JEFFERSON PENNA DE OLIVEIRA

Uma abordagem para captura automática de links de rastreamento entre especificação de casos de uso e modelos UML

Recife

2015

JEFFERSON PENNA DE OLIVEIRA

Uma abordagem para captura automática de links de rastreamento entre especificação de casos de uso e modelos UML

Dissertação apresentada à Universidade Federal Rural de Pernambuco para obtenção do título de Mestre em Informática Aplicada pelo Programa de Pós-Graduação em Informática Aplicada.

Área de Concentração: Engenharia de Software.

Orientador: Prof. Dr. Gilberto Amado de Azevedo Cysneiros Filho
Co-orientadora: Profa. Dra. Maria da Conceição Moraes Batista

Recife
2015

Ficha Catalográfica

O48a Oliveira, Jefferson Penna de
Uma abordagem para captura automática de links de rastreamento entre especificação de casos de uso e modelos UML / Jefferson Penna de Oliveira. – Recife, 2015.
88 f.: il.

Orientador(a): Gilberto Amado de Azevedo Cysneiros Filho.
Dissertação (Mestre em Informática Aplicada) – Universidade Federal Rural de Pernambuco, Departamento de Estatística e Informática, Recife, 2015.
Referências.

1. Rastreabilidade 2. Engenharia de requisitos 3. Gerência de requisitos 4. Especificação de Casos de Uso 5. UML
I. Cysneiros Filho, Gilberto Amado de Azevedo, orientador
II. Título

CDD 004

Dissertação de autoria de Jefferson Penna de Oliveira, sob o título "**Uma abordagem para captura automática de *links* de rastreamento entre Especificação de Casos de Uso e modelos UML**", apresentada à Universidade Federal Rural de Pernambuco, para obtenção do título de Mestre em Informática Aplicada pelo Programa de Pós-Graduação em Informática Aplicada, na área de concentração Engenharia de Software, aprovada em 27 de agosto de 2015 pela comissão julgadora constituída pelos doutores:

Profa. Dra. Maria da Conceição Moraes Batista

Presidente

Instituição: Universidade Federal Rural de Pernambuco

Prof. Dr. Fernando Antônio Aires Lins

Instituição: Universidade Federal Rural de Pernambuco

Profa. Dra. Maria Lencastre Pinheiro de Menezes Cruz

Instituição: Universidade de Pernambuco

Profa. Dra. Carla Taciana Lima Lourenço Silva Schuenemann

Instituição: Universidade Federal de Pernambuco

Dedico este trabalho aos meus pais, Orlando Oliveira e Isabel Penna, que mesmo em momentos difíceis, nunca abriram mão da minha educação.

Agradecimentos

Primeiramente a Deus, por sempre ter me amparado nos momentos de dificuldade e desespero e por ter me dado forças para conseguir chegar ao fim desta jornada.

À minha família, pelo apoio e por compreenderem meu período distante; aos meus pais que, com pequenas atitudes, sempre estiveram me auxiliando; em especial, à minha noiva, Fernanda Camila, que não me deixou desistir do sonho de me tornar mestre e ter transformado momentos de angústia e tristeza em momentos de força e alegria.

Ao meu orientador Prof. Dr. Gilberto Cysneiros e à minha coorientadora Profa. Dra. Maria da Conceição Moraes, por toda a disponibilidade, incentivo e dedicação, por acreditarem em mim e por me conduzirem ao caminho da ciência.

A todos os professores e amigos do curso, que ajudaram de forma direta e indireta na conclusão deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro.

Enfim, deixo registrado meus agradecimentos a todos os que contribuíram, direta e indiretamente, para que eu realizasse esta pesquisa, auxiliando-me e dando-me forças nos momentos em que mais precisei.

Resumo

Durante a especificação e documentação do software a ser criado, é comum a criação de vários artefatos que descrevem suas funcionalidades e a interação do sistema com seus usuários e outros sistemas. Documentar e entender o relacionamento entre esses artefatos é fundamental para verificar se o software desenvolvido atende a todos os requisitos especificados. Criar esses relacionamentos manualmente é uma tarefa repetitiva e propensa a erros. Além disso, para sistemas de grande porte a quantidade de relacionamentos pode ser grande e complexa. A fim de minimizar estes problemas, esta dissertação apresenta uma abordagem de rastreamento para identificar relacionamentos de rastreamento automaticamente. Rastreamento vem sendo considerada uma atividade importante no processo de desenvolvimento de software e sua implantação tende a melhorar a qualidade do software, além de apoiar as atividades de: i) reuso de código, ii) análise de impacto e iii) verificação de consistência. A abordagem apresentada nessa dissertação é baseada em regras que gera relações de rastreabilidade entre os documentos da Especificação de Casos de Uso e Diagrama de Casos de Uso UML. As regras também identificam incompletude e inconsistências entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso UML. Um modelo de referência dos relacionamentos de rastreamento entre elementos da Especificação de Casos de Uso e Diagramas de Casos de Uso UML foi proposto. Os artefatos (Especificação de Casos de Uso e Diagrama de Casos de Uso UML), são representados em XML, pois este é um padrão de fato para armazenamento e transferência de dados em ferramentas CASE. As regras são escritas em XQuery para (i) apoiar a execução de ações quando uma condição for satisfeita; (ii) identificar relações de rastreabilidade; e (iii) analisar a consistência entre os artefatos. Foi desenvolvida uma ferramenta protótipo para avaliar a abordagem. A abordagem foi avaliada com a especificação de dois sistemas e usando as métricas de *precision* e *recall*.

Palavras-chave: Rastreabilidade. Engenharia de Requisitos. Gerência de Requisitos. Especificação de Casos de Uso. UML.

Abstract

During the specification and documentation of the software to be developed is common to be created various artifacts that describe its functionalities and the system's interaction with its users and other systems. To document and understand the relationship between these artifacts helps to verify that the software developed meets all specified requirements. Create these relationships manually is a repetitive task and prone to errors. Furthermore, for large systems the amount of relationships can be large and complex. To minimize these problems, this dissertation presents an approach to identify traceability relationships automatically. Traceability has been considered an important activity in software development process and its implementation tends to improve software quality, and support the following activities: i) code reuse, ii) impact analysis and iii) consistency check. The approach presented in this thesis is based on rules that generates traceability relationships between Use Cases Specification document and UML Use Case diagram. The rules also identify inconsistencies and incompleteness of the Use Case Specification document and UML Use Case Diagram. A reference model of traceability relationships between elements of Use Cases Specification and UML Use Case Diagrams has been proposed. Artifacts (Use Cases Specification and UML Use Case Diagram), are represented in XML, for this is a de facto standard for data storage and transfer in CASE tools. Rules are written in XQuery to (i) support the implementation of actions when a condition is met; (ii) identify traceability relationships; and (iii) analyze the consistency between artefacts. One prototype tool was developed to evaluate the approach. The approach was evaluated by two systems specification and using the metrics of precision and recall.

Keywords: Traceability. Requirements Engineering. Requirements Management. Use Case Specification. UML.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

UML	Unified Modeling Language
XML	eXtensible Markup Language
OMG	Object Management Group
OMT	Object Modeling Language
OOSE	Object-Oriented Software Engineering
POO	Programação Orientada a Objetos

Sumário

1 Introdução

Todo software deve atender às necessidades dos clientes (PRESSMAN, 2011). Os requisitos de software definem os objetivos, funcionalidades, propriedades e restrições que o software deve implementar. A qualidade de um software pode ser medida através da conformidade com os requisitos definidos pelo cliente.

Chung e Leite (2009) dividem os requisitos em duas categorias: requisitos funcionais e requisitos não-funcionais. Os requisitos funcionais descrevem as funcionalidades de um sistema, ou seja, o que ele será capaz de fazer (GLINZ, 2007). Os requisitos não-funcionais descrevem propriedades e restrições, tais como: segurança, linguagem de programação ou sistema operacional (MARTINS, 2010).

Durante o processo de desenvolvimento de software, muitos artefatos são criados para documentar as necessidades que o software precisará atender, e quais funcionalidades ele deverá prover. O diagrama de Casos de Uso UML (OMG, 2015), por exemplo, é uma representação da interação dos atores (usuários e outros sistemas) com o sistema.

O documento de Especificação de Casos de Uso (COCKBURN, 1998), por sua vez, deve registrar, em texto formal, os requisitos funcionais descrevendo um guia passo-a-passo para cada uma das funcionalidades do software. Este guia conterá o fluxo de interações entre o usuário e o software (ou procedimentos internos do software), definindo um fluxo de sucesso para a funcionalidade, ou ainda como o software deverá se comportar em situações alternativas.

O diagrama de Casos de Uso UML e a Especificação de Casos de Uso possuem elementos que são relacionados. Por exemplo, atores no diagrama de Casos de Uso UML e na Especificação de Casos de Uso tem a mesma semântica. Assim, caso exista um ator no diagrama de Casos de Uso UML e na Especificação de Casos de Uso com o mesmo nome espera-se que esses dois elementos sejam relacionados.

Relacionar os artefatos criados durante as fases de concepção e desenvolvimento de software é um processo importante, pois contribui para várias atividades do desenvolvimento de software, tais como: reuso, análise de impacto, análise de consistência, compreensão do software, gerenciamento de projeto e gerenciamento de configuração (FILHO, 2011).

É possível identificar os relacionamentos entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso UML manualmente, contudo alguns fatores podem dificultar esse processo.

Em projetos que contenham artefatos grandes e complexos, um processo de relacionamento manual pode ser uma atividade exaustiva e propensa a erros (SPANOUDAKIS; ZISMAN, 2004). Outro problema em projetos desse tipo, é que ocorrem mudanças nesses artefatos durante as fazes de manutenção do software, e nem sempre estas são replicadas em todos os artefatos, gerando assim inconsistências entre eles. Analisar a consistência entre esses artefatos, apesar de importante, não é uma atividade fácil de ser realizada manualmente em projetos grandes e complexos. Assim é necessário o uso de uma abordagem capaz de relacionar automaticamente os artefatos criados durante o processo de desenvolvimento Orientado a Objetos.

Nesta dissertação, é apresentada uma abordagem de rastreabilidade para à identificação automática de relacionamentos existentes entre os elementos da Especificação de Casos de Uso e do Diagramas de Casos de Uso UML. A abordagem também dá suporte à verificação de consistência entre os artefatos analisados.

Rastreabilidade de Software é definida como a capacidade de se descrever e seguir a vida de um artefato para trás, partindo de suas origens e passando pelo seu desenvolvimento e especificação, e para frente, observando implantações subsequentes e uso, permitindo responder a perguntas sobre o produto de software e seu processo de desenvolvimento (GOTEL; FINKELSTEIN, 1994), (COEST, 2015).

Existem outros diagramas UML, além do Diagrama de Casos de Uso, que podem ser usados para detalhar os requisitos de um sistema, como por exemplo os Diagramas de Sequência, Estado e Atividade. Em virtude da similaridade estrutural entre o Diagrama de Casos de Uso com os mencionados acima (Diagramas de Sequência, Estado e Atividade), acredita-se que a abordagem criada nessa pesquisa também pode ser aplicada para relacionar esses diagramas ao documento de Especificação de Casos de Uso, contudo optou-se por validar essa hipótese em trabalhos futuros.

A escolha da Especificação de Casos de Uso, deu-se por, mesmo sendo um documento expresso em linguagem natural, possuir uma semântica estrutural bem definida, quando utilizando *templates* (ARANTES, 2010). Além disso, ele é bastante utilizado tanto na indústria como também na academia. UML, por sua vez, é uma linguagem padrão para modelagem de software e seu Diagrama de Casos de Uso é usado para representar graficamente as funcionalidades do software através de diagramas que contém: atores, casos de uso e relacionamentos.

Em virtude da heterogeneidade dos artefatos que serão analisados nessa pesquisa, esta abordagem utilizará o padrão *eXtensible Markup Language* – XML (XML, 2015) como

formato padrão de entrada para os artefatos. A escolha do XML ocorreu devido a várias razões: (a) o XML é uma linguagem padrão usada na indústria para comunicação e troca de informações entre sistemas distintos; (b) muitas ferramentas CASEs armazenam seus dados em XML e permitem a exportação dos dados nesse formato e (c) para definir e executar regras de rastreabilidade sobre documentos XML utilizando a linguagem de consulta XQuery (XQUERY, 2015).

Esta pesquisa utiliza uma abordagem orientada a regras, pois o relacionamento entre uma Especificação de Casos de Uso e o Diagrama de Casos de Uso é bem definido e conhecido. Além disso, esta pesquisa é uma extensão da abordagem de FILHO (2011) que foi aplicada para identificar, de forma automática, relações de rastreabilidade entre artefatos de softwares durante o processo de desenvolvimento de sistemas multiagentes.

XQuery é uma linguagem de consulta baseada em XML, que vem sendo amplamente utilizada para realização de recuperação de informações, manipulação de dados e consulta a documentos XML. O XQuery possui uma grande quantidade de funções nativas, mas também permite a inclusão de funcionalidades extras, permitindo assim que estas também sejam utilizadas nas regras criadas por esta abordagem.

Uma ferramenta protótipo foi desenvolvida para apoiar esta abordagem. A avaliação desta pesquisa ocorreu através da comparação dos relacionamentos de rastreamento obtidos manualmente (supostamente um conjunto completo de todos relacionamentos de rastreamento e sem relacionamentos incorretos) e pelos relacionamentos de rastreamento gerados automaticamente pela ferramenta protótipo. Para a avaliação foram escolhidos dois exemplos de aplicação o (i) BurguerDigital e (ii) Projeto Pesquisa e Movimento. O BurguerDigital propõe uma solução para gerenciamento, controle de estoque e venda de hambúrguer. O Projeto Pesquisa e Movimento é um sistema destinado a auxiliar o processo de solicitação e reservas de veículos por pesquisadores de uma instituição de ensino, dispondo também de métodos de análise e gerenciamento dessas solicitações.

Como esta pesquisa comparará os resultados das relações de rastreabilidade gerados a partir da ferramenta protótipo com os resultados tidos como correto (relacionados manualmente) as métricas *precision* e *recall* (ALI; GUEHENEUC; ANTONIOL, 2011) foram escolhidas para validar a relevância dos relacionamentos gerados pela ferramenta protótipo. O *precision* é obtido da relação entre o número de elementos recuperados que são relevantes (ou seja, elementos recuperados corretamente pela ferramenta protótipo), com o número total de elementos recuperados (ou seja, elementos recuperados correta e incorretamente pela ferramenta protótipo). O *recall* é obtido da relação entre o número de elementos recuperados

que são relevantes (ou seja, elementos recuperados corretamente pela ferramenta protótipo), com o número total de elementos relevantes (ou seja, todos os elementos identificados manualmente e que deveriam ser encontrados também pela ferramenta protótipo).

O restante deste capítulo é seguido pela descrição da hipótese, objetivos, contribuições esperadas e estrutura da dissertação.

1.1 Hipótese

A hipótese dessa dissertação é que é possível utilizar uma abordagem orientada a regras para identificar automaticamente relações de rastreabilidade entre elementos da Especificação de Casos de Uso e do Diagrama de Casos de Uso UML, além de identificar inconsistências entre estes artefatos.

1.2 Objetivos

O objetivo geral dessa pesquisa é desenvolver uma abordagem de rastreabilidade capaz de identificar automaticamente relacionamentos entre os elementos dos documentos de Especificação de Casos de Uso (COCKBURN, 1998) com os do Diagrama de Casos de Uso UML (OMG, 2015).

O objetivo principal foi dividido nos seguintes objetivos específicos:

- Definição dos tipos de relações de rastreabilidade existente entre os elementos dos artefatos de análise;
- Criar um modelo de referência que contenha as relações de rastreabilidade identificados no objetivo acima.
- Criar um conjunto de regras capaz de identificar relacionamentos entre os elementos dos documentos de Especificação de Casos de Uso e Diagrama de Caso de Uso UML.
- Criar um conjunto de regras capaz de identificar inconsistências existentes entre os documentos de Especificação de Casos de Uso e Diagrama de Caso de Uso UML.
- Criar uma ferramenta protótipo que implemente a abordagem proposta nesta pesquisa;

- Avaliar a eficácia desta pesquisa, por meio de exemplos de aplicações, utilizando a ferramenta protótipo e aferindo o índice de relevância das relações de rastreabilidade, por meio das métricas *precision* e *recall*.

1.3 Contribuições esperadas

Pretende-se com este trabalho contribuir da seguinte forma:

- Identificação automática das relações de rastreabilidade – Proposta de uma abordagem orientada a regras para relacionar automaticamente elementos da Especificação de Casos de Uso e Diagrama de Casos de Uso.

- Modelo de referência de rastreabilidade – Apresentação de tipos de relacionamentos entre elementos da Especificação de Casos de Uso e elementos do Diagrama de Casos de Uso UML;

- Elaboração de regras para identificar relacionamentos de rastreamento entre elementos da Especificação de Casos de Uso e Diagrama de Casos de Uso UML;

- Elaboração de um conjunto de regras para verificar inconsistências entre elementos da Especificação de Casos de Uso e Diagrama de Casos de Uso UML;

- Ferramenta protótipo de rastreabilidade – Implementação de uma ferramenta capaz de executar as regras previamente criadas, exibindo posteriormente os resultados dos relacionamentos e inconsistências encontradas durante a execução das regras.

1.4 Estrutura da dissertação

A seguir é descrita a forma como este trabalho está estruturado.

No Capítulo 2 a fundamentação teórica é abordada. Os principais conceitos da engenharia de requisitos, relacionados a dissertação, são apresentados.

O Capítulo 3 aborda os trabalhos relacionados.

O Capítulo 4 descreve o modelo de referência proposto.

O Capítulo 5 expõe a abordagem e as regras criadas são detalhadas nesse capítulo.

O Capítulo 6 discute a avaliação da abordagem.

Por fim, o Capítulo 7 aponta as conclusões, tecendo comentários sobre as contribuições e propostas de trabalhos futuros.

2 Fundamentação teórica

Ao longo do ciclo de vida de um software é comum ocorrer mudanças nos requisitos, ora surge uma nova demanda, ora outra precisa ser modificada. Esse fato ocorre devido alguns fatores como, por exemplo, a pouca maturidade do cliente sobre o problema ou por erros em decisões técnicas. Em muitos desses casos as modificações realizadas podem acarretar efeitos colaterais que se propagam negativamente a outras partes do software. Dessa forma é preciso avaliar e controlar todo o processo de mudança dos requisitos, para isso é preciso utilizar técnicas de gerência de requisitos.

Com o uso de técnicas de gerência de requisitos é possível realizar as modificações necessárias ao software de maneira eficiente a baixo custo (SAYÃO; LEITE, 2006). O gerenciamento de requisitos é o processo utilizado para compreender e controlar todas as mudanças nos requisitos do software. Sommerville (2007) fala sobre a importância de seguir não apenas um requisito, mas também relacioná-lo com todos os seus dependentes.

Por meio do processo de gerenciamento de requisitos é possível identificar os relacionamentos existentes entre (i) requisitos e projeto, (ii) requisitos e fontes de sua origem, contendo os motivos que levaram a sua criação e (iii) entre requisitos, apontando suas dependências. A rastreabilidade é uma propriedade que auxilia no desenvolvimento e na manutenção do software e estudos apontam que sua implantação gera maior probabilidade em se construir um software dentro do custo estimado e de melhor qualidade (GOTEL *et al*., 2012a).

2.1 Rastreabilidade

Rastreabilidade de software é definida por Gotel e Finkelstein (1994) como a habilidade de descrever e seguir o ciclo de vida de um requisito em ambas direções, para trás e para frente, ou seja, desde seu origem, especificação e desenvolvimento, até sua implantação, utilização e refinamentos.

A rastreabilidade exerce um importante papel no processo de desenvolvimento de software auxiliando atividades como análise de impacto, validação e verificação, além de apoiar as fases de evolução e manutenção (SPANOUDAKIS; ZISMAN, 2004), (GOTEL *et al*., 2012a), (KANNENBERG; SAIEDIAN, 2009). Em virtude dos benefícios citados

anteriormente a rastreabilidade vem sendo considerada como um elemento essencial ao processo de desenvolvimento de software (CLELAND-HUANG *et al*., 2014).

Apesar da sua importância, a rastreabilidade de software ainda possui problemas e desafios que precisam ser solucionados (GOTEL *et al*., 2012b). Gotel *et al*. (2012b) apontam em sua pesquisa oito desses problemas e desafios: (i) suportar todas as finalidades e necessidades dos *stakeholders*; (ii) os custos da implantação da rastreabilidade ainda precisam ser justificados por meio dos benefícios esperados; (iii) a rastreabilidade precisa ser facilmente adaptada ao surgir novas exigências dos *stakeholders*; (iv) é preciso que todos os *stakeholders* compreendam a necessidade e benefícios da rastreabilidade, além de confiar nos *links* de rastreamento gerados por uma ferramenta; (v) é preciso rastrear artefatos distintos em diversos níveis de granularidade; (vi) é preciso criar mecanismos para facilitar o reuso da rastreabilidade, permitindo simples adaptações para outros projetos; (vii) a rastreabilidade deve possuir um papel estratégico no processo de desenvolvimento de software e deve ser valorizada por todos os *stakeholders*; e (viii) a rastreabilidade precisa se tornar ubíqua, requerendo para isso a execução de todos os objetivos/desafios citados anteriormente.

Para solucionar esses problemas e desafios é preciso utilizar alguma abordagem de rastreabilidade. Existem três categorias de abordagem de rastreabilidade, a manual, as semiautomáticas e as automáticas. A escolha por uma abordagem de rastreabilidade vai depender do tipo de cada projeto. A abordagem manual não é a mais indicada em projetos grandes e complexos, assim a implantação de uma abordagem automática ou semiautomática de rastreabilidade deverá gerar um melhor custo-benefício (SAIEDIAN; KANNENBERG; MOROZOV, 2013). Estudos como o de (POHL, 1996) e Sherba (SHERBA, 2005) apontam que uma abordagem de rastreabilidade eficiente necessita responder as seguintes questões:

- Quais informações precisam ser capturadas?
- Como as informações serão capturas?
- De quais maneiras os resultados serão armazenados?
- Como será possível visualizar e consultar os resultados obtidos?

2.2 Abordagens de rastreabilidade para captura de *links* de relacionamentos

Diversas ferramentas dão suporte à captura de relações de rastreabilidade entre os artefatos criados durante o processo de desenvolvimento de software, como é o caso da *Rational DOORS* (IBM Rational) (IBM, 2015) e *CaliberRM* (FOCUS, 2015). Contudo essas

ferramentas ainda não conseguem gerar as relações de rastreabilidade de forma automatizada. Em alguns casos é possível importar a lista de requisitos a partir de documentos escritos em editores de texto ou a partir de palavras-chaves. Porém, após a importação desses artefatos é necessário que o usuário crie os relacionamentos manualmente. Porém, como foi dito anteriormente, a identificação manual de relações de rastreabilidade é um procedimento caro, trabalhoso e propenso a erros (SPANOUDAKIS; ZISMAN, 2004), (LUCIA; OLIVETO; TORTORA, 2008).

O custo atribuído à criação manual de relacionamentos de rastreabilidade varia de acordo com o entendimento do domínio e da complexidade do projeto, portanto, o esforço necessário não pode ser atribuído considerando apenas a experiência prévia. Devido a essa dificuldade em orçar um custo a esse processo, a utilização de uma abordagem manual pode elevar consideravelmente o orçamento do projeto, de modo que os benefícios gerados a partir da rastreabilidade não justifiquem a sua implantação.

Para resolver esse problema várias abordagens estão sendo propostas na literatura para automatizar a geração de relações de rastreabilidade.

2.2.1 Abordagem orientada a processos

As abordagens orientadas a processos, são úteis quando se tem um processo de desenvolvimento amadurecido. Quando implantada, pode ser considerada um produto do processo de desenvolvimento de software, porém esta abordagem não foca em ferramentas de suporte à integração e processos unificados de desenvolvimento de software.

2.2.2 Abordagem formal

As abordagens formais definem os artefatos de software e suas relações através de uma linguagem formal. Estas utilizam axiomas e expressões no processo de identificação das relações de rastreabilidade (PINHEIRO; GOGUEN, 1996). O principal problema dessa abordagem é a necessidade de se ter formação e forte conhecimento em alguma linguagem formal.

2.2.3 Abordagem baseada em técnicas de recuperação da informação

As abordagens baseadas em técnicas de recuperação da informação tratam problemas de representação, armazenamento, organização e posterior acesso a essas informações. A recuperação da informação é capaz de obter dados relevantes em documentos textuais não estruturados e em alguns casos ambíguos (BAEZA-YATES; RIBEIRO-NETO *et al*., 1999).

Para Oliveto (2008), um processo de recuperação da informação é inicializado quando um usuário executa uma consulta a um sistema. Os dados passados nessa consulta são considerados como uma declaração formal que expressa a necessidade do usuário, como por exemplo, uma foto, um título de um livro, um resultado de uma partida de futebol. Na recuperação da informação uma consulta não retorna apenas um objeto existente em uma coleção, mas vários objetos de diversas coleções, como um documento textual, uma foto ou vídeo, enfim, tudo aquilo que pode fazer referência à consulta.

A principal desvantagem dessa abordagem para identificar relações de rastreabilidade é que apenas os conteúdos dos documentos são considerados relevantes, desconsiderando possíveis benefícios que poderiam ser alcançados se fosse analisado também a estrutura do documento.

2.2.4 Abordagem orientada a regras

As abordagens orientadas a regras criam relações de rastreabilidades entre elementos de artefatos quando uma determinada condição é satisfeita. O problema dessa abordagem é que é necessário definir inicialmente todos os elementos que se deseja relacionar. Só após este processo será possível criar as regras que contemplem a busca por todos os elementos identificados anteriormente. Contudo, nem sempre a identificação dos elementos e criação das regras é uma tarefa trivial (FILHO, 2011).

2.2.5 Abordagem baseada em informações de código em tempo de execução

Esse tipo de abordagem relaciona código-fonte com outros artefatos (ex.: Especificação de Casos de Uso, ou cenário) baseado em informações de código em tempo de execução. O problema dessa abordagem é que a rastreabilidade ocorre apenas após a

construção dos artefatos não durante o processo de construção, portanto elas são mais indicadas para as fases de manutenções.

2.3 Modelos de referência de rastreabilidade

Os modelos de referência de rastreabilidade são estruturas criadas e utilizadas para demonstrar quais os tipos de informações serão capturados em determinada abordagem e como estão relacionadas.

Existem diversas propostas de classificações e modelos de referência de rastreabilidade (DAVIS, 1990), (LINDVALL; SANDAHL, 1996), (FILHO, 2011), (GHAZI, 2008), (KASSAB; ORMANJIEVA; DANEVA, 2008), (DALL'OGLIO; SILVA; PINTO, 2010).

Davis (1990) classificou a rastreabilidade quanto à sua direção, ou seja, para frente (*forward*) e para traz (*backward*). A *forward* é a capacidade de relacionar artefatos após sua criação, sendo então aproveitadas para as fases de implantação, utilização e refinamentos. O *backward*, por sua vez, é a capacidade de relacionar artefatos nas fases iniciais, desde seu conhecimento até a especificação e desenvolvimento. Lindvall e Sandahl (1996) destacam mais duas classificações de rastreabilidade, a rastreabilidade horizontal e a vertical. A rastreabilidade horizontal refere-se aos relacionamentos criados dentro de um mesmo documento, enquanto a rastreabilidade vertical trata de relacionamentos criados a partir de diferentes artefatos do processo de desenvolvimento de software.

Filho (2011) propôs um modelo de referência definindo os tipos de relacionamentos existente entre os artefatos criados ao longo do processo de desenvolvimento de sistemas multiagentes. Uma ferramenta protótipo que implementa o modelo proposto foi desenvolvida para demonstrar o processo de rastreamento entre os modelos construídos em *i**, *Prometheus* e o código-fonte escrito na linguagem *JACK*. O processo de rastreabilidade ocorre por meio da construção e execução de regras escritas em XQuery e o resultado final da rastreabilidade é exibido por meio de uma matriz de rastreabilidade.

Ghazi (2008) defende uma abordagem de rastreabilidade gerenciada através de múltiplas perspectivas. Seu modelo, que foi construído em UML, permite representar atores e papéis envolvidos no projeto, além de permitir o relacionamento desses com os artefatos do software. O modelo permite relacionar elementos de diferentes fases do processo de desenvolvimento de software, suportando rastreabilidade horizontal e vertical. Os relacionamentos gerados a partir do modelo possuem uma nomenclatura própria para indicar

o tipo de relacionamento. O modelo proposto por Ghazi (2008) permite representar a evolução dos *links* de rastreabilidade, porém não faz referência a parte de análise de impacto.

Kassab, Ormanjieva e Daneva (2008) citam que em muitos casos a implantação da rastreabilidade foca apenas nos requisitos funcionais, o que consideram um erro. Assim, apresentam um modelo de rastreabilidade capaz de rastrear os requisitos funcionais e não-funcionais de um software. Na abordagem apresentada, um requisito funcional está relacionado a um modelo, que está relacionado a uma fase do ciclo de vida do software. O requisito não-funcional, por sua vez, está relacionado a um ou mais requisitos funcionais. Esse modelo de rastreabilidade é capaz de realizar a rastreabilidade vertical e horizontal, pois o rastreamento pode ser realizado entre os artefatos independentemente da versão ao qual pertença.

Dall'oglio, Silva e Pinto (2010), apresentam um modelo de rastreabilidade criado a partir da unificação de extensão de aspectos encontrados em modelos desenvolvidos anteriormente por outros pesquisadores. O modelo apresentado suporta a configuração de granularidade e as atividades de gerenciamento de mudanças e análise de impacto. Esse modelo consegue relacionar artefatos criados durante todas as fases do processo de desenvolvimento de software.

Assim, ao desenvolver uma nova abordagem de rastreabilidade é necessário criar, ou apresentar, um modelo de rastreabilidade que detalhe. Esse modelo de rastreabilidade deverá detalhar, para cada artefato de análise, todos os elementos passíveis de relacionamento. Deverá também descrever os tipos de relacionamentos existentes e em seguida identificar como os elementos pré-identificados estão relacionados uns com os outros.

2.4 Artefatos de documentação

A seguir são apresentados os artefatos produzidos durante o processo de desenvolvimento de software que são alvo desta pesquisa.

2.4.1 Documento de Especificação de Casos de Uso

O documento de Especificação de Casos de Uso foi proposto inicialmente por Ivar Jacobson (Jacobson, 1992), sendo posteriormente aprimorado por outros autores, por exemplo, Alistair Cockburn (COCKBURN, 1999).

O documento de Especificação de Casos de Uso é o artefato que deverá conter os fluxos necessários para atender cada uma das necessidades do cliente. Ou seja, ele detalha todas as funcionalidades, destrinchando todas as interações que possam ocorrer entre o usuário e o software (ou fluxos internos do software) necessárias para atingir o sucesso da funcionalidade. Este documento também detalha os passos necessários para situações alternativas (situações diferentes do fluxo de sucesso). É importante que as descrições das funcionalidades sejam consistentes e não ambíguas, pois elas servirão de apoio para o planejamento, entendimento, desenvolvimento e testes do software.

O objetivo do documento de Especificação de Casos de Uso é apontar todos os requisitos funcionais do sistema, descrevendo as funcionalidades no formato de interação entre o sistema e atores, este último pode ser um usuário (humano), outro software, ou um componente de hardware. O documento de Especificação de Casos de Uso influência muitos outros artefatos de análise, projeto, implementação, teste e gestão de projeto.

No documento de Especificação de Casos de Uso, cada funcionalidade é descrita usando um *template*, onde deverão ser detalhadas as informações de como acontecerá as interações entre os atores e o sistema, com o intuito de exibir quais passos o ator precisa realizar para concluir uma tarefa com sucesso, ou o que ocorrerá em casos excepcionais.

Além do fluxo de interação, o *template* da Especificação de Casos de Uso possui outras informações relevantes como, por exemplo, precondições e pós-condições, que respectivamente determinam, as condições que precisam ser satisfeitas para que a funcionalidade seja inicializada e qual o resultado oriundo da finalização da execução da funcionalidade. *Templates* de Especificação de Casos de Uso foram propostos por vários autores, tais como (COCKBURN, 1998), (COCKBURN, 1999), (ROBERTSON; ROBERTSON, 2000), (TORO *et al.*, 1999) e (SINDRE; OPDAHL, 2001).

A pesquisa dos *templates* de Especificação de Casos de Uso apontou diversos elementos em comum entre eles. Contudo, outros elementos eram peculiares de cada pesquisa. Os elementos existentes em cada *template* encontrado foram analisados quando sua semântica e relevância para a compreensão histórica e de fluxo de execução do requisito. A Tabela 1 apresenta o *template* de Especificação de Casos de Uso criado com todos os elementos apurados durante a pesquisa e foram considerados relevantes para esta abordagem.

Tabela 1 - *Template* de especificação de requisitos

Nome			
Identificador		Autor	

Prioridade		Ator	
Criado em		Modificado em	
Descrição			
Requisitos não-funcionais			
Pré-condições			
Pós-condições			
Fluxo principal			
Subfluxo principal			
Fluxo secundário			
Fluxo de exceção			

Este *template* conta com quinze elementos: Nome, identificador, autor, prioridade, ator, criado em, modificado em, descrição, requisitos não-funcionais, pré-condição, pós-condição, fluxo principal, subfluxo principal, fluxo secundário e fluxo de exceção. Abaixo é descrito cada um desses elementos.

O **Nome** é representado no formato verbo/substantivo, de modo que seja suficiente para citar a que o caso de uso se refere.

O **Identificador** é uma referência única ao caso de uso. Normalmente, os requisitos funcionais são identificados pela concatenação de abreviação RF (Requisito Funcional) com uma sequência de três dígitos para representar o requisito em questão, por exemplo RF001.

Autor é a pessoa que documentou este requisito.

Prioridade é o grau de importância desse requisito no sistema. É comum definir as prioridades de três formas: essenciais, para aqueles requisitos indispensáveis ao sistema e que sem elas o software não funciona; importante, para os requisitos que o software precisa ter, mas que sem eles o software ainda entra em funcionamento, porém de forma não satisfatória; e desejáveis, para requisitos de menor valor, esses requisitos não comprometem as funcionalidades básicas do sistema, podendo ser desenvolvidos em versões posteriores, assim o software pode ser implantado de forma satisfatória.

O **Ator** refere-se a todos os usuários ou qualquer outro sistema/hardware que irão interagir com o requisito em questão.

Criado em diz respeito a data na qual o requisito foi documentado pela primeira vez.

Modificado em é a data em que o requisito em questão sofreu a última alteração.

A **Descrição** deve informar brevemente qual o intuito do requisito.

Requisito não-funcional informa necessidades como desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias que o requisito deve atender.

Pré-condições informa quais condições, obrigatoriamente, deverão ser atendidas para que o requisito seja inicializado.

Pós-condição informa qual o estado, ao final da execução, que o requisito precisará estar para ser tido como realizado.

Fluxo principal descreve, passo-a-passo, quais ações precisam ser executas pelo ator e pelo sistema para que o principal objetivo do requisito seja alcançado;

Subfluxo principal é descrito quando o fluxo principal é complexo ou muito extenso. Este possui as mesmas características do fluxo principal.

Fluxo secundário descreve os passos alternativos que o usuário pode tomar.

Por fim, o **Fluxo de exceção** é a descrição dos tratamentos de exceção que podem surgir nos fluxos anteriores.

2.4.2 Diagramas UML

A Linguagem de Modelagem Unificada (UML) é definida pelo *Object Management Group* (OMG) (OMG, 2015) como uma linguagem visual para especificar, construir e documentar os artefatos de um sistema. A UML é considerada visual por ser uma notação gráfica, apoiada por um metamodelo, que se tornou padrão (LARMAN, 2002) no auxílio a documentação e no projeto de software, em especial aqueles projetados e desenvolvidos com o paradigma de programação orientado a objetos.

A UML nasceu a partir da unificação de um grupo de linguagens de modelagem gráfica de sucesso. A primeira versão da UML, de número 0.8, conhecida até então como Processo Unificado, surgiu da união dos métodos *Booch* e *Object Modeling Language* (OMT), em 1995. Logo depois, no ano de 1996, o método *Object-Oriented Software Engineering* (OOSE) foi incorporado ao processo unificado, que passou a ser chamado de UML, em versão 0.9. No ano seguinte, o OMG aprovou a UML como uma linguagem padrão de modelagem.

O processo de modelagem e documentação de um projeto pode ser complexo e quando não são escritos sobre determinado padrão pode trazer ambiguidade e problemas durante as fases de desenvolvimento, validação e testes de um sistema.

A criação de diagramas em UML ajudam a dar uma formalidade às regras de negócio do sistema, de forma não ambígua, completa e padronizada, permitindo assim um maior entendimento do projeto por parte da equipe envolvida.

A UML, que hoje encontra-se na versão 2.4, possui 14 diagramas oficiais, classificados em (i) Diagramas de estruturas; (ii) Diagrama de comportamento e (iii) Diagrama de interações. **Diagramas de estruturas** servem para visualizar, especificar, contribuir e documentar os aspectos estáticos de um sistema. **Diagramas comportamentais** são aqueles que levam em consideração a possibilidade de ocorrer alterações no comportamento das classes. **Diagramas de interação** descrevem como grupos de objetos contribuem para o sucesso de um determinado comportamento. Os diagramas existentes em cada uma dessas classificações podem ser visualizados na Figura 1.

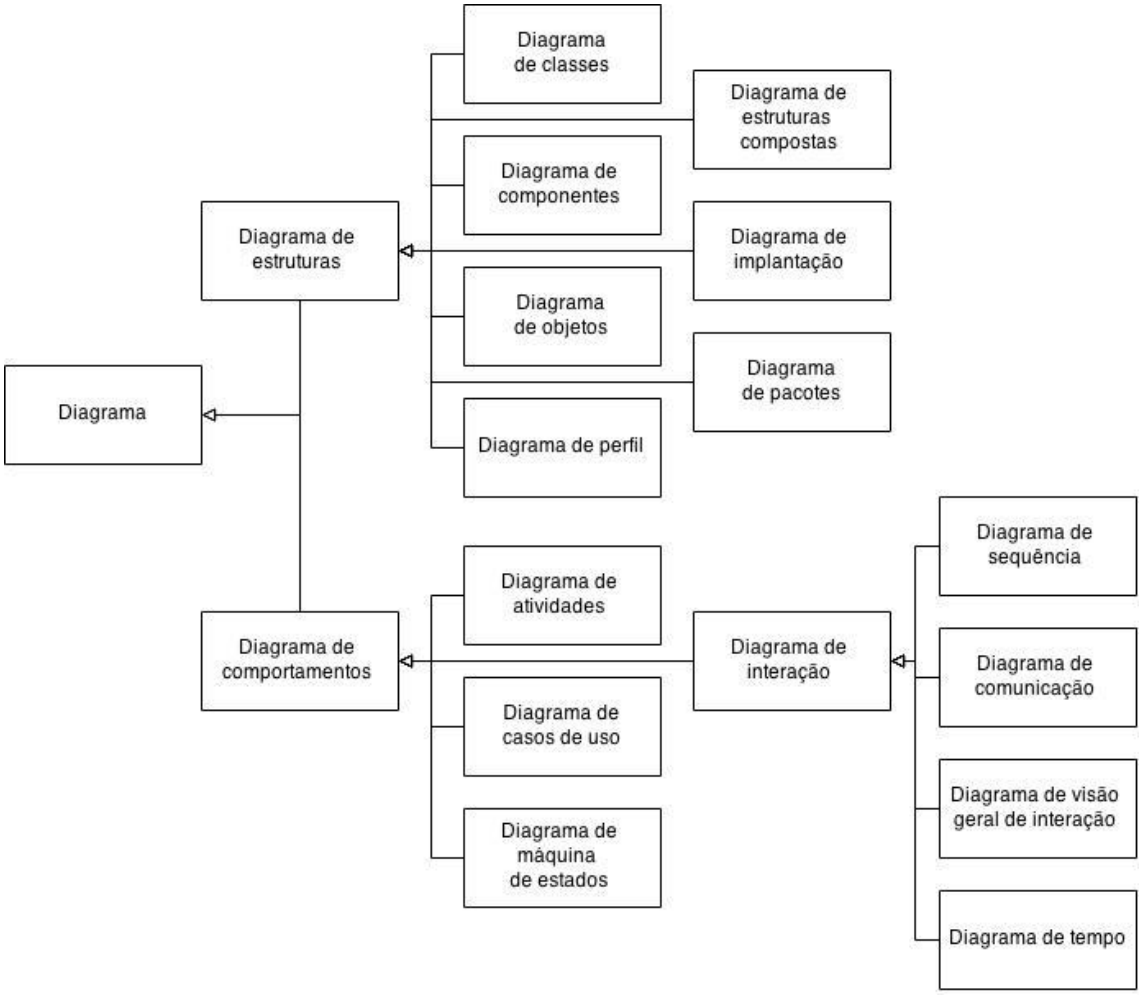

Figura 1 - Classificação estrutural dos diagramas da UML

Dentre todos os diagramas existentes na UML, optou-se por trabalhar com o Diagrama de Casos de Uso, pois, este diagrama é comumente derivado do documento de Especificação

de Casos de Uso, ou serve como base para sua criação. O Diagrama de Casos de Uso é normalmente utilizado nas fases de levantamento, documentação e análise de requisitos.

O Diagrama de Casos de Uso documenta o que o sistema faz do ponto de vista do usuário, ou seja, ele especifica as funcionalidades do sistema e quais atores podem interagir com cada uma delas. Neste diagrama não é levado em consideração o que o sistema precisa fazer para satisfazer uma funcionalidade, apenas é descrito que é possível fazê-la e quem a fará.

Um Diagrama de Casos de Uso é composto por três elementos (i) ator; (ii) caso de uso; e (iii) relacionamento.

Ator representa um usuário que irá interagir com o sistema. Este usuário poderá ser humano ou não, ou seja, o ator pode representar um outro sistema computacional. Este elemento é sempre representado graficamente através de um boneco.

Caso de Uso representa uma funcionalidade do sistema e sua forma gráfica é uma elipse.

Por fim, **Relacionamentos**, representados graficamente por setas, trata dos relacionamentos existentes entre os elementos que compõem o diagrama. A Figura 2 exibe o Diagrama de Casos de Uso de um exemplo de aplicação, o BurgerDigital, um sistema de gerenciamento de uma lanchonete.

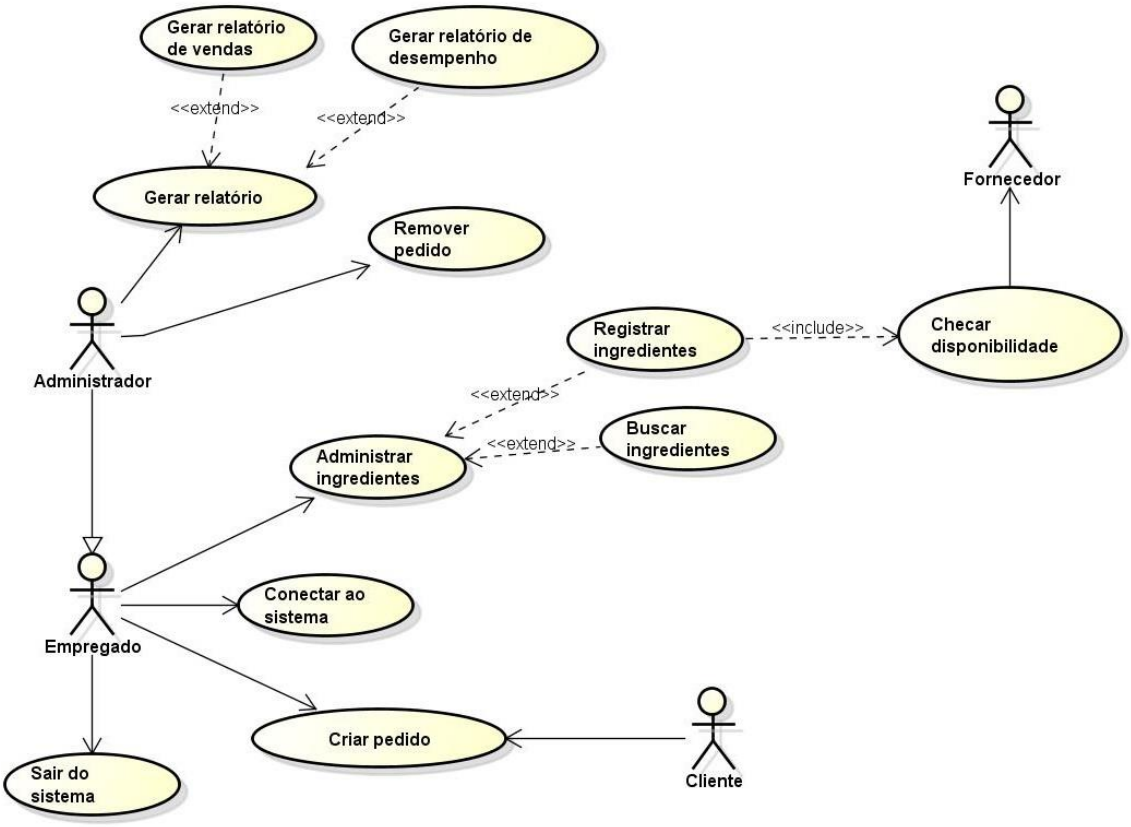

Figura 2 - Diagrama de Casos de Uso do exemplo de aplicação BurguerDigital

Na Figura 2 é possível observar o ator Administrador, o qual possui um tipo de relacionamento de generalização com o ator Empregado. Este tipo de relacionamento representa o conceito de herança, e pode ocorrer entre atores ou entre casos de uso. Graficamente, é representado por uma seta fechada que parte da classe derivada e apontando para a classe base. Pode-se observar ainda na figura em questão, que o ator Empregado está ligado aos casos de uso Criar pedido, Conectar ao sistema e Administrar ingredientes, por meio de relacionamentos de associação, graficamente representado por uma seta (ou uma linha) que liga o ator e o caso de uso. O caso de uso Administrar ingredientes é vinculado a outros dois casos de uso: Registrar ingredientes e Buscar ingredientes. Essa ligação está ocorrendo por meio do relacionamento de extensão (*extend*). Este tipo de relacionamento indica uma funcionalidade do sistema que pode opcionalmente ser incluído no fluxo do caso de uso apontado pela seta, graficamente, é exibido por uma linha tracejada com uma ponta de seta aberta direcionada ao caso de uso que irá incluir o fluxo. Esse relacionamento possui um rótulo *«extend»* sobre a linha tracejada.

Ainda na Figura 2, é possível observar o caso de uso Registrar ingredientes ligado por meio de um relacionamento de inclusão (include) ao caso de uso Checar disponibilidade. Este tipo de relacionamento indica uma adição obrigatória ao comportamento do caso de uso base.

A representação gráfica é uma seta tracejada com uma seta aberta que parte do caso de uso base e aponta para o caso de uso a ser incluído. Este relacionamento possui o rótulo «include». O caso de uso Checar disponibilidade está sendo associado ao ator Fornecedor, com a seta apontada para o ator, indicando que este é um ator externo ao sistema, ou seja, esse não é um usuário humano, mas um sistema (ou módulo) externo.

2.5 Tipos de visualização

A visualização dos *links* de rastreabilidade é essencial para o entendimento dos requisitos, seus relacionamentos e conhecimento geral do software em questão (MERTEN; JUPPNER; DELATER, 2011).

A forma mais simples e comum de exibição dos *links* de rastreabilidade é por meio de matriz de rastreabilidade, navegação em árvore e visualização com *hiperlink* (MARCUS; MALETIC; SERGEYEV, 2005). O modelo de visualização em matriz é útil quando se deseja uma visão do conjunto das relações de rastreabilidade, esta é forma mais tradicional de exibir os resultados obtidos do rastreamento, é uma estrutura de fácil compreensão, porém torna-se complexa em projetos de grandes dimensões (WINKLER; PILGRIM, 2010). A visualização por meio de navegação em árvore é uma boa opção para os casos em que precisasse navegar pelos relacionamentos, enquanto a visão com *hiperlink* são utilizados quando os objetos e as relações precisam ser apresentados sobre um mesmo ponto de vista. Heim, Ziegler e Lohmann (2008) apresentam ainda um estudo sugerindo a utilização de grafos para a visualização da rastreabilidade, segundo os autores com essa abordagem é possível realizar uma rápida análise de dependência entre os requisitos do sistema. O estudo aponta ainda que a maioria das ferramentas não fornece visualização de rastreamento em grafos.

2.6 Considerações finais

Neste capítulo foi feita uma introdução sobre rastreabilidade, qual a importância de seu uso, quais vantagens de aplicar a rastreabilidade no processo de desenvolvimento de software e foi descrito o porquê é preciso definir um modelo de referência de rastreabilidade. Foi apresentado ainda uma relação de abordagens de rastreabilidade existente e foram apresentados os artefatos que serão utilizados nessa pesquisa. Por fim foram abordados os

tipos de visualização da rastreabilidade apontando os cenários para qual cada uma delas é mais indicada.

3 Trabalhos relacionados

Várias abordagens (semi) automáticas de rastreabilidade têm sido propostas na literatura (BUCHGEHER; WEINREICH, 2011), (LEAL; FIGUEIREDO; SOUZA, 2008), (DE LUCIA; PENTA; OLIVETO, 2011), (FILHO; LENCASTRE; RODRIGUES, 2013), (LIU *el al*, 2007), (CERRI, 2007). Cada abordagem tem vantagens e desvantagens e podem ser aplicadas para um determinado propósito. A seguir são apresentadas algumas abordagens de rastreabilidade existentes na literatura.

3.1 Relacionamentos entre modelos de componentes, documentos de arquitetura de requisitos e código-fonte

Buchgeher e Weinreich (2011) demonstram a abordagem semiautomática LISA, capaz de capturar *links* de rastreabilidade a partir de modelos de componentes de arquitetura até as decisões de design, arquitetura de requisitos e código-fonte. LISA está baseado em um modelo de descrição arquitetural semiformal. A abordagem observa como os desenvolvedores estão trabalhando na arquitetura e na implementação do projeto, dessa forma consegue apoiar a criação automática de *links* de rastreabilidade. O processo da abordagem é dividido em três etapas: Na primeira etapa o desenvolvedor seleciona as decisões de projeto, estas serão o contexto a ser utilizado na abordagem. Na segunda etapa, o desenvolvedor realiza o projeto de arquitetura e a implementação das tarefas. Ao longo desse processo, eventos de modificação são criados e registrados. Esses eventos contêm a descrição da modificação, objetivos e os elementos modificados. A solicitação da alteração é enviada para um *event-logger* que se responsabiliza por capturar os alvos e gerenciar as decisões necessárias. A última etapa trata-se de uma revisão dos *links* de rastreabilidade gerados pela ferramenta.

3.2 Relacionamentos entre código-fonte e UML

Leal, Figueiredo e Souza (2008) apresenta uma abordagem semiautomática para relacionar e manter a rastreabilidade entre elementos de artefatos com sintaxe rigorosa, como códigos fontes e UML; e artefatos baseados em texto livre, como é o caso da Especificação de Casos de Uso. Os autores propõem um uso de um *templates* para escrever a especificação de casos de uso, para que esse tipo de artefato seja padronizado. Para extrair informações dos

artefatos textuais são aplicadas técnicas de Processamento de Linguagem Natural (PLN). O processo para a geração das relações de rastreabilidade segue três fases: aquisição (coleta de dados dos documentos a serem rastreados e padronização dos mesmos), análise (os dados dos extraídos são analisados e armazenados) e fase da disponibilização (os resultados são exibidos ao usuário).

3.3 Relacionamentos entre documento de requisitos e código-fonte

De Lucia, Penta e Oliveto (2011) apresentam em sua pesquisa uma abordagem para melhorar o léxico do código-fonte, realizando comparações entre as anotações e comentários com a especificação de requisitos. Este estudo analisa a similaridade existente entre requisitos e código-fonte. A pesquisa resultou em uma ferramenta para desenvolvedores que sugere termos do domínio para que possam ser utilizados no código fonte. A avaliação do estudo ocorre de forma controlada, analisando os resultados gerados por estudantes de mestrado e graduação. A ideia geral da proposta é que o termo sugerido aos desenvolvedores durante a fase de construção do software minimize o impacto de termos não significantes.

3.4 Relacionamentos entre *i*, Prometheus* e *JACK*

Filho, Lencastre e Rodrigues (2013) propõem uma abordagem orientada a regras para automatizar o relacionamento de rastreabilidade entre modelos de software heterogêneos criados durante o processo de desenvolvimento de sistemas multiagente. A abordagem proposta ataca a identificação de elementos heterogêneos e inconsistências entres os documentos, visando auxiliar as fases de validação e verificação. Filho, Lencastre e Rodrigues criaram um grupo de regras escritas em XQuery para identificar a relação de rastreabilidade entre os modelos desenvolvidos em *i**, *Prometheus* e o *Jack*. A abordagem utiliza o WordNet (WORDNET, 2015), uma base de dados lexical, para auxiliar no processo de comparação de termos existentes nos documentos (substantivos, verbos adjetivos e advérbios). O estudo, que é uma extensão de Filho (2011), propõe uma nova arquitetura para a abordagem, propondo a inclusão de uma ferramenta visual para a edição de regras e outra ferramenta para realizar a visualização dos resultados. É esperado que, diferentemente da abordagem inicial proposta por Filho, essa abordagem permita: Geradas regras para modelos distintos; navegação pelas relações de rastreabilidade geradas; o usuário possa gerenciar as propriedades das relações de

rastreabilidade; integração com ferramentas de desenvolvimento e teste de software; registro da navegação entre as relações de rastreabilidade; filtro dos resultados por relações de rastreabilidade e flexibilidade na visualização dos dados obtidos.

3.5 Relacionamentos entre código-fonte e documento de requisitos

Liu *et al*. (2007) apresentam um estudo com a proposta de identificar partes do código-fonte que estão relacionadas a funcionalidades do software. Liu *et al*. propõem uma abordagem híbrida que unifica relatos sobre a execução das funcionalidades a partir dos cenários de rastreamento e os comentários existentes no código-fonte. A indexação de semântica latente é utilizada para criar índices para as funcionalidades que são executadas a partir de um determinado cenário. A lista das funcionalidades é avaliada e ordenada por semelhança, tendo como base as funcionalidades e a descrição do requisito. O usuário interfere manualmente informando se o relacionamento entre a funcionalidade e o requisito está correto ou não. O estudo utilizou dois exemplos de aplicação como base para a avaliação de desempenho e usabilidade. Os resultados demonstram que a abordagem híbrida é mais eficaz para exibir funcionalidades relevantes na posição superior do que quando usado somente método de indexação semântica latente.

3.6 Especificação de Requisitos e Modelos de Casos de Uso

Cerri (2007) apresenta um estudo que capaz de relacionar os documentos de Especificação de Requisitos e Modelos de Casos de Uso. A pesquisa apresenta um modelo de rastreabilidade onde são descritos os relacionamentos existentes entre os documentos Especificação de Requisitos e Modelos de Casos de Uso. A abordagem proposta avalia o impacto decorrente da substituição, exclusão, inclusão de novos elementos a um dos artefatos.

3.7 Considerações finais

Por meio do levantamento da literatura, não foram encontrados trabalhos que abordassem o domínio Orientado a Objetos e que relacionassem os documentos de Especificação de Casos de Uso e Diagramas de Casos de Uso sobre a mesma perspectiva adotada desta dissertação.

Tabela 2 - Comparação entre os trabalhos relacionados

	Tipo de abordagem	**Categoria de abordagem**	**Artefatos relacionados**
Buchgeher e Weinreich	Semiautomática	Orientada a processos	Modelos de componentes, documentos de arquitetura de requisitos e código-fonte
Leal, Figueiredo e Souza	Semiautomática	Abordagem formal	Código-fonte e UML
De Lucia, Penta e Oliveto	Automática	Recuperação da informação	Documento de requisitos e código-fonte
Filho, Lencastre e Rodrigues	Automática	Orientada a regras	*i**, *Prometheus* e o *JACK*
Liu *et al*.	Semiautomática	Tempo de execução do software	Código-fonte e documento de requisitos
Cerri	Semiautomática	Abordagem formal	Especificação de Requisitos e Diagramas de Casos de Uso

Como pode ser visto na Tabela 2, nenhuma abordagem tratou relacionamentos entre Especificação de Casos de Uso e Diagramas de Casos de Uso. O estudo de Filho, Lencastre e Rodrigues é a que mais semelhante à esta pesquisa, poderem possui domínio diferente. Cerri possui um domínio semelhante ao que é proposto nesta pesquisa, porém, seu estudo foca na análise de impacto decorrente de modificações em elementos dos documentos de Especificação de Requisitos e Modelos de Casos de Uso.

4 Modelo de referência de rastreabilidade

Neste capítulo é apresentado o modelo de referência proposto nesta pesquisa para gerar relações de rastreabilidade entre elementos da Especificação de Casos de Uso com elementos do Diagrama de Casos de Uso UML. Ao longo do capítulo, exemplos são apresentados para ilustrar o modelo de referência de rastreabilidade.

4.1 Visão geral do modelo de referência

Um modelo de referência de rastreabilidade consiste de um grupo de elementos organizados de forma que possa descrever analiticamente os artefatos criados durante o processo de desenvolvimento de software e as relações existentes entre eles, podendo ser expresso por meio de linguagem textual ou gráfica (MELLOR; CLARK; FUTAGAMI, 2003).

A importância da definição e do uso dos modelos de referência foi abordada no Capítulo 2, onde é descrita a necessidade de identificar os diversos tipos de relacionamento de rastreabilidade e suas semânticas.

O Paradigma de Orientado a Objetos (POO) é um tópico comum na grade curricular de cursos de ciência da computação e afins (ANQUAN *et al*., 2010). Isto ocorre devido à importância para os profissionais de tecnologia de informação em obter esse conhecimento durante sua formação, que posteriormente é requerido para a maioria dos profissionais de tecnologia da informação.

Quando POO é utilizada como abordagem de desenvolvimento, é comum que sejam criados alguns artefatos no início do projeto (DUTOIT; PAECH, 2002), dentre eles os Diagramas de Casos de Uso UML (OMG, 2015) e o documento de Especificação de Casos de Uso (JACOBSON; 1992). UML é uma linguagem visual, o que facilita na documentação, comunicação e compreensão geral do sistema. O uso da UML minimiza a necessidade da leitura do código-fonte do software para compreender o objetivo do mesmo. Outro fator importante dos diagramas UML é que são independentes de linguagem de programação e métodos de desenvolvimento, possuindo conceitos concretos que facilitam o entendimento (TACLA, 2013).

O Documento Especificação de Casos de Uso auxilia as partes interessadas (*stakeholders*) a entender os requisitos do sistema e podem ser usados como base para as atividades de validação do software.

A proposta do modelo de referência foca nos artefatos de Especificação de Casos de Uso e no Diagrama de Casos de Uso UML. Dentre os diagramas UML, o Diagrama de Casos de Uso foi escolhido como base para o exemplo de aplicação desta pesquisa.

Foi realizada uma pesquisa dos tipos de relações de rastreabilidade comumente utilizados na literatura. Os tipos de relações encontrados na pesquisa foram analisados a fim de conhecer a relação semântica de cada uma. Em seguida, todos os elementos dos documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso foram avaliados e identificadas as relações, com base na similaridade semântica, existentes entre eles. Por fim, cada uma das relações encontradas na avaliação foi nomeada com base nos tipos de relações de rastreabilidade utilizados na literatura.

Assim, este modelo de rastreabilidade identificou quatro relações de rastreabilidade entre os artefatos analisados. O modelo proposto nesta abordagem é baseado no estudo apresentado proposto por Cysneiros e Zisman (2008), que propuseram um modelo para relacionar artefatos criados durante o desenvolvimento de sistemas multiagentes, divergindo assim do modelo aqui proposto, que atende a artefatos construídos sobre domínio Orientado a Objetos.

Os tipos de relações de rastreabilidade propostas nesta pesquisa são sobreposição, composição, utilização e dependência.

4.2 Relações de rastreabilidade

Este modelo de referência considera os principais elementos dos artefatos analisados nesta pesquisa.

A seguir, serão descritos cada um dos tipos de relação de rastreabilidade existentes neste modelo e alguns exemplos são expostos a fim de ilustrar esses relacionamentos.

A Figura 3 apresenta todos os tipos de relações identificados entre os elementos considerados relevantes que existem na Especificação de Casos de Uso e no Diagrama de Casos de Uso. A seguir, serão definidos os diversos tipos de relacionamento de rastreabilidade e serão expostos exemplos a partir da perspectiva de alguns artefatos associados.

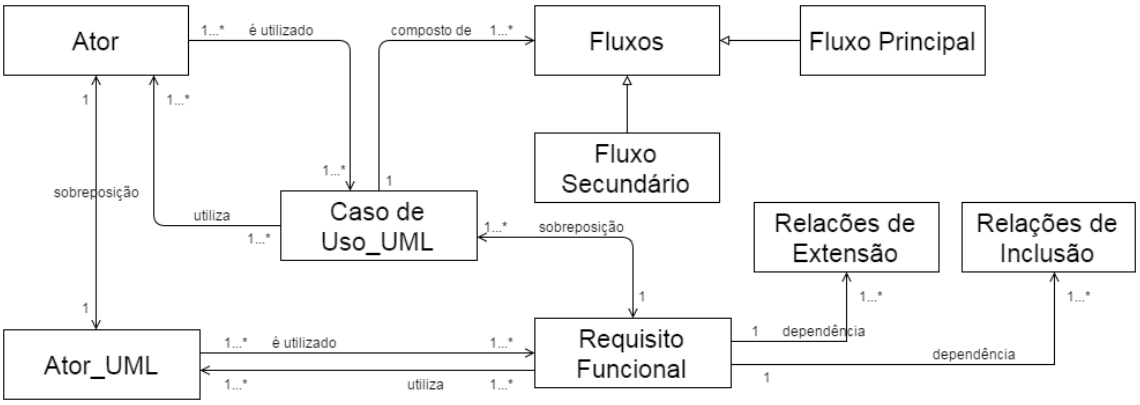

Figura 3 - Relacionamento entre os elementos do Documento de Especificação de Requisito e Diagrama UML de Casos de Uso

- Sobreposição - neste tipo de relação, um elemento E1 se sobrepõe a um elemento E2 (e um elemento E2 se sobrepõe a um elemento E1). Quando os elementos E1 e E2 referenciam a elementos de aspectos comuns, tem-se uma relação de sobreposição. Pode-se observar na Figura 3 que existem relação de rastreabilidade entre a) um ator da Especificação de Casos de Uso com um ator do Diagrama de Casos de Uso e b) um requisito funcional da Especificação de Casos de Uso com um caso de uso do Diagrama de Casos de Uso;

Para ilustrar, considere uma situação em que um ator A1 da Especificação de Casos de Uso tem uma relação de rastreabilidade do tipo sobreposição com um ator A2 do Diagrama de Casos de Uso, se o nome do ator A1 for sinônimo com o nome do ator A2, o número de requisitos vinculados ao ator A1 for semelhante ao número de casos de uso vinculados ao ator A2 considerando um limiar (por exemplo 60%). Por exemplo, o ator Administrador do Diagrama de Casos de Uso tem nome sinônimo com o ator Gerente da Especificação de Casos de Uso, neste caso o ator Administrador possui relacionamentos com dez casos de uso (Figura 2), quatro devido ao relacionamento direto e outros seis devido aos casos de uso herdado do ator Empregado, ao qual o ator Administrador possui uma relação de generalização no Diagrama de Casos de Uso, assim ele possui uma relação de sobreposição com o ator Gerente do Documento de Especificação de Casos de Uso caso ele possua pelo menos seis requisitos.

- Composição (composto de) - neste tipo de relação, um elemento E1 é composto por um elemento E2, ou seja, E1 é um elemento complexo que possui em sua formação o elemento E2. Na Figura 3 observa-se a relação de composição entre

caso de uso do Diagrama de Casos de Uso e fluxos da Especificação de Casos de Uso;

Para ilustrar, considera a situação em quem um caso de uso UC1 do Diagrama de Casos de Uso tem uma relação de composição com um fluxo F1 do Documento de Especificação de Casos de Uso quando o UC1 está relacionado com um Requisito R1 do Documento de Especificação de Casos de Uso. Por exemplo, o caso de uso Administrar ingredientes do Diagrama de Casos de Uso tem relacionamento (por sobreposição) com o requisito Gerenciar ingredientes do documento de Especificação de Casos de Uso, portando, Administrar ingredientes possui uma relação do tipo composição com os fluxos existentes no requisito Gerenciar ingredientes.

- Utilização (utiliza/é utilizado) - neste tipo de relação, um elemento E1 utiliza o um elemento E2. Este tipo de relacionamento acontece quando o elemento E1 necessita da existência do elemento E2 para que tenha seu objetivo concretizado. Pela Figura 3 observa-se esse tipo de relação entre a) um ator do Documento de Especificação de Casos de Uso com um caso de uso do Diagrama de Casos de Uso e b) um requisito funcional do Documento de Especificação de Casos de Uso com um ator do Diagrama de Casos de Uso.

Para ilustrar, considere uma situação em que um ator A1 do Diagrama de Casos de Uso possui uma relação de utilização com um requisito funcional RF1 do Documento de Especificação de Casos de Uso, isso ocorre quando o ator A1 possui uma ligação de associação com um caso de uso UC1 do Diagrama de Casos de Uso e UC1 possui uma relação do tipo sobreposição com o requisito funcional RF1. Por exemplo, o ator Funcionário do Diagrama de Casos de Uso possui uma relação com o requisito funcional Conectar ao sistema do Documento de Especificação de Casos de Uso, pois este requisito funcional possui uma relação de sobreposição com o caso de uso Conectar ao sistema do Diagrama de Casos de Uso.

- Dependência – neste tipo de relação, um elemento E1 depende de um elemento, se E1 depende da existência do elemento E2, ou seja, as mudanças do elemento E2 impactam no elemento E1. Pela Figura 3 observa-se um relacionamento desse tipo entre a) Requisito funcional da Especificação de Casos de Uso com as relações de

extensão do Diagrama de Casos de Uso e b) Requisito funcional da Especificação de Casos de Uso com as relações de inclusão do Diagrama de Casos de Uso.

Para ilustrar, considere uma situação em que um requisito funcional RF1 da Especificação de Casos de Uso possui algum tipo de com um caso de uso UC1 do Diagrama de Caso de Uso e UC1 possui um relacionamento de extensão na UML com o caso de uso UC2. Neste caso, por dependência, o RF1 também está relacionado com o caso de uso UC2.

Vale ressaltar que o modelo de referência apresentado não deve ser considerado um consenso, assim podem surgir outras pesquisas que identifiquem outras relações de rastreabilidade entre os elementos dos modelos analisados, ou ainda identificar novos elementos, existentes nesses artefatos, que não foram contemplados nesta pesquisa.

4.3 Considerações finais

Este capítulo descreveu o modelo de referência para a rastreabilidade entre os artefatos de Especificação de Casos de Uso e Diagrama de Casos de Uso UML. Foi apresentado ainda os tipos de relações de rastreabilidade existentes entre esses dois artefatos dando alguns exemplos práticos desses tipos de relações.

Tabela 3 - Comparação entre os trabalhos relacionados e esta dissertação

	Tipo de abordagem	**Categoria de abordagem**	**Artefatos relacionados**
Buchgeher e Weinreich	Semiautomática	Orientada a processos	Modelos de componentes, documentos de arquitetura de requisitos e código-fonte
Leal, Figueiredo e Souza	Semiautomática	Abordagem formal	Código-fonte e UML
De Lucia, Penta e Oliveto	Automática	Recuperação da informação	Documento de requisitos e código-fonte
Filho, Lencastre e Rodrigues	Automática	Orientada a regras	*i**, *Prometheus* e o *JACK*
Liu *et al*.	Semiautomática	Tempo de execução do software	Código-fonte e documento de requisitos
Cerri	Semiautomática	Abordagem	Especificação de Requisitos e

		formal	Diagramas de Casos de Uso
Oliveira	Automática	Orientada a regras	Especificação de Casos de Uso e Diagrama de Casos de Uso

A tabela 3 apresenta os trabalhos relacionados a esta pesquisa que foram encontrados durante a pesquisa bibliográfica. Por último foi adicionado a tabela as informações existentes nesta dissertação. Podemos notar desta maneira que esta abordagem é a única, dentre os estudos encontrados, que se propõe a utilizar uma abordagem orientada a regras para relacionar elementos da Especificação de Casos de Uso e Diagrama de Casos de Uso.

5 Abordagem de rastreabilidade

Neste capítulo será apresentada a abordagem de rastreabilidade proposta para apoiar a geração automatizada de relações de rastreabilidade entre os modelos de Documento de Especificação de Casos de Uso e Diagrama de Casos de Uso UML. Esta abordagem também auxiliará na identificação de inconsistências entre os esses dois modelos.

5.1 Visão geral do *framework*

A Figura 4 apresenta a arquitetura da abordagem proposta por Filho (2011) e que serve como base o *framework* proposto. Filho (2011) sugere que os artefatos, criados no processo de desenvolvimento de sistemas multiagentes, sejam escritos em formatos nativos por meio de ferramentas proprietárias ou ferramentas de edição de diagramas. Devido à limitação de algumas ferramentas em exportar para XML, foi proposto na arquitetura uma fase de transformação desses modelos para o padrão XML (*Modelos Baseado_XML*).

Na arquitetura proposta por Filho (2011), os modelos convertidos em XML e as regras escritas em XQuery servem como entrada para que o motor de rastreabilidade (*Motor Rastreabilidade_Analise_Completude*), com base nas regras criadas para suportar a rastreabilidade dos artefatos em análise, possa identificar as relações de rastreabilidade e as inconsistências existentes entre os artefatos. O motor de rastreabilidade utiliza o WordNet (WORDNET, 2015) para apoiar a identificação de sinônimos entre os elementos dos modelos. O uso do WordNet é importante, pois é comum que os modelos analisados sejam construídos por pessoas ou momentos distintos, e isso pode fazer com que certas nomenclaturas utilizadas sejam divergentes, mas queiram representar um mesmo objeto.

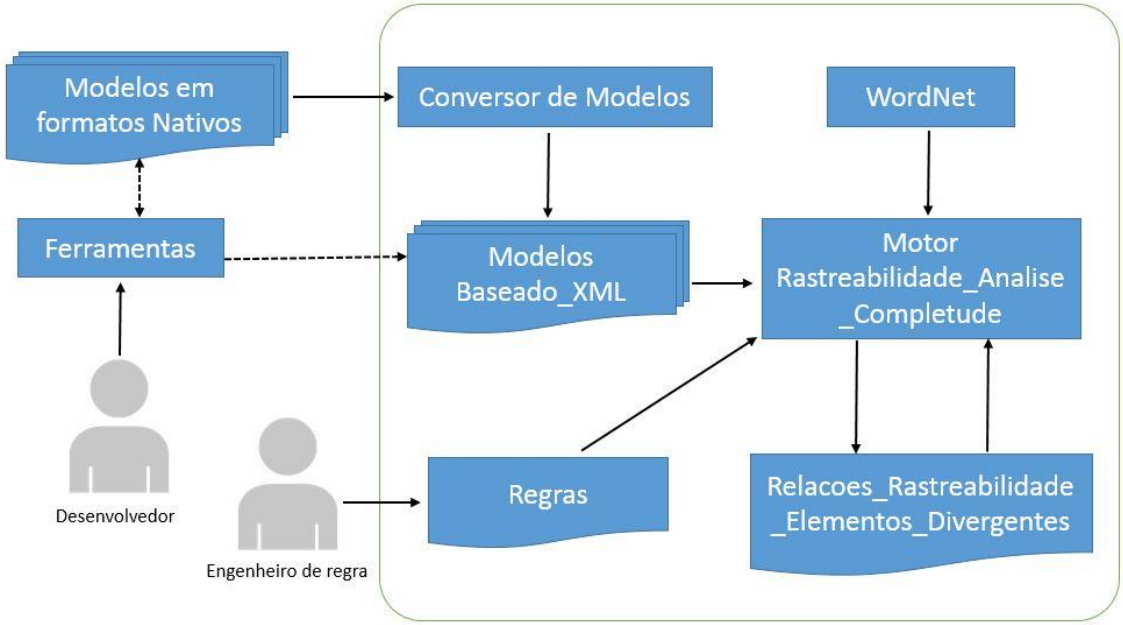

Figura 4 - Visão geral da arquitetura da abordagem proposta por Filho (2011)

A arquitetura a ser utilizada nesta pesquisa é uma simplificação da arquitetura apresentada no trabalho de Filho (2011). A abordagem apresentada nesta pesquisa restringe aos artefatos criados para o domínio Orientado a Objetos e ao relacionamento entre modelos criados a partir de ferramentas que permitam exportar diretamente para o padrão XML, como o Astah (2015), para o Diagrama de Casos de Uso, e para a Especificação de Casos de Uso, foi desenvolvido, como parte dessa pesquisa, um sistema online que possibilita a criação de novos documentos desse tipo, esse sistema recebeu o nome de iTraceWeb.

O iTraceWeb (ITRACEWEB, 2015) implementada o *template* descrito no Capítulo 2. No iTraceWeb é possível criar projetos e preencher um formulário para registrar um novo requisito, gerando assim uma lista com todos os requisitos do software, devidamente documentados em formato natural, estes podem ser posteriormente convertidos e exportados para o formato XML.

A Figura 5 exibe como ficou a arquitetura desta abordagem. Como é possível observar, a única diferença entre as arquiteturas da Figura 4 e Figura 5 é que a Figura 5 abstrai a necessidade de um conversor do modelo em formato nativo para XML. Isso ocorreu devido ao fato de que as ferramentas utilizadas para a criação dos artefatos utilizados nessa abordagem (Astah e iTraceWeb), permitirem exportar nativamente os artefatos para o padrão XML.

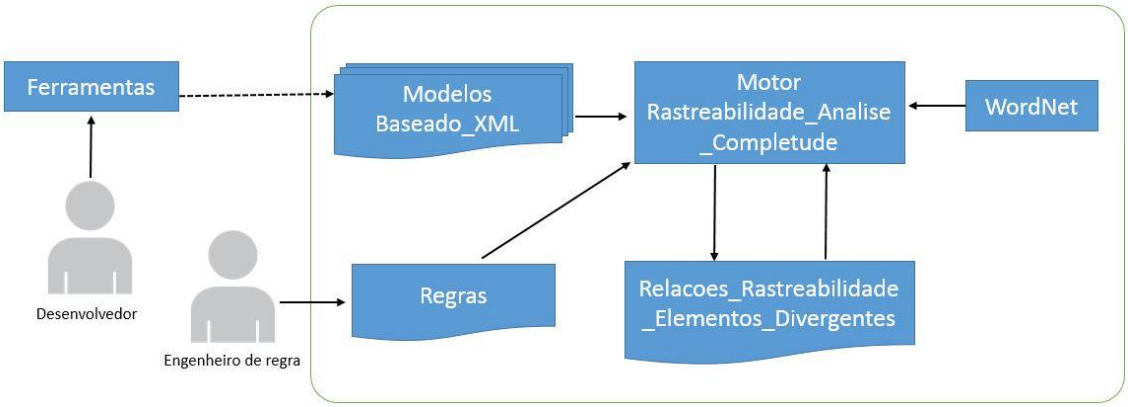

Figura 5 - Visão geral da arquitetura utilizada

Assim, a arquitetura utilizada nesta pesquisa, herda os benefícios do funcionamento da arquitetura já validada anteriormente durante o estudo de Filho (2011), porém, esta possui internamente alterações nos componentes Regras e no *Motor Rastreabilidade_Analise_Completude*. As Regras utilizadas nesta arquitetura são exclusivas para o domínio Orientado a Objetos, com foco nas relações de rastreabilidade existentes entre os artefatos de Especificação de Casos de Uso e o Diagrama de Casos de Uso UML, diferenciando da arquitetura de Filho (2011), que possuem regras para os artefatos do domínio Orientados a Agentes. O *Motor Rastreabilidade_Analise_Completude* apresentado em Filho (2011) utiliza um grupo de classes que estende o XQuery para executar ações específicas, muitas delas para tratar apenas ações sobre os artefatos do domínio alvo de sua pesquisa. Classes semelhantes foram criadas para a arquitetura utilizada nesta abordagem, só que estas atendem as necessidades dos artefatos criados para o domínio Orientado a Objetos.

A seguir é descrito o funcionamento da arquitetura utilizada nesta abordagem.

5.1.1 Fluxo de funcionamento do *framework*

Para inicializar o processo automatizado de identificação de relações de rastreabilidade entre os artefatos de Especificações de Requisitos e Diagrama de Caos de Uso UML, espera-se que o usuário já tenha criado esses artefatos anteriormente de forma nativa, utilizando as ferramentas iTraceWeb e Astah, respectivamente e em seguida exportado os resultados para o padrão XML.

De posso dos artefatos convertidos para XML, esses servirão como entrada para o motor de rastreabilidade (*Motor Rastreabilidade_Analise_Completude*). É necessário também, que um engenheiro de regra crie as regras em linguagem XQuery, de forma que essas sejam capazes de relacionar os elementos dos artefatos de entrada (modelos em XML) e

identificar inconsistências entre esses artefatos. As regras criadas também servirão como entrada para o motor de rastreabilidade.

O motor de rastreabilidade é responsável por executar as regras e, quando preciso, utilizar o WordNet para apoiar a identificação de sinônimos.

Para ilustrar o funcionamento e importância do WordNet considere *isSynonym* como uma função estendida do XQuery (mais detalhes na seção 5.3). A função *isSynonym* recebe como parâmetro dois termos que serão analisados para analisar se estes possuem características que o definam como sinônimos. Cada um desses termos é consultado na base do WordNet a fim de identificar uma lista de sinônimos para cada um dos termos. De posse das duas listas de sinônimos, uma de cada termo, é verificado se algum item da lista do primeiro termo é igual a pelo menos um item da lista do segundo termo. Se a função *isSynonym* encontrar algum item igual, os termos passados como parâmetro são considerados sinônimos. Para exemplificar considere uma situação em que a função *isSynonym* recebe como parâmetro Cliente e Freguês. O WordNet retornará uma lista de sinônimos para Cliente (Cliente, Freguês) e uma lista de sinônimos para Freguês (Freguês, Cliente). Desta forma, é possível observar que Cliente aparece na lista de sinônimos de Freguês e que Freguês aparece na lista de sinônimos de Cliente. Assim a função *isSynonym* considera Cliente e Freguês como termos sinônimos.

O resultado das relações de rastreabilidade e inconsistências são registrados em um novo documento XML (*Relacoes_Rastreabilidade_Elementos_Divergentes*). Isso ocorre para preservar os documentos originais, permitindo que durante o processo de identificação das relações, as regras possam utilizar o resultado preliminar da identificação das relações de rastreabilidade.

Assim, conforme é possível observar na Figura 5, o XML de *Relacoes_Rastreabilidade_Elementos_Divergentes* passa a servir como entrada do motor de rastreabilidade. Neste momento, o motor de rastreabilidade passa a trabalhar com quatro artefatos: as regras em XQuery, o Documento de Especificação de Casos de Uso e o Diagrama de Casos de Uso (padronizados em *Modelos Baseado_XML*) e o XML dos resultados preliminares (*Relacoes_Rastreabilidade_Elementos_Divergentes*).

5.1.2 Exemplo de uso do *framework*

Um exemplo do uso desta abordagem pode ser visto na Figura 6.

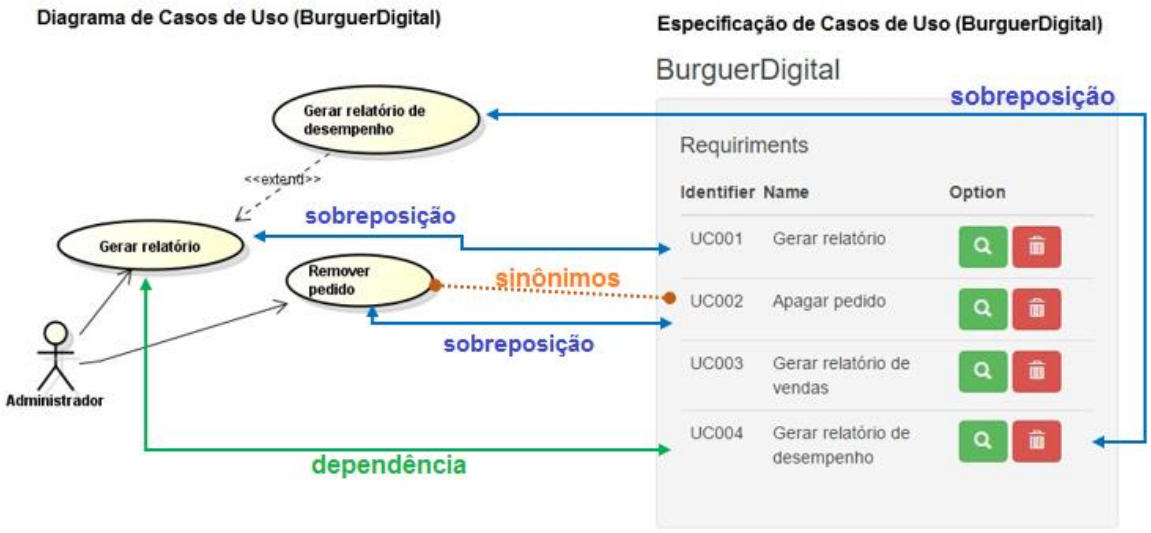

Figura 6 - Exemplo do uso de regras desta abordagem

No exemplo exibido na Figura 6, uma das regras existente no documento de Regras, identificou entre os elementos da Especificação de Casos de Uso *Gerar relatório* e *Gerar relatório de desempenho*, respectivamente, uma relação do tipo sobreposição, com os elementos do Diagrama de Casos de Uso *Gerar relatório* e *Gerar relatório de desempenho*. Após a execução da regra que identificou esses relacionamentos, o resultado preliminar foi armazenado em *Relacoes_Rastreabilidade_Elementos_Divergentes*.

Em sequência uma nova regra foi executada e verificou que, se *Gerar relatório de desempenho*, da Especificação de requisito, tem relacionamento com *Gerar relatório de desempenho*, do Diagrama de Caso de Uso e os elementos do Diagrama de Caso de Uso *Gerar relatório de desempenho* e *Gerar relatório* possuem um relacionamento (extensão), então *Gerar relatório de desempenho*, da Especificação de Casos de Uso, também possui relacionamento, do tipo dependência, com *Gerar relatório*, do Diagrama de Caso de Uso.

Pode-se observar ainda que *Remover pedido*, do Diagrama de Casos de Uso, e *Apagar pedido*, da Especificação de Casos de Uso, possuem um relacionamento, do tipo sobreposição, pois foram identificados como sinônimos pelo WordNet.

Vale notar que, como descrito na sessão 5.3, as relações do tipo de sobreposição, não levam em consideração apenas os nomes e sinônimos dos elementos, mas também faz comparação ao número de relações que cada um dos elementos analisados possui, só após esses passos que é possível considerar como uma relação de sobreposição.

Nos casos em que as regras identificam inconsistências entre os modelos, ou seja, elementos divergem semanticamente ou inexistes entre os modelos, a ferramenta não obriga o usuário a modificar os modelos. O usuário poderá decidir usar ou não os resultados obtidos da execução das regras para corrigir os artefatos.

5.2 Regras de rastreabilidade e verificação de integridade

O padrão definido para a criação das regras pode ser visto na Figura 7. A primeira parte referência as propriedades da regra e a segunda parte contém toda o código XQuery que será responsável por identificar as relações de rastreabilidade e elementos divergentes. A seguir são detalhadas cada uma dessas partes.

Figura 7 - Exemplo do uso de regras desta abordagem

Parte 1: Este bloco expõe as informações da regra. É neste local que ficam definidos o id, prioridade, tipo, elementos analisados e a descrição.

O id representa um identificar único para a regra.

A **prioridade** (*priority*) informará qual o grupo da ordem de execução que a regra pertence e se esta é ou não uma regra dependente (prioridade>1). As prioridades dependentes deverão ser pelo menos um número maior do que a regra da qual dependem. Assim as regras são agrupadas em prioridades e executadas em ordem crescente. Desta forma as regras de prioridade 1 são executadas em primeiro momento, depois as regras de prioridade 2, podendo estas utilizar os resultados do primeiro grupo de execução (prioridade=1) em seguida são

executadas as regras de prioridade 3, podendo utilizar os resultados dos dois grupos anteriores (prioridade=1 e prioridade=2) e assim sucessivamente.

O **tipo** (*type*) identifica qual é o tipo da regra, baseado nos tipos de relações proposto na Capítulo 4.

Os **elementos origem** e **destino** que estão sendo analisados nas regras são registrados respectivamente como *ElementTypeA* e *ElementTypeB.*

Por fim, a **descrição** (*description*) apresenta um breve resumo da regra.

Parte 2: Este bloco contém toda a regra escrita em XQuery e é possível subdividir na seguinte forma: (i) bloco de declarações e (ii) bloco de iteração.

O (i) bloco de declarações contém definições de namespaces, variáveis e carregamento de documento. Cada um desses elementos é detalhado a seguir:

Texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo, texto de exemplo.

A seguir é apresentado um exemplo de lista de marcadores de apenas um nível (se você finalizar cada item da lista com ponto e vírgula, elas devem ser iniciadas com letra minúsculas; se você finalizar cada item da lista com ponto, elas devem ser iniciadas com letra maiúsculas):

- *Namespaces* – são usados para carregar nomenclaturas externas e funções criadas em Java para estender o XQuery (mais detalhe na seção 5.3). Por exemplo, a seguir a declaração XQuery *java:iTrace.XQuerySynonymsFunctions* faz com que todas as funções escritas na classe *XQuerySynonymsFunctions* estejam disponíveis para uso no código XQuery (como a função *isSynonyms*).
    - *declare namespace syn = "java:iTrace.XQuerySynonymsFunctions"*
- Declaração de variáveis e documentos – esta área conterá todos os elementos que serão utilizados na regra e também os nomes dos modelos origem e destino que serão comparados durante o processamento da regra. A seguir são expostos alguns exemplos de declarações de variáveis. A primeira declaração escrita em XQuery, a variável *$umlDoc* é atribuído ao documento de Especificação de Casos de Uso, através da função estendida em Java *getUMLFileName*. A segunda variável, a *$umlActors*, é atribuída ao valor do resultado da execução da expressão XPath. Em especial a variável *$umlActors* receberá a lista de todos os atores que estão na *tag UML:Actor* localizado em qualquer parte do documento XML existente em *$umlDoc*.

- *let $umlDoc := doc(uml:getUMLFileName())*
- *let $umlActors := $umlDoc//UML:Actor*

O (ii) bloco de iteração contém a camada lógica do XQuery. Pode-se dividir o bloco de iteração em montagem de variáveis, condições, ações de identificação de relações de rastreabilidade e ações de identificação de inconsistências. A seguir a descrição:

- Montagem de variáveis – bloco em que uma variável é populada com um item de uma lista que está sendo percorrida. No exemplo abaixo a variável *$generalization* será populada com um novo elemento da lista *$generalizations* a cada iteração do laço de repetição for. O mesmo ocorre na variável *$actor* em relação a lista *$actorWithGeneralization*.
  - *for $generalization in $generalizations, $actor in $actorWithGeneralization*

- Condições – definem as condições necessárias para que uma regra seja dada como satisfeita. As condições podem ser representadas em estruturas *where*, ou como parte da expressão *if-then-else*, do XQuery. Este bloco de código pode utilizar funções estendidas em Java para contribuir com a identificação de relações de rastreabilidade e inconsistências. No exemplo abaixo a condição é considerada verdadeira se o resultado da função estendida *isSynonyms* for verdadeiro.
  - where(syn:isSynonyms($umlActor/@name,$specificationActor))

- Ações de identificação de relações de rastreabilidade – retorna um elemento XML para o componente *Motor Rastreabilidade_Analise_Completude* com as informações da relação de rastreabilidade criada. Este bloco contém o tipo da relação (*type*), um identificador único para a regra (*ruleID*), um percentual que indica o grau de confiabilidade do relacionamento criado (*degreeOfCompleteness*) e as informações sobre os elementos de origem e destino. Os elementos de origem e destino possuem informações sobre o modelo ao qual pertencem (*doc*), o tipo da relação (*type*), o nome (*name*) e um identificar para o elemento (id). O *degreeOfCompleteness* representa a porcentagem de condições satisfeitas em uma regra. Assim se o *degreeOfCompleteness* for 77% significa que 77% das condições existentes na regra foram satisfeitas. Nesse caso a relação de rastreabilidade foi encontrada, porém existem divergências entre os elementos dos modelos analisados na regra.

- Ações de identificação de inconsistências – retorna um elemento XML para o componente *Motor Rastreabilidade_Analise_Completude* descrevendo informações sobre elementos divergentes (ou em falta) entre os modelos em comparação. Esse bloco contém as informações de identificador (*idSource*), tipo (*typeSource*), nome (*nameSource*) e documento (*docSource*) do modelo origem e o tipo (*typetarget*) e documento (*docTarget*) do modelo no qual o elemento encontra-se divergente (ou inexistente).

O *template* de regras apoia a criação de dois tipos de regras de rastreabilidade: Regra para criação de relações de rastreabilidade e regras para identificação de inconsistências.

O *template* permite ainda que uma mesma regra possa identificar relações de rastreabilidade e inconsistências em um mesmo momento, porém optou-se por não criar regras desse formato para simplificar a leitura e entendimento das regras.

A seguir são exibidos exemplos dessas regras de rastreabilidade.

5.2.1 Regra para criação de relações de rastreabilidade

A Figura 8 apresenta uma regra criada para identificar relações de rastreabilidade entre casos de uso do Diagrama de Casos de Uso e requisitos funcionais da Especificação de Casos de Uso. A fim de detalhar a regra, optou-se por dividi-la em várias partes que serão explicadas a seguir.

```
<Rule id="Rule2"
        priority="1"
        type="sobreposicao"
        elementTypeA="UML caso de uso"
        elementTypeB="Especificacao requisito funcional"
        description="Esta regra identifica as relacoes entre caso de uso em diagrama UML de
        casos de uso e requisito funcional em especificacao de casos de uso">
        <XQuery>
            <![CDATA[
                declare namespace JUDE = "http://objectclub.esm.co.jp/Jude/namespace/";
                declare namespace UML = "org.omg.xmi.namespace.UML";

                declare namespace f = "java:iTrace.XQueryFunctions";
                declare namespace syn = "java:iTrace.XQuerySynonymsFunctions";
                declare namespace sim = "java:iTrace.XQuerySimilarityFunctions";
                declare namespace cc = "java:iTrace.XQueryCompletenessCheckingFunctions";
                declare namespace uml = "java:iTrace.XQueryUMLFunctions";
                declare namespace spec = "java:iTrace.XQuerySpecificationFunctions";

                declare namespace xmi="http://www.omg.org/XMI";

                let $umlDoc := doc(uml:getUMLFileName())
                let $umlUseCases := $umlDoc//UML:UseCase

                let $specificationDoc := doc(spec:getSpecificationFileName())
                let $specificationRequiriments := $specificationDoc//useCase

                for $umlUseCase in $umlUseCases, $specificationRequiriment in
                $specificationRequiriments
                    where(syn:isSynonyms(
                    $umlUseCase/@name,$specificationRequiriment/name/text()))
                    return
                     <TraceabilityRelation type="sobreposicao" ruleID="Rule2"
                     degreeOfCompleteness="100">
                                <Element doc="{uml:getUMLFileName()}"
                                         type="Especificacao requisito funcional"
                                         name="{$umlUseCase/@name}"
                                         id="{$umlUseCase/@xmi.id}">
                                </Element>
                                <Element doc="{spec:getSpecificationFileName()}"
                                         type="UML caso de uso"
                                         name="{$specificationRequiriment/name/text()}"
                                         id="{$specificationRequiriment/identifier/text()}">
                                </Element>
                    </TraceabilityRelation>
            ]]>
        </XQuery>
    </Rule>
```

Figura 8 - Regra (*Rule2*) para identificar as relações entre casos de uso do Diagrama de Casos de Uso e requisitos funcional da Especificação de Casos de Uso

A Figura 9 mostra o elemento de inicialização da regra (<*Rule*>) e suas propriedades: (i) o identificar único da regra (*id="Rule2"*); (ii) a prioridade de execução da regra (*priority="1"*); (ii) o tipo da relação criada, baseada no modelo de referência (*type="sobreposicao"*); (iv) o elemento origem (*elementTypeA="UML caso de uso"*); (v) o elemento destino (*elementTypeB="Especificacao requisito funcional"*); e (vi) uma breve descrição da regra (*description="Esta regra identifica as relacoes entre caso de uso em diagrama UML de casos de uso e requisito funcional em especificacao de casos de uso"*).

```
<Rule id="Rule2"
        priority="1"
        type="sobreposicao"
        elementTypeA="UML caso de uso"
        elementTypeB="Especificacao requisito funcional"
        description="Esta regra identifica as relacoes entre caso de uso em diagrama UML de
        casos de uso e requisito funcional em especificacao de casos de uso">
        <XQuery>
        </XQuery>
</Rule>
```

Figura 9 - Cabeçalho da regra (*Rule2*)

A Figura 10 mostra a declaração de *namespace* da regra. Cada regra contém um elemento XQuery (*<XQuery>*) que contém todo o código XQuery. O código XQuery está dentro de uma seção CDATA para evitar que os textos contidos nesse grupo sejam confundidos com elementos XML. Nesse bloco de código, os *namespaces* utilizados na regra são declarados. Por exemplo, os *namespaces f*, *syn*, *sim*, *cc*, *uml* e *spec* permitem que o código XQuery acessem funções implementadas nas classes *XQueryFunctions*, *XQuerySynonymsFunctions*, *XQueryCompletenessCheckingFunctions*, *XQueryUMLFunctions* e *XQuerySpecificationFunctions* respectivamente. As funções disponibilizadas por essas classes são descritas em detalhes na seção 5.3. Um exemplo de função implementada na classe *XQueryUMLFunctions* e utilizada na regra *Rule2* é *getUMLFileName*. Os *namespaces* JUDE, UML e XMI são *namespaces* especiais que precisam ser declarados para que seja possível acessar os elementos do documento Astah (*namespaces* JUDE e UML) e algumas expressões XPath (*namespace* XMI).

```
<XQuery>
    <![CDATA[
        declare namespace JUDE = "http://objectclub.esm.co.jp/Jude/namespace/";
        declare namespace UML = "org.omg.xmi.namespace.UML";

        declare namespace f = "java:iTrace.XQueryFunctions";
        declare namespace syn = "java:iTrace.XQuerySynonymsFunctions";
        declare namespace sim = "java:iTrace.XQuerySimilarityFunctions";
        declare namespace cc = "java:iTrace.XQueryCompletenessCheckingFunctions";
        declare namespace uml = "java:iTrace.XQueryUMLFunctions";
        declare namespace spec = "java:iTrace.XQuerySpecificationFunctions";

        declare namespace xmi="http://www.omg.org/XMI";
```

Figura 10 - Declarações de *namespaces* da regra (*Rule2*)

A Figura 11 mostra a declaração de variáveis para a regra *Rule2*. A declaração das variáveis *$umlDoc* e *$specificationDoc* atribui o modelo de Diagrama de Casos de Uso e Especificação de Casos de Uso respectivamente. À variável *$umlUseCases* é atribuído uma lista de casos de usos existente na variável *$umlDoc* por meio de uma consulta XPath (*$umlDoc//UML:UseCase*). À última variável *$specificationRequiriments* é atribuído a lista

de requisitos existente na variável *$specificationDoc* por meio de uma consulta XPath (*$specificationDoc//useCase*).

```
let $umlDoc := doc(uml:getUMLFileName())
let $umlUseCases := $umlDoc//UML:UseCase

let $specificationDoc := doc(spec:getSpecificationFileName())
let $specificationRequiriments := $specificationDoc//useCase
```

Figura 11 - Declarações de variáveis da regra (*Rule2*)

A Figura 12 mostra a parte de iteração da regra *Rule2*. O laço de repetição for percorre todos os elementos das listas *$umlUseCases* e *$specificationRequiriments* e gera uma validação *where* que é considerada verdadeira quando o nome do elemento da variável *$umlUseCase* é considerado sinônimo em relação ao nome do elemento existente na variável *$specificationRequiriment*, pela função estendida *isSynonyms*.

```
for $umlUseCase in $umlUseCases, $specificationRequiriment in $specificationRequiriments
where(syn:isSynonyms($umlUseCase/@name,$specificationRequiriment/name/text()))
    return
        ...
```

Figura 12 - Verificação condicional da regra (*Rule2*)

A Figura 13 mostra o resultado da relação de rastreabilidade criado na regra *Rule2*. A regra cria um elemento (*TraceabilityRelation*) quando a condição da regra é considerada verdadeira. O elemento *TraceabilityRelation* retorna um XML que fica armazenado em *Relacoes_Rastreabilidade_Elementos_Divergentes*. O elemento *TraceabilityRelation* contém as informações do tipo da ralação (*type="sobreposicao"*), o identificador da regra (*ruleID="Rule2"*) e o grau de integridade entre os elementos origem e destino dessa regra (*degreeOfCompleteness="100"*).

Na Figura 13 o grau de integridade está fixado em 100, mas em outros casos em que existem dependências de elementos, este valor será calculado com base nas semelhanças entre os sub elementos do elemento origem e elemento destino.

```
<TraceabilityRelation type="sobreposicao" ruleID="Rule2"
degreeOfCompleteness="100">
            <Element doc="{uml:getUMLFileName()}"
                     type="Especificacao requisito funcional"
                     name="{$umlUseCase/@name}"
                     id="{$umlUseCase/@xmi.id}">
            </Element>
            <Element doc="{spec:getSpecificationFileName()}"
                     type="UML caso de uso"
                     name="{$specificationRequiriment/name/text()}"
                     id="{$specificationRequiriment/identifier/text()}">
            </Element>
</TraceabilityRelation>
```

Figura 13 - Relação de rastreabilidade identificado pela regra (*Rule2*)

Na Figura 13 os elementos origem e destino são inseridos em um agrupador (*Element*) que contém o modelo de origem (*doc*), o tipo do modelo (*type*), o nome do elemento (*name*) e o identificador único deste elemento (*id*).

5.2.2 Regras para verificação de integridade

A Figura 14 mostra um exemplo de regra (*Rule7*) que analisa elementos divergentes (ou inexistentes) entre atores do Diagrama UML e atores da Especificação de Casos de Uso (verificação de integridade). Não será descrito, em detalhes, cada bloco da regra, pois já foi feito na subseção 5.2.1, ao explicar a regra *Rule2*.

```
<Rule id="Rule7"
    priority="3"
    type="sobreposicao"
    elementTypeA="UML Ator"
    elementTypeB="Especificacao Ator"
    description="Esta regra identifica elementos divergentes entre atores do diagrama UML e da especificacao de casos de uso">
    <XQuery>
        <![CDATA[
            declare namespace JUDE = "http://objectclub.esm.co.jp/Jude/namespace/";
            declare namespace UML = "org.omg.xmi.namespace.UML";

            declare namespace f = "java:iTrace.XQueryFunctions";
            declare namespace syn = "java:iTrace.XQuerySynonymsFunctions";
            declare namespace sim = "java:iTrace.XQuerySimilarityFunctions";
            declare namespace cc = "java:iTrace.XQueryCompletenessCheckingFunctions";
            declare namespace uml = "java:iTrace.XQueryUMLFunctions";
            declare namespace spec = "java:iTrace.XQuerySpecificationFunctions";

            declare namespace xmi="http://www.omg.org/XMI";

            let $umlDoc := doc(uml:getUMLFileName())
            let $umlActors := $umlDoc//UML:Actor

            let $specificationDoc := doc(spec:getSpecificationFileName())
            let $specificationActors := distinct-values($specificationDoc//actor/text())

            let $traceabilityDoc := doc(f:getTraceabilityFileName())
            let $sobreposicaoAtores := $traceabilityDoc//TraceabilityRelation[@type='sobreposicao']/Element[@type='Especificacao Ator']/@name

            for $specificationActor in $specificationActors
                where (exists(index-of($sobreposicaoAtores, $specificationActor/text())))
                    <TraceabilityRelation
                        type="sobreposicao"
                        ruleID="Rule7"
                        degreeOfCompleteness="0">
                        <MissingElement
                            typeSource="UML Ator"
                            idSource="{$specificationActor/text()}"
                            nameSource="{$specificationActor/text()}"
                            docSource="{spec:getSpecificationFileName()}"
                            typeTarget="Especificacao Ator"
                            docTarget="{uml:getUMLFileName()}">
                        </MissingElement>
                    </TraceabilityRelation>
        ]]>
    </XQuery>
</Rule>
```

Figura 14 - Regra (*Rule7*) para identificar atores divergentes entre o Diagrama de Casos de Uso e Requisitos

Pode-se notar na Figura 14 que a esta regra obrigatoriamente precisa ter prioridade maior que um, pois eles dependem de resultado de regras anteriores, neste caso a regra em questão tem prioridade igual a 3 (*priority="3"*). Na declaração das variáveis verifica-se que a variável *$traceabilityDoc* recebe o documento temporário dos resultados das execuções das regras. Em sequência, uma nova variável *$actorsRelationsActuals* recebe o resultado de uma consulta XPath que capturar os relacionamentos do tipo sobreposição e que o tipo do elemento origem seja igual a Ator (*UML Ator*).

Por fim, a Figura 15 exibe o resultado em um XML gerado a partir da identificação de um elemento do modelo de origem que não coincide com nenhum elemento do modelo de destino. O elemento *TraceabilityRalation* possui todas as propriedades explicadas na subseção 5.2.1, porém a propriedade *degreeOfCompletness* terá o valor zero, informando que não foi possível satisfazer nenhuma regra para relacionar o elemento em questão.

```
<TraceabilityRelation
    type="sobreposicao"
    ruleID="Rule7"
    degreeOfCompleteness="0">
        <MissingElement
            typeSource="UML Ator"
            idSource="{$specificationActor/text()}"
            nameSource="{$specificationActor/text()}"
            docSource="{spec:getSpecificationFileName()}"
            typeTarget="Especificacao Ator"
            docTarget="{uml:getUMLFileName()}">
        </MissingElement>
</TraceabilityRelation>
```

Figura 15 - Identificação de elemento divergente (*Rule7*)

O elemento origem então é inserido em um agrupador (*MissingElement*) que contém os atributos: (i) tipo do elemento origem (*typeSource*); (ii) o identificador único do elemento origem (*idSource*); (iii) o modelo do elemento origem (*docSource*); (iv) o tipo do elemento destino, que se encontra em divergência ou que não foi encontrado (*typeTarget*); e (v) o modelo que o elemento destino encontra-se ou deveria ser encontrado (*docTarget*).

5.3 Funções estendidas

Como já mencionado anteriormente, o *framework* proposto estende alguns métodos criadas em Java para suportar as regras que relacionam as Especificações de Casos de Uso com os Diagramas de Casos de Uso. Esses métodos foram divididos nas seguintes classes e podem ser vistas na Figura 16.

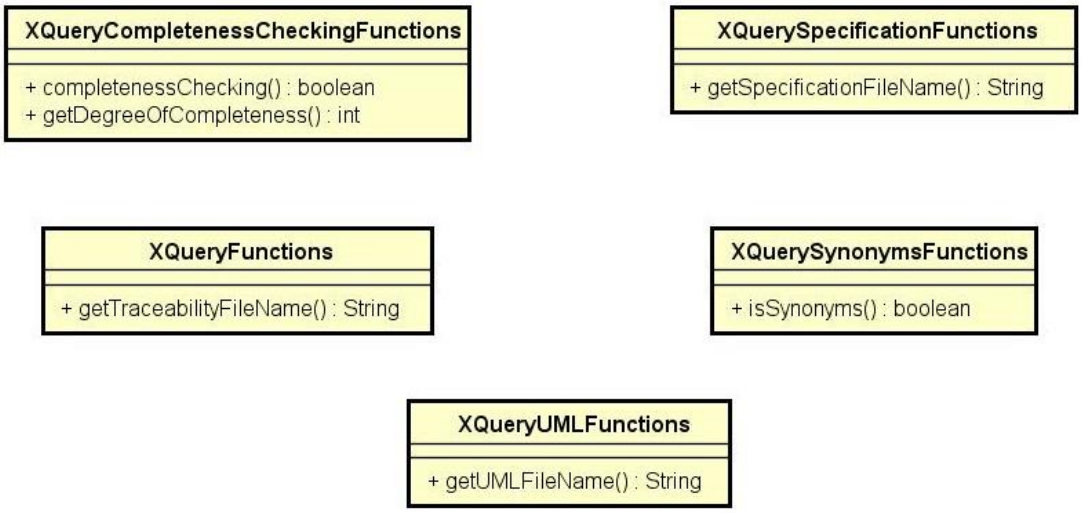

Figura 16 - Classes Java criadas para estender o XQuery

- Classe *XQueryCompletenessCheckingFunctions* – Contém métodos escritos em Java que estendem o XQuery para suportar as regras de verificação de integridade entre os modelos.
- Classe *XQueryFunctions* – Estende o XQuery para criar métodos utilizados para executar procedimentos gerais, como por exemplo, capturar o arquivo de resultados preliminares, criado e atualizado ao decorrer da execução das regras, por meio da função *getTraceabilityFileName*.
- Classe *XQuerySpecificationFunctions* – Contém métodos Java que estendem do XQuery para manipular elementos da Especificação de Casos de Uso.
- Classe *XQuerySynonymsFunctions* – Contém métodos escritos em Java que estendem XQuery para verificar, na base de dados WordNet, se os nomes dos elementos em análise podem ser considerados sinônimos.
- Classe *XQueryUMLFunctions* – Contém métodos Java que estendem do XQuery para manipular elementos do Diagrama de Casos de Uso.

Para utilizar as funcionalidades estendidas do XQuery existentes em cada uma dessas classes, é necessário declará-las e em seguida invocar o nome do método que desejado. A Figura 16 mostra quando a classe *XQueryUMLFunctions* é declarada na lista de *namespaces* (*declare namespace uml = "java:iTrace.XQueryUMLFunctions"*) e em seguida exibe o processo necessário para invocar o método *getUMLFileName* (*uml:getUMLFileName()*).

```
declare namespace uml = "java:iTrace.XQueryUMLFunctions";

let $umlDoc := doc(uml:getUMLFileName())
```

Figura 17 - Chamada ao método *getUMLFileName* da classe Java estendida

A seguir os métodos existentes em cada uma dessas classes são detalhados.

5.3.1 XQueryCompletenessCheckingFunctions

A classe *XQueryCompletenessCheckingFunctions* estende do XQuery com funções para auxiliar na validação de integridade. A principal função da classe é *completenessChecking* que verifica se uma lista de elementos A contém elementos sinônimos de uma lista de elementos B. O *getDegreeOfCompleteness* retorna o grau de semelhança entre

duas listas, assim quando todos os elementos da lista A são sinônimos de elementos da lista B então esta função retorna 1.00.

5.3.2 XQueryFunctions

A classe *XQueryFunctions* estende XQuery com funções para processos gerais. Esta classe contém a funcionalidade *getTraceabilityFileName*, responsável por retornar o documento temporário dos resultados de rastreabilidade.

5.3.3 XQuerySpecificationFunctions

A classe *XQuerySpecificationFunctions* contém funções para manipular elementos do documento de Especificação de Casos de Uso. A função *getSpecificationFileName* retorna o documento deste tipo enviado pelo usuário.

5.3.4 XQuerySynonymsFunctions

A classe *XQuerySynonymsFunctions* estende XQuery com funções para verificar se os nomes de elementos nos modelos são sinônimos. Esta classe contém o método *isSynonyms* que recebe dois termos como argumento e verifica se estes elementos são considerados sinônimos. A função *isSynonyms* utiliza como base da verificação de sinônimos a o WordNet.

5.3.5 XQueryUMLFunctions

A classe *XQueryUMLFunctions* contém funções para manipular elementos do Diagrama de Casos de Uso. A função *getUMLFileName* retorna o Diagrama de Casos de Uso enviado pelo usuário.

5.4 Fermenta protótipo

A fim de apoiar e testar o *framework* de rastreabilidade proposto, foi desenvolvida uma ferramenta protótipo para identificar automaticamente as relações de rastreabilidade e identificação de inconsistências entre a Especificação de Casos de Uso e o Diagrama de Casos

de Uso UML. A ferramenta protótipo de rastreabilidade (Figura 18) permite que os usuários criem novos projetos e enviem como entrada os modelos e as regras criadas para serem processadas e assim identificar as relações de rastreabilidade e inconsistências entre os modelos. A ferramenta protótipo é capaz de gerar uma matriz com os resultados da rastreabilidade, além de permitir a visualização das regras enviadas.

Para começar um novo projeto na ferramenta, o usuário precisa selecionar a opção *File* na área de menus e escolher a opção *New Project* no submenu (Figura 18).

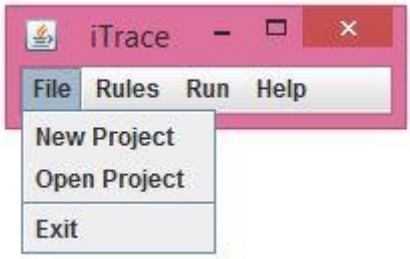

Figura 18 - Visão inicial e menu de criação de um novo projeto na ferramenta protótipo

Após escolher a opção para criar um novo projeto, uma janela é aberta (Figura 19) para que o usuário possa selecionar os locais em que se encontram os modelos de Especificação de Casos de Uso, Diagrama de Casos de Uso e Regras responsáveis pela identificação das relações de rastreabilidade e elementos divergentes entre os modelos.

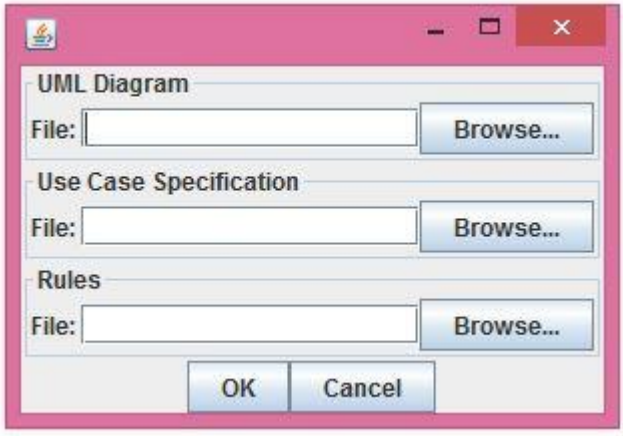

Figura 19 - Escolhendo os modelos de entrada durante a criação de um novo projeto na ferramenta protótipo

Para gerar as relações de rastreabilidade entre os modelos selecionados, o usuário seleciona o menu *Run* e o submenu *Run* (Figura 20). Após esse comando, a ferramenta protótipo executará as regras e criará o arquivo output.xml que conterá o resultado da execução das regras, este arquivo conterá todas relações de rastreabilidade identificadas e a lista dos elementos divergentes.

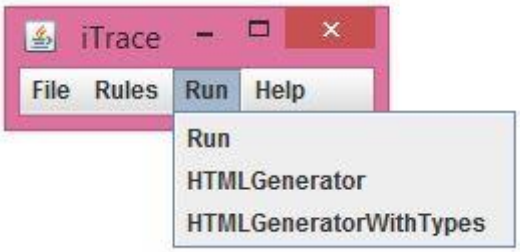

Figura 20 - Criando relações de rastreabilidade e identificando elementos divergentes na ferramenta protótipo

Para exemplificar, a Figura 21 exibe um trecho do código do arquivo output.xml gerado que contém as relações de rastreabilidade identificadas e a lista dos elementos divergentes.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Traceability>
    <TraceabilityRelation ruleID="Rule1"
        degreeOfCompleteness="100" type="sobreposicao">
        <Element id="yd-1876a66e066cf6f730d3e0275761e853" name="employee"
            doc="file:///C:/Users/Jefferson/Documents/BurgerDigital ENG - UML.xml"
            type="Especificacao Ator" />
        <Element id="employee" name="employee"
            doc="file:///C:/Users/Jefferson/Documents/BurgerDigital ENG - especificacao.xml"
            type="UML Ator" />
    </TraceabilityRelation>
    <TraceabilityRelation ruleID="Rule1"
        degreeOfCompleteness="100" type="sobreposicao">
        <Element id="2xd-6a23f988b0e96e4ca1158c58dc4e38ba" name="clients"
            doc="file:///C:/Users/Jefferson/Documents/BurgerDigital ENG - UML.xml"
            type="Especificacao Ator" />
        <Element id="Customer" name="Customer"
            doc="file:///C:/Users/Jefferson/Documents/BurgerDigital ENG - especificacao.xml"
            type="UML Ator" />
    </TraceabilityRelation>
    ...
</Traceability>
```

Figura 21 - Arquivo output.xml

Após realizar o procedimento de execução das regras e ter gerado o arquivo output.xml, o usuário pode exportar o resultado das relações de rastreabilidade e elementos divergentes em para uma página HTML que estará formatada em formato de Matriz de Rastreabilidade. Para realizar esse procedimento é preciso escolher o menu *Run* e o submenu *HTML Generator*.

A Figura 22 mostra um modelo do HTML gerado a partir da seleção do submenu *HTML Generator* pelo usuário, após ter criado o arquivo *output.xml*.

```
Root element of the doc is Traceability
Total of traceability relations: 29
<table class="sofT" cellspacing="0">
    <td colspan="3" class="TopHed">
    Traceability Relations Types between Use Case Specification and UML Diagram
    </td>
    <tr>
      <td class="helpHed">Rule ID</td>
      <td class="helpHed">UML Ator</td>
      <td class="helpHed">Especificacao Ator</td>
    </tr>
    <tr>
      <td class="helpBod">Rule1</td>
      <td class="helpBod">employee</td>
      <td class="helpBod">employee</td>
    </tr>
    <tr>
      <td class="helpBod">Rule1</td>
      <td class="helpBod">clients</td>
      <td class="helpBod">Customer</td>
    </tr>
    <tr>
      <td class="helpBod">Rule1A</td>
      <td class="helpBod">administrator</td>
      <td class="helpBod">employee</td>
    </tr>
</table>
```

Figura 22 - HTML gerado pela ferramenta protótipo a partir do resultado das relações e rastreabilidade

O HTML gerado pode ser importado para a ferramenta iTraceWeb (2015), a mesma que foi utilizada para criar o documento de Especificação de Casos de Uso, para visualizar a matriz de resultados formatada, facilitando assim a visualização das relações de rastreabilidade geradas (Figura 23).

Root element of the doc is Traceability Total of traceability relations: 29

Traceability Relations Types between Use Case Specification and UML Diagram

Rule ID	UML Ator	Especificacao Ator
Rule1	employee	employee
Rule1	clients	Customer
Rule1A	administrator	employee
Rule ID	UML caso de uso	Especificacao requisito funcional
Rule2	delete+request	Cancel+request
Rule2	generate+Report	generate+Report
Rule2	generate+Report	Generating+sales+report
Rule2	generate+Report	Generate+performance+report
Rule2	Generating+sales+report	generate+Report
Rule2	Generating+sales+report	Generating+sales+report
Rule2	Generate+performance+report	generate+Report
Rule2	Generate+performance+report	Generate+performance+report

Figura 23 - HTML gerado pela ferramenta protótipo e enviado paro o iTraceWeb

Caso o usuário deseje que a matriz de rastreabilidade contenha também informações sobre o tipo da relação de rastreabilidade utilizada por cada regra é preciso escolher o menu *Run* e o submenu *HTML Generator With Types*.

O usuário pode selecionar o menu *Rules* e o submenu *Show Rules* (Figura 24) para visualizar as regras utilizadas no projeto.

Figura 24 - Menu de visualização das regras

A Figura 25 exibe a listagem das regras no lado esquerdo da tela e os detalhes da regra selecionada (*Rule1*) ao lado direito.

Figura 25 - Listagem das regras

5.5 Discussão sobre o *framework*

Este trabalho assemelha-se ao trabalho proposto por Filho (2011), visto que é utilizado uma simplificação da arquitetura proposta em seu estudo, além de que também é utilizada uma abordagem orientada a regras, contudo, a abordagem de Filho não contempla os artefatos criados sobre o domínio Orientado a Objetos, focando apenas em artefatos do domínio Orientado a Agentes. Assim, fez-se necessário estender sua abordagem para contemplar

outros artefatos de um novo domínio, sendo necessário para isso a criação de regras específicas e a elaboração de um novo modelo de referência que detalhe os relacionamos existentes entre os modelos analisados.

À princípio, esta pesquisa utilizaria um *template* de regra próprio, assim como foi iniciado também a construção de uma ferramenta protótipo sem nenhum relacionamento com trabalhos anteriores. Contudo, foi observado ao longo da pesquisa que vários conceitos convergiam com o estudo apresentado por Filho, assim optou-se por reaproveitar o *template* de regras por ele proposto e adaptar, para atender às necessidades desta pesquisa, a ferramenta por ele apresentada.

Assim este trabalho se diferencia da proposta de Filho (2011) principalmente pelo domínio para o qual é aplicado (ou seja, sistemas orientados a objetos). Outro diferencial deste trabalho é quanto ao modelo de referência. Em seu estudo, Filho propôs um modelo de referência formatado em matriz, embora sejam utilizadas relações de rastreabilidade similares, foi apresentado o modelo de referência em formato de Diagrama de Classes UML, utilizando dos benefícios da UML para facilitar a compreensão da proposta.

5.6 Considerações finais

Este capítulo exibiu uma visão geral do *framework* de rastreabilidade baseado em regras criado para capturar relações de rastreabilidade automaticamente entre elementos da Especificação de Casos de Uso e Diagramas de Casos de Uso UML e identificar inconsistências entre esses modelos. Foi descrito o *template* das regras, explicando cada bloco de código e os elementos contidos. Foi explicado o motivo que levou a decisão de utilizar as funções Java estendidas do XQuery para apoiar as regras e foram detalhadas cada uma das funções criadas. Por fim, foi realizada uma descrição sobre a ferramenta protótipo criada.

6 Avaliação e resultados

Foi realizada a avaliação desta abordagem por meio de dois exemplos de aplicação: (i) BurguerDigital e (ii) Projeto Pesquisa e Movimento.

A decisão do uso desses exemplos de aplicação deu-se pela necessidade de testes em um exemplo simples para que pudessem ser validadas as regras criadas, tanto no âmbito das relações de rastreabilidade quanto para a análise de inconsistências (identificação de elementos divergentes ou em falta) e outro exemplo de aplicação mais complexo, que preferencialmente tivesse sido implantado em alguma empresa ou instituição, para validar a eficácia deste estudo baseando-se nos resultados obtidos pelas métricas *precision* e *recall*.

O BurguerDigital é uma abstração de um sistema de gestão de vendas e controle de estoque de lanchonetes, com ênfase em venda de hambúrguer. O BurguerDigital é um pequeno exemplo de aplicação que foi construído visando os testes das regras de rastreabilidade e identificação de inconsistências. Assim, o BurguerDigital possui os elementos necessário para validar a eficiência das regras de rastreabilidade e identificação de inconsistências entre os modelos em análise.

O Projeto Pesquisa e Movimento é um sistema real que foi desenvolvido como protótipo para a Universidade Federal Rural de Pernambuco, a fim de auxiliar o processo de solicitações e reservas de veículos por pesquisadores da instituição. Esse exemplo de aplicação foi desenvolvido por estudantes do curso de Mestrado em Informática Aplicada da própria universidade. Por ser um projeto de complexidade mais elevada, acredita-se que ele auxiliará na validação desta abordagem.

Ambos exemplos de aplicação tiveram os modelos de Diagrama de Caso de Uso e Documento de Especificação de Casos de Uso construídos no Astah e no iTraceWeb respectivamente e exportados os modelos para o padrão XML. O BurguerDigital teve seus modelos construídos ao longo desta pesquisa, com o intuito de validar as regras. O Projeto Pesquisa e Movimento, por sua vez, já possuía esses modelos desenvolvido anteriormente pelos estudantes que o construíram, contudo foi necessário migrar as informações existentes para as ferramentas mencionadas anteriormente.

A seguir são descritos os métodos utilizados para avaliar este trabalho e é apresentado de modo geral os exemplos de aplicação e o resultado da avaliação de cada um deles.

6.1 Critérios de avaliação

A avaliação desta pesquisa foi realizada com o intuído de demonstrar a hipótese apresentada no Capítulo 1. Especificamente, esta pesquisa foi avaliada para demonstrar que:

a) É possível gerar automaticamente relações de rastreabilidade entre os documentos de Especificação de Casos de Uso e Diagramas de Casos de uso;
b) Pode-se identificar automaticamente inconsistências entre os documentos de Especificação de Casos de Uso e Diagramas de Casos de uso;
c) Pode-se utilizar informações geradas a partir da análise de inconsistências para corrigir a integridade entre os elementos dos diferentes documentos, melhorando assim sua consistência;
d) Pode-se utilizar informações geradas a partir da análise de inconsistências para melhorar o número de relações de rastreabilidade identificadas pela ferramenta.

Para avaliar (a), são aferidas as métricas *precision* e *recall*, por meio de comparações entre os relacionamentos manuais, previamente realizados e livre de erros, com as relações de rastreabilidade geradas pela ferramenta protótipo. A utilização do *precision* e *recall* como métricas de validação de abordagens automáticas de relações de rastreabilidade tem sido defendida na literatura (ALI; GUEHENEUC; ANTONIOL, 2011), (LUCIA *et al*., 2011), (FILHO, 2011), (DAGENAIS; ROBILLARD, 2012), (ALI; GUENEUC; ANTONIOL, 2013). São utilizadas as seguintes relação de *precision* e *recall* apresentadas em Ali, Gueneuc e Antoniol (2013):

$$Precision = \frac{|\{\text{linksrelevantes}\} \cap \{\text{linksrecuperados}\}|}{|\{linksrecuperados\}|}$$

$$Recall = \frac{|\{\text{linksrelevantes}\} \cap \{\text{linksrecuperados}\}|}{|\{\text{linksrelevantes}\}|}$$

Onde, os ***links* recuperados** referem aos números das relações de rastreabilidade identificadas pela ferramenta e ***links* relevantes** são os números das relações de rastreabilidade geradas, previamente, de forma manual (considerando que todas as relações de rastreabilidade e inconsistências, identificadas manualmente, estão livres de erros).

A relação $|\{linksrelevantes\} \cap \{linksrecuperados\}|$ identifica o número de relações em comuns identificas pela ferramenta protótipo e pelo relacionamento manual, respectivamente.

A fim de avaliar (b) foram aferidas as métricas *precision* e *recall*, por meio de comparações entre os relacionamentos manuais, previamente realizados e livre de erros, com

os elementos inconsistentes identificados pela ferramenta protótipo. Após os primeiros resultados dessa avaliação, os modelos analisados foram modificados, na tentativa de correções de erros de inconsistências e em seguida, os modelos modificados foram submetidos a nova análise para verificar se houve alterações nos valores de *precision* e *recall*.

A fim de avaliar (c) foram apresentados exemplos em que os resultados da identificação de elementos inconsistentes foram utilizados para corrigir a integridade entre os modelos analisados, melhorando assim a integridade desses artefatos.

Por fim, para avaliar (d) foram aprimorados os modelos de análise, levando em consideração informações de elementos inconsistentes. Os modelos ajustados foram submetidos a uma nova avaliação para verificar possíveis ocorrências de melhorias no *precision* e *recall*.

Um número baixo de *precision* indica dizer que a ferramenta protótipo está identificando muitos relacionamentos, só que a maioria desses relacionamentos estão incorretos.

Um número baixo de *recall* indica que a ferramenta protótipo está encontrando um número de relacionamento muito abaixo do que deveria.

Esta pesquisa considerará valores de *precision* e *recall* próximo a 70% como aceitáveis. Esse valor foi definido com base nos resultados de outros estudos que utilizam essas métricas em seu processo de análise (FILHO, 2011), (LUCIA *et al*., 2011), (DAGENAIS; ROBILLARD, 2012).

A seguir, são apresentados o BurguerDigital e o Projeto Pesquisa e Movimento, com informações dos resultados obtidos dessa avaliação.

6.2 BurguerDigital

6.2.1 Visão geral

Esta sessão descreve a documentação de um sistema Orientado a Objetos, o primeiro exemplo de aplicação, implementa uma lanchonete especializada na venda de hambúrguer. O BurguerDigital permite as principais atividades de venda, gestão de estoque e gerenciamento de relatórios.

O BurguerDigital (Figura 2) permite que os Empregados acessem a aplicação (Conectar ao sistema) por meio de credenciais de acesso, como usuário e senha. Os

empregados poderão registrar a venda de produtos, para isso precisam registrar um pedido (Criar pedido). O empregado poderá também gerenciar os ingredientes (Administrar ingredientes) que compõe os hambúrgueres. Para esse processo é possível buscar um ingrediente (Buscar ingredientes) através de seu nome ou adicionar novos ingredientes (Registrar ingredientes). Antes de completar o cadastro de um novo ingrediente, é necessário verificar a disponibilidade (Checar disponibilidade) deste item com o um Fornecedor.

O Administrador, além de poder realizar todas as funcionalidades disponíveis ao Empregado, poderá cancelar pedidos (Remover pedido) que tenham sido cadastrados com erros ou que precise ser cancelado a pedido dos clientes.

O BurguerDigital também permite que o Administrador da lanchonete obtenha relatórios (Gerar relatório). Os tipos de relatórios disponíveis no BurguerDigital são: relatório de venda (Gerar relatório de vendas) e relatório de desempenho dos empregados (Gerar relatório de desempenho).

Aos clientes (Cliente) do BurguerDigital é disponibilizado um serviço que permite que os próprios possam registrar um pedido (Criar pedido), sem ter que para isso se dirigir a um funcionário.

6.2.2 Artefatos

O documento de Especificação de Casos de Uso foi desenvolvido utilizando a ferramenta iTraceWeb, contemplando o *template* descrito no Capítulo 2.

A Tabela 4 mostra o tipo e o número de elementos existente no documento de Especificação de Casos de Uso do BurguerDigital.

Tabela 4 - Elementos do documento de Especificação de Casos de Uso de BurguerDigital.

Tipo de elemento	Número de elementos
Ator	3
Requisito funcional	11
Fluxos (principal, secundário e exceção)	20

O Diagrama de Casos de Uso foi desenvolvido utilizando a ferramenta Astah, em uma versão de estudante, pois esta permitia extrair o modelo criado para o padrão XML.

A Tabela 5 mostra o tipo e o número de elementos existente no Diagrama de Casos de Uso do BurguerDigital.

Tabela 5 - Elementos do Diagrama de Casos de Uso de BurguerDigital.

Tipo de elemento	Número de elementos
Ator	4
Caso de Uso	11

6.2.3 Avaliação

O exemplo de aplicação BurguerDigital foi utilizado para avaliar esta pesquisa, a fim de:

- Avaliar o *precision* e *recall* da ferramenta protótipo;
- Identificar inconsistências;
- Demonstrar a abordagem em um projeto de tamanho médio.

Este exemplo de aplicação também foi utilizado para validar as regras criadas para identificar relações de rastreabilidade e inconsistências.

Para analisar a eficácia da ferramenta protótipo, foram realizadas as seguintes atividades: (i) identificação manual das relações de rastreabilidade e inconsistentes existentes entre os documentos de Especificação de Casos de Uso e do Diagrama de Casos de Uso; (ii) processamento das regras de rastreabilidade pela ferramenta protótipo, a fim de identificar automaticamente as relações de rastreabilidade e inconsistências existentes entre os modelos; e (iii) analisar a eficiência dos resultados através das métricas *precision* e *recall* obtidos a partir da comparação de (i) e (ii).

A análise automática das relações de rastreabilidade foi realizada através da execução de onze regras de rastreabilidade criadas durante esta pesquisa para identificar relacionamentos entre os modelos de Especificação de Casos de Uso e Diagrama de Casos de Uso, e cinco regras para identificar inconsistências entre os mesmos modelos.

A Tabela 6 mostra os resultados obtidos do relacionamento entre os modelos de Especificação de Casos de Uso e Diagrama de Casos de Uso para as atividades (i), (ii) e (iii) citadas acima.

Tabela 6 - Resultados da avaliação para o BurguerDigital

Número de relacionamentos gerados manualmente | {*linksrelevantes*} |	57
Número de relacionamentos gerados pela ferramenta | {*linksrecuperados*} |	45
Número de elementos inconsistentes	3
Número de relacionamentos incorretos	0

Precision	100%
Recall	78,9%

Como pode-se observar na Tabela 6, esse experimento resultou em 100% de *precision* e 78,9% de *recall*. Esses resultados mostram uma grande precisão (*precision*) da ferramenta, tendo realizado todos os quarenta e cinco relacionamentos realizados corretamente. O *recall* ficou dentro do aceitável, porem os dados apontam que existem cerca de 21% de relacionamentos existentes deixaram de ser identificados pela ferramenta protótipo.

6.3 Projeto Pesquisa e Movimento

6.3.1 Visão geral

O Projeto Pesquisa e Movimento (Figura 26) foi desenvolvido, como protótipo por estudantes do curso de Mestrado em Informática Aplicada da Universidade Federal Rural de Pernambuco, com o intuito de facilitar a solicitação de reservas de veículos por pesquisadores da instituição. O Projeto Pesquisa e Movimento dispõe de métodos de gerenciamento de solicitações de veículos e auxilia no processo de validação dos projetos de pesquisa.

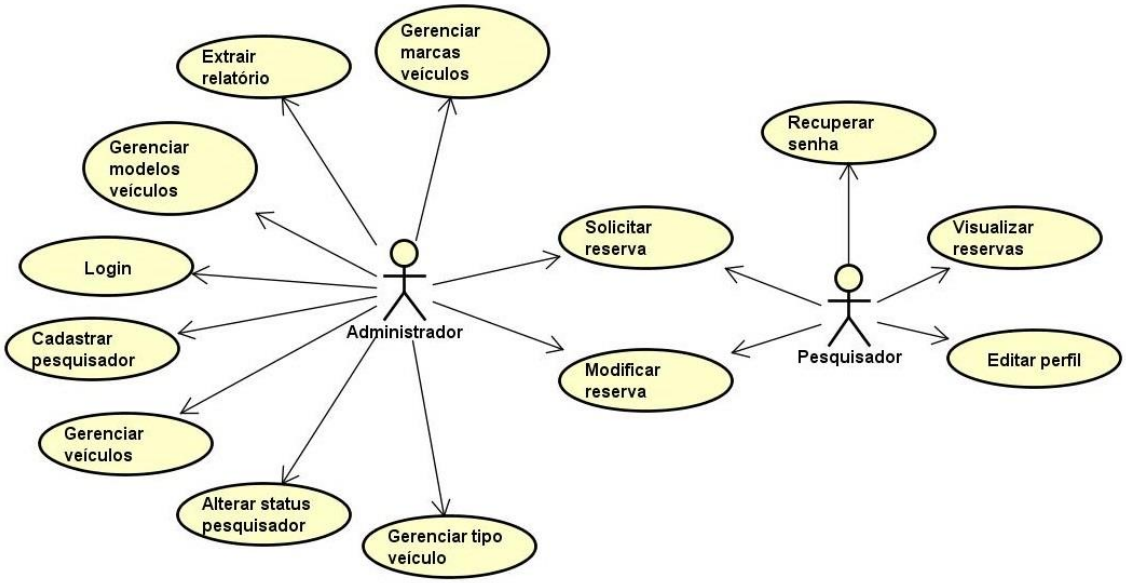

Figura 26 - Diagrama de Caso de Uso do Projeto Pesquisa e Movimento

O Projeto Pesquisa e Movimento permite que o Pesquisador e o Administrador acessem ao sistema (*Login*) por meio de credencias (usuário e senha).

O Administrador poderá gerenciar os tipos de veículos (Gerenciar tipo veículo) do sistema, assim é possível, por exemplo, cadastrar um tipo de veículo, como passeio ou de carga. O Administrador gerenciará as marcas (Gerenciar marcas veículos) e modelos de veículos (Gerenciar modelos veículos) existentes no sistema. O Administrador poderá ainda realizar o gerenciamento dos veículos existentes (Gerenciar veículos). Essas atividades de gerenciamento permitem ao Administrador cadastrar, excluir ou editar.

O acesso do Pesquisador será controlado pelos Administrador, assim é possível o Administrador poderá cadastrar um pesquisador (Cadastrar pesquisador), ou ainda desativar ou ativar o acesso (Alterar status pesquisador) desse tipo de usuário.

O Administrador poderá solicitações (Solicitar reserva) ou modificar reservas de veículos. Assim, os administradores poderão modificar informações gerais da solicitação ou modificar seu status para aprovado, rejeitado, cancelado, em uso, pendente ou finalizado.

O Administrador ainda terá a possibilidade de gerar relatórios (Extrair relatório) sobre as solicitações de veículos.

Em caso de perda da senha, o Pesquisador poderá solicitar (Recuperar senha) o envio de um link por e-mail para que possam modificá-la.

O Pesquisador poderá modificar sua senha ou seus dados pessoais (Editar perfil) a qualquer momento, desde que estejam com uma sessão ativa no sistema. O Pesquisador poderá solicitar a reserva de um veículo (Solicitar reserva), para isso informa o período de viagem, o motivo, o tipo de veículo desejado e se deseja ou não um motorista da Universidade. Após esse processo, o pesquisador precisará aguardar uma alteração na situação do pedido, esta será fornecida após a análise de um Administrador.

O Pesquisador poderá listar todas as solicitações de veículos realizadas por ele (Visualizar reservas).

A qualquer momento que anteceda a viagem, o Pesquisador poderá cancelar a solicitação (Modificar reserva), para isso precisa informar o motivo da desistência.

6.3.2 Artefatos

O documento de Especificação de Casos de Uso, desenvolvido por estudantes de Mestrado da Universidade Federal Rural de Pernambuco, foi reescrito na ferramenta iTraceWeb, contemplando o *template* descrito no Capítulo 2.

A Tabela 7 mostra o tipo e o número de elementos existente no documento de Especificação de Casos de Uso do Projeto Pesquisa e Movimento.

Tabela 7 - Elementos do documento de Especificação de Casos de Uso do Projeto Pesquisa e Movimento

Tipo de elemento	Número de elementos
Ator	2
Requisito funcional	18
Fluxos (principal, secundário e exceção)	37

O Diagrama de Casos de Uso já havia sido desenvolvido no Astah pelos estudantes que o criaram, sendo necessário apenas extraí-lo para XML.

A Tabela 8 mostra o número e o tipo de elemento existente no Diagrama de Casos de Uso do Projeto Pesquisa e Movimento.

Tabela 8 - Elementos do Diagrama de Casos de Uso do Projeto Pesquisa e Movimento

Tipo de elemento	Número de elementos
Ator	2
Caso de Uso	18

6.3.3 Avaliação

O exemplo de aplicação Projeto Pesquisa e Movimento foi utilizado para avaliar esta pesquisa, a fim de:

- Medir o *precision* e *recall* da ferramenta protótipo;
- Identificar inconsistências;
- Demonstrar como o processo de identificação de inconsistências pode apoiar o desenvolvimento do sistema, aumentando o recall.

Para analisar a eficácia da ferramenta protótipo, foram realizadas as seguintes atividades: (i) identificação manual das relações de rastreabilidade e inconsistentes existentes entre os documentos de Especificação de Casos de Uso e do Diagrama de Casos de Uso; (ii) processamento das regras de rastreabilidade pela ferramenta protótipo, a fim de identificar automaticamente as relações de rastreabilidade e inconsistências existentes entre os modelos; (iii) analisar a eficiência dos resultados, através das métricas *precision* e *recall*, obtidos a partir da comparação de (i) e (ii); (iv) identificação automatizada pela ferramenta protótipo de

inconsistências; (v) atualização dos modelos com base no resultado obtido em (iv); e (vi) nova análise da eficiência, através das métricas *precision* e *recall*, obtidos sobre o modelo ajustado em (v).

A avaliação foi realizada com onze regras de rastreabilidade para identificar relações de rastreabilidade entre os modelos de Especificação de Casos de Uso e Diagrama de Casos de Uso, e cinco regras para identificar inconsistências entre os mesmos modelos.

A Tabela 9 mostra os resultados obtidos do relacionamento entre os modelos de Especificação de Casos de Uso e Diagrama de Casos de Uso para as atividades (i), (ii) e (iii) citadas acima.

Tabela 9 - Resultados da avaliação para o Projeto Pesquisa e Movimento

Número de relacionamentos gerados manualmente | {*linksrelevantes*} |	83
Número de relacionamentos gerados pela ferramenta | {*linksrecuperados*} |	78
Número de inconsistentes	4
Número de relacionamentos incorretos	8
Precision	89,7%
Recall	83,4%

Como observado na Tabela 9 o experimento identificou 89,7% de *precision* e 83,4% de *recall*. Esses resultados mostram um bom desempenho da ferramenta tanto em *precision* como em *recall*. Neste exemplo de aplicação, alguns elementos foram relacionados incorretamente, oito ao total, por isso o *precision* teve um valor menor, se comparado ao *precision* obtido na avaliação do BurguerDigital. O *recall* por sua vez, manteve-se similar ao obtido no BurguerDigital, ou seja, a ferramenta protótipo deixou de identificar aproximadamente 17% de relações de rastreabilidade existentes.

A Tabela 10 mostra os resultados obtidos após a atualização dos documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso, com base nos resultados obtidos da análise de elementos inconsistentes.

Tabela 10 - Resultados da avaliação para o Projeto Pesquisa e Movimento após as correções dos modelos com base no resultado dos elementos inconsistentes

Número de relacionamentos gerados manualmente | {*linksrelevantes*} |	83
Número de relacionamentos gerados pela ferramenta | {*linksrecuperados*} |	83
Número de inconsistentes	0
Número de relacionamentos incorretos	0

Precision	100%
Recall	100%

Como mostrado na Tabela 10, o experimento resultou em 100% de *precision* e 100% de *recall*, demonstrando assim um aumento nos índices de *precision* e *recall* exibidos anteriormente na Tabela 9. Isso mostra que identificação de inconsistências entre os artefatos contribuiu diretamente para que, após os ajustes desses artefatos, a ferramenta identificasse todos os relacionamentos existentes entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso (*recall*) e que nenhum relacionamento fosse identificado incorretamente pela ferramenta protótipo (*precision*).

6.4 Ameaças à validade da avaliação

As ameaças de validade desta abordagem estão preocupadas com o fato de que a mesma pessoa desenvolveu o modelo de referência, os modelos de análise (Especificação de Casos de Uso e Diagrama de Casos de Uso), as regras de rastreabilidade e as regras de inconsistências entre os modelos. A seguir é discutido quais os cuidados tomados para que isso não invalidasse esta pesquisa.

O modelo de referência foi criado no inicia da pesquisa, levando em consideração apenas tipos de relações de rastreabilidade existentes na literatura (Filho, 2011) e a estrutura semântica dos elementos dos documentos de Especificação de Casos de Uso e Diagrama de Caso de Uso. Desta forma, o modelo de referência não sofrei nenhum ajuste que visasse apenas a melhoria dos resultados da avaliação.

As regras de rastreabilidade e identificação de inconsistências foram criadas para atender aos diversos tipos de relações existentes no modelo de referência, levando em consideração a semântica de cada elemento. Além disso todas as regras de rastreabilidade e inconsistências foram criadas com o intuído identificar elementos independentemente dos artefatos em análise, com a única premissa de que esses tenham sidos criados nas ferramentas especificadas nesta pesquisa (Astah e iTraceWeb).

Embora a análise dos resultados tenha sido realizada pela mesma pessoa que criou as regras de rastreabilidade e análise de consistência, os dados gerados foram obtidos de uma geração automatizada pela ferramenta protótipo e comparados com dados gerados manualmente, antes mesmo da execução da ferramenta, diminuindo assim as chances de

interferência nos resultados. Além disso, para não influenciar nos resultados, as regras criadas não sofreram mudanças durante o processo de avaliação do trabalho.

Outra questão relevante é em relação a escala de validade desta pesquisa. Como já mencionado anteriormente, foram criadas regras a serem aplicadas em qualquer projeto, independente de complexidade ou tamanho. Foram avaliados dois exemplos de aplicação onde os resultados obtidos foram similares independente de sua complexidade e dimensão.

6.5 Considerações finais

Este capítulo apresentou o resultado da avaliação desta pesquisa utilizando dois exemplos de aplicação: BurguerDigital e Projeto Pesquisa e Movimento. Foram descritos os critérios de avaliação utilizadas para demonstrar esta abordagem. Para cada exemplo de aplicação, foi realizado uma descrição da visão geral, uma apresentação dos artefatos utilizados e detalhados os resultados obtidos na avaliação. Por fim, foram discutidos os resultados obtidos.

7 Conclusão e trabalhos futuros

Este capítulo apresenta as conclusões desta pesquisa e propõe trabalhos futuros. A seção 6.1 revisa essa dissertação de modo geral. A seção 6.2 aponta como as hipóteses foram alcançadas. Na seção 6.3 são abordados os objetivos desta pesquisa. Na seção 6.4 são descritas a contribuições obtidas com esta dissertação. A seção 6.5 aponta possíveis linhas de estudo para o futuro. Por fim, na seção 6.6 são feitas as considerações finais.

7.1 Análise geral

Esta dissertação apresentou uma abordagem para identificar automaticamente relações de rastreabilidade e inconsistências entre elementos de artefato comumente criados em projetos do domínio Orientados a Objeto. Em particular, esta pesquisa focou em rastrear elementos do documento de Especificação de Casos de Uso com elementos do Diagrama de Casos de Uso UML.

Após o Capítulo 1 realizar uma introdução da pesquisa realizada, o Capítulo 2, exibiu uma visão geral da área da pesquisa. Foi discutido o que é rastreabilidade de software. Foram apresentados os principais tipos de abordagens para a captura de relações de rastreabilidade (semi) automaticamente. Também foram apresentados os artefatos Especificação de Casos de Uso e UML, este último com ênfase no Diagrama de Casos, documentos esses que são alvo da pesquisa. Por fim, são revisadas algumas técnicas para visualizar as relações de rastreabilidade.

No Capítulo 3 foi feito um levantamento de trabalhos relacionados. Foram identificadas e discutidas outras abordagens de rastreabilidade. Por fim, foi apresenta uma tabela comparativa entre as abordagens discutidas.

No Capítulo 4, foi descrito o modelo de referência de rastreabilidade utilizado em nesta abordagem. Foram descritos os tipos de relações de rastreabilidade existentes entre os modelos, sempre exibindo um exemplo para cada um dos tipos de relações. Esta pesquisa propôs quatro tipos de relações de rastreabilidade entre o documento de Especificação de Casos de Uso e Diagrama de Casos de Uso UML: sobreposição, composição, utilização e dependência.

No Capítulo 5, foi apresentado o *framework* de rastreabilidade baseado em regras para apoiar: (a) a geração automatizada das relações de rastreabilidade propostas no Capítulo 4; e

(b) identificação de inconsistências. Esta abordagem se baseou em regras e utilizou o XQuery como linguagem de consulta e processamento das regras de rastreabilidade e verificação de consistências. Nesse capítulo foi exibida uma visão geral do *framework* e explorada a estrutura que compõe a regra. Foram abordadas as funções Java que estendem o XQuery para auxiliar no processamento das regras. Por fim, a ferramenta protótipo, criada para validar esta abordagem, é descrita.

Por fim, no Capítulo 6, foi avaliado o *framework* de rastreabilidade proposto, por meio de dois exemplos de aplicação, o BurguerDigital e o Projeto Pesquisa e Movimento. Foram avaliados os resultados da identificação das relações de rastreabilidade por meio da análise das métricas *precision* e *recall*. Foi avaliado também como a identificação de inconsistências pode auxiliar na redução de divergências entre os modelos.

A Tabela 11 exibe um resumo dos resultados da análise de *precision* e *recall* para os exemplos de aplicação. Como pode-se observar, o *precision* variou entre 89,7% e 100%, enquanto que o *recall* oscilou entre 78,9% e 100%.

Tabela 11 - Resumo dos resultados de precision e recall para os exemplos de aplicação

	BurguerDigital	Projeto Pesquisa e Movimento (1)	Projeto Pesquisa e Movimento (2)
Precision	100%	89,7%	100%
Recall	78,9%	83,4%	100%

Os resultados das medições de *precision* e *recall* obtidos da avaliação são comparáveis em termos de valores com medições obtidas em outras abordagens (FILHO, 2011), (ALI; GUENEUC; ANTONIOL, 2013), (LUCIA *et al*., 2011). Contudo, é importante notar que esses estudos abordam diferentes domínios, artefatos de análise e exemplos de aplicação.

A fim de provar a hipótese apresentada no Capítulo 1, que afirma ser possível utilizar uma abordagem orientada a regras para identificar automaticamente relações de rastreabilidade entre elementos da Especificação de Casos de Uso e do Diagrama de Casos de Uso UML, além de identificar inconsistências entre estes artefatos, esta pesquisa tomou uma série de ações. A seguir essas ações são revisadas.

- Gerar automaticamente relações de rastreabilidade entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso.

Foi desenvolvido um modelo de rastreabilidade que define os tipos de relacionamentos existentes entre os elementos da Especificação de Casos de Uso e Diagramas de Caso de Uso UML. Baseado nesse modelo de rastreabilidade foi apresentado *framework* baseado em regras capaz de identificar automaticamente as relações de rastreabilidade entre elementos da especificação de Casos de Uso e Diagrama de Casos de Uso. Foi desenvolvida ainda uma ferramenta protótipo, essa implementou a abordagem desta pesquisa e apoiou a avaliação da mesma através de comparações entre os resultados por ela obtidos e os resultados gerados manualmente (livre de erros), onde foram analisadas as métricas *precision* e *recall*. Verificou-se assim a eficiência desta abordagem. A avaliação obteve bons resultados, *precision* (89,7% - 100%) e *recall* (78,9% - 100%).

- Identificar automaticamente relações de rastreabilidade e inconsistências entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso.

Foram criadas várias regras capaz de identificar relações de rastreabilidade e inconsistências entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso. O exemplo de aplicação BurguerDigital foi utilizado para validar as regras criadas, sendo estas mesmas regras utilizadas posteriormente em outro exemplo de aplicação, o Projeto Pesquisa e Movimento.

- Utilização de informações de inconsistências para corrigir a integridade entre os elementos da Especificação de Casos de Uso e do Diagrama de Casos de Uso, melhorando assim a consistência entre eles.

A ferramenta protótipo é capaz de identificar inconsistências e exibir em uma matriz a lista dos elementos divergentes, assim fica fácil para o usuário identificar quais elementos precisam ser modificados, melhorando assim a consistência entre os modelos.

Os resultados obtidos na análise de consistência entre os modelos, do exemplo de aplicação Projeto Pesquisa e Movimento, foram utilizados para gerar novos modelos, porém mais íntegros. Posteriormente, esses novos modelos foram submetidos a nova análise. Os resultados foram consideráveis, 83,4% de *precision* e 100% de *recall*.

As ações descritas acima não só provam a hipótese como também demonstram que todos os objetivos citados no Capítulo 1 também foram alcançados nesta pesquisa.

7.2 Contribuições

O trabalho apresentado nessa dissertação contribui para de modo geral para projetos Orientados a Objetos, ou outros, que utilizem em sua documentação os modelos de Especificação de Casos de Uso e Diagrama de Casos de Uso.

Esta pesquisa fornece um modelo de referência que viabiliza a identificação de relações de rastreabilidade e inconsistências entro os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso.

Assim como abordado em Ramesh e Jarke (2001), a definição de um modelo de referência que contenha regras semânticas contribuem em análises mais precisas sobre as relações de rastreabilidade e auxiliam atividades como manutenção, análise de impacto e reuso.

Foi desenvolvida uma abordagem orientada a regras para automatizar as relações de rastreabilidade e identificação de inconsistências entre os modelos de análise.

A pesquisa desenvolvida exibe o grau de confiabilidade dos relacionamentos criados (degreeOfCompleteness), com isso este trabalho facilita a validação dos relacionamentos identificados.

Este trabalho contribui ainda com a verificação de consistência dos artefatos, por meio de regras capaz de identificar elementos inconsistências ou em falta entre os modelos, contribuindo assim para a construção e manutenção dos artefatos.

7.3 Trabalhos futuros

Esta dissertação possibilita a extensão de novas pesquisas. A seguir são detalhados possíveis trabalhos que precisam ser realizados para melhorar esta abordagem.

7.3.1 Suporte a outros diagramas UML

Esta pesquisa focou em avaliar apenas o diagrama de casos de uso, contudo acredita-se que é importante também modelos de referência e regras para abordar outros diagramas importantes da UML, como é o caso do Diagrama de Sequência e Diagrama de Atividades.

7.3.2 Interpretador de documento de Especificação de Casos de Uso

Esta pesquisa limita ao usuário a criação do documento de Especificação de Casos de Uso por uma ferramenta, desenvolvida como parte desta pesquisa, o iTraceWeb. Embora essa ferramenta esteja disponível na internet e acessível a qualquer momento, pode ser importante deixar o usuário livre para criar esse artefato em outras ferramentas de edição de texto, como as ferramentas proprietárias Microsoft Word e LibreOffice. Para isso, é necessário criar um interpretador que transforme os artefatos gerados por essas ferramentas para o mesmo padrão XML gerado pelo iTraceWeb.

7.3.3 Ferramenta de visualização

Esta pesquisa focou apenas na identificação de relacionamentos de rastreabilidade e identificação de elementos divergentes e, embora a ferramenta protótipo possa gerar uma matriz de rastreabilidade dos resultados, não houve, nesse momento, preocupação em analisar a melhor forma de exibir esses dados. Uma possibilidade é integrar os resultados desta pesquisa a outros estudos que estejam em andamento, como o apresentado em Filho, Lencastre e Rodrigues (2013), que propõe uma aplicação capaz de fornecer diversas técnicas de visualização para que o usuário possa escolher a melhor visualização de acordo com as suas necessidades.

7.3.4 Editor gráfico de regras

Esta abordagem é baseada em regras escritas na linguagem XQuery, contudo a criação dessas regras exige conhecimento prévio dessa tecnologia e pode ser uma atividade complexa. Assim, acredita-se que uma ferramenta gráfica capaz de criar e editar regras visualmente, por meio do processo arrastar e soltar (*drag-and-drop*), poderá facilitar a criação o processo de criação de regras. Atualmente existe uma pesquisa realizada por Silva (2014) que tenta sanar essa lacuna.

7.4 Considerações finais

Durante a construção de projetos orientados a objetos são criados alguns documentos e em muitos casos o relacionamento de elementos contidos nesses artefatos é processo complexo e difícil de ser realizado manualmente. Esta dissertação apresentou uma abordagem para apoiar a geração automatizada de relacionamento de rastreabilidade e identificação inconsistências entre os documentos de Especificação de Casos de Uso e Diagrama de Casos de Uso UML.

Referências

ALI, N.; GUEHENEUC, Y.; ANTONIOL, G. Trust-based requirements traceability. In: Program Comprehension (ICPC), 2011 IEEE 19th International Conference on. [S.l.: s.n.], 2011. p. 111–120. ISSN 1092-8138. Citado na página 51.

ALI, N.; GUENEUC, Y.; ANTONIOL, G. Trustrace: Mining software repositories to improve the accuracy of requirement traceability links. Software Engineering, IEEE Transactions on, v. 39, n. 5, p. 725–741, May 2013. ISSN 0098-5589

ANQUAN, Jie et al. The education reform and innovation of object-oriented programming course in normal university. In: IEEE. Computer Science and Education (ICCSE), 2010 5th International Conference on. [S.l.], 2010.

ARANTES, Lucas de Oliveira. Documentação Semântica no Apoio à Integração de Dados e Rastreabilidade. 2010. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, Avenida Fernando Ferrari, 514, Centro Tecnológico, Vitória, Espírito Santo - Brasil, 2010.

ASTAH. Astah. 2015. <http://astah.net/>. Acessado: 27-07-2015.

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier et al. Modern information retrieval. [S.l.]: ACM press New York, 1999.

BUCHGEHER, G.; WEINREICH, R. Automatic tracing of decisions to architecture and implementation. In: Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on. [S.l.: s.n.], 2011. p. 46–55;

CERRI, ELISA CERRI. Um modelo de rastreabilidade entre o docuemnto de especificação de requisitos e o modelo de casos de uso do sistema. mar 2007. dissertacao — Universidade Católica do Rio Grande do Sul, mar 2007.

CHUNG, Lawrence; LEITE, JulioCesarSampaio do Prado. On non-functional requirements in software engineering. In: BORGIDA, AlexanderT. et al. (Ed.). Conceptual Modeling: Foundations and Applications. Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5600). p. 363–379. ISBN 978-3-642- 02462-7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-02463-4_19>.

CLELAND-HUANG, Jane et al. Software traceability: Trends and future directions. In: Proceedings of the on Future of Software Engineering. New York, NY, USA: ACM, 2014. (FOSE 2014), p. 55–69. ISBN 978-1-4503-2865-4.

COCKBURN, Alistair. Basic use case template. Humans and Technology, Technical Report, v. 96, 1998.

COCKBURN, Alistair. Writing effective use cases. preparation for Addison-Wesley Longman. www. infor. uva. es/˜ mlaguna/is2/materiales/BookDraft1. pdf, 1999.

COEST. Center of excellence for software traceability. 2015. <http://www.CoEST.org>. Acessado: 07-06-2015.

CSMR 2008. 12th European Conference on. [S.l.: s.n.], 2008. p. 302–305. ISSN 1534-5351.

DAGENAIS, B.; ROBILLARD, M.P. Recovering traceability links between an api and its learning resources. In: Software Engineering (ICSE), 2012 34th International Conference on. [S.l.: s.n.], 2012. p. 47–57.

DALL'OGLIO, PD; SILVA, JP; PINTO, SCCS. Um Modelo de Rastreabilidade comsuporte ao Gerenciamento de Mudanças e Análise de Impacto. [S.l.]: WER, 2010

DAVIS, Alan M. The analysis and specification of systems and software requirements. Systems and Software Requirements Engineering, IEEE Computer Society Press-Tutorial, p. 119–134, 1990.

DUTOIT, Allen H; PAECH, Barbara. Rationale-based use case specification. Requirements engineering, Springer, v. 7, n. 1, p. 3–19, 2002.

FILHO, Gilberto Amado de Azevedo Cysneiros. Software Traceability for Multi-Agent Systems Implemented Using BDI Architecture. may 2011. phdthesis — City University London, may 2011

FILHO, Gilberto Cysneiros; LENCASTRE, Maria; RODRIGUES, Adriana. Retratos: Requirement traceability tool support. Citeseer, 2013.

FILHO, Gilberto Cysneiros; ZISMAN, Andrea. Traceability and completeness checking for agent-oriented systems. In: ACM. Proceedings of the 2008 ACM symposium on Applied computing. [S.l.], 2008. p. 71–77

FOCUS, Micro. CaliberRM - Data Sheet. 2015. <https://www.microfocus.com/downloads/caliberrm-6868.aspx>. Acessado: 16-07-2015.

GLINZ, M. On non-functional requirements. In: Requirements Engineering Conference, 2007. RE '07. 15th IEEE International. [S.l.: s.n.], 2007. p. 21–26. ISSN 1090-705X.

GOTEL, O.C.Z.; FINKELSTEIN, A.C.W. An analysis of the requirements traceability problem. In: Requirements Engineering, 1994., Proceedings of the First International Conference on. [S.l.: s.n.], 1994. p. 94–101.

GOTEL, O. et al. The quest for ubiquity: A roadmap for software and systems traceability research. In: Requirements Engineering Conference (RE), 2012 20th IEEE International.

GOTEL, Orlena et al. The grand challenge of traceability (v1. 0). In: Software and Systems Traceability. [S.l.]: Springer, 2012. p. 343–409.

HEIM, Philipp; ZIEGLER, Jürgen; LOHMANN, Steffen. gfacet: A browser for the web of data. In: CITESEER. Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08). [S.l.], 2008. v. 417, p. 49–58

IBM. Rational DOORS. 2015. <http://www-03.ibm.com/software/products/pt/ratidoor>. Acessado: 16-07-2015.

IEEE, Software and Applications, 2008. COMPSAC ’08. 32nd Annual IEEE International. S.l.: s.n.], 2008. p. 247–254. ISSN 0730-3157.

ITRACEWEB. iTrace - Traceability Software. 2015. <http://jeffersonpenna.com/itrace/>. Acessado: 27-07-2015.

JACOBSON, Ivar; SPENCE, Ian; BITTNER, Kurt. Use case 2.0: The guide to succeeding with use cases. Ivar Jacobson International, p. 1–55, 2011

KANNENBERG, Andrew; SAIEDIAN, Hossein. Why software requirements traceability remains a challenge. CrossTalk The Journal of Defense Software Engineering, v. 22, n. 5, p. 14–19, 2009.

KASSAB, M.; Ormanjieva, O.; Daneva, M. A traceability metamodel for change management of non-functional requirements. In: Dosch, W. et al. (Ed.). Proceedings of the 6th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2008. Los Alamitos: IEEE Computer Society Press, 2008. p. 245–254. Disponível em: <http://doc.utwente.nl/65197/>.

LARMAN, Craig. Utilizando UML e padrões. [S.l.]: Bookman, 2002.

LEAL, Marcelio D’Oliveira; FIGUEIREDO, Mayara; SOUZA, Cleidson Ronald Botelho de. Uma abordagem semi-automática para a manutenção de links de rastreabilidade. In: WER. [S.l.: s.n.], 2008

LINDVALL, Mikael; SANDAHL, Kristian. Practical implications of traceability. Softw., Pract. Exper., Citeseer, v. 26, n. 10, p. 1161–1180, 1996.

LIU, Dapeng et al. Feature location via information retrieval based filtering of a single scenario execution trace. In: Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering. New York, NY, USA: ACM, 2007. (ASE ’07), p. 234–243. ISBN 978-1-59593-882-4. Disponível em: <http://doi.acm.org/10.1145/1321631.1321667>.

LUCIA, Andrea De; OLIVETO, Rocco; TORTORA, Genoveffa. Adams re-trace: Traceability link recovery via latent semantic indexing. In: Proceedings of the 30th International Conference on Software Engineering. New York, NY, USA: ACM, 2008. (ICSE ’08), p. 839–842. ISBN 978-1-60558-079-1. Disponível em: <http://doi.acm.org/10.1145/1368088.1368216>

LUCIA, Andrea De; PENTA, Massimiliano Di; OLIVETO, Rocco. Improving source code lexicon via traceability and information retrieval. Software Engineering, IEEE Transactions on, IEEE, v. 37, n. 2, p. 205–227, 2011

MARCUS, Andrian; MALETIC, Jonathan I; SERGEYEV, Andrey. Recovery of traceability links between software documentation and source code. International Journal of Software Engineering and Knowledge Engineering, World Scientific, v. 15, n. 05, p. 811–836, 2005

MARTINS, José Carlos Cordeiro. Gerenciando Projetos de Desenvolvimento de Software com PMI, RUP e UML. [S.l.]: Brasport, 2010

MELLOR, Stephen J; CLARK, Tony; FUTAGAMI, Takao. Model-driven development:guest editors' introduction. IEEE software, IEEE Computer Society, v. 20, n. 5, p. 14–18, 2003.

MERTEN, Thorsten; JUPPNER, D.; DELATER, A. Improved representation of traceability links in requirements engineering knowledge using sunburst and netmap visualizations. In: Managing Requirements Knowledge (MARK), 2011 Fourth International Workshop on. [S.l.: s.n.], 2011. p. 17–21.

OLIVETO, R. Traceability management meets information retrieval methods -strengths and limitations. In: Software Maintenance and Reengineering, 2008.

OMG. Unified Modeling Language (UML). 2015. <http://www.omg.org/spec/UML/>. Acessado: 18-07-2015

PINHEIRO, Francisco AC; GOGUEN, Joseph A. An object-oriented tool for tracing requirements. In: IEEE. Requirements Engineering, 1996., Proceedings of the Second International Conference on. [S.l.], 1996. p. 219.

POHL, Klaus. Process-Centered Requirements Engineering. New York, NY, USA: John Wiley & Sons, Inc., 1996. ISBN 0863801935.

PRESSMAN, Roger S. Engenharia de software: Uma Abordagem Profissional. [S.l.]: McGraw Hill Brasil, 2011.

RAMESH, Bala; JARKE, Matthias. Toward reference models for requirements traceability. IEEE Transactions on Software Engineering, v. 27, p. 58–93, 2001.

SAIEDIAN, Hossein; KANNENBERG, Andrew; MOROZOV, Serhiy. A streamlined, cost-effective database approach to manage requirements traceability. Software Quality Journal, Springer US, v. 21, n. 1, p. 23–38, 2013. ISSN 0963-9314. Disponível em: <http://dx.doi.org/10.1007/s11219-011-9166-3>.

SAYÃO, Miriam; LEITE, Julio Cesar Sampaio do Prado. Rastreabilidade de requisitos. RITA, v. 13, n. 1, p. 57–86, 2006

SHERBA, Susanne A. Towards Automating Traceability: An Incremental and Scalable Approach. 2005. Tese (Doutorado), Boulder, CO, USA, 2005. AAI3178340.

SOMMERVILLE, Ian. Engenharia de Software. 8. ed. [S.l.]: Addison Wesley, 2007. ISBN 8588639289.

SPANOUDAKIS, George; ZISMAN, Andrea. Software traceability: A roadmap. In: Handbook of Software Engineering and Knowledge Engineering. [S.l.]: World Scientific Publishing, 2004. p. 395–428.

TACLA, Cesar Augusto. Análise e projeto OO & UML 2.0. <http://www.dainf.ct.utfpr.edu.br/~tacla/UML/Apostila.pdf>, 2013.

WINKLER, Stefan; PILGRIM, Jens. A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model., Springer-Verlag New York, Inc., Secaucus,

NJ, USA, v. 9, n. 4, p. 529–565, set. 2010. ISSN 161 -1366. Disponível em: <http://dx.doi.org/10.1007/s10270-009-0145-0>.

WORDNET. WordNet: A lexical database for English. 2015. <https://wordnet.princeton.edu/>. Acessado: 27-07-2015.

XML. XML Techonology. 2015. <http://www.w3.org/standards/xml>. Acessado: 07-06-2015.

XQUERY. XQuery: An XML Query Language. 2015. <http://www.w3.org/TR/xquery/>. Acessado: 07-06-2015.