UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

JORGE DIEGO DE SOUSA SILVA

ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA NO CONTEXTO DE ENGENHARIA REVERSA

RUSSAS

2018

JORGE DIEGO DE SOUSA SILVA

ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA NO CONTEXTO DE ENGENHARIA REVERSA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Profa. Dra. Anna Beatriz dos Santos Marques

RUSSAS

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58e Silva, Jorge Diego de Sousa.
Estudo comparativo de ferramentas de análise estática no contexto de engenharia reversa : Estudo comparativo / Jorge Diego de Sousa Silva. – 2018.

68 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2018.
Orientação: Profa. Dra. Anna Beatriz dos Santos Marques.

1. Engenharia Reversa. 2. Manutenção de Software. 3. Experimento. 4. Ferramentas CASE. I. Título.

CDD 005.1

JORGE DIEGO DE SOUSA SILVA

ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA NO CONTEXTO DE ENGENHARIA REVERSA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Profa. Dra. Anna Beatriz dos Santos Marques (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Dmontier Pinheiro Aragão Jr.
Universidade Federal do Ceará (UFC)

Profa. Dra. Marília Soares Mendes
Universidade Federal do Ceará (UFC)

RESUMO

A manutenção de *software* é uma das fases mais importantes do ciclo de vida de *software*, pois após o sistema ser entregue para uso, erros tendem a surgir, e mudanças tendem à ser solicitadas pelos usuários. No entanto, há várias dificuldades na manutenção de sistemas, principalmente se forem legados, pois geralmente sistemas desse tipo possuem pouca ou nenhuma documentação, dificultando a realização de qualquer mudança. Nesse contexto, a engenharia reversa surgiu com o intuito de facilitar a manutenção de *software*, auxiliando na recuperação de componentes e documentação de sistemas. O objetivo deste trabalho é realizar um estudo comparativo de ferramentas de análise estática, no contexto de engenharia reversa, com base em componentes recuperados, critérios estabelecidos, facilidade de uso, utilidade e intenção comportamental das ferramentas. Para a obtenção de resultados preliminares, foi realizado um estudo em relação aos componentes que as ferramentas conseguiriam recuperar do sistema, foram levados em consideração elementos UML e relacionamentos (associação, composição, agregação, herança, realização e dependência). Os resultados mostram uma semelhança em relação ao número de elementos UML recuperados, já em relação aos relacionamentos, o desempenho das ferramentas foi baixo quando comparado com a quantidade de elementos UML recuperados. Foi realizado um experimento com alunos do curso de Engenharia de *Software* da Universidade Federal do Ceará - Campus Russas, onde os alunos utilizaram duas ferramentas em um sistema, e, em seguida um questionário foi aplicado para avaliar as ferramentas, levando em consideração a facilidade uso, utilidade e intenção comportamental das ferramentas. Tendo em vista o grande número de ferramentas CASE disponíveis, o presente trabalho contribui no apoio à seleção de ferramentas de engenharia reversa eficazes e fáceis de usar, indicadas com base em resultados de experimentação. Como resultados obtemos uma análise estatística dos dados do experimento e uma análise dos componentes e relacionamentos que cada ferramenta conseguiu recuperar, desta forma, auxiliando empresas de desenvolvimento de *software* na escolha por uma ferramenta de análise estática.

Palavras-chave: Manutenção de *Software*. Engenharia Reversa. Experimento. Ferramentas CASE.

ABSTRACT

Software maintenance is one of the most important phases of the software life cycle because after the system is delivered for use, errors tend to arise, and changes tend to be requested by users. However, there are several difficulties in maintaining systems, especially if they are legacy, since generally such systems have little or no documentation, making it difficult to make any changes. In this context, reverse engineering was developed with the purpose of facilitating the maintenance of software, aiding in the recovery of components and documentation of systems. The objective of this work is to perform a comparative study of static analysis tools, in the context of reverse engineering, based on recovered components, established criteria, ease of use, usefulness and behavioral intention of the tools. In order to obtain preliminary results, a study was carried out regarding the components that the tools could retrieve from the system, UML elements and relationships (association, composition, aggregation, inheritance, realization and dependence) were taken into account. The results show a similarity to the number of UML elements retrieved, already in relation to the relationships, the performance of the tools was low when compared to the amount of UML elements recovered. An experiment was carried out with students of the Software Engineering course of the Federal University of Ceará - Russian Campus, where the students used two tools in a system, and then a questionnaire was applied to evaluate the tools, taking into account ease of use , utility and behavioral intent of tools. Given the large number of CASE tools available, this paper contributes to the selection of effective and easy-to-use reverse engineering tools based on experimental results. As results we obtain a statistical analysis of the data of the experiment and an analysis of the components and relationships that each tool was able to recover, in this way, helping software development companies in the choice of a static analysis tool.

Key words: Software maintenance. Reverse engineering. Experiment. CASE tools.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CASE	*Computer-Aided Software Engineering*
JPA	*Java Persistence API*
JSF	*Java Server Faces*
TAM	*Technology Acceptance Model*
TRA	Teoria da Ação Raciocinada
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

A manutenção de *software* é uma das fases mais importantes no ciclo de vida do *software*, pois dependendo da aplicação, esta pode se tornar a mais longa fase do ciclo de vida, tendo em vista que o sistema já está em uso e tenderá a evoluir e a comportar novas mudanças (MARTINS, 2009).

Tendo em vista a importância da manutenção de *software*, várias técnicas para auxiliar na fase de manutenção surgiram ao longo dos anos, dentre elas, a engenharia reversa. A engenharia reversa de *software* é uma das atividades do processo de reengenharia de *software*, que de acordo com Pressman (2009), tem por objetivo desvendar os segredos de como um determinado produto (*software*) de uma empresa concorrente foi desenvolvido, no entanto, ele afirma que na maioria dos casos o sistema que é submetido ao processo de engenharia reversa não é o de uma empresa concorrente, e sim da própria empresa (em geral, feito há muitos anos). Esse tipo de situação é bastante comum quando a empresa em questão faz uso de sistemas legados, ou seja, sistemas que foram desenvolvidos há muitos anos e que possuem mínima ou nenhuma documentação, tornando-se difícil corrigir erros que eventualmente surgem durante o uso do sistema.

"As ferramentas de engenharia reversa extraem informações do projeto de dados, da arquitetura e procedural com base em um programa existente."(PRESSMAN, 2009, p. 670). Tendo em vista que o código fonte de um sistema geralmente possui milhares de linhas, extrair essas informações de forma manual acaba sendo inviável, tornando-se importante o uso de ferramentas para que essa recuperação seja realizada de forma automatizada.

Cindra, Barcelos e Lisbôa (2011) realizam um estudo comparativo entre quatro ferramentas de análise estática voltadas para a engenharia reversa, nesse estudo os autores aplicam tais ferramentas em um sistema e observam quais componentes cada ferramenta consegue recuperar.

1.1 Justificativa

Independentemente do domínio, complexidade ou tamanho da aplicação, o *software* continuará a evoluir com o tempo, conforme o sistema vai sendo usado, mudanças em seus componentes tendem a surgir, erros tendem a aparecer e novas funcionalidades tendem a serem adicionadas, o que torna a manutenção de *software* fundamental para manter o bom

funcionamento e a qualidade do sistema (PRESSMAN, 2009).

No contexto de engenharia reversa, é praticamente inviável trabalhar com a recuperação de componentes de sistema de forma manual, pois existem sistemas com milhares, milhões e até bilhões de linhas de código, sendo necessário o uso de ferramentas para automatizar esse processo de recuperação. Tais ferramentas são denominadas ferramentas *Computer-Aided Software Engineering* (CASE), que têm o intuito de apoiar as atividades de engenharia de *software*, dentre essas ferramentas estão as voltadas para a engenharia reversa, que serão as utilizadas no presente trabalho.

Uma das principais vantagens da realização da engenharia reversa utilizando ferramentas de análise estática é o fato de não precisar colocar o programa em execução, ou seja, as atividades podem ser realizadas somente com o código fonte. No entanto, várias ferramentas fazem esse trabalho, o que acaba tornando difícil escolher uma que melhor se adéque ao seu sistema, e nesse contexto, é de suma importância realizar a melhor escolha, para que as atividades de engenharia reversa possam ser realizadas com sucesso.

Outro benefício de utilizar ferramentas para a engenharia reversa, é a redução de custos, pois as atividades serão feitas de forma automática, não manual. Os resultados deste trabalho são de interesse para empresas que desenvolvem *software* e podem ter dificuldades para selecionar uma ferramenta adequada para este fim.

Embora Cindra, Barcelos e Lisbôa (2011) realizem um estudo comparativo entre ferramentas de engenharia reversa, tal estudo deixa de avaliar fatores importantes como a usabilidade da ferramenta no ponto de vista dos usuários, e, a própria equipe realiza a análise das ferramentas, o que pode acabar tornando o mesmo tendencioso. Terra e Bigonha (2008) também realizam um estudo comparativo envolvendo ferramentas de análise estática, no contexto de Verificação e Validação. O estudo aborda os tipos de erros que cada ferramenta consegue encontrar em um sistema, em seguida os autores analisam o desempenho de cada ferramenta. O presente trabalho se difere dos demais devido a realização de um experimento, onde fatores como a usabilidade e a utilidade das ferramentas serão avaliados utilizando um questionário, seguindo o modelo *Technology Acceptance Model* (TAM), o que torna os dados mais precisos.

O presente trabalho tem como público alvo empresas que trabalham com desenvolvimento de *software*, buscando auxiliar na escolha de uma ferramenta de engenharia reversa que melhor se adéque aos sistemas desenvolvidos e/ou mantidos pela empresa.

1.2 Escopo do trabalho

O trabalho apresenta um estudo comparativo de duas ferramentas de análise estática voltadas para a engenharia reversa, foi realizado um experimento com alunos do curso de Engenharia de *Software* na Universidade Federal do Ceará - Campus Russas.

O escopo do projeto aborda a importância da escolha de uma ferramenta adequada para a realização das atividades de engenharia reversa, tendo em vista que tal escolha pode ser fundamental para o sucesso dessas atividades. O presente trabalho busca responder as seguintes questões:

1. Quais as principais ferramentas que apoiam atividades de engenharia reversa?
2. Qual a importância da engenharia reversa no contexto de manutenção de *software*?
3. Como a realização de um experimento pode ajudar na escolha de uma ferramenta de engenharia reversa?
4. Qual a facilidade de uso e utilidade de ferramentas de análise estática?

1.3 Objetivos

Esse trabalho teve como finalidade realizar um estudo comparativo entre ferramentas de análise estática, no contexto de engenharia reversa, com base na análise de componentes recuperados e critérios estabelecidos. Os objetivos específicos da pesquisa compreendem:

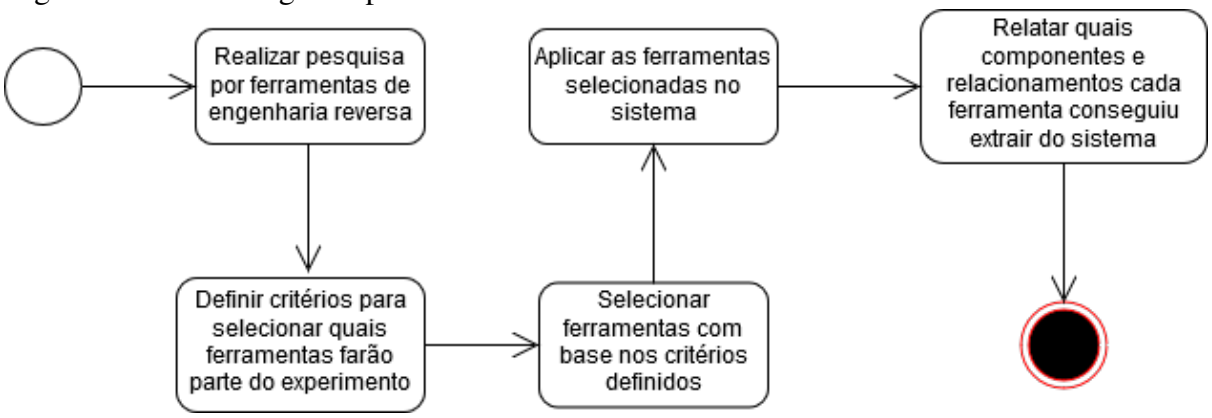
- Auxiliar na escolha de uma ferramenta de engenharia reversa;
- Discutir a importância da realização da engenharia reversa de forma automática;
- Demonstrar a importância da engenharia reversa para a manutenção de *software*;
- Caracterizar as ferramentas em relação à eficácia em termos de componentes recuperados;
- Apresentar dados em relação à facilidade de uso, utilidade e intenção comportamental das ferramentas.

1.4 Metodologia

A metodologia de trabalho consiste na realização de um experimento, que foi realizado com alunos do curso de Engenharia de *Software* na Universidade Federal do Ceará - Campus Russas, e em uma análise feita pelo autor para observar quais componentes e relacionamentos cada ferramenta conseguiria recuperar do sistema. O experimento seguiu as etapas definidas por Wohlin *et al.* (2012). A Figura 1 mostra as atividades realizadas na análise de componentes

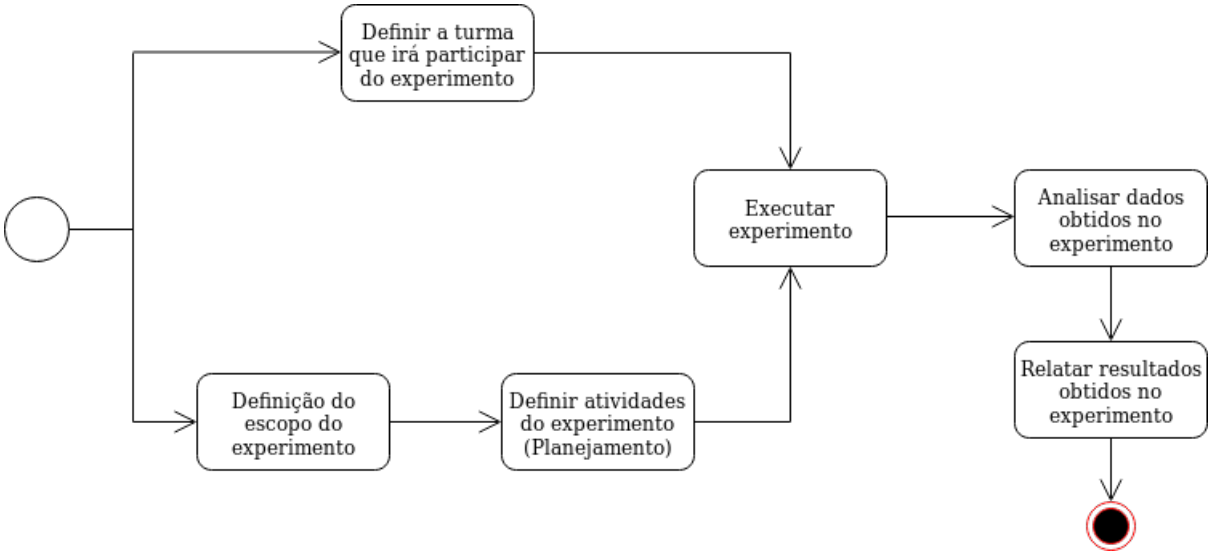
e relacionamentos recuperados pelas ferramentas no sistema. A Figura 2 mostra as atividades realizadas desde o planejamento até a execução e relato dos dados do experimento.

Figura 1 – Metodologia do processo de análise das ferramentas



Fonte: Elaborada pelo autor.

Figura 2 – Metodologia da realização do experimento



Fonte: Adaptada de Wohlin *et al.* (2012).

Como podemos observar na Figura 1, primeiramente foi realizada uma pesquisa por ferramentas de engenharia reversa, nesta pesquisa foram pré-selecionadas 5 ferramentas, em seguida foram definidos critérios para filtrar essas ferramentas, com base nesses critérios foram selecionadas duas ferramentas, a ArgoUML e a Umbrello UML Modeller. Após a seleção, as ferramentas foram aplicadas em um sistema de agendamento de exames de uma clínica médica, ao final foram relatados os resultados obtidos após o uso das ferramentas no sistema.

Na Figura 2 são ilustradas as atividades do experimento, abordando desde a definição do escopo do experimento até a análise dos resultados. Também são ilustradas as atividades

de definição da turma de alunos para participarem do experimento, bem como a execução do experimento. Estas atividades serão detalhadas no Capítulo 4.

1.5 Resultados

Para obter resultados preliminares, as ferramentas ArgoUML e Umbrello UML Modeller foram analisadas com base no número de componentes que cada uma conseguiria recuperar do sistema, foram levados em consideração elementos *Unified Modeling Language* (UML) e relacionamentos. A análise, juntamente com os resultados, serão descritos no Capítulo 5. Resumidamente, a pesquisa teve como resultado, uma análise dos componentes e relacionamentos que as ferramentas conseguiram recuperar do sistema, e, a análise dos dos dados obtidos através da realização do experimento.

1.6 Organização do trabalho

Após essa introdução, o trabalho está organizado como apresentado a seguir. No Capítulo 2 são apresentados alguns conceitos necessários para um melhor entendimento do trabalho, em seguida alguns trabalhos relacionados com a pesquisa serão mostrados no Capítulo 3. A descrição do experimento será apresentada no Capítulo 4 e, em seguida serão apresentados os resultados no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Segundo Pressman:

> Independentemente do domínio da aplicação, tamanho ou complexidade, o software continuará a evoluir com o tempo. As mudanças dirigem esse processo. No âmbito do *software*, ocorrem alterações quando são corrigidos erros, quando há adaptação a um novo ambiente, quando o cliente solicita novas características ou funções e quando a aplicação passa por um processo de reengenharia para proporcionar benefício em um contexto moderno (PRESSMAN, 2009, p. 662).

De forma análoga, Sommerville (2011) afirma que o desenvolvimento de *software* não é interrompido quando o sistema é entregue, mas continua por toda a vida útil do sistema. Depois que o sistema é implantado, para que ele se mantenha útil é inevitável que ocorram mudanças nos negócios e nas expectativas dos usuários, que geram novos requisitos para o *software*. Partes do *software* podem precisar ser modificadas para corrigir erros encontrados na operação para que o *software* se adapte às alterações de sua plataforma de hardware e *software*, bem como para melhorar seu desempenho ou outras características não funcionais.

2.1 Manutenção de *Software*

A manutenção de *software* se refere ao processo geral de mudança em um sistema depois que ele é liberado para uso (SOMMERVILLE, 2011). As alterações feitas no *software* vão de simples correções em nível de código, até mudanças mais extensas já em nível de projeto, na maioria dos casos para acomodar novos requisitos solicitados pelo usuário. Sommerville (2011) classifica a manutenção de *software* em três tipos que serão apresentados nas subseções 2.1.1, 2.1.2 e 2.1.3.

2.1.1 Manutenção Corretiva

Ascanio Junior e Ferreira (2006) relatam que esse tipo de manutenção é uma das mais dispendiosas, no entanto, quando a análise de requisitos ou o projeto da arquitetura não foram realizados com o devido cuidado, os custos podem crescer assustadoramente, o que no pior caso, pode causar um reprojeto do sistema.

Para Sommerville (2011), erros de projeto são mais caros quando comparados com erros de codificação, pois erros de projetos geralmente implicam na reescrita de vários módulos do sistema, bem como erros de requisitos, que na maioria dos casos levam a um extenso reprojeto

no sistema. Analogamente, Lientz e Swanson (1980) definem a manutenção corretiva como sendo as modificações necessárias para corrigir um determinado erro que foi reportado pelo usuário.

Outros autores vão mais a fundo em tal definição, como Otani e Machado (2008), que dividem a manutenção corretiva em dois tipos: a manutenção corretiva planejada, que é uma correção que se faz devido a um acompanhamento preditivo ou até mesmo por uma decisão gerencial, e a manutenção corretiva não planejada, que é quando a correção da falha é feita de maneira totalmente aleatória, tendo como consequência altos custos, pois nem sempre o resultado é satisfatório.

Podemos citar como exemplo um sistema de controle de estoque, suponha que o usuário tente cadastrar um produto e o sistema lance uma exceção, não permitindo o cadastro, isso já caracteriza uma manutenção corretiva.

2.1.2 Manutenção Adaptativa

Esse tipo de modificação no *software* ocorre para acomodá-lo a mudanças em seu ambiente de execução, esse tipo de manutenção inclui o trabalho para migrar o *software* para diferentes plataformas de *softwares* e *hardwares* (compiladores, sistemas operacionais, novas processadores, banco de dados, etc) (CALAZANS; OLIVEIRA, 2005).

Sommerville (2011) denomina esse tipo de manutenção como Adaptação Ambiental, pois ela se torna necessária quando algum aspecto do ambiente no qual o sistema opera (*hardware*, plataforma, *software* de apoio) sofre alguma mudança, visto isso, o sistema de aplicação deve ser modificado de maneira a se adequar a essas mudanças no ambiente.

Para Paduelli (2007), esse tipo de manutenção refere-se a adequar o *software* ao seu ambiente externo. Para ilustrar essa situação, imagine um sistema de controle de estoque, esse sistema possui um método para cadastro de produtos, recebendo como parâmetro o produto a ser cadastrado. Por uma decisão da empresa, deseja-se saber qual funcionário está cadastrando o produto, portanto, o método de cadastro passará a receber como parâmetro, além do produto, o identificador do funcionário que o está cadastrando. Dessa forma será necessário adaptar o sistema à mudança ocorrida no ambiente externo, caracterizando uma manutenção adaptativa.

2.1.3 Manutenção Perfectiva

Moreira (2015) define manutenção perfectiva como sendo a modificação do produto de *software* depois que ele é entregue, para mantê-lo funcionando em um ambiente alterado ou mudado.

De forma análoga, Sommerville (2011) ressalta que a escala de mudanças perfectivas para o *software* é, frequentemente, muito maior do que para os outros tipos de manutenção, e, geralmente ocorrem em resposta às mudanças organizacionais ou de negócios.

De acordo com Calazans e Oliveira, manutenção perfectiva é "a manutenção efetuada para aprimorar o software além dos seus requisitos funcionais originais, a partir do momento em que o usuário identifica novas funcionalidades que trarão benefício ao seu negócio"(CALAZANS; OLIVEIRA, 2005, p. 3). Como exemplo, pode-se citar o pedido do usuário por um novo relatório com informações que até então não podiam ser obtidas do banco de dados.

2.2 Reengenharia de Sistemas

A reengenharia de *software* é uma forma de reuso que permite obter o entendimento do domínio da aplicação, são recuperadas as informações das etapas de análise e projeto, organizando-as de forma coerente e reutilizável (FONTANETE *et al.*, 2003).

Para Chikofsky e Cross (1990), a reengenharia é conhecida também como renovação ou reconstrução, é um exame de um *software*, a fim de reconstruí-lo em uma nova forma e a subsequente implementação dessa forma.

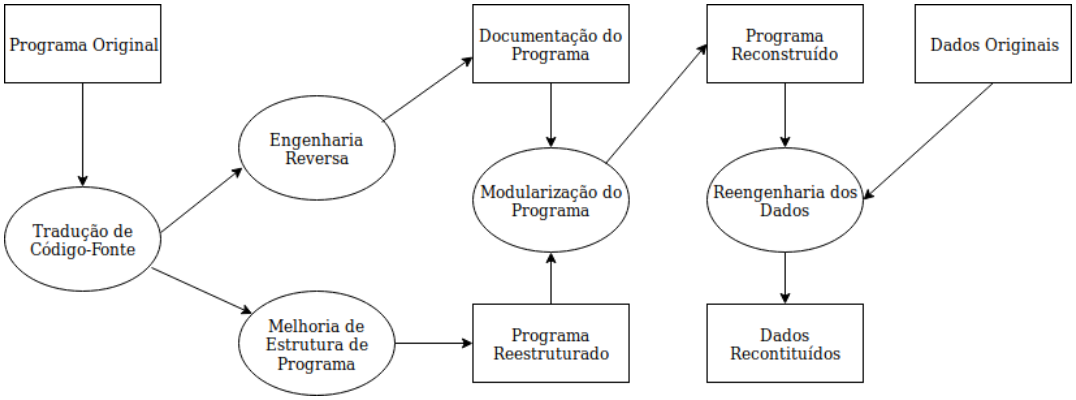
De acordo com Sommerville (2011), a reengenharia de sistema pode envolver a redocumentação do sistema, a mudança na linguagem de programação e modificações e atualizações na estrutura do sistema. A Figura 3 mostra o processo da reengenharia, bem como suas atividades, onde a entrada para esse processo é um sistema legado, e a saída é uma versão melhorada do sistema. As atividades desse processo são:

1. **Tradução de código fonte:** o processo é feito por meio de uma ferramenta, onde o código fonte é traduzido de uma linguagem antiga para uma versão melhorada da mesma, ou até mesmo para outra linguagem.
2. **Engenharia reversa:** o programa é analisado e as informações são extraídas dele, isso ajuda a documentar suas funcionalidades, bem como a organização.
3. **Melhoria de estrutura do programa:** Para facilitar o entendimento da estrutura de

controle do programa, a mesma é analisada e modificada para que se torne mais fácil de ler e entender o programa.

4. **Modularização de programa:** componentes do programa são agrupados, e caso haja redundância, os mesmos serão removidos. Esse é um processo manual e muitas vezes pode ser necessário refatorar a arquitetura do sistema.
5. **Reengenharia de dados:** os dados do programa são alterados para que as mudanças possam ser refletidas no mesmo. Pode ser necessário redefinir o esquema do banco de dados e convertê-lo para a nova estrutura.

Figura 3 – O processo de Reengenharia



Fonte: Sommerville (2011)

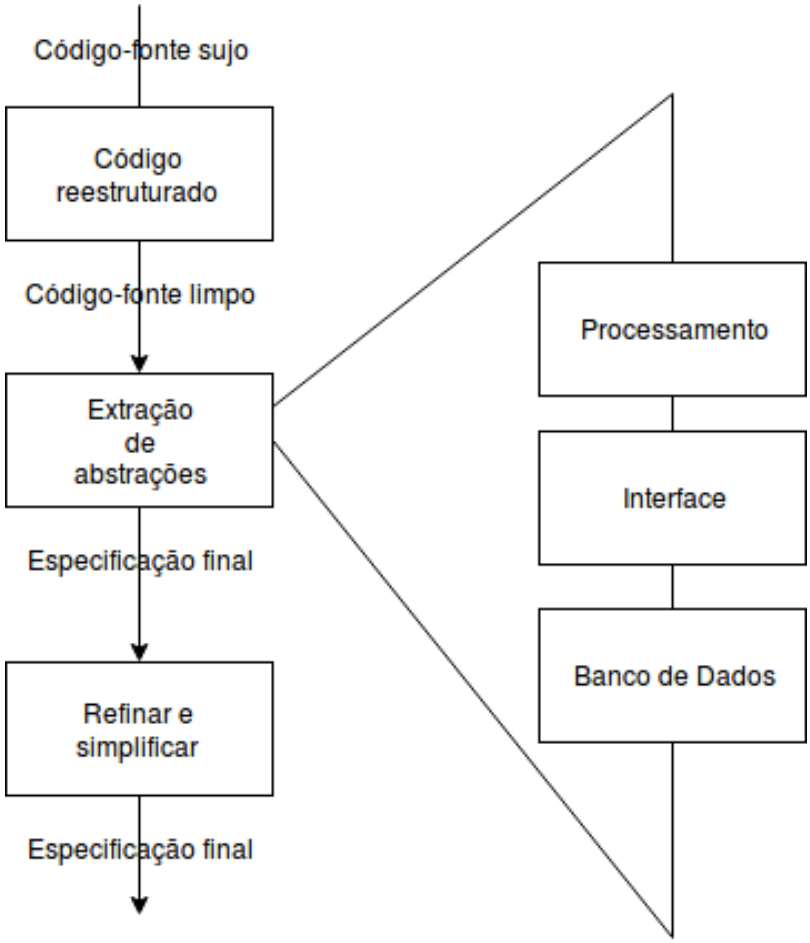
2.3 Engenharia Reversa

Pressman (2009) define engenharia reversa como sendo uma fenda mágica, onde você coloca nessa fenda uma listagem de código não documentada, criada de qualquer jeito, e do outro lado sai toda a documentação do sistema, uma descrição completa do projeto.

De acordo com a Figura 4, observa-se que (PRESSMAN, 2002) definiu o processo de engenharia reversa em 3 etapas:

- **Etapa 1: Código Reestruturado**: nesta etapa você pega um código ruim, e tem a finalidade de reestruturar esse código e torná-lo mais fácil para as etapas seguintes. O código-fonte limpo sairá como artefato para a etapa seguinte.
- **Etapa 2: Extrair Abstrações**: esta etapa consiste em extrair abstrações de fontes como banco de dados, interfaces e processamento.
- **Etapa 3: Refinar e Simplificar**: tendo o código limpo em mãos, o mesmo é refinado e simplificado, finalizando assim o processo de Engenharia Reversa.

Figura 4 – O processo de Engenharia Reversa



Fonte: Pressman (2002)

Engenharia Reversa é umas das etapas da reengenharia de sistemas, nessa etapa o programa é analisado e as informações são extraídas a partir dele, dessa forma ajudando tanto na parte de documentar a organização como na parte funcional do sistema, é um processo completamente automatizado (SOMMERVILLE, 2011).

Para Canhota Junior *et al.* (2005), engenharia reversa é uma atividade que trabalha com um produto já existente, buscando entender seu funcionamento em todas as circunstâncias, além disso, eles também definem algumas técnicas de engenharia reversa com e sem o código fonte. Na subseção 2.3.1 são apresentadas algumas das técnicas de engenharia reversa com o código fonte, pois são de maior relevância para o trabalho.

2.3.1 Técnicas de Engenharia Reversa com o código-fonte

- **Extração das Informações**: a coleta de informações do sistema a ser estudado, é uma atividade que deve ser feita logo no começo, pois essas informações podem ser extraídas de várias fontes: o código fonte, a execução, os dados (por exemplo, em banco de dados),

a documentação (se houver) etc. O tópico a seguir detalha a extração por meio da análise estática, pois é de maior relevância para o trabalho.

- **Código (análise estática)**: é a fonte mais usada dentre as demais. Permite extrair as informações mais básicas do sistema, por exemplo: componentes básicos, conexões entre os componentes, localização de um determinado componente etc. "Parser"é o nome dado as ferramentas usadas para fazer essa análise, dependendo da situação, "pasers"específicos são usados para procurar um determinado tipo de informação. Por exemplo, num programa Pascal, podemos extrair o nome de todas as funções definidas.

2.4 Análise Estática

A análise estática se refere a análise automatizada, que é uma das técnicas utilizadas no processo de Verificação e Validação (TERRA; BIGONHA, 2008). Sommerville (2011) define as técnicas de análise estática como sendo técnicas de verificação de sistema que não envolvem a execução de um programa.

De maneira semelhante, Prado *et al.* (2009) definem a análise estática de código como sendo um método que visa revisar um código fonte em busca de potenciais defeitos e vulnerabilidades, para garantir que os desenvolvedores programem de forma correta e segura.

A estrutura estática de um sistema pode ser recuperada por meio do seu código fonte, sem a necessidade de execução e monitoramento do mesmo, para representar essa estrutura, podem ser utilizados diagramas UML (VERONESE *et al.*, 2002). No contexto de engenharia reversa, realiza-la de forma automática, ou seja, utilizando ferramentas de análise estática, é essencial, tendo em vista que sistemas podem possuir até bilhões de linhas de código.

O termo análise estática diz respeito à qualquer processo de análise de código sem executá-lo, além de poder explorar um grande número de cenários hipotéticos sem ter que passar por todos os cálculos, já que o *software* não está sendo executado (CHESS; WEST, 2007).

3 TRABALHOS RELACIONADOS

Para Steinmacher *et al.* (2006), a obsolescência de sistema provoca a necessidade de alteração no mesmo, para adaptá-lo a novas tecnologias, e, principalmente para atender as exigências do negócio ou dos usuários. Eles também ressaltam que, na maioria dos casos a documentação está desatualizada ou é inexistente, portanto, todo o conhecimento que se tem sobre o sistema estar em uma linguagem de programação, estruturas de dados etc., o que se faz necessária uma forma automatizada para extrair essas informações. Nesse contexto, vários trabalhos apresentam ferramentas para realizarem esse trabalho, alguns mostram apenas a ferramenta e sua estrutura, outros vão mais a fundo e comparam as ferramentas, com o intuito de ajudar na escolha de uma que melhor se adéque ao seu sistema.

Cindra, Barcelos e Lisbôa (2011) realizam uma pesquisa sobre ferramentas de análise estática no contexto de engenharia reversa. A pesquisa é realizada utilizando quatro ferramentas, aplicando-as em um sistema e observando qual ferramenta consegue recuperar o maior número de componentes do sistema. A equipe definiu uma pontuação para cada componente, e no final, era relatada qual ferramenta atingiu a maior pontuação, com base no peso de cada componente recuperado. O trabalho concluiu que não existem ferramentas de código aberto e/ou livres de análise estática que proporcionem condições para o trabalho de engenharia reversa de forma simplificada, e que seria necessário o uso de mais de uma ferramenta para uma maior eficácia.

Um estudo realizado por Veronese *et al.* (2002) apresenta uma ferramenta de engenharia reversa que recupera modelos UML a partir de código fonte em Java. A ferramenta proposta pelo trabalho encontra-se acoplada a um ambiente de desenvolvimento de *software* chamado Odyssey, que procura fornecer suporte ao desenvolvimento de *software* baseado em modelos de domínio. É mostrada com detalhes a arquitetura da ferramenta, sendo descrito todos os seus componentes. A entrada para a ferramenta é o código fonte em Java, e a saída são informações sobre a estrutura de classes, pacotes e seus relacionamentos. O trabalho concluiu que o auxílio da ferramenta proposta diminui o tempo envolvido na etapa de compreensão, que antecede a correção, evolução, adaptação e/ou a reutilização do programa estudado, isso do ponto de vista do programador/projetista.

Freitas Neto e Mendonça (2006) apresentam um ambiente chamado RedoX-UML, que compreende o processo e uma ferramenta, destinado a auxiliar na redocumentação de aplicações legadas Cobol, utilizando a notação UML 2.0. O trabalho tem como motivação diminuir os custos de manutenção de *software* do Banco ALFA (nome fictício), que também

será objeto do estudo de caso para validar o processo. A equipe teve como ponto de partida para o desenvolvimento do RedoX-UML um outro ambiente chamado RefaX. A arquitetura do ambiente é apresentada, bem como o mapeamento da linguagem cobol para a UML e as tecnologias utilizadas no processo. O trabalho apresenta a conclusão de que o projeto RedoX-UML vem atingindo seus objetivos e já agregou valor ao processo de manutenção do Banco ALFA.

Terra e Bigonha (2008) realizaram um estudo comparativo entre ferramentas de análise estática de códigos Java, voltadas para inspeções e teste de *software*. Neste estudo são avaliadas as ferramentas *findbugs, checkstyle, PMD* e *Klcockwork Developer*. A análise é realizada levando em consideração os erros mais cometidos em projetos de *software*, para cada tipo de erro é realizado um experimento para verificar se as ferramentas alertam sobre o erro em questão. Em seguida uma tabela com todas as ferramentas e a quantidade de erros reportados é mostrada. De acordo com os autores, a ferramenta *Klcockwork Developer* foi considerada a melhor do experimento, reportando erros em todos os estudos de caso.

No estudo realizado por Cindra, Barcelos e Lisbôa (2011), a própria equipe utiliza as ferramentas, e levam em consideração somente os componentes recuperados, deixando de lado fatores importantes como a usabilidade da ferramenta, por exemplo. Os demais trabalhos apresentados nessa seção apresentam apenas ferramentas e ambientes para a engenharia reversa, mas não realizam nenhum experimento, com exceção de Terra e Bigonha (2008), onde um estudo comparativo é realizado, no entanto, as ferramentas não são voltadas para a engenharia reversa. Nos demais trabalhos, a equipe usa e avalia tais ferramentas/ambientes, diferentemente da presente pesquisa, onde grupos de usuários utilizaram as ferramentas, foi feita a caracterização desses usuários por meio de um questionário, e o experimento foi feito em um ambiente controlado. O que difere o presente trabalho dos demais apresentados, é o fato de que as ferramentas foram avaliadas por meio de um experimento, e, não serão avaliados somente a quantidade de componentes recuperados por cada ferramenta, mas também sua facilidade de uso, intenção comportamental e a usabilidade.

4 O EXPERIMENTO

Neste capítulo será mostrado o planejamento do experimento, bem como as atividades que foram realizadas durante a execução, até a análise dos resultados.

4.1 Pesquisa por Ferramentas

Inicialmente foi realizada a busca por ferramentas de engenharia reversa, para o início da pesquisa, foram tomadas como base as ferramentas utilizadas por Cindra, Barcelos e Lisbôa (2011), por serem as mais conhecidas. Em seguida, foram definidos critérios para selecionar quais dessas ferramentas iriam fazer parte do experimento, experimento esse que será melhor descrito em todo o Capítulo 4.

Foram analisados artigos e foram feitas buscas em fóruns relacionados ao tema, foram aplicados os critérios para selecionar as ferramentas que irão fazer parte do projeto. O critério inicial para adicionar as ferramentas ao conjunto inicial era somente se elas eram capazes de realizar a engenharia reversa, seguindo esse critério, foram pré-selecionadas cinco ferramentas que serão apresentadas nas subseções 4.1.1 a 4.1.5.

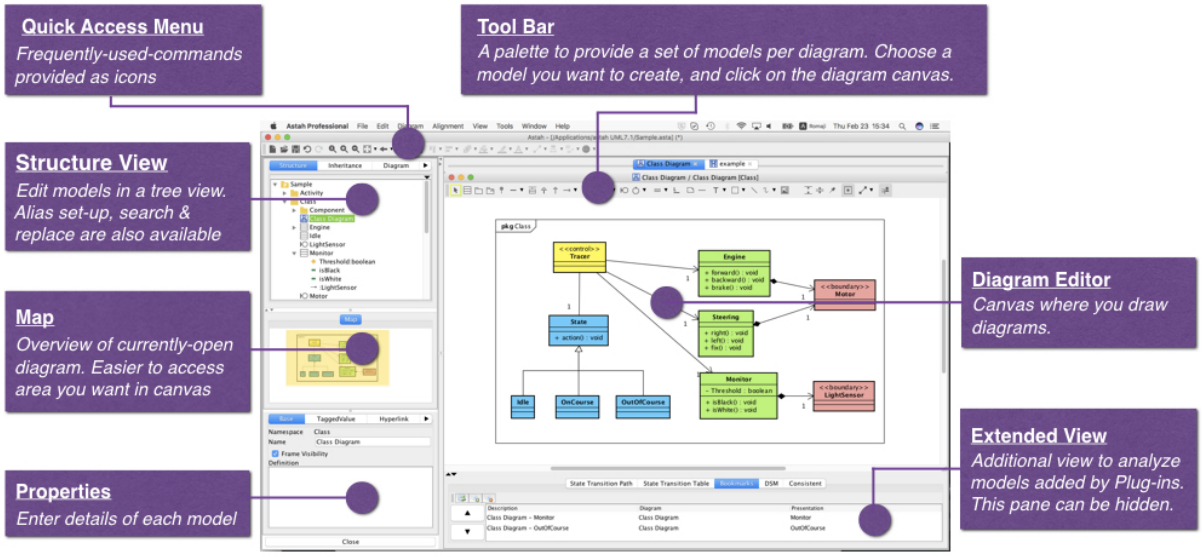
4.1.1 Astah Professional

Anteriormente denominado Jude, um acrônimo para *Java and UML Developers Environment* (Ambiente para Desenvolvedores UML e Java), a ferramenta Astah[1] foi criada pela empresa japonesa Change Vision. A ferramenta possui uma versão gratuita e uma versão paga, porém a versão paga pode ser obtida por estudantes, necessitando apenas do *e-mail* institucional.

Além disso não é necessária a instalação de nenhum *plugin* para a conversão do código para o diagrama de classe. A ferramenta permite a criação de vários diagramas da UML, além do modelo de Entidade-Relacionamento. O programa é multiplataforma e está disponível para Windows, Mac OS e Linux e necessita ter o Java instalado. No estudo comparativo realizado por Cindra, Barcelos e Lisbôa (2011), a ferramenta Astah Professional foi a que recuperou o maior número de componentes em relação as demais ferramentas analisadas. A Figura 5 mostra uma visão geral da ferramenta.

[1] <http://astah.net/editions/professional>

Figura 5 – Ferramenta Astah Professional



Fonte: <astah.net/>

4.1.2 Netbeans IDE

O Netbeans[2] é um IDE (Ambiente de Desenvolvimento Integrado) de código aberto que foi iniciado em 1996 por dois estudantes tchecos da Universidade de Charles. A IDE fornece suporte para as linguagens Java, JavaScript, HTML5, PHP, C/C++, Groovy, Ruby, dentre outras. Com a instalação do *plugin easyuml* torna-se possível realizar a engenharia reversa a partir do código fonte em Java.

A ferramenta pode ser utilizada nos sistemas operacionais Windows, Linux e Mac OS, a instalação é simples e na maioria dos casos não ocorrem problemas. Cindra, Barcelos e Lisbôa (2011) realizaram um estudo comparativo onde utilizaram quatro ferramentas, dentre elas o Netbeans IDE, que na ocasião, recuperou componentes como classes, interfaces, métodos e atributos. A Figura 6 mostra a tela inicial da ferramenta.

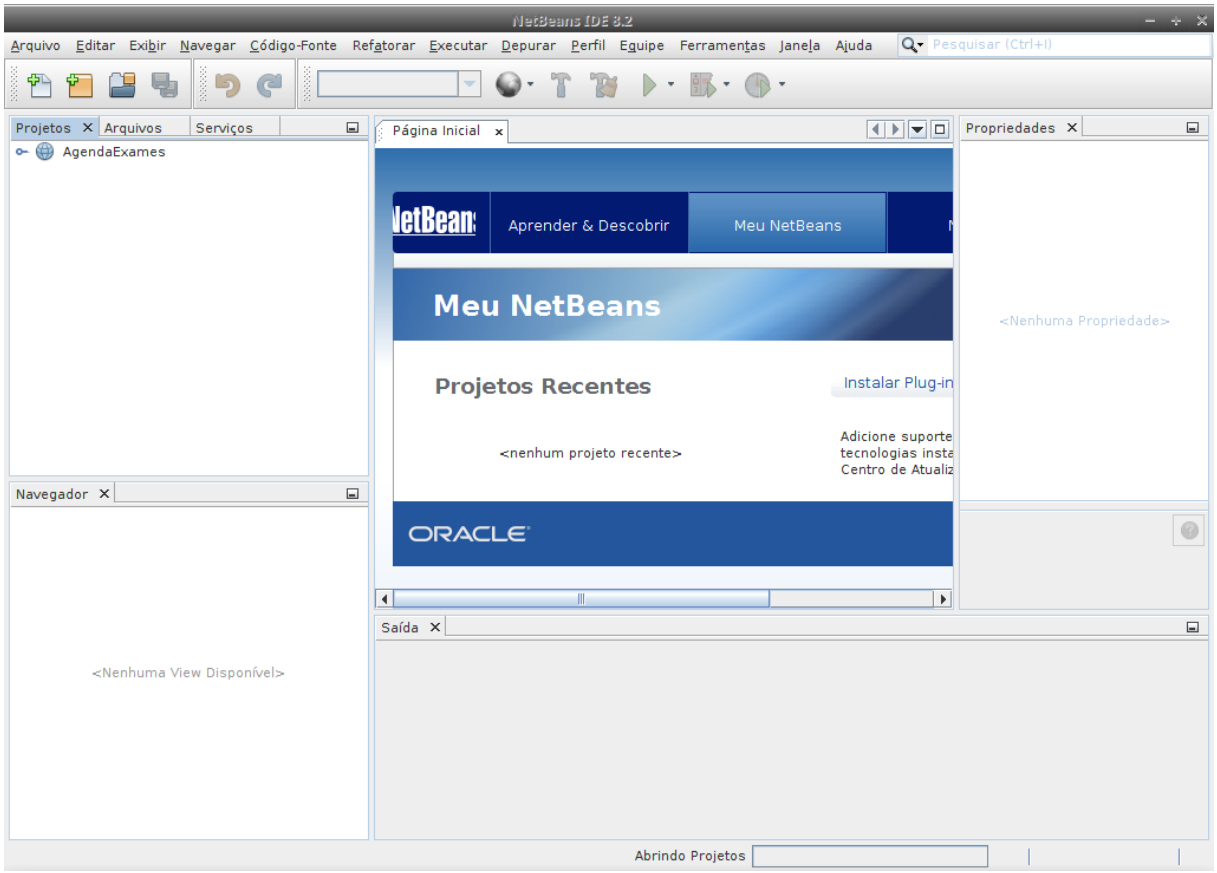
4.1.3 Umbrello UML Modeller

O Umbrello UML Modeller[3] é um programa que trabalha diversos tipos de diagramas da UML, é um projeto baseado na tecnologia KDE. Além de ser de código aberto, a ferramenta ainda da suporte a várias linguagens, permitindo a conversão de código fonte em diagramas da UML.

A ferramenta vem com o KDE SC, incluído em todas as distribuições Linux e

[2] <https://netbeans.org/>
[3] <https://umbrello.kde.org/>

Figura 6 – Ferramenta Netbeans



Fonte: <https://netbeans.org/>

também disponível através do gerenciador de pacotes do Mac OS e através do instalador para o Windows, ou seja, uma ferramenta totalmente multiplataforma.

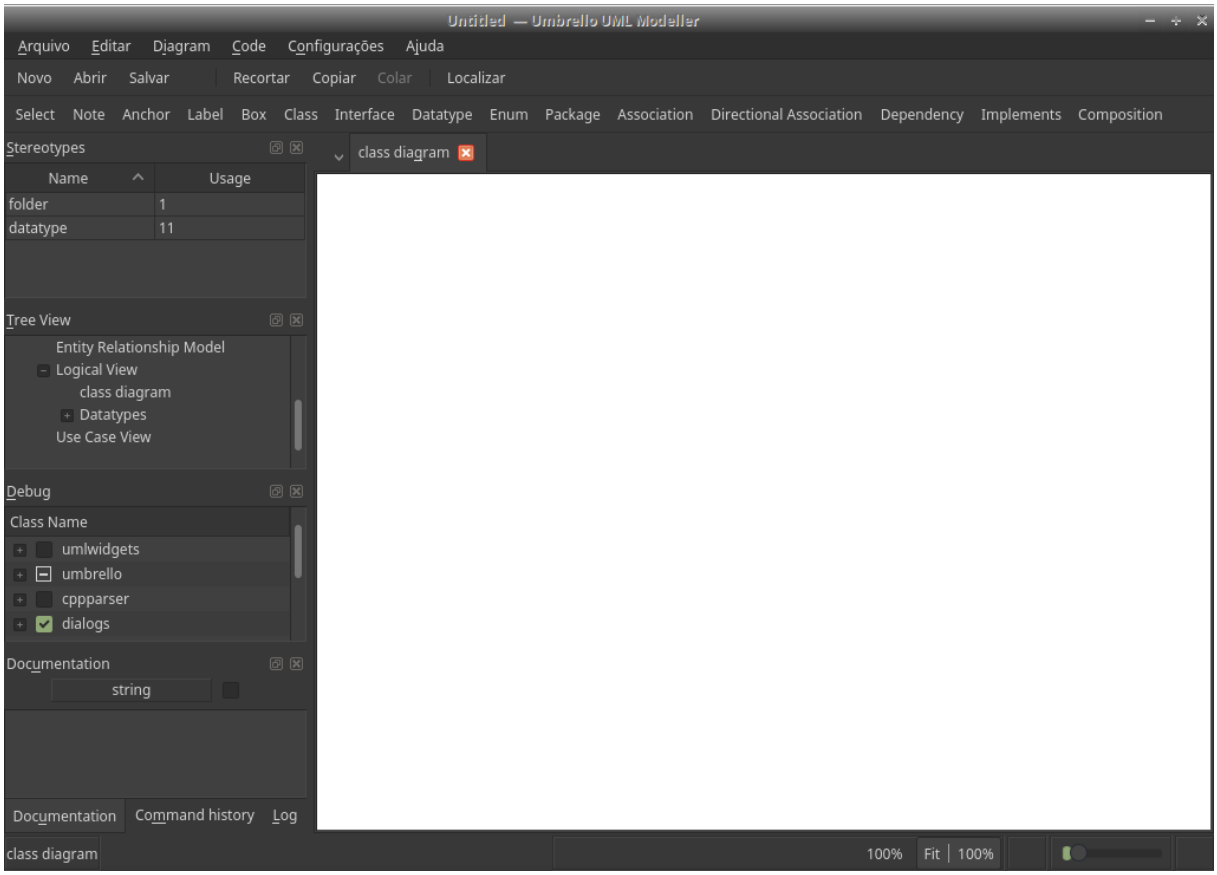
Java, C/C++, Ruby, PHP e Pascal estão entre as várias linguagens suportadas pela ferramenta, sendo possível trabalhar com os diversos diagramas da UML. Não é necessário nenhum *plugin* para realizar a engenharia reversa. A Figura 7 mostra a tela inicial da ferramenta.

4.1.4 ArgoUML

ArgoUML[4] é uma ferramenta para a modelagem de diagramas da UML, ela permite a geração de código em linguagens como Java, PHP, Ruby e C++, a ferramenta é de código livre e foi desenvolvida em Java, o que a faz totalmente multiplataforma. A *Software Development Magazine* realiza uma premiação anual entre ferramentas populares de desenvolvimento de *software* em várias categorias. Em 2003 o ArgoUML foi um dos finalistas na categoria "Ferramentas de Design e Análises". Ele recebeu um prêmio de revelação, derrotando várias ferramentas comerciais. Dentre suas funcionalidades, a exportação dos diagramas é uma delas, a exportação

[4] <http://argouml.tigris.org/>

Figura 7 – Ferramenta Umbrello UML Modeller



Fonte: <https://umbrello.kde.org/>

pode ser feita nos formatos GIF, PNG, PS, EPS, PGML e SVG. Não é necessário nenhum *plugin* para realizar a engenharia reversa. A Figura 9 mostra a tela inicial da ferramenta.

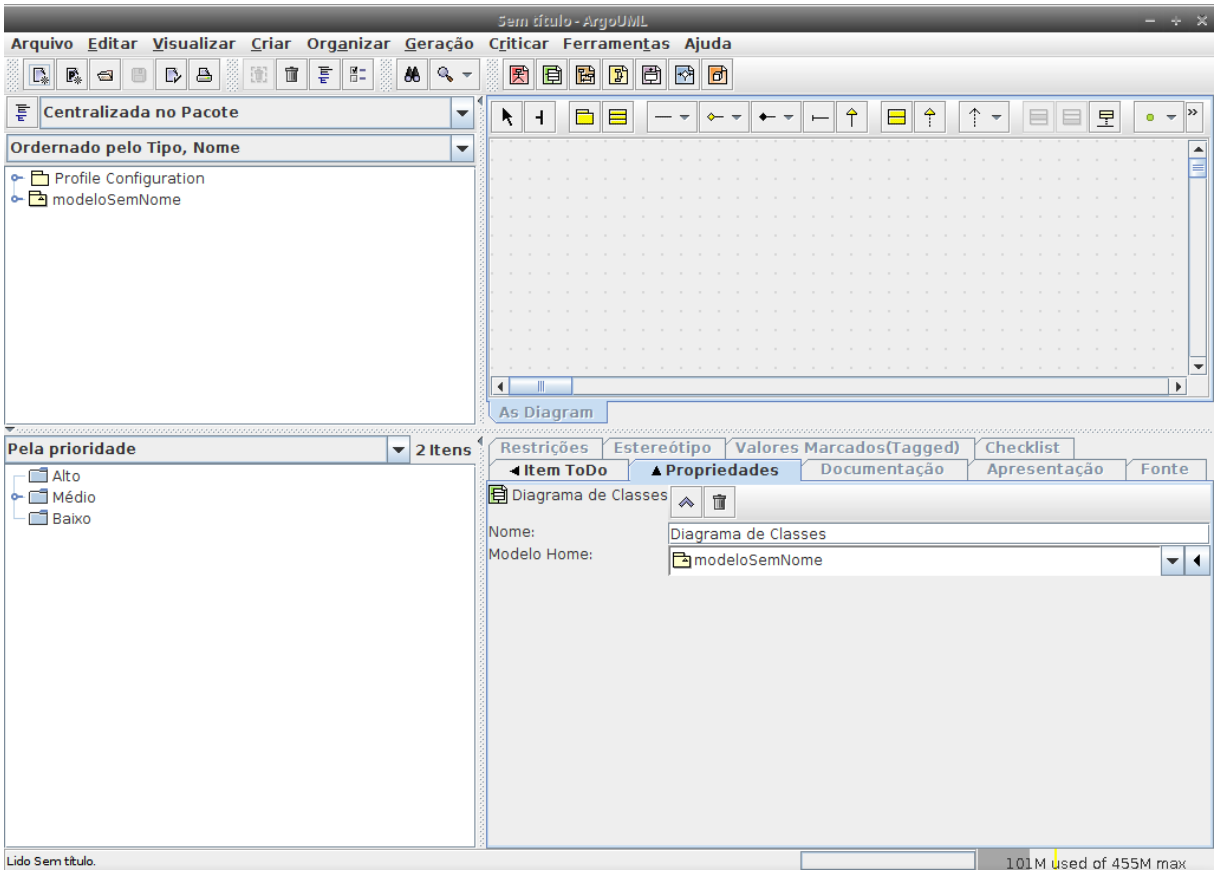
4.1.5 StarUML

StarUML[5] é uma ferramenta para modelagem de diagramas da UML, a ferramenta é paga e permite o uso gratuito somente por alguns dias, sendo necessário comprá-la após esse período. Para realizar a engenharia reversa é necessário instalar um *plugin*, o *download* é feito pela própria ferramenta. As linguagens suportadas pela ferramenta são Java, C# e C++.

A ferramenta está disponível para Linux, Windows e Mac OS, permite a criação de diagramas UML a partir do código fonte em Java. A ferramenta já ultrapassa mais de 5 milhões de *downloads*, de acordo com o site oficial. A versão atual provê total suporte a versão 2.0 da UML.

[5] <http://staruml.io/>

Figura 8 – Ferramenta ArgoUML



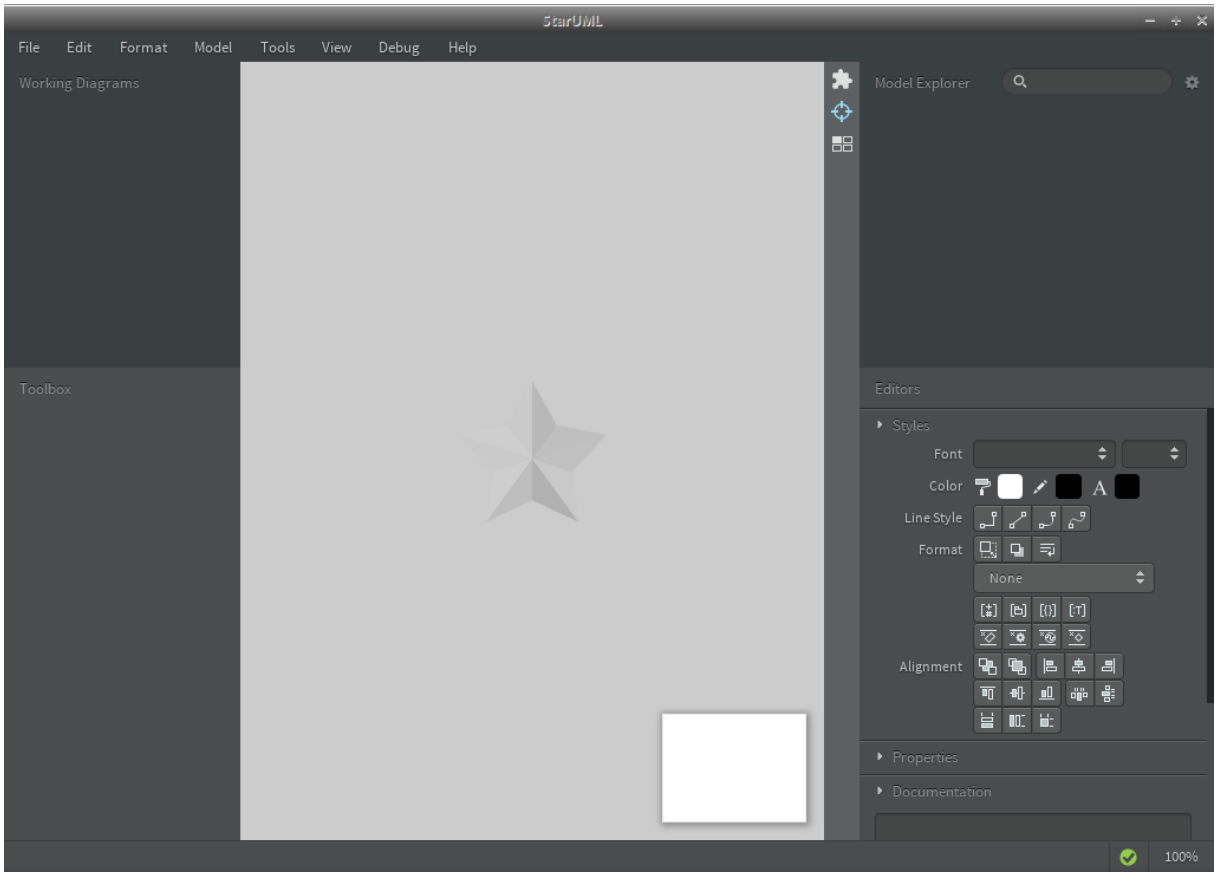
Fonte: <http://argouml-downloads.tigris.org/>

4.2 Análise das Ferramentas Identificadas

Após a seleção inicial das ferramentas, foram definidos critérios para selecionar as ferramentas que iriam fazer parte do experimento. O Quadro 1 mostra a relação entre os critérios e as ferramentas, um "x"é usado para marcar a célula caso a ferramenta atenda ao critério. As duas ferramentas que atingissem o maior número seriam as escolhidas para o experimento. A seguir serão apresentados os critérios definidos, bem como a motivação para a escolha dos mesmos, levando em consideração a disponibilidade das ferramentas, portabilidade, dentre outros.

- **Disponível para uso:** em alguns casos foram encontrados estudos relacionados à ferramentas que não se encontravam mais disponíveis para o uso, portanto essas já seriam descartadas.
- **Portabilidade:** tendo em vista a importância do trabalho para empresas que desenvolvem *software*, é desejável que as ferramentas estejam disponíveis para todas as plataformas, tendo em vista que as empresas usam os mais diversos sistemas operacionais.
- ***Open Source:*** foram dadas preferências as ferramentas *open sources*.

Figura 9 – Ferramenta StarUML



Fonte: <staruml.io/>

- **Suporte à linguagem Java:** as ferramentas serão utilizadas em um sistema desenvolvido em Java, portanto, é de suma importância que as ferramentas suportem a linguagem Java.
- **Não necessita de *plugin*:** algumas ferramentas não possuem a funcionalidade de engenharia reversa, é necessário instalar *plugins* ou até mesmo outros programas, o que pode acabar se tornando uma tarefa um pouco complicada. Portanto, foram priorizadas ferramentas que não necessitam de nenhum *plugin* para realizar as atividades de engenharia reversa.
- **Funcionalidade de Análise Estática:** a engenharia reversa pode ser realizada por meio da execução do programa (análise dinâmica), no entanto, o trabalho consiste na realização da engenharia reversa por meio da análise estática, portanto, as ferramentas deverão possuir tal funcionalidade.

Como pode ser visto no Quadro 1, as ferramentas Umbrello UML Modeller e ArgoUML atenderam os 6 critérios anteriormente definidos, e foram as escolhidas para a realização do experimento.

Quadro 1 – Seleção das ferramentas com base nos critérios

Critérios	**Ferramentas**				
	Astah Professional	Netbeans IDE	ArgoUML	Umbrello UML Modeller	StarUML
Disponível para uso	x	x	x	x	x
Portabilidade	x	x	x	x	x
Suporte a linguagem Java	x	x	x	x	x
Open Source		x	x	x	
Não necessita *plugin*	x		x	x	
Funcionalidade de análise estática	x	x	x	x	x
Total	5	5	6	6	4

Fonte: Elaborado pelo autor.

4.3 O Sistema

As ferramentas selecionadas foram usadas em um sistema para agendamentos de exames em uma clínica médica, desenvolvido pelo Professor Ms. Arthur Nascimento Assunção no período de sua graduação, como trabalho final da disciplina de Desenvolvimento de Aplicações Web.

O sistema é web e foi desenvolvido na linguagem de programação Java, e uma das regras de negócio era que o sistema fosse desenvolvido com base no modelo três camadas, utilizando *Java Server Faces* (JSF) para a camada de aplicação, e, *Java Persistence API* (JPA) e *Hibernate* para o acesso aos dados.

A Figura 10 mostra a tela inicial do sistema, onde o usuário terá a opção de fazer o login, ou então realizar o seu cadastro. No menu superior é possível ver os módulos que o sistema possui (Agenda, Paciente, Exame, Médico e Relatório), tais módulos ficam bloqueados até que o usuário faça o login no sistema.

A Figura 11 mostra o diagrama de classes do sistema em questão, o diagrama foi disponibilizado pelo próprio autor do sistema.

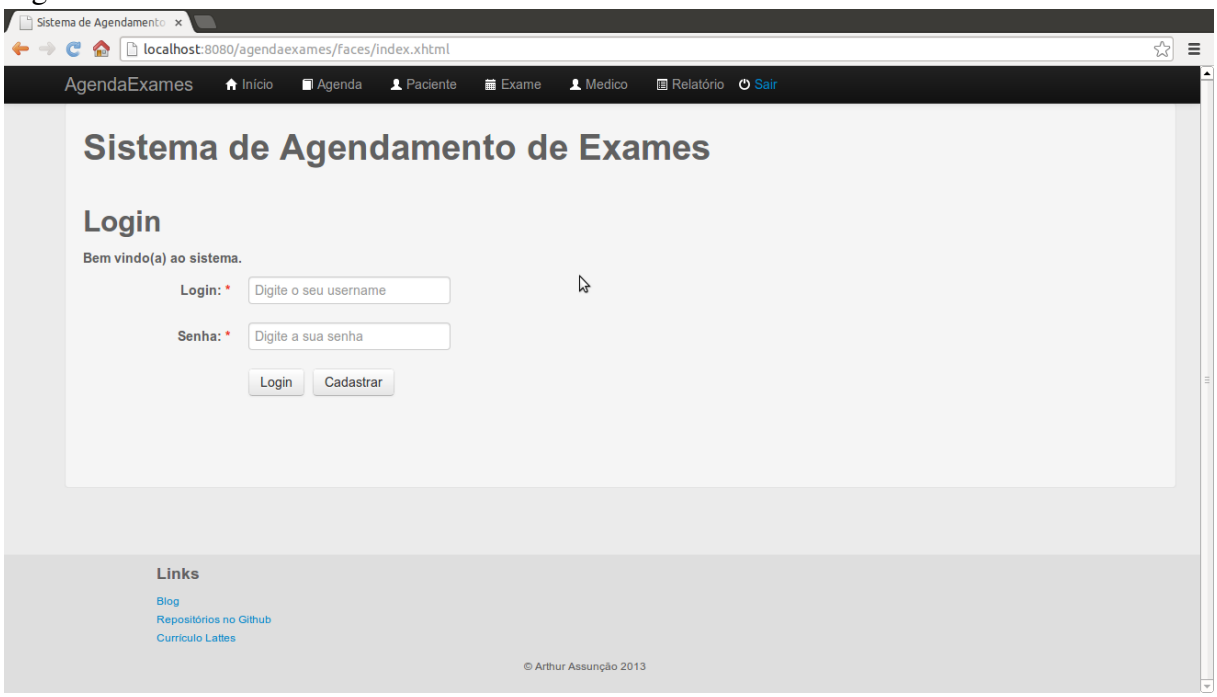
A seguir serão listadas as funcionalidades do sistema:

- Cadastrar médico;
- Cadastrar paciente;
- Marcar consulta;
- Solicitar exame;
- Visualizar agenda;
- Cadastrar agenda.

O sistema[6] está disponível na plataforma GitHub, após o *download*, basta seguir o

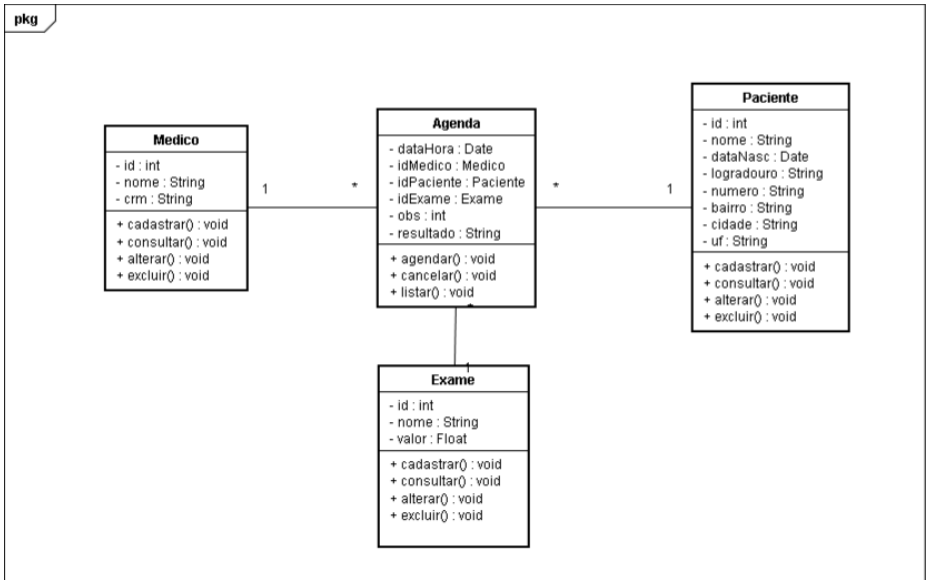
[6] <https://github.com/ArthurAssuncao/AgendaExames>

Figura 10 – Tela inicial do sistema



Fonte: Assunção (2013)

Figura 11 – Diagrama de classes do sistema



Fonte: Assunção (2013)

manual que vem juntamente com o sistema e seguir os passos para a instalação e execução do mesmo.

A motivação para a escolha do sistema partiu do conhecimento do autor em relação ao domínio do sistema, pois em vários trabalos realizados durante o período do curso, o autor sempre optou por sistema de controle de clínica médica, chegando a realizar um levantamento de requisitos em uma clínica da cidade de Russas - CE.

4.4 Planejamento do Experimento

O experimento foi planejado seguindo todas as etapas definidas por Wohlin *et al.* (2012). A seguir serão mostradas as atividades realizadas durante o planejamento do estudo experimental.

4.4.1 Contexto

Segundo Wohlin *et al.* (2012), para alcançar resultados mais gerais em um experimento, é recomendado que ele seja executado em projetos de *software* grandes e com uma equipe profissional. No entanto, conduzir um experimento em um ambiente real pode ser muito arriscado, principalmente se o que estiver sendo experimentado nunca tiver sido testado e possa comprometer o projeto e a equipe, uma alternativa para esse problema é realizar experimento com alunos em projetos acadêmicos, pois além de ser barato, são bem mais fáceis de controlar e geralmente são voltados a um contexto específico (WOHLIN *et al.*, 2012).

O experimento apresentado no presente trabalho foi executado em ambiente acadêmico, os participantes eram de uma turma de Experimentação em Engenharia de Software, da Universidade Federal do Ceará. A disciplina é ofertada no sétimo período do curso de Engenharia de Software, tendo a disciplina de Qualidade de Software como pré requisito. Embora a disciplina seja da grade de Engenharia de Software, alunos de outros cursos também podem cursá-la, no período do experimento, alunos de Engenharia de Software e Ciência da Computação faziam parte da disciplina.

Como já foi dito anteriormente, o experimento foi realizado em uma turma da disciplina de Experimentação em Engenharia de Software, porém, vale ressaltar que não era permitido ao experimento intervir no andamento da disciplina, bem como no rendimento dos alunos. Como é recomendado que toda disciplina tenha aulas práticas, o experimento acabou contribuindo para o andamento da disciplina, pois serviu para por em prática tudo que os alunos viram em sala, aboradando desde o planejamento de um experimento, até sua execução.

4.4.2 Escopo

O escopo do experimento abordou o uso de duas ferramentas de análise estática, voltadas para o contexto da engenharia reversa, onde, as ferramentas foram utilizadas em um sistema para que se pudesse avaliar aspectos de utilidade, facilidade de uso e intenção

comportamental presentes nas ferramentas. O experimento é motivado pela necessidade de identificar a ferramenta de engenharia reversa mais adequada para um determinado contexto.

- **Objeto de Estudo:** os objetos de estudo do experimento são as ferramentas de engenharia reversa.
- **Objetivo:** o objetivo do experimento é avaliar as ferramentas com base em aspectos de utilidade, facilidade de uso e intenção comportamental para auxiliar na escolha de uma ferramenta que melhor se adéque a um determinado contexto.
- **Perspectiva:** no ponto de vista do pesquisador, o mesmo deseja analisar e identificar diferenças na utilidade, facilidade de uso e intenção comportamental das ferramentas, para que possa auxiliar na escolha de uma delas.
- **Foco na Qualidade:** os principais efeitos estudados são aspectos sobre a percepção da utilidade, facilidade de uso e intenção comportamental das ferramentas.
- **Contexto:** o experimento é realizado no contexto de uma turma da disciplina de Experimentação em Engenharia de Software, do curso de Engenharia de Software da Universidade Federal do Ceará.

4.4.3 Formulação de Hipóteses

A base para a análise estatística de um experimento é o teste de hipóteses, os dados coletados durante o experimento são utilizados para rejeitar ou não uma hipótese formalmente declarada (WOHLIN *et al.*, 2012). De acordo com Wohlin *et al.* (2012), na fase de planejamento de um experimento, no mínimo duas hipóteses devem ser formuladas, uma hipótese nula e uma hipótese alternativa:

- **Hipótese Nula:** Uma hipótese nula H_0 afirma que não há tendências subjacentes reais ou padrões na configuração do experimento, é essa hipótese que o pesquisador deseja rejeitar com o maior nível de significância.
- **Hipótese Alternativa:** Uma hipótese alternativa H_a, H_1, etc, é a hipótese a favor de que a hipótese nula seja rejeitada, é usando essa hipótese que o pesquisador irá tentar rejeitar a hipótese nula.

Para o experimento descrito no presente trabalho, foram definidas 6 hipóteses, 3 nulas e 3 alternativas:

- **H_{01}:** não há diferença na percepção da facilidade de uso entre as ferramentas ArgoUML e Umbrello UML Modeller.

- **H_{A1}:** há diferença na percepção da facilidade de uso entre as ferramentas ArgoUML e Umbrello UML Modeller.
- **H_{02}:** não há diferença na percepção da utilidade entre as ferramentas ArgoUML e Umbrello UML Modeller.
- **H_{A2}:** há diferença na percepção da utilidade entre as ferramentas ArgoUML e Umbrello UML Modeller.
- **H_{03}:** não há diferença na intenção comportamental entre as ferramentas ArgoUML e Umbrello UML Modeller.
- **H_{A3}:** há diferença na intenção comportamental entre as ferramentas ArgoUML e Umbrello UML Modeller.

4.4.4 Seleção de Variáveis

Antes de começar qualquer experimento, deve-se definir as variáveis dependentes e independentes, fazer essa escolha geralmente não é fácil, e exige muito conhecimento sobre o domínio em que o experimento está sendo aplicado.

As variáveis independentes de um experimento, são aquelas variáveis que podemos mudar e controlar durante o experimento. No caso das variáveis dependentes, são o efeito que será observado a partir da configuração das variáveis independentes.

As variáveis independentes são as ferramentas e a experiência dos alunos. As variáveis dependentes são a percepção da utilidade, facilidade de uso e intenção comportamental das ferramentas.

4.4.5 Escolha do design experimental

De acordo com Wohlin *et al.* (2012), um design experimental descreve como os testes são organizados e executados, um design pode ser definido formalmente como sendo um conjunto de testes.

Wohlin *et al.* (2012) apresentam quatro tipos de design para experimentos, que segundo eles são os mais utilizados. São eles: um fator com dois tratamentos, um fator com mais de dois tratamentos, dois fatores com dois tratamentos, mais de dois fatores, cada um com dois tratamentos. O presente experimento utilizou o tipo um fator e dois tratamentos, o fator é a ferramenta de análise estática, os tratamentos são Umbrello e ArgoUML.

O princípio utilizado foi o de balanceamento, pois tínhamos dois grupos e equilibra-

mos os mesmos em números de participantes e também em relação ao tempo de experiência deles na indústria.

4.4.6 Seleção dos Participantes

Os participantes são escolhidos com base na conveniência, ou seja, os participantes são os alunos da disciplina de Experimentação em Engenharia de Software, que são uma amostra de usuários que utilizam as ferramentas.

Os grupos selecionados para o experimento realizaram algumas atividades de engenharia reversa, após a realização dessas atividades, um questionário foi aplicado. Os questionários estão disponíveis nos apêndices B e C respectivamente.

4.4.7 Caracterização do perfil dos participantes do experimento

Foi aplicado um questionário para caracterizar o perfil dos participantes do experimento, tal questionário buscou obter informações sobre o nível de conhecimento dos participantes em relação à manutenção de *software*, desenvolvimento de *software* e engenharia reversa. Um outro objetivo do questionário foi descobrir se os participantes já tiveram alguma experiência na indústria, para que os grupos fiquem balanceados na hora da divisão. O questionário está disponível no apêndice A.

Caso o participante já tenha tido alguma experiência na indústria, ele terá uma visão diferente de um que não teve, em relação ao uso das ferramentas. Isto torna importante a aplicação desse questionário para caracterizar o perfil dos participantes.

4.4.8 Balanceamento dos Grupos do Experimento

Com base nas informações coletadas através do Questionário de Caracterização dos Participantes, os grupos foram balanceados para que as equipes ficassem bem divididas, por exemplo, se dois participantes têm cada um 3 anos de experiência na indústria, eles serão colocados um para cada grupo, pois assim evitará de que um grupo totalmente experiente seja formado, assim como um grupo totalmente inexperiente.

No questionário, foram levados em consideração o tempo de experiência na indústria, conhecimento em manutenção de *software* e conhecimento em engenharia reversa. Dessa forma, foi garantido que os grupos ficassem balanceados e nenhum grupo tivesse vantagem sobre o

outro, embora não houvesse nenhum tipo de competição.

4.4.9 Preparação

Agora serão mostradas as atividades que ocorreram na fase de preparação do experimento, bem como uma descrição detalhada de cada uma delas. Os participantes do experimento não sabiam o que de fato seria estudado, o pesquisador informou que eles iriam utilizar uma ferramenta de engenharia reversa, mas o que de fato seria estudado não foi passado pra eles, ou seja, eles não tinham conhecimento da hipótese formulada anteriormente. Foi garantido o anonimato de todos os alunos participantes.

4.4.9.1 Elaboração dos artefatos roteiro e cenário

Para cada uma das ferramentas foi elaborado um roteiro e um cenário, que foram entregues na hora do experimento. O cenário contém uma breve descrição do sistema, juntamente com seus requisitos funcionais e um breve contexto fictício que envolvia a suposição de que eles estavam sendo contratados por uma empresa para realizar modificações em um sistema de clínica médica. Os cenários estão disponíveis nos Apêndices F e G.

O roteiro contém as atividades que os participantes devem realizar e algumas dicas para a realização das mesmas. Primeiramente foram definidos alguns procedimentos iniciais, pois antes que eles pudessem começar a realizar as tarefas era necessário importar o código fonte do sistema para as ferramentas. Com o código fonte importado, os participantes puderam começar a realizar as atividades descritas nos roteiros. Os roteiros estão disponíveis nos Apêndices D e E.

4.4.9.2 Definição das atividades

A seguir são mostradas as atividades necessárias para que os participantes concluíssem o experimento. As atividades foram pensadas tendo em vista uma das limitações das ferramentas, ambas permitem somente a extração do diagrama de classes, outros diagramas não são possíveis de extrair.

- **Atividades de engenharia reversa:**

1. **Extrair diagrama de classes:** os participantes deverão importar o projeto do sistema, em seguida tentarão extrair o diagrama de classes do módulo de cadastro de médicos, sistema em questão.

2. **Atualizar diagrama de classes:** será fornecida aos participantes uma lista com todos os requisitos do sistema, os participantes devem incluir um novo requisito nessa lista, e, após a extração do diagrama de classes, eles devem modificá-lo de maneira que o diagrama passe a atender o requisito que foi incluído.

4.4.9.3 Preparação do Ambiente

O ambiente foi preparado com antecedência, a instalação das ferramentas nos computadores foi solicitada pela orientadora e realizada pela equipe de técnicos de laboratórios da Universidade Federal do Ceará. Em seguida o autor disponibilizou o código fonte do sistema para as máquinas, o código foi disponibilizado em uma pasta na área de trabalho, e para realizar as tarefas do experimento, os participantes deveriam localizar o código e importar para a ferramenta.

Foi definido que cada grupo utilizaria uma ferramenta, os grupos foram divididos no laboratório, um grupo para o lado direito e outro para o lado esquerdo. Após a divisão dos grupos, cada grupo ficou com uma ferramenta para que pudessem realizar as atividades, no entanto, as atividades foram realizadas de forma individual dentro dos grupos.

Depois que foi feita a divisão, foram entregues aos participantes 3 artefatos, o termo de consentimento, o cenário do experimento, e o roteiro do experimento, onde continha as atividades que eles deveriam realizar. No cenário continha as informações necessárias sobre o sistema, para que ficasse claro o contexto sobre o qual eles estavam trabalhando. Após a entrega dos artefatos, o termo de consentimento foi recolhido e o experimento foi iniciado.

4.5 Execução do Experimento

O experimento foi realizado em um período de 2 horas, com 23 participantes, todos os participantes responderam com antecedência o questionário de caracterização do perfil do participante, para que os grupos pudessem ser balanceados anteriormente.

Foi solicitado aos participantes que após a realização das atividades exportassem o diagrama obtido na pasta do projeto, colocando seu nome e a ferramenta que utilizou. No final do experimento o autor coletou os dados, no entanto, alguns participantes faltaram, e outros não salvaram os resultados. Dos 26 esperados, foram coletados resultados de 23 participantes, o restante ou faltaram, ou não salvaram o resultado das tarefas.

Quando algum participante concluía as atividades, lhe era entregue um questionário

para avaliar a ferramenta que ele utilizou. Os questionários foram elaborados seguindo o modelo TAM, que será explicado na próxima seção. Depois que o participante respondia o questionário, o experimento era concluído.

4.6 Avaliação das ferramentas utilizando o modelo TAM

O *Technology Acceptance Model*(TAM), geralmente conhecido como modelo TAM, foi proposto por Davis (1989) como uma adaptação do modelo Teoria da Ação Raciocinada (TRA), porém, segundo Davis (1989), a modificação do TRA ocorreu devido ao fato dele ser tão universal, tal modificação foi feita para criar modelos de aceitação em tecnologia da informação, que é o caso do TAM (SILVA, 2008). De acordo com Davis (1989), o modelo TAM foi projetado com o intuito de compreender a relação casual entre o uso real do computador e as variáveis externas de aceitação dos usuários. As pessoas tendem a usar ou não uma determinada tecnologia com o objetivo de melhorar seu desempenho no trabalho (DAVIS, 1989).

A presente pesquisa utilizou o modelo TAM 3, proposto por Venkatesh e Bala (2008), onde são levados em consideração a facilidade de uso da ferramenta/sistema, utilidade da(o) ferramenta/sistema e a intenção comportamental do usuário ao utilizar a ferramenta/sistema.

O Apêndice B e o Apêndice C mostram os questionários que serão utilizados para avaliar as ferramentas ArgoUML e Umbrello UML Modeller respectivamente. Os questionários foram elaborados seguindo o modelo TAM e foram aplicados após o uso de cada ferramenta pelos grupos do experimento. Esses questionários foram elaborados utilizando também a escala Likert, que é comumente utilizada em pesquisas de opinião questionários.

A escala Likert é um formato de classificação comum para pesquisa, onde os respondentes classificam a qualidade de alto a baixo ou de mau a pior, usando 5 ou 7 níveis (ALLEN; SEAMAN, 2007). A presente pesquisa utilizará 5 níveis para avaliar a facilidade de uso das ferramentas, utilidade e intenção comportamental seguindo a escala Likert juntamente com o modelo TAM 3.

5 RESULTADOS

Os resultados que serão apresentados neste capítulo serão em relação à análise dos componentes recuperados pelas ferramentas e em relação ao experimento que foi realizado. A análise levou em consideração os elementos UML e os relacionamentos que cada ferramenta conseguiu recuperar, como é mostrado nos Quadro 2 e Quadro 3 respectivamente.

As ferramentas foram utilizadas no sistema descrito no Capítulo 4 seção 4.3, rodando em um computador com o processador *Intel Pentium* T2390 1.86 GHz, com 4GB de memória RAM e com sistema operacional Linux Mint.

5.1 Análise dos componentes recuperados

As ferramentas foram avaliadas com base nos elementos e relacionamentos que cada uma recuperou, foram levados em consideração os seguintes elementos UML: pacotes, classes, atributos, métodos, interfaces, componentes e esteriótipos. Além dos componentes, os seguintes relacionamentos foram considerados: associação, composição, agregação, herança/generalização, realização e dependência.

O Quadro 2 mostra os elementos UML recuperados pelas ferramentas, um "x"é marcado na célula caso a ferramenta recupere o componente por meio da engenharia reversa. Já o Quadro 3 mostra os relacionamentos que cada ferramenta conseguiu recuperar, caso isso ocorra, um "x"é marcado na célula correspondente.

Quadro 2 – Elementos UML recuperados pelas ferramentas

	Ferramentas	
Elementos UML	ArgoUML	Umbrello UML Modeller
Pacotes	x	x
Classes	x	x
Atributos	x	x
Métodos	x	x
Interfaces	x	x
Componente	x	
Esteriótipos	x	
Total	7	5

Fonte: Elaborado pelo autor.

Quadro 3 – Relacionamentos recuperados pelas ferramentas

Relacionamentos	**Ferramentas**	
	ArgoUML	Umbrello UML Modeller
Associação		
Composição		x
Agregação		
Herança/Generalização		
Realização	x	x
Dependência		x
Total	1	3

Fonte: Elaborado pelo autor.

5.1.1 Análise dos elementos UML obtidos

O que pode ser destacado em relação aos elementos recuperados pelas ferramentas é o fato de que a ferramenta ArgoUML recuperou todos os componentes que foram analisados, enquanto a ferramenta Umbrello Modeller UML deixou de recuperar os elementos componentes e esteriótipos.

Outro fator a ser observado, é o fato de nenhuma das ferramentas analisadas gerarem o diagrama de classes automaticamente, após a importação do código fonte, é necessário arrastar as classes para um diagrama vazio, formando assim o diagrama com os atributos e relacionamentos recuperados.

Um ponto positivo em relação as duas ferramentas é que ambas permitem a alteração do diagrama após ele ter sido gerado, ou seja, caso precise ser realizada alguma mudança no sistema, essa mudança poderá ser feita no diagrama para que em seguida a engenharia reversa seja novamente realizada, dessa maneira, refletindo a mudança no código fonte.

5.1.2 Análise dos relacionamentos obtidos

Com base nos relacionamentos recuperados, percebeu-se que o desempenho das ferramentas no sistema não foi bom quando comparado com os componentes UML recuperados, como se pode observar no Quadro 3, a ferramenta ArgoUML recuperou apenas um relacionamento em todo o sistema, que foi o de realização, já a ferramenta Umbrello UML Modeller recuperou os relacionamentos de realização, composição e dependência, totalizando três relacionamentos recuperados.

Outro ponto a ser destacado é o fato da ferramenta ArgoUML ter tido um alto desempenho na recuperação de elementos, recuperando os sete que foram avaliados.

A ferramenta Umbrello UML Modeller teve um desempenho parecido com a ferra-

menta ArgoUML em relação ao número de elementos recuperados, recuperando cinco dos sete analisados. Nos relacionamentos a ferramenta Umbrello UML Modeller se saiu melhor que a ferramenta ArgoUML, recuperando relacionamentos como composição e dependência, desta forma se destacando da ferramenta ArgoUML.

Vale ressaltar que estes resultados não significam que a ferramenta não é capaz de recuperar um dado relacionamento, o relacionamento de herança por exemplo, não foi utilizado na implementação do sistema, ou seja, não seria possível que as ferramentas pudessem recuperá-los. Os dados apresentam os elementos e relacionamentos que as ferramentas conseguiram recuperar do sistema utilizado, que é o descrito no Capítulo 4 seção 4.3. O relacionamento de herança por exemplo, que nenhuma das ferramentas encontrou, realmente não está presente no sistema, de acordo com o diagrama de classes mostrado na Figura 11.

5.2 Análise dos dados do experimento

Os dados resultantes da fase de execução do experimento serão analisados nesta seção, temos como objetivo tirar conclusões desses dados. No entanto, para garantir que as conclusões sejam válidas devemos interpretar os dados do experimento (WOHLIN *et al.*, 2012).

Para a análise dos dados foram utilizadas duas ferramentas, a SPSS [1] desenvolvida pela IBM, e o Google Planilhas[2].

5.2.1 Afirmativas dos questionários aplicados

Como já foi dito, após a realização do experimento foi aplicado um questionário para que os participantes pudessem avaliar as ferramentas. O questionário foi elaborado seguindo o modelo TAM 3, que abordam três critérios em sua avaliação.

Serão listadas agora as afirmativas contidas no questionário, para facilitar o entendimento dos resultados da pesquisa. Para cada ferramenta foi aplicado o mesmo questionário, mudando apenas o nome da ferramenta, para listar as afirmações usaremos o nome da ferramenta ArgoUML.

PEOU - Percepção da Facilidade de Uso:

- PEOU1: Minha interação com a ferramenta ArgoUML é clara e compreensível.
- PEOU2: Interagir com a ferramenta ArgoUML não requer muito do meu esforço mental.

[1] <www.ibm.com/software/br/analytics/spss/>
[2] <https://www.google.com/intl/pt-BR/sheets/about/>

- PEOU3: Eu acho a ferramenta ArgoUML fácil de usar.
- PEOU4: Eu acho fácil realizar atividades de engenharia reversa com a ferramenta ArgoUML.

PU - Percepção da Utilidade:

- PU1: Usar a ferramenta ArgoUML melhora minha performance ao realizar atividades de engenharia reversa.
- PU2: Usar a ferramenta ArgoUML durante a realização das atividade de engenharia reversa aumenta minha produtividade.
- PU3: Usando a ferramenta ArgoUML minha eficácia na realização das atividades de engenharia reversa aumenta.
- PU4: Eu acho a ferramenta ArgoUML útil na realização das atividades de engenharia reversa.

BI - Intenção Comportamental:

- BI1: Assumindo que tive acesso à ferramenta ArgoUML, pretendo usá-la novamente.
- BI2: Dado que eu tinha acesso a ferramenta ArgoUML, eu previ que à usaria.

5.2.2 Estatística descritiva

Segundo Wohlin *et al.* (2012), as estatísticas descritivas dizem respeito a apresentação dos dados coletados após a realização do experimento, essas estatísticas podem ser usadas para descrever e apresentar graficamente aspectos interessantes do conjunto de dados coletados. Ele ainda ressalta a importância de usar essas estatísticas a fim de identificar *outliers*, que são pontos anormais no conjunto de dados, geralmente dados que, numericamente, se distanciam-se muito dos demais.

O objetivo da estatística descritiva é dar ao analista uma ideia de como o conjunto de dados está distribuído, são normalmente utilizadas antes de realizar os testes de hipóteses, para garantir o completo entendimento do conjunto de dados que será testado para aceitar ou rejeitar uma determinada hipótese (WOHLIN *et al.*, 2012).

5.2.2.1 Análise gráfica

Como já foi dito anteriormente, foram analisados três aspecto em cada ferramenta, são eles: facilidade de uso (PEOU), percepção da utilidade (PU) e intenção comportamental (BI). O questionário que foi elaborado para avaliar as ferramentas seguindo a escala de Likert, as

alternativas de respostas foram: Concordo Totalmente, Concordo Amplamente, Neutro, Discordo Amplamente e Discordo Totalmente.

A Tabela 1 mostra os dados obtidos sobre a percepção dos participantes em relação a facilidade de uso (PEOU) das ferramentas. O aspecto analisado tinha quatro afirmações das quais os participantes poderiam escolher dentre as afirmativas disponíveis.

Tabela 1 – Percepção sobre a Facilidade de Uso

Aspecto	**Alternativas**				
	C. Totalmente	C. Amplamente	Neutro	D. Amplamente	D. Totalmente
PEOU1-ArgoUML	0%	8%	23%	54%	15%
PEOU1-Umbrello	8%	15%	23%	46%	8%
PEOU2-ArgoUML	0%	15%	8%	46%	31%
PEOU2-Umbrello	0%	31%	0%	46%	23%
PEOU3-ArgoUML	0%	23%	15%	38%	23%
PEOU3-Umbrello	0%	23%	38%	23%	15%
PEOU4-ArgoUML	15%	15%	23%	15%	31%
PEOU4-Umbrello	15%	31%	23%	23%	8%

Fonte: Elaborado pelo autor.

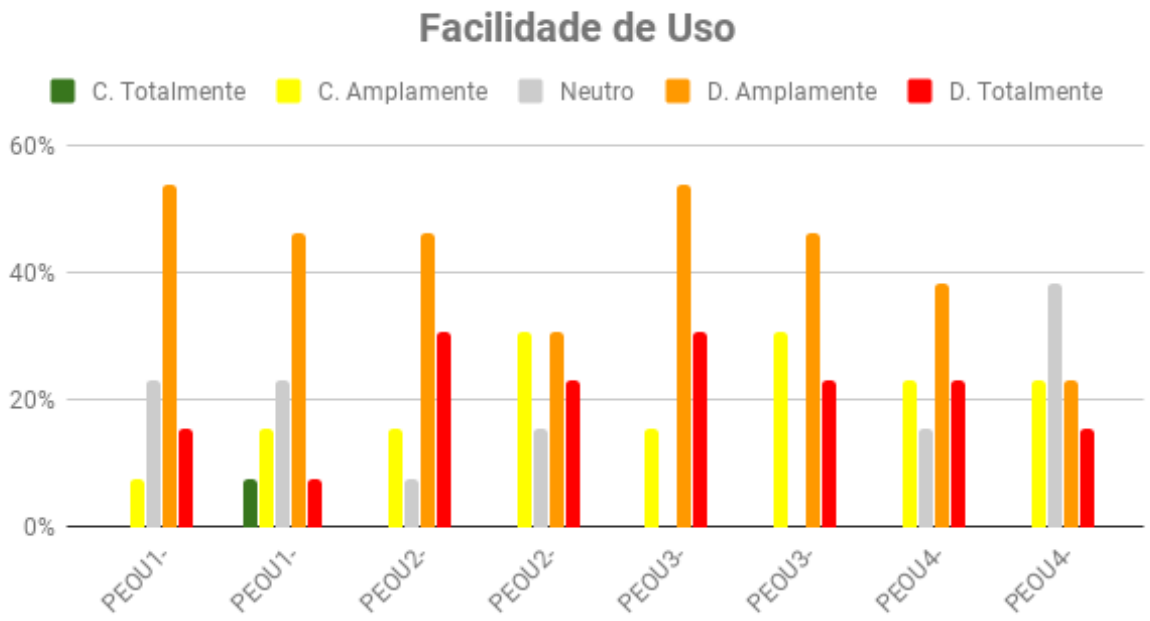
Como podemos observar, com exceção da afirmação PEOU1 da ferramenta Umbrello, em todas as outras afirmativas nenhum dos participantes concordaram totalmente. Ainda em relação a afirmação PEOU1 da ferramenta Umbrello, 46% discordam amplamente da afirmativa, 23% ficaram neutros, 15% concordaram amplamente e 15% discordam totalmente.

Levando em consideração a mesma afirmação em relação a ferramenta ArgoUML, podemos observar que nenhum dos participantes concordaram totalmente, 8% concordam amplamente, 23% ficaram neutro, mesma quantidade da ferramenta Umbrello, 54% discordam amplamente e 15% discordam totalmente.

A Figura 12 mostra um gráfico com os dados em relação a facilidade de uso das ferramentas ArgoUML e Umbrello UML Modeller. Graficamente é perceptível notar que em todas as afirmações em ambas as ferramentas a maioria dos participantes escolheram discordar amplamente (barra laranja no gráfico), o que mostra que as ferramentas não tem uma boa facilidade de uso de acordo com os participantes.

A Tabela 2 mostram os dados coletados em relação ao aspecto de percepção da utilidade das ferramentas (PU). Nesse aspecto, a afirmação PU4 de ambas as ferramentas atingiram 54%, o grau de discordância nesse aspecto foi baixo, pois podemos observar que a opção Discordo Totalmente obteve apenas 8% nas afirmações PU1, PU2 e PU3, e 0% nas afirmações restantes.

Figura 12 – Facilidade de Uso



Fonte: Elaborado pelo autor.

Tabela 2 – Resultados relacionados à percepção sobre a Utilidade

Aspecto	**Alternativas**				
	C. Totalmente	C. Amplamente	Neutro	D. Amplamente	D. Totalmente
PU1-ArgoUML	8%	15%	54%	15%	8%
PU1-Umbrello	31%	31%	23%	8%	8%
PU2-ArgoUML	15%	31%	23%	23%	8%
PU2-Umbrello	15%	38%	8%	31%	8%
PU3-ArgoUML	0%	38%	31%	23%	8%
PU3-Umbrello	15%	23%	54%	8%	0%
PU4-ArgoUML	54%	15%	31%	0%	0%
PU4-Umbrello	54%	23%	23%	0%	0%

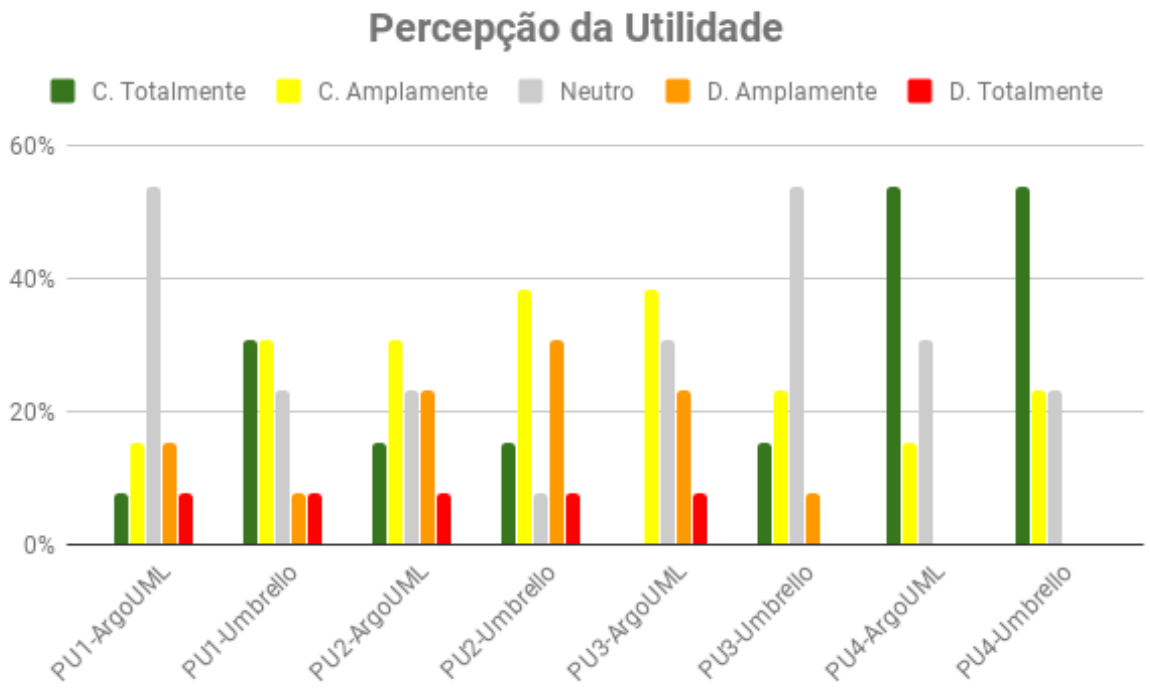
Fonte: Elaborado pelo autor.

A Figura 13 mostram os dados da Tabela 2 dispostos graficamente. Pode-se observar que a 54% dos participantes ou se mantiveram neutros ou concordaram totalmente com as alternativas.

O grau de discordância nesse aspecto foi baixo em ambas as ferramentas, a opção Discordo Amplamente atingiu no máximo 31%, que foi na afirmação PU2 da ferramenta Umbrello. A opção Discordo Totalmente atingiu no máximo 8% nas afirmações PU1, PU2 de ambas as ferramentas, e na afirmação PU3 da ferramenta ArgoUML.

O último aspecto que foi analisado foi a Intenção Comportamental (BI), a Tabela 3 mostram os dados coletados por meio do questionário aplicado ao final do experimento.

Figura 13 – Percepção da Utilidade



Fonte: Elaborado pelo autor.

Diferentemente dos outros dois aspectos analisados, este tem apenas duas afirmativas, mas as opções de escolha são as mesmas.

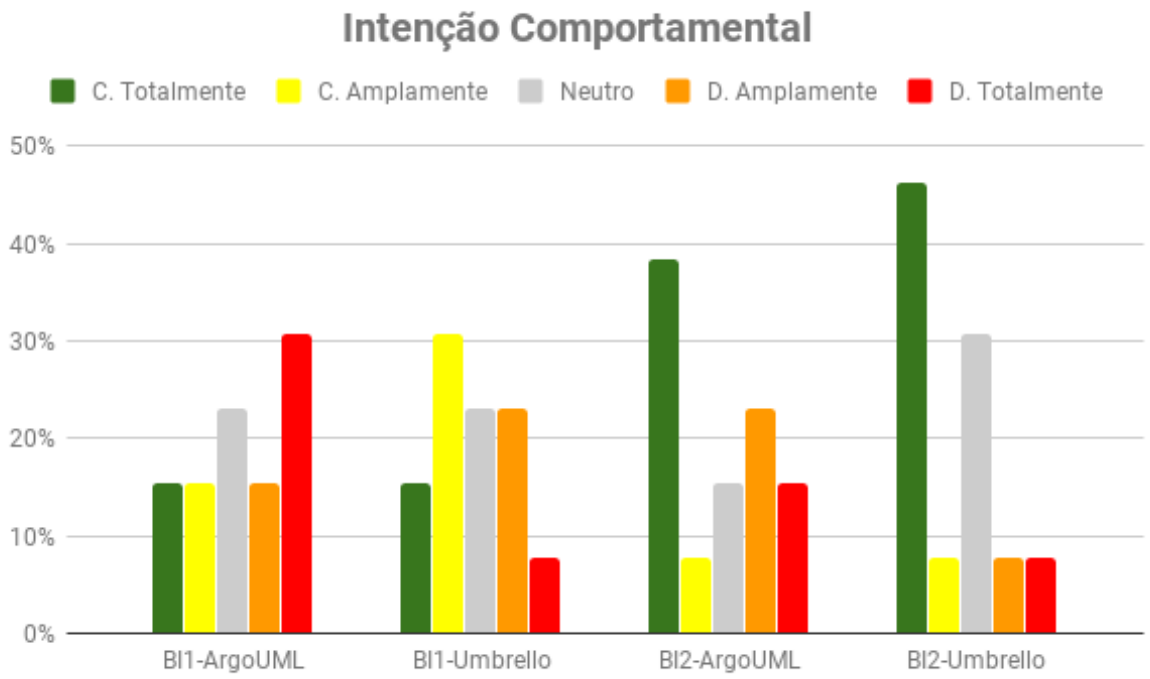
Tabela 3 – Resultados relacionados à percepção sobre a Intenção Comportamental

Aspecto	**Alternativas**				
	C. Totalmente	C. Amplamente	Neutro	D. Amplamente	D. Totalmente
BI1-ArgoUML	31%	15%	23%	15%	15%
BI1-Umbrello	8%	23%	23%	31%	15%
BI2-ArgoUML	15%	23%	15%	8%	38%
BI2-Umbrello	8%	8%	31%	8%	46%

Fonte: Elaborado pelo autor.

A Figura 14 mostram graficamente os dados tabulados na Tabela 3, as opções Concordo Amplamente e Neutro tiveram porcentagens de escolha parecidas, como podemos observar no gráfico. Também podemos notar que o grau de concordância da afirmativa BI1 da ferramenta ArgoUML passou dos 31%, algo que não ocorreu nas outras afirmações com esta opção de escolha. Sobre o grau de discordância nesse aspecto, podemos observar que a afirmação BI2 da ferramenta ArgoUML atingiu 38%, enquanto a mesma afirmação da ferramenta Umbrello atingiu 46%. A afirmação BI1 da ferramenta Umbrello obteve uma porcentagem de 31% na opção Discordo Amplamente, sendo a maior porcentagem obtida nessa opção.

Figura 14 – Intenção Comportamental



Fonte: Elaborado pelo autor.

5.2.2.2 Análise através de gráficos Boxplots

Outra maneira de analisar os dados coletados é por meio de gráficos *Boxplot*, ou gráficos de caixas. De acordo com Capela e Capela (2011), o gráfico *boxplot* é uma das figuras mais utilizadas para descrever graficamente uma variável que está sendo estudada, ele permite avaliar a simetria dos dados, a dispersão e a existência ou não de outliers, sendo especialmente adequado para a comparação de dois ou mais grupos.

Os valores que podem ser observados no *boxplot* são o valor mínimo, o valor máximo, valores discrepantes, a mediana, que equivale ao segundo quartil. O *boxplot* é dividido em três quartis, o intervalo entre cada quartil equivale a 25% dos dados, no exemplo da mediana, temos que 25% dos dados estão abaixo e 25% dos dados estão a cima.

Os *boxplots* que serão mostrados a seguir foram gerados através da ferramenta SPSS, durante seu período gratuito. Para cada pergunta relacionada ao aspecto, o participante poderia deveria escolher a alternativa seguindo a escala Likert. Para cada alternativa foi atribuído um valor, como mostra o Quadro 4.

Os *boxplots* foram gerados através da soma dos valores atribuídos as alternativas da escala Likert, como cada valor equivale a uma alternativa, temos como realizar os cálculos necessários através dos valores atribuídos a elas.

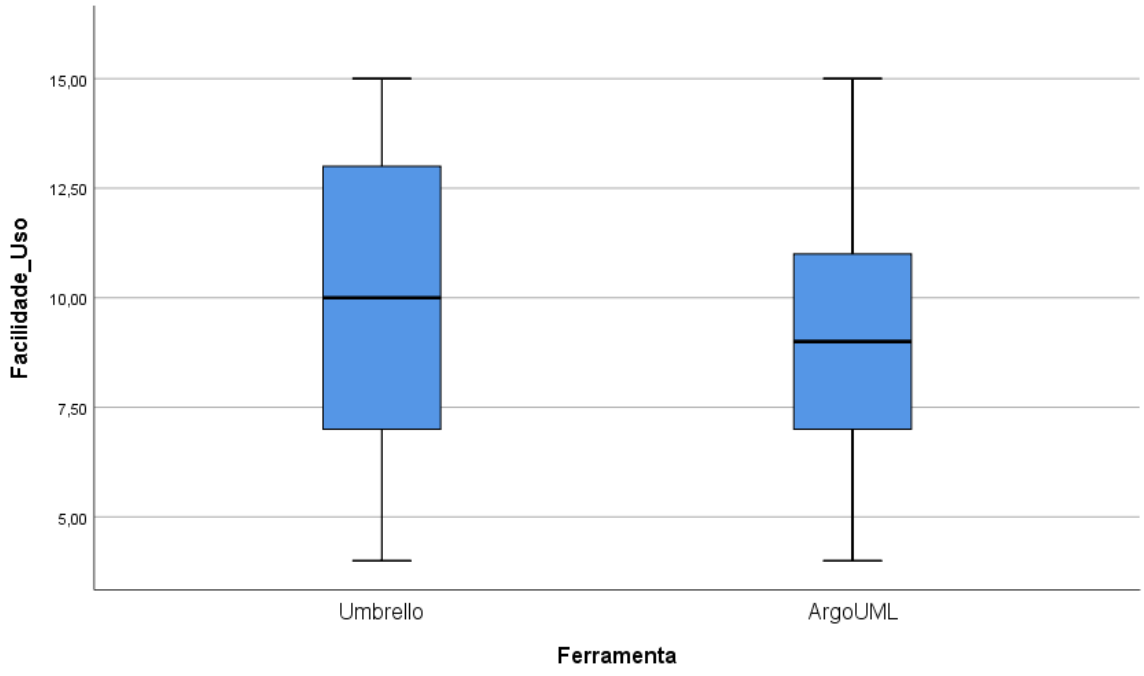
Quadro 4 – Valores para a escala Likert

Alternativas					
	C. Totalmente	C. Amplamente	Neutro	D. Amplamente	D. Totalmente
Valores	5	4	3	2	1

O objetivo do experimento é avaliar aspectos de utilidade, facilidade de uso e intenção comportamental das ferramentas. Para cada um destes aspectos um *boxplot* foi gerado.

A Figura 15 mostra o *boxplot* do aspecto facilidade de uso. Em ambas as ferramentas o valor máximo da soma máxima das alternativas atingiu o valor 15, na ferramenta Umbrello notemos que a mediana (linha preta dentro da caixa) está acima da mediana da ferramenta ArgoUML. Com isso podemos afirmar que 25% dos valores da ferramenta Umbrello estão abaixo do valor 10, e 25% dos valores da ferramenta ArgoUML estão abaixo do valor 9. Esses dados explicam o baixo grau de concordância, já que ambas as ferramentas não atingiram um valor alto no critério em questão.

Figura 15 – Boxplot - Facilidade de Uso

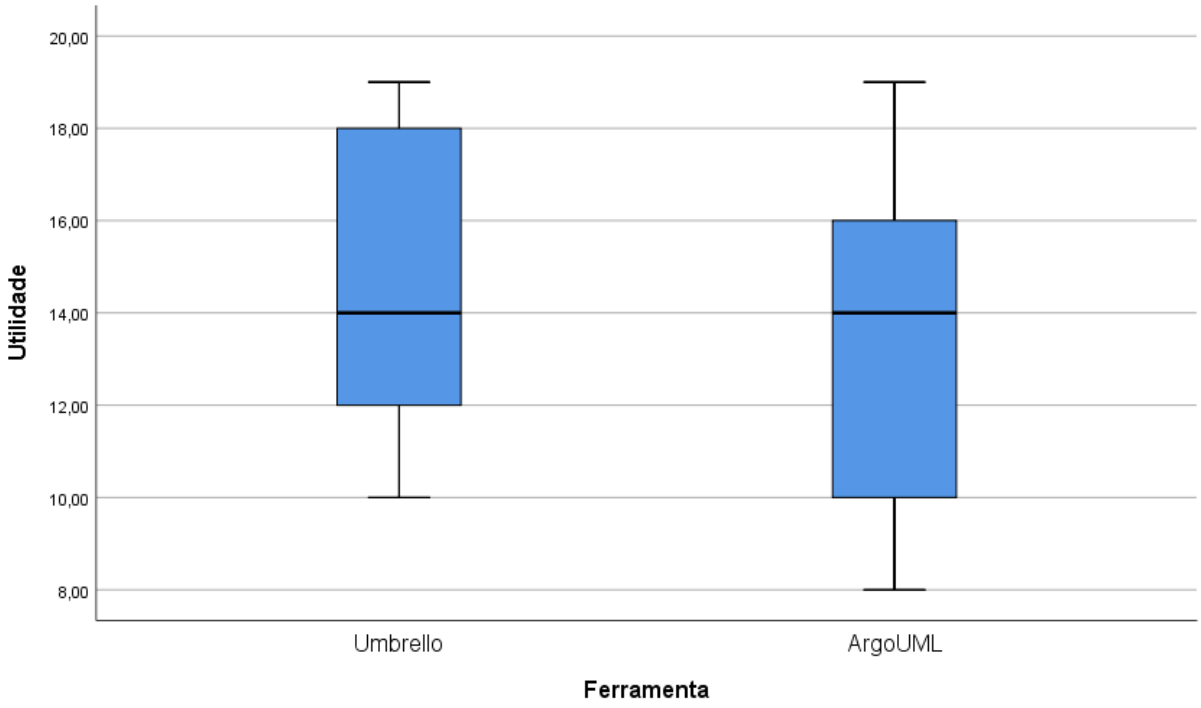


Fonte: Elaborado pelo autor.

A Figura 16 mostra o *boxplot* do aspecto utilidade. Neste aspecto podemos observar que o grau de discordância da ferramenta ArgoUML foi menor do que o da ferramenta Umbrello, pois o valor mínimo obtido pela ferramenta Umbrello foi igual a 10, enquanto o da ferramenta

ArgoUML foi abaixo de 10. Observando a mediana podemos notar que as ferramentas se mantiveram iguais. Em ambas as ferramentas o grau de concordância foi alto, a mediana foi igual a 14 nas duas ferramentas.

Figura 16 – Boxplot - Utilidade

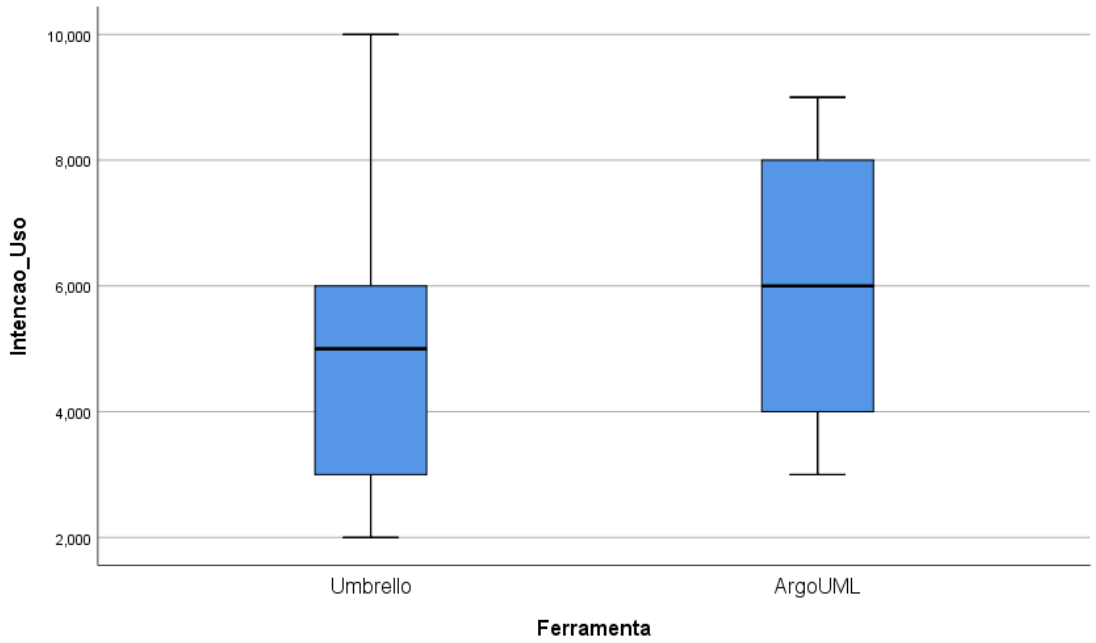


Fonte: Elaborado pelo autor.

Nos outros dois aspectos avaliados anteriormente, a soma máxima que poderia ser obtida era 20, pois eram 4 afirmações, ou seja, supondo que o participante escolhesse a opção 5 em todas as afirmações, o valor da soma seria 20. No aspecto de intenção comportamental, o valor máximo da soma será 10, pois são apenas duas afirmações.

A Figura 17 mostra o *boxplot* do aspecto intenção comportamental. O grau de discordância da ferramenta Umbrello foi alto, pois o valor mínimo foi obtido, e o conjunto de dados ficou muito próximo desse valor. A mediana da ferramenta ArgoUML atingiu o valor 8, enquanto o da ferramenta Umbrello atingiu 6, o que é esperado devido ao conjunto de dados ter se aproximado do valor mínimo obtido.

Figura 17 – Boxplot - Intenção Comportamental



Fonte: Elaborado pelo autor.

5.2.3 Teste de hipóteses

Segundo Wohlin *et al.* (2012), o objetivo do teste de hipótese é verificar se é ou não possível rejeitar as hipóteses nulas definidas na fase de planejamento do experimento, baseado em alguma distribuição estatística.

No caso da presente da pesquisa, o objetivo é rejeitar as hipóteses nulas definidas na seção 4.4 subseção 4.4.3. O teste de hipóteses foi realizado na ferramenta SPSS, duas tabelas foram geradas, uma com as estatísticas dos grupos, e outra com o teste de amostras independentes. A Figura 18 mostra a tabela gerada pela ferramenta com as estatísticas dos grupos. A tabela possui as colunas de média dos valores, erro desvio e erro padrão da média. Cada linha é um critério que foi avaliado em relação as duas ferramentas.

A média dos valores do critério utilidade foi maior na ferramenta ArgoUML, assim como o erro desvio e o erro padrão da média. No critério facilidade de uso, a média também foi maior na ferramenta ArgoUML, o erro desvio foi maior na ferramenta Umbrello e o erro padrão média foi maior na ferramenta Umbrello. No critério intenção comportamental, tanto a média como o erro desvio foram maiores na ferramenta Umbrello, já o erro padrão da média foi maior na ferramenta ArgoUML.

Figura 18 – Teste de Hipóteses - Estatísticas dos Grupos.

Estatísticas de grupo

	Ferramenta	N	Média	Erro Desvio	Erro padrão da média
Utilidade	Umbrello	13	9.3077	2.98286	.82730
	ArgoUML	13	10.5385	3.50275	.97149
Facilidade_Uso	Umbrello	13	13.6923	4.06990	1.12879
	ArgoUML	13	15.3077	3.44927	.95665
Intencao_Uso	Umbrello	13	7.00000	2.345208	.650444
	ArgoUML	13	6.00000	2.081666	.577350

Fonte: Elaborado pelo autor.

O erro padrão da média fornece uma indicação da provável precisão da média da amostra como uma estimativa da média da população. Quanto menor o erro padrão, menor a dispersão e mais provável é que qualquer média de amostra esteja próxima à média da população. A Média consiste na divisão da soma pela contagem de elementos, dados ou eventos, resultando em um valor único que representa o conjunto de valores. É influenciado por valores discrepantes, ou seja, valores que se distanciam muito dos valores do conjunto.

A Figura 19 mostra a tabela gerada pela ferramenta com os dados do teste de Levene e do teste-t, ambos os testes servem para verificar a homogeneidade dos dados, ou seja, para verificar se rejeitamos ou aceitamos as hipóteses nulas definidas anteriormente, levando em consideração um intervalo de confiança de 95%, consequentemente um nível de significância de 0.05, pois o nível de significância é dado por 1 - 95%, onde 95% é o intervalo de confiança.

Para verificarmos se aceitamos ou não as hipóteses nulas, devemos levar em consideração a coluna Sig.(2 extremidades), ou seja, o *p-value* da Figura 19, se o valor for maior que o nível de significância (0.05) aceitaremos as hipóteses nulas, se for menor aceitamos as hipóteses alternativas. Cada critério possui dois valores na coluna Sig.(2 extremidades), variâncias iguais assumidas e variâncias iguais não assumidas.

Como podemos observar, nos três aspectos avaliados os valores foram maiores do que 0.05, sendo 0.344 e 0.345 para a Utilidade, 0.286 e 0.286 para Facilidade de uso, e, 0.262 e 0.262 para a Intenção comportamental. Com esses valores não rejeitamos as hipóteses nulas, ou seja, significa dizer que estatisticamente não há diferença significativa entre as ferramentas, levando em consideração os critérios que foram avaliados. Isso significa que não podemos afirmar estatisticamente que as ferramentas apresentam diferenças nos critérios avaliados, embora tenha

Figura 19 – Teste de Hipóteses - Teste-t e Teste de Levene.

Teste de amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
Utilidade	Variâncias iguais assumidas	.387	.540	-.965	24	.344	-1.23077	1.27601	-3.86433	1.40279
	Variâncias iguais não assumidas			-.965	23.406	.345	-1.23077	1.27601	-3.86787	1.40633
Facilidade_Uso	Variâncias iguais assumidas	.442	.513	-1.092	24	.286	-1.61538	1.47964	-4.66922	1.43845
	Variâncias iguais não assumidas			-1.092	23.372	.286	-1.61538	1.47964	-4.67357	1.44280
Intencao_Uso	Variâncias iguais assumidas	.000	1.000	1.150	24	.262	1.000000	.869718	-.795011	2.795011
	Variâncias iguais não assumidas			1.150	23.667	.262	1.000000	.869718	-.796349	2.796349

Fonte: Elaborado pelo autor.

sido perceptível a ascensão de uma ferramenta sobre a outra em alguns critérios.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Neste capítulo será apresentada uma breve discussão sobre os resultados do trabalho, bem como as considerações finais e os trabalhos futuros.

6.1 Discussão sobre os resultados

Nos resultados obtidos pelo experimento ficou clara a avaliação negativa dos participantes em relação aos critérios avaliados nas ferramentas. No questionário de caracterização do perfil dos participantes não foi perguntado se eles já tinham algum conhecimento prévio sobre as ferramentas que estavam sendo avaliadas, este fator poderia influenciar no resultado, pois se eles já conhecessem a ferramentas, não teriam tanta dificuldade em usá-la.

Durante a realização do experimento, houveram várias dúvidas dos participantes, principalmente em relação a importação do código fonte do sistema para a ferramenta, essas dúvidas não foram levadas em consideração na análise dos resultados da pesquisa, quando ocorriam dúvidas o autor ia até o participante e auxiliava-o na resolução do problema.

Outro fator a ser discutido é em relação as dicas presentes no roteiro do experimento, tais dicas foram disponibilizadas com o intuito de auxiliar os participantes na realização das atividades. As dicas também foram pensadas para fazer com que o participante não desista do experimento caso não consiga realizar as tarefas, pois com as dicas, a realização das tarefas ficava mais fácil de ser realizada.

6.2 Conclusões

O presente trabalho apresentou um estudo comparativo entre ferramentas de análise estática, no contexto de engenharia reversa. O estudo foi realizado através de um experimento controlado, em uma turma da disciplina de Experimentação em Engenharia de Software. O estudo visou o auxiliar empresas de desenvolvimento de *software* a escolher uma ferramenta de engenharia reversa que melhor se adéque ao contexto da empresa.

O objetivo da pesquisa foi avaliar as ferramentas de engenharia reversa em relação a utilidade, facilidade de uso e intenção comportamental. Além desses critérios, o autor realizou uma análise das ferramentas em relação ao componentes que cada uma das ferramentas conseguiria recuperar do sistema utilizado na pesquisa.

É de suma importância a utilização de ferramentas de engenharia reversa, tendo em

vista que sistemas podem ter milhões de linhas de código, o que torna inviável fazer a recuperação de modelos e componentes do sistema de forma manual. Existe uma variedade muito grande de ferramentas de engenharia reversa disponíveis no mercado, o que acaba dificultando a escolha de uma que se adéque ao contexto de uma determinada empresa, daí se da a importância da presente pesquisa.

O experimento contou com 26 participantes, que foram divididos em dois grupos, cada grupo com 13 participantes, os grupos foram balanceados levando em consideração a experiência de mercado e conhecimento do participante em relação a engenharia reversa, manutenção de *software* e desenvolvimento de *software*.

A presente pesquisa é importante também para o meio acadêmico, pois durante a revisão da literatura não foram encontrados trabalhos que avaliassem ferramentas de engenharia reversa levando em consideração os critérios avaliados por esta pesquisa, com exceção do trabalho de Cindra, Barcelos e Lisbôa (2011), no entanto, o trabalho leva em consideração somente os componentes que as ferramentas conseguem recuperar, deixando de lado os aspectos de usabilidade, facilidade de uso e intenção comportamental.

Após o experimento os dados foram coletados e a análise estatística dos dados foi feita utilizando a ferramenta da SPSS da IBM. Os dados mostraram que estatisticamente as ferramentas não apresentaram diferença significativa em relação aos critérios avaliados, no entanto, os dados da pesquisa continuam sendo importantes para empresas de desenvolvimento de *software*, pois em alguns dos critérios uma ferramenta se sobressaiu em relação a outra, mesmo que estatisticamente essa diferença seja insignificante, levando em consideração o nível de significância de 0.05.

No Capítulo 1 seção 1.3 foram apresentados os objetivos que a pesquisa buscou alcançar. Por meio do experimento realizado, é possível auxiliar empresas de desenvolvimento de *software* a escolherem uma ferramenta de engenharia reversa que melhor se adéque ao contexto da empresa. Durante a realização da pesquisa foram discutidos e apresentados pontos que ressaltam a importância da manutenção de *software*, bem como a utilização de ferramentas de engenharia reversa para a recuperação de componentes e modelos de sistemas que possuem pouca ou nenhuma documentação.

Os dados em relação aos componentes recuperados pelas ferramentas foram apresentados no Capítulo 5 seção 5.1, foi mostrado desde a etapa de pesquisa por ferramentas até a análise dos componentes que cada ferramenta conseguiu recuperar do sistema. Através da

realização do experimento foi possível apresentar os dados propostos nos objetivos de pesquisa, em relação a facilidade de uso, utilidade e intenção comportamental das ferramentas.

Portanto, concluí-se que os dados da presente pesquisa são de suma importância tanto para o meio acadêmico quanto para o mercado de trabalho, os objetivos propostos pela pesquisa foram alcançados com excelência por meio da realização do experimento e da análise de componentes recuperados, além de prover discussões durante a realização da pesquisa e um auto-aprendizado que será de suma importância para o crescimento acadêmico e profissional do autor.

6.3 Trabalhos Futuros

Este trabalho abre oportunidades para pesquisas futuras, em relação tanto ao experimento, quanto a análise de componentes recuperados pelas ferramentas. Algo que pode ser feito, é realizar a análise de componentes utilizando as ferramentas em um outro sistema, pois como foi dito anteriormente, alguns dos componentes e relacionamentos que as ferramentas não conseguiram recuperar não estavam presentes no sistema, como a herança por exemplo.

Outro fator a ser observado é o grupo de alunos que participaram do experimento, um novo estudo pode ser realizado em um ambiente empresarial. É possível que o resultado seja diferente, tendo em vista que os grupos foram balanceados levando em consideração a experiência na indústria, logo, se o experimento for realizado no meio industrial, todos os participantes já terão experiência, o que irá afetar diretamente o resultado da pesquisa.

Um novo estudo pode ser realizado também com outras ferramentas, para isso, poderiam ser definidos novos critérios para a escolha dessas ferramentas, as mesmas poderiam ser aplicadas em um outro sistema, e, o experimento poderia ser realizado com outro grupo de alunos, ou, em meio industrial, pois esses são os fatores que influenciam diretamente na pesquisa.

Futuramente poderá ser realizado uma análise dos diagramas de classes gerados pelos participantes durante o experimento, tendo em vista que após a realização das atividades os participantes salvaram os resultados, o autor terá como verificar se as tarefas foram realizadas de maneira correta, uma análise quantitativa e qualitativa poderá ser realizada em cima desses dados.

Esses trabalhos futuros têm como objetivo enfatizar ainda mais a necessidade de melhorias nas ferramentas de apoio a engenharia reversa, pois levando em consideração os dados do experimento, é notável a avaliação negativa dos participantes em relação aos critérios

considerados na avaliação, sendo essa uma das principais contribuições da pesquisa.

REFERÊNCIAS

ALLEN, I. E.; SEAMAN, C. A. Likert scales and data analyses. **Quality Progress**, United States–US, v. 40, n. 7, p. 64–65, 07 2007. Disponível em: https://search.proquest.com/docview/214764202?accountid=26598. Acesso em: 21 jun 2018.

ZAGO JUNIOR, A.; FERREIRA, M. A. G. V. Manutenção corretiva baseada em padrões e antipadrões de casos de uso. In: **WORKSHOP DE MANUTENÇÃO DE SOFTWARE MODERNA(WMSWM)**, 3., 2006, São Paulo. Anais[...]. Espírito Santo: UNIFOR; UFRJ, 2006. p. 74-81.

ASSUNÇÃO, A. N. **Tela de Login**. 2013. Disponível em: https://github.com/ArthurAssuncao/AgendaExames/blob/master/screenshots/tela_login.png. Acesso em: 24 maio 2018.

CALAZANS, A. T. S.; OLIVEIRA, M. Avaliação de estimativa de tamanho para projetos de manutenção de software. In: **Proc. of Argentine Symposium on Software Engineering**. Buenos Aires, Argentina: Toffano Seidel Calazans et al., 2005. p. 65-74.

CANHOTA JUNIOR, A. J. S. da; SOUZA, D. A. de; MOUTINHO, D. dos S.; LOHNEFINK, F. P. Engenharia reversa. 2005. Disponível em: http://www2.ic.uff.br/~otton/graduacao/informaticaI/apresentacoes/eng_reversa.pdf. Acesso em: 22 abr 2018.

CAPELA, M. V.; CAPELA, J. M. Elaboração de gráficos box-plot em planilhas de cálculo. In: **CONGRESSO DE MATEMÁTICA APLICADA E COMPUTACIONAL DA REGIÃO SUDESTE–CNMAC Sudeste**. Araraquara: UNESP. Anais[...]. 2011. v. 1. p. 361-364.

CHESS, B.; WEST, J. **Secure Programming with Static Analysis**. Boston: Addison-Wesley Professional, 2007.

CHIKOFSKY, E. J.; CROSS, J. H. Reverse engineering and design recovery: A taxonomy. IEEE software, Cambridge-USA, v. 7, n. 1, p. 13–17, 1990.

CINDRA, J. da S.; BARCELOS, M. R. dos S.; LISBÔA, J. C. Uma pesquisa sobre ferramentas case para engenharia reversa estática. **Humanas Sociais & Aplicadas**, v. 1, n. 2, p. 45–51, 2011.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, JSTOR, p. 319–340, 1989.

FONTANETE, V.; GARCIA, V. C.; BOSSONARO, A. A.; PEREZ, A. B.; PRADO, A. F. do. Reengenharia de sistemas legados baseada em componentes usando transformações. São Carlos/SP, 2003.

FREITAS NETO, J. J. de; MENDONÇA, N. das C. Redox-uml: Redocumentação de aplicações legadas cobol usando xml e uml. In: **Workshop de Manutenção de Software Moderna WMSWM**. Espírito Santo, Brasil: UNIFOR/UFRJ, 2006. p. 27–42.

LIENTZ, B. P.; SWANSON, E. B. **Software maintenance management**. Reading, MA: Addison-Wesley, 1980. Disponível em: https://cds.cern.ch/record/101847. Acesso em: 15 jul 2018.

MARTINS, C. **Qualidade de Software: A importância da manutenção**. 2009. Disponível em: http://celsoavmartins.blogspot.com.br/2009/04/importancia-da-manutencao.html. Acesso em: 19 mai 2018.

MOREIRA, R. T. **Um perfil de capacidade para a melhoria do processo em micro e pequenas organizações orientadas à manutenção e evolução de produtos de software.** Universidade Federal de Pernambuco, 2015.

OTANI, M.; MACHADO, W. V. A proposta de desenvolvimento de gestão da manutenção industrial na busca da excelência ou classe mundial. **Revista Gestão Industrial**, v. 4, n. 2, 2008. p. 1-16.

PADUELLI, M. M. **Manutenção de Software:** problemas típicos e diretrizes para uma disciplina específica. 2007. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional.) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2007.

PRADO, D.; BETTIN, A. X.; TOBAR, C. M.; PAGANO, V. A. A ferramenta de análise estática klocwork integrada a um processo formal de revisão de código, nível 3 do cmmi. **VIII Simpósio Brasileiro de Qualidade de Software, SBQS**, 2009.

PRESSMAN, R. **Engenharia de software**. McGraw-Hill, 2002. ISBN 9788586804250.

PRESSMAN, R. **Engenharia de Software - 7.ed.:**. Porto Alegre, Brasil: McGraw Hill Brasil, 2009. ISBN 9788580550443.

SILVA, P. M. Modelo de aceitação de tecnologia (tam) aplicado ao sistema de informação da biblioteca virtual em saúde (bvs) nas escolas de medicina da região metropolitana do recife. PPGCI/UFPB, 2008.

SOMMERVILLE, I. **Engenharia de software**. 9th. ed. PEARSON BRASIL, 2011. ISBN 9788579361081.

STEINMACHER, I.; AMORIM, É. F.; SCHIAVONI, F. L.; HUZITA, E. H. M. Geca: Uma ferramenta de engenharia reversa e geração automática de código. **Simpósio Brasileiro de Sistemas de Informação (SBSI) 2006 Proceedings (em cd). Curitiba**, 2006.

TERRA, R.; BIGONHA, R. S. Ferramentas para análise estática de códigos java. In: **III Encontro Brasileiro de Teste de Software**. Recifie: EBTS, 2008. p. 1–5.

VENKATESH, V.; BALA, H. Technology acceptance model 3 and a research agenda on interventions. **Decision sciences**, Wiley Online Library, v. 39, n. 2, p. 273–315, 2008.

VERONESE, G.; CORREA, A.; WERNER, C.; JEZINI, N. F. ares: Uma ferramenta de engenharia reversa java-uml. **SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE–SBES**, v. 16, p. 347–352, 2002.

WOHLIN, C.; R., P.; MARTIN, H.; M., O. C.; R., B.; W., A. **Experimentation in Software Engineering**. 1. ed. Springer-Verlag Berlin Heidelberg, 2012. ISBN 978-3-642-29043-5,978-3-642-29044-2. Disponível em: http://gen.lib.rus.ec/book/index.php?md5= C9E97D7982CEAD36A8629317D6F0B15. Acesso em: 19 set 2018.

APÊNDICE A – QUESTIONÁRIO PARA CARACTERIZAÇÃO DOS PARTICIPANTES DO EXPERIMENTO

CARACTERIZAÇÃO DO PARTICIPANTE

Pesquisa: "UM ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA NO CONTEXTO DE ENGENHARIA REVERSA."

Organização: Universidade Federal do Ceará

Participante: _____

Prezado Senhor (a), O formulário abaixo será utilizado para compreender seu grau de familiaridade com os diversos aspectos relacionados a manutenção e desenvolvimento de *software*. As informações coletadas serão tratadas confidencialmente.

Conhecimento sobre manutenção de *software*:

Em relação ao grau do seu conhecimento prévio sobre manutenção de *software*, marque os itens abaixo que melhor se aplicam à sua resposta.

Ao responder, considere experiências práticas como participação em projetos focados em garantir melhoria na qualidade de produtos com foco na manutenção de sistemas.

- (__) não possuo nenhum conhecimento prévio sobre manutenção de *software*.
- (__) tenho algumas noções de manutenção de *software* adquiridas através de leituras/palestras.
- participei de (____) projeto(s) sobre manutenção de *software* em sala de aula.
- participei de (____) projeto(s) sobre manutenção de *software* na indústria.

Conhecimento sobre engenharia reversa:

- (__) não possuo nenhum conhecimento prévio sobre engenharia reversa.
- (__) tenho algumas noções de engenharia reversa adquiridas através de leituras/palestras.
- participei de (____) projeto(s) envolvendo engenharia reversa em sala de aula.
- participei de (____) projeto(s) envolvendo engenharia reversa na indústria.

Experiência com Desenvolvimento de *Software*:

Marque os itens relacionados aos papéis em que você atuou:

- (__) Não possuo experiência com o Desenvolvimento de *Software*.
- (__) Engenheiro de Requisitos/Projetista. Tempo de Experiência: (__) Anos (ou(__) Meses) Ambiente: (__) Indústria (__) Acadêmico
- (__) Desenvolvedor/Programador. Tempo de Experiência: (__) Anos (ou (__) Meses) Ambiente: (__) Indústria (__) Acadêmico

- (__) Designer. Tempo de Experiência: (__) Anos (ou (__) Meses) Ambiente: (__) Indústria (__) Acadêmico
- (__) Gerente de Projeto de Desenvolvimento. Tempo de Experiência: Anos (ou (__) Meses) Ambiente: (__) Indústria (__) Acadêmico
- (__) Arquiteto de *Software*. Tempo de Experiência: Anos (ou (__) Meses) Ambiente: (__) Indústria (__) Acadêmico
- (__) Outros: Ambiente: (__) Indústria (__) Acadêmico

Agradecemos pela colaboração!

APÊNDICE B – QUESTIONÁRIO UTILIZADO PARA AVALIAR A FERRAMENTA ARGOUML SEGUINDO O MODELO TAM

AVALIAÇÃO DA FERRAMENTA ARGOUML

Pesquisa: "UM ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA NO CONTEXTO DE ENGENHARIA REVERSA."

Organização: Universidade Federal do Ceará

Participante: _____

Prezado Senhor (a), O formulário abaixo será utilizado para avaliar sua experiência na utilização da ferramenta ArgoUML. As informações coletadas serão tratadas confidencialmente.

Percepção da Utilidade da ferramenta ArgoUML:

Questão 1: Usar a ferramenta ArgoUML melhora minha performance ao realizar atividades de engenharia reversa.

Concordo [__][__][__][__][__] Discordo

Questão 2: Usar a ferramenta ArgoUML durante a realização das atividade de engenharia reversa aumenta minha produtividade.

Concordo [__][__][__][__][__] Discordo

Questão 3: Usando a ferramenta ArgoUML minha eficácia na realização das atividades de engenharia reversa aumenta.

Concordo [__][__][__][__][__] Discordo

Questão 4: Eu acho a ferramenta ArgoUML útil na realização das atividades de engenharia reversa.

Concordo [__][__][__][__][__] Discordo

Percepção de facilidade de uso da ferramenta ArgoUML:

Questão 5: Minha interação com a ferramenta ArgoUML é clara e compreensível.

Concordo [__][__][__][__][__] Discordo

Questão 6: Interagir com a ferramenta ArgoUML não requer muito do meu esforço mental.

Concordo [__][__][__][__][__] Discordo

Questão 7: Eu acho a ferramenta ArgoUML fácil de usar.

Concordo [__][__][__][__][__] Discordo

Questão 8: Eu acho fácil realizar atividades de engenharia reversa com a ferramenta ArgoUML.

Concordo [__][__][__][__][__] Discordo

Intenção comportamental:

Questão 9: Assumindo que tive acesso à ferramenta ArgoUML, pretendo usá-la novamente.

Concordo [__][__][__][__][__] **Discordo**

Questão 10: Dado que eu tinha acesso a ferramenta ArgoUML, eu previ que à usaria.

Concordo [__][__][__][__][__] **Discordo**

APÊNDICE C – QUESTIONÁRIO UTILIZADO PARA AVALIAR A FERRAMENTA UMBRELLO UML MODELLER SEGUINDO O MODELO TAM

AVALIAÇÃO DA FERRAMENTA UMBRELLO UML MODELLER

Pesquisa: "UM ESTUDO COMPARATIVO DE FERRAMENTAS DE ANÁLISE ESTÁTICA NO CONTEXTO DE ENGENHARIA REVERSA."

Organização: Universidade Federal do Ceará

Participante: _____

Prezado Senhor (a), O formulário abaixo será utilizado para avaliar sua experiência na utilização da ferramenta Umbrello UML Modeller. As informações coletadas serão tratadas confidencialmente.

Percepção da Utilidade da ferramenta Umbrello UML Modeller:

Questão 1: Usar a ferramenta Umbrello UML Modeller melhora minha performance ao realizar atividades de engenharia reversa.

Concordo [__][__][__][__][__] Discordo

Questão 2: Usar a ferramenta Umbrello UML Modeller durante a realização das atividade de engenharia reversa aumenta minha produtividade.

Concordo [__][__][__][__][__] Discordo

Questão 3: Usando a ferramenta Umbrello UML Modeller minha eficácia na realização das atividades de engenharia reversa aumenta.

Concordo [__][__][__][__][__] Discordo

Questão 4: Eu acho a ferramenta Umbrello UML Modeller útil na realização das atividades de engenharia reversa.

Concordo [__][__][__][__][__] Discordo

Percepção de facilidade de uso da ferramenta Umbrello UML Modeller:

Questão 5: Minha interação com a ferramenta Umbrello UML Modeller é clara e compreensível.

Concordo [__][__][__][__][__] Discordo

Questão 6: Interagir com a ferramenta Umbrello UML Modeller não requer muito do meu esforço mental.

Concordo [__][__][__][__][__] Discordo

Questão 7: Eu acho a ferramenta Umbrello UML Modeller fácil de usar.

Concordo [__][__][__][__][__] Discordo

Questão 8: Eu acho fácil realizar atividades de engenharia reversa com a ferramenta Umbrello UML Modeller.

Concordo [__][__][__][__][__] Discordo

Intenção comportamental:

Questão 9: Assumindo que tive acesso à ferramenta Umbrello UML Modeller, pretendo usá-la novamente.

Concordo [__][__][__][__][__] **Discordo**

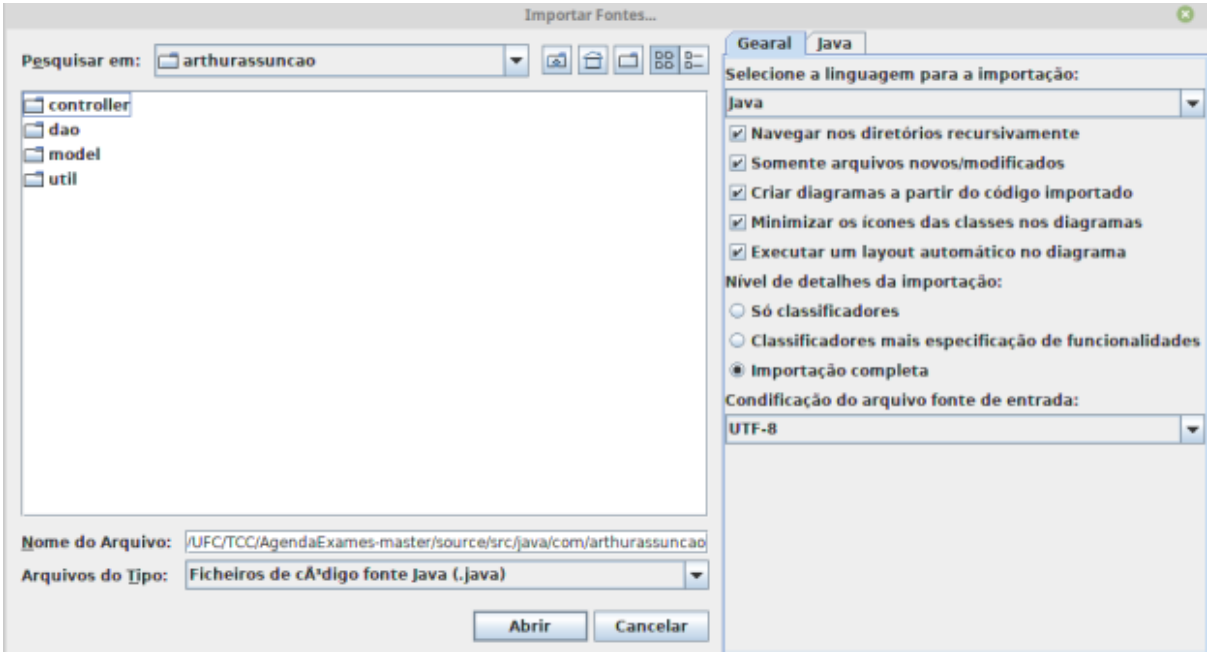
Questão 10: Dado que eu tinha acesso a ferramenta Umbrello UML Modeller, eu previ que à usaria.

Concordo [__][__][__][__][__] **Discordo**

APÊNDICE D – *ROTEIRO DE EXECUÇÃO - ARGOUML*

Você recebeu um documento com a especificação do cenário de um sistema e seus requisitos funcionais. Baseado neste cenário, você deve executar as tarefas abaixo utilizando o ambiente configurado para sua execução com a ArgoUML. Procedimentos Iniciais:

- Importar os códigos fonte do sistema, os códigos estarão na pasta Sistema Agenda na sua área de trabalho. Sistema: AgendaMaster.
    1. Vá em Arquivo – Importar Fontes e selecione as classes que deseja importar.
    2. Você irá se deparar com uma tela parecida com a tela abaixo, selecione as classes e clique em abrir.
    3. Como as classes estão em pacotes diferentes, você deverá fazer o procedimento acima para todos os pacotes do sistema, para que consiga abrir todas as classes.
    4. A imagem abaixo mostra os pacotes do sistema, você precisa entrar nos pacotes e selecionar as classes que deseja importar.



Agora o ambiente está configurado para que você possa iniciar as tarefas descritas abaixo.

Tarefa 1: Realizar a extração do diagrama de classes relacionado ao módulo de cadastro de médicos. OBSERVE DICAS!

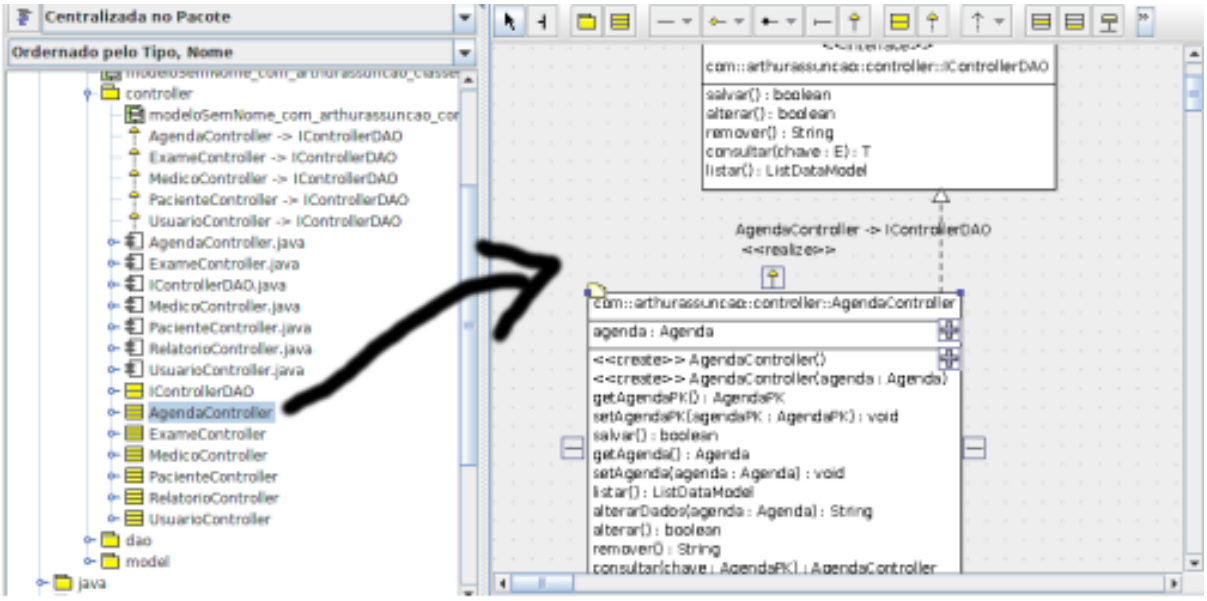
Passo 1: Identificar e abrir as classes relacionadas ao módulo solicitado.

Passo 2: Gerar o diagrama de classes das classes identificadas.

Passo 3: Exporte o diagrama no formato png e salve na pasta do projeto com a seguinte

nomenclatura: seunome-DiagramaTarefa1-ArgoUML.

DICAS: Para gerar o diagrama: Clique em cima da classe e arraste para o espaço vazio na tela principal do programa, faça isso com todas as classes relacionadas com o módulo solicitado. Se as classes tiverem ligação, a ferramenta automaticamente ligará as classes.



Tarefa 2: Agora você deve incluir um novo requisito funcional no sistema, o requisito é Cadastrar Secretária. Você deve alterar o diagrama de classes para que ele se adéque ao novo requisito. OBSERVE DICAS!

Requisito **Cadastrar Secretária**:

Descrição: O sistema deverá permitir o cadastro de secretárias, um médico só pode ter uma secretária, e uma secretária só pode atender a um médico.

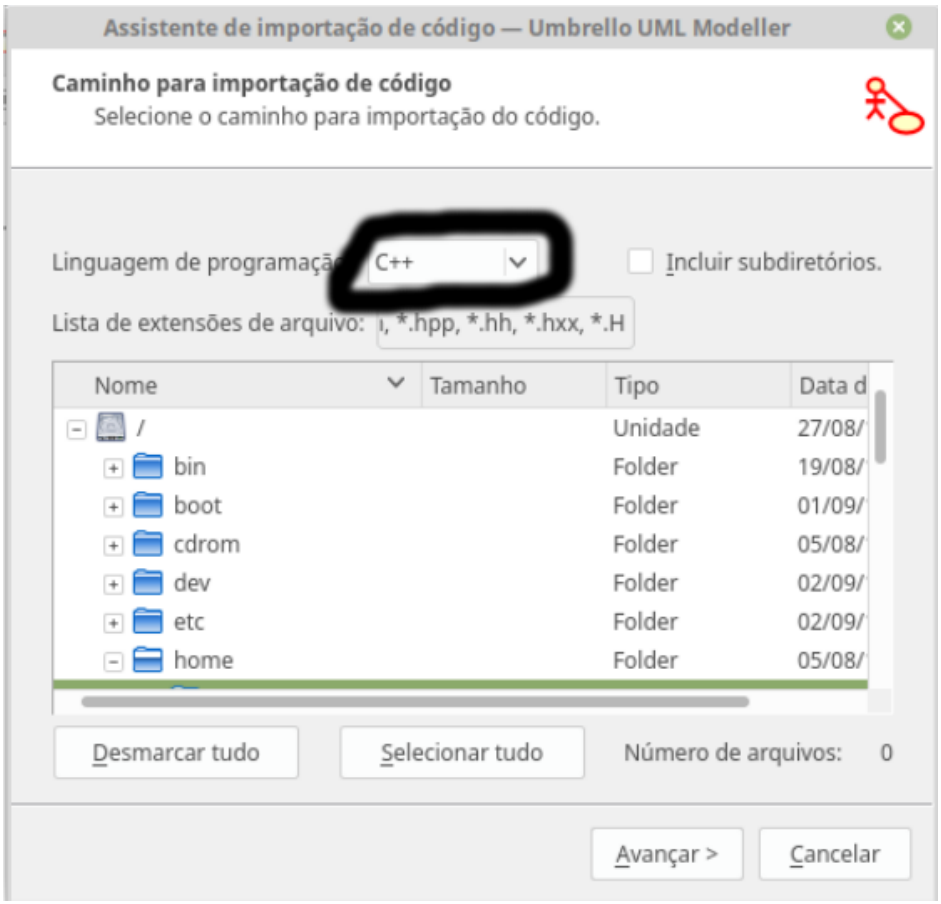
Ator: Funcionário.

DICAS: Para criar uma nova classe, clique com o botão direito em cima do pacote onde deseja inserir a classe, em seguida clique em criar novo elemento do modelo, depois em nova classe.

APÊNDICE E – *ROTEIRO DE EXECUÇÃO - UMBRELLO*

Você recebeu um documento com a especificação do cenário de um sistema e seus requisitos funcionais. Baseado neste cenário, você deve executar as tarefas abaixo utilizando o ambiente configurado para sua execução com a ferramenta Umbrello. Procedimentos Iniciais:

- Importar os códigos fonte do sistema, os códigos estarão na pasta Sistema Agenda na sua área de trabalho. Sistema: AgendaMaster.
    1. Vá em Arquivo – Importar Fontes e selecione as classes que deseja importar.
    2. Você irá se deparar com uma tela parecida com a tela abaixo, selecione as classes e clique em abrir.
    3. Como as classes estão em pacotes diferentes, você deverá fazer o procedimento acima para todos os pacotes do sistema, para que consiga abrir todas as classes.
    4. A imagem abaixo mostra os pacotes do sistema, você precisa entrar nos pacotes e selecionar as classes que deseja importar.



Agora o ambiente está configurado para que você possa iniciar as tarefas descritas abaixo.

Tarefa 1: Realizar a extração do diagrama de classes relacionado ao módulo de cadastro de

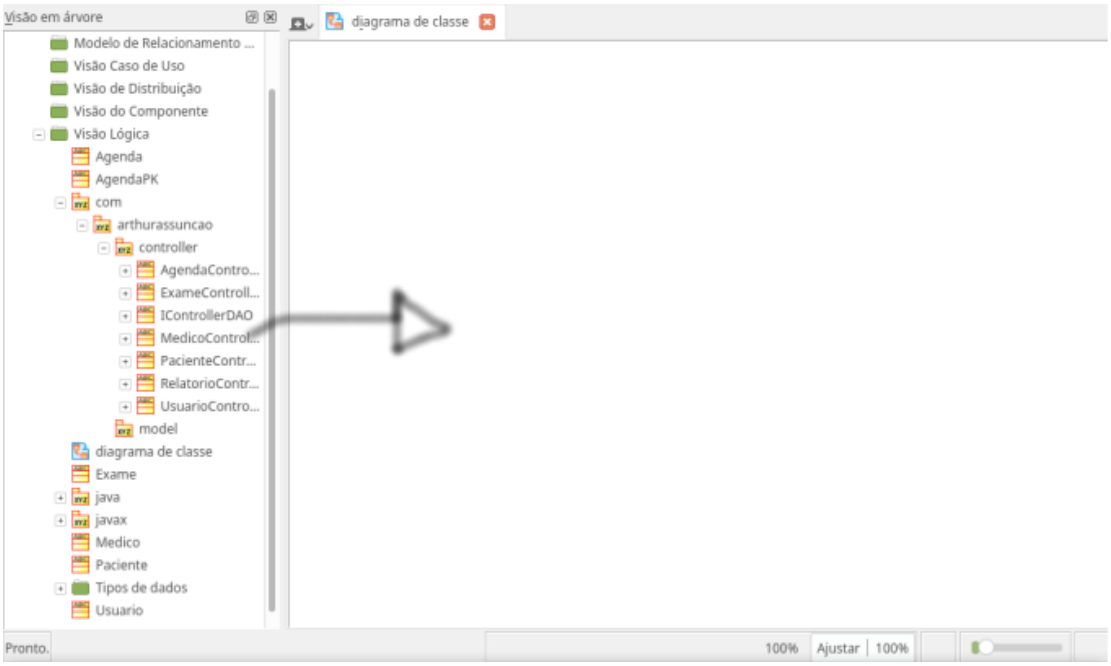
médicos. OBSERVE DICAS!

Passo 1: Identificar e abrir as classes relacionadas ao módulo solicitado.

Passo 2: Gerar o diagrama de classes das classes identificadas.

Passo 3: Exporte o diagrama no formato png e salve na pasta do projeto com a seguinte nomenclatura: seunome-DiagramaTarefa1-Umbrello.

DICAS: Para gerar o diagrama: Clique em cima da classe e arraste para o espaço vazio na tela principal do programa, faça isso com todas as classes relacionadas com o módulo solicitado. Se as classes tiverem ligação, a ferramenta automaticamente ligará as classes.



Tarefa 2: Agora você deve incluir um novo requisito funcional no sistema, o requisito é Cadastrar Secretária. Você deve alterar o diagrama de classes para que ele se adéque ao novo requisito. OBSERVE DICAS!

Requisito **Cadastrar Secretária**:

Descrição: O sistema deverá permitir o cadastro de secretárias, um médico só pode ter uma secretária, e uma secretária só pode atender a um médico.

Ator: Funcionário.

DICAS: Para criar uma nova classe, clique com o botão direito em cima do pacote onde deseja inserir a classe, em seguida clique em criar novo elemento do modelo, depois em nova classe.

Experimento de Engenharia Reversa

APÊNDICE F – *CENÁRIO - ARGOUML*

1

CENÁRIO DO EXPERIMENTO DE ENGENHARIA REVERSA

Imagine que você foi contratado por uma empresa de desenvolvimento de software, para manter um sistema de clínica médica. O sistema já está em produção e os usuários deste sistema pediram que você fizesse algumas modificações no sistema em questão. Para fazer isso você irá utilizar a ferramenta ArgoUML, para extrair o diagrama de classes do sistema, e fazer as modificações necessárias.

O sistema em questão, atende aos seguintes requisitos funcionais:

- Cadastrar médico;
  → **Descrição:** O sistema deverá permitir o cadastro de médicos, solicitando seu nome, endereço, CRM e CPF.
  → **Ator:** Funcionário.
- Cadastrar paciente;
  → **Descrição:** O sistema deverá permitir o cadastro de pacientes, solicitando seu nome, endereço, cpf, histórico hospitalar, telefone e e-mail.
  → **Ator:** Funcionário.
- Marcar consulta;
  → **Descrição:** O sistema deverá permitir o agendamento de consultas, solicitando os dados do paciente e do médico envolvido.
  → **Ator:** Paciente.
- Solicitar exame;
  → **Descrição:** O sistema deverá permitir aos médicos que solicitem exames aos seus pacientes, sendo necessário informar o tipo de exame e o nome do paciente em questão.
- → **Ator:** Médico.
- Visualizar agenda;
  → **Descrição:** O sistema deverá permitir aos médicos visualizarem a sua agenda do dia.
  → **Ator:** Médico.
- Cadastrar agenda.
  → **Descrição:** O sistema deverá permitir o cadastro de agendas, solicitando o nome do médico ao qual a agenda será cadastrada.
  → **Ator:** Funcionário.

Segue abaixo uma imagem do sistema:

Em seguida lhe será entregue um roteiro com as modificações que você deverá fazer no sistema.

Agradeço sua Colaboração.

APÊNDICE G – *CENÁRIO - UMBRELLO*

1

CENÁRIO DO EXPERIMENTO DE ENGENHARIA REVERSA

Imagine que você foi contratado por uma empresa de desenvolvimento de software, para manter um sistema de clínica médica. O sistema já está em produção e os usuários deste sistema pediram que você fizesse algumas modificações no sistema em questão. Para fazer isso você irá utilizar a ferramenta Umbrello UML Modeller, para extrair o diagrama de classes do sistema, e fazer as modificações necessárias.

O sistema em questão, atende aos seguintes requisitos funcionais:

- Cadastrar médico;
  → **Descrição:** O sistema deverá permitir o cadastro de médicos, solicitando seu nome, endereço, CRM e CPF.
  → **Ator:** Funcionário.
- Cadastrar paciente;
  → **Descrição:** O sistema deverá permitir o cadastro de pacientes, solicitando seu nome, endereço, cpf, histórico hospitalar, telefone e e-mail.
  → **Ator:** Funcionário.
- Marcar consulta;
  → **Descrição:** O sistema deverá permitir o agendamento de consultas, solicitando os dados do paciente e do médico envolvido.
  → **Ator:** Paciente.
- Solicitar exame;
  → **Descrição:** O sistema deverá permitir aos médicos que solicitem exames aos seus pacientes, sendo necessário informar o tipo de exame e o nome do paciente em questão.
- → **Ator:** Médico.
- Visualizar agenda;
  → **Descrição:** O sistema deverá permitir aos médicos visualizarem a sua agenda do dia.
  → **Ator:** Médico.
- Cadastrar agenda.
  → **Descrição:** O sistema deverá permitir o cadastro de agendas, solicitando o nome do médico ao qual a agenda será cadastrada.
  → **Ator:** Funcionário.

Segue abaixo uma imagem do sistema:

Em seguida lhe será entregue um roteiro com as modificações que você deverá fazer no sistema.

Agradeço sua Colaboração.