos?*

As respostas dos sujeitos nos questionários deram origem às observações sobre a utilidade das informações em cada experimento, apresentadas nas seções anteriores. Com base nestas observações e considerando os quatro experimentos, conclui-se que 95.83% dos participantes demonstraram satisfação ao utilizar a abordagem, 100% afirmou que esta foi útil para a atividade solicitada direcionando a compreensão do sistema ao fornecer a localização dos elementos de código que implementas as características.

Sobre as visões, perguntas sobre quais visões e como estas realmente auxiliaram na atividade fizeram parte dos questionários, e a partir das respostas apresentadas pelos participantes, constatou-se que 100% dos participantes (grupo *Controle* e *Parcial*) que geraram o Mapeamento, a consideraram útil para atividade, pois, esta direciona a busca pelos elementos de código. As visões Mapeamento com Classificação, Classificação das Classes e Grafo de Chamadas foram consideradas úteis por 25% dos participantes que as geraram (grupo *Parcial*). Logo, o que se pode concluir sobre as visões é que o Mapeamento se fez claramente útil para todos os participantes, enquanto as demais visões somente foram consideradas úteis para algumas atividades e para um grupo pequeno de pessoas. Este resultado pode ter sido causado pela dificuldade de compreensão por parte do sujeito sobre como as informações destas três visões pode auxiliar nas atividades de manutenção, o que pode ter sido gerado por falhas nos treinamentos ou por que estas não contribuíram substancialmente para as atividades executadas.

Outra informação útil que deve ser levada pra trabalhos futuros, é que a abordagem foi integrada ao IDE Eclipse para possibilitar a análise estática dos dados, propiciando informações adicionais que levam à compreensão do sistema. Neste sentido, as ferramentas do Eclipse foram úteis para a compreensão dos sistemas. Segundo os sujeitos, as visões da abordagem foram úteis para direcionar a busca pelos elementos de código de interesse e localizá-los, porém outras informações geradas pelo Eclipse foram úteis para melhorar a compreensão sobre o comportamento do sistema e levar a compreensão. Logo, a integração com o Eclipse e o uso de suas ferramentas foi bastante útil para refinar e complementar as informações dinâmicas geradas pelas visões levando à compreensão do sistema.

Concluindo, as informações da visão Mapeamento gerada pela abordagem foram consideradas úteis pelos participantes para a localização das características. Em relação às outras visões não se obtém uma conclusão segura sobre a utilidade das mesmas. Mas na maioria dos casos, somente esta informação não leva a compreensão necessária para a execução da atividade, onde o uso de informações estáticas se faz útil. A satisfação demonstrada pelos participantes ao utilizar a abordagem reforça a hipótese que esta contribui nas atividades que envolvem compreensão de sistemas

### 4.5.4 Sobre a Dificuldade da Atividade

Outra variável qualitativa medida e obtida através dos questionários foi o grau de dificuldade atribuído, pelos participantes, a cada atividade solicitada nos experimentos. Esta variável foi indagada, pois pode indicar se com o uso da abordagem, os sujeitos tiveram menor ou maior dificuldade na execução da atividade. Para medir esta variável a seguinte pergunta de pesquisa foi elaborada:

*P4) Os grupos apresentaram variação expressiva quanto ao grau de dificuldade atribuído à atividade? Qual grupo atribuiu maior grau de dificuldade às atividades?*

Ao solicitar a opinião dos participantes sobre o nível de dificuldade da atividade, estes apresentaram respostas variadas que separadas por experimento não levaram às conclusões, porém, quando agrupados os dados dos 4 experimentos, a conclusão é mais robusta. Para melhorar a compreensão destas respostas, foram montados gráficos que apresentam de maneira visível os resultados. A Figura 4.27 apresenta estes gráficos com os resultados observados sobre o nível de dificuldade das atividades, segundo os sujeitos participantes. Como se pode notar nos gráficos, os grupos que utilizaram a abordagem classificaram as atividades em sua maioria como mediana e fácil, enquanto o grupo *Controle* classificou como difícil e mediana, o que demonstra uma possível maior dificuldade enfrentada pelo grupo *Controle* para executar a atividade quando comparada ao grupo *Parcial* e *Completa*. Foi notado que as atividades onde o grupo *Controle* classificou, em maioria, como difícil, foram justamente as atividades de Correção (100% difícil) e Melhoria (67% difícil), cujos grupos que utilizaram a abordagem apresentaram um desempenho muito superior e apenas 17% dos participantes as classificaram como difíceis, apresentado maioria das opiniões como atividades medianas.

O que se pode concluir destes resultados é que os participantes dos grupos *Controle* consideraram as atividades com nível de dificuldade maior, provavelmente por que tiveram maior dificuldade para executá-las. Este menor nível de dificuldade percebido pelos grupos *Parcial* e *Completa* pode ter sido ocasionado pelo uso da abordagem, que é a única variável controlada diferente entre os grupos. Lembrando que todos os grupos foram montados de maneira homogênea e os sujeitos selecionados são em sua maioria profissional inseridos no mercado de trabalho, que são acostumados com atividades de manutenção em sistemas, e

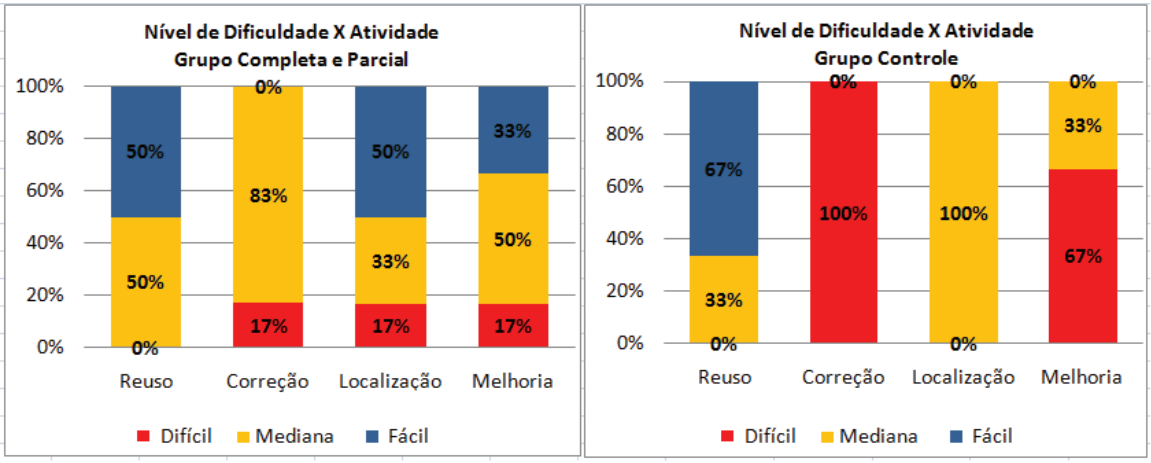



Figura 4.27: Gráficos com o nível de dificuldade das atividades conforme parecer dos participantes dos experimentos.

exceto pelo uso ou não da abordagem, foram submetidos a fatores e condições de trabalho semelhantes.

Um menor grau de dificuldade apontado pelos participantes dos grupos que utilizaram a abordagem é indicador adicional de que o objetivo desta abordagem, em auxiliar na compreensão de sistemas tenha sido cumprido.

### 4.5.5 Sobre o Espaço de Busca

A abordagem proposta, por meio de suas visões, pretende direcionar a busca pelos elementos de código ligados às características de interesse e assim contribuir para a compreensão do código fonte. Um dos fatores que possivelmente possibilita este direcionamento é a redução do espaço de busca inicial, por meio das visões que apresentam as características de interesse e a rastreabilidade para o código fonte. Acreditando que esta redução do espaço de busca inicial direciona o analista e pode influenciar no desempenho dos participantes foi formulada a seguinte pergunta:

*P5) Quando ocorre redução do espaço de busca inicial, também, ocorre redução no tempo de resposta e aumento na taxa de acerto do analista em atividades de manutenção em sistemas desconhecidos?*

Conforme se pode observar na Figura 4.28 e nas observações 5.1, os quatro experimentos realizados apresentaram reduções no espaço de busca inicial, uma vez que as visões apresentaram menos elementos de código do que o todo (código fonte) e estes são direcionados pelo interesse. Porém, os experimentos de Reuso e Localização não apresentam reduções expressivas, quando comparadas aos experimentos de Correção e Melhoria que apresentaram reduções no espaço de busca com uma média de 5.89% do espaço de busca original. Os gráficos do experimento Reuso, Correção e Melhoria da Figura 4.28 apresentam as médias dos espaços de busca iniciais sobre o número de métodos, que é a menor granularidade de elementos de código utilizados nestes experimentos. O gráfico do ex-

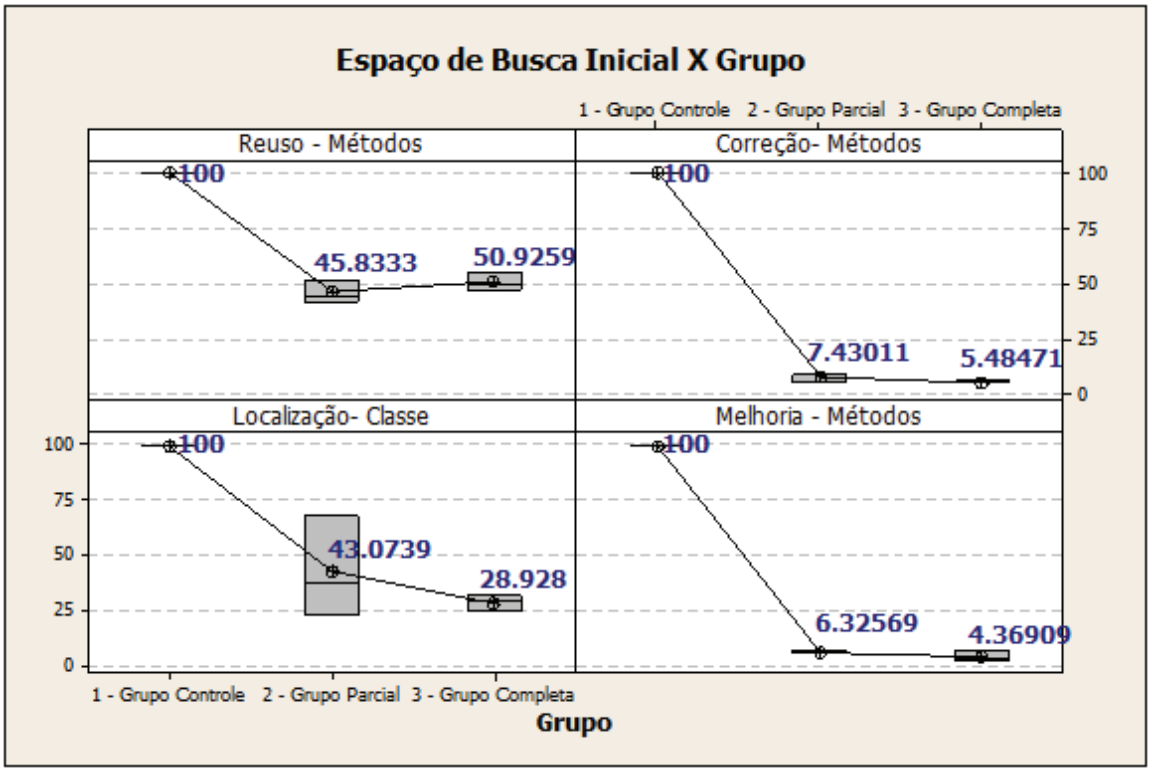



Figura 4.28: Gráficos com espaço inicial médio dos grupos nos experimentos/atividades, considerando o menor nível de granularidade dos elementos de código apresentados no primeiro contato com o sistema como espaço de busca.

perimento Localização da Figura 4.28 apresenta esta média calculada sobre as classes do código fonte, que para este experimento foi a menor granularidade utilizada pelos participantes, porém, os valores sobre os espaços de busca iniciais por métodos são conhecidos e foram 12.88% para o grupo *Parcial* e 8.27% para o grupo *Completa*.

A Figura 4.29 contempla os gráficos com as curvas das médias de redução do espaço de busca inicial, taxa de acerto e tempo gasto pelos grupos. Este gráfico tem a intenção de apenas apresentar o comportamento destas variáveis permitindo a comparação destes comportamentos entre os experimentos. Lembrando que estas variáveis possuem desvios apresentados nos gráficos de tempo, taxa de acerto e espaço de busca para cada experimento, que devem ser considerados nesta discussão. Com base nos gráficos da Figura 4.29 e observações 5.3 das Seções 4.1.5, 4.2.5, 4.3.5 e 4.4.5 deste trabalho, observa-se o comportamento semelhante nas curvas dos gráficos para os experimentos de Reuso, Correção e Melhoria. Assim, nota-se que os grupos *Parcial* e *Completa* apresentam espaços de busca inicial médio menor, tempo médio gasto com as atividade menor e taxa de acerto médio maior.

Porém, o experimento de Localização apresentou um comportamento diferente dos demais, pois, apesar do espaço de busca inicial ser 100% do código fonte para o grupo *Controle*, a busca foi facilitada pela estrutura do código que retorna a maioria dos elementos procurados nesta atividade através das ferramentas presentes no IDE Eclipse,

conforme já explicado. Além do mais, o espaço de busca inicial para os grupos *Parcial* e *Completa* não foram expressivamente reduzidos para este experimento, que exigiu a execução de muitas características. Como consequência, o tempo médio de execução do grupo *Controle* foi menor, principalmente devido ao tempo de execução da abordagem embutido nos minutos contemplados para o grupo *Parcial* e Completo. Porém, a taxa de acerto foi menor para o grupo *Controle*, mas foi um valor muito próximo ao grupo *Completa*, não sendo possível concluir qual abordagem foi melhor.

Contudo, o que se pode concluir é que a abordagem reduz o espaço de busca inicial, e que o uso da abordagem implica em taxas de acerto e tempos de resposta melhores ou iguais ao uso de uma abordagem tradicional, sendo melhor em 3 dos 4 experimentos aqui apresentados. Os experimentos de Correção e Melhoria, que apresentaram as reduções do espaço de busca mais expressivas, foram os que apresentaram as melhores performances dos grupos que utilizaram a abordagem, com as taxas de acerto e tempos médios expressivamente melhores do que o grupo *Controle*. Porém, para algumas atividades e sistemas, mesmo com a redução do espaço de busca inicial, os resultados não são melhores. Isto provavelmente ocorre quando a característica se encontra organizada no código fonte, como no experimento Localização. O que reforça o uso da abordagem em atividades onde o interesse se encontra disperso e/ou entrelaçado, dificultando a atividade, sendo este o problema que deu origem à abordagem proposta.

Pensando na qualidade do espaço de busca inicial, ou seja, informações contidas nas visões da abordagem, e sabendo que elementos falsos positivos e falsos negativos podem ser apresentados nas visões, devido a um mau roteiro de coleta dos rastros, ou por ocorrência de eventos invisíveis a observação do usuário, elaborou-se a seguinte pergunta:

*P6) A presença de elementos de código falsos positivos e a ausência de elementos de interesse (falsos negativos) nas visões interferem no desempenho, taxa de acerto e tempo de resposta, do analista que utiliza a abordagem?*

Para responder esta pergunta, são analisados os dados dos gráficos presentes nas Figuras 4.30, 4.4, 4.10, 4.16 e 4.22, que em conjunto apresentam o percentual de falsos negativos, médias das taxas de acerto e médias do tempo de resposta de cada experimento. Conforme se pode notar nestes, o experimento de Localização foi o que apresentou a maior taxa de falsos negativos presentes nas visões da abordagem, sendo uma média de 28.57% das classes de interesse para o grupo *Parcial* e 26.19% para o grupo *Completa*. O experimento de Reuso apresentou uma média 0.00% das classes e 16.67% dos métodos no grupo *Parcial* e 11.11% das classes e 35.45% dos métodos de interesse para o grupo *Completa*. Para os experimentos de Correção e Melhoria não tiveram elementos falsos negativos.

O que se observou individualmente nos experimentos é que nos de Reuso e Localização o número de falsos negativos não impactou a ponto de tornar a abordagem pior quando comparada ao grupo *Controle*. Nos experimentos de Correção e Melhoria, não ocorreu a presença de falsos negativos, logo não geraram impactos. Porém, quando se analisa o

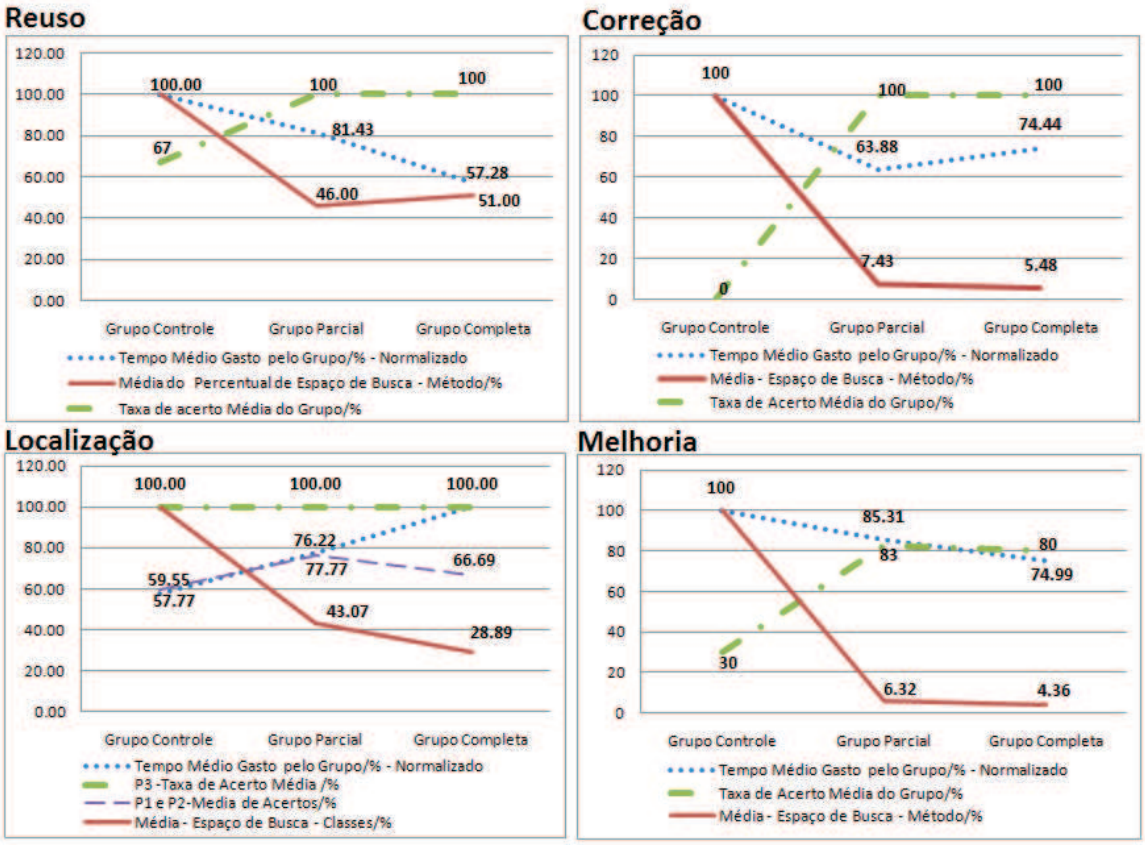



Figura 4.29: Gráficos de tempo versus espaço de busca, taxa de acerto e grupo dos 4 experimentos realizados.

conjunto de experimentos, se percebe que os experimentos que não apresentaram falsos negativos foram os que apresentaram os melhores desempenhos em termos de tempo e taxa de acerto.

A presença de elementos falsos negativos e falsos positivos nos resultados gerados por análise dinâmica é um fator problemático em diversos trabalhos. Alguns trabalhos inclusive propõem soluções para tratar a presença deste tipo de elemento. Como uma solução, muitos trabalhos relacionados optaram pelo refinamento utilizando análise estática, como este trabalho, que utilizou a análise estática através das ferramentas do IDE Eclipse disponibilizadas. O uso da análise estática para obter os elementos falsos negativos foi efetivo nesta localização para os experimentos de Reuso, porém não foram efetivas no experimento de Localização, pois apenas 33.33% dos sujeitos que utilizaram a abordagem descobriram os FN, por decidirem buscá-los. Os demais confiaram nas visões e apresentaram apenas os elementos listados nestas. Porém, mesmo com este resultado, os grupos que utilizaram a abordagem apresentaram mais taxa de acerto na localização das classes das características de interesse, apesar de ter ocorrido uma variação muito pequena nesta taxa ao comparar com o grupo *Controle*.

A presença de falsos negativos não é um ponto positivo da abordagem. Porém, o que se percebe é que mesmo com sua ocorrência, o uso da abordagem se fez útil, já que não

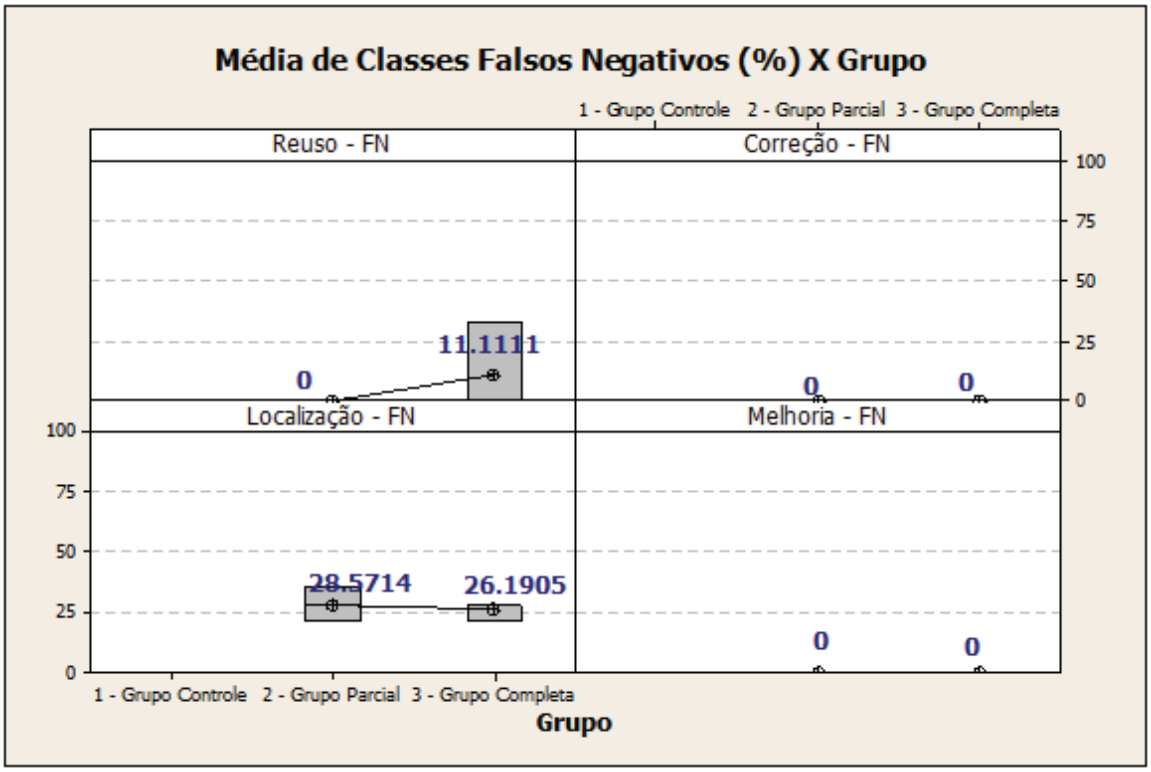



Figura 4.30: Gráficos com o percentual de classes falsas negativas presentes nas visões dos grupos nos experimentos.

apresentou nenhum resultado neste trabalho em que se pode afirmar que os resultados foram piores do que o uso de uma abordagem tradicional. Mas a falta destes elementos faz com que o analista tenha que despender tempo para localizá-los, e com isso prejudica o seu desempenho. Porém, provavelmente, as visões auxiliam na localização dos elementos faltantes. No entanto, se o percentual de falsos negativos fosse 100%, e nenhum elemento de interesse fosse apresentado nas visões, a consequência seria a inutilidade da abordagem proposta. Esta situação somente ocorreria se os roteiros não contemplassem nenhuma característica de interesse, por falha de definição por parte do analista. Outra observação é que o número de falsos negativos variou de participante para participante, dependendo do roteiro definido e executado por este, por exemplo, elementos que foram considerados falsos negativos por um sujeito, foram listados nas visões de outro sujeito para o mesmo experimento.

Os sujeitos participantes não foram indagados sobre os falsos positivos, uma vez que a presença destes seria dificilmente percebida por eles, que a partir dos elementos apresentados nas visões realiza a análise do código para filtrar os elementos de código que são de interesse para a atividade, ignorando assim muitos elementos, dentre este os falsos positivos. Porém, nos estudos preliminares foi avaliado este percentual de falsos positivos, a fim de verificar a precisão e acurácia da abordagem, e os resultados foram considerados bons, conforme se pode ver na Tabela 3.3. Como em todos os experimentos ocorreu uma taxa de acerto melhor para os grupos que utilizaram a abordagem, então, se pode afirmar

que o que o número de falsos positivos não impactou ao ponto de tornar a abordagem pior quando comparada ao grupo *Controle* neste quesito. Quanto ao desempenho de tempo, em três dos quatro experimentos, os tempos dos grupos que utilizaram a abordagem foram melhores e novamente a possível presença de falsos positivos, também, não impactou ao ponto de tornar pior o uso da abordagem quando comparada ao grupo *Controle*. Porém, a presença de falsos positivos pode impactar na localização dos elementos de interesse nas visões, conforme reclamação de alguns sujeitos do experimento de Melhoria, onde ocorreu a listagem de um volume considerável de elementos falsos positivos. Porém, neste experimento, o resultado dos grupos que utilizaram a abordagem foi superior.

Contudo, o que se pode concluir sobre a existência de elementos falsos positivos e negativos é que a presença de elementos falsos negativos e positivos. Provavelmente, influencia negativamente na localização dos elementos de interesse, na qualidade do espaço de busca apresentado pelas visões. Esta conclusão é reforçada pelo fato de que os experimentos que não apresentaram falsos negativos foram os que apresentaram melhor desempenho. Porém, mesmo com a possível ocorrência de elementos falsos negativos e positivos nas visões da abordagem, os resultados de tempo e taxa de acerto não foram piores para os grupos que utilizaram a abordagem, sendo inclusive melhores em 3 dos 4 experimentos.

Nesta seção foram apresentados os resultados gerais dos experimentos controlados, que responderam às perguntas sobre o tempo de execução, taxa de acerto, utilidade da abordagem, espaço de busca e nível de dificuldade. Estas respostas foram formuladas com base nos resultados dos 4 experimentos realizados, um total de 36 observações individuais sobre cada pergunta, que tornam mais robustas as discussões sobre as hipóteses deste trabalho, que serão apresentadas no próximo capítulo visando verificar se os objetivos deste foram atingidos.

# Capítulo 5

# Trabalhos Relacionados

Problemas de compreensão de sistemas, durante anos, motivou o aparecimentos de numerosas abordagens de análise dinâmica, com diferentes técnicas e ferramentas. Em [Cornelissen et al. 2008], foi realizada uma revisão sistemática com o objetivo de contextualizar e investigar este conjunto de propostas, permitindo caracterizar os esforços e melhorias no âmbito da análise dinâmica. Esta revisão foi utilizada como referência para a avaliação dos trabalhos aqui relacionados, e ofereceu uma visão panorâmica do que constitui as principais contribuições do campo, servindo como uma fonte de apoio para identificação de lacunas e oportunidades. O corpo de investigação deste trabalho é composto de 176 artigos selecionados criteriosamente. Os artigos foram classificados levando em consideração os quatro critérios abaixo apresentados:

1. **Atividade:** descreve o que está sendo realizado ou contribuição (por exemplo: análise de características);
2. **Alvo:** reflete o tipo de linguagem de programação ou plataforma para a qual a abordagem mostra-se aplicável (por exemplo: orientação a objeto);
3. **Método:** descreve os métodos de análise dinâmica que são utilizados na realização da atividade (por exemplo: visões e/ou filtragens);
4. **Avaliação:** define o modo que o trabalho foi validado (por exemplo: estudos de caso utilizando sistema de código aberto).

Todos os 176 artigos foram abrangentemente avaliados para seleção dos artigos com maior grau de relacionamento com este trabalho, levando em consideração os critérios apresentados (atividade, alvo, método e avaliação), seus títulos, resumos e número de citações. Com base nestes dados foram selecionados os mais relacionados ao trabalho aqui apresentado. Alguns outros trabalhos estudados durante o desenvolvimento deste trabalho, que contribuíram para este e foram considerados relevantes, também, serão apresentados nesta seção.

## 5.1 Wilde e Scully

Wilde e Scully [Wilde e Scully 1995] foram pioneiros no tema de localização de características em 1995 com sua ferramenta Reconhecimento de Programas (*Software Reconnaissance*), estabelecendo as relações entre as características e o código fonte. Wilde e colegas continuaram a investigação nesta área nos anos seguintes, com um forte foco na avaliação [Wilde et al. 2003, Wilde et al. 2001, Wilde e Casey 1996]. Em [Wilde et al. 2003] apresentaram uma comparação entre três abordagens que possuem o mesmo objetivo que o trabalho corrente: localizar o código que implementa uma característica específica de um programa para corrigir um problema ou acrescentar uma funcionalidade. Este trabalho descreve um estudo de caso para localizar duas características em uma amostra de código mal estruturado de um sistema legado desenvolvido em Fortran. Foram comparados os desempenhos das abordagens *Software Reconnaissance*, Grafo de Dependências e o conhecido *Grep*. Nos resultados, tanto o método *Software Reconnaissance* como o Grafo de Dependências localizaram as duas características de interesse, apesar de algumas dificuldades encontradas e adaptações necessárias devidas à própria natureza emaranhada do código. O *Grep*, que suportou pesquisa por método funcionou bem para uma das duas características, mas foi ineficiente para a segunda, que era mais complexa.

O método *Software Reconnaissance* usa casos de testes para localizar as características e é baseada na comparação dos rastros de execução destes diferentes casos de teste. O sistema em avaliação é instrumentado de modo a coletar os rastros de execução a partir de casos de testes, os quais são conduzidos com as características desejadas e outros sem as características desejadas, a fim de se obter os elementos de código que são específicos da característica em análise. Os rastros são analisados para procurar os blocos ou decisões que são executadas com a característica de interesse. O método *Software Reconnaissance* foi experimentado com eficácia em código C de um certo número de empresas, e, é bastante citado pelas literaturas correlatas, devido ao pioneirismo deste método.

O método de Grafo de Dependências para localização de característica foi descrito por Chen e Rajlich [Chen e Rajlich 2000]. Este método opera sobre funções e variáveis globais, assim como, chamadas de funções e fluxos de dados do código de um sistema alvo. O método envolve uma pesquisa sobre as dependências dos componentes gráficos, que tem início em um nó de partida, e na ausência de outras informações utiliza a função main(). Em cada etapa um componente (geralmente uma função C) é selecionado para uma visita. O mantenedor identifica, manualmente, características utilizando este método. Ele analisa o código fonte, a dependência gráfica, e a documentação para entender o componente e decidir se o componente é relevante ou não para a característica. A pesquisa, em seguida, procede para outro componente até que todos os componentes relacionados com a característica são encontrados e entendidos. Esta abordagem utiliza apenas dados

estáticos e busca manual para identificação das características, o que não é recomendável para sistemas grandes, *multi-treaded* e orientados a objetos.

O terceiro método analisado utilizou o *Grep* para localizar o código fonte que implementa as características. *Grep* é um aplicativo para linha de comando de sistemas que faz buscas no conteúdo dos arquivos. Este escaneia os arquivos linha por linha procurando pelos argumentos especificados na sintaxe do comando. Todas as linhas contendo aquele argumento serão impressas na saída padrão.

A abordagem apresentada nesta dissertação possui pontos em comum tanto com método de Grafo de Dependências quanto com o *Software Reconnaissance*. Já o método *Grep* apresenta uma metodologia totalmente diferente, pois esta realiza buscas por palavras nas linhas de código para apontar os possíveis elementos que implementam a característica. O método *Software Reconnaissance* usa casos de testes (roteiros) para obter os rastros de execução e localizar as características, de maneira semelhante à abordagem apresentada nesta dissertação. A partir destes rastros criam-se as visões que irão auxiliar na compreensão das características, dentre estas um mapeamento que possui o mesmo intuito do mapeamento apresentado nesta dissertação, porém a estrutura e técnica para definição dos elementos de código que implementam a característica se diferem. Wilde e Scully propõem a execução do rastro com e sem a característica para a partir desta informação definir o que é relevante para a característica e apresentar. Na abordagem apresentada nesta dissertação, o usuário é livre para executar os rastros desta maneira, ou somente com a característica de interesse fazendo com que a visão apresente todos os elementos executados e associando-os à característica. Outra diferença entre as visões é que o método *Software Reconnaissance* lida com uma característica por vez e não foca no relacionamento entre estas como na abordagem apresentada nesta dissertação. A visão Mapeamento desta dissertação possui integração com o IDE Eclipse para viabilizar o refinamento da informação dinâmica por meio da análise estática, enquanto o método *Software Reconnaissance* não adota o uso de informações estáticas. A abordagem corrente fornece um mapeamento das características para o código fonte assim como a *Software Reconnaissance*, acrescido de mais três visões obtidas através das informações dinâmicas, que podem ser refinadas por informações estáticas, o que não é feito na abordagem de Wilde e Scully.

O método de Grafo de Dependência e a abordagem apresentada neste trabalho possuem em comum o objetivo de localizar características no código fonte. Porém, a abordagem proposta por Chen e Rajlich utiliza apenas análise estática, enquanto a abordagem proposta nesta dissertação utiliza análise dinâmica refinada pela análise estática. Um ponto em comum, é que ambas apresentam as chamadas realizadas entre as unidades computacionais (visão Grafo de Chamadas do trabalho corrente). Porém, a abordagem de Chen e Rajlich utiliza análise estática para obter esta informação e a abordagem desta

dissertação utiliza as chamadas dos métodos apresentadas no rastro de execução (análise dinâmica), com filtro para selecionar os métodos de interesse.

O trabalho [Wilde et al. 2003] de Wilde e colegas contribuíram para o trabalho desta dissertação ao apresentar um panorama comparativo sobre as três abordagens *Software Reconnaissance*, Grafo de Dependências e *Grep*, por meio de um estudo de caso que aponta os pontos positivos e negativos de cada abordagem. Estes pontos foram considerados na definição das visões da abordagem desta dissertação.

## 5.2 Simmons

Em [Simmons et al. 2006], também é apresentada uma abordagem para o problema da localização de características. Os autores afirmam que a maior parte dos métodos para resolver este problema envolve a geração e análise de rastros de execução. Eles apresentam um trabalho exploratório realizado em parceria com a Motorola, que estava interessada em estabelecer quais passos seriam necessários para localização de características utilizando ferramentas comerciais conhecidas atualmente e utilizadas no seu domínio. Um estudo foi conduzido para identificar eventuais obstáculos que possam dificultar o uso destas ferramentas para a localização de características em seus sistemas. Inicialmente, os autores concluíram que para localizar e entender uma característica, precisa-se de um programa que suporte três passos: análise dinâmica, diferenciação dos rastros e análise estática. Com base nestes três pontos foram analisadas as ferramentas comerciais *Metrowerks CodeTEST* e *Klocwork inSight*, as quais foram integradas com a *TraceGraph* [Lukoit et al. 2000], uma ferramenta acadêmica de comparação de rastros. Esta integração foi testada utilizando um sistema de código aberto do mesmo domínio das aplicações para o qual a Motorola pretendia utilizá-las, o *Apache Web Server* (227 KLOC). O objetivo principal do estudo de caso desenvolvido foi identificar os obstáculos que poderiam tornar a integração destas ferramentas não aplicável para a localização de características. Como resultado, a combinação das ferramentas foi eficaz na localização, compreensão e documentação das características. E os participantes do estudo concluíram estes passos em um prazo médio de 3 a 4 horas por funcionalidade, estudando apenas poucas centenas de linhas de um sistema com mais de 200.000 linhas.

Comparando o estudo de Simmons e colegas com esta dissertação, destacam-se os seguintes pontos: os objetivos dos trabalhos se diferem, pois o trabalho de Simmons e colegas se constitui em um trabalho exploratório para análise e integração de ferramentas para localização de características utilizando análise dinâmica, enquanto o trabalho corrente propõem uma abordagem (ferramentas e procedimentos) para a localização das características e realiza estudos controlados com grupos de sujeitos para avaliar o uso desta em comparação a um grupo que não a utiliza, por meio de atividades de manutenção reais. Porém, em ambos os trabalhos, o problema tratado é o mesmo, localização de caracte-

rísticas e melhoria da compreensão do sistema utilizando análise dinâmica. O uso de visões foi escolhido por ambas as abordagens como método utilizado para viabilizar a compreensão do sistema. Em ambos os trabalhos utiliza-se casos de teste (roteiros) para realizar a coleta dos rastros e viabilizar a análise dinâmica do sistema alvo. O processo de coleta dos rastros é semelhante entre as abordagens, porém na abordagem de Simmons e colegas se deve ter um teste que contempla a característica desejada e outro que não a contempla, assim como em [Wilde e Scully 1995]. Simmons e colegas criaram um modelo arquitetural que apresenta os componentes do código que são relevantes para uma característica. Na abordagem apresentada nesta dissertação, a visão Mapeamento foi criada com o mesmo intuito de documentar e possibilitar a visualização dos elementos de código que implementam uma característica, porém, esta utiliza uma estrutura de árvore para apresentar as características e as classes e métodos que as implementam. A principal contribuição do trabalho de Simmons e colegas para o trabalho apresentado nesta dissertação é a definição dos três principais passos para localização de características: análise dinâmica, diferenciação dos rastros e análise estática.

## 5.3 Lukoit e Wilde

Em [Lukoit et al. 2000], os autores apresentam a ferramenta *TraceGraph*, que dá suporte ao processo de localização de características, particularmente para sistemas grandes e interativos utilizando análise dinâmica. Esta propicia uma visualização simples dos rastros de execução de um sistema permitindo que o analista execute a característica de interesse e imediatamente visualize como a execução varia. *TraceGraph* combina dois conceitos: 1) *feedback* imediato durante a execução do programa (os arquivos de rastro são escritos e analisados continuamente, enquanto o programa executa); 2) visualização gráfica dos resultados, onde cada coluna correspondente a um período de tempo e cada linha a um componente do programa. A representação (visão) montada por Lukoit e colegas, apresenta uma matriz onde as células são pintadas de cinza se o componente foi executado naquele período de tempo ou branca caso contrário. Células pretas são usadas para enfatizar a primeira vez que um componente é executado. As diferenças entre componentes executados com e sem a característica podem ser facilmente distinguidos pelo padrão de células. *TraceGraph* traz informações a respeito de "quando"determinado componente foi executado durante a coleta dos rastros. Segundo os autores, com base nestas informações o analista poderá compreender comportamentos do sistema alvo. Os dois estudos de caso realizados indicaram que esta abordagem pode prover a efetiva localização das características no sistema alvo. *TraceGraph* compartilha algumas das limitações da técnica *Software Reconnaissance* [Wilde e Scully 1995] tais como: sua aplicabilidade restrita as características controláveis pelo usuário, excluindo assim o acesso a elementos "comuns"no código; e dependência dos dados de entrada, o que pode ocasionar problemas

com falso positivos e com elementos não cobertos pelos em função do roteiro de execução escolhido.

Comparando o trabalho de Lukoit [Lukoit et al. 2000] com o apresentado nesta dissertação, percebe-se que ambos tratam a localização de características utilizando rastros de execução para apresentar visões que pretendem melhorar a compreensão do sistema. Lukoit e colegas realizam a coleta dos rastros com um caso de teste onde a característica é contemplada e outro onde esta não é contemplada, assim como a abordagem apresentada em [Wilde e Scully 1995], enquanto na abordagem corrente, este procedimento é opcional, dependendo da necessidade do analista. As visões propostas por ambas as abordagem se diferem na estrutura de representação da informação e no tipo de informação, apesar de ambas utilizarem o tempo de execução para apontar o código que implementa a característica. Lukoit e colegas utilizam uma matriz, marcada com cores para indicar quais elementos de código foram executados em um determinado tempo de execução, e apontar se este implementa a característica de interesse. Enquanto, a abordagem corrente apresenta os elementos de código que implementam a característica separadamente por meio de uma árvore com as respectivas associações, além de outras visões com informações sobre as chamadas entre métodos (na forma de grafos) e a participação das classes na implementação do conjunto de características (na forma de lista). A informação de tempo e ordem de execução não foi utilizada nas visões da abordagem desta dissertação. A maioria das sugestões feitas pelos analistas que utilizaram a abordagem corrente está relacionada a se ter uma visão que apresente a sequência das chamadas em ordem temporal, o que se torna uma proposta para trabalhos futuros. Outro ponto importante, é que as limitações apresentadas pela técnica *Software Reconnaissance* e *TraceGraph* aqui apresentadas também se aplicam a abordagem corrente, devida a coleta dos rastros utilizando casos de teste e a associação do rastro coletado à característica de interesse, presente nas duas abordagens. E este tipo de problema é característico das técnicas que utilizam análise dinâmica.

## 5.4 Eisenbarth, Koschke e Simon

Em [Eisenbarth et al. 2002] e [Eisenbarth et al. 2003], Eisenbarth e colegas apresentam uma abordagem para localização de característica baseada em análise dinâmica, estática e em reticulados conceituais [1], sendo consideravelmente citada na literatura. O primeiro passo desta abordagem é a definição dos cenários baseados nas características de interesse, estes são documentados na forma de casos de teste. Um cenário pode ser composto por mais de uma característica. O segundo passo é a extração dos grafos de dependência utilizando análise estática do sistema. Logo depois, inicia-se a análise dinâmica com a coleta do rastro e obtenção do reticulado conceitual, que apresenta o relacionamento das

[1] do inglês, *concept lattices*

unidades computacionais com as características e destas com os cenários. O resultado apresenta as unidades computacionais classificando-as conforme sua participação na implementação de um conjunto de características. Por exemplo, unidades computacionais que são específicas implementam apenas uma característica do conjunto. No último passo, a análise estática é aplicada, para refinar o resultado da análise dinâmica, como por exemplo, identificando e adicionando unidades computacionais que não foram listadas ou que foram mal classificadas devida a uma má definição do cenário. Para validação da abordagem, Eisenbarth e colegas realizaram estudos de caso apresentados em [Eisenbarth et al. 2001] e demonstraram os ganhos da abordagem e que a combinação de análise dinâmica refinada pela análise estática reduz o espaço de busca drasticamente.

A abordagem apresentada por Eisenbarth e a apresentada nesta dissertação possuem uma característica comum que é a localização de característica utilizando análise dinâmica refinada pela análise estática. Porém, em [Eisenbarth et al. 2002] pretende-se obter a arquitetura do sistema em análise, enquanto a abordagem corrente foi utilizada em atividades de manutenção fornecendo informações que fossem úteis para a compreensão do código fonte, não tendo sido utilizada para se obter a arquitetura do sistema. As abordagens possuem pontos em comum como a coleta dos rastros de execução para utilizá-los na definição dos elementos de código que implementam as características. A visão de mapeamento característica-unidade apresentada por eles é semelhante à visão Mapeamento desta dissertação. As visões Classificação das Classes e Mapeamento com Classificação desta dissertação tiveram seus critérios de classificação inspirados por este trabalho de Eisenbarth, porém, a abordagem desta dissertação não utiliza Análise de Conceitos para a definição dos relacionamentos entre as características e os elementos de código, nem para a classificação destes elementos quanto a sua participação na implementação das características. Como pontos em comum, a abordagem apresentada nesta dissertação possui os mesmos problemas apresentados por [30], o resultado gerado pela análise dinâmica necessita de refinamento quando a coleta dos rastros (cenários) é mal definida ou executada. Isto reforça que a ocorrência deste tipo de problema é inerente à análise dinâmica. Eisenbarth utiliza a análise estática para o refinamento das informações dinâmicas com a finalidade de corrigir e melhorar os resultados para obter a arquitetura. Para o trabalho desta dissertação, a análise estática deve ser realizada se necessária, ficando a cargo do analista avaliar a necessidade e qual tipo de informação se pode obter com o uso do IDE Eclipse que irá lhe auxiliar na atividade de manutenção completando a informação dinâmica. Este trabalho de Eisenbarth foi importante para confirmar que o refinamento da análise dinâmica, utilizando a análise estática realmente traz melhores resultados para a compreensão do sistema, reforçando a importância da integração realizada com o IDE Eclipse na abordagem corrente.

## 5.5 Eisenberg e Volder

Em [Eisenberg e Volder 2005], Eisenberg e Volder apresentam uma técnica automatizada para localização característica utilizando análise dinâmica dos rastros de execução, com o objetivo de auxiliar desenvolvedores no mapeamento das características para o código fonte relevante. Segundo os autores, a principal contribuição de seu trabalho é uma alternativa robusta, denominada Rastro Dinâmico de Características (*Dynamic Feature Traces* - DFTs), cuja principal característica é a aplicação de uma heurística para determinar a relevância de um elemento de código para uma característica ao invés da classificação binária de relevância de trabalhos anteriores. A DFT é composta de duas partes: 1) classificação: conjunto ordenado de todos os métodos chamados durante a execução do conjunto de testes de execução; 2) chamadas: conjunto de todas as chamadas de métodos observadas nos rastros de execução dos testes realizados. Esta informação ajuda a averiguar porque um método é relevante para uma característica. DFTs são criados em três etapas. Primeiramente, o desenvolvedor particiona o conjunto de testes. Em seguida, a ferramenta de DFT realiza a análise dos rastros de execução destes testes e cria as classificações e os conjuntos de chamadas. A pontuação do método é obtida pelo cálculo da média entre três heurísticas definidas (multiplicidade, especialização e profundidade). O desenvolvedor pode usar o *JQuery* [Janzen e De Volder 2003], um navegador de código, para visualizar o DFT e explorar com mais detalhes de forma incremental a base de código. Este trabalho foi validado por meio de um estudo comparativo com os métodos *Software Reconnaissance* e *Execution Slices* [Wong et al. 1999], que utilizam decisões binárias para definir quais são os elementos relevantes. Neste contexto, uma decisão binária pode ser formulada como uma expressão booleana em termos da inclusão nos conjuntos de casos de teste executados, normalmente, subtraindo os elementos presentes em casos de testes que contém o interesse dos presentes em outro teste que não contém o interesse. Os elementos de código resultantes desta subtração são chamados de UCOMPs [2], sendo conjuntos de componentes únicos que implementam exclusivamente um recurso exercitado e dão nome a esta técnica. Nesta comparação foram avaliados três sistemas de código aberto em Java. Estes experimentos foram realizados pelo próprio autor, e os resultados demonstraram os benefícios dos DFTs em comparação a UCOMPs, pois foram menos sensível em relação à qualidade da entrada e mais eficiente quando usado por desenvolvedores que não estão familiarizados com o sistema alvo. Após os experimentos, foi constatado que a técnica DFTs apresenta melhores resultados quando aplicado um conjunto grande de testes para a coleta dos rastros, onde todas as características relevantes são testadas em pelo menos um caso de teste. Foi observado também que roteiros mal definidos, que não cobrem perfeitamente as características, podem gerar resultados incompletos.

[2] do inglês, *Unique Compoonents*

Como se pode notar esta abordagem apresenta pontos em comum quando comparada a abordagem desta dissertação. Ambas tratam o mesmo problema, localização de característica para melhorar a compreensão do sistema. Ambas utilizaram a análise dinâmica de rastros de execução como técnica, ferramentas desenvolvidas em AspectJ e sistemas alvo orientados a objeto. Quanto aos passos definidos pelas abordagens, o uso dos casos de teste (roteiros) para coleta dos rastros, a geração de visões integradas a um browser ou IDE que possibilite a exploração dos elementos de código relacionados às características, também, são pontos em comum destas duas abordagens. Além disso, ambas apresentam o mesmo problema relativo aos roteiros mal definidos que podem gerar resultados incompletos, características em técnicas que utilizam análise dinâmica. A abordagem de Eisenberg e Volder e a corrente possuem características próprias. Eles têm como objetivo verificar se a abordagem DFTs apresenta resultados melhores (mais completos e corretos) ao comparar estes com outra técnica que utiliza classificação binária. A avaliação foi feita por meio de um estudo de caso em que se realiza a comparação com uma abordagem que utiliza classificação binária, verificando se os resultados gerados estão corretos. Por outro lado, o trabalho corrente pretende verificar se a abordagem proposta, que não utiliza heurística para classificação, auxilia o analista durante atividades que envolvem a compreensão de sistemas através das visões definidas. Além disso, a avaliação experimental foi conduzida com grupos de sujeitos humanos, onde variáveis como tempo e taxa de acerto foram medidos. Ambas apresentam visões com o mapeamento dos elementos de código que implementam as características de interesse e com as chamadas entre os métodos, porém, as estruturas destas visões são diferentes, o que inclui os filtros e a forma de apresentação, assim como as ferramentas utilizadas. Este trabalho de Eisenberg e Volder inspirou o uso de classificações na abordagem corrente. Porém, nesta classificação, os DFTs utilizam de uma heurística (classificação gradual) para definir a relevância de um elemento de código, enquanto, a abordagem desta dissertação classificou as classes relacionadas a uma característica utilizando um cálculo simples baseado no número de participações deste elemento no conjunto de características. Resumindo, são trabalhos que possuem como objetivo a localização de características utilizando análise dinâmica para melhorar a compreensão de sistemas, porém com abordagens e validações distintas. Este trabalho de Eisenberg e Volder foi importante para a definição do uso de classificações dos elementos de código na abordagem corrente, porém não foi utilizada a mesma técnica.

## 5.6 Greevy e Ducasse

O trabalho [Greevy et al. 2006] utiliza rastros de execução para gerar diferentes visões que pretendem auxiliar na compreensão das características. Os autores afirmam que ao analisar a evolução de um sistema, é necessário saber como e quais funcionalidades

foram modificadas, para recuperar tanto a intenção da mudança quanto a extensão, identificando quais artefatos são afetados por esta. O objetivo deste trabalho de Greevy e colegas foi mostrar que sobre a perspectiva de características de um sistema, compreender as razões subjacentes das mudanças realizadas é uma informação valiosa para a compreensão do sistema. Os autores combinaram modelos estáticos de código fonte com os modelos dinâmicos do comportamento de características para obter um mapeamento entre as características e código. Uma das visões propostas apresenta a análise da evolução do sistema, combinando os históricos de evoluções. Eles utilizam uma visualização simples para mostrar características como agrupamentos de entidades estruturais e apresentam as mudanças nas características para revelar a respectiva extensão e o tipo de alterações no código. Eles descrevem as mudanças de uma forma que revela quantas e quais características são afetadas pela mudança e classificam os elementos de código baseado do nível de participação na característica, assim como Wong et al. [Wong et al. 1999] e Eisenbarth [Eisenbarth et al. 2003]. O nível de participação depende da característica e do conjunto de características. Este é obtida através de um cálculo baseado no número de participações deste elemento no conjunto de características. Segundo eles, as principais contribuições foram: a descrição de uma nova abordagem para analisar a evolução de um sistema em termos de características; a caracterização das mudanças, de uma forma que reflita como as regras funcionais de sistemas mudam; a introdução de uma visualização simples das características, como um agrupamento de entidades de código (por exemplo, classes) para uma versão de um sistema.

Na perspectiva das características, os autores caracterizam a participação de elementos do código em características, de forma isolada ou em grupo. Os elementos são classificados em: elementos não cobertos pela análise, elementos específicos de uma característica, elementos compartilhados por um grupo de características e elementos compartilhados por todas as características. Na perspectiva de unidades computacionais, a existência de características em elementos de código é caracterizada indicando o nível de relação entre características a respeito dos elementos compartilhados. Assim como na abordagem de [Wong et al. 1999] e [Eisenbarth et al. 2003], os autores caracterizaram os artefatos de código fonte baseados no nível de participação nas características. Este nível de participação depende muito das características que estão em análise. Em [Greevy et al. 2006,Greevy e Ducasse 2005] é abordada a localização de características utilizando análise dinâmica visando à melhoria da compreensão de sistemas, assim como a abordagem desta dissertação. Porém, eles trabalham com as mudanças nas características para revelar a extensão e o tipo de alterações no código, apresentando quantas e quais características são afetadas pela mudança. Enquanto, a abordagem proposta nesta dissertação não define visões especificas para se trabalhar com mudanças de código em diferentes versões, mas permite que o analista avalie dois sistemas e indique as alterações conforme apresentado no experimento de localização (Seção 3.4.3). Sob o ponto de vista do impacto de uma

alteração em um código que implementa uma ou mais características, o trabalho deles trouxe contribuições para a abordagem proposta nesta dissertação, pois foram utilizados os critérios de classificação da participação dos elementos de código (única característica, grupo pequeno de característica, grupo grande de características ou de infraestrutura) como base para os critérios definidos para as visões Classificação das Classes e Mapeamento com Classificação.

## 5.7 Robillard, Murphy e Weigand

Em [Robillard e Murphy 2007] é abordado o mesmo problema desta dissertação e eles propõem a localização e documentação de interesses em forma de artefatos chamados grafos de interesse [3], que representam as estruturas do programa relacionadas ao código fonte com a finalidade de reduzir custos durante a atividade de investigação e ajudar o desenvolvedor na evolução de programas de maneira sistemática. A ferramenta FEAT, proposta por Robillard, é utilizada para investigar as relações entre os interesses capturados e o código base. Este trabalho não utiliza análise dinâmica e não trabalha com localização de características, mas trouxe algumas contribuições a esta dissertação, pois apresenta um objetivo comum que é a melhoria da compreensão de sistemas por meio do mapeamento do interesse para o código fonte e da documentação das visões geradas, as quais devem ser refinadas para serem utilizadas futuramente pelos analistas do sistema. Este trabalho contribuiu para a definição da *ConcernMapper* [Robillard e Weigand-Warr 2005], como uma ferramenta que foi utilizada na abordagem proposta nesta dissertação para a montagem e manutenção do mapeamento das características para os elementos de código que as implementam.

Em [Robillard e Weigand-Warr 2005], ocorre o preenchimento manual das informações para a *ConcernMapper*, enquanto que na abordagem apresentada nesta dissertação, criou-se a ferramenta *TraceToConcern* que, com base nos rastros de execução coletados e nas marcações de tempo de início e fim de cada característica, monta a visão Mapeamento apresentada no formato de leitura da *ConcernMapper*, possibilitando a integração com o IDE Eclipse.

O trabalho deles auxiliou a definição dos experimentos mostrados nesta dissertação. A avaliação deles incluía experimentos controlados com sujeitos humanos, visando responder se a abordagem *Concern Graphs* contribui para a compreensão dos sistemas.

[3] do inglês, *concern graphs*

## 5.8 Sobreira e Maia

Em [Sobreira e Maia 2008] foi proposto um método e uma ferramenta chamada Featincode para a análise do espalhamento de características através da interpretação gráfica da interseção entre características e elementos do código fonte. A ferramenta coleta e representa rastros de programas *multi-threaded* para as características selecionadas pelo desenvolvedor e apresenta algumas matrizes que ajudam a analisar onde as características estão implementadas. O método e a ferramenta proposta foram validados com a análise de algumas características da ferramenta CASE *ArgoUML*. A conclusão é que a abordagem pode reduzir o esforço para compreender onde as características estão implementadas e quais elementos do código são específicos de uma característica.

O trabalho apresentado nesta dissertação utilizou de uma parte da ferramenta Featincode, a *TraceExtractor*, e o mesmo modelo de extração para a obtenção dos rastros de execução. Porém, são abordagens diferentes que apresentam visões com informações distintas baseadas na mesma estrutura de rastros.

## 5.9 Quante

Em [Quante 2008], são apresentadas técnicas de análise automática de sistemas para extrair informações de arquitetura. Grafos de processo dinâmico de objeto (*Dynamic Object Process Graph* - DOPG) descrevem o fluxo de controle de uma aplicação a partir da perspectiva de um único objeto. Em uma série de trabalhos [Quante 2007, Quante e Koschke 2006], os autores propuseram esta técnica que extrai uma projeção de um grafo de fluxo de controle (*Control Flow Graph* - CFG) de um sistema e só mantém as partes do CFG que foram executadas. Esta técnica pretende proporcionar uma compreensão de como um componente está sendo usado em uma aplicação. Em pesquisas anteriores, foram realizados estudos de caso que indicaram que esta técnica pode ser útil para a compreensão de programas. Em [Quante 2008], a técnica foi validada empiricamente sobre a sua utilidade na prática, por meio da realização de um conjunto de estudos de casos, investigando as seguintes questões: 1) A disponibilidade da visão DOPGs leva a respostas mais rápidas para questões de compreensão de sistemas que estão de alguma forma relacionadas a um determinado componente? 2) Estas perguntas são respondidas menos erroneamente? O estudo foi realizado com 27 sujeitos humanos, 2 estudos de caso, 2 sistemas, 1 grupo controle e 1 grupo experimental para cada experimento. Os autores constataram evidências sobre a utilidade da técnica.

Como se pode notar, este trabalho trata o mesmo problema da compreensão de sistemas, utilizando análise dinâmica, porém, a técnica DOPG se difere muito da apresentada nesta abordagem, a visão gerada apresenta uma visão arquitetural totalmente diferente das visões apresentadas nesta dissertação. O principal motivo da presença deste

artigo nesta seção é a discussão da avaliação executada pelos autores. Conforme afirmado em [Quante 2008, Cornelissen et al. 2008], experimentos controlados para verificar os benefícios de uma ferramenta/abordagem realizados utilizando sujeitos humanos e com casos reais, são raramente executados. Dos 176 artigos avaliados em [Cornelissen et al. 2008] apenas 6 se encaixaram neste tipo de experimento, devida a dificuldade de execução dos mesmos. Tanto [Quante 2008] como o trabalho desta dissertação apresentam este tipo de avaliação. A avaliação da técnica DOPG foi muito parecida com a avaliação montada para este trabalho, apresentando uma condução muito próxima ao da metodologia adotada pela autora deste trabalho, confirmando que este seguiu todos os passos para controle de um experimento deste porte e dificuldade. Porém, dentre os pontos específicos de uma avaliação e outra, o que se destacou é que os sujeitos em [Quante 2008] responderam perguntas sobre o sistema e não executaram atividades reais de manutenção como a maioria dos experimentos apresentados nesta dissertação. A real manutenção eleva o nível de compreensão necessária. Dos pontos em comuns, nota-se a integração com o IDE Eclipse visando à melhoria da compreensão. As ameaças internas e externas a validação do trabalho [Quante 2008]apresentou muitos pontos em comum com as ameaças apresentadas no trabalho corrente. As perguntas respondidas pela validação do trabalho dele também foram respondidas pela abordagem apresentada nesta dissertação.

## 5.10 Salah, Mancoridis, Antoniol e Di Penta

Em [Salah et al. 2005,Salah et al. 2006], Salah e colegas apresentaram uma abordagem para compreensão de grandes sistemas, utilizando visões criadas a partir da submissão dos sistemas à análise dinâmica do ponto de vista de diferentes cenários. Neste trabalho, os autores apresentam uma abordagem que usa análise dinâmica para extrair visões de um sistema em diferentes níveis: visões de caso de uso, visão de interação dos módulos e visão de interação de classes. Algumas visões são baseadas em grafos de hierarquia que suportam a exploração da arquitetura do programa e outras são baseadas em métricas e focam em revelar a complexidade do sistema. A finalidade desta abordagem é permitir que os desenvolvedores localizem as características de interesse rapidamente e revelar as relações entre as classes e os módulos, apresentados durante a coleta dos rastros, assim como as interações entre as *threads* para um sistemas *multi-threaded*, que não são detectadas pela análise estática. Esta abordagem proposta foi avaliada utilizando um sistema de grande porte, o navegador web Mozilla, através de um estudo de caso que demonstrou a utilidade desta abordagem para as atividades de manutenção que requerem uma compreensão detalhada de partes específicas de um sistema de grande porte. As limitações encontradas na abordagem foram a queda de desempenho quando executada sobre sistemas de grande porte, o fato de somente poder ser aplicada a programas cujos componentes são executados no mesmo espaço de processo, além de a construção de uma

visão completa da execução de um sistema utilizando análise dinâmica ser uma tarefa difícil, devido à dificuldade inerente à execução de todos os possíveis casos de uso de um programa. Esta abordagem é mais utilizada para executar um conjunto de casos que são relevantes para uma tarefa de manutenção específica.

Novamente um ponto comum entre o trabalho deles e o desta dissertação é o uso de análise dinâmica para compreensão de sistemas, apresentando o relacionamento entre os elementos de código das características de interesse. Outros pontos em comum são a coleta dos rastros utilizando marcações e a geração de um conjunto de visões, a partir destes rastros, tendo em vista melhorar a compreensão do analista. Em ambos trabalhos foram utilizados filtros para reduzir o excesso de informação. O trabalho deles foca na visualização de grafos mostrando relacionamentos entre as entidades casos de uso, módulos, classes e métodos com base nas chamadas entre estes, enquanto que o trabalho corrente apresenta o Grafo de Chamadas entre os métodos filtrados, além de outras visões diferentes.

## 5.11 Antoniol e Guéhéneuc

Em [Antoniol e Gueheneuc 2005], é apresentada uma abordagem para localização de características baseada em análise dinâmica e estática para programas de grande porte, com processamento concorrente e orientados a objetos. Este trabalho apresenta três passos de execução, que contempla três diferentes técnicas, sendo estas: 1) análise estática; 2) identificação das características utilizando análise dinâmica, filtragem através da base de conhecimento e classificação probabilística; 3) apresentação das características através de um mapa 3D. A análise dinâmica serve como filtro para a análise estática, modelos estáticos, como diagramas de classes, são gerados e destacam diferenças entre as características. As características servem como ponto de ligação entre um modelo de arquitetura do programa e o seu comportamento dinâmico. O modelo de arquitetura é construído primeiramente com o auxilio da análise estática do código fonte. Em seguida as características são identificadas para prover aos mantenedores subconjuntos da arquitetura (ou micro-arquiteturas) representando classes, métodos e campos que participam nas características de interesse. As características são identificadas por análise dinâmica com base em dois diferentes roteiros de execução: aqueles relacionados a característica de interesse e aqueles que não exercitam a característica. Intervalos são marcados nestes roteiros para escolha de eventos relevantes para a característica. Os eventos relevantes são classificados quantificando a probabilidade de um evento (criação de um objeto, execução de um fragmento do código fonte) seja relevante para a característica, e associam-se os eventos mais relevantes à respectiva característica. A filtragem da base de conhecimento é necessária para remoção dos eventos "obviamente"irrelevantes, ajudando a reduzir a quantidade de dados a serem analisados. O modelo de característica é construído com base nas micro-arquiteturas, incluindo somente classes, métodos e campos ativados explicitamente ao

executar o roteiro que exercita a característica de interesse. A comparação entre as características é feita destacando-se as diferenças entre as micro-arquiteturas com o objetivo de permitir aos mantenedores compreender e comparar o comportamento de funcionalidades com roteiros distintos. Para destacar estas diferenças é calculado o conjunto de transformações no modelo que são necessárias para transformar uma micro-arquitetura em outra. Esta proposta permite a ordenação, com índice de relevância de classes e métodos que contribuem para uma micro-arquitetura implementando uma característica. Além disso, permite identificar diferenças entre características. Para visualizar o resultado desta abordagem, os autores utilizaram uma matriz em 3D que utiliza cores e volume das figuras para apresentar as entidades mais relevantes relacionadas às características. Para validarem a abordagem proposta, os autores realizaram uma comparação entre a micro-arquitetura associada a uma dada característica nos navegadores Firefox e Mozilla, uma vez que são sistemas que implementam características similares ou idênticas. A abordagem permitiu a identificação e comparação das micro-arquiteturas de aplicação da característica no Firefox e Mozilla, e assim, pode ajudar os mantenedores a comparar e compreender a arquitetura e o comportamento de dois programas do mesmo domínio.

Ao comparar o trabalho de Antoniol e Guéhéneuc com a abordagem corrente nota-se que ambos possuem o objetivo de auxiliar os desenvolvedores no processo de manutenção de programas utilizando localização de características. Porém, o trabalho [Antoniol e Gueheneuc 2005] apresenta uma abordagem diferente da apresentada nesta dissertação, realizando a remoção dos eventos considerados irrelevantes e apresentando uma classificação probabilística sobre os elementos mais relevantes para o sistema. Este trabalho foi selecionado pra fazer parte desta seção por apresentar uma técnica para diminuir o número de elementos apresentados nas visões com a remoção dos eventos irrelevantes, que ajuda a reduzir a quantidade de dados dinâmicos, auxiliando no processo de entendimento do programa. Este é um problema apresentado na abordagem corrente que pode ser tratado seguindo esta metodologia para selecionar os elementos que serão apresentados nas visões, pois, estas apresentam todos os elementos executados, sem repetições.

## 5.12 Zhao e Zhang

No trabalho [Zhao et al. 2004], Zhao e Zhang apresentam um método estático e não-interativo para localização de características e melhoria da compreensão de sistemas chamado SNIAFL (*Static Non-Interactive Approach to Feature Location*). Uma técnica de recuperação de informação (RI) é usada para revelar as conexões básicas entre características e unidades computacionais no código fonte. Uma representação estática do código fonte (*Branch-Reserving Call Graph* - BRCG) é utilizada para recuperar unidades relevantes e específicas para cada característica. O interesse principal desta abordagem é a localização de funções (unidades computacionais) relevantes e funções específicas de

uma característica. Funções específicas são aquelas utilizadas na implementação de uma característica exclusivamente. Funções relevantes são as que estão envolvidas na implementação da característica, mas não são necessariamente exclusivas. A idéia básica é usar RI para revelar as conexões entre características e funções. As características são descritas em linguagem natural e nomes significativos para os identificadores devem ter sido usados no código fonte. Um conjunto inicial de funções específicas de cada característica é buscado através de técnicas de RI que se baseiam no casamento entre as descrições dessas funções com os identificadores no código. Este conjunto de funções específicas é utilizado como base para a descoberta do conjunto de funções relevantes. Para isto é utilizada a BRCG, um grado de chamadas que utiliza informações estáticas para apresentar a seguência das chamadas destas funções especificas, possibilitando a obtenção dos relacionamentos que leva à recuperação das funções relevantes. O processo pode ser resumido em quatro passos: 1) obter as conexões iniciais específicas entre características e funções; 2) obter as funções iniciais específicas; 3) determinar funções e rastros relevantes; 4) determinar as funções específicas finais. Para o passo 1 utiliza-se RI para filtrar as informações específicas das características e recuperar as conexões iniciais. O segundo passo é ordenar as funções em cada característica de acordo com o resultado da recuperação e escolher a função específica inicial para cada característica. O terceiro passo é obter as funções relevantes e a sequência das chamadas obtidas através do BRCG. O último passo é a análise das funções relevantes para determinar as funções específicas finais. A avaliação da abordagem foi realizada por meio de um estudo experimental sobre um sistema GNU, escrito em linguagem C. No estudo experimental, são apresentados os dados quantitativos detalhados e os resultados qualitativos da análise. Segundo os autores, a principal vantagem desta abordagem sobre as demais citadas, que utilizam análise dinâmica, está no melhor suporte para automação, uma vez que a ligação entre as características e a implementação é feita através de RI ao invés de execução interativa do sistema que pode exigir o planejamento e execução de muitos roteiros de execução do sistema. Dentre as desvantagens estão a definição prévia de descrições para as características, a dependência do uso de identificadores na implementação que sejam consistentes com a descrição de características, e a inflexibilidade na escolha da granularidade mínima do mapeamento, limitada as funções (ou métodos) no sistema.

SNIAFL e a abordagem apresentada nesta dissertação, possuem o mesmo objetivo de auxiliar nas atividades que envolvem a compreensão e manutenção de sistemas utilizando localização de características no código fonte. Porém, estas possuem muitas diferenças, começando pela técnica adotada pelas abordagens, uma utiliza análise dinâmica e a outra análise estática combinada com RI. A técnica SNIAFL foi desenvolvida para trabalhar com linguagem C e a abordagem corrente com Java. As informações geradas pelas abordagens se diferem, inclusive quanto a granularidade da análise, a abordagem desta dissertação trabalha com a granularidade até o nível de classes e métodos enquanto SNIAFL trabalha

somente no nível de funções. Porém, um ponto em comum nas visões é que ambas as abordagens classificam as unidades computacionais como específicas ou relevantes, onde a definição é a mesma para ambas as abordagens. Isso, reafirma a importância deste tipo de informação para a compreensão de sistemas. Na abordagem apresentada nesta dissertação, esta classificação é apresentada na visão Mapeamento com Classificação e as classificações são como específica, compartilhada (relevante) e presente em todas. Mesmo assim, a técnica SNIAFL e a abordagem corrente são razoavelmente diferentes. Este trabalho foi relacionado nesta seção, devido ao fato de RI poder vir a ser utilizada para refinar as informações dinâmicas, diminuindo ainda mais o espaço de busca. Nos experimentos realizados para a abordagem corrente, os analistas puderam refinar as informações dinâmicas utilizando as ferramentas do Eclipse, onde uma das ferramentas mais utilizadas foi "Search"que realiza busca por palavra chave. Os analistas, com base nos elementos mapeados pela análise dinâmica buscaram nos códigos mapeados as palavras que mediante ao requisito apresentado para a atividade poderiam direcionar localização do interesse. Por este motivo, uma futura integração da abordagem corrente com técnicas de RI, que são mais robustas que o simples "Search"do Eclipse pode ser uma solução para montar a classificação dos elementos com maior probabilidade de conter o interesse para uma dada atividade de manutenção.

# Capítulo 6

# Conclusão

Neste trabalho foi apresentada uma abordagem para localização de características utilizando análise dinâmica refinada pela análise estática, e um estudo experimental controlado utilizando sujeitos humanos com o objetivo de avaliar o impacto do uso desta abordagem em atividades de manutenção de programas. Conforme afirmado em [Cornelissen et al. 2008], experimentos como este são raramente realizados devida a dificuldade de execução, apesar de a participação de seres humanos nas avaliações ser importante para o campo compreensão de sistemas e constituir em importante contribuição. O estudo contou com 1 (um) estudo de caso para avaliar a precisão da abordagem que contemplou 2 (duas) observações individuais, 4 (quatro) experimentos controlados, cada um aplicado a 3 grupos distintos (controle, abordagem completa e abordagem parcial), que deu origem a 36 observações individuais. Estas observações individuais foram agrupadas conforme as variáveis independentes, visando observar as principais variáveis dependentes, tempo e taxa de acerto, que resultou nas observações sobre resultados apresentadas no Capítulo 4. As conclusões aqui tiradas serão baseadas nestes resultados gerais, que contemplam os 4 experimentos e 1 estudo de caso. O conjunto nos traz informações mais confiáveis, pois avalia um número maior de observações. Estes resultados mostraram que:

- O tempo de execução com o uso da abordagem foi melhor em situações onde as características a serem localizados não estavam bem modularizados em unidades específicas; e os elementos de código de interesse não possuem nomes intuitivos;
- A taxa de acertos com o uso da abordagem foi pelo menos similar ao não uso da abordagem, tendo, porém, um desempenho claramente superior em atividade de correção de erro e pequenas melhorias no sistema (não houve estudos sobre manutenções extensas);
- As informações da visão Mapeamento gerada pela abordagem foram úteis para a localização das características. Porém, somente as visões da abordagem não levam a compreensão necessária para a execução da manutenção, onde o uso de informações estáticas, como consulta ao código fonte, é necessária;

- Os experimentos onde ocorreu uma expressiva redução do espaço de busca inicial com maior qualidade da informação contida neste, ou seja, sem falsos negativos, apresentaram os melhores resultados de desempenho, tempo e taxa de acerto;
- As atividades de manutenção foram consideradas com um nível de dificuldade menor pelos grupos que utilizaram a abordagem. Isso ocorreu principalmente nas atividades onde a localização dos elementos de código foi dificultada pelo tamanho do sistema, espalhamento e entrelaçamento presente no código fonte (Correção e Melhoria).

Com base nestes resultados serão rediscutidas as hipóteses, que irão fundamentar as conclusões deste trabalho e verificar se o objetivo principal foi atingido. Seguem as hipóteses e as discussões.

H1) Sujeitos que utilizam a abordagem apresentam melhor tempo de execução em atividades de manutenção sobre sistemas desconhecidos do que sujeitos utilizando uma abordagem tradicional. *Argumentação inicial*: Quando a abordagem é utilizada sobre sistemas desconhecidos, a localização da característica potencialmente seria mais rápida do que quando não utilizada e, conseqüentemente, a resposta a uma atividade ligada à compreensão do sistema que requeresse a localização da característica, também, seria mais rápida. A localização de características utilizando análise dinâmica reduz e organiza o espaço de busca do sujeito pela característica de interesse, que geralmente se encontra dispersa no código fonte.

Em 3 dos 4 experimentos realizados, o tempo de execução dos sujeitos que utilizaram a abordagem foi menor, e conforme resultados, este tempo foi melhor com o uso da abordagem em situações onde as características a serem localizadas não estavam bem modularizadas ou o elemento de código de interesse não apresentou nome intuitivo. O resultado foi semelhante ao uso de uma abordagem tradicional, em situação que o código estava bem modularizado, não se apresentando espalhado ou entrelaçado. Os resultados, também, indicaram a utilidade da abordagem na localização das características de interesse e uma redução organizada do espaço de busca inicial por meio das visões da abordagem, que tem indício de influência no desempenho, uma vez que os melhores desempenhos foram nos experimentos com maior redução e qualidade do espaço de busca inicial. Logo, estes resultados confirmam a argumentação apresentada na hipótese H1 para os experimentos que apresentaram o problema tratado por este trabalho, referente aos códigos entrelaçados e dispersos que dificultam a localização da característica.

H2) Sujeitos que utilizam a abordagem apresentam maior correção nas respostas durante a execução de atividades de manutenção em sistemas desconhecidos do que sujeitos que utilizam a abordagem tradicional. *Argumentação inicial:* As visões desta abordagem auxiliam o desenvolvedor provêem informações dinâmicas para a compreensão do sistema, como as classes e métodos executados para uma característica de interesse, a presença de

elementos de código em comum entre características de interesse e as chamadas entre os métodos executados para as características. O não uso da abordagem usualmente implica em navegação em informações estáticas, que apesar de ser potencialmente útil, ocorre sem necessariamente a aplicação de nenhum filtro. Porém, com o uso da abordagem apresentada, ocorre um direcionamento para informações das características de interesse potencialmente induzindo respostas mais precisas.

A taxa de acertos com o uso da abordagem foi pelo menos similar ao não uso da abordagem, apresentando resultados melhores em 3 dos 4 experimentos. E, as visões foram úteis através da localização das características de interesse e redução do espaço de busca inicial, levando ao direcionamento na busca pelas informações, o que pode acarretar em respostas mais precisas. Além do mais, os participantes que utilizaram a abordagem indicaram menor nível de dificuldade nas atividades propostas nos experimentos, quando comparados ao grupo que não utilizou a abordagem. O único experimento em que os grupos que utilizaram a abordagem apresentaram um resultado um pouco melhor, mas não se podendo afirmar que foi superior, foi no experimento em que a localização das características foi facilitada, por que o código de interesse não se encontrava espalhado ou entrelaçado. Logo, para o problema ao qual esta abordagem foi proposta, ela se apresentou útil com uma taxa de acerto superior ao não uso desta.

Portanto, as hipóteses foram confirmadas para manutenções que apresentam o problema tratado e comprova-se que com o uso da abordagem, os resultados, em tempo de execução e taxa de acerto, são melhores ou pelo menos similar ao uso de uma abordagem amplamente utilizada, e, portanto, sendo útil para a compreensão dos sistemas e respondendo positivamente ao objetivo proposto. O desempenho somente foi similar em uma atividade onde a localização dos elementos de código que implementam as características foi facilitada pela estrutura do sistema, não se enquadrando no problema do espalhamento e entrelaçamento aqui tratado, levando à concluir que a abordagem não se faz útil em situações como esta.

Além dos objetivos já discutidos, é esperado conforme Seção 1.2, que a abordagem auxilie na atividade de documentação da rastreabilidade entre características de interesse e o código fonte que a implementa, através da localização das classes e métodos que implementam as características. Através da visão Mapeamento, a abordagem apresenta uma proposta de rastreabilidade das características para o código fonte, que pode ser refinada, arquivada e utilizada futuramente por outros analistas. A rastreabilidade tradicional não foi amplamente adotada devido aos custos com sua atualização, que leva a problemas de consistência dos dados ou não utilização desta prática, sendo bastante raro encontrar boa documentação de rastreabilidade [Wilde et al. 2003]. Entretanto, com o uso da abordagem proposta, esta rastreabilidade poderá ser obtida dinamicamente, a qualquer momento e trazendo dados atualizados. Por este motivo, acredita-se que a abordagem pode contribuir para o uso e consistência desta prática. Porém, para uma adoção em larga escala,

é necessário tratar o problema referente aos falsos positivos e falsos negativos presentes nesta visão, o que constitui uma proposta para trabalhos futuros que serão apresentados na próxima seção. No entanto, conforme apresentado nos estudos preliminares, o uso da abordagem apresentou resultados com boa precisão e acurácia, podendo ser refinados manualmente com a exclusão e inclusão manual de elementos de código, realizada por um analista com bom conhecimento sobre o sistema e que após esta etapa poderá ser utilizado como documento de rastreabilidade por outros analistas menos experientes. Com base nestas observações considera-se que este objetivo foi satisfatoriamente alcançado.

Conforme os resultados apresentados, pode-se concluir que este trabalho cumpriu o principal objetivo proposto de apresentar como contribuição uma abordagem que auxilia o desenvolvedor na compreensão das características dos sistemas utilizando análise dinâmica e estática, e esclarecer com um estudo controlado do uso desta abordagem perguntas ligadas ao objetivo de auxiliar na compreensão das características, em relação ao tempo de localização da característica e à taxa de acerto nas atividades de manutenção, com informações que direcionam a compreensão. As respostas às perguntas demonstraram os bons resultados da abordagem nas questões estudadas.

Como trabalhos futuros, sugere-se seguir a mesma metodologia adotada neste trabalho para a avaliação de técnicas mais sofisticadas de localização de características utilizando rastros de execução dedicados a tarefas específicas de manutenção. As contribuições deste tipo de pesquisa são mais robustas e confiáveis por serem executadas por pessoas não envolvidas com a pesquisa e envolver um número maior de observações.

Ainda como trabalhos futuros, melhorias na abordagem proposta poderiam torná-la mais útil, como propostas com novas visões utilizando as informações dos rastros de execução ou soluções para os problemas apresentados. Uma proposta futura seria uma nova visão com as chamadas entre os métodos que aponte a ordem temporal da chamada entre estes. Segundo alguns sujeitos que participaram dos experimentos, esta visão seria de grande utilidade para a compreensão dos sistemas, além do mais, esta informação já se encontra disponível no arquivo de rastros gerado. Outra sugestão dos participantes é a montagem de uma visão que apresente o comportamento dos objetos de interesse, executados durante a análise dinâmica, apresentando quando estes são criados, quais métodos os recebem, quais são os objetos de uma classe, etc. Outra melhoria seria a integração ou criação de técnicas para minimizar o volume de falsos positivos e falsos negativos na abordagem proposta, pois, a existência deste tipo de elemento prejudica o uso das informações dinâmicas. Uma sugestão é a integração de técnicas que envolvem recuperação de informação (RI), como em [Zhao et al. 2004], para obter informações estáticas que possam refinar de maneira direcionada os resultados gerados dinamicamente, montando uma classificação dos elementos com maior probabilidade de conter o interesse para uma dada atividade de manutenção. Diversas melhorias baseadas nas diferentes visões apresentadas nos trabalhos relacionados podem ser incorporadas à abordagem proposta visando torná-

la mais robusta para atender as fábricas de sistemas e *outsources* em larga escala. Os trabalhos relacionados apresentam muitas visões diferentes das apresentadas neste trabalho que se mostraram úteis para a compreensão de sistemas, e podem ser reproduzidas e até melhoradas para este fim.

Os resultados indicaram que uma vez que as abordagens dinâmicas para localização de características foram efetivas na compreensão de sistemas, vale a pena investir neste tipo de solução, principalmente no formato de *plugins* que possibilitam o uso destas visões dentro de IDEs amplamente utilizados pelos desenvolvedores, a fim de garantir maior aderência e possibilitar o uso de ferramentas que já fazem parte do cotidiano dos desenvolvedores, levando estas soluções que utilizam análise dinâmica a um uso mais amplo e difundido.

# Referências Bibliográficas


[Antoniol e Gueheneuc 2005] Antoniol, G. e Gueheneuc, Y.-G. (2005). Feature identification: a novel approach and a case study. In *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, pp. 357 – 366.

[Antoniol et al. 2005] Antoniol, G., Merlo, E., Guéhéneuc, Y.-G., e Sahraoui, H. (2005). On feature traceability in object oriented programs. In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, TEFSE '05, pp. 73–78, Long Beach, California, New York, NY, USA. ACM.

[Arnold 1996] Arnold, R. S. (1996). *Software Change Impact Analysis*. IEEE Computer Society Press, Los Alamitos, CA, USA.

[Baniassad et al. 2002] Baniassad, E. L. A., Murphy, G. C., Schwanninger, C., e Kircher, M. (2002). Managing crosscutting concerns during software evolution tasks: an inquisitive study. In *Proceedings of the 1st international conference on Aspect-oriented software development*, AOSD '02, pp. 120–126, Enschede, The Netherlands, New York, NY, USA. ACM.

[Batory 2006] Batory, D. (2006). A Tutorial on Feature Oriented Programming and the AHEAD Tool Suite. In *Generative and Transformational Techniques in Software Engineering*, volume 4143 de *Lecture Notes in Computer Science*, pp. 3–35. Springer.

[Chen e Rajlich 2000] Chen, K. e Rajlich, V. (2000). Case study of feature location using dependence graph. In *Program Comprehension, 2000. Proceedings. IWPC 2000. 8th International Workshop on*, pp. 241 –247.

[Cornelissen et al. 2008] Cornelissen, B., Zaidman, A., van Deursen, A., Moonen, L., e Koschke, R. (2008). A Systematic Survey of Program Comprehension through Dynamic Analysis. Technical Report TUD-SERG-2008-033, Delft University of Technology.

[Dijkstra 1997] Dijkstra, E. W. (1997). *A Discipline of Programming*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[Dijkstra 2002] Dijkstra, E. W. (2002). *The structure of the "The"multiprogramming system*, pp. 139–152. Springer-Verlag New York, Inc., New York, NY, USA.

[Eick et al. 1992] Eick, S., Steffen, J., e Sumner, E.E., J. (1992). Seesoft-a tool for visualizing line oriented software statistics. *Software Engineering, IEEE Transactions on*, 18(11):957 –968.

[Eisenbarth et al. 2001] Eisenbarth, T., Koschke, R., e Simon, D. (2001). Aiding program comprehension by static and dynamic feature analysis. In *Software Maintenance, 2001. Proceedings. IEEE International Conference on*, pp. 602 –611.

[Eisenbarth et al. 2002] Eisenbarth, T., Koschke, R., e Simon, D. (2002). Incremental location of combined features for large-scale programs. In *Software Maintenance, 2002. Proceedings. International Conference on*, pp. 273 – 282.

[Eisenbarth et al. 2003] Eisenbarth, T., Koschke, R., e Simon, D. (2003). Locating Features in Source Code. *IEEE Trans. Software Eng.*, 29(3):210–224.

[Eisenberg e Volder 2005] Eisenberg, A. D. e Volder, K. D. (2005). Dynamic Feature Traces: finding Features in Unfamiliar Code. In *Proc. 21st Int. Conf. on Software Maintenance (ICSM)*, pp. 337–346. IEEE C.S.

[Gotel e Finkelstein 1994] Gotel, O. e Finkelstein, C. (1994). An analysis of the requirements traceability problem. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pp. 94 –101.

[Greevy e Ducasse 2005] Greevy, O. e Ducasse, S. (2005). Correlating Features and Code Using a Compact Two-Sided Trace Analysis Approach. In *Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on*, pp. 314 – 323.

[Greevy et al. 2006] Greevy, O., Ducasse, S., e Gîrba, T. (2006). Analyzing software evolution through feature views. *J. Softw. Maint. Evol.: Res. Pract.*, 18(6):425–456.

[Janzen e De Volder 2003] Janzen, D. e De Volder, K. (2003). Navigating and querying code without getting lost. In *Proceedings of the 2nd international conference on Aspect-oriented software development*, AOSD '03, pp. 178–187, Boston, Massachusetts, New York, NY, USA. ACM.

[Lehman 1980] Lehman, M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060 – 1076.

[Liu et al. 2007] Liu, D., Marcus, A., Poshyvanyk, D., e Rajlich, V. (2007). Feature location via information retrieval based filtering of a single scenario execution trace. In *Proc. 22nd Int. Conf. on Automated Software Engineering (ASE)*, pp. 234–243. ACM.

[Lukoit et al. 2000] Lukoit, K., Wilde, N., Stowell, S., e Hennessey, T. (2000). TraceGraph: Immediate Visual Location of Software Features. In *Proc. 16nd Int. Conf. on Software Maintenance (ICSM)*, pp. 33–39. IEEE C.S.

[Neumuller e Grunbacher 2006] Neumuller, C. e Grunbacher, P. (2006). Automating Software Traceability in Very Small Companies: A Case Study and Lessons Learned. In *Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on*, pp. 145 –156.

[Parnas 1979] Parnas, D. L. (1979). *On the criteria to be used in decomposing systems into modules*, pp. 139–150. Yourdon Press, Upper Saddle River, NJ, USA.

[Quante 2007] Quante, J. (2007). Online Construction of Dynamic Object Process Graphs. In *Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference on*, pp. 113 –122.

[Quante 2008] Quante, J. (2008). Do Dynamic Object Process Graphs Support Program Understanding? – A Controlled Experiment. In *Proc. 16th Int. Conf. on Program Comprehension (ICPC)*, pp. 73–82. IEEE C.S.

[Quante e Koschke 2006] Quante, J. e Koschke, R. (2006). Dynamic object process graphs. In *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*, pp. 10 pp. –90.

[Ramesh et al. 1997] Ramesh, B., Stubbs, C., Powers, T., e Edwards, M. (1997). Requirements traceability: Theory and practice. *Ann. Softw. Eng.*, 3:397–415.

[Richner e Ducasse 1999] Richner, T. e Ducasse, S. (1999). Recovering High-Level Views of Object-Oriented Applications from Static and Dynamic Information. In *Proc. 15th Int. Conf. on Software Maintenance (ICSM)*, pp. 13–22. IEEE C.S.

[Robillard 2003] Robillard, M. P. (2003). *Representing concerns in source code*. PhD thesis, Department of Computer Science, The University of British Columbia, Vancouver, Canada.

[Robillard e Murphy 2007] Robillard, M. P. e Murphy, G. C. (2007). Representing concerns in source code. *ACM Trans. Softw. Eng. Methodol.*, 16(1):3:1–38.

[Robillard e Weigand-Warr 2005] Robillard, M. P. e Weigand-Warr, F. (2005). ConcernMapper: simple view-based separation of scattered concerns. In *Eclipse '05: Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pp. 65–69, San Diego, California, New York, NY, USA. ACM.

[Rohatgi et al. 2008] Rohatgi, A., Hamou-Lhadj, A., e Rilling, J. (2008). An Approach for Mapping Features to Code Based on Static and Dynamic Analysis. In *Proc. 16th Int. Conf. on Program Comprehension (ICPC)*, pp. 236–241. IEEE C.S.

[Salah et al. 2005] Salah, M., Mancoridis, S., Antonio, G., e Di Penta, M. (2005). Towards employing use-cases and dynamic analysis to comprehend Mozilla. In *Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on*, pp. 639 – 642.

[Salah et al. 2006] Salah, M., Mancoridis, S., Antoniol, G., e Penta, M. D. (2006). Scenario-Driven Dynamic Analysis for Comprehending Large Software Systems. In *Proc. 10th European Conf. on Software Maintenance and Reengineering (CSMR)*, pp. 71–80. IEEE C.S.

[Schmerl et al. 2006] Schmerl, B. R., Aldrich, J., Garlan, D., Kazman, R., e Yan, H. (2006). Discovering Architectures from Running Systems. *IEEE Trans. Software Eng.*, 32(7):454–466.

[Sefika et al. 1996] Sefika, M., Sane, A., e Campbell, R. H. (1996). Architecture-Oriented Visualization. In *Proc. 11th Conf. on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pp. 389–405. ACM.

[Simmons et al. 2006] Simmons, S., Edwards, D., Wilde, N., Homan, J., e Groble, M. (2006). Industrial tools for the feature location problem: an exploratory study. *J. Softw. Maint. Evol.: Res. Pract.*, 18(6):457–474.

[Sobreira e Maia 2008] Sobreira, V. e Maia, M. d. A. (2008). Analyzing Feature Scattering with Visual Information of Execution Traces. *INFOCOMP(UFLA)*, pp. 21–30.

[Tarr et al. 2002] Tarr, P., Ossher, H., e Sutton, S.M., J. (2002). Hyper/J trade;: multi-dimensional separation of concerns for Java trade;. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pp. 689 – 690.

[Walker et al. 1998] Walker, R. J., Murphy, G. C., Freeman-Benson, B. N., Wright, D., Swanson, D., e Isaak, J. (1998). Visualizing Dynamic Software System Information through High-Level Models. In *Proc. 13th Conf. on Object-Oriented Programming Systems, Languages & Applications (OOPSLA)*, pp. 271–283. ACM.

[Wilde et al. 2001] Wilde, N., Buckellew, M., Page, H., e Rajlich, V. (2001). A Case Study of Feature Location in Unstructured Legacy Fortran Code. In *Proc. 5th European Conf. on Software Maintenance and Reengineering (CSMR)*, pp. 68–76. IEEE C.S.

[Wilde et al. 2003] Wilde, N., Buckellew, M., Page, H., Rajlich, V., e Pounds, L. (2003). A Comparison of Methods for Locating Features in Legacy Software. *J. Syst. Software*, 65(2):105–114.

[Wilde e Casey 1996] Wilde, N. e Casey, C. (1996). Early field experience with the Software Reconnaissance technique for program comprehension. In *Proc. Int. Conf. on Software Maintenance (ICSM)*, pp. 312–318. IEEE C.S.

[Wilde e Scully 1995] Wilde, N. e Scully, M. C. (1995). Software Reconnaissance: Mapping Program Features to Code. *J. Softw. Maint.: Res. Pract.*, 7(1):49–62.

[Wong et al. 1999] Wong, W., Gokhale, S., Horgan, J., e Trivedi, K. (1999). Locating program features using execution slices. In *Application-Specific Systems and Software Engineering and Technology, 1999. ASSET '99. Proceedings. 1999 IEEE Symposium on*, pp. 194 –203.

[Zhao et al. 2004] Zhao, W., Zhang, L., Liu, Y., Sun, J., e Yang, F. (2004). SNIAFL: towards a static non-interactive approach to feature location. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pp. 293 – 303.

# Anexo A

# Formulário para seleção dos participantes

As Figuras A.1, A.2, A.3, A.4, A.5, A.6 e A.7 apresentam o formulário de avaliação dos participantes, contemplando as perguntas realizadas visando verificar o conhecimento destes em desenvolvimento Java e uso do IDE Eclipse para seleção e classificação, conforme Tabela 3.4 .

**Faculdade de Computação**
**Pós Graduação em Ciência da Computação**

# Projeto de Pesquisa de Mestrado

Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

## Perfil dos Participantes dos Experimentos

### 1- Dados Pessoais

**Nome completo:**
**Endereço:** **Número:**
**Bairro:** **Complemento:**
**Estado:**
**Cidade:**
**Telefone:**
**Data de Nascimento:**

**Grupo: ( ) 1 ( ) 2 ( )3 <não preencha este campo>**

**Possui disponibilidade para participar dos experimentos? Quais dias da semana e horários?**
**( )Sim ( )Não**

### 2- Formação Acadêmica:

**Maior nível de escolaridade**

**Grau de escolaridade:**

( ) Segundo Grau incompleto
( ) Segundo Grau cursando
( ) Segundo Grau completo
( ) Superior incompleto
( ) Superior cursando
( ) Superior completo
( ) Pós graduação incompleto
( ) Pós graduação cursando
( ) Pós graduação completo
( ) Mestrado incompleto
( ) Mestrado cursando
( ) Mestrado completo
( ) Doutorado incompleto
( ) Doutorado cursando
( ) Doutorado completo

**Instituição:**

Figura A.1: Formulário de avaliação dos analistas participantes - página 1.

**Especialização:**
**Localidade:**
**Ano de Início:** **Ano de Conclusão:**

## 3- Idiomas:

**Inglês**

**Leitura :** ( ) Sem ( ) Regular ( ) Bom ( ) Excelente

**Escrita :** ( ) Sem ( ) Regular ( ) Bom ( )Excelente

**Conversação:** ( ) Sem ( ) Regular ( ) Bom ( )Excelente

**Inglês técnico**

**Leitura :** ( ) Sem ( ) Regular ( ) Bom ( ) Excelente

**Escrita :** ( ) Sem ( ) Regular ( ) Bom ( )Excelente

**Conversação:** ( ) Sem ( ) Regular ( ) Bom ( )Excelente

## 4- Cursos e Certificações:

**Descreva os cursos e certificações mais importantes que você concluiu:**

-
-
-
-
-
-
-
-
-
-
-

## 5- Experiência Profissional:

**Último emprego ou atual:**
**Nome da Empresa:**
**Cargo:**
**Ramo de Atividade:**
**Data de Admissão:** **Data de Saída:**

Figura A.2: Formulário de avaliação dos analistas participantes - página 2.

## 6- Perguntas específicas:

**1) Qual o tempo de experiência como desenvolvedor de sistemas?**

Tempo: _____ meses *<se não houver experiência coloque zero>*

**2) Se sim, qual o nível de conhecimento como desenvolvedor:**

( ) Estagiário
( ) Junior
( ) Pleno
( ) Sênior
Justifique:_____
_____
_____
_____

**3) Qual o nível de conhecimento em programação orientada a objetos (OO):**

( ) Básico
( ) Intermediário
( ) Avançado
( ) Sênior

**4) Qual o seu nível de conhecimento em modelagem de sistemas orientados a objeto?**

( ) Básico
( ) Intermediário
( ) Avançado
( ) Sênior

**5) Qual o nível de conhecimento na linguagem de programação Java:**

( ) Básico
( ) Intermediário
( ) Avançado
( ) Sênior

**6) Você já efetuou atividades de desenvolvimento/manutenção em sistemas desenvolvidos em Java:**

( ) SIM
( ) Não

**7) Que tipo de atividade (marque todas que já tiver efetuado)?**

( ) Correções de erros
( ) Implementar novas funcionalidades
( ) Desenvolver sistemas completos
( ) Reutilização de código fonte
( ) Melhorias na solução
( ) Mudanças na arquitetura da solução
( ) Mudanças utilizando switches Java

Figura A.3: Formulário de avaliação dos analistas participantes - página 3.

( ) Outros:
_____
_____
_____

**8) Você já utilizou o IDE Eclipse?**
( )Sim
( ) Não

**9) Se sim, qual o nível de domínio sobre o Eclipse:**
( ) Básico
( ) Intermediário
( ) Avançado
**Justifique:**
_____
_____
_____
_____
_____
_____
_____
_____

**10) Se você tiver que realizar uma atividade de manutenção em um sistema desenvolvido em Java por uma terceira pessoa, em que este sistema não possui nenhuma documentação de apoio, quais seriam os seus possíveis procedimentos/estratégias, utilizando o IDE Eclipse, para obter as informações e conhecimento necessário para realizar atividade de manutenção?**

**11) Sejam os dois jogos definidos abaixo: JogoAdivinha e JogoMemória.Existem algumas duplicações de código entre as classes. Indique as mudanças necessárias para eliminar as duplicações abaixo. Fica ao seu critério reescrever o código todo ou apenas indicar as alterações necessárias.**

**Elimine as seguintes duplicações, indicando a padrão de projeto utilizado quando for o caso:**
1. **Variáveis de instância palavras[] e jogadores[]**

Figura A.4: Formulário de avaliação dos analistas participantes - página 4.

2. **Construtor JogoAdivinha e JogoMemória**
3. **Variáveis de instância e métodos comuns em JogadorAdivinha e JogadorMemória**
4. **Definir um método único para os métodos de JogadorMemória usando polimorfismo paramétrico (vencedor() e ganhadorRodada())**

```
package prova2_2008_1.sempadrao;
import javax.swing.*;
public class JogoAdivinha {
   public String palavras[] = {"boi", "cavalo", "pato", "galo", "peixe"};
   JogadorAdivinha jogadores[] = new JogadorAdivinha[3];
   JogoAdivinha () {
           String palavra;
           convidaJogadores();
           brincaComPalavras();
           JOptionPane.showMessageDialog(null,jogadores[vencedor()].nome + " ganhou!!!");
   }
   void convidaJogadores () {
           for (int i=0; i<3; i++)
                   jogadores[i] = new JogadorAdivinha(JOptionPane.showInputDialog("Escreva seu nome"));
   }
   void brincaComPalavras () {
           do {
              for (int i=0; i<3; i++) {
                 String palavra = JOptionPane.showInputDialog("Adivinhe uma palavra");
                 if (temPalavra(palavra))
                    jogadores[i].aumentaPontos();
              }
           } while (JOptionPane.showInputDialog("Quer parar? (s/n)").equals("n"));
   }
   boolean temPalavra (String p) {
           for (int i = 0; i<palavras.length; i++)
                   if (palavras[i].equals(p))
                           return true;
           return false;
   }
   int vencedor () {
           int maioral=0;
           for (int i=1; i<3; i++)
                   if (jogadores[i].melhor(jogadores[maioral]))
                           maioral = i;
           return maioral;
   }
   public static void main (String args[]) {
           new JogoAdivinha();
   }
}

class JogadorAdivinha {
        String nome;
        int    pontos;
        JogadorAdivinha (String n) {
                nome = n;
        }
        void aumentaPontos () {
                pontos++;
        }
        boolean melhor (JogadorAdivinha outro) {
                if (this.pontos > outro.pontos)
                        return true;
                return false;
        }
}
```

Figura A.5: Formulário de avaliação dos analistas participantes - página 5.

```
package prova2_2008_1.sempadrao;
import javax.swing.*;
public class JogoMemoria {
   public String palavras[] = null;
   JogadorMemoria jogadores[];
   JogoMemoria () {
           String palavra;
           convidaJogadores();
           brincaComPalavras();
           JOptionPane.showMessageDialog(null,jogadores[vencedor()].nome + " ganhou!!!");
   }
   void convidaJogadores () {
           int nroJogadores = Integer.parseInt(JOptionPane.showInputDialog("Digite nro. jogadores"));
           palavras = new String[nroJogadores];
           jogadores = new JogadorMemoria[nroJogadores];
           for (int i=0; i<nroJogadores; i++) {
                   jogadores[i] = new JogadorMemoria(JOptionPane.showInputDialog("Escreva seu nome"));
                   palavras[i] = JOptionPane.showInputDialog("Forneça uma palavra qualquer");
           }
   }
   void brincaComPalavras () {
      do {
         for (int i=0; i<jogadores.length; i++) {
            for (int j=0; j < jogadores.length; j++)
               if (i!=j) {
                  String palavra = JOptionPane.showInputDialog(jogadores[i].nome +
                                                   ", lembre a palavra do "+jogadores[j].nome);
                  if (palavras[j].equals(palavra))
                       jogadores[i].aumentaPontos();
                }
          }
          jogadores[ganhadorRodada()].aumentaRodadasGanhas();
          JOptionPane.showMessageDialog(null,"O ganhador da rodada foi " + jogadores[ganhadorRodada()].nome);
              for (int i=0; i<jogadores.length;i++)
                 jogadores[i].pontos = 0;
         } while (JOptionPane.showInputDialog("Quer parar? (s/n)").equals("n"));
   }
   boolean temPalavra (String p) {
           for (int i = 0; i<palavras.length; i++)
                   if (palavras[i].equals(p))
                           return true;
           return false;
   }
   int vencedor () {
           int maioral=0;
           for (int i=1; i< jogadores.length; i++)
                   if (jogadores[i].melhor(jogadores[maioral]))
                           maioral = i;
           return maioral;
   }
   int ganhadorRodada () {
           int ganhador=0;
           for (int i=1; i<jogadores.length; i++)
                   if (jogadores[i].pontos > jogadores[ganhador].pontos)
                           ganhador = i;
           return ganhador;
   }
   public static void main (String args[]) {
           new JogoMemoria();
   }
}

class JogadorMemoria {
       String nome;
       int    pontos;
       int    rodadasGanhas;
       JogadorMemoria (String n) {
              nome = n;
       }
       boolean melhor (JogadorMemoria outro) {
              if (this.rodadasGanhas > outro.rodadasGanhas)
                     return true;
              return false;
       }
       void aumentaPontos () {
              pontos++;
       }
      void aumentaRodadasGanhas () {
         rodadasGanhas++;
       }
}
```

Figura A.6: Formulário de avaliação dos analistas participantes - página 6.

**12) Você já utilizou a rastreabilidade entre as funcionalidades e o código fonte para melhor compreender um sistema?**

( ) Sim ( ) Não

**13) Você já montou uma estrutura de rastreabilidade para algum sistema?**

( ) Sim ( )Não

*Obrígada pela sua colaboração nesta pesquisa.*

Figura A.7: Formulário de avaliação dos analistas participantes - página 7.

# Anexo B

# Questionários dos Experimentos

As próximas seções, deste anexo, apresentam os questionários preenchidos pelos participantes durante os experimentos realizados. Estes questionários apresentam perguntas que visam responder às perguntas de pesquisa.

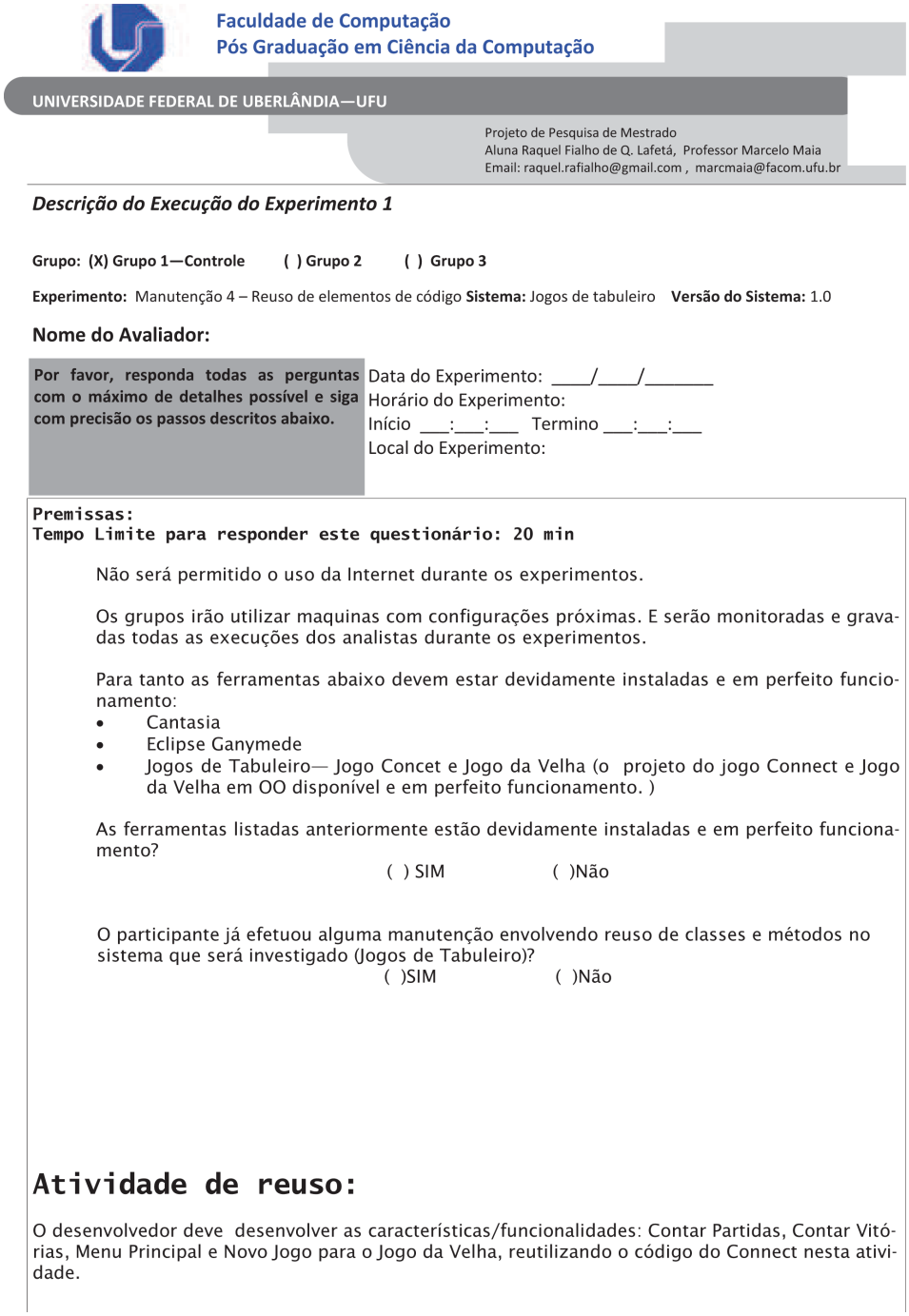
## B.1 Questionário Experimento 1 - Reuso

### B.1.1 Questionário do Grupo Controle - Experimento 1 - Reuso

As Figuras B.1, B.2 e B.3 apresentam as páginas do questionário aplicado ao grupo *Controle* no experimento 1 - Reuso.

### B.1.2 Questionário dos Grupos Parcial e Completa - Experimento 1 - Reuso

As Figuras B.4, B.5, B.6, B.7 e B.8 apresentam as páginas do questionário aplicado aos grupos *Parcial e Completa* no experimento 1 - Reuso.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento 1***

**Grupo: (X) Grupo 1—Controle ( ) Grupo 2 ( ) Grupo 3**

**Experimento:** Manutenção 4 – Reuso de elementos de código **Sistema:** Jogos de tabuleiro **Versão do Sistema:** 1.0

**Nome do Avaliador:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos abaixo.**

Data do Experimento: ____/____/_____
Horário do Experimento:
Início ___:___:___ Termino ___:___:___
Local do Experimento:

**Premissas:**
**Tempo Limite para responder este questionário: 20 min**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar maquinas com configurações próximas. E serão monitoradas e gravadas todas as execuções dos analistas durante os experimentos.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Cantasia
- Eclipse Ganymede
- Jogos de Tabuleiro— Jogo Concet e Jogo da Velha (o projeto do jogo Connect e Jogo da Velha em OO disponível e em perfeito funcionamento. )

As ferramentas listadas anteriormente estão devidamente instaladas e em perfeito funcionamento?

( ) SIM ( )Não

O participante já efetuou alguma manutenção envolvendo reuso de classes e métodos no sistema que será investigado (Jogos de Tabuleiro)?

( )SIM ( )Não

## Atividade de reuso:

O desenvolvedor deve desenvolver as características/funcionalidades: Contar Partidas, Contar Vitórias, Menu Principal e Novo Jogo para o Jogo da Velha, reutilizando o código do Connect nesta atividade.

Figura B.1: Questionário do grupo Controle no experimento 1 - Reuso - página 1.

A tela do jogo da velha deve ficar da seguinte maneira:


**Descrição das Características/Funcionalidades solicitadas:**

- **Contar Partidas:** Esta funcionalidade deve contar o número de partidas realizadas pelos jogadores e disponibilizar este valor em um menu lateral do Jogo da Velha;

- **Contar Vitórias:** Esta funcionalidade deve contar o número de vitórias de cada jogador no conjunto de partidas realizadas, e disponibilizar este valor em um menu lateral do Jogo da Velha;

- **Menu Principal:** Esta funcionalidade se caracteriza em um Botão que possibilita voltar ao menu principal (tela inicial do jogo—figura abaixo) dos jogos de tabuleiro.

- **Novo Jogo:** Esta funcionalidade reinicia o jogo e incrementa o número de partidas.

**Resposta e teste da precisão:**

**1) Indique aqui as classes e métodos que você reutilizou na atividade solicitada e como.**

–
–
–
–
–
–
–
–
–
–

**Perguntas:**

**Pergunta para todos os participantes**

- 2) Vocês consideraram esta atividade de manutenção como :
  ( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- 3) Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–
–

Figura B.2: Questionário do grupo Controle no experimento 1 - Reuso - página 2.

- 4) Utilizou alguma funcionalidade do IDE Eclipse pra auxiliar na atividade? Qual?
  ( )SIM ( )Não

-

- 5) Resuma o procedimento e ferramental utilizado no desenvolvimento:

-
-
-
-
-
-

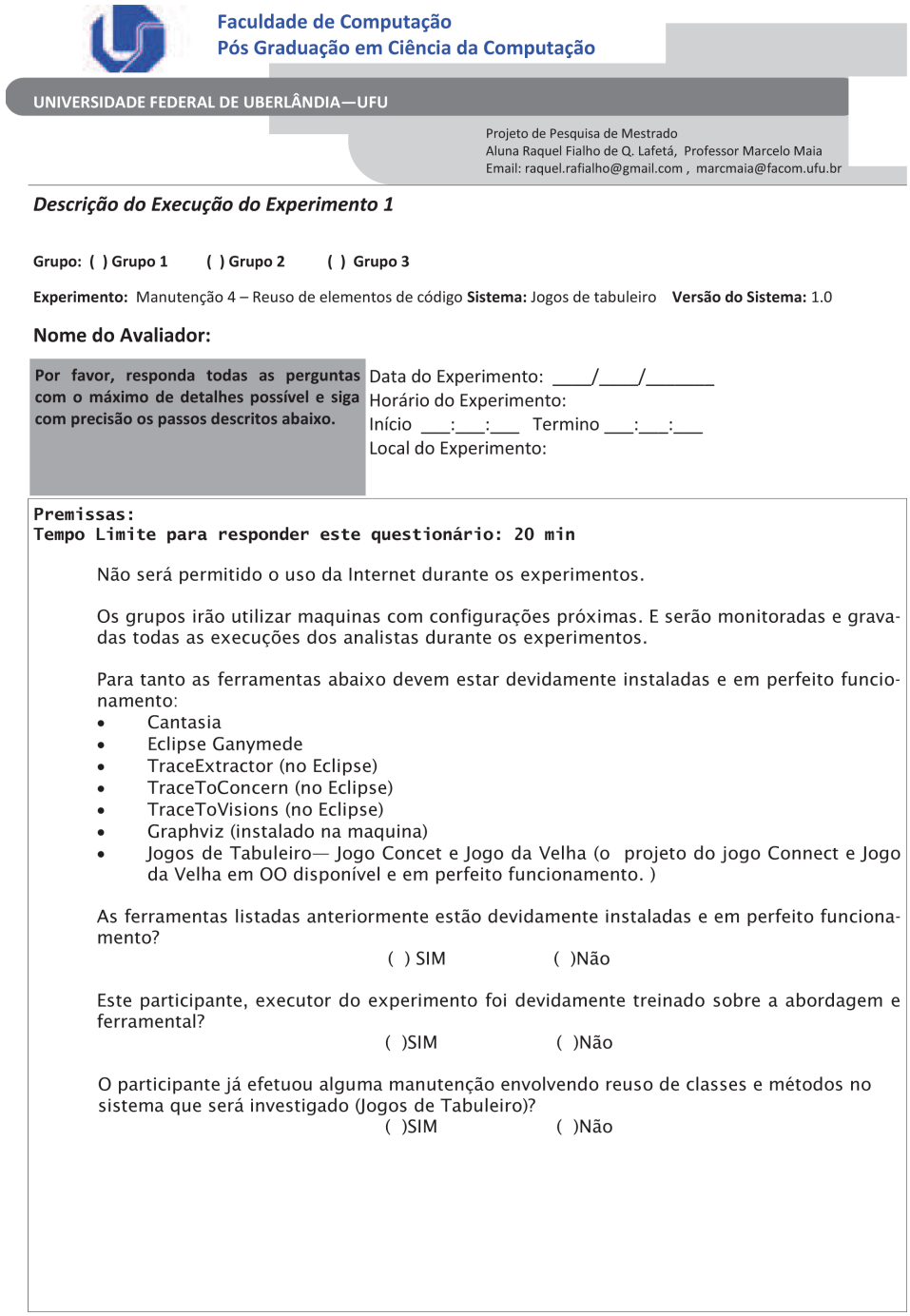
- 6) O IDE propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção?
  ( )SIM ( )Não

- 7) O IDE, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?
  ( )SIM ( )Não

-

*Obrígada pela sua colaboração!*

Figura B.3: Questionário do grupo Controle no experimento 1 - Reuso - página 3.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento 1***

**Grupo: ( ) Grupo 1 ( ) Grupo 2 ( ) Grupo 3**

**Experimento:** Manutenção 4 – Reuso de elementos de código **Sistema:** Jogos de tabuleiro **Versão do Sistema:** 1.0

**Nome do Avaliador:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos abaixo.**

Data do Experimento: ____/____/_____
Horário do Experimento:
Início ___:___:___ Termino ___:___:___
Local do Experimento:

**Premissas:**
**Tempo Limite para responder este questionário: 20 min**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar maquinas com configurações próximas. E serão monitoradas e gravadas todas as execuções dos analistas durante os experimentos.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Cantasia
- Eclipse Ganymede
- TraceExtractor (no Eclipse)
- TraceToConcern (no Eclipse)
- TraceToVisions (no Eclipse)
- Graphviz (instalado na maquina)
- Jogos de Tabuleiro— Jogo Concet e Jogo da Velha (o projeto do jogo Connect e Jogo da Velha em OO disponível e em perfeito funcionamento. )

As ferramentas listadas anteriormente estão devidamente instaladas e em perfeito funcionamento?

( ) SIM ( )Não

Este participante, executor do experimento foi devidamente treinado sobre a abordagem e ferramental?

( )SIM ( )Não

O participante já efetuou alguma manutenção envolvendo reuso de classes e métodos no sistema que será investigado (Jogos de Tabuleiro)?

( )SIM ( )Não

Figura B.4: Questionário dos grupos Parcial e Completa no experimento 1 - Reuso - página 1.

## Atividade de reuso:

O desenvolvedor deve desenvolver as características/funcionalidades: Contar Partidas, Contar Vitórias, Menu Principal e Novo Jogo para o Jogo da Velha, reutilizando o código do Connect nesta atividade.

A tela do jogo da velha deve ficar da seguinte maneira:


**Descrição das Características/Funcionalidades solicitadas:**

- **Contar Partidas:** Esta funcionalidade deve contar o número de partidas realizadas pelos jogadores e disponibilizar este valor em um menu lateral do Jogo da Velha;

- **Contar Vitórias:** Esta funcionalidade deve contar o número de vitórias de cada jogador no conjunto de partidas realizadas, e disponibilizar este valor em um menu lateral do Jogo da Velha;

- **Menu Principal:** Esta funcionalidade se caracteriza em um Botão que possibilita voltar ao menu principal (tela inicial do jogo—figura abaixo) dos jogos de tabuleiro.


- **Novo Jogo:** Esta funcionalidade reinicia o jogo e incrementa o número de partidas.

Figura B.5: Questionário dos grupos Parcial e Completa no experimento 1 - Reuso - página 2.

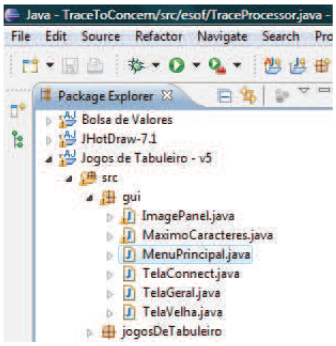
**Roteiros:**

Este roteiro deverá ser seguido, somente, pelos analistas que irão utilizar a abordagem, através deste roteiro os rastros serão coletados corretamente.

**Primeiro passo:**

Crie uma pasta com o seguinte nome: "Manutenção2_Reuso_NomeDoParticipante" , dentro desta pasta crie outras duas pastas com os seguintes nomes: "Rastros" e "Resultados".

O analista deverá inicia um jogo do tipo Connect MultiPlayer (rodar a classe MenuPrincipal.java como projeto AspectJ) e guardar os rastros para cada uma das funcionalidades da maneira descrita abaixo.




**Segundo passo é coletar os rastros:**

- **Contar Vitorias e Contar Partidas:** Analista deve iniciar o jogo **Conect Multiplayer**, e jogar de maneira que um dos jogadores ganhe a partida (como é multiplayer, o analista irá jogar pelos dois jogares e induzir um a ganhar), no momento em que o usuário ganhar irá aparecer uma janela informando que o jogador X venceu, neste momento o usuário deverá, no TraceExtractor dar o nome da característica (nome: **Contar Vitorias e Partidas**) e ativar o "Start" do TraceExtractor (abordagem), apertar o botão "OK" e, logo depois, ativar o "End" do TraceExtractor.

- **Menu Principal:** O usuário deve iniciar um **Conect do tipo MultiPlayer**, com o jogo em uso, o usuário deverá, no TraceExtractor, dar o nome da característica (**Menu Principal**) e ativar o "Start" (abordagem), apertar o botão "Menu principal" (a funcionalidade será executada e voltará para o menu principal dos jogos de tabuleiro) e ativar o "End" (abordagem);

- **Novo Jogo:** O usuário deve iniciar um **Conect do tipo MultiPlayer**, com o jogo em uso, o usuário deverá, no TraceExtractor, dar o nome da característica (**Novo Jogo**) e ativar o "Start" (abordagem), apertar o botão "Novo Jogo" (a funcionalidade será executada e o jogo será reiniciado e será contado no "Contar partida" (incrementando o valor) como uma partida realizada, logo depois o analista deve ativar o "End" (abordagem);

Figura B.6: Questionário dos grupos Parcial e Completa no experimento 1 - Reuso - página 3.

**Resposta e teste da precisão:**

**1) Indique aqui as classes e métodos que você reutilizou na atividade solicitada e como.**

–
–
–
–
–
–
–
–
–
–

**Perguntas:**

**Pergunta para todos os participantes**

- 2) Vocês consideraram esta atividade de manutenção como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- 3) Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–
–
–

**Responda as perguntas abaixo somente se fez uso da abordagem:**

- 4) A abordagem o auxiliou na compreensão do sistema para execução da atividade?
( ) SIM ( )NÃO

- 5) A abordagem (procedimentos e ferramental) listou elementos de código (classes e métodos) que foram reutilizados na nova funcionalidade? Se sim, quais?

–
–
–
–
–

- 6) A abordagem (procedimentos e ferramental) **não** listou algum elemento de código (classes e métodos) que foram reutilizados na nova funcionalidade? Se sim, quais?

–
–
–
–
–

- 7) Quais visões da abordagem você utilizou?

( ) Visão 1— ConcernMapper
( ) Visão 2— Call Graph
( ) Visão 3– Corte Funcional
( ) Visão 4—Corte Funcional da Característica

Figura B.7: Questionário dos grupos Parcial e Completa no experimento 1 - Reuso - página 4.

( ) Nenhuma das anteriores

- 8) Quais visões o auxiliou realmente na compreensão do problema e a encontrar onde deve ocorrer a correção:

( ) Visão 1— ConcernMapper
( ) Visão 2— Call Graph
( ) Visão 3– Corte Funcional
( ) Visão 4—Corte Funcional da Característica
( ) Nenhuma das anteriores

- 9) Como estas visões o auxiliaram na atividade de manutenção?

–
–
–
–
–
–

- 10) Como você define o grau de usabilidade da abordagem (ferramental)?

( )Excelente
( )Boa
( )Regular
( )Péssima

- 11) A granularidade das visões foi suficiente para auxiliar na descrição da característica no propósito de auxiliar em uma manutenção? Justifique.
( ) SIM ( )NÃO

- 12) A abordagem propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção? Justifique.
( )SIM ( )NÃO

- 13) E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?
( )SIM ( )NÃO

- 14) A Abordagem permite que o programador indique os requisitos que fundamentam as Unidades Computacionais (Classes e Métodos) diretamente do ambiente de codificação, para não haver interrupção no trabalho de programação.
( )SIM ( )NÃO

- 15) O que você achou da abordagem montada? Como você acredita que ela possa contribuir para a compreensão de um código fonte.

Figura B.8: Questionário dos grupos Parcial e Completa no experimento 1 - Reuso - página 5.

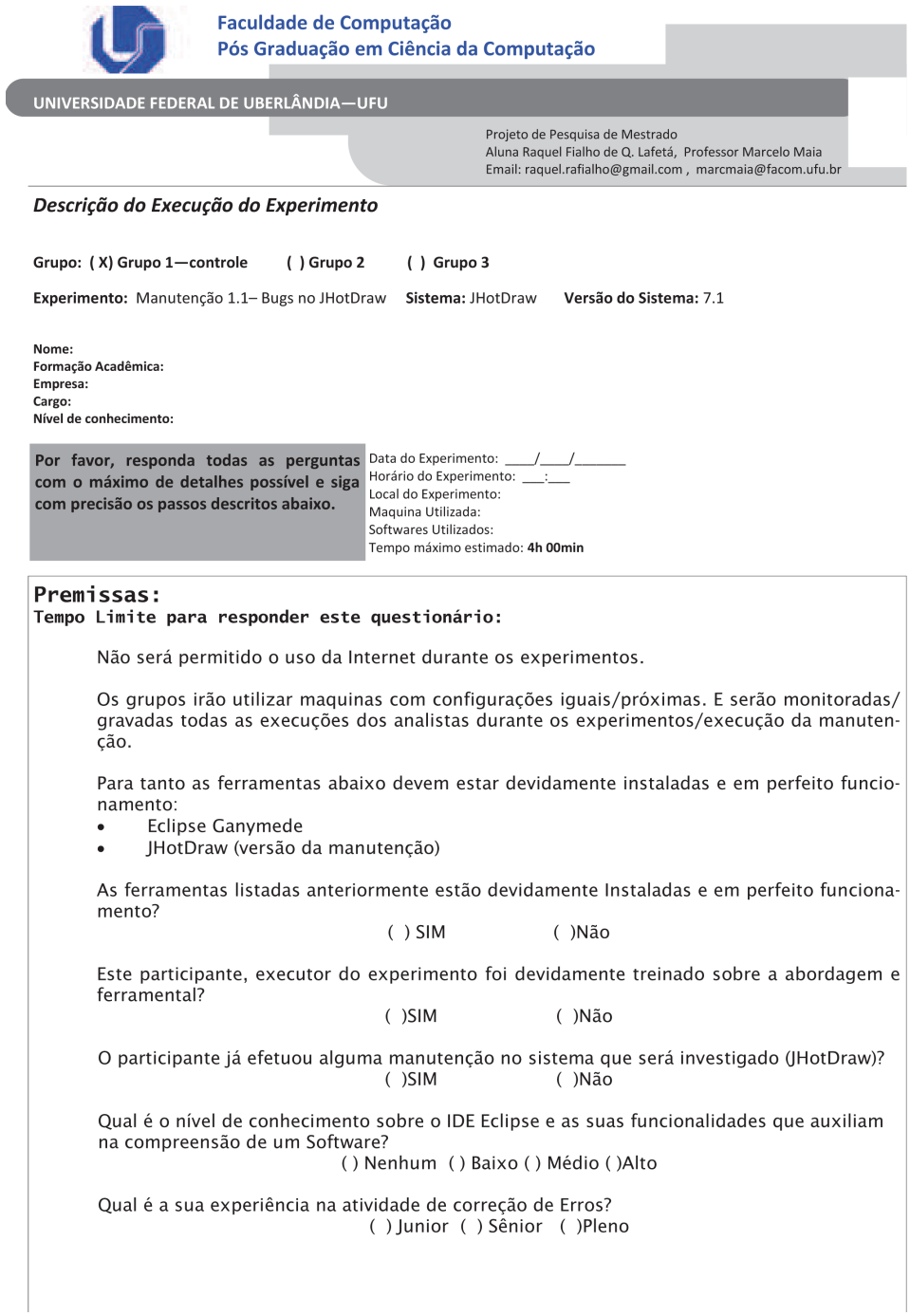
## B.2 Questionário Experimento 2 - Correção

### B.2.1 Questionário do Grupo Controle - Experimento 2 - Correção

As Figuras B.9, B.10, B.11, B.12 e B.13 apresentam as páginas do questionário aplicado ao grupo *Controle* no experimento 2 - Correção.

### B.2.2 Questionário dos Grupos Parcial e Completa - Experimento 2 - Correção

As Figuras B.14, B.15, B.16, B.17, B.18 e B.19 apresentam as páginas do questionário aplicado aos grupos *Parcial e Completa* no experimento 2 - Correção.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento***

**Grupo:** **( X) Grupo 1—controle** **( ) Grupo 2** **( ) Grupo 3**

**Experimento:** Manutenção 1.1– Bugs no JHotDraw **Sistema:** JHotDraw **Versão do Sistema:** 7.1

**Nome:**
**Formação Acadêmica:**
**Empresa:**
**Cargo:**
**Nível de conhecimento:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos abaixo.**

Data do Experimento: ____/____/_____
Horário do Experimento: ___:___
Local do Experimento:
Maquina Utilizada:
Softwares Utilizados:
Tempo máximo estimado: **4h 00min**

**Premissas:**
**Tempo Limite para responder este questionário:**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar maquinas com configurações iguais/próximas. E serão monitoradas/ gravadas todas as execuções dos analistas durante os experimentos/execução da manutenção.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:
- Eclipse Ganymede
- JHotDraw (versão da manutenção)

As ferramentas listadas anteriormente estão devidamente Instaladas e em perfeito funcionamento?
( ) SIM ( )Não

Este participante, executor do experimento foi devidamente treinado sobre a abordagem e ferramental?
( )SIM ( )Não

O participante já efetuou alguma manutenção no sistema que será investigado (JHotDraw)?
( )SIM ( )Não

Qual é o nível de conhecimento sobre o IDE Eclipse e as suas funcionalidades que auxiliam na compreensão de um Software?
( ) Nenhum ( ) Baixo ( ) Médio ( )Alto

Qual é a sua experiência na atividade de correção de Erros?
( ) Junior ( ) Sênior ( )Pleno

Figura B.9: Questionário do grupo Controle no experimento 2 - Correção- página 1.

**Objetivo:**

Descobrir onde um determinado "Bug" (Erro) deve ser corrigido no sistema. Segue a descrição de um determinado problema no sistema em análise, para que seja corrigido pelo desenvolvedor as classes e métodos envolvidas na atividade.

**Roteiro para se obter o erro:**

**Tempo limite para executar o experimento:**

A funcionalidade TextTool permite criar novos textos dentro de figuras e modificar os textos existente dentro destas. Porém esta funcionalidade possui um erros na versão 7.1 do sistema JHotDraw. A funcionalidade Undo sobre os objetos do tipo TextTool não funciona. O Undo é uma funcionalidade do JHotDraw que desfaz as alterações executadas sobre a figura em construção.

**Reporduzindo o erro/Roteiro de execução:**

**1 passo:** Crie no Jhotdraw uma figura, um quadrado;

**2 passo:** selecione a funcionalidade Text (ver figura 1) e clique sobre o quadrado, irá aparecer uma caixa de texto, digite o seu nome na caixa e tecle ENTER (ver figura 2);

**3 passo:** O usuário deve clicar no botão EDIT para visualizar todas as possibilidade de Undo disponíveis (figura 2), repare que o último Undo deveria ser o Undo digitar texto, mas devido ao erro, o único Undo que aparece é o "Undo Create Figure" que foi a última ação antes da criação do texto (quando foi criado o quadrado).

**Correção:**

Corrija este erro, utilizando a abordagem desenvolvida para reproduzir os interesses e coletar os rastros que podem auxiliar nesta correção.

A seguinte correção deve ser executada, o sistema JHotDraw 7.1 deve ser capaz de realizar o Undo sobre a ferramenta do tipo TextTool—Text. Deve aparecer no menu do EDIT a opção de "Undo TextTool", que deve desfazer a uma alteração realizada sobre este objeto. A funcionalidade Undo guarda o histórico de alterações e conforme se utiliza o Undo (ctrl z) ele regride todas as alterações na ordem que estas foram realizadas.

**É importante lembrar que esta funcionalidade já está implementada e somente necessita de uma correção.**

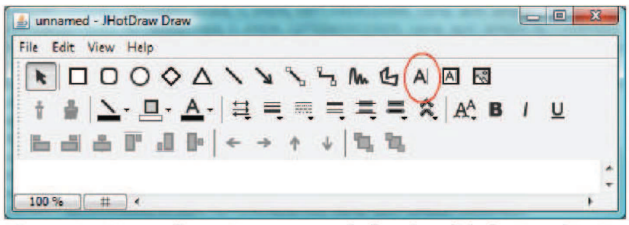



**Figura 1— Marcação na Ferramenta da funcionalidade Text (TexTool)**

Figura B.10: Questionário do grupo Controle no experimento 2 - Correção- página 2.

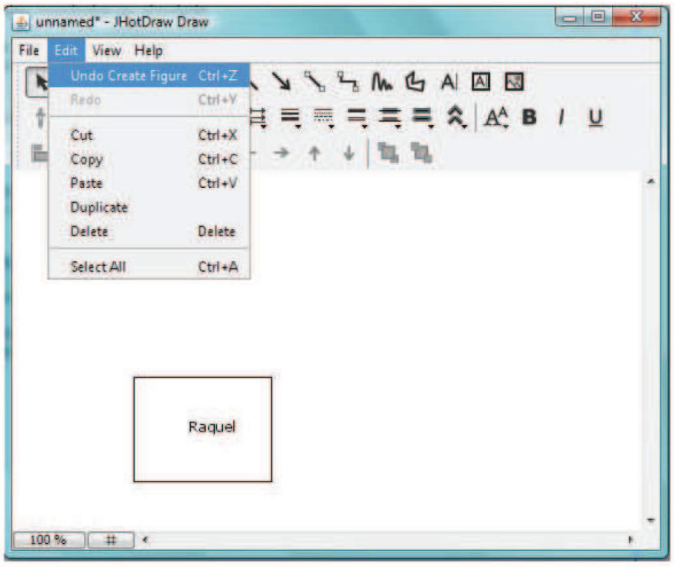



**Figura 2— Reproduzindo o erro**

**Resultado:**
**Tempo para responder as perguntas:**

**Indique aqui as classes e métodos envolvidos na correção:**

–
–
–
–
–
–

**Qual foi a manutenção realizada para corrigir o erro?**

–
–
–
–
–
–
–
–
–
–

**Perguntas:**

**Pergunta para todos os participantes**

- Vocês consideraram esta atividade de manutenção como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- Qual foi a maior dificuldade encontrada durante a execução da atividade?

**Se não utilizou a abordagem responda as perguntas abaixo:**

- Utilizou alguma funcionalidade do IDE Eclipse para auxiliar na atividade? Qual?
( ) SIM ( )Não

Resuma o procedimento e ferramental utilizado para encontrar a solução do Erro:

Figura B.12: Questionário do grupo Controle no experimento 2 - Correção- página 4.

- O IDE propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção?

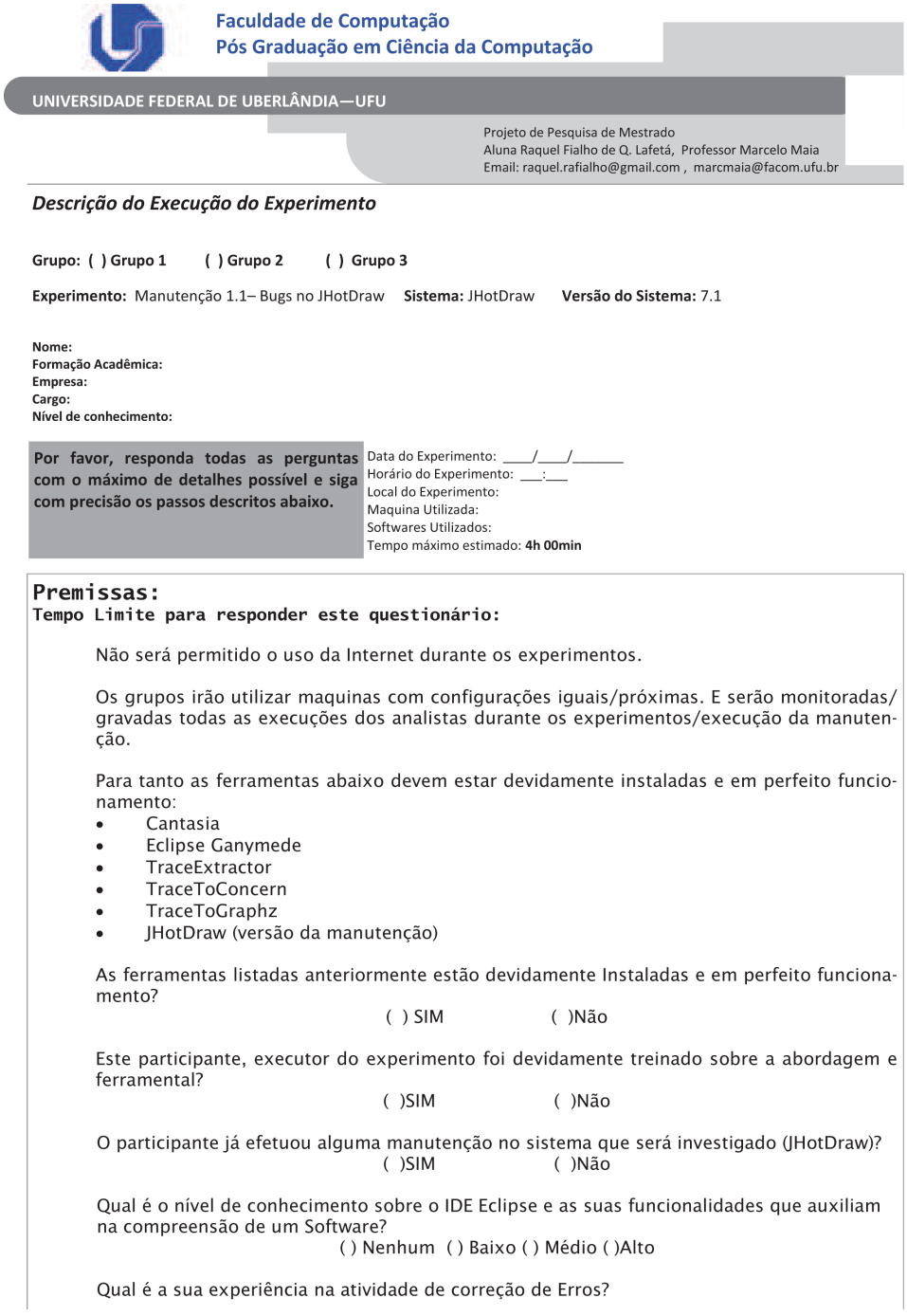
  ( )SIM ( )NÃO

- E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?

  ( )SIM ( )NÃO

*Obrigada pela sua colaboração!*

Figura B.13: Questionário do grupo Controle no experimento 2 - Correção- página 5.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento***

**Grupo: ( ) Grupo 1 ( ) Grupo 2 ( ) Grupo 3**

**Experimento:** Manutenção 1.1– Bugs no JHotDraw **Sistema:** JHotDraw **Versão do Sistema:** 7.1

**Nome:**
**Formação Acadêmica:**
**Empresa:**
**Cargo:**
**Nível de conhecimento:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos abaixo.**

Data do Experimento: ____/____/_____
Horário do Experimento: ___:___
Local do Experimento:
Maquina Utilizada:
Softwares Utilizados:
Tempo máximo estimado: **4h 00min**

**Premissas:**
**Tempo Limite para responder este questionário:**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar maquinas com configurações iguais/próximas. E serão monitoradas/ gravadas todas as execuções dos analistas durante os experimentos/execução da manutenção.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Cantasia
- Eclipse Ganymede
- TraceExtractor
- TraceToConcern
- TraceToGraphz
- JHotDraw (versão da manutenção)

As ferramentas listadas anteriormente estão devidamente Instaladas e em perfeito funcionamento?

( ) SIM ( )Não

Este participante, executor do experimento foi devidamente treinado sobre a abordagem e ferramental?

( )SIM ( )Não

O participante já efetuou alguma manutenção no sistema que será investigado (JHotDraw)?

( )SIM ( )Não

Qual é o nível de conhecimento sobre o IDE Eclipse e as suas funcionalidades que auxiliam na compreensão de um Software?

( ) Nenhum ( ) Baixo ( ) Médio ( )Alto

Qual é a sua experiência na atividade de correção de Erros?

Figura B.14: Questionário dos grupos Parcial e Completa no experimento 2 - Correção- página 1.

( ) Junior ( ) Sênior ( )Pleno

**Objetivo:**

Descobrir onde um determinado "Bug" (Erro) deve ser corrigido no sistema. Segue a descrição de um determinado problema no sistema em análise, para que seja corrigir pelo desenvolvedor as classes e métodos envolvidas na atividade.

**Roteiro para se obter o erro:**

**Tempo limite para executar o experimento:**

A funcionalidade TextTool permite criar novos textos dentro de figuras e modificar os textos existente dentro destas. Porém esta funcionalidade possui um erros na versão 7.1 do sistema JHotDraw. A funcionalidade Undo sobre os objetos do tipo TextTool não funciona. O Undo é uma funcionalidade do JHotDraw que desfaz as alterações executadas sobre a figura em construção.

**Reporduzindo o erro/Roteiro de execução:**

**1 passo:** Crie no Jhotdraw uma figura, um quadrado;

**2 passo:** selecione a funcionalidade Text (ver figura 1) e clique sobre o quadrado, irá aparecer uma caixa de texto, digite o seu nome na caixa e tecle ENTER (ver figura 2);

**3 passo:** O usuário deve clicar no botão EDIT para visualizar todas as possibilidade de Undo disponíveis (figura 2), repare que o último Undo deveria ser o Undo digitar texto, mas devido ao erro, o único Undo que aparece é o "Undo Create Figure" que foi a última ação antes da criação do texto (quando foi criado o quadrado).

**Correção:**

Corrija este erro, utilizando a abordagem desenvolvida para reproduzir os interesses e coletar os rastros que podem auxiliar nesta correção.

A seguinte correção deve ser executada, o sistema JHotDraw 7.1 deve ser capaz de realizar o Undo sobre a ferramenta do tipo TextTool—Text. Deve aparecer no menu do EDIT a opção de "Undo TextTool", que deve desfazer a uma alteração realizada sobre este objeto. A funcionalidade Undo guarda o histórico de alterações e conforme se utiliza o Undo (ctrl z) ele regride todas as alterações na ordem que estas foram realizadas.

**É importante lembrar que esta funcionalidade já está implementada e somente necessita de uma correção.**

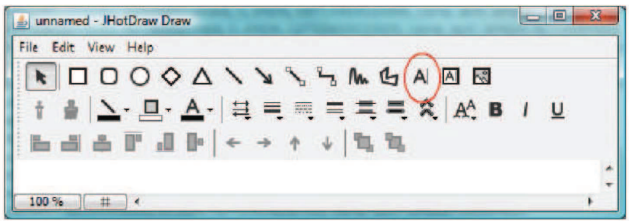



**Figura 1— Marcação na Ferramenta da funcionalidade Text (TexTool)**

Figura B.15: Questionário dos grupos Parcial e Completa no experimento 2 - Correção-página 2.

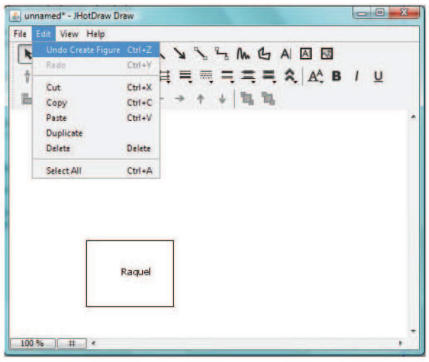



Figura 2— Reproduzindo o erro

**Resultado:**

**Tempo para responder as perguntas:**

**Indique aqui as classes e métodos envolvidos na correção:**

–
–
–
–
–
–

**Qual foi a manutenção realizada para corrigir o erro?**

–
–
–
–
–
–
–
–
–
–
–

Figura B.16: Questionário dos grupos Parcial e Completa no experimento 2 - Correção-página 3.

**Perguntas:**
**Tempo para responder as perguntas:**

**Pergunta para todos os participantes**

- Vocês consideraram esta atividade de manutenção como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–
–
–

**Responda as perguntas abaixo somente se fez uso da abordagem:**

- A abordagem o auxiliou na compreensão do problema e em encontrar o mesmo?
( ) SIM ( )NÃO

- A abordagem (procedimentos e ferramental) listou todos os elementos (classes e métodos) que estão envolvidos na correção?

–
–
–
–

–

Figura B.17: Questionário dos grupos Parcial e Completa no experimento 2 - Correção-página 4.

- Quais visões da abordagem você utilizou?

( )Visão 1— ConcernMapper
( )Visão 2— Call Graph
( )Visão 3– Corte Funcional
( )Visão 4—Corte Funcional da Característica
( ) Nenhuma das anteriores

- Quais visões o auxiliou realmente na compreensão do problema e a encontrar onde deve ocorrer a correção:

( )Visão 1— ConcernMapper
( )Visão 2— Call Graph
( )Visão 3– Corte Funcional
( )Visão 4—Corte Funcional da Característica
( ) Nenhuma das anteriores

- Como estas visões o auxiliaram na atividade de manutenção?

–
–
–
–
–

- Como você define o grau de usabilidade da abordagem (ferramental)?

( )Excelente
( )Boa
( )Regular
( )Péssima

- A granularidade das visões foi suficiente para auxiliar na descrição da característica no propósito de auxiliar em uma manutenção? Justifique.

( ) SIM ( )NÃO

- A abordagem propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção? Justifique.

( )SIM ( )NÃO

Figura B.18: Questionário dos grupos Parcial e Completa no experimento 2 - Correção-página 5.

- E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?

  ( )SIM ( )NÃO

- A Abordagem permite que o programador indique os requisitos que fundamentam as Unidades Computacionais diretamente do ambiente de codificação, para não haver interrupção no trabalho de programação.

  ( )SIM ( )NÃO

*Obrigada pela sua colaboração!*

Figura B.19: Questionário dos grupos Parcial e Completa no experimento 2 - Correção-página 6.

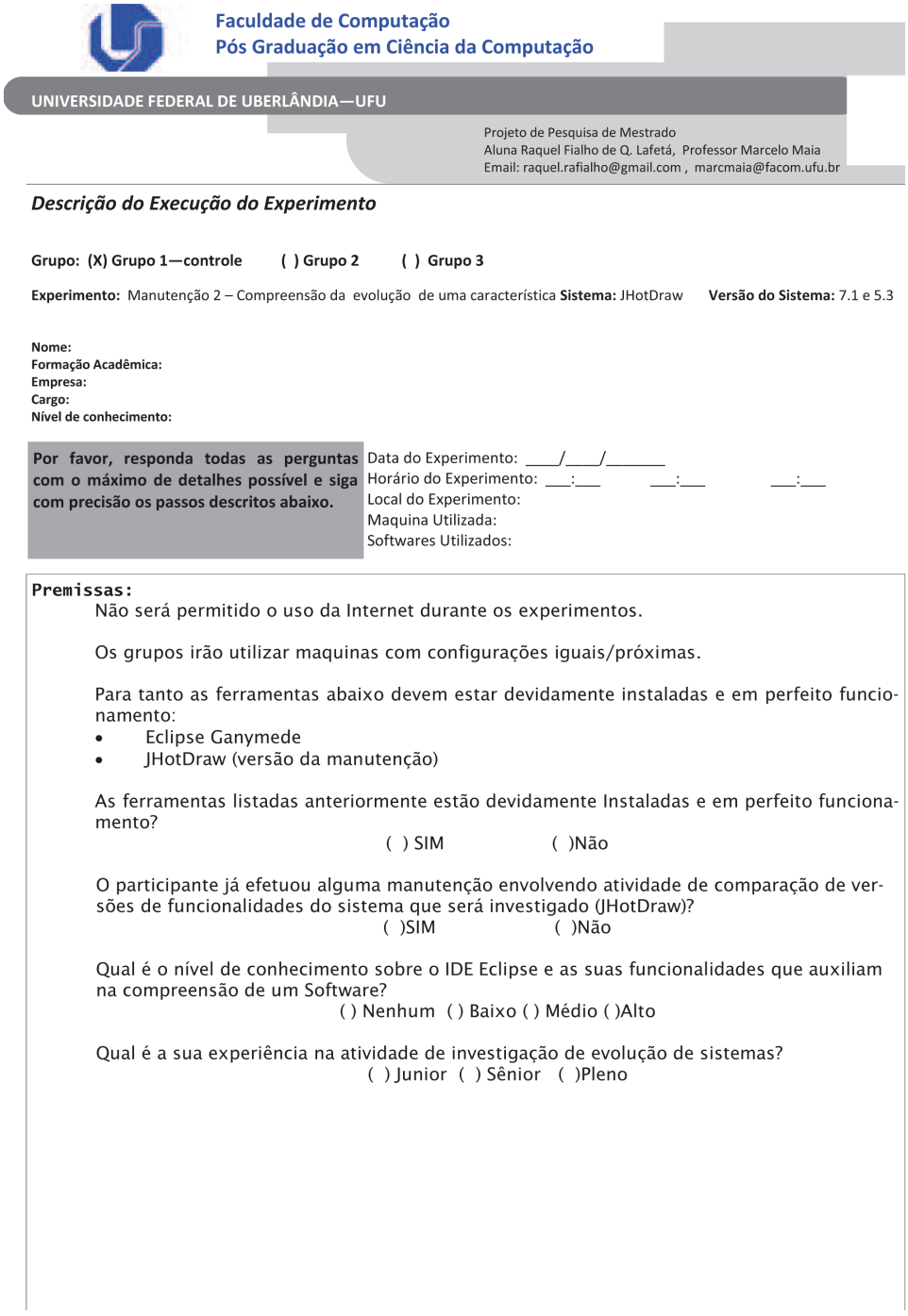
## B.3 Questionário Experimento 3 - Localização

### B.3.1 Questionário do Grupo Controle - Experimento 3 - Localização

As Figuras B.20, B.21, B.22, B.23 e B.24 apresentam as páginas do questionário aplicado ao grupo *Controle* no experimento 3 - Localização.

### B.3.2 Questionário dos Grupos Parcial e Completa - Experimento 3 - Localização

As Figuras B.25, B.26, B.27, B.28 e B.29 apresentam as páginas do questionário aplicado aos grupos *Parcial e Completa* no experimento 3 - Localização.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento***

**Grupo: (X) Grupo 1—controle ( ) Grupo 2 ( ) Grupo 3**

**Experimento:** Manutenção 2 – Compreensão da evolução de uma característica **Sistema:** JHotDraw **Versão do Sistema:** 7.1 e 5.3

**Nome:**
**Formação Acadêmica:**
**Empresa:**
**Cargo:**
**Nível de conhecimento:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos abaixo.**

Data do Experimento: ____/____/_____
Horário do Experimento: ___:___ ___:___ ___:___
Local do Experimento:
Maquina Utilizada:
Softwares Utilizados:

**Premissas:**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar maquinas com configurações iguais/próximas.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Eclipse Ganymede
- JHotDraw (versão da manutenção)

As ferramentas listadas anteriormente estão devidamente Instaladas e em perfeito funcionamento?

( ) SIM ( )Não

O participante já efetuou alguma manutenção envolvendo atividade de comparação de versões de funcionalidades do sistema que será investigado (JHotDraw)?

( )SIM ( )Não

Qual é o nível de conhecimento sobre o IDE Eclipse e as suas funcionalidades que auxiliam na compreensão de um Software?

( ) Nenhum ( ) Baixo ( ) Médio ( )Alto

Qual é a sua experiência na atividade de investigação de evolução de sistemas?

( ) Junior ( ) Sênior ( )Pleno

Figura B.20: Questionário do grupo Controle no experimento 3 - Localização- página 1.

**Objetivo:**

Apontar as diferenças/evoluções que ocorreram em uma dada característica para versões distintas de um sistema.

Será avaliada a evolução das características/funcionalidades Undo/Redo do JHotDraw que ocorreu na versão 7.1. O executor deste experimento deverá avaliar as versões 5.3 e 7.1 e apontar qual foi a evolução que ocorreu nas características Undo/Redo de uma versão para a outra, os pontos de divergência e em comum.

**Reproduzindo a Característica/Roteiro:**

O executor deste experimento deve reproduzir os interesses na versão 7.1 e na versão 5.3.

Para reproduzir as funcionalidade, o executor deve desenhar uma figura utilizando todos as funcionalidades **do sistema**, logo depois ele deve coletar separadamente os rastros de cada opção de Undo (Undo desenhar, Undo pintar, Undo transformar, etc.) e Redo presentes no Menu (EDIT) no JHotDraw.

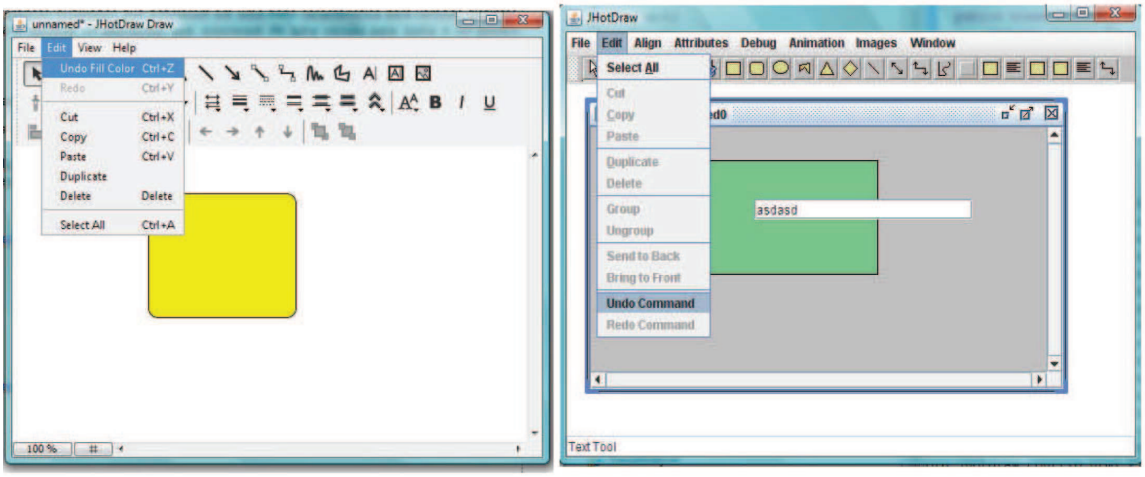



**Figura2 - Exemplo de uso do Undo/Redo na versão 7.1 e na versão 5.3 respectivamente.**

Figura B.21: Questionário do grupo Controle no experimento 3 - Localização- página 2.

**Perguntas:**

**Quais classes estão relacionadas ao Undo/Redo do JHotDraw 7.1?**

**Quais classes estão relacionadas ao Undo/Redo do JHotDraw 5.3?**

**Resuma qual foi a alteração que ocorreu nas Características Undo/Redo da versão 5.3 para a Versão 7.1 do JHotDraw.**

-
-

Figura B.22: Questionário do grupo Controle no experimento 3 - Localização- página 3.

**Perguntas gerais:**

- Vocês consideraram esta atividade compreensão como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–

**Se não utilizou a abordagem responda as perguntas abaixo:**

- Utilizou alguma funcionalidade do IDE Eclipse para auxiliar na atividade? Qual?

( ) SIM ( )Não

–

- Resuma o procedimento e ferramental utilizado para executar o experimento:

–
–

–
–
–

- O IDE propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção?

( )SIM ( )NÃO

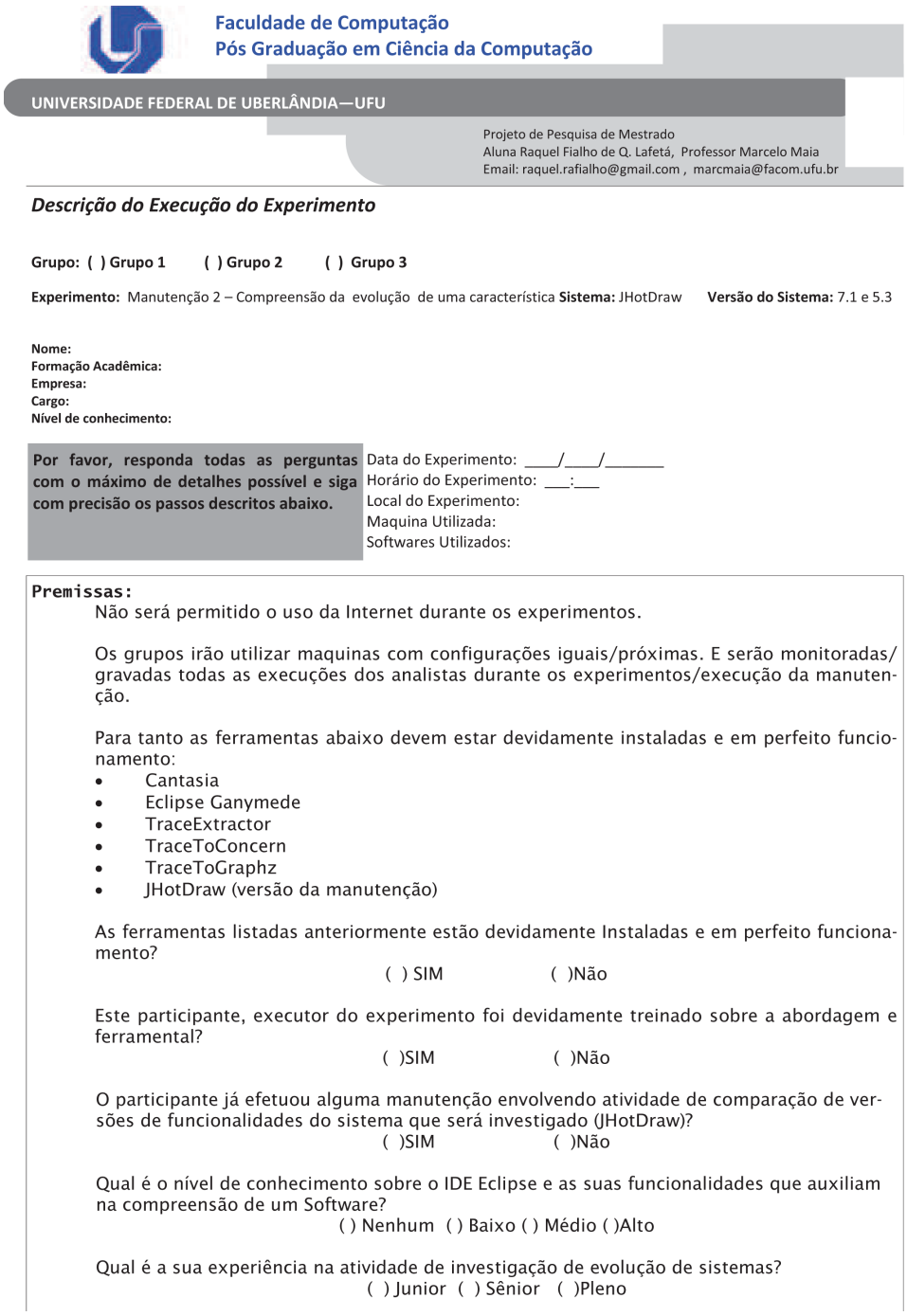
Figura B.23: Questionário do grupo Controle no experimento 3 - Localização- página 4.

- E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?

( )SIM ( )NÃO

*Obrigada pela sua colaboração!*

Figura B.24: Questionário do grupo Controle no experimento 3 - Localização- página 5.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento***

**Grupo: ( ) Grupo 1 ( ) Grupo 2 ( ) Grupo 3**

**Experimento:** Manutenção 2 – Compreensão da evolução de uma característica **Sistema:** JHotDraw **Versão do Sistema:** 7.1 e 5.3

**Nome:**
**Formação Acadêmica:**
**Empresa:**
**Cargo:**
**Nível de conhecimento:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos abaixo.**

Data do Experimento: ____/____/_____
Horário do Experimento: ___:___
Local do Experimento:
Maquina Utilizada:
Softwares Utilizados:

**Premissas:**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar maquinas com configurações iguais/próximas. E serão monitoradas/gravadas todas as execuções dos analistas durante os experimentos/execução da manutenção.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Cantasia
- Eclipse Ganymede
- TraceExtractor
- TraceToConcern
- TraceToGraphz
- JHotDraw (versão da manutenção)

As ferramentas listadas anteriormente estão devidamente Instaladas e em perfeito funcionamento?

( ) SIM ( )Não

Este participante, executor do experimento foi devidamente treinado sobre a abordagem e ferramental?

( )SIM ( )Não

O participante já efetuou alguma manutenção envolvendo atividade de comparação de versões de funcionalidades do sistema que será investigado (JHotDraw)?

( )SIM ( )Não

Qual é o nível de conhecimento sobre o IDE Eclipse e as suas funcionalidades que auxiliam na compreensão de um Software?

( ) Nenhum ( ) Baixo ( ) Médio ( )Alto

Qual é a sua experiência na atividade de investigação de evolução de sistemas?

( ) Junior ( ) Sênior ( )Pleno

Figura B.25: Questionário dos grupos Parcial e Completa no experimento 3 - Localização - página 1.

**Objetivo:**

Apontar as diferenças/evoluções que ocorreram em uma dada característica para versões distintas de um sistema.

Será avaliada a evolução das características/funcionalidades Undo/Redo do JHotDraw que ocorreu na versão 7.1. O executor deste experimento deverá avaliar as versões 5.3 e 7.1 e apontar qual foi a evolução que ocorreu nas características Undo/Redo de uma versão para a outra, os pontos de divergência e em comum.

**Reproduzindo a Característica/Roteiro:**

O executor deste experimento deve reproduzir os interesses na versão 7.1 e na versão 5.3 e coletar os rastros utilizando a abordagem desenvolvida.

Para reproduzir as funcionalidade, o executor deve desenhar uma figura utilizando todos as funcionalidades **do sistema**, logo depois ele deve coletar separadamente os rastros de cada opção de Undo (Undo desenhar, Undo pintar, Undo transformar, etc.) e Redo presentes no Menu (EDIT) no JHotDraw.

A coleta de rastros e uso da abordagem deve ser realizada para as duas versões separadamente. Os resultados gerados pela abordagem devem ser utilizados, visando auxiliar na comparação das duas versões e apontamento das divergências e convergências.

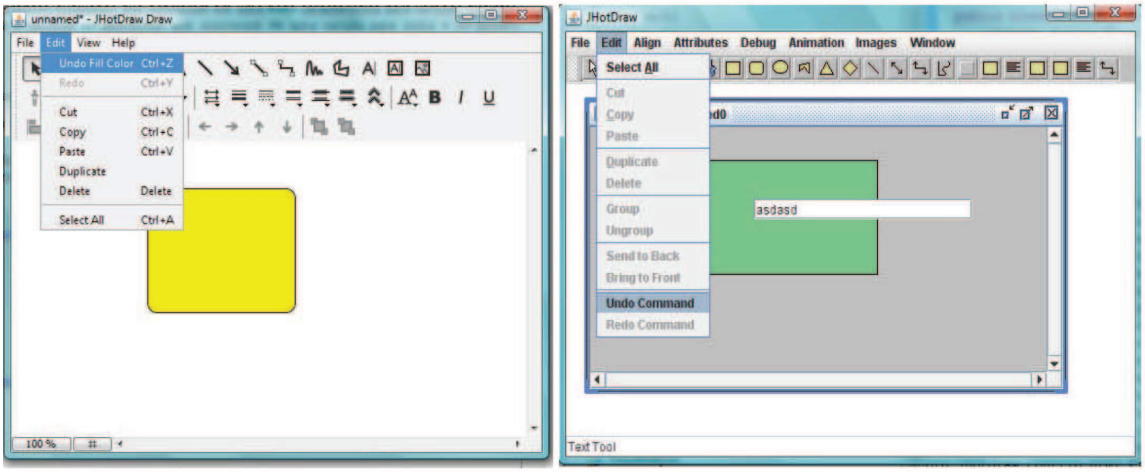



**Figura2 - Exemplo de uso do Undo/Redo na versão 7.1 e na versão 5.3 respectivamente.**

Figura B.26: Questionário dos grupos Parcial e Completa no experimento 3 - Localização - página 2.

**Perguntas:**

**Quais classes estão relacionadas ao Undo/Redo do JHotDraw 7.1?**

**Quais classes estão relacionadas ao Undo/Redo do JHotDraw 5.3?**

**Resuma qual foi a alteração que ocorreu nas Características Undo/Redo da versão 5.3 para a Versão 7.1 do JHotDraw.**

-
-

Figura B.27: Questionário dos grupos Parcial e Completa no experimento 3 - Localização - página 3.

**Perguntas gerais:**

- Vocês consideraram esta atividade compreensão como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–

–
–

- A abordagem o auxiliou na compreensão da evolução? Como?
( )SIM ( )NÃO

- A abordagem (procedimentos e ferramental) listou todos os elementos (classes e métodos) que estão envolvidos na evolução?

–
–
–
–
–

- Quais visões da abordagem você utilizou?

( )Visão 1— ConcernMapper
( )Visão 2— Call Graph
( )Visão 3– Corte Funcional
( )Visão 4—Corte Funcional da Característica
( )Nenhuma das Anteriores

- Quais visões o auxiliou realmente na compreensão da evolução e a apontar onde ocorreram as alterações:

( )Visão 1— ConcernMapper
( )Visão 2— Call Graph
( )Visão 3– Corte Funcional
( )Visão 4—Corte Funcional da Característica
( )Nenhuma das anteriores

Figura B.28: Questionário dos grupos Parcial e Completa no experimento 3 - Localização - página 4.

Como estas visões o auxiliaram na atividade de compreensão da evolução?

–
–
–
–

–

–

- Como você define o grau de usabilidade da abordagem (ferramental)?

( )Excelente
( )Boa
( )Regular
( )Péssima

- A granularidade das visões foi suficiente para auxiliar na descrição da característica no propósito de auxiliar em uma manutenção? Justifique.

( ) SIM ( )NÃO

- A abordagem propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção? Justifique.

( )SIM ( )NÃO

- E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?

( )SIM ( )NÃO

- A Abordagem permite que o programador indique os requisitos que fundamentam as Unidades Computacionais diretamente do ambiente de codificação, para não haver interrupção no trabalho de programação.

( )SIM ( )NÃO

*Obrigada pela sua colaboração!*

Figura B.29: Questionário dos grupos Parcial e Completa no experimento 3 - Localização - página 5.

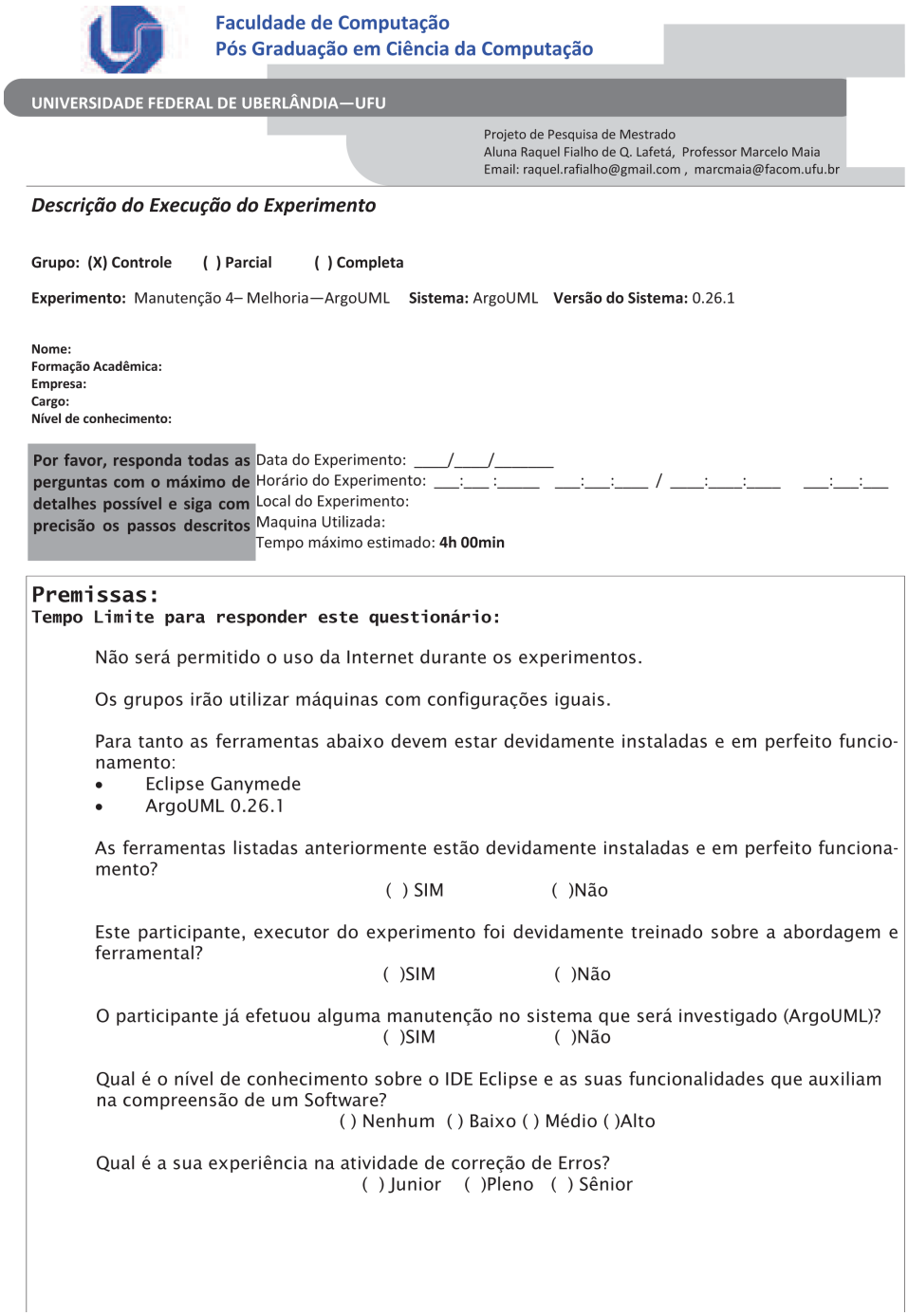
## B.4 Questionário Experimento 4 - Melhoria

### B.4.1 Questionário do Grupo Controle - Experimento 4 - Melhoria

As Figuras B.30, B.31, B.32, B.33 e B.34 apresentam as páginas do questionário aplicado ao grupo *Controle* no experimento 4 - Melhoria.

### B.4.2 Questionário dos Grupos Parcial e Completa - Experimento 4 - Melhoria

As Figuras B.35, B.36, B.37, B.38, B.39 e B.40 apresentam as páginas do questionário aplicado aos grupos *Parcial e Completa* no experimento 4 - Melhoria.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento***

**Grupo: (X) Controle ( ) Parcial ( ) Completa**

**Experimento:** Manutenção 4– Melhoria—ArgoUML **Sistema:** ArgoUML **Versão do Sistema:** 0.26.1

**Nome:**
**Formação Acadêmica:**
**Empresa:**
**Cargo:**
**Nível de conhecimento:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos**

Data do Experimento: ____/____/_____
Horário do Experimento: ___:___ :_____ ___:___:____ / ____:____:____ ___:___:___
Local do Experimento:
Maquina Utilizada:
Tempo máximo estimado: **4h 00min**

**Premissas:**
**Tempo Limite para responder este questionário:**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar máquinas com configurações iguais.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Eclipse Ganymede
- ArgoUML 0.26.1

As ferramentas listadas anteriormente estão devidamente instaladas e em perfeito funcionamento?

( ) SIM ( )Não

Este participante, executor do experimento foi devidamente treinado sobre a abordagem e ferramental?

( )SIM ( )Não

O participante já efetuou alguma manutenção no sistema que será investigado (ArgoUML)?

( )SIM ( )Não

Qual é o nível de conhecimento sobre o IDE Eclipse e as suas funcionalidades que auxiliam na compreensão de um Software?

( ) Nenhum ( ) Baixo ( ) Médio ( )Alto

Qual é a sua experiência na atividade de correção de Erros?

( ) Junior ( )Pleno ( ) Sênior

Figura B.30: Questionário do grupo Controle no experimento 4 - Melhoria - página 1.

**Objetivo:**
Executar uma melhoria no sistema ArgoUML.

**Melhoria:**
O sistema ArgoUML é uma ferramenta para desenho de diagramas UML. Uma de suas funcionalidades é a inserção de comentários associados aos objetos do diagrama. Este objeto comentário sofre aumento em sua largura conforme se insere um texto neste. Se o texto não possui quebras de linha, o objeto comentário será tão largo quanto o texto inserido. E a ferramenta não permite que este objeto seja reduzido, enquanto não inserir quebras de linha no texto.

A melhoria solicitada é que o objeto Comentário tenha tamanho máximo fixo em altura e largura e que automaticamente se tenha uma quebra de linha no texto para que este se adapte ao tamanho do objeto. Se o tamanho do texto for maior que o tamanho máximo do objeto, que deve permitir apresentar até 160 caracteres, o restante do texto deve ficar oculto.

**Roteiro de execução sugerido:**

**Passo 1:** Inicie a coleta, dando um nome para a funcionalidade "Criar Classe" e clique em "Start" na ferramenta *Trace Mark Control*. Insira uma classe no diagrama na área destinada a modelagem, para isto clique no Ícone 1 - Figura 1, e logo depois clique na área destinada a modelagem. Clique em "End" para finalizar a coleta;

**Passo 2:** Realize a coleta da funcionalidade "Criar Comentário". Clique no ícone (ícone 2— Figura 1) que insere o objeto comentário e, logo depois, clique na área de modelagem. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Inserir texto - Comentário". Clique no objeto Comentário duas vezes e o marcador de texto irá aparecer. Insira um texto com uns 200 caracteres (grande), e sem espaço. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Inserir quebra de linha no texto - Comentário". Clique no objeto Comentário duas vezes e o marcador de texto irá aparecer sobre o texto já inserido. Insira quebras de linha (clique em ENTER) no texto. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Alterar a largura do objeto Comentário". Clique no objeto Comentário, irá parecer pequenos quadrados nas bordas (seta 3 - Figura 1). Ao clicar nestes quadrados e arrastar será possível alterar o tamanho do comentário. Reduza o tamanho em Largura. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Alterar a altura do objeto Comentário". Clique no objeto Comentário, irá aparecer pequenos quadrados nas bordas. Ao clicar nestes quadrados e arrastar será possível alterar o tamanho do comentário. Aumente o tamanho em altura. Clique em "End" para finalizar a coleta;

**ESTE ROTEIRO É UMA SUGESTÃO, O ANALISTA PARTICIPANTE DESTE EXPERIMENTO PODE MONTAR OUTROS ROTEIROS SE ACHAR NECESSÁRIO PARA A COMPREENSÃO DO PROBLEMA.**

Figura B.31: Questionário do grupo Controle no experimento 4 - Melhoria - página 2.

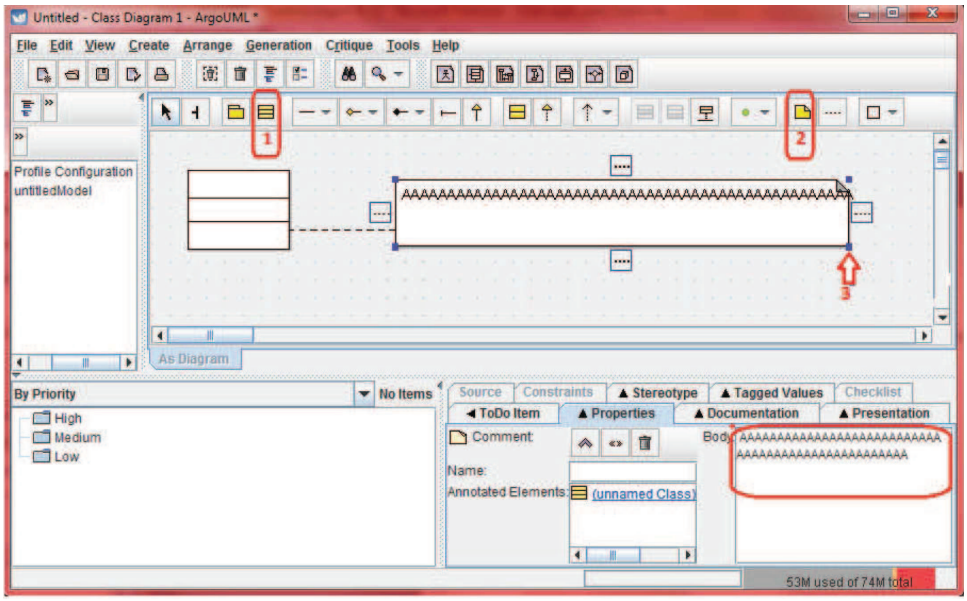



Figura 1— Ferramenta ArgoUML, exemplo de uso.

**Resultado:**

**Indique aqui as classes e métodos envolvidos na atividade:**

| Classes: | Métodos: |
|---|---|
| – | — |
| – | — |
| – | — |
| – | — |
| – | — |
| – | — |

**Qual foi a alteração realizada para esta melhoria no sistema?**

–
–
–
–
–
–
–
–
–
–
–

Figura B.32: Questionário do grupo Controle no experimento 4 - Melhoria - página 3.

**Perguntas:**
**Tempo para responder as perguntas:**

**Pergunta para todos os participantes**

- Vocês consideraram esta atividade de manutenção como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–
–
–

- Utilizou alguma funcionalidade do IDE Eclipse para auxiliar na atividade? Qual?
( ) SIM ( )Não

- Resuma o procedimento e ferramental utilizado para encontrar a solução do Erro:

- O IDE propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção?
( )SIM ( )NÃO

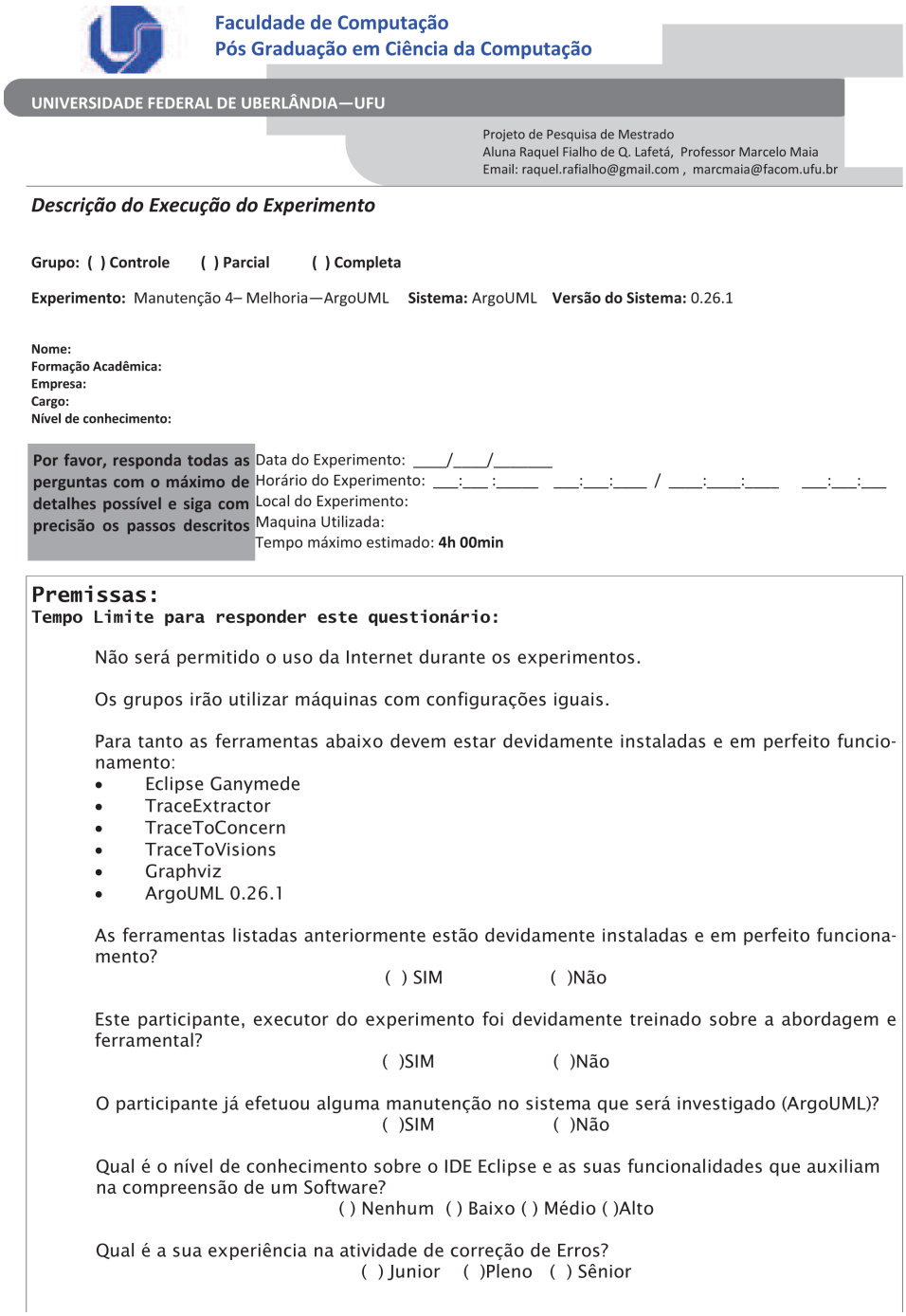
Figura B.33: Questionário do grupo Controle no experimento 4 - Melhoria - página 4.

- E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?

( )SIM ( )NÃO

*Obrigada pela sua colaboração!*

Figura B.34: Questionário do grupo Controle no experimento 4 - Melhoria - página 5.



Faculdade de Computação
Pós Graduação em Ciência da Computação

UNIVERSIDADE FEDERAL DE UBERLÂNDIA—UFU

Projeto de Pesquisa de Mestrado
Aluna Raquel Fialho de Q. Lafetá, Professor Marcelo Maia
Email: raquel.rafialho@gmail.com , marcmaia@facom.ufu.br

***Descrição do Execução do Experimento***

**Grupo: ( ) Controle ( ) Parcial ( ) Completa**

**Experimento:** Manutenção 4– Melhoria—ArgoUML **Sistema:** ArgoUML **Versão do Sistema:** 0.26.1

**Nome:**
**Formação Acadêmica:**
**Empresa:**
**Cargo:**
**Nível de conhecimento:**

**Por favor, responda todas as perguntas com o máximo de detalhes possível e siga com precisão os passos descritos**

Data do Experimento: ____/____/_____
Horário do Experimento: ___:___ :_____ ___:___:____ / ____:____:____ ___:___:___
Local do Experimento:
Maquina Utilizada:
Tempo máximo estimado: **4h 00min**

**Premissas:**
**Tempo Limite para responder este questionário:**

Não será permitido o uso da Internet durante os experimentos.

Os grupos irão utilizar máquinas com configurações iguais.

Para tanto as ferramentas abaixo devem estar devidamente instaladas e em perfeito funcionamento:

- Eclipse Ganymede
- TraceExtractor
- TraceToConcern
- TraceToVisions
- Graphviz
- ArgoUML 0.26.1

As ferramentas listadas anteriormente estão devidamente instaladas e em perfeito funcionamento?

( ) SIM ( )Não

Este participante, executor do experimento foi devidamente treinado sobre a abordagem e ferramental?

( )SIM ( )Não

O participante já efetuou alguma manutenção no sistema que será investigado (ArgoUML)?

( )SIM ( )Não

Qual é o nível de conhecimento sobre o IDE Eclipse e as suas funcionalidades que auxiliam na compreensão de um Software?

( ) Nenhum ( ) Baixo ( ) Médio ( )Alto

Qual é a sua experiência na atividade de correção de Erros?

( ) Junior ( )Pleno ( ) Sênior

Figura B.35: Questionário dos grupos Parcial e Completa no experimento 4 - Melhoria - página 1.

**Objetivo:**
Executar uma melhoria no sistema ArgoUML.

**Melhoria:**
O sistema ArgoUML é uma ferramenta para desenho de diagramas UML. Uma de suas funcionalidades é a inserção de comentários associados aos objetos do diagrama. Este objeto comentário sofre aumento em sua largura conforme se insere um texto neste. Se o texto não possui quebras de linha, o objeto comentário será tão largo quanto o texto inserido. E a ferramenta não permite que este objeto seja reduzido, enquanto não inserir quebras de linha no texto.

A melhoria solicitada é que o objeto Comentário tenha tamanho máximo fixo em altura e largura e que automaticamente se tenha uma quebra de linha no texto para que este se adapte ao tamanho do objeto. Se o tamanho do texto for maior que o tamanho máximo do objeto, que deve permitir apresentar até 160 caracteres, o restante do texto deve ficar oculto.

**Roteiro de execução sugerido:**

**Passo 1:** Inicie a coleta, dando um nome para a funcionalidade "Criar Classe" e clique em "Start" na ferramenta *Trace Mark Control.* Insira uma classe no diagrama na área destinada a modelagem, para isto clique no Ícone 1 - Figura 1, e logo depois clique na área destinada a modelagem. Clique em "End" para finalizar a coleta;

**Passo 2:** Realize a coleta da funcionalidade "Criar Comentário". Clique no ícone (ícone 2— Figura 1) que insere o objeto comentário e, logo depois, clique na área de modelagem. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Inserir texto - Comentário". Clique no objeto Comentário duas vezes e o marcador de texto irá aparecer. Insira um texto com uns 200 caracteres (grande), e sem espaço. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Inserir quebra de linha no texto - Comentário". Clique no objeto Comentário duas vezes e o marcador de texto irá aparecer sobre o texto já inserido. Insira quebras de linha (clique em ENTER) no texto. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Alterar a largura do objeto Comentário". Clique no objeto Comentário, irá parecer pequenos quadrados nas bordas (seta 3 - Figura 1). Ao clicar nestes quadrados e arrastar será possível alterar o tamanho do comentário. Reduza o tamanho em Largura. Clique em "End" para finalizar a coleta;

**Passo 3:** Realize a coleta da funcionalidade "Alterar a altura do objeto Comentário". Clique no objeto Comentário, irá aparecer pequenos quadrados nas bordas. Ao clicar nestes quadrados e arrastar será possível alterar o tamanho do comentário. Aumente o tamanho em altura. Clique em "End" para finalizar a coleta;

**ESTE ROTEIRO É UMA SUGESTÃO, O ANALISTA PARTICIPANTE DESTE EXPERIMENTO PODE MONTAR OUTROS ROTEIROS SE ACHAR NECESSÁRIO PARA A COMPREENSÃO DO PROBLEMA.**

Figura B.36: Questionário dos grupos Parcial e Completa no experimento 4 - Melhoria - página 2.

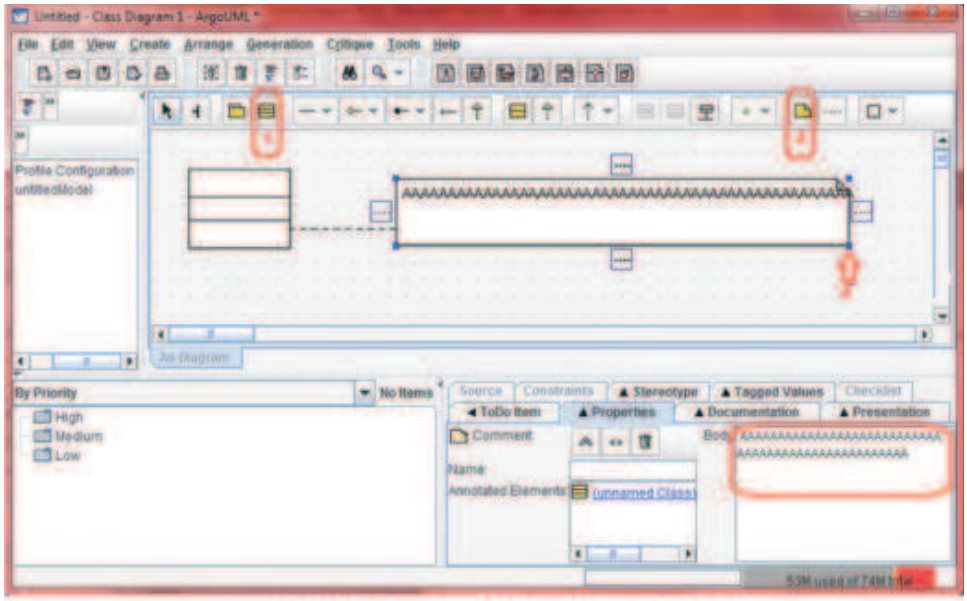



Figura 1— Ferramenta ArgoUML, exemplo de uso.

**Resultado:**

**Indique aqui as classes e métodos envolvidos na atividade:**

**Classes:** **Métodos:**

**Qual foi a alteração realizada para esta melhoria no sistema?**

Figura B.37: Questionário dos grupos Parcial e Completa no experimento 4 - Melhoria - página 3.

**Perguntas:**
**Tempo para responder as perguntas:**

**Pergunta para todos os participantes**

- Vocês consideraram esta atividade de manutenção como :
( )Complexa ( )Difícil ( ) Mediana ( ) Fácil ( )Muito Fácil

- Qual foi a maior dificuldade encontrada durante a execução da atividade?

–
–
–
–
–

- A abordagem o auxiliou na compreensão do sistema e em localizar onde deveria ser realizada a melhoria? Como?
( ) SIM ( )NÃO

–
–
–
–
–
–
–

- A abordagem (procedimentos e ferramental) listou todos os elementos (classes e métodos) que estão envolvidos na atividade? ( ) SIM ( ) NÃO . Se NÃO, quais?

–
–
–
–
–
–
–
–
–

- Quais visões da abordagem você utilizou/executou?

( )Visão 1— ConcernMapper
( )Visão 2— Call Graph
( )Visão 3– Corte Funcional
( )Visão 4—Mapeamento com Classificação
( ) Nenhuma das anteriores

Figura B.38: Questionário dos grupos Parcial e Completa no experimento 4 - Melhoria - página 4.

- Quais visões o auxiliou realmente nesta atividade?

( )Visão 1— ConcernMapper
( )Visão 2— Call Graph
( )Visão 3– Corte Funcional
( )Visão 4– Mapeamento com Classificação
( ) Nenhuma das anteriores

- Como estas visões o auxiliaram na atividade de manutenção?

–
–
–
–
–

- Como você define o grau de usabilidade da abordagem (ferramental)?

( )Excelente
( )Boa
( )Regular
( )Péssima

- A granularidade das visões foi suficiente para auxiliar na descrição da característica no propósito de auxiliar em uma manutenção? Justifique.

( ) SIM ( )NÃO

- A abordagem propicia um bom suporte para documentar a lista de métodos que precisam ser alterados na manutenção? Justifique.

( )SIM ( )NÃO

- E, também, apresenta ferramental que propicia investigações adicionais e bom suporte para investigação do algoritmo?

( )SIM ( )NÃO

Figura B.39: Questionário dos grupos Parcial e Completa no experimento 4 - Melhoria - página 5.

- A Abordagem permite que o programador indique os requisitos que fundamentam as Unidades Computacionais (classes e métodos) diretamente do ambiente de codificação, para não haver interrupção no trabalho de programação.
  ( )SIM ( )NÃO

- Você utilizou alguma funcionalidade do Eclipse (não da abordagem) para esta atividade?
  ( ) SIM ( ) NÃO. Se SIM, qual?

*Obrigada pela sua colaboração!*

Figura B.40: Questionário dos grupos Parcial e Completa no experimento 4 - Melhoria - página 6.