UNIVERSIDADE FEDERAL DO PAMPA

Mário Alan de Oliveira Lima

Um Projeto de Interface para Ferramentas de Modelagem UML Baseadas em Eclipse

Alegrete
2017

Mário Alan de Oliveira Lima

Um Projeto de Interface para Ferramentas de Modelagem UML Baseadas em Eclipse

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Me. João Pablo Silva da Silva

Coorientador: Prof. Me. Jean Felipe Patikowski Cheiran

Alegrete
2017

Mário Alan de Oliveira Lima

Um Projeto de Interface para Ferramentas de Modelagem UML Baseada em Eclipse

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 29 de novembro de 2017
Banca examinadora:

Prof. Me. João Pablo Silva da Silva
Orientador
UNIPAMPA

Prof. Me. Jean Felipe Patikowski Cheiran
Coorientador
UNIPAMPA

Prof. Dr. Amanda Meincke Melo
UNIPAMPA

Prof. Dr. Gilleanes Thorwald Araujo Guedes
UNIPAMPA

Dedico este trabalho em primeiro lugar a Deus que iluminou o meu caminho durante esta caminhada. À Sabrina, pessoa com quem amo compartilhar a vida. Com você tenho me sentido mais vivo de verdade. Obrigado pelo carinho e paciência nas horas que tinha que me dedicar aos estudos.

Ao meu filho Mário Henrique, maior presente que o Pai Celestial me deu, meus pais, irmã e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta estapa da minha vida.

AGRADECIMENTOS

Agradeço a Deus por ter me dado saúde e força para superar as dificuldades. Obrigado a minha esposa e filho, que nos momentos de minha ausência dedicados ao estudo superior, sempre entederam que o futuro é feito a partir da constante dedicação no presente. Aos meus pais que sempre foram exemplo de dignidade e que foram muito importante para a minha formação. A todos os professores que me acompanharam durante a graduação, em especial ao Prof. João Pablo e ao Prof. Jean Cheiran, responsáveis pela realização deste trabalho. Meus agradecimentos aos amigos e a todos que direta ou indiretamente fizeram parte da minha formação.

“Ele fortalece o cansado
e dá grande vigor ao que está sem forças.
Até os jovens se cansam
e ficam exaustos,
e os moços tropeçam e caem;
mas aqueles que esperam no Senhor
renovam as suas forças.
Voam alto como águias;
correm e não ficam exaustos,
andam e não se cansam.
(Bíblia Sagrada, Isaías 40:29-31)

RESUMO

Evidências mostram que as ferramentas *Computer-Aided Software Engineering* (CASE) são muitas vezes mais orientadas para funcionalidades do que para facilidade de uso, sendo difíceis de usar, aprender e dominar. Este trabalho descreve alguns problemas de usabilidade nas ferramentas CASE de modelagem, que utilizam a Linguagem de Modelagem Unificada (UML), baseadas na plataforma Eclipse, e relata a prototipação e a avaliação de um projeto de interface de usuário para melhoria de usabilidade nessas ferramentas. O objetivo deste trabalho é desenvolver um protótipo para ferramentas CASE de modelagem UML baseadas na plataforma Eclipse, utilizando estratégias que melhorem a usabilidade. Uma pesquisa na literatura especializada foi utilizada como método de busca para identificar problemas de usabilidade que os desenvolvedores enfrentam no uso das ferramentas CASE de modelagem UML. Os resultados desta pesquisa sugerem vários problemas em interfaces de usuários, como a representação de funcionalidades e a complexidade das interfaces de usuário. Para desenvolver o protótipo, foram utilizadas as técnicas de *card sorting*, prototipagem e teste de usabilidade com usuários. A partir da execução do *card sorting*, com sete alunos do curso de graduação em Engenharia de Software da UNIPAMPA, obteve-se a arquitetura do primeiro protótipo. Com a arquitetura resultante do *card sorting*, foi prototipada uma interface de usuário para ferramentas CASE de modelagem UML, utilizando a técnica de prototipação em papel e a ferramenta Axure para a criação de um protótipo funcional. O protótipo foi avaliado por alunos dos cursos de Engenharia de Software e Ciência da Computação. Esta avaliação foi um teste de usabilidade em que os avaliadores executaram algumas tarefas. A partir do teste de usabilidade foi possível analisar a interação dos avaliadores com o protótipo. Os resultados do teste de usabilidade demonstram que a falta de conhecimento sobre os ícones de uma ferramenta dificultam muito a sua utilização. Houve também uma divergência entre os avaliadores sobre em qual posição um menu deveria estar posicionado e nenhum dos avaliadores conseguiu executar a primeira tarefa que envolvia um novo método de modelar. A prototipação de interfaces de usuário melhora alguns problemas de usabilidade e ajuda a criar ambientes de desenvolvimento de software com mais usabilidade. O resultado deste trabalho é um protótipo de interface de usuário para ferramentas CASE de modelagem UML prototipado e avaliado por alunos da área da computação, que pode servir de orientação para o desenvolvimento de novas ferramentas de modelagem.

Palavras-chave: Estratégias de Usabilidade. Interface de Usuário. Ferramentas Case.

ABSTRACT

Evidence shows that Computer-Aided Software Engineering (CASE) tools are often more feature-oriented than easy-to-use, being difficult for using, learning, and mastering. This paper describes some usability issues in modeling CASE tools that use the Unified Modeling Language (UML), based on the Eclipse platform, and reports the prototyping and evaluation of a user interface design to improve usability in these tools. The objective of this work is to develop a prototype for CASE tools for UML modeling based on the Eclipse platform, using strategies that improve usability. A research in the specialized literature was used as a search method to identify usability problems that developers face in using the CASE tools for UML modeling. The results of this research suggest several problems in user interfaces, such as the representation of functionalities and the complexity of user interfaces. To develop the prototype, the techniques of textit card sorting, prototyping and usability testing with users were used. From the execution of the card sorting, with seven students of the undergraduate program in Software Engineering from UNIPAMPA, the architecture of the first prototype was obtained. With the resulting card sorting architecture, a user interface for UML modeling CASE tools was prototyped, using the paper prototyping technique and the Axure tool to create a functional prototype. The prototype was evaluated by students of the Software Engineering and Computer Science. This program evaluation was a usability test in which the evaluators performed some tasks. From the usability test it was possible to analyze the interaction of the evaluators with the prototype. The results of the usability test demonstrate that the lack of knowledge about the icons of a tool makes their use very difficult. There was also a divergence among evaluators about where a menu should be positioned and none of the evaluators could perform the first task involving a new modeling method. Prototyping user interfaces improves some usability issues and helps create more usable software development environments. The result of this work is a user interface prototype for CASE tools for UML modeling prototyped and evaluated by students in the field of computing, which can serve as a guide for the development of new modeling tools.

Key-words: Usability Strategies. User Interfaces. Case Tools.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AUI - *Adaptative User Interface*
CASE - *Computer-Aided Software Engineering*
ES - Engenharia de *Software*
IDE - *Integrated Development Environment*
IEEE - *Institute of Electrical and Electronics Engineers*
IHC - Interação Humano-Computador
IU - Interface de Usuário
QuIDE - *Quantum Integrated Development Environment*
RBUIS - *Role Based User Interface Simplification*
SWEBOK - *Software Engineering Body of Knowledge*
UML - *Unified Modeling Language*
UNIPAMPA - Universidade Federal do Pampa

SUMÁRIO

1 INTRODUÇÃO

A usabilidade é um conceito diretamente relacionado à computação, pois, quando um software é desenvolvido, na maioria das vezes seus usuários são um público heterogêneo. Ela é a área do conhecimento relacionada à simplicidade e intuitividade de um software, uma página web ou uma ferramenta computacional (BARBOSA; SILVA, 2010).

Quanto maior for a usabilidade durante a interação do usuário com o produto, mais pessoas podem utilizá-lo com facilidade. A qualidade de um produto está diretamente relacionada ao nível de preocupação com a usabilidade e implementação de um design centrado no usuário no projeto da interface de usuário. Para garantir uma boa usabilidade, existem diversas técnicas de interação humano-computador (IHC) que podem ser utilizadas, como a análise e modelagem de usuários, a análise e modelagem de tarefas (cenários / modelo de tarefas), a modelagem de comunicação e o *storyboarding* (CYBIS; BETIOL; FAUST, 2007).

De forma geral, a usabilidade é muito discutida em relação ao usuário comum dos meios computacionais, isto é, aos usuários não especialistas em computação, mas em relação às ferramentas utilizadas pelos desenvolvedores de software, a usabilidade ainda necessita ser melhor explorada (DILLON; THOMPSON, 2016). É possível encontrar problemas de usabilidade em diversas ferramentas CASE, e existe uma dificuldade de aprendizado, principalmente, por programadores iniciantes que, ao se depararem com a quantidade de menus e opções à sua disposição ficam, confusos (ZOU et al., 2008).

A IHC e a Engenharia de Software (ES), diferente do que deveria acontecer, muitas vezes têm uma convivência difícil, que remonta aos primeiros dias da IHC, quando o design centrado no usuário foi apresentado como o oposto, e às vezes como uma substituição, para a filosofia orientada pelo sistema geralmente utilizada na engenharia de software (NORMAN; DRAPER, 1986).

Embora numerosos pesquisadores e profissionais da IHC vejam o Design Centrado no Usuário como um processo e como um conjunto de metodologias específicas para projetar e desenvolver aplicativos de software interativos, a IHC não é considerada um tópico central na ES. Por exemplo, o Corpo de Conhecimento da Engenharia de Software (SWEBOK)[1], uma iniciativa do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) para a definição de conhecimento e práticas da ES, define a IHC como uma "disciplina relacionada", denominada "ergonomia de software" (SEFFAH; GULLIKSEN; DESMARAIS, 2006).

A usabilidade é considerada uma das muitas exigências não funcionais e atributos de qualidade. A usabilidade é essencial no processo de desenvolvimento de sistemas interativos, onde objetiva-se desenvolver softwares que tornem produtiva e satisfatória a realização de tarefas pelos usuários.

[1] https://www.computer.org/web/swebok

1.1 Motivação

Ao realizar a pesquisa sobre trabalhos relacionados, foram encontrados diversos problemas de usabilidade, como interface de usuário muito complexa (AKIKI; BANDARA; YU, 2016) e dificuldade de dominar a grande quantidade de recursos dos IDEs (RYCERZ et al., 2015). Em ferramentas baseadas na plataforma Eclipse, é muito comum encontrar usuários com problemas relacionados à usabilidade, no sentido de não saberem nortear-se sobre onde salvar determinadas informações, ou até mesmo configurar preferências de interface[2].

Outros problemas encontrados são a negligência na concepção, no desenvolvimento de interfaces de software (ZOU et al., 2008) e a utilização de uma abordagem que se preocupa mais com as funcionalidades do que com a interface com usuário (DILLON; THOMPSON, 2016). A ferramenta de modelagem Papyrus não é nada intuitiva para uso. Ao executar um tutorial sobre a ferramenta não se conseguiu descobrir como utilizar muitas das funcionalidades como visualizar campos dentro do diagrama, gerar código a partir de um diagrama UML, ou mesmo exportar um diagrama[3].

A experiência do usuário com a plataforma Eclipse é muito importante ao utilizar ferramentas de modelagem desenvolvidas nessa plataforma, como o Papyrus ou o Modelio, pois, devido à abundância de opções em menus e barras de ferramentas suas interfaces de usuários são bastante complexas. Para usuários familiarizados com o Eclipse, as coisas parecem bastante naturais. Como IDE, o Eclipse é fácil de aprender. Mas, uma vez que se trata de modelagem, as coisas não são claramente tão intuitivas quanto deveriam (FINSTAD, 2010)[4].

Diante do exposto, é possível melhorar a usabilidade do Eclipse e de *plug-ins* desenvolvidos para ele? Que estratégias poderiam ser adotadas? Como mitigar esses problemas de usabilidade, aumentando a satifação dos desenvolvedores ao executar suas atividades de desenvolvimento com o auxílio do Eclipse?

1.2 Objetivo

Este trabalho tem por objetivo geral criar um protótipo de uma ferramenta CASE de modelagem UML, com um bom nível de usabilidade, baseado nas qualidades das ferramentas de modelagem desenvolvidas na plataforma Eclipse e em ferramentas de modelagem proprietárias. Para atingir tal objetivo, este trabalho busca descobrir quais estratégias precisam ser executadas durante o desenvolvimento de ferramentas CASE, para que seus usuários, os desenvolvedores de software, possam trabalhar utilizando ferramentas com menos problemas de usabilidade. Para isso, os objetivos específicos abaixo deverão

[2] https://eclipse.org/
[3] http://www.eclipse.org/papyrus/index.php
[4] https://www.modelio.org/

ser atingidos:

- Investigar sobre o estado da arte quanto às estratégias necessárias para garantir um bom nível de usabilidade em ferramentas CASE;
- Projetar e prototipar uma versão de interface gráfica com um bom nível de usabilidade;
- Consolidar o protótipo.

1.3 Metodologia

Este trabalho tem como tipo de pesquisa, a pesquisa explicativa, buscando demonstrar a importância e os resultados da utilização das estratégias de melhoria de usabilidade em interfaces de usuário (CERVO; SILVA, 2006).

Para execução deste trabalho, foi criada uma estrutura analítica de trabalho, que é um processo de divisão do trabalho do projeto em partes menores e mais facilmente gerenciáveis. O objetivo geral foi dividido em três objetivos específicos: investigação sobre o estado da arte; projeto e prototipação; e avaliação e consolidação. Cada um dos objetivos específicos foi dividido em tarefas.

Na etapa de investigação sobre o estado da arte, foi realizado o planejamento da pesquisa, a pesquisa propriarmente dita e a consolidação dos resultados. Durante a etapa de projeto e prototipação foi planejada a prototipação, definidas as funcionalidades e características do protótipo, foram realizados dois *card sorting* para definir a arquitetura do protótipo e desenvolvido o protótipo. Por fim, na etapa de avaliação e consolidação foi planejada a avaliação, os protótipos avaliados e os resultados consolidados.

1.4 Contribuições

Este trabalho apresenta estratégias para a melhoria da usabilidade de ferramentas CASE de modelagem UML, permitindo que ferramentas com melhor usabilidade possam ser desenvolvidas e apresenta um protótipo de interface de usuário, avaliado por alunos de cursos da computação, que pode ser implementado no desenvolvimento de novas ferramentas. Outra contribuição, que não estava no planejamento, foi a visão comportamental desenvolvida no protótipo e a modelagem comportamental, algo que não foi visualizado em nenhuma das ferramentas de modelagem analisadas.

1.5 Organização deste trabalho

Este trabalho apresenta estratégias que melhoram a usabilidade em ferramentas CASE e um protótipo de interface para ferramentas CASE de modelagem UML e está

organizado como segue. No Capítulo 2 é apresentada uma fundamentação teórica a respeito dos conceitos básicos de usabilidade. No Capítulo 3 o assunto é desenvolvido com a apresentação dos artigos relacionados. No Capítulo 4 são apresentadas análise de ferramentas de modelagem, o projeto e o desenvolvimento do protótipo. No Capítulo 5 são apresentados a avaliação do protótipo, o resultado dessa avaliação e as contribuições deste trabalho para a melhoria da usabilidade das ferramentas CASE de modelagem. E, por fim, as considerações finais encerram o trabalho no Capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICO-METODOLÓGICA

Neste Capítulo é apresentada a fundamentação teórico-metodológica que serve como base introdutória aos conceitos que são utilizados durante o desenvolvimento do trabalho aqui apresentado. Na seção 2.1 é abordada a interação humano-computador, em seguida, na seção 2.2 a usabilidade é apresentada. Na seção 2.3 a prototipação é abordada. Na seção 2.4 é apresentada a técnica de *card sorting*, a qual foi utilizada para a elaboração de um dos protótipos. Na seção 2.5 é abordado o teste de usabilidade com usuários, técnica que foi utilizada para avaliar o protótipo apresentado por este trabalho. Na seção 2.6 é apresentada a definição do que é uma ferramenta CASE. Por fim, na seção 2.7 são apresentadas as lições do Capítulo.

2.1 Interação Humano-Computador (IHC)

Segundo Hewett et al. (1992), a IHC é a área do conhecimento interessada no projeto, implementação e avaliação de sistemas computacionais interativos para uso do homem, simultaneamente com os fenômenos associados a esse uso.

Barbosa e Silva (2010) concordam com as afirmações de Hewett, e afirmam ainda que a engenharia de software está preocupada com o desenvolvimento de sistemas interativos mais eficientes, robustos, sem erros e com uma manutenção simples. No entanto, a área de IHC está preocupada com a qualidade de uso desses sistemas e na sua influência na vida dos seus usuários.

2.2 Usabilidade

Dentro da IHC, a usabilidade aborda a forma como o usuário se comunica com a máquina e como a tecnologia responde à interação do usuário (CYBIS; BETIOL; FAUST, 2007).

Na norma que define os critérios de qualidade, ISO/IEC 9126 (1991), a usabilidade é definida como sendo um conjunto de atributos relacionados com o esforço necessário para o uso de um software, e relacionados com a avaliação individual de tal uso, por um conjunto específico de usuários.

Especificamente no Brasil existe a NBR ISO/IEC 9126-1 (2001) que define a usabilidade como a capacidade do software de ser compreendido, aprendido, operado e atraente ao usuário, quando usado sob condições especificadas.

Sommerville (2011) decreve a usabilidade como uma propriedade que reflete a facilidade ao se usar um sistema. Ela depende do software, seus operadores e seu ambiente operacional. Ela é a qualidade que caracteriza o uso dos softwares.

Cybis, Betiol e Faust (2007) afirmam que a usabilidade não é uma qualidade inerente de um sistema, no entanto necessita de compatibilidade entre as características de sua interface e as características de seus usuários ao buscarem determinados fins em

determinadas circunstâncias de uso. Uma mesma interface pode propiciar interações que satisfaçam usuários experientes e deixe muito a desejar quando utilizada por novatos.

A usabilidade possui diversos métodos de design de interfaces como alocação de funções, design paralelo, ergonomia física, design participativo, entre outros. São abordados dois, sendo eles a prototipação e o *card sorting*[1]. A usabilidade pode ser medida através de quatro características: eficiência em uso, eficácia, tolerância a erros e facilidade de uso.

2.3 Prototipação

A indústria de software tem duas preocupações: tempo e dinheiro. Antes de investir no desenvolvimento de um produto é preciso certificar-se que é viável, o que ressalta a importância do protótipo, que é uma versão preliminar de um produto que permite que sejam exploradas suas ideias e apresente o conceito geral de design para usuários. O protótipo pode ser qualquer modelo, desde desenhos em papel (de baixa fidelidade) até algo que permite o clique de alguns pedaços de conteúdo em um site totalmente funcional (de alta fidelidade)[2].

Protótipos são representações de design de interface e são classificados quanto ao seu grau de funcionalidade embutido. Desde *wireframes* ou maquetes, que permitem a visualização dos signos estáticos e metalinguísticos da interface, em múltiplos níveis de detalhe, até protótipos funcionais, que abrangem também os signos dinâmicos (BARBOSA; SILVA, 2010).

A prototipação desenvolve um design inicial de uma interface ou produto, usado para capturar conceitos e leiaute iniciais, para obter feedback dos usuários, participantes do projeto e partes interessadas.

A prototipagem pode ser utilizada durante as várias fases do processo de design, em diferentes níveis de fidelidade, dos esboços em papel aos protótipos funcionais mais detalhados, quase no nível de detalhamento de uma interface final.

2.4 *Card Sorting*

O método de classificação de cartões é usado para obter informações sobre as associações e o agrupamento de itens de dados específicos. Os participantes são convidados a organizar cartões com itens individuais e não classificados em grupos e podem, dependendo da técnica, fornecer também rótulos para esses grupos[3].

Em um processo de design centrado no usuário, ele é usado para desenvolver uma arquitetura de páginas web, fluxos de trabalho, menus, barras de ferramentas e

[1] (http://www.usabilitybok.org, acessado em 01/06/2017)
[2] (http://www.usability.gov, acessado em 01/06/2017)
[3] (http://www.usabilitybok.org, acessado em 01/06/2017)

outros elementos do projeto do sistema. Segundo Spencer e Warfel (2004), sete a dez participantes consistem em uma amostra significativa e ideal, quando o *card sorting* é praticado individualmente.

O *card sorting* pode ser do tipo "aberto" ou "fechado". No *card sorting* aberto, segundo Spencer e Warfel (2004), os participantes recebem cartas contendo as funcionalidades da aplicação sem agrupamento pré-estabelecido. Eles, então, classificam os cartões em grupos da forma como julgam ser apropriadas e, por fim, nomeiam os agrupamentos. Seu uso é recomendado para novas aplicações ou melhorias em projetos já existentes.

No *card sorting* fechado, da mesma maneira que no aberto, os participantes recebem cartas contendo as funcionalidades da aplicação. A diferença está no fato de receberem um conjunto pré-estabelecido de grupos. Eles, então, organizam suas cartas dentre os grupos oferecidos. Dependendo da flexibilidade da pesquisa, podem ser aceitas alterações das nomenclaturas dos grupos ou mesmo sugestões de inserção de novos agrupamentos por parte dos participantes. Seu uso é recomendado quando é necessário acrescentar novos conteúdos em uma arquitetura já existente ou para obter um feedback após definida uma arquitetura através do *card sorting* aberto (SPENCER; WARFEL, 2004).

2.5 Teste de Usabilidade

Cybis, Betiol e Faust (2007) afirmam que o teste de usabilidade envolve a observação de usuários ao executar tarefas com um sistema de software. O artefato a ser avaliado pode ser um protótipo de papel, um *wireframe*, um storyboard, um produto em desenvolvimento, um protótipo ou um produto completo. Os testes de usabilidade também podem ser realizados em produtos competitivos para entender seus pontos fortes e fracos.

O teste de usabilidade pode ser uma avaliação formativa, que é realizada no início do processo de projeto para encontrar problemas para melhorar o produto, ou avaliação somativa, para validar o projeto contra metas específicas (CYBIS; BETIOL; FAUST, 2007).

Segundo Nielsen e Mack (1994), o teste envolve o recrutamento de usuários direcionados como participantes do teste e solicita a esses usuários que completem um conjunto de tarefas. Um facilitador de teste conduz o teste através de um protocolo de teste, enquanto as sessões de teste são normalmente gravadas por um operador de vídeo ou uma ferramenta de teste automatizada.

Os testes de usabilidade devem ser realizados com participantes representativos dos usuários reais ou potenciais do sistema. Para alguns testes, os usuários devem possuir certos conhecimentos e experiência específicos de domínio, produto e aplicação. O teste de usabilidade consiste em cinco fases principais: planejamento, teste piloto, sessões de

teste, pós-teste e análise, interpretação e apresentação dos resultados[4].

2.6 Ferramentas CASE

Nos anos 80 e 90, havia um ponto de vista generalizado de que a melhor maneira para desenvolver o melhor software era por meio de um cuidadoso planejamento do projeto, qualidade da elicitação de requisitos, do uso de métodos de análise e projeto apoiado por ferramentas CASE (SOMMERVILLE, 2011).

A palavra CASE é uma abreviatura de *Computer Aided Software Engineering*, essa abreviatura classifica todas as ferramentas baseadas em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes (ISO/IEC 14102, 2008).

A engenharia de software é integrada com essas ferramentas auxiliadas por computador que abrangem múltiplas fases do ciclo de vida do desenvolvimento de software. Tais ferramentas desempenham múltiplas funções e, portanto, interagem potencialmente com o ciclo de vida a ser executado pelo processo de software (ISO/IEC 14102, 2008).

Dentre os vários tipos de ferramentas CASE de modelagem UML, as análisadas neste trabalho para desenvolvimento do protótipo foram os softwares livres Modelio[5], Papyrus[6] e UML Designer[7] e os softwares proprietários Astah Professional[8], Enterprise Architect[9] e IBM Rational Rose[10].

2.7 Lições do Capítulo

Foi apresentado neste Capítulo alguns conceitos necessários para o melhor entendimento do trabalho aqui apresentado. A IHC é uma disciplina interessada no desenvolvimento de sistemas computacionais interativos para uso humano.

Dentre as áreas da IHC, encontra-se a usabilidade que é o grau em que um produto é usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico (NBR 9241-11, 2002). Uma das técnicas da usabilidade, a prototipação, pode variar quanto ao grau de fidelidade e de funcionalidade.

O *card sorting* é um método de agrupamento de cartões para verificar como desenvolvedores visualizam a arquitetura de uma ferramenta CASE de modelagem. E para realizar a avaliação de usabilidade existe o teste de usabilidade com o usúario, que é uma técnica de pesquisa utilizada para avaliar um produto.

[4] (http://www.usabilitybok.org, acessado em 01/06/2017)
[5] https://www.modelio.org/
[6] https://www.eclipse.org/papyrus/
[7] www.umldesigner.org/
[8] astah.net/editions/professional
[9] https://www.sparxsystems.com.au/products/ea/index.html
[10] https://www.ibm.com/software/products/pt/rosemod

3 TRABALHOS RELACIONADOS

Neste Capítulo são apresentados os trabalhos que possuem alguma relação com o trabalho proposto nesse documento. Na seção 3.1 é apresentada a metodologia de pesquisa utilizada para seleção dos trabalhos relacionados. Na seção 3.2 são apresentados os resultados encontrados com a revisão da literatura. Por fim, na seção 3.3, é feita uma breve conclusão sobre o que foi discutido neste Capítulo.

3.1 Protocolo de Revisão

Para pesquisar pelos trabalhos relacionados, inicialmente se utilizou uma metodologia de revisão sistemática, que consiste da delimitação do problema através de uma ou mais perguntas, neste caso, as perguntas definidas foram **"quais são as estratégias que uma ferramenta CASE necessita seguir para melhorar a sua usabilidade para desenvolvedores?"** e **"como essas estratégias foram avaliadas?"**.

Após isso, definiu-se a *scopus* como ferramenta de pesquisa em bases de dados. Definiu-se as seguintes *strings* para a pesquisa: *usability, feature, integrated development enviroment*. Para selecionar os resultados, os seguintes critérios foram utilizados: tamanho mínimo de 04 páginas e propor características de usabilidade para ferramentas CASE.

Foram utilizadas somente palavras chaves em inglês, pois existem poucos trabalhos sobre o assunto em português. Foi utilizado como processo de busca o *Snowballing*, que é uma metodologia de pesquisa que realiza uma busca inicial e a partir dos artigos que passam pelos critérios de inclusão, aplica-se uma busca nos artigos que foram citados ou citam aqueles já aprovados, diversas iterações podem ser feitas, encerrando quando nada mais passar pelos critérios de inclusão.

Após a pesquisa foram encontrados 22 trabalhos relacionados, dos quais foram selecionados 07 que atendiam todos os critérios, aplicou-se *Snowballing* e a partir desses 07 artigos nada mais foi aprovado pelos critérios de inclusão. Então foi tomada a decisão de não aplicar essa metodologia, em substituição a mesma o processo de busca foi alterado para uma *DataBase Search (DB Search)*.

Utilizou-se então as *strings "Usability AND Feature AND (IDE OR Integrated Development Environment)"* e *"Usability AND (IDE OR Integrated Development Environment)"* para realizar a pesquisa. Com a primeira *strings* se encontrou 07 artigos e com a segunda 03. Foi realizada a união dessas duas buscas e aplicados os critérios de exclusão, sendo 01 artigo excluído por ser uma publicação repetida.

3.2 Resultado da Revisão

A Tabela 1 apresenta os artigos selecionados, dos quais foram extraídas as seguintes informações: contexto, objetivo, metodologia e resultados.

Tabela 1 – Artigos selecionados.

Título do Artigo	**Autores**	**Ano**
Adapting the user interface of Integrated Development Environments (IDEs) for novice users	Zou, Y.; Lerner, M.; Leung, A.; Morisson, S. e Wringe, M.	2008
An empirical analysis of the evolution of user-visible features in an integrated development environment	Hou, D. e Wang, Y.	2009
Engineering Adaptive Model-Driven User Interfaces	Akiki, P.; Bandara, A. e Yu, Y.	2016
Software development and tool usability	Dillon, B. e Thompson, R.	2016
Teaching quantum computing with the QuIDE simulator	Rycerz, K.; Patrzyk, J.; Patrzyk, B. e Bubak, M.	2015
You can touch this: Touchifying an IDE	Biegel, B.; Hoffmann, J.; Lipinski, A. e Diehl, S.	2014
Usability of a domain-specific language for a gesture-driven IDE	Bacíkova, M.; Maricák, M. e Vancík, M.	2015

Fonte: elaborado pelo autor (2017)

Zou et al. (2008) declaram que a usabilidade de uma interface de usuário é muitas vezes negligenciada na concepção e no desenvolvimento de aplicações de software. Um IDE é propenso a problemas de usabilidade devido à quantidade de funcionalidades oferecida através de sua interface de usuário. Uma vez que um IDE tem como alvo uma ampla gama de usuários, os requisitos de usabilidade para um IDE varia consideravelmente.

Usuários novatos muitas vezes têm dificuldade em compreender muitas das características fornecidas em um IDE. Os autores propõem uma arquitetura de uma IU Adaptativa, *Adaptative User Interface* (AUI), que fornece uma interface simplificada para o IDE Eclipse.

Os autores desenvolveram algoritmos adaptativos, que modificam o sistema de menus existente para o Eclipse com base em padrões estatísticos de interação do usuário. Esses algoritmos adaptativos realizam uma análise custo-benefício ao modificar o sistema de menus. Os algoritmos determinam as mudanças ideais que reduzem o tempo necessário para os usuários principiantes ao pesquisar elementos do menu. Um protótipo AUI é desenvolvido como um *plug-in* do Eclipse para usuários principiantes desse IDE. Através de um estudo de caso inicial, foi demonstrado que os benefícios da AUI em melhorar a usabilidade do Eclipse.

Hou e Wang (2009) afirmam que os IDEs ajudam a aumentar a produtividade do programador ao automatizar muito trabalho administrativo (repetitivo). Os autores pesquisaram sobre as características de um IDE e como elas mudam e amadurecem.

Um estudo empírico foi realizado, analisando quantitativa e qualitativamente um total de 645 entradas de nota de lançamento do "*What's New*" em 7 releases do Eclipse. Os resultados mostram que a maioria das alterações são aperfeiçoamentos ou adições incrementais à arquitetura de recursos criada em versões anteriores. Motivado por isso, uma análise mais aprofundada sobre usabilidade é realizada para caracterizar como essas mudanças impactam os programadores e sua eficácia na utilização do IDE, para esta

análise os autores utilizam modelos de avaliação de usabilidade.

Akiki, Bandara e Yu (2016) nos mostram que aplicativos de software de grande escala podem apresentar centenas de interfaces de usuário complexas. Os autores buscam simplificar a interface de usuário baseada em funções como um mecanismo para aumentar a usabilidade.

O método de integração foi avaliado e medido através do estabelecimento e aplicação de métricas técnicas posteriormente, foi realizado um estudo de usabilidade para avaliar se as interfaces de usuário simplificadas mostram uma melhora em relação as suas contrapartidas iniciais. E os resultados mostraram que as interfaces de usuário simplificadas melhoram significativamente a eficiência, eficácia e a usabilidade.

Dillon e Thompson (2016) afirmam que as ferramentas CASE são utilizadas em todas as fases do ciclo de vida do software, com particular ênfase no período de manutenção. As evidências mostram que essas ferramentas são subutilizadas, mesmo pelos desenvolvedores que as criam. Uma delas, os ambientes de desenvolvimento integrados, foram criados para capacitar os desenvolvedores, mas eles permaneceram praticamente inalterados desde o final dos anos 90.

Os autores examinam os desafios da criação de ferramentas de desenvolvimento, analisam a usabilidade de duas ferramentas usadas com freqüência e sugerem que a má utilização da ferramenta pode estar inibindo o desenvolvimento de software mais eficiente. Para considerar a hipótese, eles comparam a usabilidade das ferramentas comumente encontradas em dois IDEs populares: Eclipse e Visual Studio.

Para realizar o experimento, eles selecionaram as funções “Adicionar Novo Método” e “Renomear”, por serem suportados por ambas IDEs e comumente executadas. E para avaliar, eles utilizaram os princípios de usabilidade de Nielsen(NIELSEN; MACK, 1994). Eles chegaram à conclusão que muitas ferramentas acabam caindo em desuso porque têm baixa usabilidade e é mais fácil para os desenvolvedores executarem as tarefas manualmente.

Rycerz et al. (2015) apresentam a ideia de que a computação quântica está cada vez mais popular, portanto, há uma necessidade de introduzir este tópico para estudantes de Ciência da Computação e Engenharia. Esse artigo apresenta o conceito de um curso impulsionado pelo *Quantum Integrated Development Environment* (QuIDE), o objetivo do curso é reunir aspectos teóricos com trabalhos práticos realizados no QuIDE.

O curso incluiu o entendimento do conceito de portões quânticos, registros e uma série de algoritmos e o QuIDE é avaliado com base na escala de usabilidade e comparado com outra ferramenta utilizada anteriormente para o mesmo fim. O resultado da avaliação do QuIDE mostra que as características mais utéis dessa ferramenta são semelhantes às suposições feitas na funcionalidade do simulador.

Biegel et al. (2014) afirmam que o *touch screen* é diversas vezes muito intuitivo, mas suas características de manipulação direta também ajudam a reduzir a carga cog-

nitiva. Como o desenvolvimento de software exige demandas cognitivas complexas, o objetivo deles era explorar as vantagens da manipulação direta para suportar processos de engenharia de software. Os autores demonstraram como o *touch screen* pode ser usados dentro de um ambiente de desenvolvimento integrado profissional.

Para isso, eles enriqueceram o Eclipse com opções multi-toques que podiam ser usadas tanto para controlar a interface gráfica do usuário como para acionar ferramentas de refatoração embutidas. Os primeiros resultados sugerem que o uso de um dispositivo de toque adicional dentro da configuração de desktop clássica permite um fluxo de trabalho preciso e rápido.

Bačíková, Maričák e Vančík (2015) declaram que as interfaces de usuário estão avançando em todas as direções. O uso de dispositivos de tela de toque e a adaptação de suas interfaces de usuário vive seu auge. No entanto, os ambientes de desenvolvimento integrados que são usados para desenvolver as mesmas interfaces de usuário estão parados no tempo.

O objetivo dos autores é projetar uma nova maneira de interação do usuário para IDEs comuns com a ajuda do toque. O grupo-alvo são dispositivos híbridos formados por um teclado físico e uma tela de toque integrada ou separada. Nesse artigo, os autores descrevem um conjunto de gestos de propósito geral e de domínio específico que representa uma linguagem para trabalhar com um IDE dirigido ao toque.

Foram realizados dois estudos com desenvolvedores da indústria e da universidade e desenvolvidos um protótipo de um IDE gestual para avaliar a usabilidade da abordagem apresentada. O resultado dessa abordagem foi o levantamento da necessidade de implementar *plug-ins* para suportar os gestos e integrar capacidades de aprendizagem que permitiriam ao sistema se adaptar ao utilizador e ao seu estilo de desenho e permitir ao utilizador adicionar novos gestos personalizados através do motor de modelo de código nativo da IDE.

3.2.1 Questão 1 - Quais são as estratégias que uma ferramenta CASE precisa seguir para melhorar a sua usabilidade para desenvolvedores?

Para que uma ferramenta CASE de modelagem UML apresente um bom nível de usabilidade, existem algumas estratégias que podem ser tomadas pelos desenvolvedores para garantir a usabilidade, nesta seção veremos algumas dessas estratégias. Alguns fatores relacionados ao usuário das ferramentas são muito relevantes e têm que ser levados em consideração durante o desenvolvimento das ferramentas, fatores como o idioma nativo, o nível de conhecimento de modelagem e o nível de conhecimento das regras da UML.

Zou et al. (2008) e Akiki, Bandara e Yu (2016) propõem interfaces adaptativas. Zou et al. (2008) propõem algoritmos adaptativos que modificam a IU, através de uma arquitetura *Adaptive User Interface*, que modificam os menus do Eclipse com base em padrões estatísticos de interação com o usuário.

Akiki, Bandara e Yu (2016) utilizam a arquitetura Cedar, arquitetura que é composta por três camadas do lado do servidor (componentes de decisão, componentes de adaptação e comportamento adaptativo e modelos de interface do usuário), que são acessados por meio de serviços da camada cliente-componentes e do Cedar Studio, para gerenciar a partir de um servidor a interação do usuário com a IDE, e de acordo com esta interação simplificar a IU, utilizando a *Role-Based User Interface Simplification* (RBUIS). A arquitetura pode ser utilizada também para simplificar a IU de sistemas legados.

A RBUIS compreende os seguintes elementos que possibilitam a otimização da IU: Modelos de IU baseados em funções que suportam minimização de conjunto de recursos atribuindo papéis a modelos de tarefas para fornecer um conjunto de recursos mínimo com base no contexto de uso. Esta abordagem permite uma realização prática do conceito de design de interface multicamada. Os modelos de comportamento adaptativo baseados em funções suportam a otimização da IU através de fluxos de trabalho que representam o comportamento adaptativo da IU visualmente e através do código.

A adaptação foi aplicada nos modelos de interface de usuário. Posteriormente, os modelos de comportamento adaptativo estão ligados a papéis para especificar como a IU foi otimizada para cada conjunto de usuários (AKIKI; BANDARA; YU, 2016). O feedback do usuário para refinamento permite ao usuário reverter otimizações de IU e escolher possíveis otimizações de IU alternativas. Mantendo os usuários envolvidos aumenta seu controle da IU e das características afetadas por mecanismos de adaptação / redução.

Hou e Wang (2009) propõem dois modelos para análise quantitativa e qualitativa da evolução do Eclipse. Usando o primeiro, um modelo funcional baseado em atividades, verifica-se que a grande maioria das mudanças são refinamentos ou adições incrementais à arquitetura de recursos criada em versões anteriores. Usando o segundo modelo, uma análise de usabilidade detalhada foi realizada para caracterizar mais estas mudanças em termos de seu impacto potencial na eficiência com que os programadores utilizam o IDE.

Dillon e Thompson (2016) discute sobre o desenvolvimento de software a partir do ponto de vista *bottom-up* e *top-down*, e propõem que o último ponto de vista permite construir softwares com maior usabilidade.

Rycerz et al. (2015) realizam uma comparação entre QuIDE e o LibQuantum, um IDE e uma library respectivamente, para computação quântica e propõem algumas característica para melhorá-los, sendo elas: suporte para diferentes sistemas operacionais, recursos de editor de código mais avançados, destaque de sintaxe, interface gráfica e comutação on-line entre a GUI e o código, passo a passo de execução, suporte para uma variedade de portões diferentes, suporte para a construção de portões complexos e portões de agrupamento, *drag* e *drop* no circuito de projeto, melhor documentação e mensagens de erro mais descritivas.

Biegel et al. (2014) e Bačíková, Maričák e Vančík (2015) propõem interfaces com

suporte ao toque. Biegel et al. (2014) comparam o suporte por toque normal do Eclipse e propõem um protótipo que pode ser usados tanto para controlar a interface gráfica do usuário como para acionar ferramentas de refatoração embutidas no IDE. E Bačíková, Maričák e Vančík (2015) apresentam um conjunto de gestos de domínio específico para IDEs orientadas ao toque.

3.2.2 Questão 2 - Como essas estratégias foram avaliadas?

Cinco dos artigos avaliados utilizaram usuários para a avaliação das suas estratégias de melhoria de usabilidade em interfaces, um artigo fez uma análise das modificações sofridas pelo Eclipse em várias atualizações e um artigo fez uma comparação da execução de algumas operações de refatoração em dois IDEs.

Para examinar a eficácia dos algoritmos adaptativos propostos, Zou et al. (2008) recrutaram cinco estudantes de pós-graduação para usar a IU adaptativa como seu ambiente de desenvolvimento de software. O sistema de menus foi monitorado enquanto os usuários estavam trabalhando no Eclipse. O protótipo coletou os dados de uso em segundo plano, enquanto os usuários interagiam com a interface do usuário. Também avaliaram a flexibilidade da implementação atual do sistema de menus do Eclipse com base em sua experiência no desenvolvimento do protótipo no estudo de caso.

Akiki, Bandara e Yu (2016) avaliaram a *Cedar*, o estudo envolveu 23 participantes, analisando 8 IUs (quatro originais e quatro simplificadas) com base em 3 critérios (eficiência, eficácia e usabilidade percebida), resultando em 552 amostras.

Segundo Rycerz et al. (2015), a usabilidade do QuIDE foi avaliada por vários alunos de um curso de graduação e comparada ao LibQuantum utilizando-se da Escala de Usabilidade do Sistema (SUS).

O protótipo de Biegel et al. (2014) foi avaliado por 8 pessoas e, para cada pessoa, foi realizada uma sessão de 45 minutos em que os participantes tiveram que resolver pequenos exercícios de usabilidade usando apenas um teclado e um monitor de toque em vez de um mouse.

Para avaliar os gestos, Bačíková, Maričák e Vančík (2015) criaram um protótipo de um IDE para *Android* de acordo com o projeto descrito nesse artigo e realizou uma avaliação de usabilidade. Esta foi executada por cinco participantes e visou à avaliação qualitativa.

O modelo de Hou e Wang (2009) foi aplicado em 645 entradas *"What's New"* em sete versões do Eclipse IDE, que foram analisadas quantitativa e qualitativamente. Descobriu-se que a usabilidade é o maior componente do trabalho. A partir das entradas uma análise detalhada da evolução da usabilidade é realizada e padrões de modificações foram discutidos.

Para considerar sua hipótese, Dillon e Thompson (2016) compararam a usabilidade das operações de refatoração: "renomear método" e "criar novo método", utilizando os

IDEs Eclipse e Visual Studio.

3.3 Lições do Capítulo

A estratégia de IU adaptativa pode ter benefícios em termos de melhorar a usabilidade, mas também pode haver elevados custo associado a sua implementação. Concorda-se que a abordagem *Top-Down* pode ajudar a desenvolver melhores IU, porque ela parte do pressuposto que o projeto da interface e sua posterior implementação é prioridade, e a partir dele as outras camadas do software são desenvolvidas.

Dentre as diversas formas de avaliação, a mais utilizada pelos trabalhos relacionados foi a avaliação com o usuário. Fato que ratifica a importância do usuário no processo de desenvolvimento das interface de usuário, para que estas IU obtenham uma boa aceitação perante o seu público-alvo, pois ferramentas que não são bem aceitas pelo seu público-alvo acabam caindo em desuso (DILLON; THOMPSON, 2016).

A partir dos artigos estudados pode-se concluir que o suporte ao toque é pequeno nas atuais IDEs e a melhoria de sua implementação pode criar novas técnicas de programação, facilitando a modelagem e a interação entre o desenvolvedor e a IDE.

4 PROJETO DE INTERFACE

Neste Capítulo, é apresentado o desenvolvimento do trabalho de melhoria da usabilidade das interfaces de usuário das ferramentas CASE de modelagem UML desenvolvidas na plataforma Eclipse. Este trabalho pode auxiliar desenvolvedores de ferramentas CASE de modelagem UML a decidirem que estratégias devem utilizar para garantir um bom nível de usabilidade em suas ferramentas. Na Tabela 2, as ferramentas CASE de modelagem UML são analisadas para extrair as suas qualidades e, posteriormente, construir o protótipo baseado nas funcionalidades das ferramentas analisadas. Na seção 4.2 o projeto e a prototipação das interfaces de usuário é abordado. Por fim, na seção 4.3, são apresentadas as lições do Capítulo.

4.1 Análise

Para desenvolver o protótipo foram analisadas seis ferramentas CASE de modelagem UML, sendo três softwares livres (Modelio, Papyrus e UML Designer) e três softwares proprietários (Astah Professional, Enterprise Architect e IBM Rational Rose). Para executar a análise foi utilizado o percurso cognitivo, que é um método de avaliação de usabilidade no qual um ou mais avaliadores trabalham através de uma série de tarefas e solicita um conjunto de perguntas na perspectiva do usuário (BARBOSA; SILVA, 2010).

O foco do percurso cognitivo é entender a capacidade de aprendizado do sistema para usuários novos[1]. As tarefas analisadas foram a construção de diagramas de classes, adição de classes ao diagrama e adição de propriedades à classe, a Tabela 2 resume a análise das ferramentas.

Tabela 2 – Análise das Ferramentas de Modelagem.

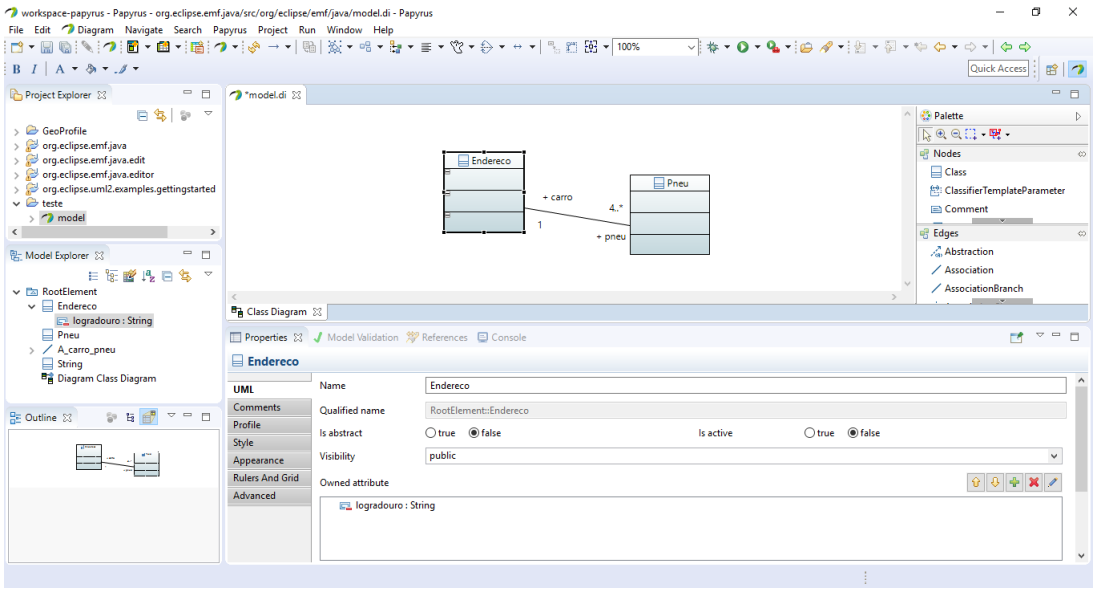
Ferramenta	**Tempo de execução**	**Informação desnecessária**	**Excesso de ícones**	**Facilidade de aprendizagem**
Papyrus	13 min	X	X	
Modelio	11 min	X	X	
UML Design	12 min	X	X	
IBM Rational Rose	9 min		X	
Enterprise Architect	7 min			X
Astah Professional	6 min			X

Fonte: elaborado pelo autor (2017)

O Papyrus (Figura 1) tem problemas de usabilidade. Para executar as tarefas foi necessário cerca de 13 minutos. O fato de compartilhar a interface do Eclipse faz com que várias opções de menu que não interessam para a modelagem atrapalhem a execução das tarefas.

[1] (http://www.usabilitybok.org, acessado em 01/06/2017)

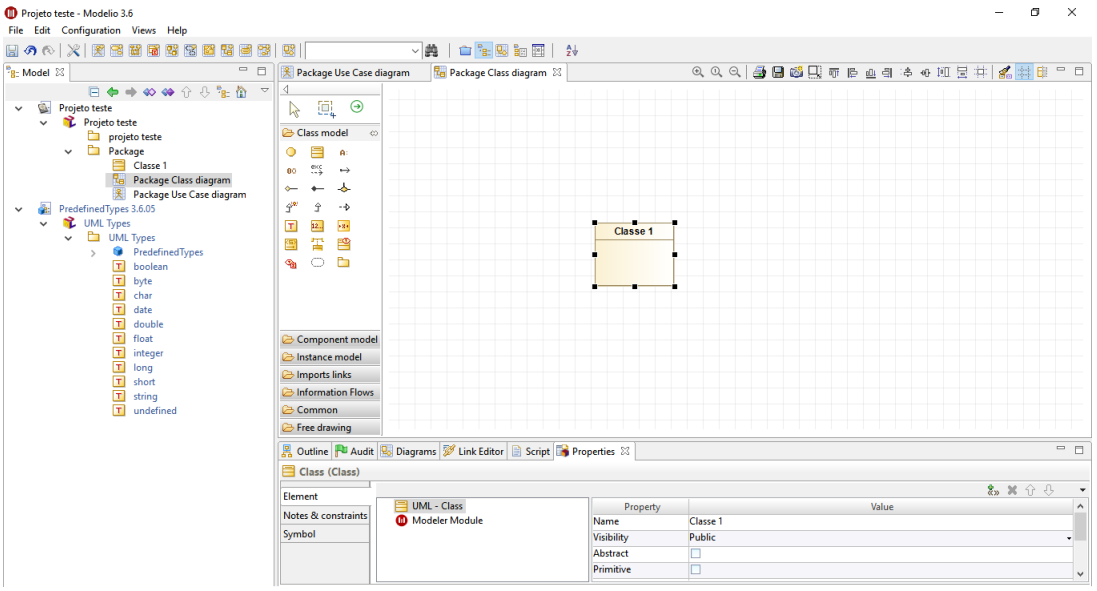
Figura 1 – Papyrus



Fonte: elaborado pelo autor (2017)

Por outro lado, o Modelio (Figura 2), tem uma usabilidade um pouco melhor que o Papyrus e produz diagramas que permitem uma boa visualização das informações. As propriedades não podem ser adicionadas tão rapidamente quanto com o Astah Professional, mas esta pequena diferença não chega a atrapalhar na execução das tarefas.

Figura 2 – Modelio

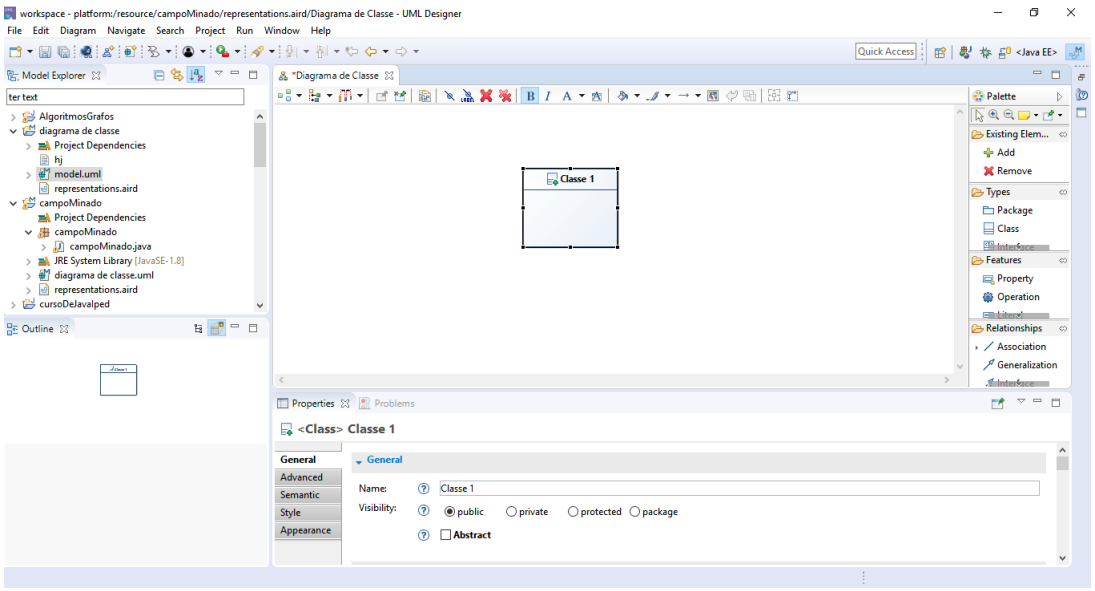


Fonte: elaborado pelo autor (2017)

Da mesma forma que as outras ferramentas de modelagem desenvolvidas na plataforma Eclipse, o UML Designer (Figura 3) peca no excesso de ícones na tela, segundo Miller (1956), os seres humanos têm uma capacidade limitada de processamento de in-

formações e o excesso acaba por atrapalhar. Outro problema foi a dificuldade inicial de acessar os menus de modelagem, são necessários seis cliques para criar um novo pacote, e após isso, ainda é exibida uma tela com sete abas, fazendo com que o usuário procure até descobrir que para modelar, ele vai precisar acessar a aba "Design".

Figura 3 – UML Designer



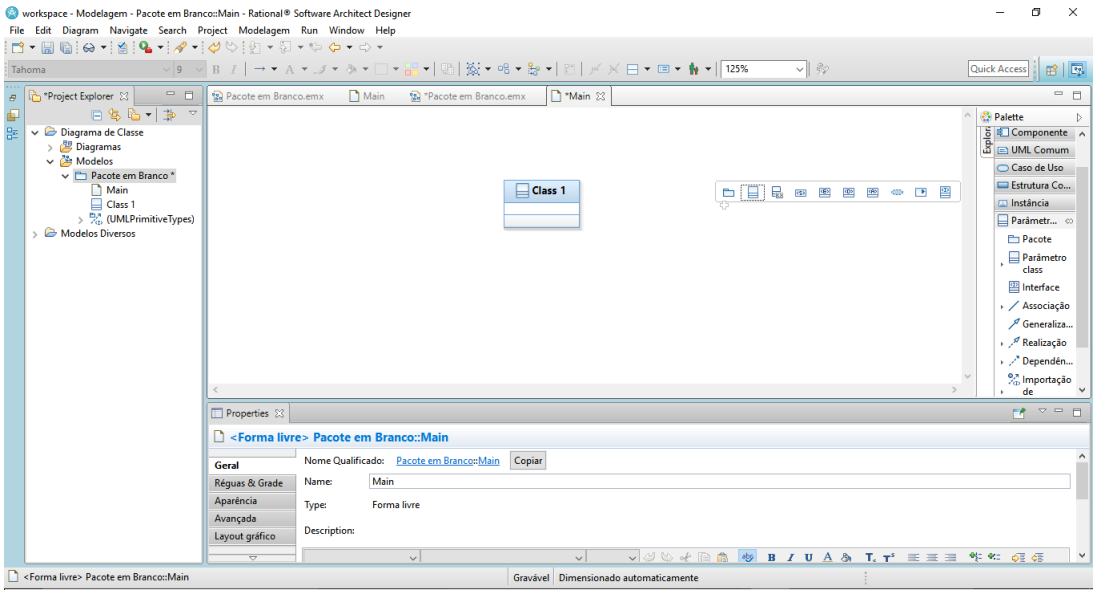
Fonte: elaborado pelo autor (2017)

O IBM Rational Rose (Figura 4) exige menos cliques que as ferramentas de modelagem desenvolvidas na plataforma Eclipse, com dois cliques é possível criar um diagrama de classes. Como os objetivos de design da Rational Rose têm de acomodar uma ampla gama de requisitos, a modelagem UML rápida e explorativa torna-se mais intuitiva e menos complexa.

De todas as seis ferramentas, a mais produtiva foi o Enterprise Architect (Figura 5), pois, tem uma interface simples e elementar, que permite criar um diagrama de classes com apenas dois cliques, ferramenta de fácil aprendizagem, que permite uma navegação entre os artefatos, facilitando a percepção das relações entre os diversos artefatos de modelagem. É uma ótima escolha como uma ferramenta de modelagem ágil que oferece um bom suporte para o desenvolvimento orientado a modelos, incluindo a geração automatizada de documentação e código.

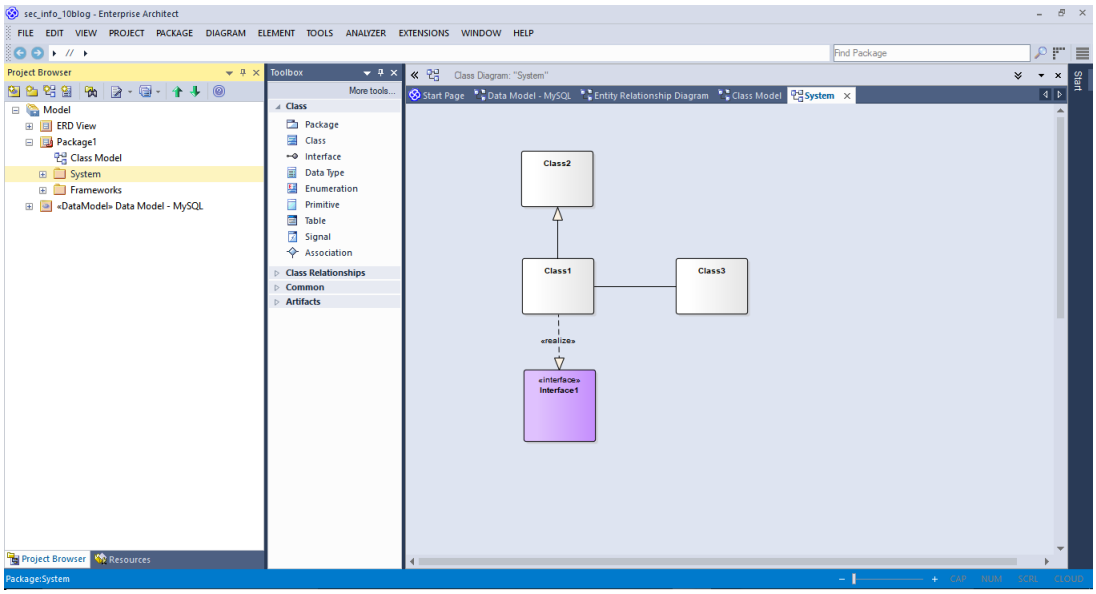
Uma das ferramentas de modelagem mais utilizadas pelos estudantes universitários é o Astah Professional (Figura 6), esta ferramenta tem uma interface simples e fácil de utilizar, mas a navegação entre os artefatos não é tão funcional quanto no Enterprise Architect, por exemplo, em um diagrama de sequência não é possível acessar informações sobre uma classe que esteja no diagrama, a única informação disponível é o nome da classe. Possui uma interface mais limpa do que as ferramentas desenvolvidas na plataforma Eclipse.

Figura 4 – IBM Rational Rose



Fonte: elaborado pelo autor (2017)

Figura 5 – Enterprise Architect

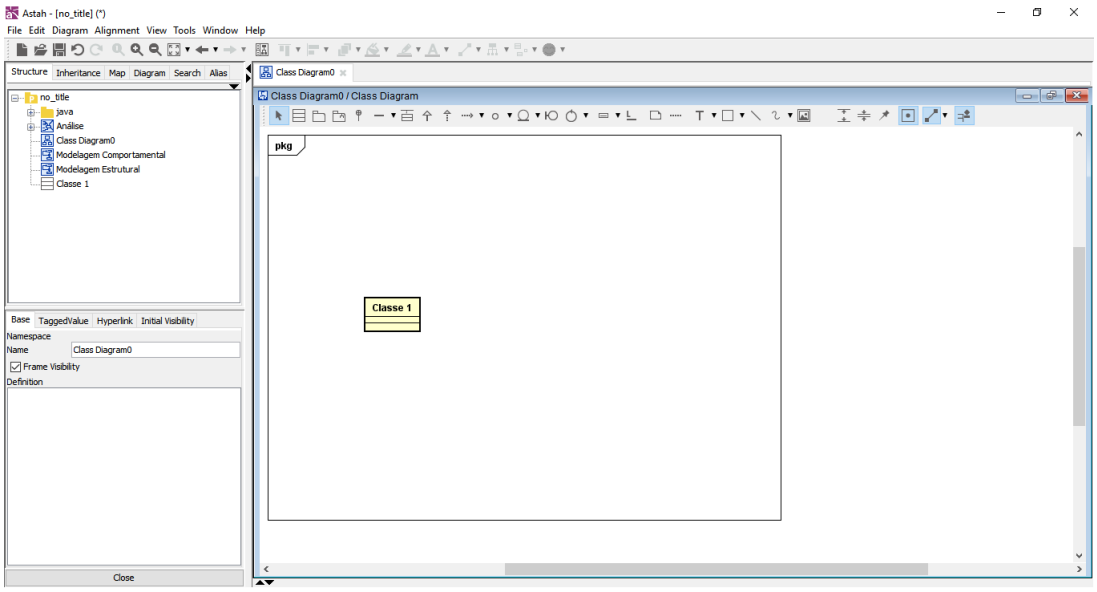


Fonte: elaborado pelo autor (2017)

4.2 Projeto e Prototipação

Este trabalho propõe a criação de um protótipo utilizando as técnicas de IHC, com a finalidade de melhorar a usabilidade das ferramentas CASE de modelagem UML baseadas na plataforma Eclipse. Para atingir tal objetivo, buscou-se descobrir quais estratégias precisam ser executadas durante o desenvolvimento de ferramentas CASE, dentre as quais foram apresentadas várias delas no Capítulo 3, para que seus usuários, os desen-

Figura 6 – Astah Professional



Fonte: elaborado pelo autor (2017)

volvedores de software, possam trabalhar utilizando ferramentas com menos problemas de usabilidade.

4.2.1 Planejamento da Prototipação

De acordo com o que foi levantado pela investigação sobre o estado da arte, foram encontradas as seguintes estratégias de melhoria de usabilidade para ferramentas CASE: utilização interfaces adaptativas, implementação do desenvolvimento com *touch screen*, aumento da utilização da linguagem natural e design centrado no usuário.

Este trabalho é focado no design centrado no usuário e no desenvolvimento *top-down*, na tentativa de fornecer melhores interfaces para os desenvolvedores. Dessa maneira, foi projetada e prototipada uma versão de IU, desenvolvida incrementalmente, com constantes consultas aos desenvolvedores para obter feedback das mudanças na interface e possíveis melhorias.

O protótipo foi desenvolvido a partir do zero, com funcionalidades levantadas pelo autor e posterior aplicação de *card sorting* para agrupá-las. Ao mesmo tempo, foram avaliadas as qualidades, aplicando a técnica do percurso cognitivo simplificado, de seis ferramentas CASE de modelagem UML, sendo três softwares livres (Modelio, Papyrus e UML Designer) e três softwares proprietários (Astah Professional, Enterprise Architect e IBM Rational Rose). Essas qualidades foram incorporadas ao protótipo e avaliadas se de fato melhoraram a usabilidade.

4.2.2 Definição das Funcionalidades e Características

Funcionalidades podem ser levantadas pela análise do ciclo de vida do negócio e do ciclo de vida das entidades, dentro do escopo do desenvolvimento de um projeto, identificando, então, as atividades necessárias para a criação e gerenciamento do negócio e as entidades manipuladas por estes (FURLAN, 1998).

Para definir as funcionalidades utilizadas no *card sorting* foi realizado um *brainstorm* com o orientador e o coorientador deste trabalho e realizada uma pesquisa em páginas na internet de ferramentas CASE de modelagem UML. Como resultado, chegou-se a uma lista com 30 funcionalidades. A Tabela 3 apresenta as funcionalidades levantadas durante esse processo.

Tabela 3 – Lista de Funcionalidades.

Funcionalidades		
1. Autocompletar	2. Drag-and-drop	3. Emissão de avisos com melhores práticas
4. Botões de controle de interrupção de execução na janela de Saída	5. Espaço de trabalho personalizável	6. Geração de diagramas UML a partir de especificações
7. Teclas de atalho	8. Corretor de sintaxe	9. Substituir, no projeto inteiro, linhas de texto ou código através de parâmetros especificados
10. Criação dos diagramas UML	11. Criação dos elementos do diagrama	12. Definir propriedades dos elementos
13. Suporte à diagramação dos elementos	14. Rastreabilidade entre diferentes modelos ou elementos	15. Atalhos para criar as ligações entre elementos
16. Engenharia de Código-Fonte (geração, sincronização e importação/reversa)	17. Suporte total à modelagem UML 2.5	18. Modelagem de Processos de Negócio
19. Matriz de Rastreabilidade	20. Interface de Importação e Exportação XMI	21. Gerenciamento de Requisitos
22. Integração com Aplicativos de Controle de Versão	23. Geração de Relatórios em HTML e RTF (Rich Text File)	24. Suporte ao Planejamento de Testes
25. Engenharia de dados (geração de scripts e engenharia reversa de dados)	26. Suporte à arquitetura dirigida a modelos	27. Log de mudanças em modelos
28. Segurança (papéis e permissões)	29. Suporte a SysML	30. Geração de Código a partir de diagramas comportamentais (ex: Sequência, Atividades, Estados)

Fonte: elaborado pelo autor (2017)

4.2.3 Desenvolvimento dos Protótipos

O primeiro protótipo desenvolvido foi criado a partir do zero com as funcionalidades levantadas na Tabela 3 e aplicado o *card sorting*. Após o levavantamento das funcionalidades, procurou-se ferramentas para executar o *card sorting* on-line e com funcionalidades de análise dos dados que facilitassem a extração de informações.

Encontrou-se o “Optimal Workshop”, uma ferramenta web que permite montar a estrutura do *card sorting*, podendo ser aberto ou fechado. Na versão grátis é possível enviar para até dez participantes. É possível visualizar o tempo de execução, quantos participantes executaram e o mais importante, a matrix de similaridade, que é o que vai dizer que funcionalidades se agrupam entre si[2].

A partir desse agrupamento, o protótipo foi desenvolvido, buscando atender aquilo que os desenvolvedores visualizam como uma interface ideal. Este estudo de classificação de cartões foi lançado em 06 de junho de 2017 e fechado em 12 de junho de 2017. A última resposta foi recebida em 09 de junho de 2017. Sete pessoas participaram, todos estudantes do curso de graduação em Engenharia de Software da UNIPAMPA, e classificaram todos os 30 cartões em uma média de 6 grupos. A média de tempo que os participantes levaram para execução da tarefa foi de 19 minutos e 17 segundos.

Conforme a Figura 7, pode-se ver o percentual de similaridade entre as funcionalidades, se todos os participantes agruparam duas funcionalidades a sua similaridade é 100, se nenhum agrupou a sua similaridade é 0. A partir dessa matriz e com a definição do percentual de similaridade de agrupamento foi definida a arquitetura do protótipo.

Figura 7 – Matriz de Similaridade

Fonte: elaborado pelo autor (2017)

[2] https://www.optimalworkshop.com/

Os grupos de funcionalidades resultantes deste primeiro *card sorting* são apresentados nas Figura 8 e Figura 9. A Figura 8 apresenta as funcionalidades relacionadas à melhoria de produtividade, melhoria da experiência do usuário, engenharia de requisitos e suporte.

Figura 8 – Primeiro grupo de funcionalidades

Melhoria da produtividade do usuário	Melhoria da experiência do usuário	Engenharia de requisitos	Suporte
Autocompletar	Teclas de atalho	Log de mudanças em modelos	A diagramação dos elementos
Drag-and-drop	Espaço de trabalho personalizável	Integração com aplicativos de controle de versão	UML 2.5
Importação e exportação XML	Corretor de sintaxe	Gerenciamento de requisitos	MDA
Substituir código através de parâmetros especificados	Emissão de avisos com melhores práticas	Rastreabilidade entre diferentes modelos ou elementos	SysML
		Matriz de rastreabilidade	Planejamento de testes

Fonte: elaborado pelo autor (2017)

A Figura 9 apresenta as funcionalidades relacionadas à modelagem, automação e duas funcionalidades que não foram agrupadas em nenhum grupo.

Figura 9 – Segundo grupo de funcionalidades

Modelagem	Automação	Não agrupados
Modelagem de processos de negócio	Geração de Código a partir de diagramas comportamentais (Sequência, Atividades, Estados)	Atalhos para criar as ligações entre elementos
Definir propriedades dos elementos	Engenharia de Dados (geração de scripts e engenharia reversa)	Segurança (papéis e permissões)
Criação dos elementos do diagrama	Engenharia de Código-Fonte (geração, sincronização e importação/ reversa)	
Criação dos diagramas UML		
Geração de diagramas UML a partir de especificações		

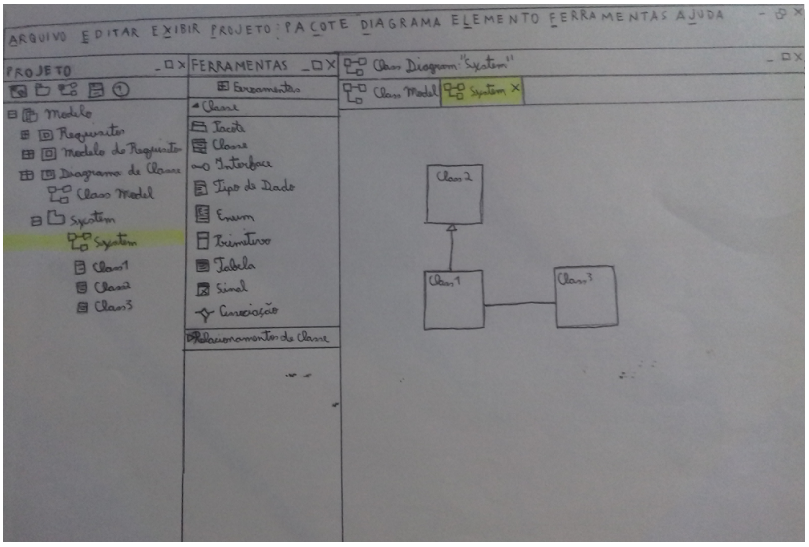
Fonte: elaborado pelo autor (2017)

A primeira versão do protótipo (Figura 10) foi desenvolvida utilizando lápis e papel. Nesta versão foi estabelecida a organização inicial da interface em função das funcionalidades levantadas.

Após a validação do protótipo em papel, foi iniciada a prototipação com a ferramenta Axure[3], desenvolvendo um protótipo funcional. Nesta versão foi refatorado a posição do menu de detalhamento, foi excluído o menu vertical de ferramentas e foram

[3] https://www.axure.com/

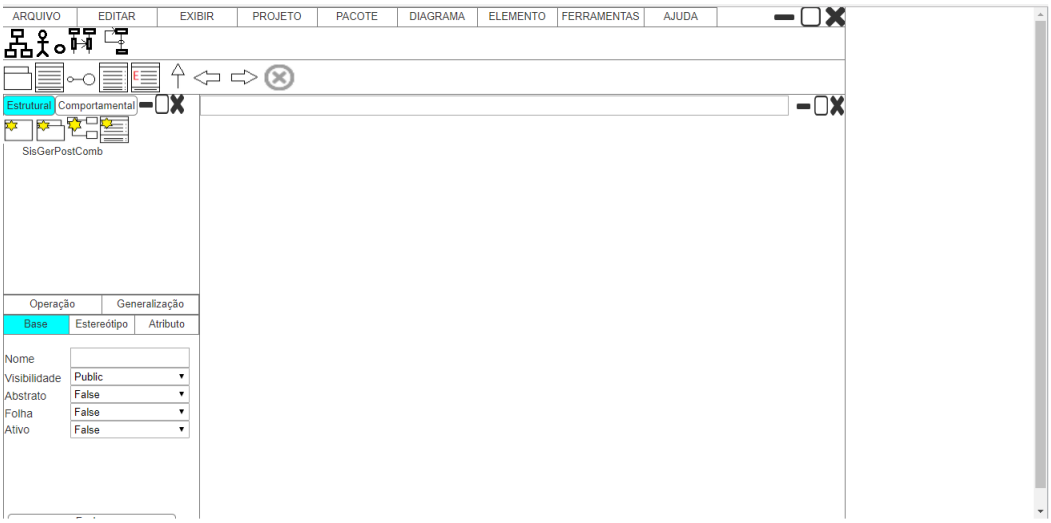
Figura 10 – Protótipo feito em papel



Fonte: elaborado pelo autor (2017)

adicionados a visão comportamental, um menu de diagramas fixo e um menu de elementos dos diagramas que exibia as opções específicas de cada diagrama. Com essas modificações chegou-se a versão atual do protótipo (Figura 11);

Figura 11 – Protótipo Funcional

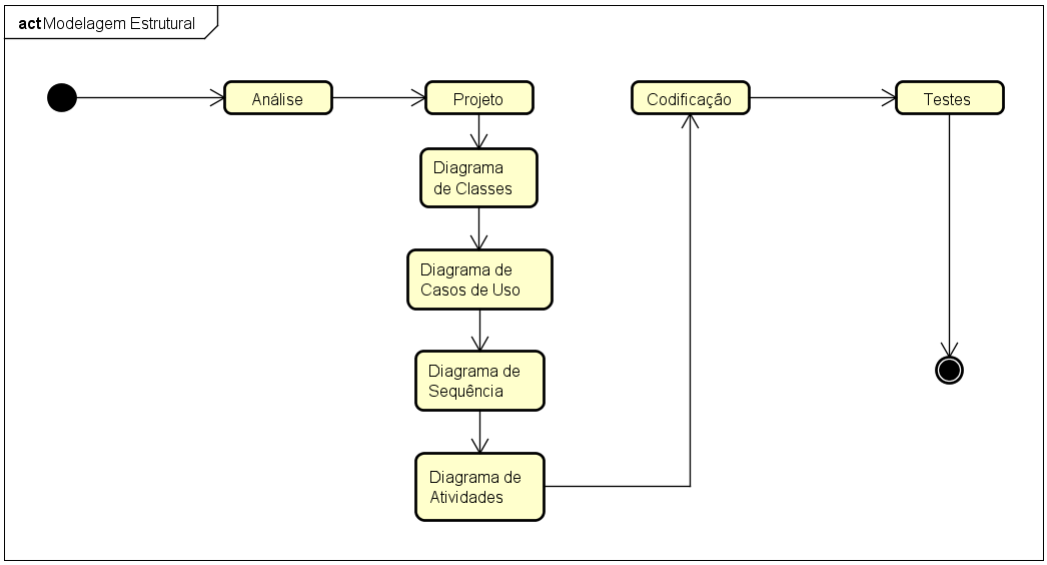


Fonte: elaborado pelo autor (2017)

A maioria das ferramentas utilizadas pela indústria segue a abordagem estrutural, onde a modelagem é realizada analogamente ao processo de desenvolvimento de software cascata. O Modelo Cascata é um modelo de desenvolvimento de software seqüencial no qual o desenvolvimento é visto como um fluir constante para frente, através das fases de análise de requisitos, projeto, implementação, testes, integração, e manutenção de software (SOMMERVILLE, 2011). Na modelagem acontece algo semelhante, conforme

o diagrama exibido na Figura 12, os analistas modelam todo o projeto, na maioria das vezes, na seguinte sequência: diagrama de classes, diagrama de casos de uso, diagrama de sequência e diagrama de atividades. Modelando um a um, iniciando o próximo diagrama apenas ao terminar o anterior.

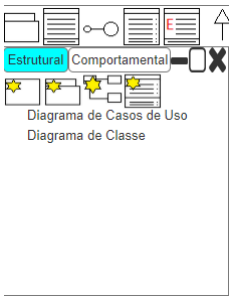
Figura 12 – Modelagem estrutural



Fonte: elaborado pelo autor (2017)

O protótipo tem a visão estrutural (Figura 13) e foi adicionada uma visão comportamental (Figura 14). Esta visão tem forte influência das metodologias ágeis e permite que o analista modele o projeto por caso de uso, permitindo uma maior independência entre os artefatos de modelagem e que sejam entregues pacotes com partes do projeto para os desenvolvedores. A visão comportamental permitiria livre fluxo entre os diagramas e reuso dos diagramas entre diferentes casos de uso, conforme o diagrama exibido na Figura 15.

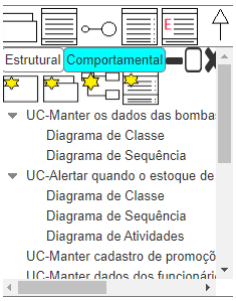
Figura 13 – Visão estrutural



Fonte: elaborado pelo autor (2017)

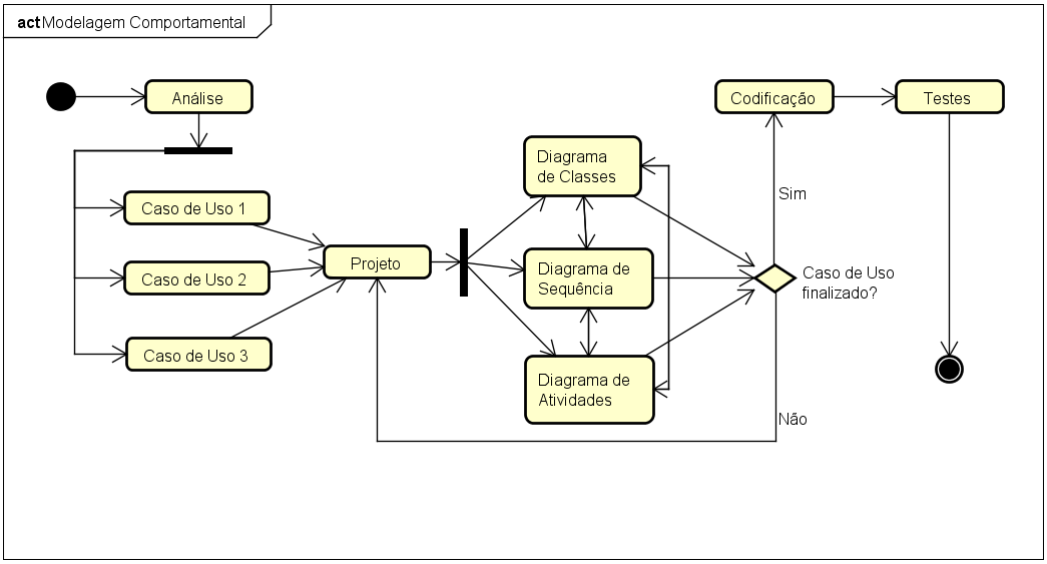
A rastreabilidade de artefatos de software é amplamente reconhecida como uma fator importante para o desenvolvimento efetivo e manutenção de um software. Infe-

Figura 14 – Visão comportamental



Fonte: elaborado pelo autor (2017)

Figura 15 – Modelagem comportamental

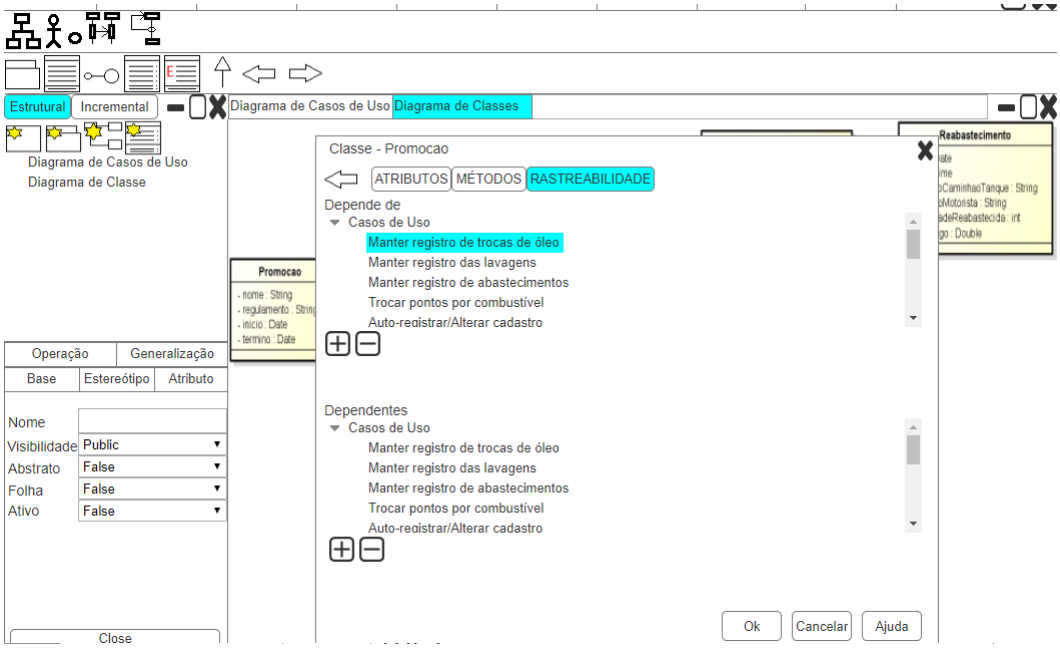


Fonte: elaborado pelo autor (2017)

lizmente, a falta de ferramentas de suporte automatizadas, ainda mantém a tarefa de relacionar as dependências entre artefatos de software como uma tarefa demorada. Por esse motivo, muitas vezes informações de rastreabilidade ficam desatualizadas ou completamente ausentes durante o desenvolvimento de software. O protótipo deste trabalho permite facilitar e agilizar a tarefa de implementar as dependências entre os artefatos, ao clicar em cada caso de uso ou classe é possível acessar na aba "Rastreabilidade" as informações dos artefatos que dependem do artefato selecionado e das dependências do artefato selecionado Figura 16.

O protótipo exibido neste trabalho é apenas um extrato, a versão completa do protótipo pode ser acessada no seguinte link: "https://drive.google.com/drive/folders/1Qdfe74GuaK6hySJFjI1eCQVlBRyzanla?usp=sharing".

Figura 16 – Listas de Dependência



Fonte: elaborado pelo autor (2017)

4.3 Lições do Capítulo

Neste Capítulo foi planejado o desenvolvimento do protótipo, utilizando técnicas de IHC. Para a modelagem da arquitetura do protótipo foram identificados, através de pesquisa e entrevistas, trinta funcionalidades necessárias a uma ferramenta CASE de modelagem UML.

A partir desse conjunto de funcionalidades e com auxílio de sete alunos do curso de graduação em Engenharia de Software, foi executado o *card sorting*, como método de arquitetura de informação, e as funcionalidades foram agrupadas em sete grupos para a prototipação, que é uma das soluções para os problemas de usabilidade das interfaces de usuário.

5 AVALIAÇÃO DO PROJETO

Neste Capítulo, é apresentado a avaliação do protótipo e seus resultados. Na seção 5.1 é apresentado o protocolo de avaliação do projeto interface de usuário para ferramentas de modelagem UML. Na seção 5.2 os resultados da avaliação são discutidos com a finalidade de extrair melhorias para a interface do protótipo. Por fim, na seção 5.3, são apresentadas as lições do Capítulo.

5.1 Protocolo

Para executar a avaliação foi desenvolvido o protocolo abaixo, onde o autor deste trabalho assumiu a função de observador. Foi desenvolvido um plano de teste de usabilidade para definir usuários, tarefas e procedimentos. Foram selecionadas as tarefas mais importantes e os grupos de usuários a serem testados. A tarefa foi escolhida com base nos recursos disponíveis para teste e na criticidade.

Cinco usuários representativos foram recrutados e realizaram a avaliação entre os dias 06 e 10 de novembro de 2017. Dos cinco avaliadores, dois eram estudantes do curso de Ciência da Computação e três do curso de Engenharia de Software, todos com pelo menos três anos e meio de curso e algum conhecimento de UML e modelagem. Como os objetivos do teste são encontrar problemas e verificar o impacto do produto nos usuários, cinco avaliadores são suficientes (NIELSEN; LANDAUER, 1993).

As sessões de teste foram planejadas para que permitissem haver tempo para dar instruções, executar o teste e realizar uma entrevista pós-teste. As sessões de teste com cerca de meia hora exigiram o agendamento prévio com os avaliadores. Para melhor aproveitamento das avaliações e para que pequenos, mas importantes detalhes, não fossem perdidos, as sessões de avaliação foram gravadas.

Para avaliação, o observador ficou ao lado do avaliador para solicitar e questionar quando necessário. O cenário de teste foi preparado juntamente com outros materiais adicionais, incluindo o Termo de Confidencialidade (Apêndice A) e o Termo de Consentimento Livre e Esclarecido (Apêndice B), informando sobre a participação e a gravação.

O protótipo de interface foi avaliado por cinco avaliadores. Segundo Nielsen e Landauer (1993), executar testes múltiplos ajudam a melhorar o design e não apenas documentar suas fraquezas. Após a primeira avaliação com cinco participantes é possível identificar cerca de 85% dos problemas de usabilidade, o que permite melhorar a interface de usuário, resolvendo esses problemas e redesenhando a interface.

Neste teste de usuário foi proposto um cenário de teste (subseção 5.1.1), com 10 tarefas para que o avaliador as executasse e identificasse possíveis melhorias na usabilidade da interface de usuário do protótipo. De acordo com o cenário, o avaliador estaria sendo contratado por uma empresa que segue o Manifesto para o desenvolvimento ágil de

software[1] e iria trabalhar utilizando uma nova metodologia de modelagem, a modelagem comportamental.

5.1.1 Cenário de Teste

Nesta subseção é apresentado o cenário de teste com o usuário, suas tarefas e as telas resultantes de cada tarefa.

"Suponha que você foi contratado por uma empresa de desenvolvimento de software para fazer parte de uma equipe de desenvolvedores. Esta empresa utiliza o modelo de desenvolvimento comportamental, fortemente inspirado nos métodos ágeis. Onde valorizamos software em funcionamento mais que documentação abrangente e responder a mudanças mais que seguir um plano.

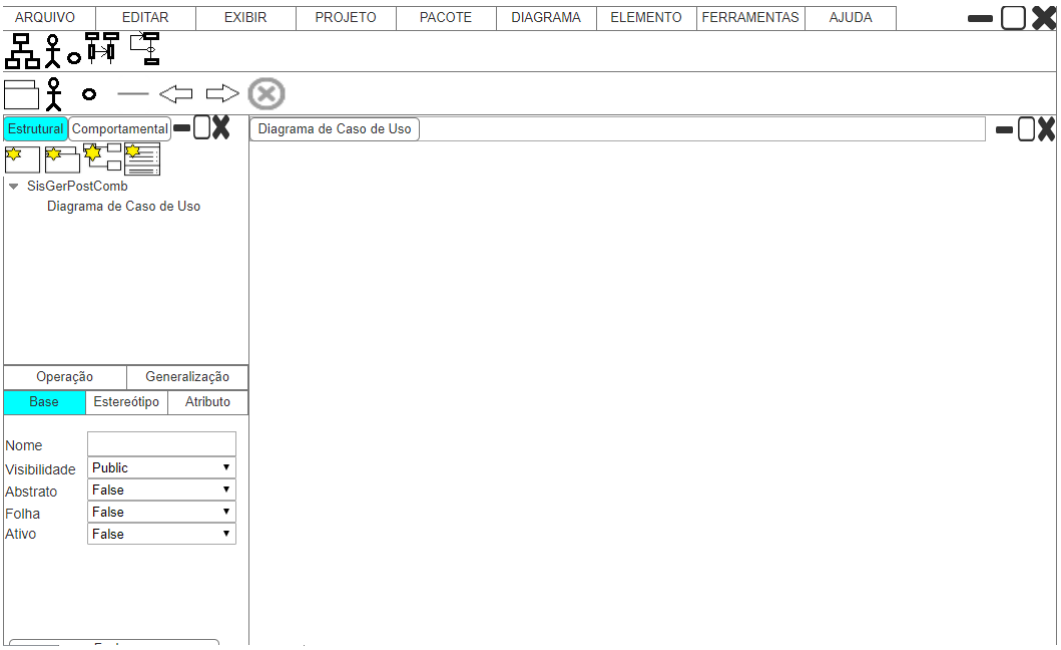
Os artefatos de software são modelados por casos de uso e podem ser reaproveitados entre os casos de uso, teremos diagrama de classe do caso de uso 1 e diagrama de classe do caso de uso 2, com as informações necessárias a implementação de seus respectivos casos de uso.

Isso vai aumentar o desacoplamento entre os casos de uso, o gerente de projeto poderia designar dois funcionários diferentes para trabalharem cada um em um caso de uso. E ao finalizar o funcionário A entregaria os artefatos gerados pelo seu trabalho no caso de uso, independente se o funcionário B terminou ou não. Seu primeiro trabalho será desenvolver um sistema de gerenciamento de postos de combustíveis, e você ficará encarregado de executar as seguintes tarefas:"

Passo 1: Criar o diagrama de casos de uso (Figura 17);

[1] http://agilemanifesto.org/

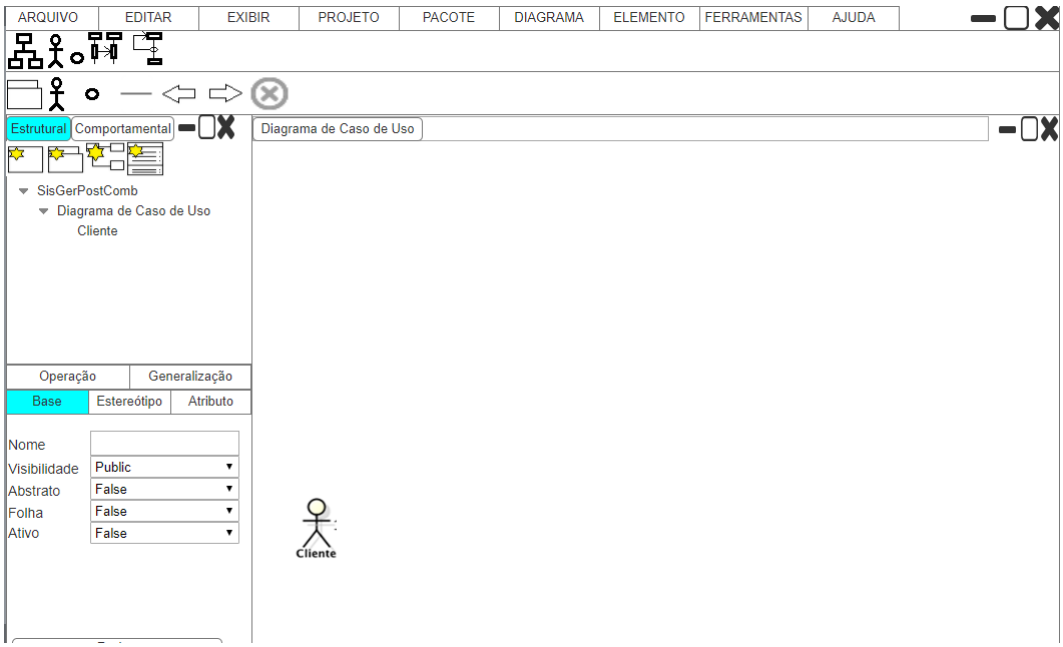
Figura 17 – Diagrama de casos de uso criado



Fonte: elaborado pelo autor (2017)

Passo 2: Criar o ator Cliente (Figura 18);

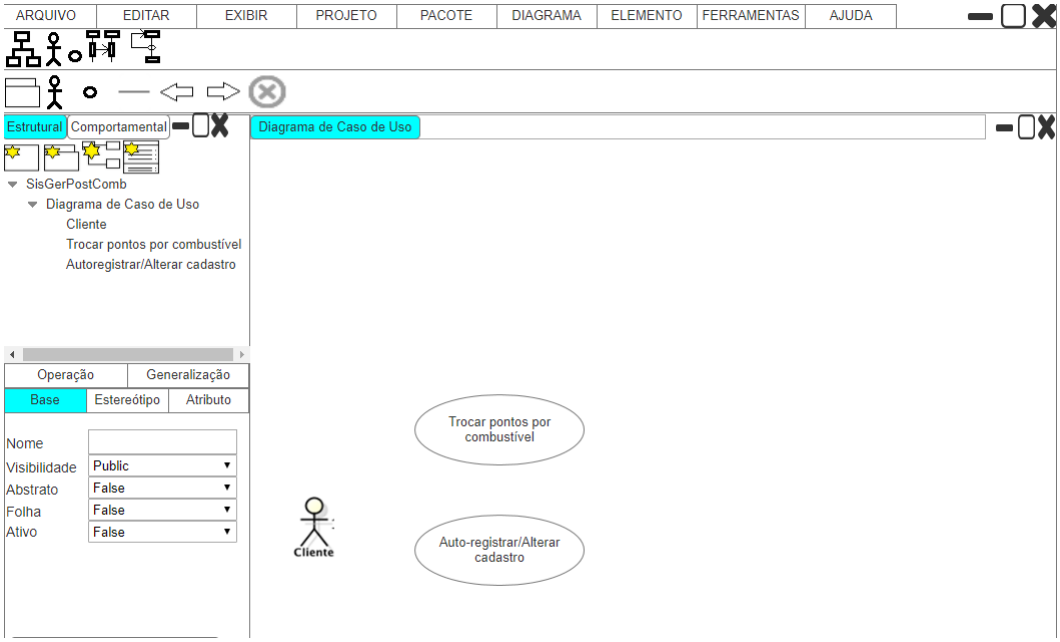
Figura 18 – Ator criado



Fonte: elaborado pelo autor (2017)

Passo 3: Criar os casos de uso Trocar pontos por combustível e Autoregistrar/Alterar cadastro (Figura 19);

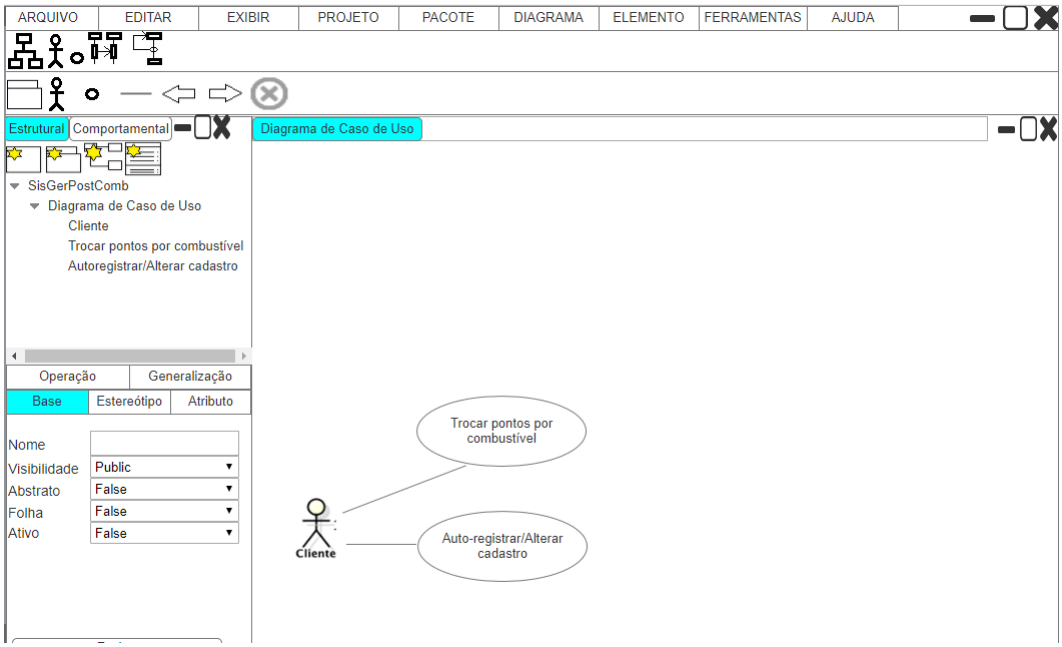
Figura 19 – Casos de uso criados



Fonte: elaborado pelo autor (2017)

Passo 4: Relacionar o ator Cliente com os dois casos de uso (Figura 20);

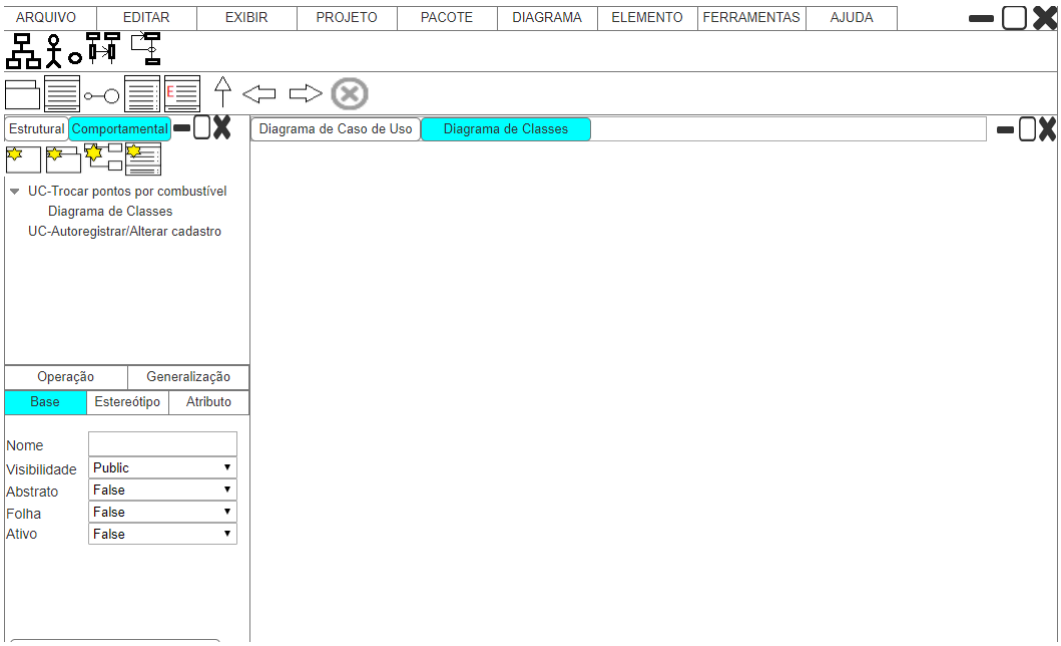
Figura 20 – Ator e casos de uso relacionados



Fonte: elaborado pelo autor (2017)

Passo 5: Criar um diagrama de classes no caso de uso Trocar pontos por combustível (Figura 21);

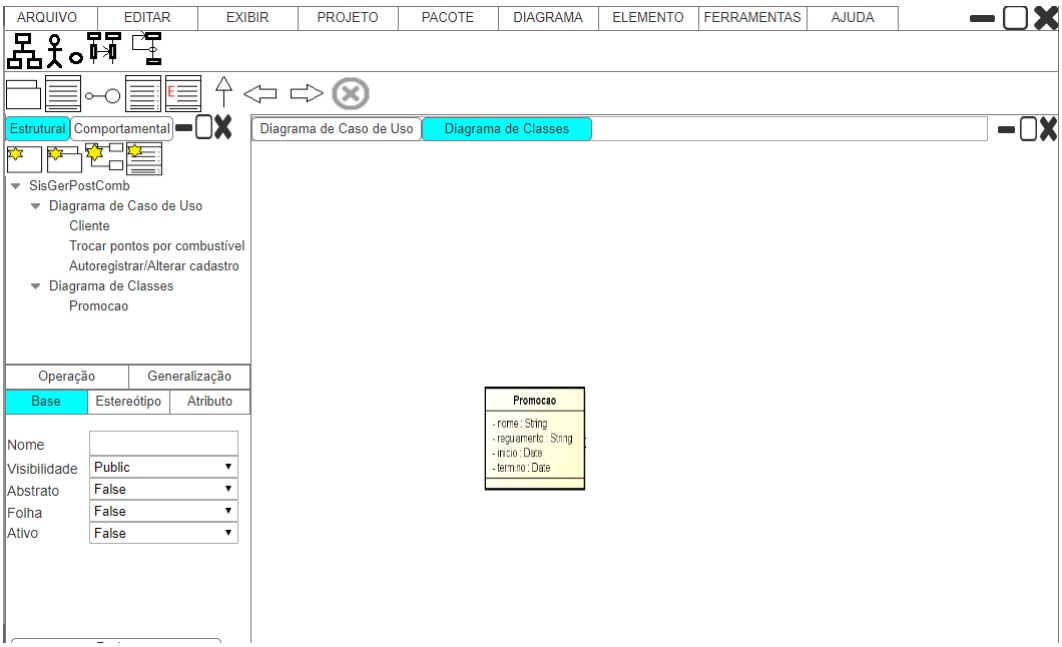
Figura 21 – Diagrama de classes criado



Fonte: elaborado pelo autor (2017)

Passo 6: Criar a classe Promocao (Figura 22);

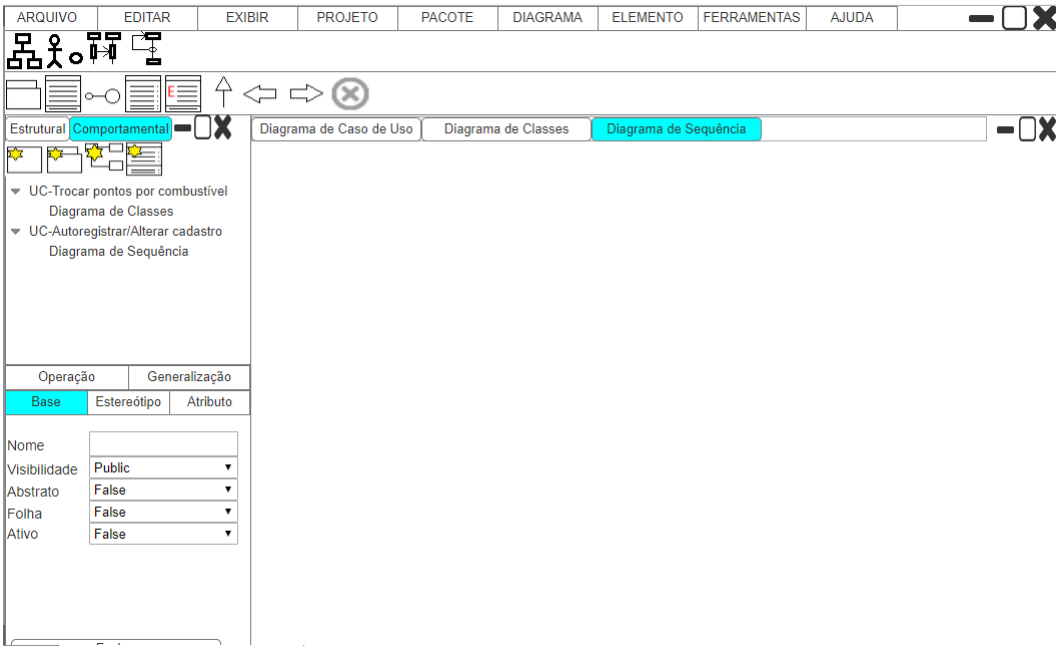
Figura 22 – Classe Promocao criada



Fonte: elaborado pelo autor (2017)

Passo 7: Criar um diagrama de sequência no caso de uso Autoregistrar/Alterar cadastro (Figura 23);

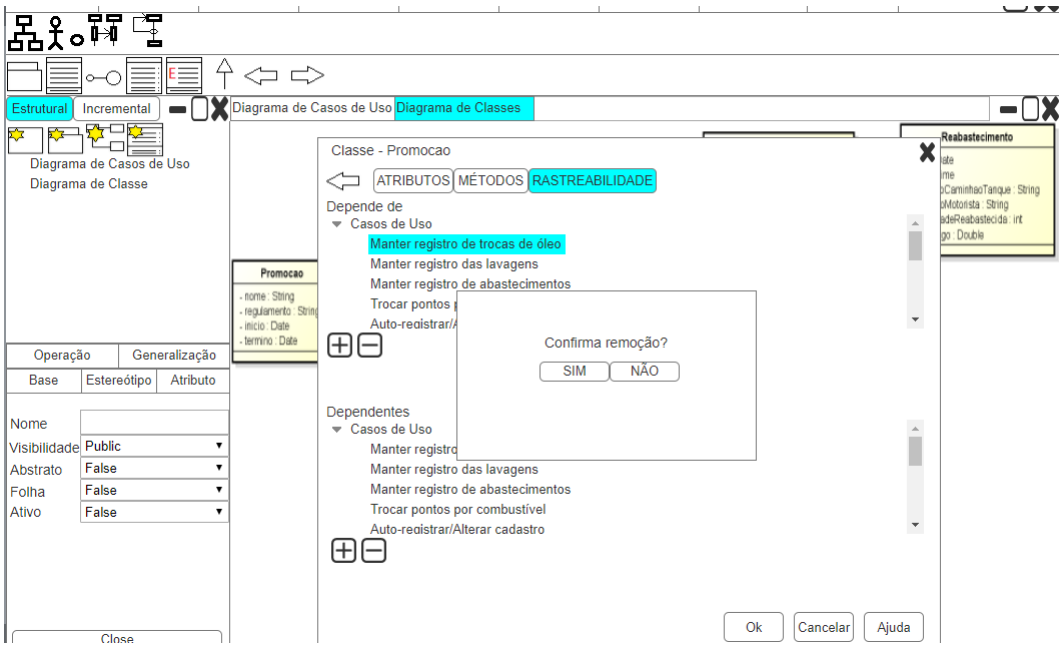
Figura 23 – Diagrama de sequência criado



Fonte: elaborado pelo autor (2017)

Passo 8: Na classe Promocao, excluir o caso de uso Manter registro de troca de óleo de uma das listas de dependência (Figura 24);

Figura 24 – Excluindo o caso de uso Manter registro de troca de óleo de uma das listas de dependência

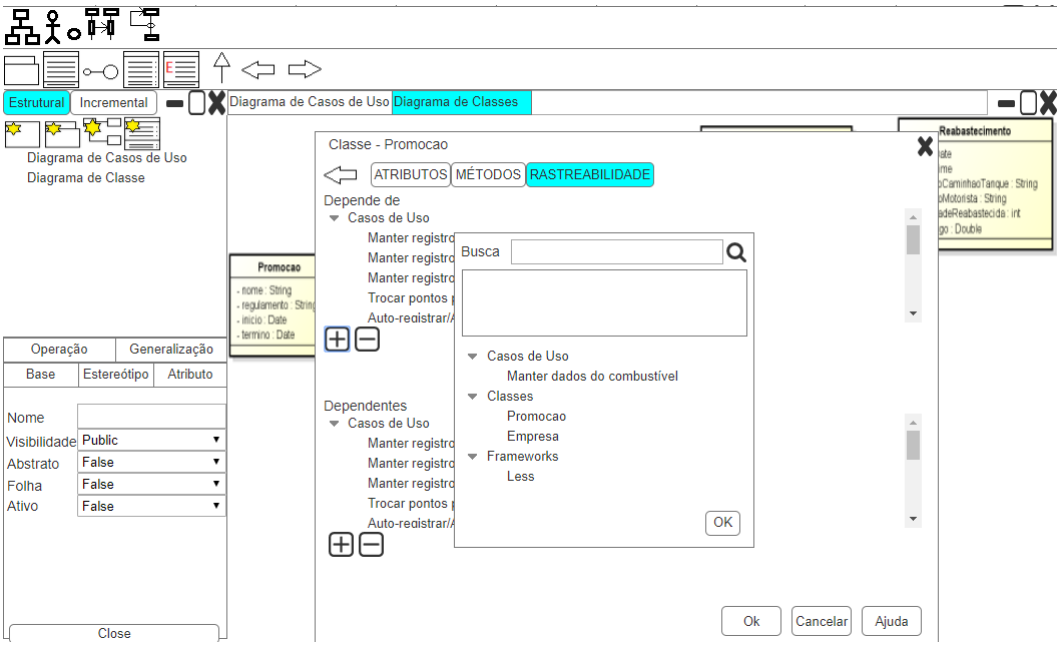


Fonte: elaborado pelo autor (2017)

Passo 9: Na classe Promocao, adicionar o caso de uso Manter dados do combus-

tível a uma das listas dependência (Figura 25);

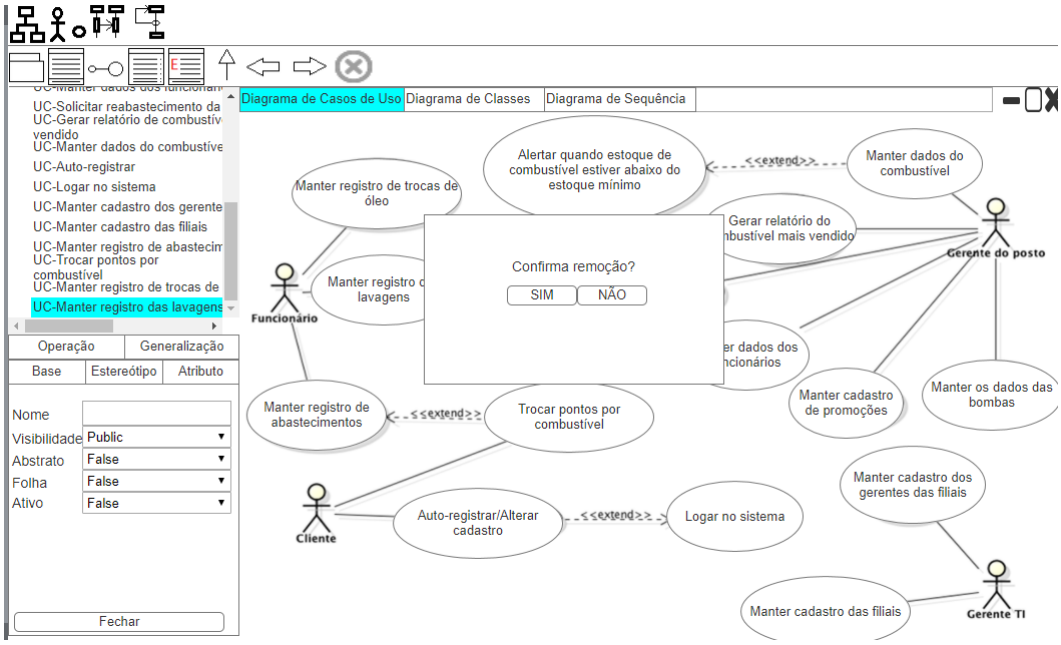
Figura 25 – Adicionando o caso de uso Manter dados do combustível a uma das listas dependência



Fonte: elaborado pelo autor (2017)

Passo 10: Deletar o caso de uso de uso Manter registro das lavagens (Figura 26).

Figura 26 – Deletando o caso de uso de uso Manter registro das lavagens



Fonte: elaborado pelo autor (2017)

5.2 Resultado

Os cinco avaliadores demoraram em média 9 minutos e 28 segundos para execução total das tarefas e obtiveram uma taxa média de sucesso na execução das tarefas de 80%. Na Tabela 4 é possível visualizar a taxa de sucesso e o tempo de execução de cada avaliador. A taxa de sucesso é medida pela correta execução das 10 tarefas, sendo cada tarefa executada corretamente equivalente a 10%.

Tabela 4 – Taxa de sucesso na execução das tarefas

Participante	**Taxa de sucesso (%)**	**Tempo de execução (min)**
1	90	8
2	80	14
3	80	12
4	80	12
5	70	16
Total	80	9,5

Fonte: elaborado pelo autor (2017)

Alguns avaliadores tentaram arrastar e soltar ícones na tentativa de criar os diagramas, e tentam também utilizar o botão direito do mouse para criar relações entre os casos de uso e o ator, ações que não são possíveis devido a limitações da ferramenta de prototipação, mas são possíveis em ferramentas reais e melhoram a interação com a ferramenta e a usabilidade.

Uma dificuldade comum entre todos os avaliadores foi executar as tarefas que envolviam criar artefatos nos casos de uso, utilizando a visão comportamental. Eles tentaram criar utilizando a visão estrutural, clicaram em diversos botões e somente conseguiram concluir a tarefa após receberem uma dica sobre o que é modelagem comportamental. A Tabela 5 apresenta algumas das observações mais relevantes realizadas pelos avaliadores sobre o protótipo, os avaliadores notaram que a modelagem comportamental facilita e agiliza o trabalho de modelagem.

De modo geral os avaliadores, assim como qualquer pessoa que não seja especialista em modelagem e não tenha um domínio total sobre os ícones da UML e ferramentas de modelagem, na maiorias das vezes têm dificuldades em saber o que faz determinado ícone ou como realizar determinada tarefa. Essa dificuldade foi contornada no protótipo, utilizando ícones autoexplicativos e legendas sobre o que faz cada ícone, mas ainda assim poderia ser minimizada, colocando um "+" nos ícones de criar novo elemento e a exibição permanente de uma legenda com o significado dos ícones.

5.3 Lições do Capítulo

Neste Capítulo foi apresentado o protocolo de avaliação do protótipo de interface e a diferença de entre a modelagem estrutural e a modelagem comportamental. A mo-

Tabela 5 – Observações dos Avaliadores

Observações
"Nós nos acostumamos a modelar utilizando a visão estrutural, mas fica até bem mais fácil modelar utilizando a visão comportamental, fica mais rápido modelar depois que o usuário aprende a utilizar esta outra visão."
"O protótipo ficou interessante e o jeito de modelar ficou mais prático. Os ícones tem uma grande importância para facilitar o aprendizado da ferramenta."
"A modelagem comportamental poderia quebrar a dependência entre os diagramas."
"O protótipo tem uma interface bem usual, muitos programas utilizam essa mesma interface: menus, barra de tarefas e ícones."

Fonte: elaborado pelo autor (2017)

delagem comportamental em uma ferramenta CASE de modelagem pode produzir novas técnicas e processos de modelagem, sendo muito útil para os desenvolvedores que trabalham com métodos ágeis.

A partir dos resultados é possível verificar que o protótipo ainda tem muitas oportunidades de melhoria, e o fato de alguns avaliadores discordarem de qual posição um menu deve estar posicionado, somente evidencia que além de um bom planejamento, com realização de avaliações, os desenvolvedores devem permitir que os usúarios possam modificar o leiaute da interface da forma que melhor lhe agrade.

6 CONSIDERAÇÕES FINAIS

Este trabalho tem como objetivo criar um protótipo utilizando as técnicas de IHC, com a finalidade de melhorar a usabilidade das ferramentas CASE de modelagem UML baseada em Eclipse, avaliar o protótipo e consolidar o protótipo com as melhores soluções, sendo utilizado como estudo de caso ferramentas CASE de modelagem utilizadas pela indústria atualmente.

O estudo comprovou que ferramentas que não têm uma interface de usuário com uma boa usabilidade, provavelmente caem em desuso. Sua importância está relacionada à sua própria atividade, pois como desenvolvedores podem construir ferramentas para outros desenvolvedores sem se preocupar com usabilidade das interfaces.

Verificou-se que não existe uma solução perfeita para desenvolver boas IU, mas sim, boas práticas que devem ser seguidas. Os desenvolvedores de ferramentas devem sempre buscar a excelência na qualidade de uso e um design centrado no usuário, que é o público-alvo do seu produto. Para isso, as estratégias aqui implementadas devem ser implementadas durante a execução dos processos de desenvolvimento de software.

Este trabalho contribuiu com estratégias para a melhoria da usabilidade de ferramentas CASE de modelagem UML baseada em Eclipse, permitindo que ferramentas com melhor usabilidade possam ser desenvolvidas e apresentando um protótipo de IU, que pode ser implementado no desenvolvimento de novas ferramentas ou na melhoria de ferramentas já existentes. Uma inovação que foi apresentada pelo protótipo, a visão comportamental, permite o desenvolvimento de um nova forma de modelar, a modelagem comportamental, que segundo os avaliadores torna a modelagem mais prática do que a modelagem estrutural.

Uma das limitações deste trabalho foram as limitações da própria ferramenta de prototipação, que não permitiam a utilização do botão direito do mouse e o *drag and drop* de elementos nos diagramas. Uma limitação derivada de pesquisas exploratórias é que essas dependem da interpretação do pesquisador com base nas declarações obtidas. As interpretações das percepções dos avaliadores podem sofrer influência das percepções do observador da avaliação.

Como trabalho futuro, sugere-se melhorar o protótipo, oferecendo mais funcionalidades para os usuário e ampliar o cenário de teste, permitindo uma interação livre entre o usuário e a ferramenta. Para aumentar o valor científico deste trabalho, seria relevante a possibilidade de aplicar o projeto de IU aqui apresentado em uma ferramenta CASE de modelagem real, validando o resultado das avaliações do protótipo. Este trabalho faz parte de um trabalho maior, uma ferramenta de modelagem de requisitos para sistema autoadaptativos, onde a interface de usuário aqui desenvolvida, futuramente será implementada.

REFERÊNCIAS

AKIKI, P.; BANDARA, A.; YU, Y. Engineering adaptive model-driven user interfaces. **IEEE Transactions on Software Engineering**, v. 42, n. 12, p. 1118–1147, 2016. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85006710450&doi=10.1109%2fTSE.2016.2553035&partnerID=40&md5=f56f6372b900ca991623f1aefbd140ad>. Citado 5 vezes nas páginas 24, 33, 34, 35 e 36.

BAČÍKOVÁ, M.; MARIČÁK, M.; VANČÍK, M. Usability of a domain-specific language for a gesture-driven IDE. In: **Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015**. [S.l.: s.n.], 2015. ISBN 9788360810651. Citado 3 vezes nas páginas 34, 35 e 36.

BARBOSA, S.; SILVA, B. **Interação Humano-Computador**. [S.l.]: Editora Campus-Elsevier, 2010. Citado 4 vezes nas páginas 23, 27, 28 e 39.

BIEGEL, B. et al. U can touch this: Touchifying an IDE. In: **8th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2014 - Proceedings**. [S.l.: s.n.], 2014. ISBN 9781450328609. Citado 3 vezes nas páginas 33, 35 e 36.

CERVO, A.; SILVA, R. **Metodologia Científica**. [S.l.]: Prentice Hall Brasil, 2006. Citado na página 25.

CYBIS, W.; BETIOL, W.; FAUST, R. **Ergonomia e Usabilidade: conhecimentos, métodos e aplicações**. [S.l.]: Novatec Editora Ltda, 2007. Citado 3 vezes nas páginas 23, 27 e 29.

DILLON; THOMPSON. Software development and tool usability. In: **Program Comprehension (ICPC), 2016 IEEE 24th International Conference on**. [S.l.: s.n.], 2016. Citado 6 vezes nas páginas 23, 24, 33, 35, 36 e 37.

FINSTAD, K. **The usability metric for user experience**. [S.l.]: Interact. Comput., 2010. Citado na página 24.

FURLAN, J. D. **Modelagem de Objetos através da UML**. [S.l.]: Makron Books, 1998. Citado na página 44.

HEWETT et al. **ACM SIGCHI Curricula for Human-Computer Interaction**. [S.l.]: ACm SIGCHI Report, 1992. Citado na página 27.

HOU, D.; WANG, Y. An empirical analysis of the evolution of user-visible features in an integrated development environment. In: . [s.n.], 2009. p. 122–135. Cited By 11. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-77951553792&doi=10.1145%2f1723028.1723044&partnerID=40&md5=6d53241317b4565dceccc2b5e06c7d9c>. Citado 3 vezes nas páginas 32, 35 e 36.

http://www.usabilitybok.org. **Usability Body of Knowledge**. acessado em 01/06/2017. Citado 3 vezes nas páginas 28, 30 e 39.

http://www.usability.gov. **The Research-Based Web Design and Usability Guidelines, Enlarged/Expanded edition.** acessado em 01/06/2017. Citado na página 28.

ISO/IEC 14102. **ISO/IEC 14102**. [S.l.]: ISO, 2008. Citado na página 30.

ISO/IEC 9126. **ISO/IEC 9126**. [S.l.]: ISO, 1991. Citado na página 27.

MILLER, G. A. **The magical number seven, plus or minus two: Some limits on our capacity for processing information**. [S.l.]: Psychological Review, 1956. Citado na página 40.

NBR 9241-11. **NBR 9241-11**. [S.l.]: ABNT, 2002. Citado na página 30.

NBR ISO/IEC 9126-1. **NBR ISO/IEC 9126-1**. [S.l.]: ISO, 2001. Citado na página 27.

NIELSEN, J.; LANDAUER, T. K. **A mathematical model of the finding of usability problems**. [S.l.]: Proceedings of ACM INTERCHI'93 Conference (Amsterdam, The Netherlands), 1993. Citado na página 51.

NIELSEN, J.; MACK, R. **Heuristic evaluation**. [S.l.]: Usability Inspection Methods, John Wiley and Sons, New York, NY., 1994. Citado 2 vezes nas páginas 29 e 33.

NORMAN, D. A.; DRAPER, S. W. **User Centered System Design; New Perspectives on Human-Computer Interaction**. [S.l.]: L. Erlbaum Associates Inc. Hillsdale, NJ, USA, 1986. Citado na página 23.

RYCERZ, K. b. et al. Teaching quantum computing with the quide simulator. In: . [s.n.], 2015. v. 51, n. 1, p. 1724–1733. Cited By 0. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84939166723&doi=10.1016%2fj.procs.2015.05.374&partnerID=40&md5=16a60b17733ad3c3a0115c919a714d57>. Citado 4 vezes nas páginas 24, 33, 35 e 36.

SEFFAH, A.; GULLIKSEN, J.; DESMARAIS, M. C. **Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle**. [S.l.]: Springer Science And Business Media, 2006. Citado na página 23.

SOMMERVILLE, I. **Engenharia de Software 9ª Ed.** [S.l.]: Pearson Prentice Hall, 2011. Citado 3 vezes nas páginas 27, 30 e 47.

SPENCER, D.; WARFEL, T. **Card sorting: a definitive guide**. [s.n.], 2004. Disponível em: <http://boxesandarrows.com/card-sorting-a-definitive-guide>. Citado na página 29.

ZOU, Y. et al. Adapting the user interface of integrated development environments (ides) for novice users. **Journal of Object Technology**, v. 7, n. 7, p. 55–74, 2008. Cited By 2. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-55549119307&partnerID=40&md5=b512b491293fd3db8b041eb0e0c360cf>. Citado 5 vezes nas páginas 23, 24, 32, 34 e 36.

Apêndices

APÊNDICE A – TERMO DE CONFIDENCIALIDADE

Título do projeto: Um Projeto de Interface para Ferramentas de Modelagem UML Baseada em Eclipse
Pesquisadores responsáveis: João Pablo Silva da Silva, Jean Felipe Patikowski Cheiran
Pesquisador participante: Mário Alan de Oliveira Lima
Campus/Curso: Alegrete/Engenharia de Software
Telefone para contato: 55 999473612 / 55 3421-8400 (ramal 3056)
Local da coleta de dados: Campus da UNIPAMPA

Os pesquisadores do presente trabalho se comprometem a preservar a privacidade e o anonimato dos participantes cujos dados serão coletados (1) na gravação de fala durante o teste de usabilidade, (2) na gravação da tela do computador durante o teste de usabilidade, e (3) nas respostas das perguntas realizadas durante os testes de usabilidade. Concordam, igualmente, que estas informações serão utilizadas única e exclusivamente para execução do presente trabalho. As informações somente poderão ser divulgadas preservando o anonimato dos participantes e serão mantidas em poder dos responsáveis pela pesquisa, professores João Pablo Silva da Silva e Jean Felipe Patikowski Cheiran e pelo acadêmico Mário Alan de Oliveira Lima por um período de 5 anos. Após este período, os dados serão destruídos.

Alegrete, 26 de outubro de 2017.

.....
João Pablo Silva da Silva
SIAPE 1864649

.....
Jean Felipe Patikowski Cheiran
SIAPE 2078666

.....
Mário Alan de Oliveira Lima
CPF 031.873.985-29

APÊNDICE B – TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Título do projeto: UM Projeto de Interface para Ferramentas de Modelagem UML Baseada em Eclipse

Pesquisadores responsáveis: João Pablo Silva da Silva, Jean Felipe Patikowski Cheiran

Pesquisador participante: Mário Alan de Oliveira Lima

Instituição: Universidade Federal do Pampa – Unipampa

Você está sendo convidado para participar, como voluntário, de testes de usabilidade no trabalho de conclusão de curso (TCC) intitulado "Um Projeto de Interface para Ferramentas de Modelagem UML Baseada em Eclipse". Esse trabalho de conclusão de curso tem como objetivo propor uma interface para modelagem UML em ferramentas CASE com alta usabilidade.

Você pode a qualquer momento solicitar esclarecimentos sobre quaisquer aspectos do TCC ou do teste de usabilidade (dados coletados, identificação dos participantes, demais envolvidos, etc.). Você também poderá interromper a participação (retirando seu consentimento) a qualquer momento sem sofrer qualquer tipo de penalidade ou prejuízo.

Após ler e tirar suas dúvidas sobre as informações a seguir, se aceitar fazer parte do teste, assine ao final deste documento, que está em duas vias. Uma delas é sua e a outra será arquivada pelo pesquisador responsável.

O que você precisará fazer no teste de usabilidade:

1. Responder algumas perguntas antes (e talvez durante) do teste.
2. Aceitar a gravação da tela do computador e da fala durante o teste.
3. Falar em voz alta o que estiver passando por sua cabeça enquanto estiver realizando as tarefas.
4. Tentar realizar as tarefas.

Riscos que você corre ao participar da pesquisa:

1. Frustrar-se ou irritar-se por não conseguir completar uma tarefa ou por não obter ajuda do pesquisador durante o teste.

Benefícios da pesquisa:

1. Os resultados desses testes permitirão avaliar a qualidade geral da interface proposta e identificar melhorias, colaborando para criação de ferramentas de modelagem mais usáveis.

Participar dessa pesquisa não gera nenhum custo. Você também não receberá qualquer vantagem financeira.

Seu nome e identidade serão mantidos em sigilo, e os dados da pesquisa serão armazenados pelos pesquisadores responsáveis. Os resultados poderão ser divulgados no texto final do TCC, em publicações ou outras formas de divulgação respeitando sempre o anonimato.

Nome do Participante da Pesquisa:.....

.....
Assinatura do Participante da Pesquisa

Nome do Pesquisador: Mário Alan de Oliveira Lima

.....
Assinatura do Pesquisador Responsável

Local e data .....

C RESULTADO DO *CARD SORTING*

C.1 Visão Geral

Este *card sorting* foi realizado na ferramenta OptimalSort[1], foi iniciado em 07 de setembro de 2017 e finalizado em 22 de setembro de 2017. A última resposta foi recebida em 22 de setembro de 2017. Foram completas 7 execuções e 1 foi abandonada. Participaram 8 pessoas, localizadas no Brasil, e 7 (87%) dessas pessoas classificaram os 14 cartões em uma média de 4 grupos.

C.2 Análise

C.2.1 Participantes

As Tabela 6 apresenta o resultado do *card sorting* por participante.

Tabela 6 – Resultado por participante

Participante	**Tempo de execução**	**Cartões organizados**	**Categorias criadas**	**Categorias nomeadas**
1	3:46	100%	3	100%
2	9:42	100%	5	100%
3	18:08	100%	3	100%
4	4:46	100%	4	100%
5	27:08	100%	4	100%
6	5:51	100%	5	100%
7				
8	8:14	100%	5	100%

Fonte: elaborado pelo autor (2017)

C.2.2 Cartões

As Tabela 7, Tabela 8, Tabela 9, Tabela 10, Tabela 11, Tabela 12, Tabela 13, Tabela 14, Tabela 15, Tabela 16, Tabela 17, Tabela 18, Tabela 19 e Tabela 20 apresentam o resultado do *card sorting* por cartão.

C.2.3 Matriz de rastreabilidade

A Figura 27 exibe o resultado do *card sorting* utilizando uma matriz de similaridade.

[1] https://www.optimalworkshop.com

Tabela 7 – Cartão 1

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
A interface do software aproveita a experiência do usuário na utilização de outras ferramentas de modelagem	6	Experiência de Usuário	1	1
		Modelagem de Software	1	1
		Personalização	2	1
		Princípios de usabilidade complementares e opcionais	1	1
		UX	1	1
		Usabilidade	2	2

Fonte: elaborado pelo autor (2017)

Tabela 8 – Cartão 2

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
A interface do software exibe apenas as funcionalidade necessárias, evitando o excesso de informações e opções	6	Comunicação	1	1
		Experiência de Usuário	2	1
		FeedBack	2	1
		Princípios de usabilidade altamente desejáveis	2	1
		Recursos de Usabilidade	3	1
		Usabilidade	6	2

Fonte: elaborado pelo autor (2017)

Tabela 9 – Cartão 3

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
A partir de um artefato eu posso rastrear o seus requisitos	6	Navegabilidade	1	1
		Rastreabilidade	2	2
		Funcionalidades	3	1
		vizibilidade	3	1
		Execução de Tarefas	4	1
		Funcionalidades facilitadoras altamente desejáveis	4	1

Fonte: elaborado pelo autor (2017)

Tabela 10 – Cartão 4

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Ao clicar em uma classe no diagrama de classes eu posso navegar para os artefatos aos quais aquela classe está relacionada	6	Rastreabilidade	1	2
		Funcionalidades	1	1
		Navegabilidade	2	1
		vizibilidade	2	1
		Funcionalidades facilitadoras altamente desejáveis	3	1
		Modelagem de Software	4	1

Fonte: elaborado pelo autor (2017)

Tabela 11 – Cartão 5

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Ao errar a execução de uma tarefa a interface oferece opções de desfazer o erro facilmente	7	Princípios de usabilidade altamente desejáveis	1	1
		Controle	3	1
		Execução de Tarefas	3	1
		Comunicação	4	1
		Feedback ao Usuário	4	1
		FeedBack	5	1
		Usabilidade	7	1

Fonte: elaborado pelo autor (2017)

Tabela 12 – Cartão 6

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Ao executar tarefas no software mensagens de erro, de sucesso e de ajuda são exibidas	6	Execução de Tarefas	2	1
		Feedback ao Usuário	2	1
		FeedBack	2	2
		Comunicação	3	1
		Princípios de usabilidade altamente desejáveis	3	1
		Estilos de Interface	4	1

Fonte: elaborado pelo autor (2017)

Tabela 13 – Cartão 7

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Ao implementar um diagrama de maquina de estados eu posso simular a sua execução	6	Requisitos?	1	1
		Simulação	1	1
		vizibilidade	1	1
		Funcionalidades	1	2
		Funcionalidades facilitadoras complementares e opcionais	2	1
		Modelagem de Software	5	1

Fonte: elaborado pelo autor (2017)

Tabela 14 – Cartão 8

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Ao passar o mouse por cima dos ícones o software exibe uma mensagem com uma descrição da funcionalidade	6	Funcionalidades facilitadoras altamente desejáveis	1	1
		Recursos de Usabilidade	1	1
		Comunicação	2	1
		Estilos de Interface	3	1
		Usabilidade	3	1
		FeedBack	4	2

Fonte: elaborado pelo autor (2017)

Tabela 15 – Cartão 9

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Eu posso acessar rápido e claramente as informações dos artefatos	7	Feedback	1	1
		Feedback ao Usuário	1	1
		Usabilidade	2	1
		Modelagem de Software	3	1
		Comunicação	5	1
		Princípios de usabilidade altamente desejáveis	5	1
		UX	6	1

Fonte: elaborado pelo autor (2017)

Tabela 16 – Cartão 10

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Eu posso prototipar interfaces arrastando e soltando elementos	7	Estilos de Interface	1	1
		Interface com Usuário	1	1
		Funcionalidades	2	1
		Funcionalidades facilitadoras altamente desejáveis	2	1
		Modelagem	2	1
		Requisitos?	2	1
		UX	4	1

Fonte: elaborado pelo autor (2017)

Tabela 17 – Cartão 11

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Existem botões que me permitem controlar a execução na janela de Saída	7	Execução de Tarefas	1	1
		Controle	2	1
		Estilos de Interface	2	1
		Itens que não sei classificar	2	1
		Simulação	3	1
		Usabilidade	4	1
		UX	5	1

Fonte: elaborado pelo autor (2017)

Tabela 18 – Cartão 12

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
Existem informações sobre o que está acontecendo com o software durante a sua utilização (status de execução)	6	Usabilidade	1	1
		FeedBack	2	2
		Recursos de Usabilidade	2	1
		Simulação	2	1
		Feedback ao Usuário	3	1
		Princípios de usabilidade altamente desejáveis	4	1

Fonte: elaborado pelo autor (2017)

Tabela 19 – Cartão 13

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
O sistema permite personalizar a interface de trabalho	6	Controle	1	1
		Funcionalidades facilitadoras complementares e opcionais	1	1
		Personalização	1	1
		UX	2	1
		Usabilidade	3	2
		Recursos de Usabilidade	4	1

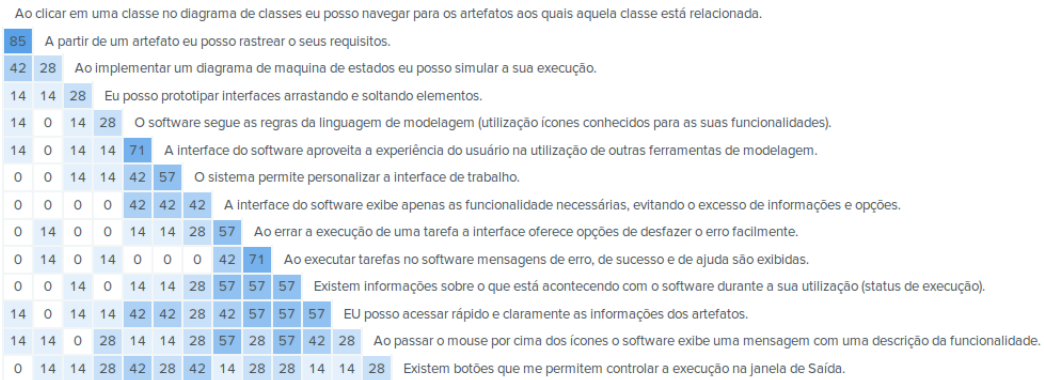
Fonte: elaborado pelo autor (2017)

Tabela 20 – Cartão 14

Cartão	**Categorizações**	**Categorias**	**Posição média**	**Frequência**
O software segue as regras da linguagem de modelagem (utilização ícones conhecidos para as suas funcionalidades)	6	Itens que não sei classificar	1	1
		Modelagem	1	1
		Modelagem de Software	2	1
		Experiência de Usuário	3	1
		UX	3	1
		Usabilidade	4	2

Fonte: elaborado pelo autor (2017)

Figura 27 – Matriz de similaridade



Fonte: elaborado pelo autor (2017)