UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA DSL PARA A GERÊNCIA DE CONFIGURAÇÃO DE UM SISTEMA DE GERENCIAMENTO INTEGRADO DE REDES

ROSANGELA PIERONI

ORIENTADORA: PROFA. DRA. ROSÂNGELA APARECIDA DELLOSSO PENTEADO

São Carlos - SP
Setembro/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA DSL PARA A GERÊNCIA DE CONFIGURAÇÃO DE UM SISTEMA DE GERENCIAMENTO INTEGRADO DE REDES

ROSANGELA PIERONI

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: engenharia de software

Orientadora: Dra. Rosângela Aparecida Dellosso Penteado

São Carlos - SP
Setembro/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

P619dd

Pieroni, Rosangela.
Desenvolvimento de uma DSL para a gerência de configuração de um sistema de gerenciamento integrado de redes / Rosangela Pieroni. -- São Carlos : UFSCar, 2015.
175 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Engenharia de software. 2. Desenvolvimento orientado por modelos. 3. Linguagem específica de domínio. 4. Gerenciamento integrado de redes. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

"Desenvolvimento de Uma DSL para a Gerência de Configuração de Um Sistema de Gerenciamento Integrado de Redes"

Rosangela Pieroni

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Profa. Dra. Rosângela Dellosso Penteado
(Orientadora - DC/UFSCar)

Prof. Dr. Hélio Crestana Guardia
(DC/UFSCar)

Prof. Dr. Heitor Augustus Xavier Costa
(UFLA)

São Carlos
Outubro/2014

Dedico esse trabalho para minha família, amigos e professores.

AGRADECIMENTO

Agradeço a Deus pela minha vida.

Eternamente grata a oportunidade de voltar, depois de quase vinte anos, a esta Universidade.

Agradeço à professora Rosangela pela paciência, dedicação, compreensão e confiança em mim depositada.

Agradeço à minha família pelo apoio.

Agradeço aos colegas do laboratório, principalmente ao Matheus Carvalho Viana.

Agradeço ao meu gerente Mario Massato Harada pelo apoio e compreensão e aos colegas de trabalho pelo pronto atendimento em participar do experimento e principalmente ao José Luis Schifferli Lopes, ao Alex Françoso de Morais Junior e ao Marco Antonio Miquelino pela paciência e colaboração.

“A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.”

Albert Einstein

RESUMO

Um sistema de gerenciamento integrado de redes tem o objetivo de gerenciar uma rede de telecomunicações, independentemente da tecnologia dos elementos de rede, a fim de identificar as causas de problemas, proporcionando a tomada de decisão mais assertiva para saná-los e deixar a rede disponível e operacional. Dessa forma, o mercado que está cada vez mais exigente com o nível de qualidade dos produtos e serviços de telecomunicações é atendido. Diante dessa necessidade, o sistema de gerenciamento integrado de redes de uma empresa real precisa atender rapidamente às expectativas dos clientes com relação às solicitações de novas funções do sistema, assim como realizar atualizações tecnológicas periodicamente. Porém, o desenvolvimento do sistema de gerenciamento integrado de redes dessa empresa utiliza tecnologias centradas em código-fonte, o que implica em grande esforço e dificulta a sua reutilização. Neste contexto, este projeto de mestrado apresenta uma proposta para facilitar o desenvolvimento do sistema de gerenciamento integrado de redes utilizando as técnicas de MDD (Desenvolvimento Dirigido a Modelos). MDD enfatiza a utilização de modelos no desenvolvimento do software. Esses modelos especificam as regras de negócio de domínio em um alto nível de abstração, ou seja, independentemente da linguagem de programação e da plataforma de execução do software. Com ferramentas computacionais adequadas são realizadas as transformações dos modelos em código-fonte. A utilização de MDD pode proporcionar reúso de software de forma processual, possibilitar um desenvolvimento mais rápido, com menor custo, produzir um software flexível e possibilitar modificações mais rapidamente. Para aplicar a abordagem de MDD, uma linguagem específica de domínio (DSL) para a gerência de configuração do sistema de gerenciamento integrado de redes foi desenvolvida. Tem como objetivo facilitar a inclusão de uma nova tecnologia a ser gerenciada pelo sistema e ao mesmo tempo proporcionar um desenvolvimento mais rápido, com menos erros de código. Para avaliar a proposta foi realizado um experimento com a participação dos atuais desenvolvedores de software de uma empresa real. Os resultados obtidos mostraram que, segundo a análise descritiva, houve redução do tempo gasto no desenvolvimento das aplicações quando foi utilizada a DSL em comparação com o desenvolvimento baseado em especialização das classes. Porém, os testes das hipóteses não apresentaram tal redução. A inserção de erros no código ocorreu somente nas aplicações desenvolvidas utilizando especialização das classes. O uso da DSL proporcionou uma visão mais ampla da regra de negócio e não exigiu conhecimento da linguagem de programação, pois o código foi gerado automaticamente.

Palavras-chave: Desenvolvimento Dirigido a Modelos (MDD), Linguagens Específica de Domínio (DSL), Gerenciamento Integrado de Redes.

ABSTRACT

The integrated network management system aims to manage the telecommunications network, regardless of network elements technology in order to identify the causes of problems, providing a more assertive decision making to remedy these problems and make the network available and operational. Thus, can to cater to the market that is increasingly demanding with the level of quality telecommunications products and services. Given this need, integrated networks management system belongs to a real company needs to respond quickly to customers' expectations with respect to requests for new functions system, and perform technology upgrades periodically. However, the integrated network management system development use centered-source technologies, and it requires great effort and hinders reuse. In this context, this master's project presents a proposal to facilitate the integrated network management system development using the techniques of MDD (Model Driven Development). MDD emphasizes the use of models in software development. These models specify the rules of business domain at a high level of abstraction, i.e., regardless of the programming language and to run the software platform. With appropriate computational tools transformations of models in the source code are performed. The use of MDD can provide software reuse procedurally, enabling faster development, lower cost, produce a flexible software and enable faster modifications. To apply MDD approach, a Domain Specific Language (DSL) for the configuration management of integrated network management system setting is designed to facilitate the inclusion of a new technology to be managed by the system, while providing faster development and with fewer errors of code. To evaluate the proposal, an experiment was conducted with the participation of software developers that know the system and work in this real company. The results showed that according to the descriptive analyses there was a reduction in the time spent in application development when it was used DSL compared to the specialization of classes. However, hypotheses tests showed no such reduction. The insertion of errors in the code occurred only in applications developed using specialization classes. The use of DSL provided a broader view of the business rule and you do not need knowledge of the programming language, once the code was automatically generated.

Keywords: Model Driven Development (MDD), Domain Specific Languages (DSL), Integrated Network Management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3GPP - *3rd Generation Partnership Project*

ASN - *Abstract Syntax Notation*

ATL – *Atlas Transformation Language*

AuC - Authentication Center

BER – *Basic Enconding Rules*

CIM – *Computer Independent Model*

CMIP – *Common Information Protocol*

CMMI - Capability Maturity Model Integration

CPE – *Customer Premises Equipment* - equipamento de comunicação que está localizado nas instalações do cliente, que pertence a ele ou é alugado

DSL – *Domain Specific Language*

EIR - *Equipment Identity Register*

EMF – *Eclipse Modeling Framework*

EPC - *Evolved Packet Core*

E-UTRAN - *Evolved UMTS Terrestrial Radio Access Network*

FMP - *Feature Modeling Plug-in for Eclipse*

FMT - *Feature Modeling Tool*

FTTb - *Fiber-To-The-building* ou *Fiber-To-The-basement*

FTTc - *Fiber-To-the-cabinet* ou *Fiber-To-The-curb*

FTTh - *Fiber-To-The-home*

FTTn - *Fiber-To-The-node*

FTTp - *Fiber-To-The premises*

FTTx – *Fiber to the x*

GI – Gerência Integrada

GME – *Generic Modeling Environment*

GMT – *Generative Modeling Tools*

GPON - *Gigabit-capable Passive Optical Networks*

GSM - *Global System for Mobile*

HLR - *Home Location Register*

HSS - *Home Subscriber Server*

IDE – *Integrated Development Environment*

IHC – Interface Homem Computador

IOS – *Internetwork Operation System*

IP – *Internet Protocol*

JET – *Java Emitter Templates*

JMI - *Java Metadata Interface*

LPS – Linhas de Produto de Software

LTE - *Long-Term Evolution*

MAC - *Medium Accesss Control*

MDA - *Model-driven architecture*

MDD – *Model-Driven Development*

MIB- *Management Information Base*

MME - *Mobility Management Entity*

MOF - *Meta Object Facility*

MPLS - *Multiprotocol Label Switching*

MTL - *Model to Text Language*

M2C – *Model-To-Code*

M2M – *Model-To-Model*

M2T – *Model-To-Text*

NMS – *Network Management System*

OCL - *Object Constraint Language*

OID – *Object Identifier*

OLT – *Optical Line Terminal*

OMG - *Object Management Group*

ONU – *Optical Network Unit*

ONT – *Optical Network Terminal*

OO – Orientação a Objetos

OTAN – Organização do Tratado do Atlântico Norte

PCC - Política e Controle de Carga

PCRF - *Policy and Charging Resource Function*

PDH - *Plesiochronous Digital Hierarchy*

P-GW - *Packet Data Network Gateway*

PIM - *Plataform Independent Model*

PON - *Passive Optical Network*

PSM - *Plataform Specific Model*

PU – Processo Unificado

QoS - *Quality of Service*

QVT - *Query, Views and Transformations*

RAM - *Random-access memory*

RFC – *Request for Comments*

RLC - *Radio Link Control*

SDH - *Synchronous Digital Hierarchy*

S-GW - *Serving Gateway*

SHDSL - *Symmetric High-Speed Digital Subscriber Line*

SMI – *Structure of Management Information*

SNMP - *Simple Network Management Protocol*

TT – *Trouble Ticket*

UDP – *User Data Protocol*

UE - *User Equipment*

UML - *Unified Modeling Language*

UMTS - *Universal Mobile Telecommunications System*

WCDMA - *Wideband Code Division Multiple Access*

WDM - *Wavelength-Division Multiplexing*

XMI - *XML Metadata Interchange*

XML- *Extensible Markup Language*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contexto

A reutilização de software busca aumentar a qualidade e a produtividade no desenvolvimento de software, evitando a replicação de esforço e reaproveitando ao máximo possível as experiências adquiridas em projetos anteriores. Porém, os problemas inerentes ao desenvolvimento de software da maneira como é conduzido atualmente, baseado principalmente em código-fonte e sistemas de software cada vez mais complexos, dificultam a reutilização do software (LUCREDIO, 2009).

O software precisa sobreviver a um cenário de mudanças constantes, sejam mudanças de requisitos funcionais do sistema, de linguagens de programação, de ambientes de desenvolvimento, de ambientes computacionais de execução e de paradigmas de comunicação. Diante desse cenário, o desenvolvimento do software se torna cada vez mais complexo e a utilização de tecnologias centradas em código-fonte parece não ser o mais adequado (FRANCE; RUMPE, 2007).

O desenvolvimento orientado a modelos (*Model Driven Development* - MDD) surge como uma alternativa para ser aplicado em cenários que exigem mudanças constantes enfatizando a importância de modelos no processo de desenvolvimento de software (MELLOR; CLARK; FUTAGAMI, 2003). No MDD, os modelos são incorporados como parte integrante do produto final por meio de técnicas de modelagem e geração de código. A proposta é que os desenvolvedores de software utilizem modelos de alto nível de abstração para se protegerem das complexidades da plataforma de implementação (FRANCE; RUMPE, 2007). Com essa proposta, eles podem dedicar maior atenção ao domínio do problema (HUTCHINSON *et al*., 2011), pois parte da complexidade do

software fica escondida dentro dos geradores automáticos dos artefatos de implementação, que refletem a solução expressa nos modelos de alto nível de abstração (SCHMIDT, 2006).

O desenvolvimento de software orientado a modelos (MDD) pode: proporcionar reúso de software de forma mais processual, possibilitar um desenvolvimento mais rápido, com menor custo, produzir um software flexível e possibilitar realizar modificações mais rapidamente (CZARNECKI e ANTKIWICZ, 2005). Além disso, existem ferramentas que permitem a criação de linguagens específicas de domínio (DSL) gráficas para representar as regras de negócio do domínio, proporcionando melhor entendimento, melhor visualização do problema e melhor comunicação (LUCRÉDIO, 2010; DEURSEN *et al.*, 2000).

Com o crescimento em número e em diversidade de produtos e serviços na área de telecomunicações, o mercado passou a exigir maior nível de qualidade desses produtos e serviços. Gerenciar os vários elementos das redes de comunicação de maneira integrada se torna importante, pois possibilita, por exemplo, a identificação da causa dos problemas nessas redes. Esse gerenciamento permite que a tomada de decisão seja mais assertiva para sanar a causa dos problemas tornando a rede disponível e operacional.

Em uma empresa real, cujo nome será omitido por questão de confidencialidade, o gerenciamento integrado de redes refere-se àquele que é independente da tecnologia de rede que está sendo gerenciada, uma vez que as funções do sistema de gerenciamento são semelhantes para os diferentes recursos. Porém, as particularidades de cada uma das tecnologias de rede devem ser consideradas para a obtenção de informações de gerenciamento dos seus elementos. Para desenvolver este sistema de gerenciamento integrado de redes é necessário ter conhecimento específico das diversas tecnologias às quais pertencem os elementos de rede. Os potenciais clientes deste sistema de gerenciamento integrado de redes são os que já possuem uma rede de telecomunicações, composta por elementos de várias tecnologias, e que têm interesse de, justamente, fazer o gerenciamento integrado desses elementos. No processo de desenvolvimento deste sistema, quando elementos de uma nova tecnologia devem ser incorporados, é de responsabilidade do especialista de negócio, em conjunto com o analista de requisitos, fazer análise das informações dos novos elementos, e do arquiteto, analisar a arquitetura do sistema e o código para maximizar o reúso dos artefatos de

software. Porém, o reúso atualmente está concentrado no código-fonte, em alguns padrões de software e em componentes.

Considerando as características do sistema de gerenciamento integrado de redes, há indícios da importância e da necessidade em aplicar técnicas de reúso de software em um nível mais alto de abstração. A aplicação do desenvolvimento orientado a modelos propõe que as regras de negócio do domínio se concentrem nos modelos de mais alto nível e que ferramentas de transformação de modelo para código escondam dos desenvolvedores a complexidade da utilização das plataformas de desenvolvimento. Dessa forma, a utilização de MDD pode acelerar o desenvolvimento de sistemas de gerenciamento integrado de redes e, por conseqüência, seria possível atender mais rapidamente o mercado que está cada vez mais exigente com a qualidade de prestação de serviços de telecomunicações.

1.2 Motivação e Objetivos

A motivação e o objetivo para realizar este projeto de mestrado partem de dois pontos iniciais: um teórico que considera especificamente a abordagem MDD para a criação de uma DSL para reusar software; e um prático, que se trata de um sistema de gerenciamento integrado de redes que, por solicitação do mercado, precisa incorporar e monitorar novos elementos de redes, com novas e diferentes tecnologias, que são acrescentados às redes de telecomunicações.

O sistema de gerenciamento integrado de redes é um dos tipos de sistema que podem ser desenvolvidos utilizando o MDD para a criação de uma DSL. Isso ocorre, pois ele possui um conjunto de funções comuns para o gerenciamento dos elementos de rede, e requisitos que atendem às particularidades das tecnologias dos elementos de rede que estão sendo gerenciados.

O reúso de requisitos é uma atividade na engenharia de software que pode contribuir para a redução do tempo e do custo de desenvolvimento de software (CZARNECKI e ANTKIWICZ, 2005). Porém, a prática do reúso de requisitos não é uma tarefa simples, exige que a especificação dos requisitos tenha um alto nível de abstração para ser reutilizada (HARRISON *et al*., 2007). Neste contexto surge a necessidade da criação de um ambiente com aplicação de técnicas e de um ferramental que auxilie a

especificação dos requisitos de software de modo que possam ser reutilizados em outros projetos.

Para que seja possível fazer uma especificação de requisitos em um nível mais alto de abstração é necessário que o analista de requisitos tenha amplo conhecimento do domínio de negócio do sistema. No contexto do sistema de gerenciamento integrado de redes trata-se dos requisitos comuns, independentemente da tecnologia de rede, e daqueles que são particulares de cada tecnologia.

A motivação para realizar este projeto de mestrado é desenvolver um ferramental para auxiliar os desenvolvedores do sistema de gerenciamento integrado de redes a aplicar o reúso desde a fase de requisitos até a geração de código para incorporar novos elementos de uma nova tecnologia de rede. Neste contexto, serão aplicados os conceitos de MDD para a criação de uma DSL com a especificação das regras de negócio para a criação de uma nova tecnologia de rede. Dessa forma, será possível fazer reúso de requisitos de modo que o software seja desenvolvido dentro das metas de prazo e com redução de esforço. Com a conclusão deste projeto, será possível avaliar os benefícios alcançados com o uso de MDD para desenvolver sistemas de gerenciamento integrado de redes e a possibilidade de geração automática de código para esse sistema.

O objetivo deste projeto de mestrado é utilizar as técnicas de MDD para desenvolver uma linguagem específica de domínio para a gerência de configuração de um sistema de gerenciamento integrado de redes. Um gerador de código, para realizar as transformações *Model-To-Code* (M2C) a partir da DSL criada, também foi desenvolvido a fim de tornar o processo de desenvolvimento parcialmente automatizado. Ressalta-se que, mesmo assim, é requerido do desenvolvedor interação, ainda considerável, para o completo desenvolvimento do sistema.

1.3 Metodologia de Desenvolvimento do Trabalho

A metodologia usada para desenvolver este projeto de mestrado se iniciou com a parte teórica que constou de estudo sobre as abordagens utilizadas para realizar reúso de software. Uma revisão sistemática da literatura (RSL) foi conduzida utilizando a ferramenta StArt, desenvolvida na UFSCar, que organiza os textos e os classifica segundo critérios definidos pelo usuário (StArt, v.1.06.2).

Conhecendo os problemas existentes no desenvolvimento do sistema de gerenciamento integrado de redes, tanto pela falta de reúso de requisitos como de software, e com realização das disciplinas de mestrado que apresentaram abordagens para o reúso de software, surgiu o interesse em aplicar a abordagem do MDD no sistema de gerenciamento integrado de redes. Espera-se, desse modo, minimizar, na prática profissional, os problemas encontrados e construir sistemas mais completos e manuteníveis.

Dentre as funções do sistema de gerenciamento integrado de redes, os esforços de análise do código foram concentrados nas funções da gerência de configuração. Após essa fase de análise, foi possível desenvolver o modelo de mais alto nível, usando MDD, que contém as regras de negócio do domínio para descobrir os elementos de redes que especificam as particularidades de novas tecnologias de rede. Dessa forma, foi construída uma linguagem específica de domínio para a gerência de configuração, assim como um gerador de código, para realizar as transformações *Model-To-Code* (M2C) para a DSL criada. Tanto a DSL quanto o gerador de código foram aplicados por meio de experimento em um ambiente industrial com a finalidade de observar a ocorrência das vantagens do uso do MDD para desenvolver o software de forma mais rápida e com menos erros embutidos no código.

Como contribuições deste projeto de mestrado têm-se: a viabilidade da aplicação de MDD em um sistema já existente que requer reúso de funções comuns e específicas para novas tecnologias; a avaliação da possibilidade de se adotar uma nova forma de desenvolvimento de software na indústria; a apresentação para a equipe de desenvolvimento de uma nova forma de reúso com base em modelos de alto nível de abstração; a observação da aplicação da teoria de MDD em um sistema de uma empresa real.

1.4 Organização da Dissertação

Esta dissertação está organizada em seis capítulos, incluindo este, e dois apêndices, além das referências.

No Capítulo 2 estão reunidos os principais conceitos que foram utilizados para que o objetivo do projeto fosse alcançado. Dentre esses conceitos estão os seguintes: áreas

de gerenciamento, SNMP (*Simple Network Management Protocol*), redes de comunicação, desenvolvimento de software orientado a modelos, arquitetura dirigida a modelos, linguagem específica de domínio e algumas das principais ferramentas voltadas para o MDD.

No Capítulo 3 são apresentados o sistema de gerenciamento integrado de redes, suas funções e a forma como atualmente é desenvolvido.

No Capítulo 4 há a descrição do processo para a criação da DSL para a gerência de configuração do sistema de gerenciamento integrado de redes, que compreende a criação do metamodelo e a transformação do modelo para código, por meio da especificação de *templates*. Além disso, também é descrita a utilização da DSL criada.

No Capítulo 5 são apresentadas algumas definições sobre testes estatísticos, a definição, o planejamento, a aplicação do experimento realizado e a análise dos resultados nele obtidos.

No Capítulo 6 são apresentadas as considerações finais, as contribuições e as limitações deste trabalho, os trabalhos relacionados bem como sugestões para trabalhos futuros.

No Apêndice A estão ilustrados os formulários utilizados no experimento.

No Apêndice B é apresentado o passo a passo para a criação e utilização da DSL e do gerador de código utilizando as ferramentas adotas para este projeto.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

A reutilização de software auxilia no desenvolvimento de software com mais qualidade e produtividade, uma vez que há redução de esforço repetitivo e adoção de soluções que agregam conhecimento prévio (LUCRÉDIO, 2009).

O desenvolvimento orientado a modelos é uma das técnicas disponíveis para a realização de desenvolvimento com reúso, que enfatiza a utilização de modelos de alto nível de abstração. Esses modelos especificam as regras de negócio do domínio e, por meio de ferramentas adequadas, podem ser transformados em código para plataformas específicas. Dessa forma, o reúso pode ser feito em níveis mais altos de abstração e os desenvolvedores ficam protegidos da complexidade da implementação e podem se dedicar ao domínio do negócio (CZARNECKI e ANTKIWICZ, 2005).

Com a aplicação das técnicas de MDD na gerência de configuração do sistema de gerenciamento integrado de redes, a definição das regras de negócio para os novos elementos de rede de uma tecnologia específica pode ser feita nos modelos de alto nível de abstração e a complexidade de implementação fica encapsulada nas ferramentas de transformação. Além disso, o desenvolvimento do sistema pode ser mais rápido e com redução de custos para atender mais prontamente o mercado, cuja exigência está cada vez maior com a qualidade de produtos e serviços de rede de telecomunicações.

Neste capítulo são apresentados alguns conceitos sobre áreas de gerenciamento, SNMP, redes de comunicação que fazem parte do domínio de negócio do sistema de gerenciamento integrado de redes e sobre desenvolvimento dirigido a modelos que serviram de embasamento para a realização deste projeto de mestrado. Os conceitos de

áreas de gerenciamento, SNMP (*Simple Network Management Protocol*), redes de comunicação, de desenvolvimento dirigido a modelos, de MDA e de linguagem específica de domínio são apresentados, respectivamente nas Seções 2.4, 2.5, 2.6, 2.7, 2.6 e 2.7. Na Seção 2.8 são apresentadas as ferramentas que possibilitam a utilização de MDD e na Seção 2.9 estão as considerações finais deste capítulo.

2.2 Áreas de gerenciamento

O gerenciamento de rede possui cinco áreas comuns, conhecidas como FCAPS (MORRIS, 2003) desenvolvidas para o modelo de gerência OSI, contudo não deixa de ser compatível com o modelo SNMP:

- F – *Fault Management* (Gerência de falhas): responsável por detectar, localizar e corrigir os problemas de hardware ou software de uma rede. Atualmente existem sistemas de gerência que provêm antecipação de falhas por meio de rotinas de diagnóstico executadas em períodos de tempo pré-definidos e/ou por meio de correlação de alarmes, limiares e *syslog* para diagnosticar a iminência de determinada falha.
- C – *Configuration Mangement* (Gerência de configuração): compreende os registros de inventário de hardware e software, histórico de modificação dos dispositivos, inicialização dos sistemas que compõem a rede (como o sistema operacional e a configuração de um roteador), além de registros de topologia física, lógica e histórico de status dos dispositivos que compõe a rede.
- A – *Accouting Management* (Gerência de registros, logs ou bilhetes): possui a finalidade de registrar a utilização da rede para permitir contabilizar a utilização dos recursos da mesma, muito utilizado por provedores de acessos (ISPs) por motivos de tarifação de serviços, tais como: acesso discado, X.25, frame-relay, etc.
- P – *Performance Management* (Gerência de performance): responsável por saber ou definir se determinada rede está com um bom desempenho, uma vez que a rede pode ser vista de forma diferente por usuários de aplicações distintas; ou seja, enquanto ela pode ser considerada como rápida e eficaz para uma determinada aplicação também pode ser considerada extremamente lenta

ou incompatível para outra. Através da gerência de *performance* os administradores de rede podem monitorar certas variáveis chaves como *throughput*, tempo de resposta, disponibilidade, permitindo definir como e onde o desempenho da rede pode ser melhorado.

- S – *Security Management* (Gerência de segurança): responsável por regular e administrar o acesso aos recursos de rede e às determinadas informações, incluindo tarefas como: verificar o privilégio de acesso à rede dos usuários, detectar e registrar tentativas de acesso não autorizadas. Normalmente a autenticação, autorização e *accounting* de acesso a rede é feito de forma centralizada, e uma estrutura muito comum neste controle de acesso (principalmente para roteadores e *switchs*) é a utilização de um servidor Tacacs.

2.3 SNMP

O protocolo IP (*Internet Protocol*) vem se tornando o principal meio de interconexão de redes. Uma das principais funções de um sistema de gerenciamento integrado de redes é a interoperatividade com vários sistemas de gerência, independentemente de seu fabricante, respeitando alguns padrões. Esses padrões, atualmente, são o SNMP (*Simple Network Management Protocol*) e CMIP (*Common Information Protocol* – referência OSI). Sendo o SNMP o mais usado (HARNEDY, 1997).

O protocolo SNMP foi desenvolvido para permitir que dispositivos de rede que utilizam o protocolo IP possam ser gerenciados remotamente, através de um conjunto de operações.

Esse protocolo usa o modelo gerente/agente. Um servidor com a função de gerente (*Network Management System- NMS*) comunica-se com o agente de gerência de rede (instalado no dispositivo a ser gerenciado) através do protocolo de gerência.

O gerente é responsável por buscar as informações no agente (por meio de *polling*) ou por receber *traps* enviadas pelo agente informando alterações de *status*. Para receber *traps* não é necessário que o gerente faça uma solicitação prévia.

O agente é um software executado no dispositivo gerenciado, podendo ser um programa separado, ou seja, um *daemon*, em um servidor Unix; ou incorporado, por

exemplo, no Cisco, o IOS (*Internetwork Operating System*), para prover informações ao gerente.

2.3.1 Funcionamento do protocolo SNMP

O protocolo SNMP utiliza-se do UDP (*User Data Protocol*) como protocolo de transporte na comunicação entre o gerente e o agente. Um dos motivos pela escolha de um protocolo que não oferece garantia na comunicação é o fato do UDP ter menos *overhead*, reduzindo assim o impacto do sistema de gerência na *performance* da rede. As portas usadas pelo SNMP são:

- UDP porta 161 – para envio e recebimento de solicitações;
- UDP porta 162 – para receber *traps* dos elementos gerenciados.

SNMPv1 e SNMPv2 utilizam a *string* denominada como *community* para o estabelecimento de comunicação confiável entre o gerente e o agente. O agente pode ser configurado com três tipos de *community*: *read-only*, *read-write* e *trap*. A *community read-only* permite apenas leitura de dados no agente, a *read-write* permite leitura e modificação de dados e valores no agente, e *trap* permite envio de informações do agente para o gerente de forma assíncrona.

Uma das formas de entender o SNMP é entender o SMI (*Structure of Management Information*) que pode ser definido como: SMIv1 *(Structure of Management Information 1*, RFC 1155) e SMIv2 (*Structure of Management Information 2*, RFC 2578).

O SMIv1 define precisamente como os objetos gerenciados são denominados e especifica a qual tipo de dados eles estão associados. Esta definição pode ser resumida em três atributos: nome, tipo ou sintaxe e codificação.

O nome ou OID (*Object Identifier*) define um objeto como único. Pode aparecer na forma numérica ou amigável. Em ambos os casos os nomes são grandes e inconvenientes. Assim as aplicações SNMP buscam facilitar a navegação através destes nomes de forma simplificada.

A notação utilizada para definir um objeto é organizada em uma hierarquia em forma de árvore, onde o nome do objeto é formado pelo nome do caminho percorrido do nó principal até o ponto desejado, sendo estes separados por “.”. O nó principal denominado como *root* ou *root-node* não aparece no nome final, diz-se que este é definido como “ ”.

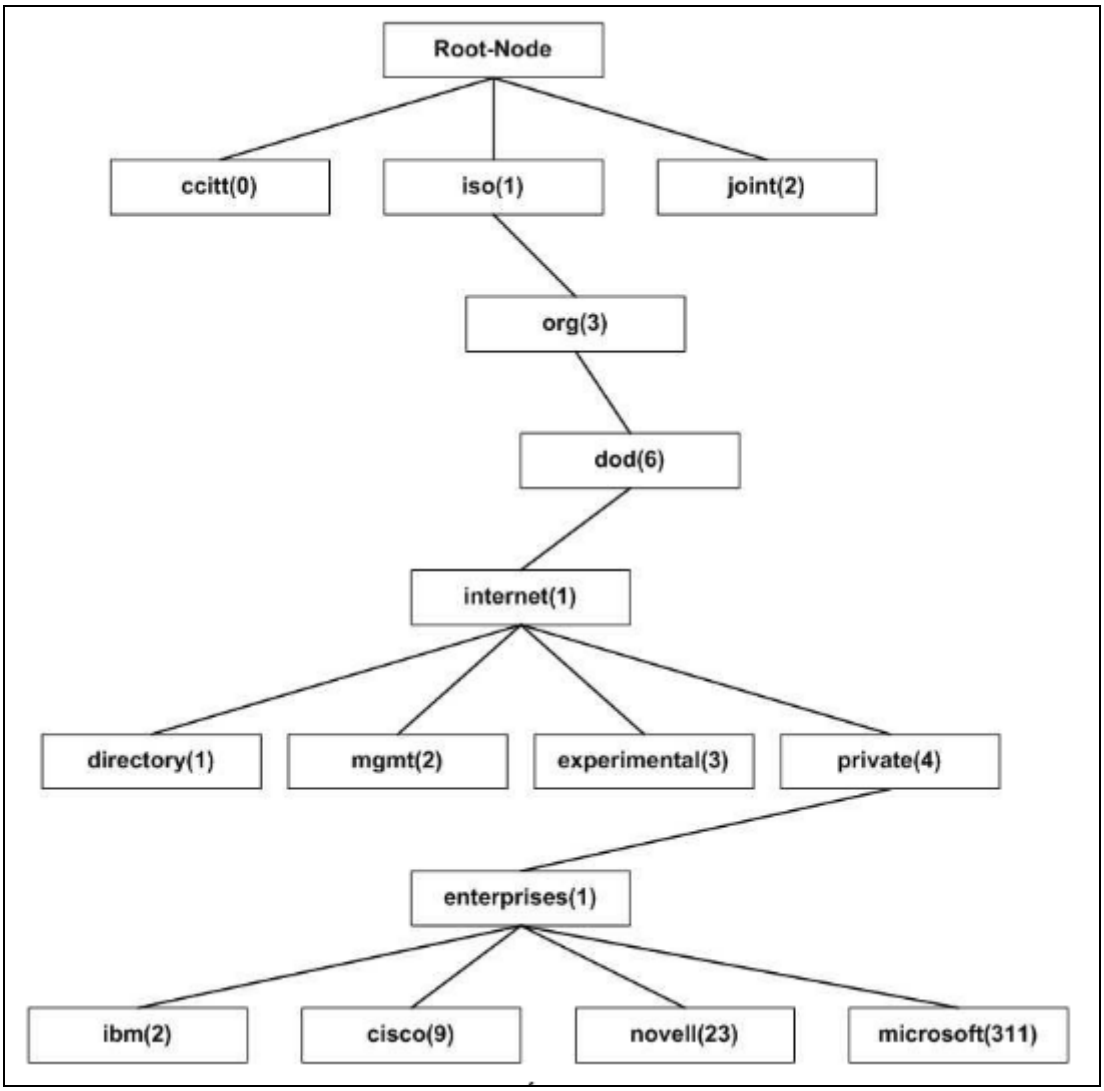

Figura 2-1: Árvore SMIv1 (Fonte: MAURO, SCHMIDT, 2001)

Dessa forma, por exemplo, percorrendo a árvore SMIv1, ilustrada na Figura 2-1, *iso, org, dod, internet, private, enterprises, microsoft*, a representação na forma OID é 1.3.6.1.4.1.311.

O tipo de dados de um objeto gerenciado é definido usando notações do ASN.1 (*Abstract Syntax Notation One*). O ASN.1 especifica como os dados são representados e transmitidos entre o gerente e o agente, dentro do contexto do protocolo SNMP, o que torna esta notação independente do tipo de máquina.

Uma instância de um objeto gerenciado é codificada em uma string de octetos usando BER (*Basic Enconding Rules*). O método BER define como os

objetos são codificados e decodificados para permitir sua transmissão em um meio físico qualquer.

SMIv2 (*Structure of Management Information 2*, RFC 2578), ilustrada na Figura 2-2, amplia a árvore de objetos SMI adicionando o braço snmpV2 á sub-árvore Internet, adiciona novos tipos de dados e modifica a definição de um objeto adicionando novos campos, dando mais controle sobre a forma como um objeto é acessado, permite aumentar uma tabela adicionando mais colunas e oferece descrições melhores.

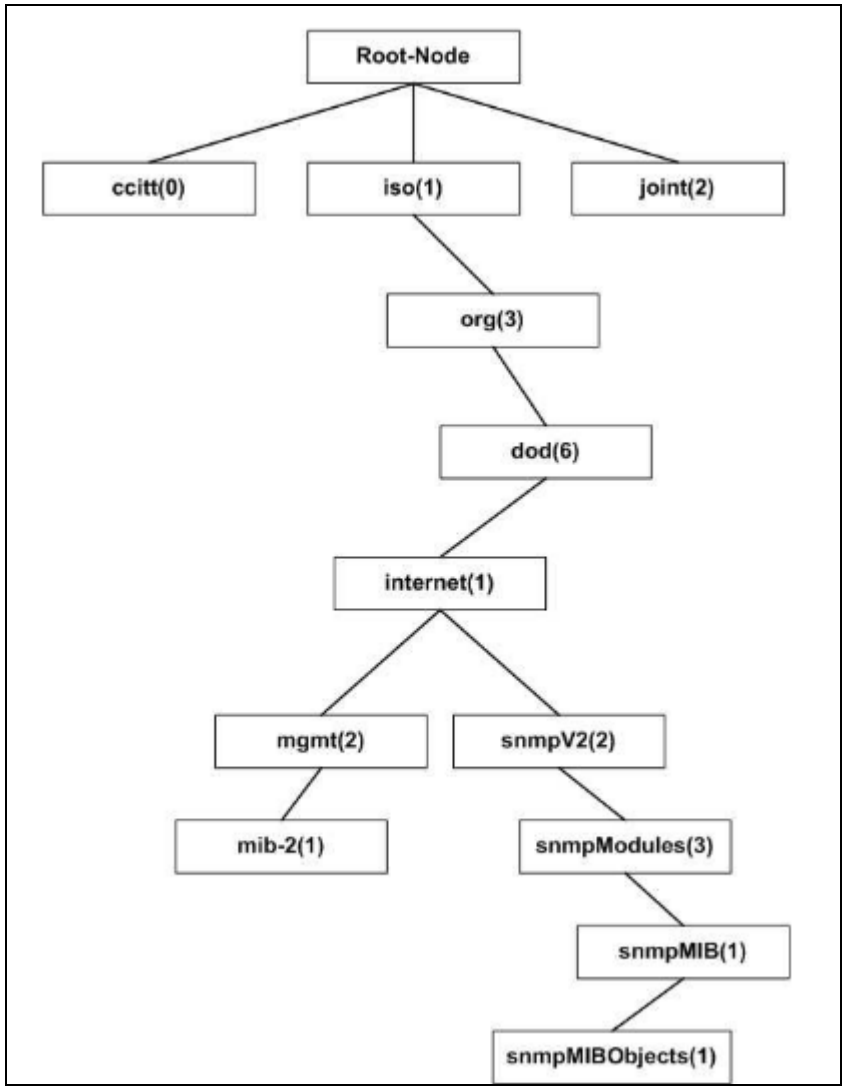

Figura 2-2: Árvore SMIv2 (Fonte: MAURO, SCHMIDT, 2001)

A SMIv1 (RFC 1155) define como os objetos gerenciados são nomeados e especifica os respectivos tipos de dados associados. A SMIv2 (RFC 2578) oferece otimizações para o SNMPv2.

2.3.2 MIB

A MIB (Base de Informação de Gerenciamento) são variáveis dispostas de forma hierárquica (árvore) e expressem diversos tipos de valores que, dentre outras coisas, servem para gerenciar e analisar as redes de computadores. Ela pode ser considerada como um banco de dados de objetos gerenciados, que por sua vez, são consultados/manipulados pelos agentes SNMP.

A MIB pode ser entendida como uma tabela de um banco de dados. Por exemplo, tabela de clientes, o objeto gerenciável (variável MIB) como um atributo da tabela, por exemplo, nome_cliente, e a SMI como a definição/documentação do tipo de dado a ser armazenado no atributo para representar o nome do cliente, por exemplo, varchar(50), ou seja, a SMI é o método para definir objetos gerenciados, enquanto a MIB é a definição (por meio da sintaxe SMI) dos próprios objetos. Como um dicionário, que mostra como pronunciar uma palavra e, em seguida, apresenta o significado ou a definição dessa mesma palavra, uma MIB define um nome em texto de um objeto gerenciado e explica o seu significado.

Na MIB os objetos podem ser simples ou em tabelas. O objeto é simples quando é identificado por um caminho disposto entre a raiz e o objeto desejado. O objeto fica organizado em tabela quando tem várias instâncias, onde as colunas representam os objetos e as linhas representam as instâncias.

Todos os agentes, independente do tipo de elemento gerenciado, possui uma MIB padrão denominada MIB-2. Porém também existem os módulos MIB específicos para determinados tipos de elementos gerenciados, por exemplo, módulos especiais para roteadores, repetidores, interfaces Ethernet, ATM etc.

Um dos principais grupos de gerenciamento é o MIB-II cujo OID é o 1.3.6.1.2.1.

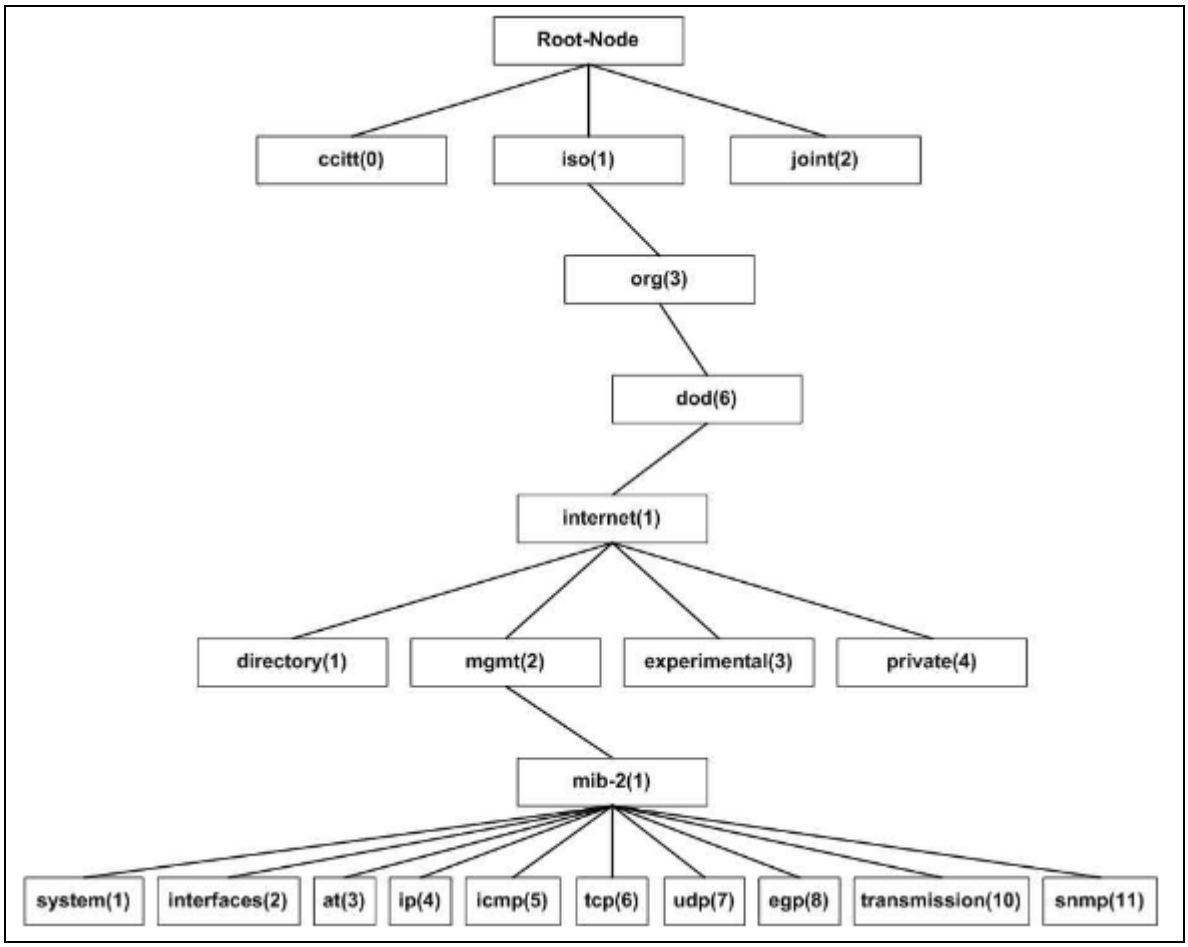

Figura 2-3: Sub-árvore MIB-II (Fonte: MAURO, SCHMIDT, 2001)

Uma breve descrição do grupo MIB-2 pode ser vista na tabela Tabela 2-1.

Tabela 2-1 Grupo MIB-2

Nome da sub-árvore	OID	Descrição
system	1.3.6.1.2.1.1	Define uma lista de objetos que pertencem ao sistema operacional, como *uptime* do sistema, contato do sistema, nome do sistema.
interfaces	1.3.6.1.2.1.2	Mantém históricos de *status* de cada interface da entidade gerenciada, como *status* de *up* ou *down*, octetos enviados e recebidos, erros, descartes, etc.
at	1.3.6.1.2.1.3	grupo at (*address translation)* está obsoleto e

		existe para manter compatibilidade.
ip	1.3.6.1.2.1.4	mantém históricos de diversos aspectos referentes do protocolo IP, inclusive roteamento.
icmp	1.3.6.1.2.1.5	trata aspectos como pacotes de erro ICMP, descartes, etc.
tcp	1.3.6.1.2.1.6	trata entre outras coisas do estatus do TCP: *closed*, *listen*, *synSent*, etc
udp	1.3.6.1.2.1.7	rata de estatísticas do UDP, datagramas recebidos e enviados, etc.
egp	1.3.6.1.2.1.8	trata de várias estatísticas sobre EGP e mantém uma tabela de vizinhança EGP.
transmission	1.3.6.1.2.1.10	Atualmente não há nenhuma definição de objeto neste grupo, mas outras especificações de mídia são definidas usando esta sub-árvore.
snmp	1.3.6.1.2.1.11	Medição da performance SNMP na entidade gerenciada, tratando coisas como número de pacotes SNMP recebidos e enviados.

2.4 Redes de Comunicação

Nas subseções seguintes estão descritas as características das tecnologias de rede consideradas para desenvolver o sistema de gerenciamento integrado de rede: IP, LTE e FTTx.

2.4.1 Rede IP

O gerenciamento de falha e de degradação de desempenho na rede IP foi padronizado pela família de protocolos SNMP (*Simple Network Management Protocol*), que se encontra na versão 2 (SNMPv2). Refere-se a um conjunto de padrões para gerenciamento que inclui um protocolo, uma especificação de estrutura de dados e um conjunto de objetos de dados (TANENBAUM, 2011).

A arquitetura de gerenciamento é composta pelos seguintes elementos: estação de gerenciamento, agente de gerenciamento, Base de Informações de Gerenciamento (MIB) e protocolo de gerenciamento de redes. A estação de gerenciamento serve como interface entre o gerente e o sistema de gerenciamento de rede. O agente de gerenciamento responde às solicitações de informações e de ações da estação de gerenciamento e deve também prover assincronamente informações importantes que não foram solicitadas por essa estação por meio do protocolo de comunicação. Os recursos a serem gerenciados são representados como objetos e a coleção de objetos é referenciada como Base de Informações de Gerenciamento (MIB). A forma de comunicação entre a estação de gerenciamento e o agente é definida pelo protocolo de gerenciamento de rede, o SNMP.

Os elementos gerenciados pelo sistema de gerenciamento integrado de rede para a tecnologia IP são qualquer equipamento que possua identificação na rede e o protocolo SNMPv2.

2.4.2 Rede LTE

LTE (*Long Term Evolution*), padrão para comunicação sem fio para terminais móveis, é a nova geração das redes móveis padronizada pelo 3GPP (*3rd Generation Partnership Project)* (D'ÁVILA, 2009). O objetivo dessa rede é aumentar o *throughput* do usuário, melhorar a capacidade do setor e reduzir a latência do plano do usuário proporcionando mais mobilidade. Essa tecnologia fornece suporte baseado em IP com *QoS (Quality of Service)* fim-a-fim e tem por objetivo minimizar a complexidade do sistema e dos equipamentos dos usuários, para permitir a distribuição flexível do espectro com novas frequências ou das faixas já utilizadas e para permitir a coexistência dessa rede com outras já implantadas como o GSM (*Global System for Mobile)* e o WCDMA

(*Wideband Code Division Multiple Access)* (CASTRO, 2009), além de oferecer altas taxas de *downlink* e *uplink*. A topologia da rede LTE está representada na Figura 2-4.

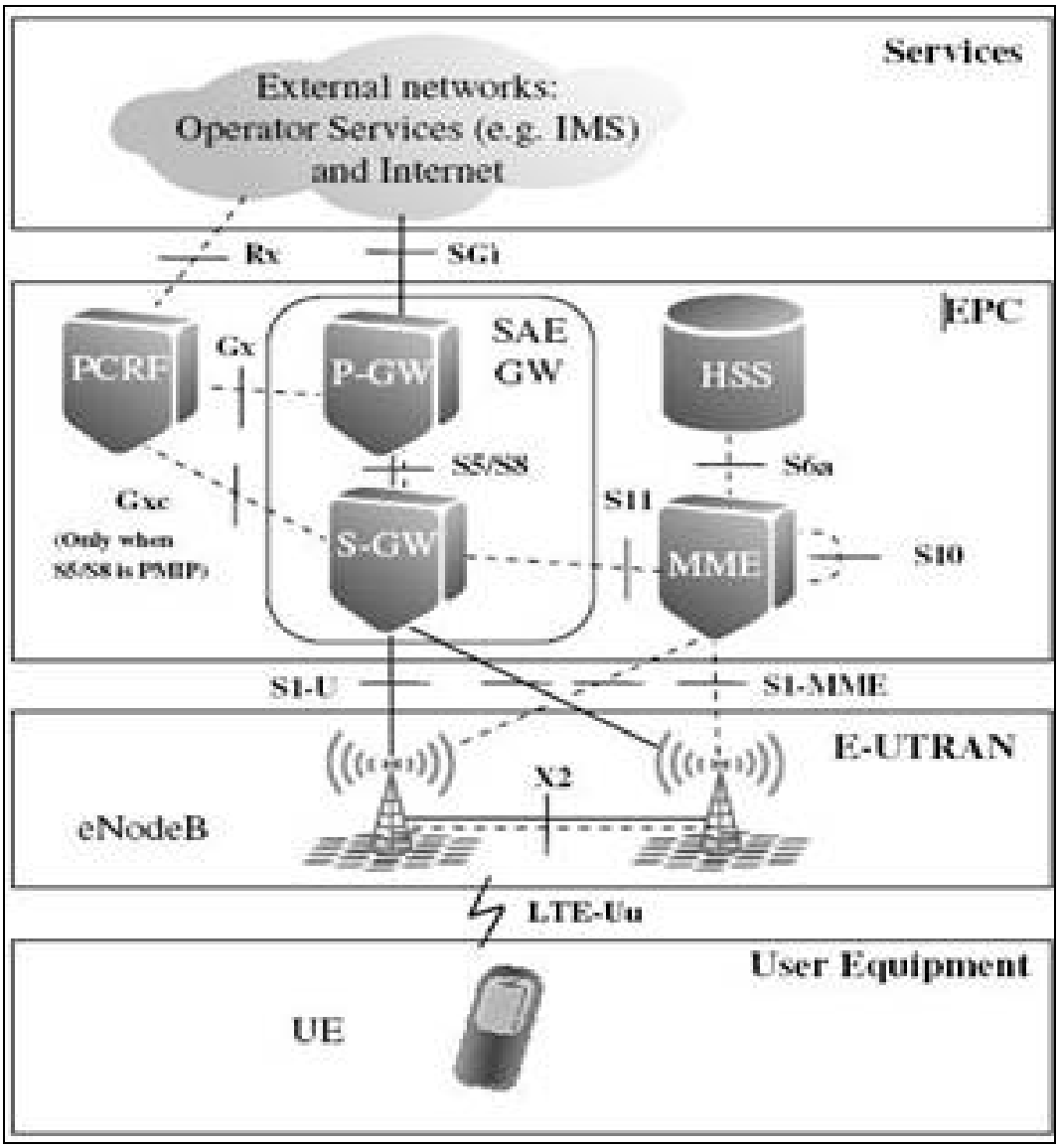

Figura 2-4 Topologia LTE (Fonte: D'ÁVILA, 2009)

Essa rede possui quatro grandes domínios (D'Ávila, 2009):

- *User Equipament* (EU): dispositivo de acesso do usuário;
- E-UTRAN (*Evolved UMTS Terrestrial Radio Access Network)*: composta de uma rede *mesh* de *eNodeBs* que se comunicam por meio da interface X2. A *eNodeB* contém as camadas física (PHY), *Medium Accesss Control* (MAC), *Radio Link Control* (RLC) e o protocolo de controle de pacotes de dados. Inclui a funcionalidade de compressão de cabeçalho, criptografia, gestão de recursos

do rádio, controle de admissão, negociação de *QoS* no *uplink* e *broadcast* contendo informações da célula;

- EPC: contém os elementos da rede, que desempenham as principais funções do sistema e são definidos como:
    - MME (*Mobility Management Entity*): principal elemento de controle no EPC (*Evolved Packet Core*), que tem entre as suas funções: autenticação, segurança, gerenciamento de mobilidade, gerenciamento de perfil do usuário, conexão e autorização de serviços;
    - S-GW (*Serving Gateway*): faz o roteamento dos pacotes de dados dos usuários entre a rede LTE e outras tecnologias como o 2G/3G, utilizando a interface S4. Gerencia e armazena informações do UE (*User Equipment)* como parâmetros de serviços IP suportados e informações sobre o roteamento interno dos pacotes na rede;
    - P-GW (*Packet Data Network Gateway*): é o roteador de borda entre o EPC (*Evolved Packet Core)* e redes de pacotes externas. Realiza a filtragem e controle de pacotes requeridos para os serviços em questão. Tipicamente, o P-GW aloca endereços IP para os equipamentos dos usuários para que eles possam se comunicar com outros dispositivos localizados em redes externas;
    - PCRF (*Policy and Charging Resource Function*): elemento de rede responsável pelo PCC (Política e Controle de Carga). Provê o *QoS* (*Quality of Service*) adequado para que os serviços solicitados possam utilizar os recursos apropriados;
    - HSS (*Home Subscriber Server*): banco de dados de registro do usuário que executa funções equivalentes às do HLR (*Home Location Register*), AuC (*Authentication Center*) e EIR (*Equipment Identity Register)*.
- Serviços: provê a interligação do LTE com outras redes.

Essa arquitetura permite redução considerável de custos referentes à operação e à aquisição de equipamentos, uma vez que o E-UTRAN pode ser compartilhado por várias operadoras. Já no EPC, cada uma possui equipamentos próprios e define a sua própria topologia e os seus elementos de núcleo da rede com MME, S-GW e P-GW.

No sistema de gerenciamento integrado de redes, o equipamento gerenciando é a *eNodeB*.

2.4.3 Rede FTTx

A rede FTTx (*Fiber To The x*) sendo que a letra “x” pode designar os seguintes pontos de acesso à rede: FTTn *Fiber-To-The-node*, FTTc - *Fiber-To-the-cabinet* ou Fiber-To-The-curb, FTTb - *Fiber-To-The-building* ou *Fiber-To-The-basement*, FTTh - *Fiber-To-The-home* e FTTp - *Fiber-To-The premises*. Essa rede usa a tecnologia PON (*Passive Optical Network*) (SANCHEZ, 2013). Na PON não há equipamentos ativos no meio do enlace entre cliente e prestador de serviço, existem apenas equipamentos passivos que se situam mais próximos do cliente para entregar os dados com altas taxas de velocidade para banda larga (PINHEIRO, 2013). Uma arquitetura simplificada é apresentada na Figura 2-5.

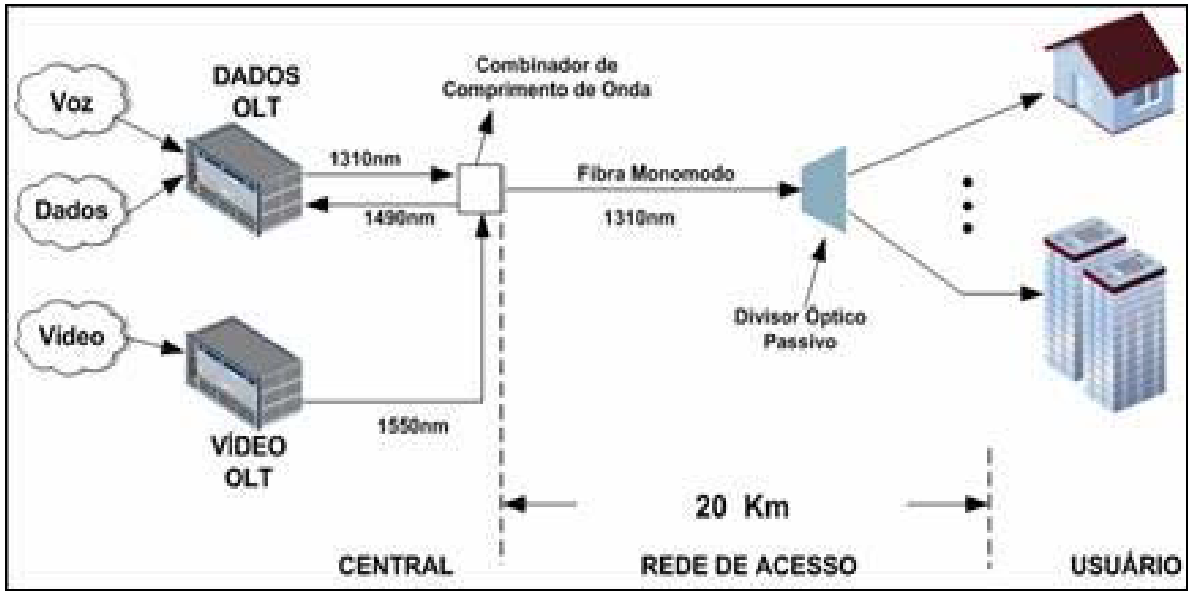

Figura 2-5: Arquitetura Fundamental de uma rede PON (Fonte: KEISER, 2006)

A rede PON utiliza uma configuração de rede ponto-multiponto, uma única fibra é compartilhada por diversos pontos finais de atendimento (residências e empresas). Assim, pode-se usar no armário de distribuição diversos divisores ópticos (*splitters*) na mesma fibra.

Como a conexão é ponto-multiponto, os terminais ópticos no cliente (ONU – *Optical Network Unit*) devem-se orientar para executar determinadas funções, como, por exemplo, filtrar apenas as informações daquele usuário e, também, coordenar para que pela multiplexação os sinais que saem do cliente não colidam com outras informações.

O tráfego de informações *downstream* é transmitido em modo *broadcasting*, ou seja, a informação é transmitida a todos os elementos da rede. A mesma informação chega a todos os usuários, sendo necessário utilizar um sistema de criptografia das informações para manter privacidade na comunicação.

Várias topologias podem ser a aplicadas a uma Rede Óptica de Acesso, como por exemplo, barra, estrela, anel e árvore que serão comentadas a seguir.

Topologia Barra (Figura 2-6): provê uma conectividade ponto-multiponto entre OLT (*Optical Line Terminal)* e ONT/ONU (*Optical Network Terminal /Optical Network Unit)*, mas qualquer falha no enlace principal causa a desconexão com os usuários.

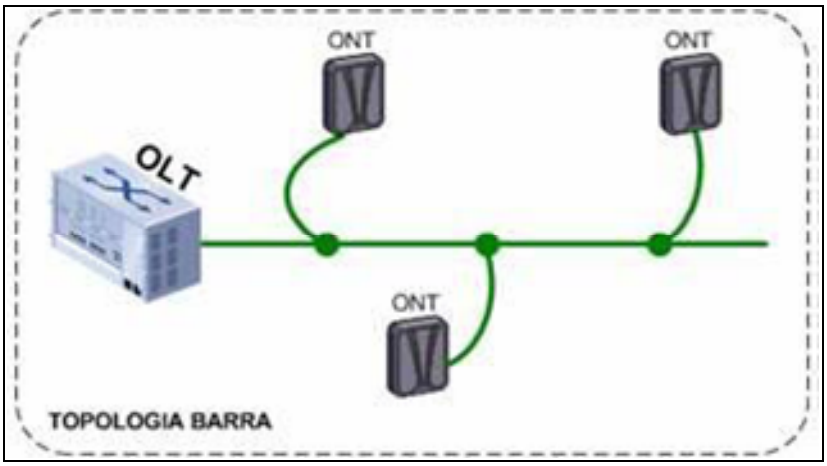

Figura 2-6: Representação Topologia Barra (Fonte: KEISER, 2006)

Topologia Estrela (Figura 2-7): provê uma conectividade ponto-a-ponto entre OLT e ONT/ONU. Esta topologia permite entrega de banda dedicada de altas taxas aos usuários finais e também possui um baixo custo em operação, administração e manutenção.

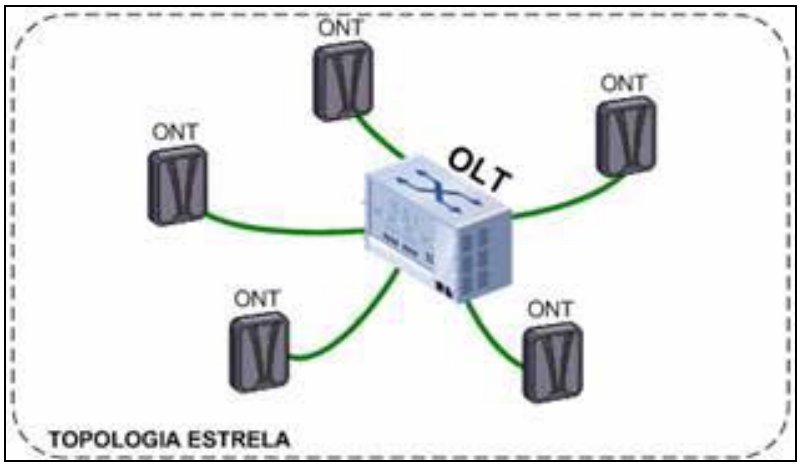

Figura 2-7: Representação Topologia Estrela (Fonte: KEISER, 2006)

Topologia em Anel (Figura 2-8): oferece a vantagem ponto-multiponto da OLT para a ONT/ONU. Permite implementação de mecanismos de proteção – enlace com redundância – mas possui dificuldades para as funções de operação, administração e manutenção.

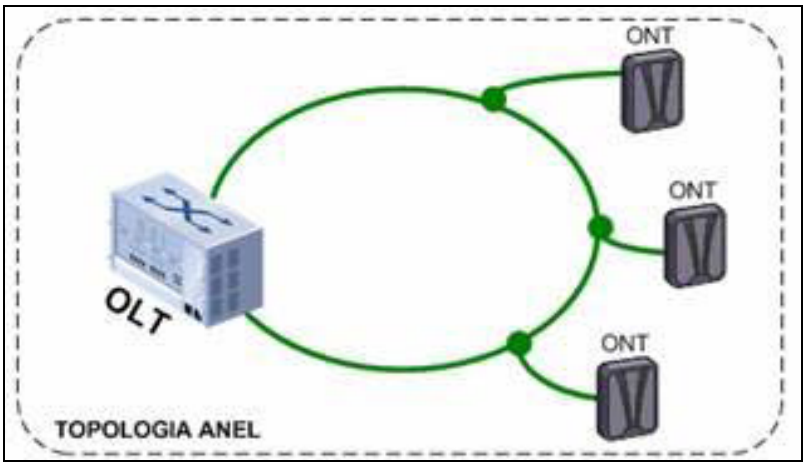

Figura 2-8: Representação Topologia Anel (Fonte: KEISER, 2006)

Topologia Árvore (Figura 2-9): provê conectividade ponto-multiponto que oferece a vantagem de infraestrutura compartilhada entre todos os usuários, possuindo assim uma importante redução nos custos de implementação e manutenção na rede de acesso. Esta arquitetura é uma das mais difundidas nos estudos relacionados à Rede PON.

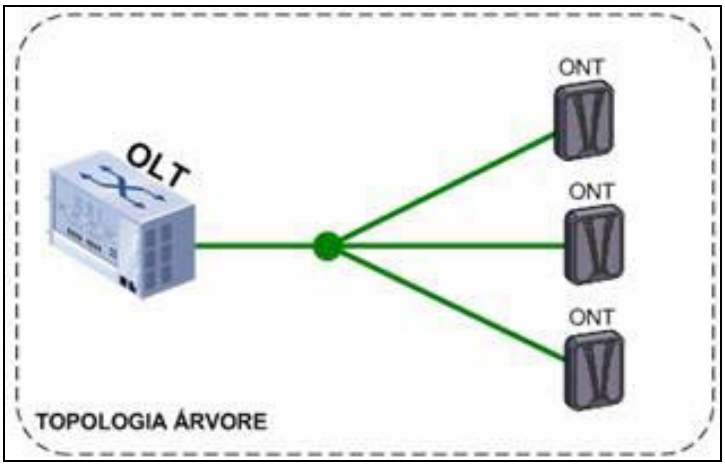

Figura 2-9: Representação Topologia Árvore (Fonte: KEISER, 2006)

Em uma rede óptica PON, os elementos passivos (cabos ópticos, divisores passivos, conectores e acopladores) ficam localizados na planta externa, onde ocorre a distribuição óptica. Os únicos elementos ativos são a OLT na central e a ONU ou ONT que ficam próximos ao cliente.

A OLT normalmente instalada dentro da Central da Operadora, controla o fluxo de informações bidirecional para a ONT/ONU. Geralmente, a distância da OLT até ONT/ONU é de 20 km. Uma mesma OLT controla mais de uma ONT, como é possível observar na Figura 2-10.

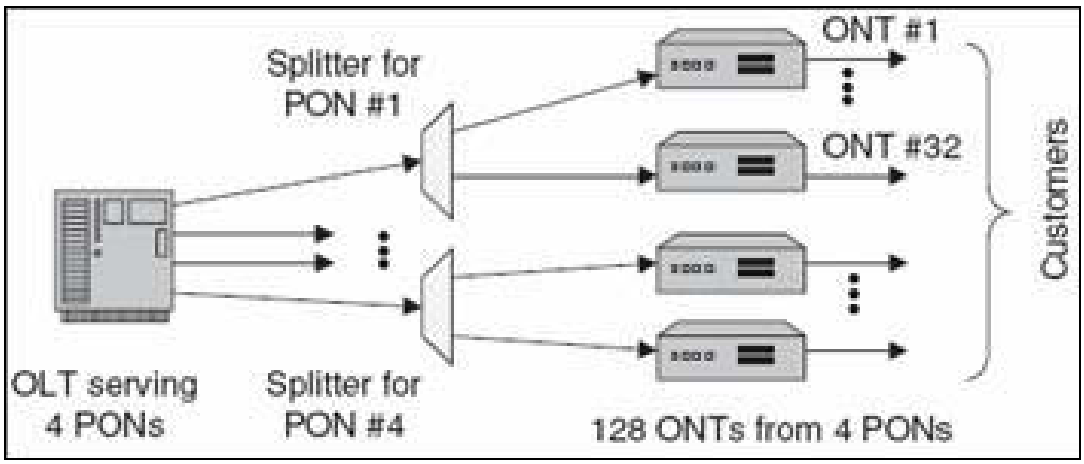

Figura 2-10: Atendimento de quatro redes PON por uma OLT (Fonte: KEISER, 2006)

A OLT se conecta a ONU ou ONT, que é o equipamento que provê a interface entre os dados do cliente, vídeo e redes de telefonia.

O terminal de rede óptica fica instalado na casa do cliente e permite que ele escolha a taxa de banda larga para prover seus serviços. Esse equipamento permite a alocação de banda dinâmica, ou seja, transmite em curtos espaços de tempo que são controlados pela OLT, tendo assim uma intensa utilização da banda alocada. Existe uma imensa variedade de ONTs no mercado, de diversos fabricantes, com funcionalidade e configurações para atender diversas larguras de banda.

A ONU é um equipamento instalado ao ar livre, que tem uma bateria reserva para eventuais quedas de energia e medidas de proteção do equipamento devem ser tomadas quanto a mudanças climáticas e a atuação de vândalos.

A ligação da ONU até o equipamento do cliente pode ser realizada através de cabo par metálico, cabo coaxial, ligação de fibra óptica independente ou ainda uma conexão sem fios.

Além de realizar a interface da OLT com o cliente, a ONU também é responsável pela multiplexação e demultiplexação dos serviços cliente/operadora e operadora/cliente, e mais o fornecimento de energia. Também existem diversas ONU no mercado e a mesma deve ser adquirida de acordo com a aplicação FTTx.

No sistema de gerenciamento integrado de redes, os equipamentos gerenciados são as OLTs e as ONUs ou ONTs.

2.5 MDD (*Model Driven Development*)

No desenvolvimento de software, comumente são criados modelos que representam as abstrações das entidades do mundo real para auxiliar no entendimento comum do que deve ser implementado. Esses modelos são especificados segundo uma linguagem de modelagem, por exemplo, a *Unified Modeling Language* (UML) (LANGE; CHAUDRON, 2005). Eles são criados antes da implementação e servem também como uma forma de documentação. Porém, esses modelos, embora importantes para o entendimento e construção do software, não fazem parte efetivamente do mesmo. Eles deixam de ser atualizados e somente o código-fonte passa a ser detentor da informação do que foi entendido e implementado. Dessa forma, há a necessidade de se criar modelos para serem utilizados como documentação, na construção e na manutenção do software (LUCRÉDIO, 2009).

O desenvolvimento orientado a modelos (*Model Driven Development* – MDD) surge enfatizando a importância dos modelos no processo de desenvolvimento de software (MELLOR; CLARK; FUTAGAMI, 2003). Nessa abordagem de desenvolvimento de software, os modelos de abstração de mais alto nível são refinados em outros modelos até que sejam transformados em código-fonte. Essa transformação é realizada de forma automática por meio do uso de ferramentas. Os modelos de alto nível de abstração, que descrevem as funções do sistema, têm importância uma vez que passam a ser parte integrante do software e deixam de ser somente documentação ou orientação para o desenvolvimento do software (DURELLI, 2011). Com a utilização de modelos de alto nível de abstração e a automação para a geração de código, os desenvolvedores de software ficam protegidos das complexidades da plataforma de implementação (FRANCE; RUMPE, 2007), e podem ter maior dedicação ao domínio do problema (HUTCHINSON *et al*., 2011). Pois parte da complexidade do software fica escondida dentro dos geradores automáticos dos artefatos de implementação, que refletem a solução expressa nos modelos de alto nível de abstração (SCHMIDT, 2006).

O uso de modelo facilita a obtenção dos requisitos dos *stakeholders*, pois é uma forma mais direta e livre de detalhes de implementação (DURELLI, 2011). No MDD, a visão de um sistema é representada por um modelo. Todos os modelos são especificados de acordo com a linguagem do seu metamodelo. Na Figura 2-11 está ilustrada a relação entre modelo e metamodelo, com uma comparação entre o desenvolvimento dirigido a modelos (MDD) e o desenvolvimento orientado a objetos (OO). No desenvolvimento orientado a objetos o objeto é uma instância de uma classe, que por sua vez herda as características e comportamentos de sua superclasse. No desenvolvimento orientado a modelo um sistema é representado por um modelo, que por sua vez segue todas as regras conforme especificado em seu metamodelo.

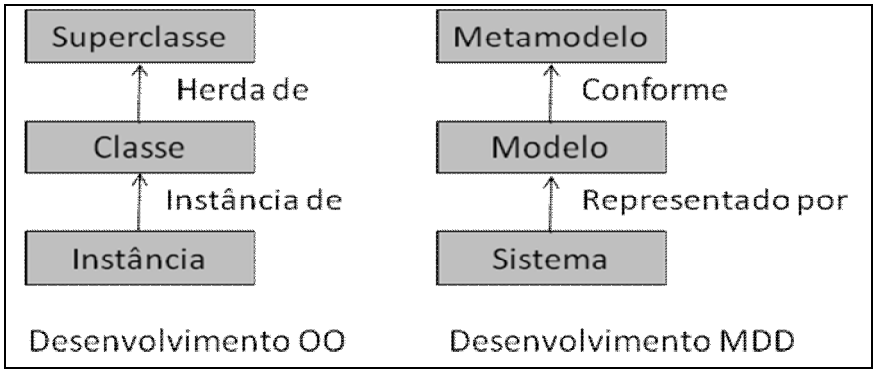

Figura 2-11: Paradigma de desenvolvimento OO e MDD (Fonte: SILVA, 2010)

As vantagens do MDD, citadas por Kleppe, Warmer e Bast (2003), Bahnot *et al*. (2005) e Mernik, Heering e Sloane (2005), são:

- Produtividade: pode-se utilizar mais tempo na criação de novos modelos de alto nível, pois a geração de código fica no âmbito das transformações implementadas nas ferramentas usadas para esse propósito;
- Portabilidade: um mesmo modelo pode ser transformado em código para diferentes plataformas;
- Interoperabilidade: a partir do momento que um mesmo modelo pode ser transformado em código para plataformas distintas, o software pode ser executado em ambiente heterogêneo, sem alterar sua funcionalidade. Além disso, é possível gerar adaptadores e conectores utilizando tecnologias independentes de plataforma, para que esses códigos de diferentes plataformas possam se comunicar;
- Manutenção e documentação: como no MDD os modelos fazem parte do software, qualquer alteração é feita no modelo e não diretamente no código; assim, modelo e código ficam consistentes. Desta forma, a documentação mantém-se atualizada, o que facilita as tarefas de manutenção;
- Comunicação: como os modelos são mais abstratos do que o código, não é preciso que clientes e especialistas do domínio tenham conhecimento técnico da plataforma. Dessa forma, a comunicação fica mais efetiva para identificar regras de negócio por meio dos modelos;
- Reutilização: no MDD, a reutilização fica mais abrangente (modelo e código), pois o código pode ser re-gerado a partir do novo contexto do modelo;

- Verificações e otimizações: a verificação semântica pode ser feita de forma mais eficiente uma vez que se utiliza modelos e as otimizações específicas de domínio podem ser executadas;
- Corretude: os geradores não introduzem erros acidentais e permitem que a identificação de erros conceituais aconteça em um nível mais alto de abstração.

As desvantagens do MDD, segundo os autores Ambler (2003), Thomas (2004), são:

- Rigidez: grande parte do código gerado, por meio da ferramenta de transformação, não é de autoria do desenvolvedor e não deve ser alterado;
- Complexidade: para fazer uso de MDD, devem ser adotadas ferramentas de modelagem, transformação e geradores de código que são mais complexos de se trabalhar;
- Desempenho: geradores de código acabam incluindo código desnecessário e podem não resultar em desempenho ótimo;
- Curva de aprendizado: é necessário treinamento para se trabalhar com as ferramentas de modelagem, transformação e geradores de código;
- Alto investimento inicial: a construção de uma infraestrutura de reutilização orientada a modelos requer mais tempo e esforço. Mas, posteriormente, o uso do MDD resultará em ganhos significativos.

Na Figura 2-12 é apresentado um esquema dos elementos necessários para a transformação de modelo para código.

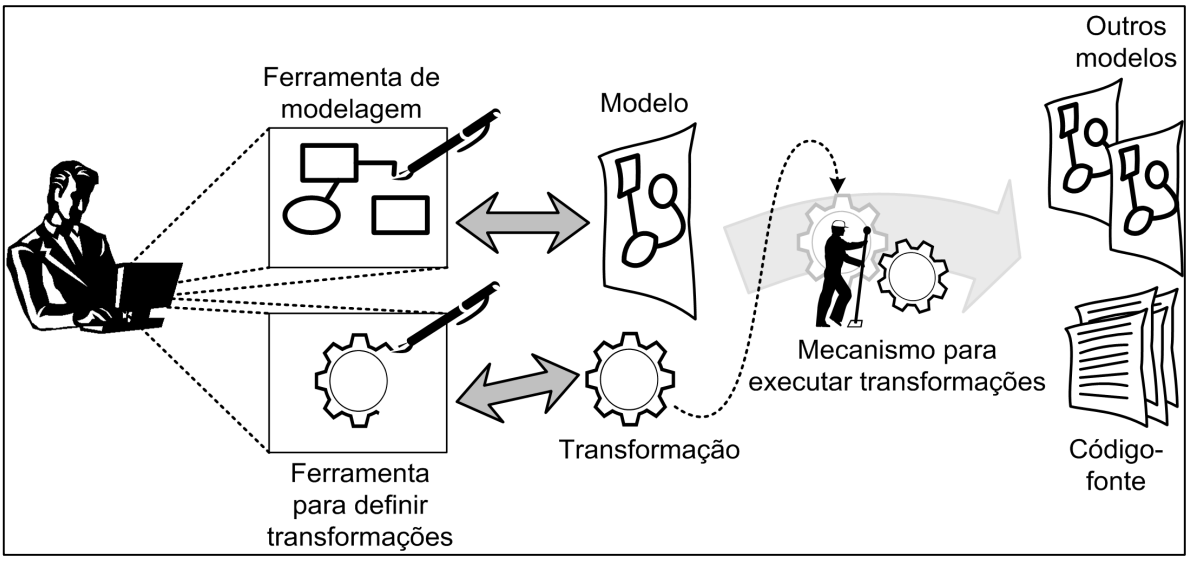

Figura 2-12: Principais elementos do MDD (Fonte: LUCRÉDIO, 2009)

Os modelos são as entradas para as transformações que irão gerar outros modelos ou o código-fonte. Para especificar esses modelos, que representam o negócio do domínio, é necessário usar uma ferramenta de modelagem. Esses modelos devem ser semanticamente completos e corretos para que possam ser entendidos pelas ferramentas computacionais a fim de corrigir ou indicar erros no modelo. Para definir as transformações, seja de modelo para modelo, ou de modelo para código, é necessária uma ferramenta computacional para construir as regras de mapeamento. Também é necessário um mecanismo que efetivamente aplique as transformações e mantenha as informações de rastreabilidade, possibilitando conhecer a origem de cada elemento gerado, seja no modelo ou no código-fonte.

2.6 MDA (*Model Driven Architecture*)

A preocupação da MDA é especificar o software independentemente de plataforma, ou seja, independente de sistema operacional, de linguagem de programação, de arquitetura e de servidor de aplicação entre outras restrições. Portabilidade, interoperabilidade e reusabilidade por meio da separação de interesses arquiteturais são os objetivos da MDA (OMG, 2003) que define três classes de modelos, a forma como eles podem ser especificados e o relacionamento entre eles. As classes de modelo definidas na MDA são:

- *Computer Independent Model* (CIM) especifica, por meio do modelo de domínio, o que é esperado do sistema. É descrito em formato não computacional, o que o impede de ser usado pelas ferramentas de transformação;
- *Plataform Independent Model* (PIM) consiste no modelo de domínio em formato computacional, independente da plataforma a ser utilizada, sem detalhes de implementação. Porém, o nível de independência de plataforma é aquele que torna o modelo possível de ser utilizado em diferentes plataformas de tipos similares;
- *Plataform Specific Model* (PSM) é o conjunto das especificações do PIM com a descrição de como um sistema usa um tipo particular de plataforma, ou seja, é um PIM com os elementos específicos de uma plataforma. Um PIM pode ser

transformado em um ou mais PSMs. Para cada plataforma, um PSM específico é gerado.

O processo de desenvolvimento se inicia com a especificação do CIM pelo especialista de negócio em conjunto com o usuário. O CIM é transformado em PIM por um arquiteto de software, mas sem adição de detalhes de plataforma. O PIM é transformado em PSM para uma plataforma específica pelo especialista da plataforma. Esse processo está esquematizado na Figura 2-13.

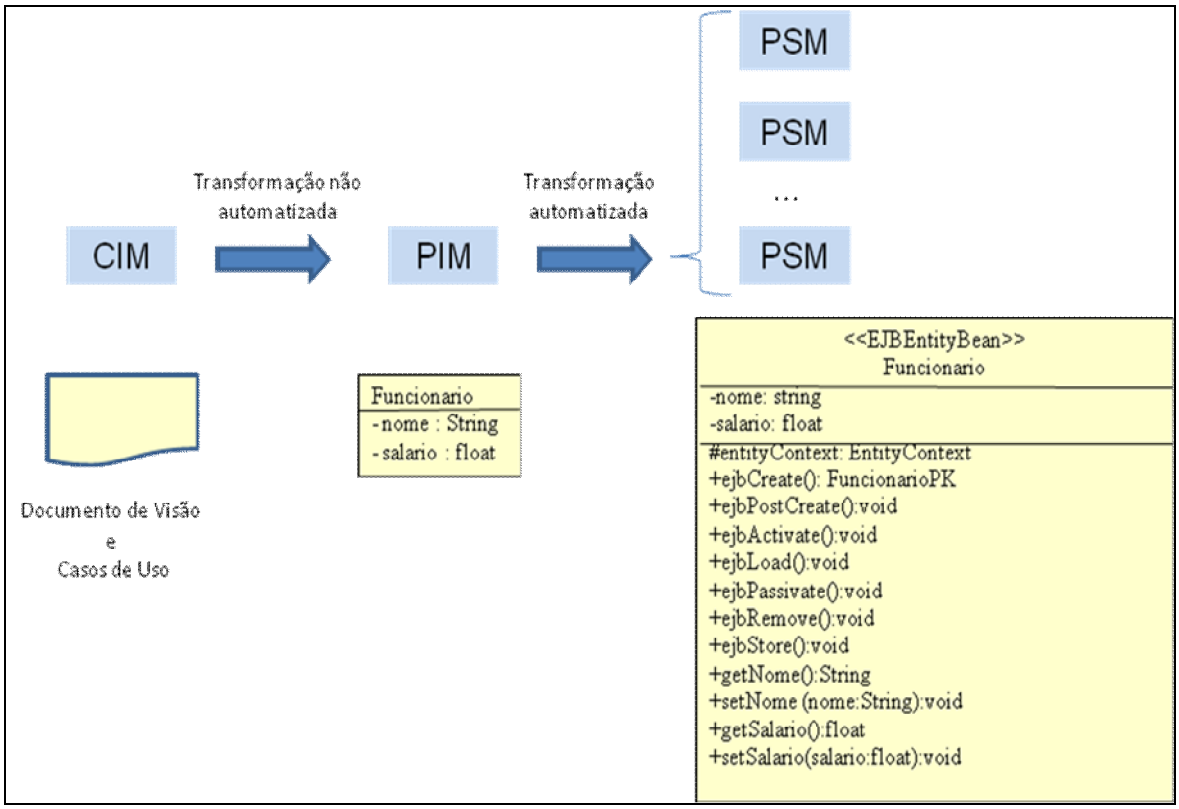

Figura 2-13: Processo de desenvolvimento MDA

Na MDA, os modelos são transformados em outros modelos até serem transformados no código-fonte. Dependendo da complexidade da transformação do CIM até o PSM, podem ocorrer várias transformações horizontais dentro de uma mesma camada de abstração, ou seja, vários modelos PIM, até se chegar ao PSM. A transformação do CIM em PIM dificilmente é automatizado por causa das interpretações dos requisitos. Mas, a transformação do PSM em código-fonte é passível de automação, pois o PSM especifica a plataforma de implementação (LUCRÉDIO, 2009).

A organização arquitetural da MDA é feita em quatro níveis. Essa organização está ilustrada na Figura 2-14. O nível mais baixo é a camada M0 que representa o sistema. Um modelo representa esse sistema no nível M1. Esse modelo está conforme seu metamodelo definido no nível M2 e o metamodelo está conforme o meta-metamodelo no nível M3.

O nível principal da MDA é o nível M3 que permite a construção coordenada entre modelos, baseada em diferentes metamodelos. A utilização do padrão *Meta Object Facility* (MOF) no nível M3 permite o uso de metamodelos padronizados. Além disso, o MOF permite mapear MDA com outros padrões como, por exemplo, XML (por meio do padrão XMI) ou Java (por meio do padrão JMI). Outra linguagem bastante utilizada é a Ecore (BUDINSKY, 2003) usada no *Eclipse Modeling Framework* (EMF) (EMF, 2009).

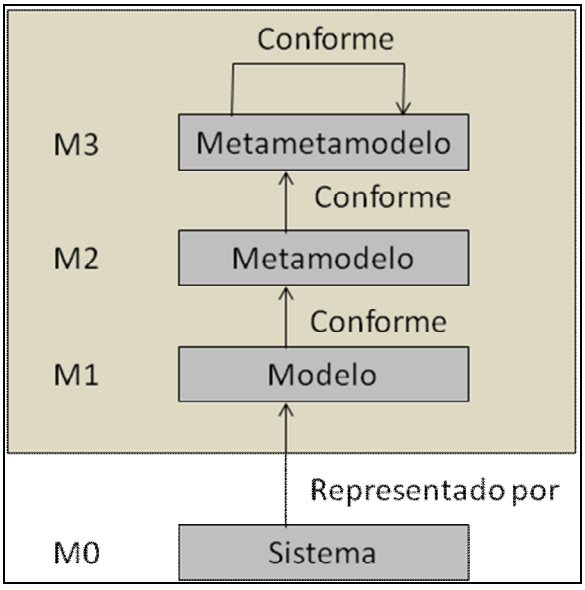

Figura 2-14: Processo de desenvolvimento MDA

O nível M2 define o metamodelo de uma linguagem usando a linguagem de metamodelagem especificada no nível M3. Quando um metamodelo é construído, é definida uma linguagem de modelagem que escreve modelos do nível M1. Por exemplo, o nível M3 usa o MOF, o nível M2 define o metamodelo UML e o nível M1 tem modelos específicos segundo o metamodelo do nível M2, ou seja, modelo UML de um sistema. O nível M0 são objetos reais de um problema conforme modelos do nível M1 (GARCÍA-MAGARIÑO *et al.*, 2009).

MDA fornece um framework que integra os padrões da OMG (Kent, 2002). Esses padrões compreendem linguagens de modelagem apropriadas para escrever PIMs e PSMs. Dentre esses padrões tem-se:

- MOF *(Meta Object Facility*): linguagem que define as demais linguagens permitindo que diversas ferramentas possam ler e escrever todas as linguagens padronizadas pela OMG. É a linguagem que está definida no nível M3 e que serve de base para a definição dos metamodelos do nível M2. O MOF é um padrão orientado a objetos que permite a definição de classes com atributos e relacionamentos;
- OCL *(Object Constraint Language*): linguagem de consulta e expressão para os modelos MOF e, consequentemente, UML. Com o uso da OCL, modelos mais precisos e extensíveis podem ser criados;
- QVT (*Query, Views and Transformations*): foi definida como uma linguagem padrão para escrever a definição de transformações entre modelos escritos em MOF. Permite criar *views* e fazer consultas em um modelo.

2.7 DSL

Uma linguagem específica de domínio ou *Domain Specific Language* (em inglês, DSL) é definida como uma linguagem computacional que é utilizada em um domínio particular, com o intuito de realizar tarefas específicas (FOWLER, 2010); diferentemente das linguagens de propósito geral como, por exemplo, Java. DSLs são usualmente declarativas, e com enfoque apenas em um domínio de problema em particular (DEURSEN *et al.*, 2000). As DSLs são utilizadas no MDD para facilitar a modelagem e as transformações, o fato de serem linguagens restritas a um domínio facilita a geração do código proposto pelo MDD (DJUKIC *et al*., 2011).

O propósito da DSL é simplificar o desenvolvimento de aplicações especializadas em um problema específico. Mas, exige maior compreensão sobre o domínio.

As vantagens da DSL são as abstrações específicas de um domínio que são pré-definidas e representam diretamente os conceitos do domínio da aplicação, aumentam o nível de abstração, geram códigos concisos, preparam o código para o reúso, geram

documentação suficiente e proporcionam o entendimento do sistema aos *stakeholders*, aos especialistas do domínio e aos desenvolvedores.

As desvantagens estão na necessidade de ter na equipe algum especialista no negócio do domínio e de reunir e definir o conhecimento relevante, ou seja, os especialistas devem possuir maturidade nos problemas de domínio.

DSL é composta por duas partes: a sintaxe abstrata e a sintaxe concreta. A sintaxe abstrata é a definição das regras de negócio do domínio, os elementos e os relacionamentos entre eles. Essa parte pode ser definida por meio de um metamodelo a partir das regras de negócio do domínio. A sintaxe concreta é a utilização da DSL construída. DSL pode ter notação gráfica, em forma de árvores ou textual. (RUMPE *et al*., 2010; CUADRADO e MOLINA, 2009). A utilização da DSL é determinada pela ferramenta escolhida para fazer o desenvolvimento da DSL.

As transformações são realizadas nos modelos criados durante a utilização da DSL desenvolvida com a intenção de gerar outros artefatos como código em alguma linguagem de programação (MAGALHÃES *et al*., 2013; RUMPE *et al.*, 2010). As ferramentas de transformação mais utilizadas são as que podem ser configuradas por meio de *templates*. Dessa forma, a ferramenta tem acesso às informações do modelo e as combinam com os *templates* gerando os artefatos (GRONBACK, 2009).

Um *template* é formado por partes fixas e por partes que podem variar (SARASA-CABEZUELO *et al*., 2012). As partes fixas são copiadas sem modificação do *template* para o artefato gerado e as partes que podem variar são formadas por comandos de linguagem de transformação que foram colocadas no *template* e que são dependentes das informações que são lidas do modelo especificado utilizando a DSL.

2.8 Ferramentas

A utilização de ferramentas que possuem os conceitos de MDD pode ser de grande valia para proporcionar o reúso de software de forma mais processual e mais independente da habilidade individual dos membros da equipe de desenvolvimento de software. Além disso, essas ferramentas podem possibilitar um desenvolvimento mais rápido, com menor custo e um software flexível, sendo possível realizar modificações mais rapidamente. Algumas ferramentas que atendem aos conceitos de MDD e que

podem ser utilizadas para a criação de DSL são: a) *Generic Modeling Environment* (GME) (GME, 2013), b) *DSL Tools plugin* para a plataforma *Visual Studio* da *Microsoft* (Visual, 2013) e c) *Eclipse Modeling Framework* (EMF) (EMF, 2013). Essas ferramentas permitem a criação de DSL gráficas.

Dentre as alternativas *open-source*, *Eclipse Modeling Framework* (EMF) apresenta a maior quantidade de ferramentas e de projetos relacionados à modelagem e às tecnologias MDD, dentre as quais várias implementam os padrões OMG, além de possibilitar a criação de editores gráficos específicos. Sua organização lógica define o desenvolvimento das duas partes que compõem uma DSL (a sintaxe abstrata e a sintaxe concreta) e a transformação de modelo para modelo e de modelo para texto.

EMF permite que quase todo o conjunto de ferramentas DSL seja desenvolvido, incluindo a definição de diagramas, as transformações, os *templates* para a geração de código, a serialização de modelos e a persistência (GRONBACK, 2009). Muitas dessas características são desenvolvidas por meio de modelos EMF. O *Graphical Modeling Framework* (GMF) usa vários modelos EMF na geração de ambientes de modelagem específicas de domínio.

EMF é uma ferramenta com alto potencial de uso para MDA, uma vez que possui relacionamento com os padrões MOF (*Meta Object Facility*), UML (*Unified Modeling Language*), XML (*Extensible Markup Language)* e XMI (*XML Metadata Interchange)*. Além disso, Eclipse provê uma plataforma de código aberto e implementações que servem de referência para muitas das especificações MDA.

No EMF é possível representar modelos de acordo com o seu metamodelo denominado Ecore. O metamodelo Ecore do EMF foi especificado para mapear modelos para implementação Java. EMF permite a criação de editores baseados nos modelos construídos. Existem bibliotecas e *plug-ins* que podem ser usados para gerar um conjunto de classes Java a partir dos metamodelos, tornando possível a visualização e a edição da linguagem de modelagem correspondente (GARCÍA-MARGARIÑO *et al*., 2009). Dessa forma, em um contexto no qual uma DSL é um metamodelo, Ecore é um metametamodelo. EMF utiliza um subconjunto de mapeamentos de metamodelo para Java, otimizado para manipulação somente em memória (ECLIPSE, 2006), como esquematizado na Figura 2-15. Essa abordagem o deixa menos abrangente se comparado com a combinação JMI/MOF (*Java Metadata Interface* / *Meta Object Facility*)

(MOORE *et al*., 2004). Por outro lado, EMF gera todo o código de acesso ao metamodelo e o coloca em memória, deixando-o rápido.

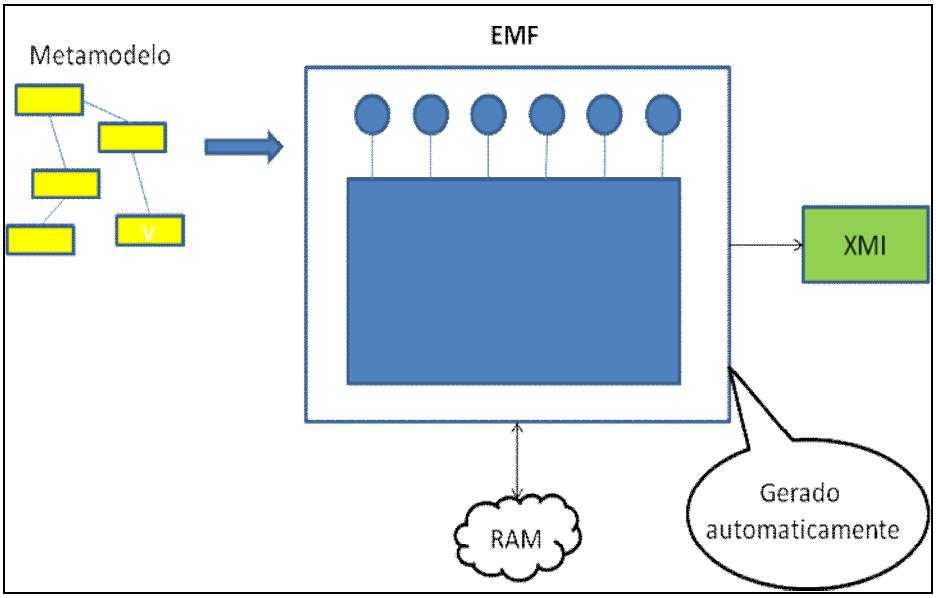

Figura 2-15: Eclipse Modeling Framework

Na plataforma Eclipse, há um projeto chamado GMT (*Generative Modeling Tools*) que serve de incubadora para projetos de ferramentas, de linguagens e de *frameworks* relacionados ao desenvolvimento orientado a modelos. Dentre esses projetos, o M2M (modelo para modelo) fornece um *framework* para linguagens de transformações de modelo para modelo, com dois componentes:

- QVT (*Query/View/Transformation*): um padrão para realizar transformações entre modelos nos quais as linguagens são definidas usando MOF;
- ATL (*Atlas Transformation Language*): linguagem composta por construções declarativas e imperativas que especificam um conjunto de regras que realizam transformações entre modelos.

Com relação às transformações de modelo para texto, no projeto M2T, existem cinco componentes principais:

- JET (*Java Emitter Templates*): mecanismo de geração de código baseado em *templates* (CLEAVELAND, 1988). O *template* é composto por código Java e por marcações (*tags*) que implementam comandos condicionais, de laços, e de formatação entre outras funções. Com isso, é possível ter a geração de

arquivos Java. JET pode ser usado com os modelos EMF sendo, portanto, possível utilizá-lo para gerar código para uma DSL;

- *Xpand*: permite representações XML de transformações similares a *scripts* Ant (ANT, 2009), encadeadas em um *workflow*. Possui uma sintaxe própria baseada na linguagem *Xtend* (XTEND, 2009) e expressões de outras linguagens para completar a sintaxe e a semântica;
- *Acceleo*: implementação do padrão OMG MOF *Model to Text Language* (MTL) (OMG, 2006), ou seja, é um mecanismo de geração de código baseado em *templates*. O *template* é composto por código e por marcações (*tags*) que implementam comandos condicionais, de laços, e de formatação entre outras funções. *Acceleo*, assim como JET, pode ser usado com os modelos EMF possibilitando a geração de código para uma DSL;
- M2T Core: *framework* que permite soluções M2T independentemente da linguagem;
- M2T *Shared*: infraestrutura de componentes compartilháveis entre diferentes linguagens M2T.

2.9 Considerações Finais

Neste capítulo foram apresentados os conceitos de áreas de gerenciamento, SNMP, redes de comunicação com enfoque em algumas das tecnologias que são gerenciadas de modo integrado pelo sistema de gerenciamento integrado de redes, MDD, MDA e DSL. Os elementos de rede que pertencem a uma tecnologia e que possuem uma MIB que utiliza SNMP são apropriados para serem gerenciados pelo sistema de gerenciamento integrado de redes. Para mostrar como é realizada a especificação de uma nova tecnologia no sistema de gerenciamento integrado de redes, utilizando a abordagem de especialização de classes e a DSL desenvolvida, foram selecionadas duas tecnologias que estão aptas para este fim. Atualmente, em uma empresa real, a especificação de uma nova tecnologia no sistema é realizada utilizando especialização de classes. Essa abordagem, baseada em código-fonte, aplicada em um sistema que precisa atender rapidamente a constantes mudanças, dificulta a reutilização do software, que está cada vez mais complexo exigindo um maior esforço para adicionar novas funções e

realizar manutenção. Neste cenário, MDD surge como uma alternativa para amenizar esses problemas relacionados com o processo de desenvolvimento do software.

MDD enfatiza a importância de tornar os modelos de níveis mais altos de abstração, e que representam o negócio do domínio, parte integrante do desenvolvimento do software. Com a utilização de técnicas de modelagem e a geração automática de código juntamente com ferramentas apropriadas, a complexidade da implementação do código fica não perceptível aos desenvolvedores, que podem se dedicar exclusivamente ao problema do domínio. Com o uso de MDA é possível definir classes de modelos e as transformações entre elas até se chegar ao código-fonte utilizando linguagens de modelagem apropriadas para escrevê-las. A concretização dos conceitos de MDD é obtida por meio do desenvolvimento de uma DSL, utilizando ferramentas que possuem melhores características para desenvolvê-la e que realizam as transformações de modelo para código por meio de *templates*.

A partir dos conceitos de MDD, MDA, DSL e de ferramentas apropriadas é possível analisar a gerência de configuração do sistema de gerenciamento integrado de redes e desenvolver uma DSL para realizar a especificação de uma nova tecnologia de rede a ser incorporada no sistema. O objetivo da utilização desta DSL em um exemplo real é avaliar seus benefícios, como proteger os desenvolvedores da complexidade da plataforma de implementação, proporcionar reúso de software a partir dos modelos de alto nível de abstração, possibilitar um desenvolvimento mais rápido e produzir um software mais flexível.

No Capítulo 3 serão apresentadas as funções comuns e as específicas e dependentes da tecnologia de rede de um sistema de gerenciamento integrado de redes desenvolvido por uma empresa real. Também será descrito o processo de desenvolvimento adotado atualmente por essa empresa para esse sistema.

Capítulo 3

DESENVOLVIMENTO DE SISTEMA DE GERENCIAMENTO INTEGRADO DE REDES

3.1 Considerações Iniciais

O sistema de gerenciamento integrado de redes é composto por várias funções que são comuns aos diversos objetos gerenciados e outras que são específicas e dependem da tecnologia de rede. As funções comuns, independentes de tecnologia de rede, se concentram na organização de agrupamentos de elementos de redes, na exibição para o usuário do posicionamento desses elementos tanto geograficamente quanto topologicamente, e na manutenção de um cadastro de endereços IP. Os endereços IP podem ser informados individualmente ou em intervalos de endereços e são insumos para o sistema realizar o processo de descoberta de elementos de rede. As funções específicas da gerência de configuração são aquelas que dependem diretamente da tecnologia de rede. Elas se concentram na descoberta dos elementos que pertencem a uma determinada tecnologia de rede e no cadastramento do tipo do elemento de rede, intrínseco à tecnologia.

A proposta deste projeto é implementar os conceitos de MDD com o desenvolvimento de uma DSL para especificar uma nova tecnologia de rede na gerência de configuração do sistema de gerenciamento integrado de redes, ou seja, desenvolver uma DSL sobre um conjunto de funções específicas de um sistema real. Dessa forma, neste capítulo são apresentados o sistema de gerenciamento integrado de redes e o

processo de desenvolvimento de software realizado em uma empresa real, com destaque para a especificação de uma nova tecnologia de rede no sistema.

Na Seção 3.2 são apresentadas as funções do sistema de gerenciamento integrado de redes, tanto as comuns quanto as específicas e dependentes da tecnologia de rede. Na Seção 3.3 são ilustrados a forma de como é realizado o desenvolvimento deste sistema em uma empresa real e o conjunto de funções sobre as quais são aplicados os conceitos de MDD apresentados no Capítulo 2. Na Seção 3.4 estão apresentadas as considerações finais.

3.2 Sistema de Gerenciamento Integrado de Redes

Um sistema de gerenciamento integrado de redes é composto por cinco áreas de gerenciamento: gerência de configuração, gerência de falhas, gerência de desempenho, gerência de contabilização e gerência de segurança do sistema (SORTICA, 1999). Porém, o sistema de gerenciamento integrado de redes desenvolvido por uma empresa real contempla atualmente três dessas gerências, a saber: gerência de configuração, gerência de falhas e gerência de segurança. Dessa forma, somente as funções existentes no sistema para essas três gerências serão descritas nesse projeto.

A gerência de configuração realiza a descoberta dos elementos de rede pertencentes às tecnologias conhecidas pelo sistema. A rede pode ser IP, WiMax, WiFi, GPON (*Gigabit-capable Passive Optical Networks*), LTE, WDM (*Wavelength-Division Multiplexing)* entre outras. A partir do momento da sua descoberta, o elemento de rede passa a ser gerenciado. A gerência de falhas trata os eventos recebidos dos elementos de redes gerenciados e, para os eventos que são classificados como alarmes, são aplicadas as regras de correlação a fim de descobrir, de forma mais assertiva, a causa raiz do problema na rede de telecomunicações. Dessa forma, o processo de abertura de ordem de serviço para os técnicos de campo sanarem o problema e deixar a rede operacional e disponível novamente se torna mais eficaz. O diferencial do sistema de gerenciamento integrado de redes desenvolvido pela empresa real é a função de correlação de alarmes, pois a correlação ocorre em nível de elemento, de rede e de infra-estrutura, independentemente da tecnologia do elemento de rede. A gerência de segurança trata do controle de acesso dos perfis dos usuários do sistema. Na Figura 3-1

está representado o sistema de gerenciamento integrado de redes enfatizando o gerenciamento de múltiplas tecnologias de redes.

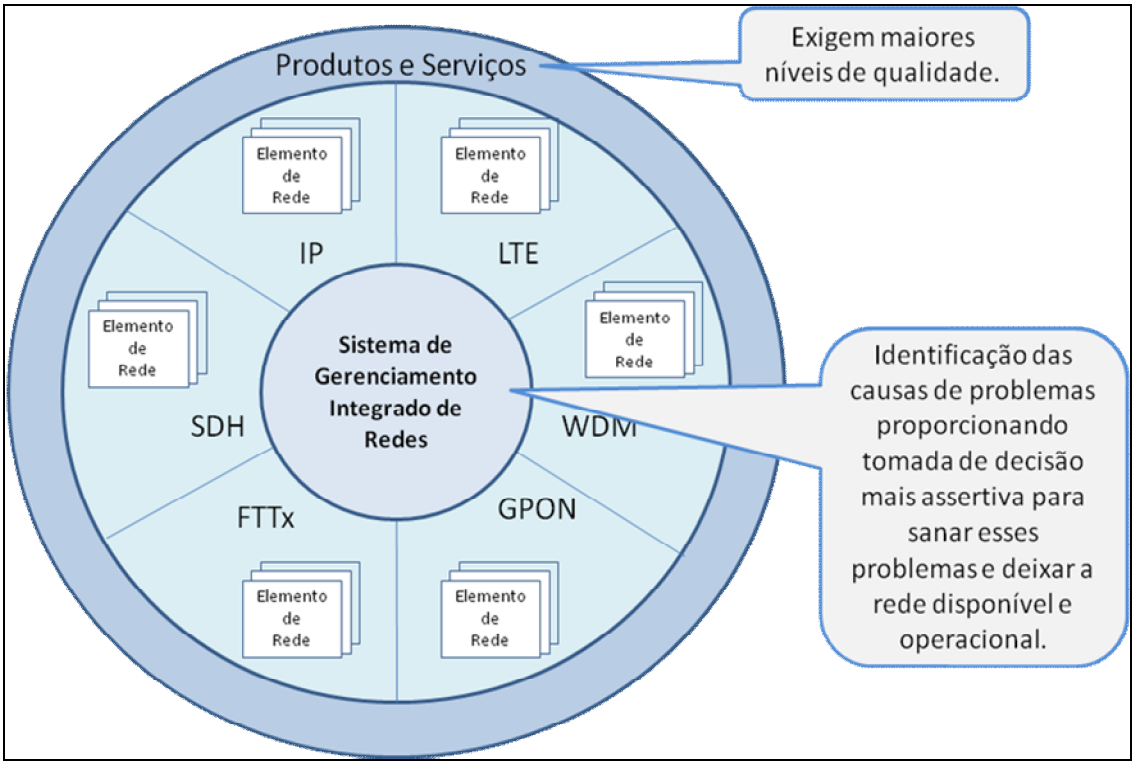

Figura 3-1: Sistema de gerenciamento integrado de redes enfatizando o gerenciamento de múltiplas tecnologias de redes

Para as áreas de gerência de configuração e de gerência de falha, além das funções comuns, existem as funções específicas das tecnologias de rede que podem ser gerenciadas. Dessa forma, para desenvolver o sistema de gerenciamento integrado de redes, é necessário ter conhecimento específico das diversas tecnologias às quais pertencem os elementos de rede.

Na Figura 3-2 estão ilustradas esquematicamente as funções comuns e as específicas das áreas de gerência do sistema de gerenciamento integrado de redes de uma empresa real.

Nas Seções 3.2.1, 3.2.2 e 3.2.3 estão brevemente descritas as funções comuns e as específicas da gerência de configuração, da gerência de falhas e da gerência de segurança, respectivamente.

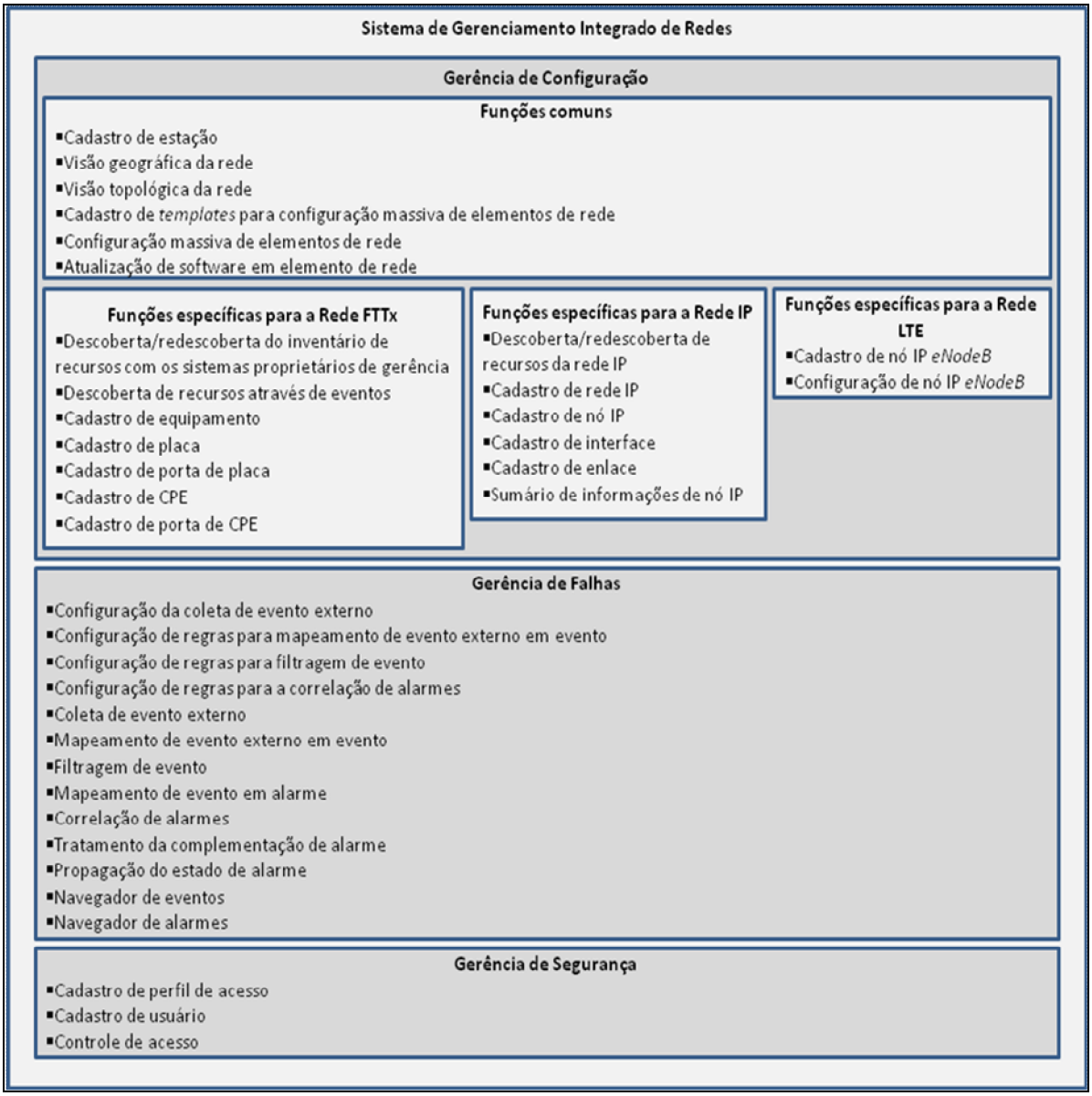

Figura 3-2: Sistema de gerenciamento integrado de redes enfatizando as funções comuns e específicas das áreas de gerência

3.2.1 Gerência de Configuração

A gerência de configuração deve conhecer a tecnologia e a topologia da rede na qual os equipamentos estarão contidos, como será feita a descoberta dos elementos de rede e como esses elementos estarão agrupados. Dessa forma, as funções da gerência de configuração comuns a qualquer tecnologia de rede são:

- **Cadastro de estação**: informações quanto à localização dos equipamentos que compõem a rede gerenciada;
- **Visão geográfica da rede**: visualização gráfica das estações, e hierarquicamente dos seus equipamentos, em suas respectivas localizações geo-referenciadas em mapa;
- **Visão topológica da rede:** visualização topológica dos elementos de rede bem como a interconexão entre eles;
- **Cadastro de *templates* para configuração massiva de elementos de rede:** definição de parâmetros específicos para cada tipo de tecnologia que permitirá a descoberta massiva dos seus elementos de rede;
- **Configuração massiva de elementos de rede:** por meio da seleção de um *template* de configuração massiva, cujo cadastro deve ter sido realizado previamente, o usuário deve fornecer a identificação de um ou mais elementos de rede que serão submetidos à configuração e, a seguir, solicitar a operação de configuração. Os elementos de rede fornecidos devem ser da mesma tecnologia. O sistema, por meio do protocolo de gerência, envia a configuração definida no *template* para cada um dos elementos de rede fornecidos, realizando as ações necessárias e as específicas ao tipo de elemento de rede envolvido na configuração. Após o término da operação de configuração massiva dos elementos de rede, o sistema de gerência atualiza sua própria base de dados com os elementos de rede cuja execução da operação foi bem-sucedida a partir dos valores dos atributos definidos no *template*. O usuário pode consultar o resultado da operação de configuração em cada elemento submetido à configuração massiva;
- **Atualização de software em elemento de rede:** o usuário solicita a atualização (*upgrade*) de software em elementos da rede gerenciada. O usuário deve fornecer o arquivo que contém a imagem do software e solicitar a atualização do software no elemento de rede.

Para que seja possível o gerenciamento integrado de redes com diferentes tecnologias, se faz necessário ter funções de configuração com as particularidades de cada uma das tecnologias de rede. Essas funções estão descritas brevemente a seguir.

3.2.1.1 Gerência de Configuração para Rede IP

As funções específicas para a tecnologia de rede IP para a gerência de configuração estão apresentadas a seguir:

- **Descoberta/redescoberta de recursos da rede IP:** o sistema descobre os recursos da rede gerenciada e mantém sua base de dados de inventário atualizada. O usuário pode configurar a descoberta/redescoberta com relação à frequência de atualização da base de inventário, ao escopo dos recursos da rede que devem ser gerenciados e às restrições de resposta da rede às solicitações do sistema;
- **Cadastro de rede IP:** o sistema deve manter o cadastro dos equipamentos do inventário. Um equipamento pode estar contido em uma estação e pode ter várias placas. O sistema deve permitir ao usuário realizar as operações de consulta e de alteração que se limita em associar o equipamento a uma estação;
- **Cadastro de nó IP:** o sistema deve manter o cadastro das placas dos equipamentos do inventário. Uma placa deve obrigatoriamente estar contida em um equipamento. Uma placa pode ter nenhuma ou mais portas de placa;
- **Cadastro de interface:** o sistema deve manter o cadastro das portas das placas dos equipamentos do inventário. Uma porta de placa deve obrigatoriamente estar contida em uma placa. Uma porta de placa contém nenhum ou mais CPE;
- **Cadastro de enlace:** o sistema deve manter o cadastro dos CPEs associados às portas das placas dos equipamentos do inventário. Um CPE deve obrigatoriamente estar contido em uma porta de placa. Um CPE contém nenhum ou mais portas de CPE;
- **Sumário de informações de nó IP:** o sistema deve manter o cadastro das portas dos equipamentos CPE do inventário. Uma porta de CPE deve obrigatoriamente estar contida em um CPE.

3.2.1.2 Gerência de Configuração para Rede LTE

As funções específicas para a tecnologia de rede LTE para a gerência de configuração estão apresentadas a seguir:

- **Cadastro de nó IP *eNodeB*:** o sistema deve manter o cadastro dos nós IP do tipo *eNodeB* no inventário. Um nó IP *eNodeB* pode estar contido em uma estação. O sistema deve permitir ao usuário realizar as operações de consulta, de alteração (associar nó IP *eNodeB* a uma estação) e de exclusão. A inclusão no cadastro é realizada pela descoberta/redescoberta de recursos da rede IP;
- **Configuração de nó IP *eNodeB*:** o sistema deve configurar o nó IP do tipo *eNodeB*. O sistema deve permitir ao usuário definir os valores dos parâmetros de configuração e solicitar ao sistema o envio desses parâmetros ao equipamento *eNodeB*.

3.2.1.3 Gerência de Configuração para Rede FTTx

As funções específicas para a tecnologia de rede FTTx para a gerência de configuração estão apresentadas a seguir:

- **Descoberta/redescoberta do inventário de recursos com os sistemas proprietários de gerência:** o sistema deve manter sua base de dados de inventário de recursos físicos da rede atualizada e sincronizada com o inventário de recursos presentes nos sistemas proprietários de gerência (por exemplo, *racks* de equipamento, placa, CPE). A sincronização de inventário é realizada por meio das interfaces de mediação com os sistemas de gerência periodicamente e é necessária uma configuração com os seguintes parâmetros: identificação do sistema de gerência, informações de acesso (usuário e senha) ao sistema proprietário de gerência e periodicidade da redescoberta. A sincronização do inventário pode resultar em inclusão de um novo recurso na base de inventário do sistema ou alteração dos dados de um recurso já existente na base de inventário do sistema;
- **Descoberta de recursos por meio de eventos:** o sistema deve manter sua base de dados de inventário de recursos físicos da rede por meio da recepção de eventos externos, ou seja, o sistema deve extrair de um evento externo

recebido a identificação do recurso (por exemplo, *rack* de equipamento, placa e CPE). A descoberta de recursos por meio dos eventos resulta na inclusão de um novo recurso na base de inventário do sistema**;**

- **Cadastro de equipamento:** o sistema deve manter o cadastro dos equipamentos do inventário de recursos. Um equipamento está contido em uma estação. Um equipamento contém nenhuma ou mais placas. O sistema deve permitir ao usuário realizar as operações de consulta e alteração, limitado em associar equipamento a uma estação;
- **Cadastro de placa:** o sistema deve manter o cadastro das placas dos equipamentos do inventário de recursos. Uma placa deve obrigatoriamente estar contida em um e apenas um equipamento. Uma placa contém nenhuma ou mais portas de placa;
- **Cadastro de porta de placa:** o sistema deve manter o cadastro das portas das placas dos equipamentos do inventário de recursos. Uma porta de placa deve obrigatoriamente estar contida em uma placa. Uma porta de placa contém nenhum ou mais CPE;
- **Cadastro de CPE:** o sistema deve manter o cadastro dos CPEs associados às portas das placas dos equipamentos do inventário de recursos. Um CPE deve obrigatoriamente estar contido em uma porta de placa. Um CPE contém nenhuma ou mais portas de CPE;
- **Cadastro de porta de CPE:** o sistema deve manter o cadastro das portas dos equipamentos CPE do inventário de recursos. Uma porta de CPE deve obrigatoriamente estar contida em um CPE.

3.2.2 Gerência de Falhas

A gerência de falhas contempla as funções desde a coleta dos eventos externos até o tratamento e a apresentação das informações de falhas ao usuário. A fim de evitar confusão no uso de termos utilizados nesta gerência, seguem-se algumas definições:

- **Evento externo** – é uma mensagem que tem origem externamente e é enviada ao sistema, informando ou a ativação ou a normalização ou a mudança de estado de um alarme. As notificações de evento externo podem ser transportadas em diferentes protocolos e em diferentes formatos;

- **Evento** – é uma notificação de evento externo que foi coletada e formatada pelo sistema;
- **Alarme** – é o registro de alarme gerado a partir de um evento, conforme condições estabelecidas para essa geração.

Na Figura 3-3 está apresentado o fluxo do tratamento das notificações de eventos externos recebidos pelo sistema.

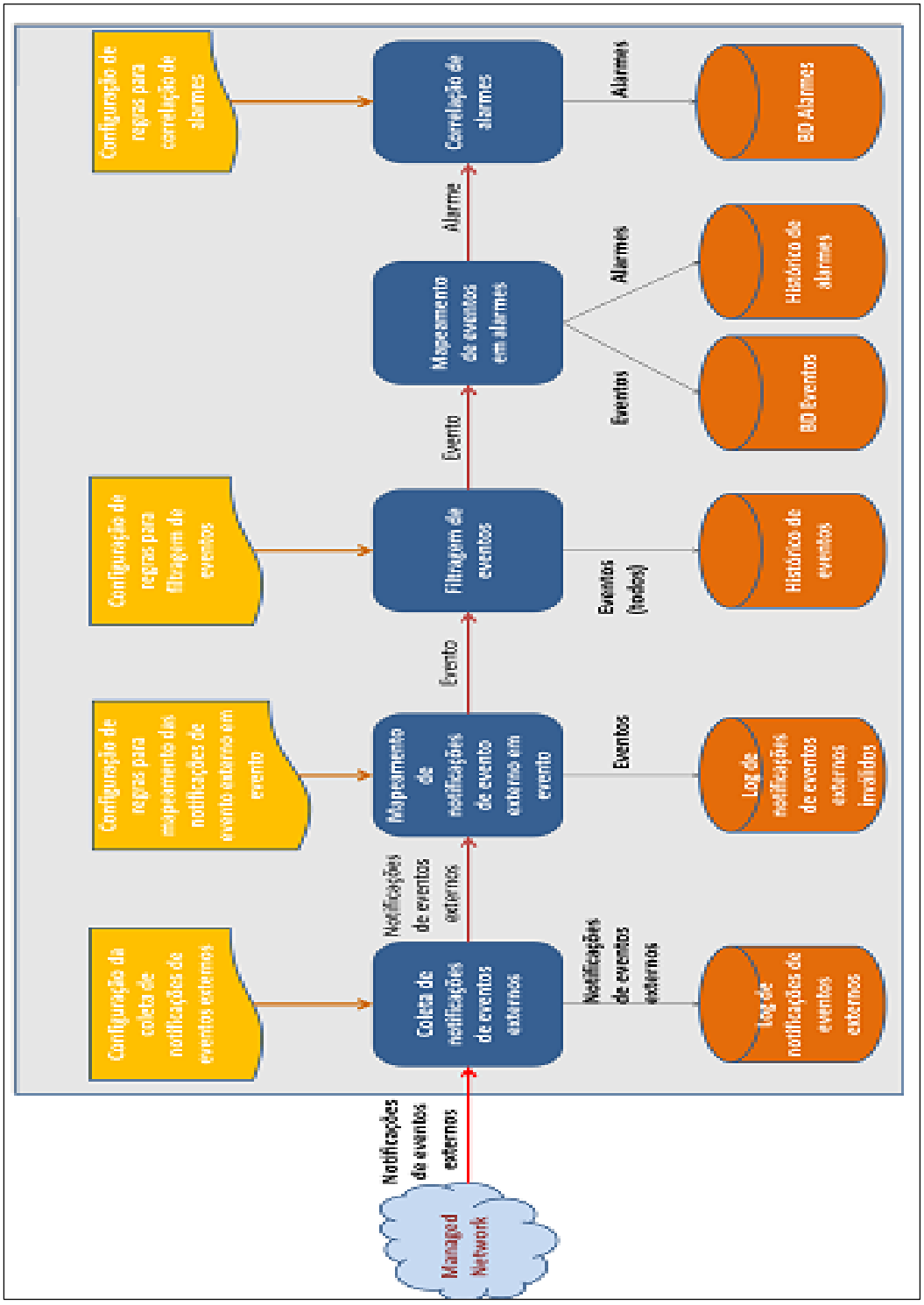

Figura 3-3: Fluxo de eventos externos recebidos pelo sistema

As funções da gerência de falhas estão apresentadas a seguir:

- **Configuração da coleta de evento externo:** permite a configuração dos parâmetros necessários para a coleta de eventos externos, tais como, estado da coleta de notificações (habilitado/desabilitado), porta de recepção das

notificações, critérios para filtrar as notificações de eventos externos coletadas e critérios para armazenar em *log* as notificações de eventos externos coletadas;

- **Configuração de regras para mapeamento de evento externo em evento:** permite a configuração das regras de conversão das mensagens de eventos externos em eventos "internalizados" no sistema para diferentes tecnologias de diferentes fornecedores;
- **Configuração de regras para filtragem de evento:** permite a configuração de regras de filtragem dos eventos, que deve atuar antes da aplicação das regras de correlação. Seu propósito é descartar eventos com características que não agregam valor à correlação, tomando por base os critérios definidos para filtragem dos eventos, seja porque simplesmente não devem ser tratados pelo sistema ou por quê são inválidos, duplicados, intermitentes ou não persistentes. Os eventos retidos pelos critérios de filtragem deverão ser registrados no *log* de eventos filtrados, com a informação do critério de filtragem aplicado;
- **Configuração de regras para a correlação de alarmes:** permite a configuração de regras de correlação de alarmes, cujo objetivo é encontrar a causa raiz de um conjunto de alarmes;
- **Coleta de evento externo:** o sistema realiza a coleta das notificações de evento externo de acordo com a configuração da coleta. De acordo com essa configuração, o sistema poderá descartar notificações de eventos externos que foram filtrados e armazenar em *log* as notificações de eventos externos coletados em seus formatos originais. As notificações de eventos externos que não forem descartados serão encaminhadas para tratamento pelo sistema. As notificações de eventos externos, provenientes dos sistemas proprietários de gerência, serão recebidas por meio das interfaces de mediação de falhas específicas para cada sistema proprietário;
- **Mapeamento de evento externo em evento:** após a coleta das notificações de eventos externos, o sistema deve convertê-los para um formato interno ao sistema, de modo a permitir um tratamento unificado e consistente dos mesmos, conforme configuração das regras de mapeamento. O sistema deve também aplicar um filtro de "validade" às notificações de eventos externos, ou seja, caso não haja uma regra de mapeamento aplicável a uma determinada notificação de evento externo, o sistema não realizará o mapeamento e deverá

armazenar a notificação em um *log* de notificações de eventos externos inválidos;

- **Filtragem de evento:** o sistema possui a função de filtragem dos eventos "internalizados", que deve atuar antes da aplicação das regras de correlação. Seu propósito é descartar eventos com características que não agregam valor à operação do sistema ou à correlação, tomando por base os critérios definidos para filtragem dos eventos. Todos os eventos tratados pela filtragem devem ser armazenados em base de dados, no histórico de eventos. O evento retido por critério de filtragem deverá ser armazenado no histórico com a informação do critério de filtragem aplicado;
- **Mapeamento de evento em alarme:** todos os eventos que não foram retidos pelos critérios de filtragem são classificados como alarmes, armazenados no histórico de alarmes e encaminhados para a correlação de alarmes;
- **Correlação de alarmes:** o sistema correlaciona um conjunto de alarmes usando as regras de correlação a fim de identificar, com maior precisão, a causa raiz do problema das redes supervisionadas pelos sistemas proprietários de gerência. Após a correlação, o sistema classifica os alarmes em causa raiz e consequência;
- **Tratamento da complementação de alarme:** depois das etapas de coleta e de correlação, o sistema realiza o tratamento para complementação de informações e eventual encaminhamento do alarme a um sistema de *Trouble Ticket* (TT). Esse tratamento pode ser a obtenção de medição da fibra óptica envolvida nos recursos afetados pelo alarme e/ou a obtenção dos serviços afetados pelo alarme e/ou a obtenção de elemento comum aos recursos afetados pelo alarme e/ou a abertura de notificação no sistema de TT. A aplicabilidade desses tratamentos dependerá de uma combinação de regras de negócio e do tipo do alarme envolvido;
- **Propagação do estado de alarme:** o sistema atualiza o estado de alarme de um recurso e faz a propagação desse estado para os recursos de nível de hierarquia superior;
- **Navegador de eventos:** o sistema permite ao usuário consultar os eventos que estão na base de eventos do sistema;

- **Navegador de alarmes:** o sistema permite ao usuário consultar os alarmes que estão na base de alarmes do sistema. O sistema permite ao usuário realizar em um determinado alarme, consulta detalhada das informações do alarme, exclusão do alarme, atuação no registro do alarme (ex.: normalização, reconhecimento etc.) e correlacionar manualmente os alarmes.

3.2.3 Gerência de Segurança

A gerência de segurança administra o acesso aos recursos de rede e às determinadas informações, incluindo tarefas como: verificar o privilégio de acesso à rede dos usuários, detectar e registrar tentativas de acesso não autorizadas. As funções compreendidas na gerência de falhas estão apresentadas a seguir:

- **Cadastro de perfil de acesso:** o sistema permite realizar o cadastro de perfil de acesso, que consiste na configuração das autorizações de acesso dentro de uma sessão de usuário;
- **Cadastro de usuário:** o sistema permite realizar o cadastro de usuários, no qual são configurados os dados de usuários, *login* e senha e a associação aos perfis de acesso;
- **Controle de acesso:** o sistema valida o acesso de um usuário por meio de *login* e senha e permitir o acesso somente às funções do perfil associado a ele. O sistema bloqueia o acesso de usuários ao sistema quando for atingido o limite da quantidade de usuários simultâneos estabelecidos pela licença de software do sistema.

3.3 Desenvolvimento do Sistema de Gerenciamento Integrado de Redes de uma Empresa Real

Em uma empresa real, o desenvolvimento do sistema de gerenciamento integrado de redes é realizado seguindo um Processo Unificado (PU). Esse processo é usado para o desenvolvimento de software visando à construção de sistemas orientados a objetos e combina os ciclos iterativo e incremental. As quatro fases do Processo Unificado,

concepção, elaboração, construção e transição, são divididas em várias iterações, sendo que cada iteração resulta em um incremento, que é uma versão do sistema que contém funções adicionais ou melhoradas em comparação com a versão anterior. As características do Processo Unificado são: dirigido por casos de uso, centrado na arquitetura e focado nos riscos. Os casos de uso são usados para obter os requisitos funcionais e refinar o conteúdo das iterações. A arquitetura deve ser o centro dos esforços da equipe do projeto para que o sistema seja obtido. Uma das entregas mais importantes desse processo é a arquitetura executável, criada durante a fase de Elaboração, e consiste na implementação parcial do sistema que tem por objetivo validar a arquitetura e servir como base para o desenvolvimento do sistema. Os riscos mais críticos são encontrados no início do ciclo de vida do projeto. A Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML) é usada na elaboração de todos os artefatos do sistema (JACOBSON, 1999).

A empresa em questão é certificada em CMMI (*Capability Maturity Model Integration*) nível 3 (CMMi, 2009) e o processo de software definido é seguido para garantir a qualidade do produto final.

O escopo do sistema de gerenciamento integrado de redes que será tratado neste projeto de mestrado é o da gerência de configuração, precisamente quando ocorre a necessidade de se acrescentar ao sistema a definição de uma nova tecnologia de rede com os novos elementos de rede que devem ser descobertos e, a partir de então, gerenciados.

O sistema de gerenciamento integrado de redes é composto por um processo de descoberta de elementos de rede. Esse processo atua sobre um intervalo de endereços IP definido pelo usuário, verifica se o elemento encontrado no endereço IP é de uma das tecnologias que o sistema deve gerenciar e recupera os valores de algumas variáveis da MIB (*Management Information Base*) desse elemento. As variáveis da MIB, cujos valores devem ser recuperados, foram definidas pelo especialista de rede. A MIB é um banco de dados usado para gerenciar as entidades em uma rede de comunicações. Na maioria das vezes, esse banco está associado ao SNMP (*Simple Network Management Protocol*).

Em razão da grande quantidade de variáveis, essas são organizadas em uma hierarquia que pode ser descrita como uma árvore, sendo que a raiz não tem nome, mas os níveis abaixo da raiz têm diferentes organizações. Alguns órgãos de padronização internacional têm suas referências logo abaixo da raiz e cada nó da árvore possui um

rótulo com uma descrição textual e um número. Os objetos da árvore que não são folhas agregam vários outros objetos relacionados, sendo que esses descrevem as informações mantidas nos agentes SNMP (agentes são os elementos gerenciados). Nesse contexto, uma variável da MIB é a instância de um objeto da árvore, que realmente pode ser manipulada pelo protocolo SNMP.

Um objeto é identificado por um caminho disposto entre a raiz e o objeto gerenciado. Existem dois tipos de objetos gerenciados: os escalares, que definem uma única instância do objeto e os tabulares, que definem várias instâncias de objetos relacionados que são agrupados em tabelas MIB. Um identificador de objeto (ou ID do objeto ou OID) identifica unicamente um objeto gerenciado na hierarquia da MIB (KUROSE, 2010). Um exemplo de MIB está ilustrado na Figura 3-4.

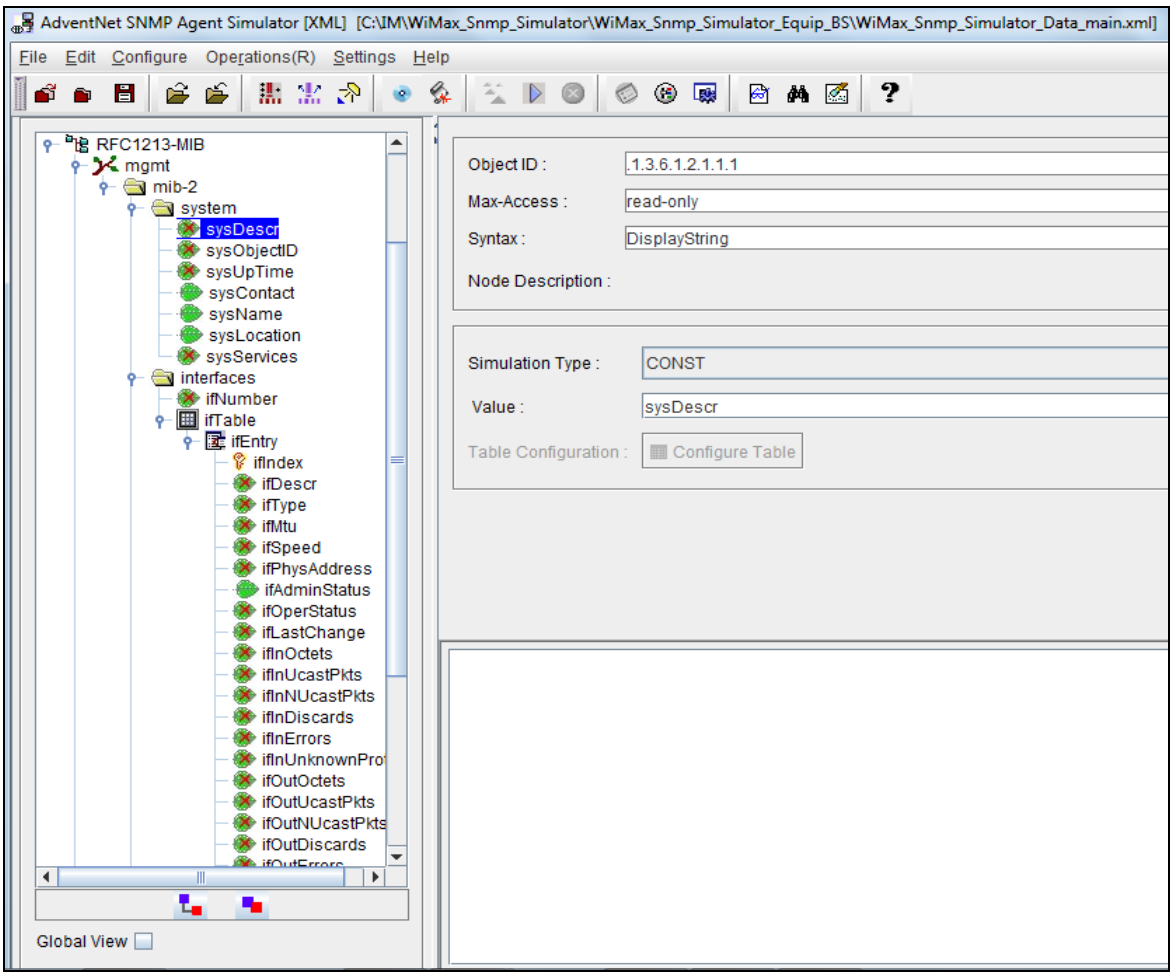

Figura 3-4: Exemplo de uma Base de Informação de Gerenciamento (MIB)

A alteração no sistema, para realizar a descoberta de um novo elemento de rede pertencente a uma nova tecnologia, por exemplo, a tecnologia Cisco, compreende a implementação de duas classes Java: *DiscoveryCiscoSession* e *DiscoveryCisco.* A classe *DiscoveryCiscoSession* é responsável por registrar a nova tecnologia na lista de tecnologias gerenciadas pelo sistema e a classe *DiscoveryCisco* é responsável por definir os atributos dos elementos de rede da nova tecnologia que deverão ser gerenciados pelo sistema. Nessa classe são encapsuladas as particularidades da nova tecnologia a ser gerenciada pelo sistema. Geralmente, o protocolo de comunicação usado para a descoberta é o SNMP, sendo que o equipamento deve ter uma MIB da qual são recuperadas as informações de gerenciamento.

Na Figura 3-5 está ilustrado o diagrama de classes que representa as classes usadas para a descoberta dos elementos de rede da tecnologia Cisco.

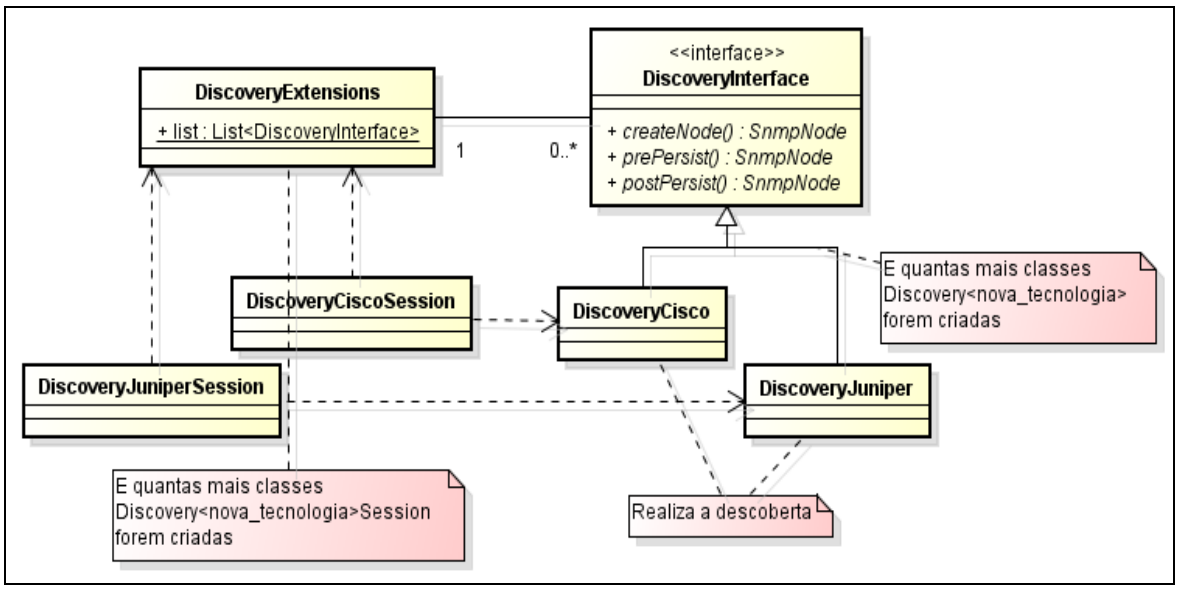

Figura 3-5: Diagrama de classes para a descoberta de elementos de rede da tecnologia Cisco

A classe *DiscoveryInterface* representa uma interface que contém os métodos que devem ser implementados nas classes derivadas, as que contém a especialização das tecnologias, ou seja, suas particularidades. A classe *DiscoveryExtensions* é responsável por manter uma lista de tecnologias que são gerenciadas pelo sistema.

Na Figura 3-6 estão representadas as classes responsáveis que auxiliam na descoberta de um nó na rede, por meio de protocolo SNMP, para todas as tecnologias presentes na lista de tecnologias gerenciadas pelo sistema.

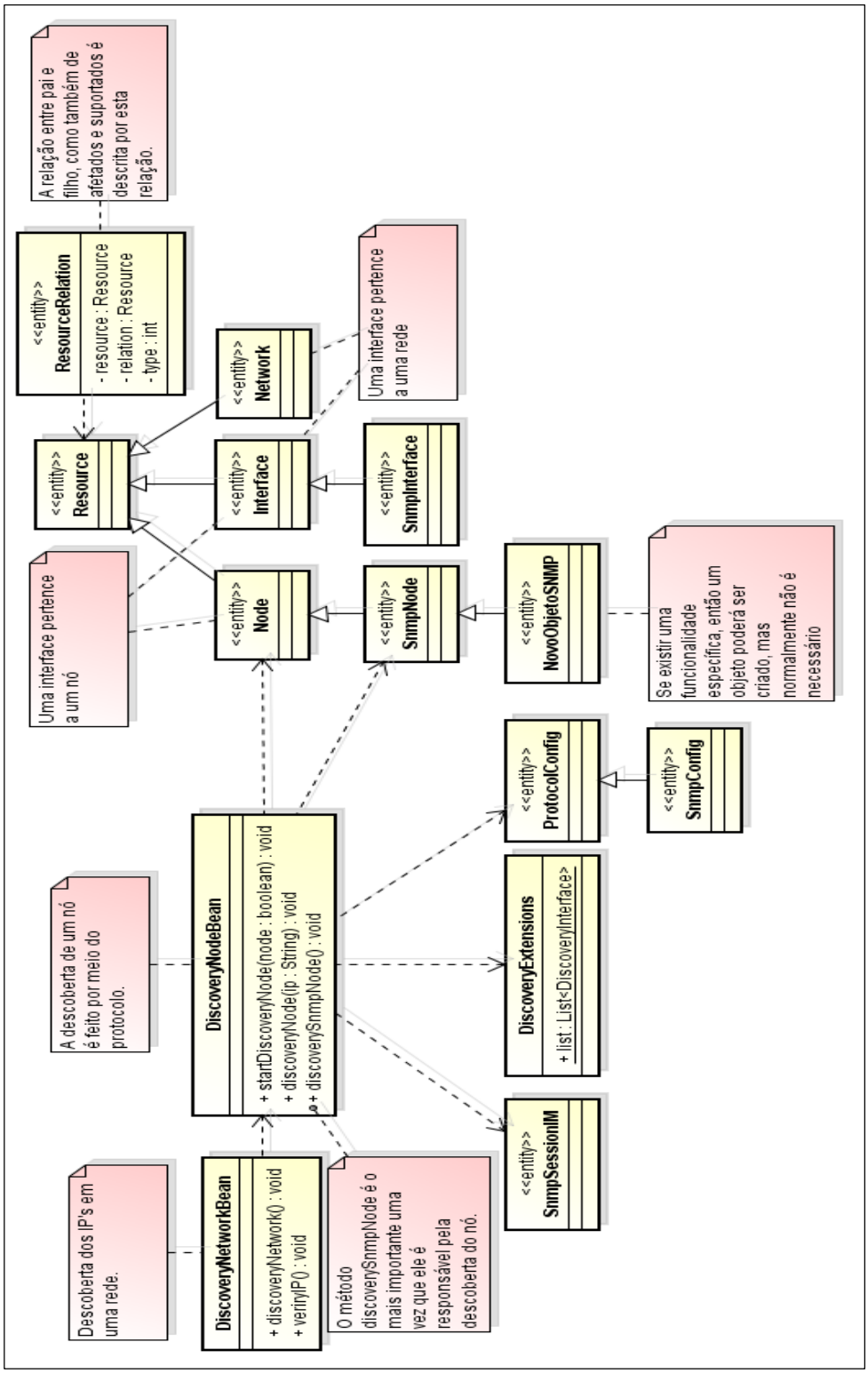

Figura 3-6: Diagrama de classes para a descoberta dos nós SNMP

O código fonte Java para a implementação da classe *DiscoveryCiscoSession.java* é o exibido no Quadro 3-1 e o da classe *DiscoveryCisco.java* é exibido no Quadro 3-2. Nos dois casos a tecnologia Cisco é a escolhida.

Quadro 3-1: *DiscoveryCiscoSession.java*

```
1. package com.gi.discovery.cisco;
2. import java.util.logging.Logger;
3. import javax.annotation.PostConstruct;
4. import javax.ejb.Singleton;
5. import javax.ejb.Startup;
6. import com.gi.discovery.DiscoveryExtension;
7. @Singleton
8. @Startup
9.  public class DiscoveryCiscoSession {
10.      private DiscoveryCisco instance;
11.      private Logger log = Logger.getLogger("com.gi.discovery.cisco");
12.      @PostConstruct
13.      public final void init() {
14.         log.info("Descoberta Cisco registrada");
15.         instance = new DiscoveryCisco();
16.         DiscoveryExtension.list.add(instance);
17.      }
18.      }
```

Quadro 3-2: *DiscoveryCisco.java*

```
1. package com.gi.discovery.cisco;
2. import java.util.List;
3. import java.util.Properties;
4. import java.util.logging.Logger;
5. import com.gi.discovery.DiscoveryInterface;
6. import com.gi.discovery.NodeExtension;
7. import com.gi.discovery.NodeExtensionDAO;
8. import com.gi.discovery.NodeExtensionField;
9. import com.gi.discovery.NodeExtensionTable;
10.      import com.gi.entity.ip.SnmpConfig;
11.      import com.gi.entity.ip.SnmpNode;
12.      import com.gi.snmp.SnmpSessionIM;
13.      import com.gi.snmp.TableValues;
14.      public class DiscoveryCisco implements DiscoveryInterface {
```

```
15.      /** Logger. */
16.      private Logger log = Logger.getLogger("com.gi.discovery.cisco");
17.      @Override
18.      public String getExtensionName() {
19.      return "CiscoEx";
20.      }
21.      @Override
22.      public SnmpNode createNode(SnmpNode dbNode, SnmpSessionIM session,
   String ipV4, String hostName, String dnsName, String ipV6, SnmpConfig sc,
   String name, String ipForwarding, String numPorts, String sysObjectID) {
23.      if (dbNode != null) {
24.      return dbNode;
25.      }
26.      if (sysObjectID != null &&
   sysObjectID.equals(".1.3.6.1.4.1.13727.2300.1.1.1")) {
27.      return new SnmpNode();
28.      }
29.      return null;
30.      }
31.      @Override
32.      public SnmpNode prePersist(SnmpNode node, SnmpSessionIM session,
   String name, String ipForwarding, String numPorts,
33.      String sysObjectID) {
34.      if (sysObjectID != null &&
   sysObjectID.equals(".1.3.6.1.4.1.13727.2300.1.1.1")) {
35.      node.setModel("WiMax BS");
36.      node.setSubType("WiMax BS");
37.      }
38.      return null;
39.      }
40.      @Override
41.      public SnmpNode postPersist(SnmpNode dbNode, SnmpSessionIM session,
   String name, String ipForwarding, String numPorts, String sysObjectID) {
42.      if (sysObjectID != null &&
   !sysObjectID.equals(".1.3.6.1.4.1.13727.2300.1.1.1")) {
43.      return null;
44.      }
45.      NodeExtensionField extField = null;
46.      try {
47.      NodeExtension ne = new NodeExtension();
48.      ne.setNode(dbNode.getResource());
```

```
49.      ne.setNome(getExtensionName());
50.      extField = new NodeExtensionField();
51.      extField.setFiledSetName("wlOlsr");
52.      extField.addFieldValue("wlOlsrIpGateway",
   session.getOID(".1.3.6.1.4.1.13727.2300.2.1.1.2.1.1.0"));
53.      extField.addFieldValue("wlOlsrInternetGateway",
   session.getOID(".1.3.6.1.4.1.13727.2300.2.1.1.2.1.2.0"));
54.      ne.addExtensionField(extField);
55.      extField = new NodeExtensionField();
56.      extField.setFiledSetName("ethNetworkPing");
57.      extField.addFieldValue("ethNetworkPingIp",
   session.getOID(".1.3.6.1.4.1.13727.2300.2.1.1.3.4.1.0"));
58.      extField.addFieldValue("ethNetworkPingCommand",
   session.getOID(".1.3.6.1.4.1.13727.2300.2.1.1.3.4.2.0"));
59.      ne.addExtensionField(extField);
60.      NodeExtensionTable extTable = new NodeExtensionTable();
61.      extTable.setFiledSetName("ethNetworkIpsecTable");
62.      TableValues table = new TableValues();
63.      table.addColName(".1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.1", "Estado");
64.      table.addColName(".1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.2", " Modo");
65.      table.addColName(".1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.3", "IP
   Origem");
66.      table.addColName(".1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.6", "IP
   Destino");
67.      session.getTable(table);
68.      table.reorderTable(1);
69.      List<String> lines = table.getLines();
70.      for (String lineKey : lines) {
71.         Properties prop = table.getLineProperties(lineKey);
72.         extTable.addFieldValue(lineKey,"Estado",
   prop.getProperty("Estado"));
73.         extTable.addFieldValue(lineKey," Modo", prop.getProperty("
   Modo"));
74.         extTable.addFieldValue(lineKey,"IP Origem", prop.getProperty("IP
   Origem"));
75.         extTable.addFieldValue(lineKey,"IP Destino", prop.getProperty("IP
   Destino"));
76.      }
77.      ne.addExtensionTable(extTable);
78.      NodeExtensionDAO dao = new NodeExtensionDAO();
79.      dao.persist(ne);
```

```
80.      } catch (Exception e) {
81.          e.printStackTrace();
82.      }
83.      return null;
84.      }
85.      }
```

Para especificar outra tecnologia, o nome da classe *DiscoveryCiscoSession.java* deveria ser alterado para *Discovery<nova_tecnologia>Session.java*, sendo <nova_tecnologia> o nome da nova tecnologia escolhida. Nas linhas 1, 9, 10, 11, 14 e 15 do Quadro 3.2 o nome da tecnologia Cisco é substituído pelo nome da <nova_tecnologia>. O nome da classe DiscoveryCisco.java deve ser alterado para Discovery<nova_tecnologia>.java. Nas linhas 1, 14, 16 e 19 do Quadro 3.2 o nome da tecnologia Cisco é substituído pelo nome da <nova_tecnologia>. Nas linhas 26, 34 e 42, o OID da tecnologia Cisco deve ser substituído pelo OID da <nova_tecnologia>. Nas linhas 35 e 36, deve ser informado um rótulo para a <nova_tecnologia>, o qual será usado para apresentar a tecnologia na IHC do sistema. A linha 50 refere-se ao código para instanciar um novo nó da MIB. A linha 51 corresponde ao código para atribuir o nome do nó da MIB ao nó instanciado. A linha 52 refere-se ao código que define o nome do campo e a sua identificação OID da MIB. O código dessa linha irá se repetir quantas vezes for necessário para adicionar todos os OIDs que se deseja gerenciar, que estão na MIB da tecnologia nova. Na linha 54 deve-se colocar em uma lista o nó instanciado. Na Figura 3-7 tem-se a ilustração da parte da MIB ao qual se refere o código das linhas 50 a 54.

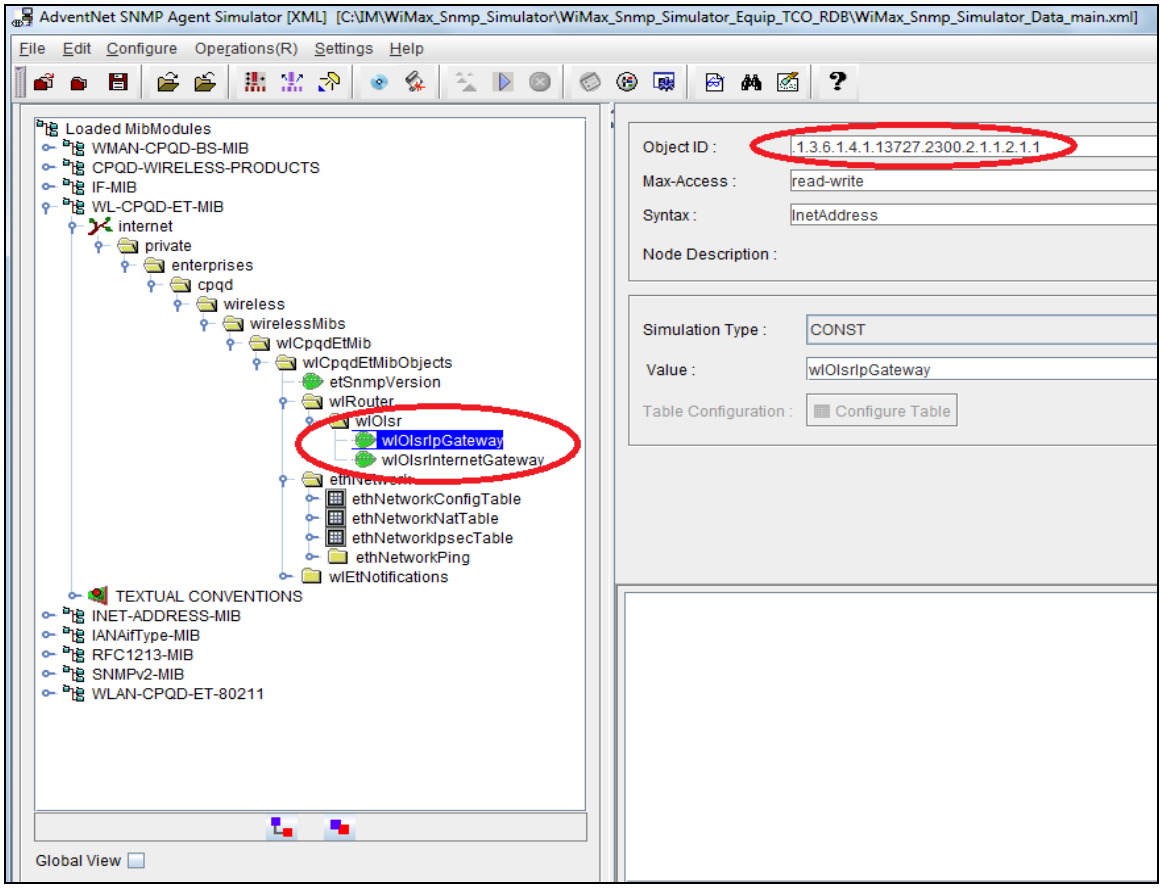

Figura 3-7: Nó da MIB que corresponde ao código das linhas 50 a 54 da classe DiscoveryCisco.java

Nas linhas 55 a 59, o código é semelhante ao código das linhas 50 a 54, mas está instanciando outro nó da MIB com outros OIDs que também serão gerenciados. Para quantos nós tiver na MIB com variáveis do tipo campo, e que serão gerenciadas pelo sistema, código semelhante ao das linhas 50 a 54 deverão ser repetidos. A linha 60 refere-se à instanciação de uma tabela para indicar que o formato da variável da MIB que será ser gerenciada é do tipo tabela. Na linha 61 faz-se a atribuição de um nome para a tabela e na linha 62, a instanciação de uma nova tabela de valores. O código da linha 63 irá se repetir quantas vezes for necessário até que todas as colunas da tabela do nó da MIB que serão gerenciadas sejam adicionadas à tabela instanciada. Na Figura 3-8 está ilustrada a variável *ethNetworkIpsecTable* do tipo tabela na MIB ao qual se refere ao código da linha 61.

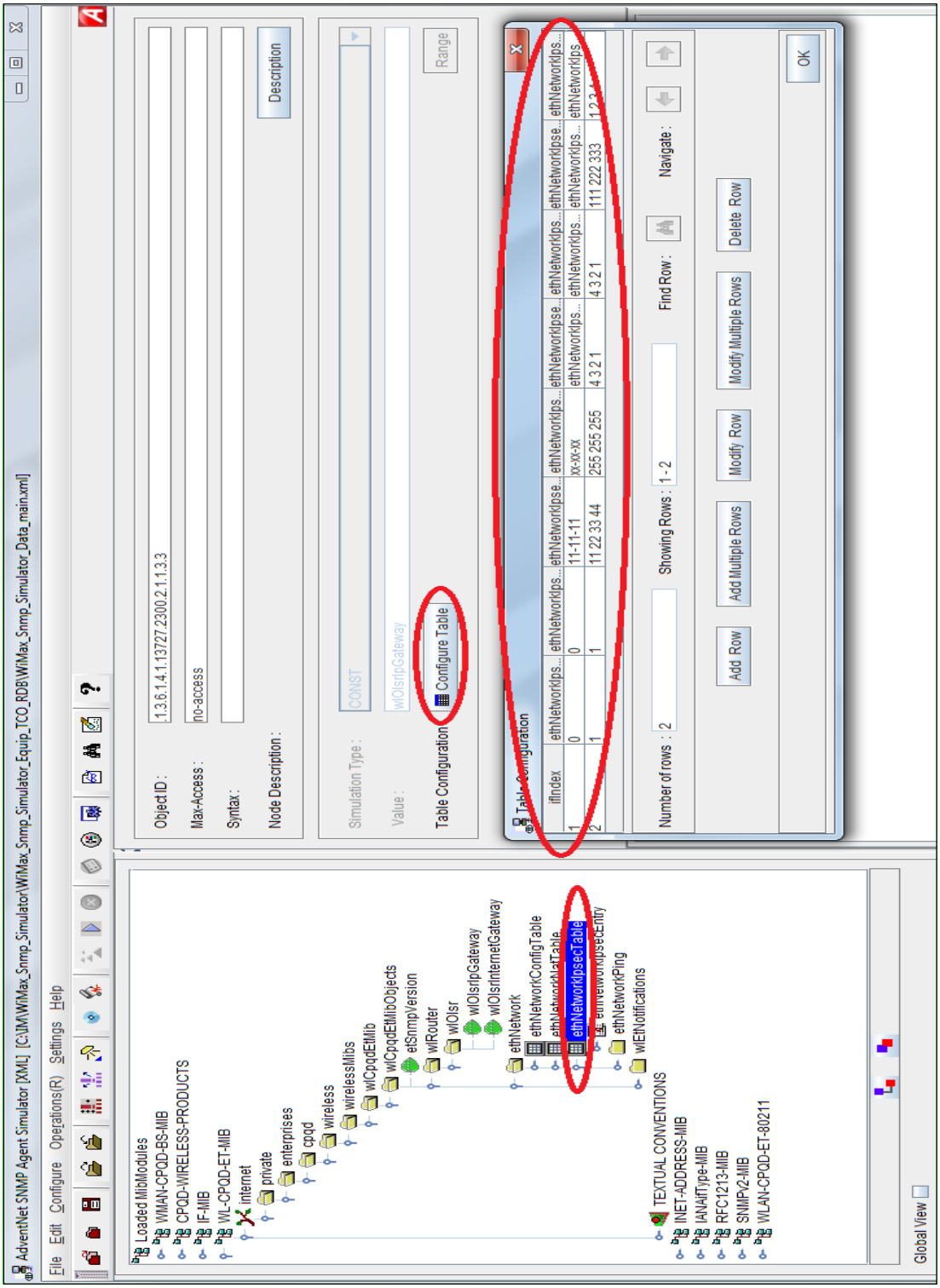

Figura 3-8: Nó da MIB que corresponde ao tipo tabela

Na Figura 3-9 é exibida a ilustração de uma das colunas da tabela *ethNetworkIpsecTable* que será gerenciada de acordo com o código da linha 64.

O código da linha 72 irá se repetir quantas vezes for necessário para atribuir o rótulo que será apresentado ao usuário na IHC do sistema para cada coluna da tabela.

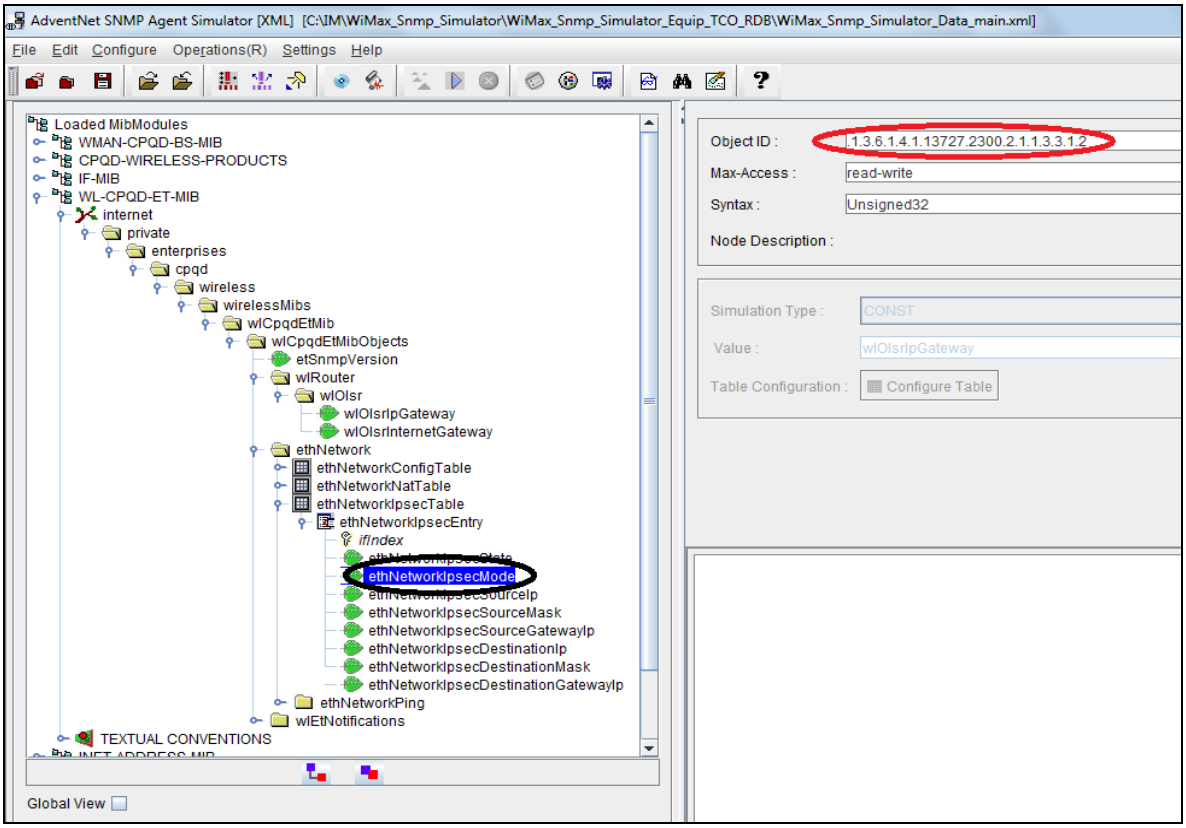

Figura 3-9: Nó da MIB que corresponde a uma das colunas da tabela ethNetworkIpsecTable

3.4 Considerações Finais

Neste capítulo foram apresentadas as funções comuns e as específicas das três áreas de gerência que compõem o sistema de gerenciamento integrado de redes: gerência de configuração, gerência de falhas e gerência de segurança.

As funções do sistema de gerenciamento integrado de redes foram analisadas e a especificação de uma nova tecnologia de rede na gerência de configuração foi selecionada para ser o negócio do domínio. Dessa forma, foi apresentada, em detalhes, a especificação da tecnologia Cisco, utilizando especialização de classes. A partir desse exemplo é possível especificar outras tecnologias de redes no sistema.

Atualmente, no cenário alvo considerado neste projeto, o processo de desenvolvimento desse sistema exige que o desenvolvedor crie duas novas classes e escreva o código Java com as características da nova tecnologia de rede que será gerenciada pelo sistema. Uma das desvantagens dessa forma de desenvolvimento é o

aumento do risco de inserção tanto de erros de digitação ou mesmo de código Java. Para deixar a documentação atualizada, conforme a implementação, o desenvolvedor também deveria atualizar o diagrama de classes existente, colocando as duas novas classes que são inseridas de acordo com a tecnologia escolhida. O analista de requisitos também deveria atualizar os documentos de requisitos especificando as características da nova tecnologia de rede. Normalmente, essas atividades não são realizadas, ficando a documentação incompatível com o código implementado.

No Capítulo 4 é apresentada a construção de uma DSL para a gerência de configuração do sistema de gerenciamento integrado de redes, aplicando os conceitos de MDD, com o objetivo de facilitar a tarefa do desenvolvedor quando for inserir uma nova tecnologia de rede na lista de tecnologias que devem ser gerenciadas pelo sistema.

Capítulo 4

UMA DSL PARA A GERÊNCIA DE CONFIGURAÇÃO DO SISTEMA DE GERENCIAMENTO INTEGRADO DE REDES

4.1 Considerações Iniciais

O sistema de gerenciamento integrado de redes deve estar sempre atualizado com as modificações realizadas. Para que isso ocorra há necessidade de que facilidades sejam propiciadas aos desenvolvedores de modo que tarefas possam ser realizadas com o apoio computacional.

No Capítulo 3 foram apresentadas as funções comuns e as específicas das três áreas de gerência que compõem o sistema de gerenciamento integrado de redes. Também foi descrito como é realizado o desenvolvimento do sistema de gerenciamento integrado de redes, em uma empresa real, com ênfase na especificação de uma nova tecnologia de rede que passa a ser gerenciada pelo sistema. Dentre as tarefas que o desenvolvedor precisa realizar consta a criação de duas novas classes em um modelo já existente e a escrita do código Java com os detalhes da nova tecnologia. Para que a documentação permaneça atualizada há necessidade também da inserção de detalhes no diagrama de classes.

Com o uso de MDD, os modelos de alto nível de abstração passam a ser parte integrante do produto final por meio de técnicas de modelagem e de geração de código e o desenvolvedor fica protegido da complexidade da plataforma de implementação

(FRANCE; RUMPE, 2007). Aplicando esse conceito no sistema de gerenciamento integrado de redes, o desenvolvedor deixa de ser o responsável por implementar o código Java para inserir uma nova tecnologia de rede na lista de tecnologias que devem ser gerenciadas pelo sistema, e pode se dedicar ao negócio do domínio, ou seja, a conhecer os detalhes da nova tecnologia que será inserida na lista de tecnologias gerenciadas pelo sistema. MDD também pode proporcionar o reúso de software de forma mais processual e possibilitar um desenvolvimento mais rápido, com menor custo, e possibilitar realizar modificações mais rapidamente (CZARNECKI e ANTKIWICZ, 2005).

Neste capítulo é mostrado o desenvolvimento de uma DSL para auxiliar na especificação de novas tecnologias de rede que podem ser inseridas no sistema de gerenciamento integrado de redes.

EMF (ECLIPSE MODELING FRAMEWORK, 2013) foi o ferramental utilizado para desenvolver a DSL, sua organização lógica define o desenvolvimento das duas partes que compõem uma DSL. A ferramenta Acceleo (ACCELEO, 2013) foi utilizada para realizar a transformação do modelo em código Java por meio da definição de *templates*.

Este capítulo está organizado da seguinte forma: na Seção 4.2 é apresentada a construção da DSL e a sua utilização é ilustrada na Seção 4.3. Na Seção 4.4 estão as considerações finais.

4.2 Construção da DSL

4.2.1 Metamodelo

MDD tem por princípio a ênfase do desenvolvimento baseado em modelos. Um sistema pode ser representado por um modelo. Uma forma de especificar vários modelos é com a criação de um metamodelo. Dessa forma, foi especificado um metamodelo com os conceitos que estão no negócio de domínio, ou seja, a descoberta de elementos de rede de uma nova tecnologia.

No EMF os modelos são representados de acordo com o seu metamodelo denominado Ecore. Assim sendo, em um contexto onde uma DSL é um metamodelo, o Ecore é um metametamodelo. O metamodelo da DSL para a especificação de uma nova

tecnologia de rede da gerência de configuração para o sistema integrado de redes pode ser observado na Figura 4-1.

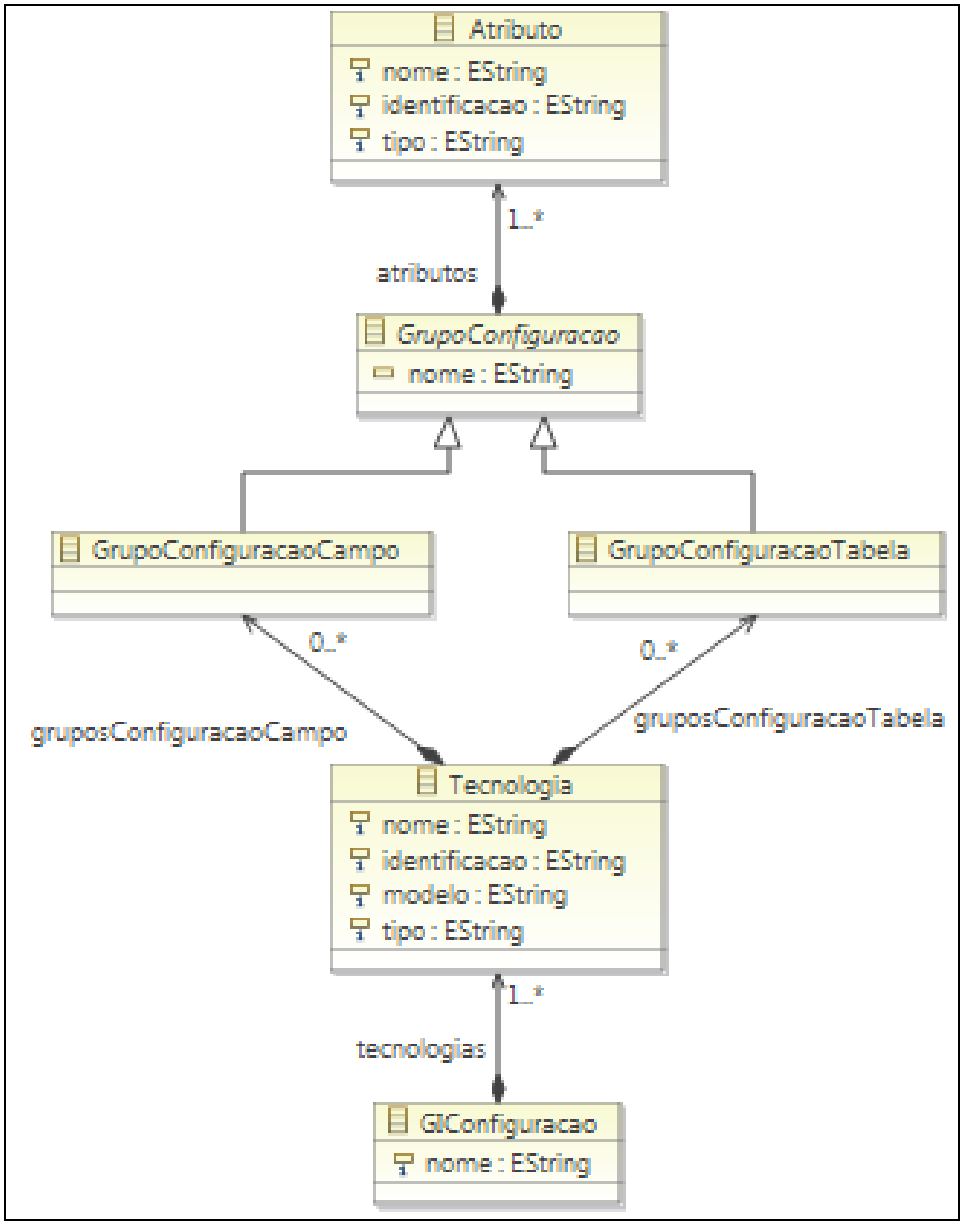

Figura 4-1: Metamodelo para a gerência de configuração

O metamodelo é composto pelos elementos e pelos relacionamentos entre esses elementos representando as regras de negócio do domínio. A EClass GIConfiguracao é a classe "*container*" do metamodelo GI para a gerência de configuração, que deve ter obrigatoriamente um nome para a configuração do sistema de gerenciamento integrado

de redes, representado por meio do atributo nome. A GIConfiguracao pode ter várias tecnologias, mas pelo menos uma é obrigatória.

A EClass Tecnologia indica a nova tecnologia de rede que será especificada e pode possuir nenhum ou vários grupos de configuração de campos e ou de tabelas. Esses grupos são compostos por atributos que caracterizam alguma configuração da tecnologia da rede. A diferença entre os grupos é disposição das informações na MIB. As informações do grupo de tabelas estão dispostas em forma de tabela na MIB do elemento de rede, enquanto as informações do grupo de campos são campos da MIB. Um grupo de configuração deve possuir pelo menos um atributo. Os atributos estão representados pela EClass Atributo. Os atributos dos grupos da configuração da tecnologia são fornecidos pelo especialista, pois são informações intrínsecas da tecnologia de rede e devem ser exatamente os mesmos que aparecem na MIB dos elementos de rede.

EMF disponibiliza meios para que seja gerado um *plug-in* para facilitar a criação de modelos conforme o metamodelo especificado. A criação desse *plug-in* não é o assunto principal deste projeto, pois é específico da ferramenta utilizada para desenvolver a DSL. Contudo, o passo a passo para a geração deste *plug-in* pode ser encontrado no Apêndice B.

4.2.2 Especificação dos *Templates*

Neste projeto é utilizada a ferramenta Acceleo (ACCELEO, 2013) para transformar o modelo especificado, de acordo com o metamodelo da gerência de configuração, em código Java usando *templates*. Os *templates* criados correspondem a dois arquivos Java: *Discovery<TECNOLOGIA>Session* e *Discovery<TECNOLOGIA>.*

Discovery<TECNOLOGIA>Session é responsável por registrar a nova tecnologia na lista de tecnologias gerenciadas pelo sistema. *Discovery<TECNOLOGIA>* é responsável por definir os atributos dos elementos de rede da nova tecnologia que deverão ser gerenciados pelo sistema. Atualmente na empresa em questão, os desenvolvedores implementam esses arquivos manualmente. O protocolo de comunicação usado para realizar a descoberta é SNMP; dessa forma, o equipamento deve ter uma MIB da qual são recuperadas as informações de gerenciamento.

Para facilitar o trabalho dos desenvolvedores, com apoio da ferramenta Acceleo foram criados dois módulos[1]: *tecnologia.mtl* descrito no Quadro 4.1 e *main.mtl* descrito no Quadro 4.2. *tecnologia.mtl* é constituído do *template* das duas classes Java e *main.mtl* é o arquivo principal que inicia a transformação do modelo em código utilizando os *templates* especificados.

O passo a passo para a criação desses *templates* é específico da ferramenta Acceleo e pode ser encontrado no Apêndice B.

Quadro 4-1: *template* tecnologia.mtl

```
[comment encoding = UTF-8 /]
[module tecnologia('http://gi')]
[template public generateTecnologia(aTecnologia : Tecnologia)]
[file (aTecnologia.nome.prefix('Discovery').concat('.java'),
false, 'UTF-8')]

/** package*/
/** import */
/** Aqui são colocados os packages e os imports. Foram omitidos
por questões de privacidade */

public class Discovery[aTecnologia.nome.toUpperFirst()/]
implements DiscoveryInterface
{
     @Override
     public String getExtensionName() {
          return "[aTecnologia.nome.toUpperFirst()/]Ex";
     }

     @Override
     public SnmpNode createNode(SnmpNode dbNode, SnmpSessionIM
session,String ipV4, String hostName, String dnsName, String
ipV6,SnmpConfig sc, String name, String ipForwarding, String
numPorts, String sysObjectID) {

          if (dbNode != null) {
               return dbNode;
          }

          if (sysObjectID != null &&
sysObjectID.equals("[aTecnologia.identificacao/]")) {
               return new SnmpNode();
          }
```

[1] Módulo é a nomenclatura usada pela ferramenta Acceleo para o arquivo que contém o *template*.

```
          return null;
     }

     @Override
     public SnmpNode prePersist(SnmpNode node, SnmpSessionIM
session, String name, String ipForwarding, String numPorts,
     String sysObjectID) {

          if (sysObjectID != null &&
sysObjectID.equals("[aTecnologia.identificacao/]")) {
               node.setModel("[aTecnologia.modelo/]");
               node.setSubType("[aTecnologia.tipo/]");
          }
          return null;
     }

     @Override
     public SnmpNode postPersist(SnmpNode dbNode, SnmpSessionIM
session,String name, String ipForwarding, String numPorts, String
sysObjectID) {

          if (sysObjectID != null &&
!sysObjectID.equals("[aTecnologia.identificacao/]")) {
               return null;
          }

          NodeExtensionField extField = null;
         try {
            NodeExtension ne = new NodeExtension();
            ne.setNode(dbNode.getResource());
            ne.setNome(getExtensionName());

               [for (grCampo : GrupoConfiguracaoCampo |
aTecnologia.gruposConfiguracaoCampo)]
               extField = new NodeExtensionField();
               extField.setFiledSetName("[grCampo.nome/]");
               [for (atr : Atributo | grCampo.atributos)]
               extField.addFieldValue("[atr.nome/]",
session.getOID("[atr.identificacao/]"));
               [/for]
               ne.addExtensionField(extField);
               [/for]

               [for (grTabela : GrupoConfiguracaoTabela |
aTecnologia.gruposConfiguracaoTabela)]

               NodeExtensionTable extTable = new
NodeExtensionTable();
               extTable.setFiledSetName("[grTabela.nome/]");
```

```
                TableValues table = new TableValues();
                [for (atr : Atributo | grTabela.atributos)]
                table.addColName("[atr.identificacao/]",
"[atr.nome/]");
                [/for]
                session.getTable(table);

                table.reorderTable(1);
             List<String> lines = table.getLines();
             for (String lineKey : lines) {
                    Properties prop =
table.getLineProperties(lineKey);
                    [for (atr : Atributo | grTabela.atributos)]
                    extTable.addFieldValue(lineKey,"[atr.nome/]",
prop.getProperty("[atr.nome/]"));
                    [/for]
             }
             ne.addExtensionTable(extTable);
               [/for]

             NodeExtensionDAO dao = new NodeExtensionDAO();
             dao.persist(ne);
          } catch (Exception e) {
                    e.printStackTrace();
               }
         return null;
     }
}

[/file]

[file
(aTecnologia.nome.prefix('Discovery').concat('Session.java'),
false, 'UTF-8')]

/** package*/
/** import */
/** Aqui são colocados os packages e os imports. Foram omitidos
por questões de privacidade */

@Singleton
@Startup
public class Discovery[aTecnologia.nome.toUpperFirst()/]Session {

     private Discovery[aTecnologia.nome.toUpperFirst()/] instance;

    @PostConstruct
    public final void init() {
```

```
            instance = new
Discovery[aTecnologia.nome.toUpperFirst()/]();
            DiscoveryExtension.list.add(instance);
        }
}

[/file]
[/template]
```

Quadro 4-2: *template main.mtl*

```
 [comment encoding = UTF-8 /]
[module main('http://gi')/]
[import eclipse::acceleo::gi::files::tecnologia/]

[template public mainTecnologia(aTecnologia : Tecnologia)]

     [comment @main /]
     [aTecnologia.generateTecnologia()/]

[/template]
```

Acceleo disponibiliza meios para que seja gerado um *plug-in* para facilitar a geração do código a partir do modelo especificado. O passo a passo para a geração desse *plug-in* está disponibilizado no Apêndice B.

4.3 Utilização da DSL

Neste projeto a DSL é utilizada por meio dos dois *plug-ins* construídos: um para especificar o modelo, conforme o metamodelo da gerência de configuração, e outro para realizar a transformação do modelo especificado em código Java. Para isto, na ferramenta Eclipse, deve-se criar um novo projeto do *plug-in* que contém o metamodelo da gerência de configuração e especificar um modelo com as definições da nova tecnologia de rede.

A especificação de uma nova tecnologia de rede depende exclusivamente dos requisitos que o especialista de redes define. Um exemplo de especificação de modelo de

gerência de configuração que contém as definições de uma nova tecnologia de rede está descrito no Quadro 4.3.

Quadro 4-3: Um exemplo de especificação de uma nova tecnologia de rede

```
IP: 10.202.35.63

Nome da Tecnologia: Cisco
Identificação da Tecnologia: .1.3.6.1.4.1.13727.2300.1.1.1
Modelo: WiMax BS
Tipo: WiMax BS

Grupos de Campos

Nome do Grupo1: wlOlsr
Nome do Campo1: wlOlsrIpGateway
Identificação do Campo1: .1.3.6.1.4.1.13727.2300.2.1.1.2.1.1.0
Nome do Campo2: wlOlsrInternetGateway
Identificação do Campo2: .1.3.6.1.4.1.13727.2300.2.1.1.2.1.2.0

Nome do Grupo2 ethNetworkPing
Nome do Campo1: ethNetworkPingIp
Identificação do Campo1: .1.3.6.1.4.1.13727.2300.2.1.1.3.4.1.0
Nome do Campo2: ethNetworkPingCommand
Identificação do Campo2: .1.3.6.1.4.1.13727.2300.2.1.1.3.4.2.0

Grupos de Tabelas

Nome do Grupo1: ethNetworkIpsecTable
Nome da Coluna1: Estado
Identificação da Coluna1: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.1
Nome da Coluna2: Modo
Identificação da Coluna2: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.2
Nome da Coluna3: IP Origem
Identificação da Coluna 2: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.3
Nome da Coluna4: IP Destino
Identificação da Coluna4: .1.3.6.1.4.1.13727.2300.2.1.1.3.3.1.6
```

O modelo para a especificação da nova tecnologia de rede, descrita no Quadro 4.3, criado por meio da utilização do *plug-in*, está representado na Figura 4-2.

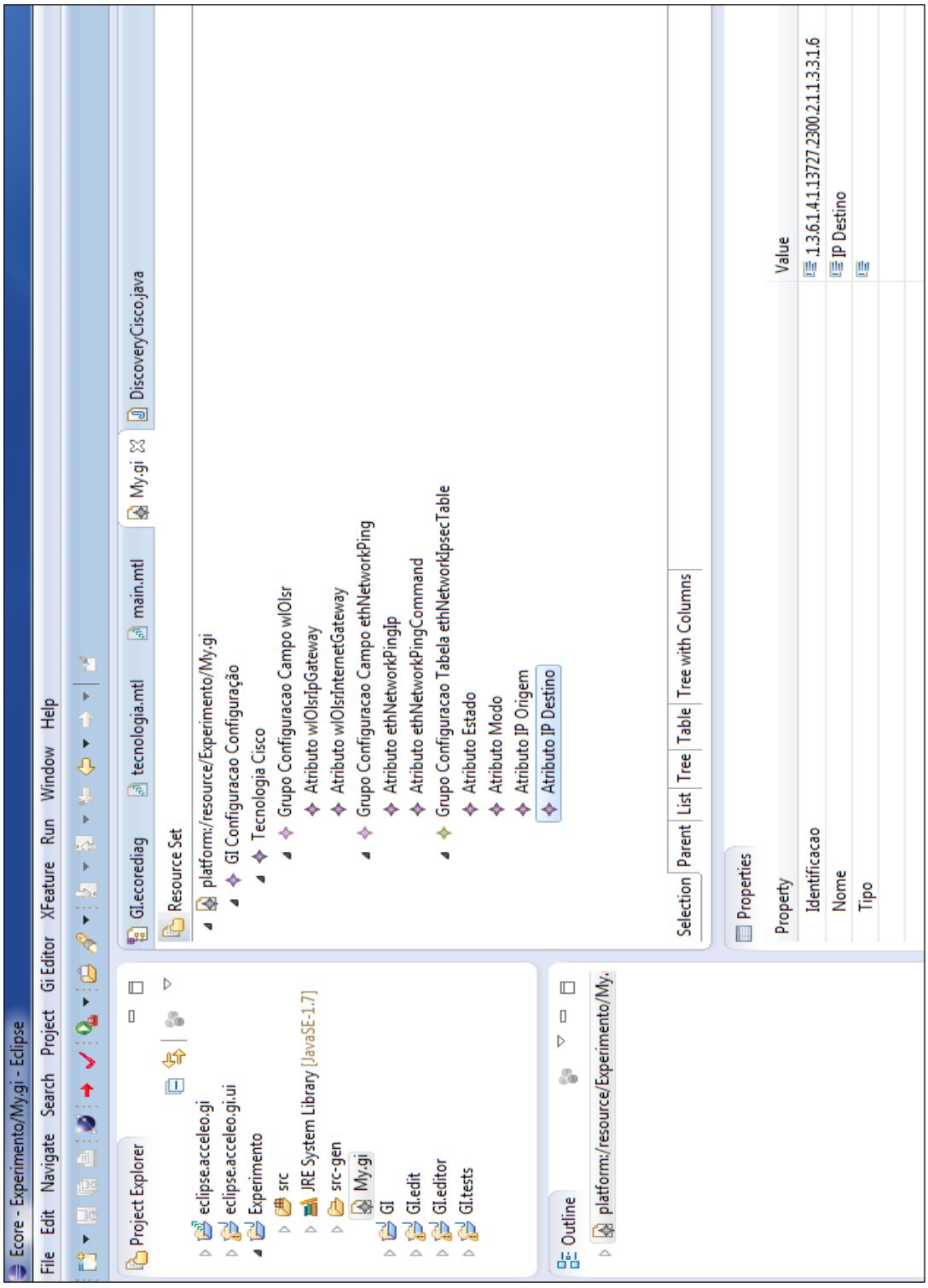

Figura 4-2: Modelo da especificação da nova tecnologia de rede

Após a especificação da nova tecnologia de rede, pode-se solicitar a geração do código-fonte Java correspondente ao modelo, como representado na Figura 4-3.

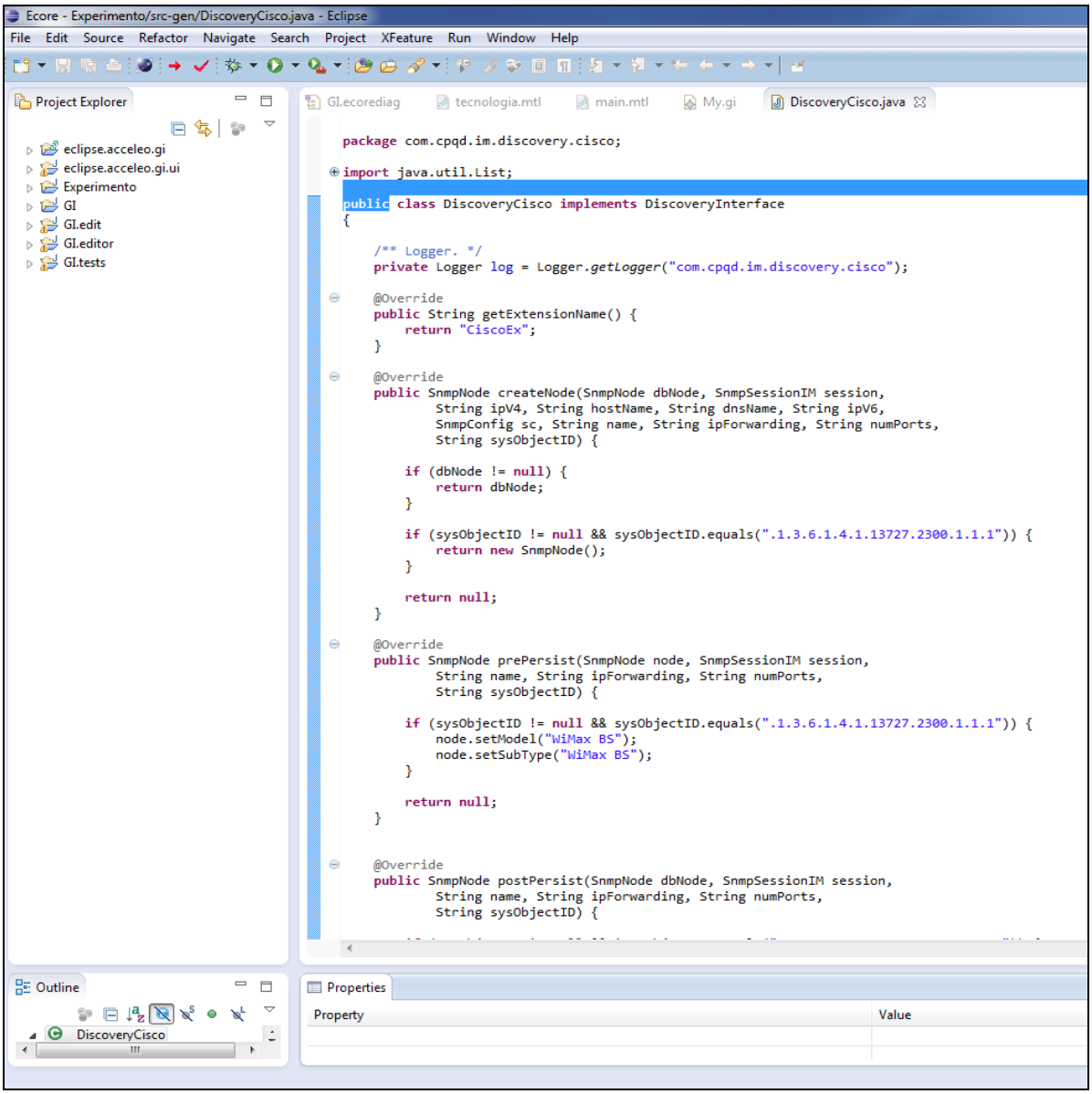

Figura 4-3: Código gerado para o modelo

O passo a passo para a utilização dos *plug-ins* pode ser encontrado no Apêndice B.

4.4 Considerações Finais

Neste capítulo foi apresentada uma abordagem para a construção de uma DSL para a gerência de configuração do sistema de gerenciamento integrado de redes com a finalidade de facilitar a inclusão de uma nova tecnologia de rede que passará a ser gerenciada pelo sistema de gerenciamento integrado de redes.

A sintaxe abstrata da DSL foi desenvolvida utilizando a ferramenta EMF para a especificação do metamodelo e a ferramenta Acceleo para a transformação do modelo em código Java por meio de *templates*. As duas ferramentas dispõem de facilidades para a geração de *plug-in* para Eclipse. A sintaxe concreta da DSL foi realizada com a utilização dos *plug-ins* gerados. Para a construção da DSL, é necessário o conhecimento sobre o negócio do domínio e, para a utilização da DSL, são necessárias as informações do especialista de redes.

O passo a passo de como se utiliza a DSL desenvolvida pode ser seguido por qualquer desenvolvedor da empresa real para especificar uma nova tecnologia de rede, que passará a ser gerenciada pelo sistema de gerenciamento de redes. Essa nova tecnologia de rede é especificada por meio de um modelo construído conforme as regras de negócio do domínio especificadas no metamodelo da DSL desenvolvida. Dessa forma, o desenvolvedor escolhe a nova tecnologia de rede em alto nível de abstração, sem precisar se preocupar com detalhes da plataforma de implementação Java. Além disso, a DSL possui mecanismos de validação que reduzem a possibilidade dos desenvolvedores especificarem uma nova tecnologia de rede de forma incorreta. Após a nova tecnologia de rede ser especificada usando o modelo, o desenvolvedor solicita a geração do código-fonte corresponde. Dessa forma, a DSL proporciona maior eficiência no desenvolvimento, pois o código-fonte é gerado a partir do modelo.

Além disso, a DSL também proporciona melhor qualidade na aplicação uma vez que reduz a possibilidade de erros na modelagem da especificação da nova tecnologia de rede e, consequentemente, os erros na geração do código-fonte. No processo de desenvolvimento adotado pela empresa real baseado na especialização de classe, o código para especificar a nova tecnologia de rede deve ser digitado pelo desenvolvedor, o que aumenta a possibilidade da inserção de erros de digitação ou erros de código Java. O código-fonte gerado a partir do modelo na DSL é exatamente igual ao código que deve

ser digitado pelo desenvolvedor na abordagem de especialização de classes na especificação de uma nova tecnologia de rede.

Com o uso da DSL desenvolvida a especificação de novas tecnologias de rede passa a ser feita de forma mais processual, sendo que os desenvolvedores podem dedicar mais tempo para conhecer as tecnologias de rede ao invés de despender tempo com particularidades de implementação.

As dificuldades que podem ser encontradas para a utilização da DSL são as relacionadas com a não familiaridade com o ambiente de desenvolvimento do Eclipse, pois, neste projeto, foram utilizadas as ferramentas da plataforma Eclipse para desenvolver a DSL. Nas abordagens tanto de especialização de classes quanto da DSL, o desenvolvedor precisa ter conhecimento sobre as tecnologias de rede.

No Capítulo 5 está descrita a avaliação da proposta que consistiu da utilização da DSL desenvolvida e descrita nesse capítulo. O objetivo é verificar se o uso da DSL proporciona a diminuição da inserção de erros no código e possibilita um desenvolvimento mais rápido e, consequentemente, com menor custo.

Capítulo 5

AVALIAÇÃO DA PROPOSTA

5.1 Considerações Iniciais

No Capítulo 3 foi apresentada a forma como é realizado o desenvolvimento de um sistema de gerenciamento integrado de redes, em uma empresa real, com relação à especificação de uma nova tecnologia de rede. No Capítulo 4 foi apresentada a DSL que foi desenvolvida para definir a nova tecnologia à qual pertencerão os novos elementos de rede que passarão a ser gerenciados pelo sistema. As duas formas de desenvolvimento propõem o reúso de software, a primeira com especialização de classes e a segunda com o uso da DSL.

Para usar qualquer uma das duas formas, o desenvolvedor deve seguir regras bem definidas para a construção dos artefatos de software. O objetivo do desenvolvimento da DSL foi automatizar a construção do código, permitindo ao desenvolvedor ter maior dedicação à definição de uma nova tecnologia de rede, em um nível mais alto de abstração. Além de proporcionar a atualização da documentação, uma vez que o modelo passa a ser parte integrante do software.

Desse modo, ao utilizar a DSL espera-se que haja diminuição da inserção de erros no código, que seja possível um desenvolvimento mais rápido e, consequentemente, com menor custo. Com o propósito de verificar na prática as vantagens que essa abordagem fornece foi realizado um experimento com profissionais da empresa real que conhecem o sistema de gerenciamento integrado de redes.

Esse experimento foi planejado e executado seguindo a abordagem definida por Wohlin *et al*. (2000), que é composta por três fases: 1) definição e planejamento, em que são especificados o contexto, as hipóteses, as variáveis, os participantes, os instrumentos e o modelo do experimento; 2) operação, em que ocorre a preparação e a execução do

experimento com os participantes; e 3) análise dos dados, em que os dados são organizados e analisados por meio de técnicas estatísticas.

Na Seção 5.2 é descrito o experimento que avaliou o reúso utilizando a DSL desenvolvida e na Seção 5.3 são comentadas as considerações finais deste capítulo.

5.2 Experimento

Nesta seção é apresentado o experimento conduzido com o objetivo de analisar as vantagens propiciadas pelo uso da DSL desenvolvida e descrita no Capítulo 4 quando comparada com a utilização de especialização de classes para definir uma nova tecnologia de rede no sistema de gerenciamento integrado de redes. Para tanto serão considerados: o tempo despendido para especificar uma nova tecnologia de rede e o número de erros observados no código-fonte quando aplicadas as duas abordagens de desenvolvimento.

5.2.1 Definição e Planejamento do Experimento

Esse experimento foi definido como apresentado na Tabela 5-1. As etapas da fase de planejamento do experimento estão descritas nas Subseções 5.2.1.1 a 5.2.1.5.

Análise da	definição de uma nova tecnologia de rede na gerência de configuração, utilizando a DSL desenvolvida, para que elementos dessa tecnologia sejam descobertos e gerenciados pelo sistema de gerenciamento integrado de redes.
Com o propósito de	avaliação.
A partir do ponto de vista dos	desenvolvedores da equipe de desenvolvimento de software do sistema de gerenciamento integrado de redes de uma empresa real
Com respeito à	eficiência (tempo) e à dificuldade (número de erros) no uso da DSL desenvolvida neste projeto de mestrado quando comparada com a especialização de classes, método utilizado atualmente, para a definição de uma nova tecnologia de rede.

Tabela 5-1 Definição do Experimento

5.2.1.1 Contexto e Participantes

O contexto desse experimento foi multi-teste dentro de um objeto de estudo (WOHLIN *et al*., 2000). Consistiu em testes experimentais executados por um grupo de

indivíduos para estudar uma única abordagem, que se trata da definição de uma nova tecnologia de rede na gerência de configuração utilizando a DSL desenvolvida. O experimento foi realizado em uma empresa real. Participaram do experimento dez desenvolvedores, os quais direta ou indiretamente conhecem o sistema. Ressalta-se que os participantes possuem diferentes visões do sistema, ou seja, de acordo com a sua função na equipe de desenvolvimento (analistas de requisitos, arquiteto, implementadores e analistas de testes). Dentre esses desenvolvedores, 40% possuem conhecimento básico em linguagem Java, 60% possuem conhecimento avançado em linguagem Java, 10% possuem conhecimento em MDD e DSL na teoria e na prática; e 90% possuem conhecimento de MDD e DSL advindo do treinamento.

5.2.1.2 Formulação das Hipóteses

A aplicação de um experimento para um determinado grupo de pessoas tem por objetivo obter respostas para algumas perguntas que são feitas sobre o objeto de estudo (WOHLIN *et al.,* 2000). Para cada pergunta são definidas as métricas (indicadores) que orientam a coleta dos dados e as hipóteses dos possíveis resultados.

Para definir essas hipóteses deve-se sempre considerar uma hipótese nula, que é aquela que considera que não haverá diferença significativa entre os resultados obtidos quando forem aplicadas as diferentes abordagens sobre o objeto de estudo. E todas as demais hipóteses alternativas que consideram os outros possíveis resultados. Por exemplo:

- Hipótese alternativa 1: considera a abordagem X mais eficiente do que a abordagem Y;
- Hipótese alternativa 2: considera a abordagem X menos eficiente do que a abordagem Y.

Alguns cálculos podem ser realizados sobre os dados coletados para se obter um resultado e verificar se alguma das hipóteses alternativas foi verdadeira. Por exemplo, os cálculos de média e de porcentagem podem indicar que, em um determinado experimento, o tempo gasto pelos participantes para desenvolver uma aplicação foi menor quando a abordagem X foi utilizada do que quando a abordagem Y foi utilizada. Contudo, são necessários testes estatísticos para comprovar que esse resultado é significativo, refutando a hipótese nula.

As questões, as métricas e as hipóteses definidas para este experimento foram:

- **Questão 1 (Q1)**: A DSL torna a definição de uma nova tecnologia de rede na gerência de configuração mais eficiente quando comparada com a utilização da especialização das classes, que é a forma como é feito o desenvolvimento hoje?
  - **Métrica 1 (M1)**: tempo (t) gasto pelos participantes na definição de uma nova tecnologia de rede na gerência de configuração
    - **Hipótese Nula ($H1_0$)**: Não há diferença significativa utilizando a DSL e fazendo a especialização da classe quanto ao tempo gasto para a definição de uma nova tecnologia de rede na gerência de configuração. Isso pode ser formalizado como $t_{DSL} = t_{especialização}$.
    - **Hipótese Alternativa ($H1_1$)**: Com a DSL o tempo gasto pelos participantes na definição de uma nova tecnologia de rede na gerência de configuração é menor do que especializando as classes. Isso pode ser formalizado como $t_{DSL} < t_{especialização}$.
    - **Hipótese Alternativa ($H1_2$)**: Com a DSL o tempo gasto pelos participantes na definição de uma nova tecnologia de rede na gerência de configuração é maior do que especializando as classes. Isso pode ser formalizado como $t_{DSL} > t_{especialização.}$
- **Questão 2 (Q2)**: A DSL reduz a chance dos participantes cometerem erros no código-fonte durante a definição de uma nova tecnologia de rede na gerência de configuração?
  - **Métrica 2 (M2)**: números de problemas (p) encontrados no código-fonte dos artefatos gerados pelos participantes.
    - **Hipótese Nula ($H2_0$)**: Não há diferença significativa no número de problemas encontrados no código-fonte dos artefatos gerados pelos participantes utilizando ou não a DSL. Isso pode ser formalizado como $p_{DSL} = p_{especialização}$.
    - **Hipótese Alternativa ($H2_1$)**: Com a DSL o número de problemas encontrados na geração dos artefatos é menor do que com a especialização das classes. Isso pode ser formalizado como $p_{DSL} < p_{especialização}$.
    - **Hipótese Alternativa ($H2_2$)**: Com a DSL o número de problemas encontrados na geração dos artefatos é maior do que com a

especialização das classes. Isso pode ser formalizado como $p_{DSL} > p_{especialização}$.

5.2.1.3 Variáveis

Esse experimento teve as seguintes variáveis independentes: 1) o sistema de gerenciamento integrado de redes; 2) ambiente Eclipse; 3) a maneira como é feito hoje o desenvolvimento do sistema de gerenciamento integrado de redes em uma empresa real, ou seja, por meio de especialização de classes para definir uma nova tecnologia de rede; 4) a linguagem de programação Java, 5) a utilização da DSL desenvolvida; 6) as especificações das novas tecnologias de redes, cujos elementos serão descobertos e gerenciados pelo sistema (as duas especificações utilizadas pelo experimento têm o mesmo nível de complexidade).

As variáveis dependentes são as seguintes: 1) eficiência, que está relacionada com o tempo gasto para definir a nova tecnologia de rede e 2) o número de problemas encontrados nos artefatos gerados em decorrência de erros cometidos pelos participantes durante a definição da nova tecnologia de redes.

5.2.1.4 Modelo

O modelo de experimento utilizado foi um fator com dois tratamentos pareados (WOHLIN *et al*., 2000). Nesse experimento, o fator foi o uso de uma abordagem para definir a nova tecnologia de rede, enquanto os tratamentos foram as abordagens aplicadas: DSL e especialização de classes.

O experimento seguiu o formato em que os participantes são alocados em grupos homogêneos para que o nível de experiência deles não impactasse nos resultados. Um Formulário de Caracterização de Participante (Apêndice A) foi elaborado e distribuído entre os participantes para conhecer o nível de experiência de cada um. Nesse formulário os participantes responderam questões de múltipla escolha a respeito do seu conhecimento sobre: 1) programação Java, 2) padrões de software, 3) *frameworks*, 4) MDD e 5) DSL. Cada questão possuía a seguinte pontuação:

0 – quando o participante não tinha conhecimento algum;

1 – quando o participante tinha apenas conhecimento teórico;

2 – quando o participante tinha conhecimento teórico e prático.

Nessa fase de planejamento do experimento, foi decidido que os participantes seriam divididos em dois grupos (G1 e G2), sendo que cada grupo deveria ser composto com o mesmo número de participantes que obtiveram pontuação baixa (0-3), média (4-7) e alta (8-12) no Formulário de Caracterização de Participante (Apêndice A).

Para que pudessem realizar o experimento, os participantes foram treinados na definição de uma nova tecnologia de rede, tanto utilizando a DSL quanto fazendo a especialização das classes.

Na Tabela 5-2 é apresentado como foram definidas as atividades de desenvolvimento para os participantes do G1 e do G2.

Tabela 5-2 Atividades do Experimento

Atividade	**Aplicação**	**Definição de uma nova tecnologia de rede**	
		Grupo 1 (G1)	**Grupo 2 (G2)**
1	Base Station WiMax	Especialização	DSL
2	TCO RDB	DSL	Especialização

5.2.1.5 Instrumentação

Os participantes receberam os seguintes materiais para a execução do experimento: documento com a especificação da nova tecnologia de rede; o Eclipse EE configurado com o sistema de gerenciamento integrado de redes; o Eclipse *Modeling* com os *plug-ins* da DSL; e o Formulário de Coleta de Dados (Apêndice A). No Formulário de Coleta de Dados os participantes preenchiam o tempo gasto no desenvolvimento do sistema, as interrupções com erros e dúvidas, além de relatarem suas opiniões e sugestões a respeito do uso da DSL e da especialização das classes.

5.2.2 Operação do Experimento

Depois de definir e planejar o experimento, sua fase de operação foi realizada em duas etapas: Preparação e Execução.

5.2.2.1 Preparação

Alguns dias antes da realização do experimento, os participantes preencheram o Formulário de Caracterização de Participante (Apêndice A), reportando sua experiência

com relação aos conceitos e tecnologias utilizados no experimento. Os participantes também foram treinados na definição de uma nova tecnologia de rede utilizando tanto a DSL quanto a especialização das classes. Um experimento piloto também foi realizado para que os participantes obtivessem experiência com a realização das atividades do experimento.

5.2.2.2 Execução

Primeiramente, os participantes foram divididos em dois grupos conforme sua pontuação no Formulário de Caracterização de Participante (Apêndice A). Cada grupo ficou com 5 participantes. O G1 ficou com 1 participante com pontuação baixa (0-3), 3 com pontuação média (4-7) e 1 com pontuação alta (8-12). O G2 ficou com 1 participante com pontuação baixa (0-3), 3 com pontuação média (4-7) e 1 com pontuação alta (8-12). Após tomarem conhecimento de qual grupo pertenciam, os participantes receberam o material para executarem a atividade. O tempo limite era de 30 minutos para cada atividade.

Na Atividade 1 os participantes do G1 definiram uma nova tecnologia de rede especializando as classes e os participantes do G2 definiram a mesma tecnologia de rede utilizando a DSL. Cada participante fez sua atividade individualmente e cronometrou o tempo gasto desde a modelagem (DSL) ou especialização das classes até a geração do código-fonte. Depois disso, com o cronômetro parado, os participantes executavam o sistema para verificar se a execução era bem sucedida ou não. Quando ocorriam erros de execução do sistema com relação à nova tecnologia de rede, os participantes anotavam os erros no Formulário de Coleta de Dados (Apêndice A) e, então, mediam o tempo gasto para corrigir os problemas durante o desenvolvimento e executavam o sistema novamente. O tempo gasto com interrupções também foi anotado pelos participantes para que esse tempo fosse abatido do tempo total de desenvolvimento da atividade.

A Atividade 2 foi executada da mesma forma que a Atividade 1. Contudo, nessa segunda atividade, os participantes do G1 utilizaram a DSL e os participantes do G2 utilizaram a especialização de classes para a mesma tecnologia de rede.

5.2.3 Análise dos Dados do Experimento

Os dados do experimento estão apresentados na Tabela 5-3. Os participantes desenvolveram as atividades de acordo com o planejado. A análise dos dados é apresentada nas subseções seguintes.

5.2.3.1 Análise Descritiva dos Dados

Na Tabela 5-3 pode ser visto que houve uma redução no tempo gasto pelos participantes na definição da nova tecnologia de rede quando foi utilizada a DSL em comparação com a especialização da classe. Aproximadamente, 54,41% do tempo total do experimento foi gasto com a especialização da classe e 45,59% com a DSL, como pode ser observado na última linha da Tabela 5-3. Analisando o tempo gasto por cada um dos participantes em ambas as abordagens, especialização de classes e DSL, 4 deles levaram o mesmo tempo usando as duas abordagens e 6 deles levaram menos tempo usando a abordagem da DSL. De acordo com o *feedback* dado pelos participantes no formulário, esse resultado é devido ao fato de que na abordagem DSL o código é gerado automaticamente e corretamente. Por outro lado, quando os participantes desenvolveram a aplicação aplicando a abordagem de especialização de classes, eles levaram mais tempo corrigindo os problemas encontrados. Considerando que os participantes encontraram mais facilidade em especificar a nova tecnologia de rede utilizando a DSL, infere-se, portanto, que os participantes mostraram uma habilidade maior no manuseio da DSL.

Com relação aos problemas, aproximadamente, 100% deles foram encontrados na definição de uma nova tecnologia de rede utilizando a especialização da classe e 0% utilizando a DSL. Os principais problemas encontrados foram dois: erro de digitação das especificações da nova tecnologia de rede e erro de código Java. Ambos relacionados com o conhecimento com que os participantes possuem sobre o negócio do domínio e o sistema de gerenciamento integrado de redes. De acordo com o *feedback* dos participantes, a principal razão para essa diferença é a facilidade do uso da DSL em comparação com a necessidade de escrever o código Java na especialização das classes. O modelo criado com a utilização da DSL proporciona uma visão mais ampla da nova tecnologia de rede, enquanto que para escrever as classes precisa conhecer Java

para implementar os blocos de código para cada grupo de configuração e seus respectivos atributos.

Tabela 5-3 Dados coletados do experimento

	Especialização			DSL		
Aplicação	**Participante**	**Tempo (min)**	**Problemas**	**Participante**	**Tempo (min)**	**Problemas**
BS WiMax	S1	17	2	S6	6	0
	S2	16	0	S7	15	0
	S3	13	1	S8	11	0
	S4	17	0	S9	15	0
	S5	12	0	S10	16	0
	Média	**15**	**0,6**	**Média**	**12,6**	**0**
TCO RDB	S6	10	1	S1	13	0
	S7	15	0	S2	14	0
	S8	11	1	S3	9	0
	S9	15	1	S4	13	0
	S10	22	4	S5	12	0
	Média	**14,6**	**1,4**	**Média**	**12,2**	**0**
Média Geral		**14,8**	**1**		**12,4**	**0**
Porcentagem		**54,41**	**100**		**45,59**	**0**

5.2.3.2 Teste das Hipóteses

Os testes estatísticos aplicados sobre os dados coletados com o experimento foram os sugeridos por Wohlin *et al*. (2000): *Shapiro-Wilk*, *Paried T-Test* e *Wilcoxon Signed Rank*. Eles foram executados com o apoio da ferramenta *Action* (ESTATCAMP, 2013), que é um software para análise estatísticas, integrado ao *MS Excel*. Uma explicação detalhada sobre esses testes foge do escopo deste projeto de mestrado, mas pode ser encontrada no *Portal Action*[2].

Tempo

O teste *Shapiro-Wilk* (WOHLIN, 2000) é aplicado para verificar se um conjunto de dados segue, ou não, uma distribuição normal, que possui graficamente o formato de um sino simétrico em relação à sua média (Figura 5-1). Se o P-valor (resultado) do teste *Shapiro-Wilk* sobre um conjunto de dados for menor que 0,05 significa que a chance dos dados seguirem uma distribuição normal é menor do que 5%. Quando esse resultado

[2] http://www.portalaction.com.br/

ocorre, considera-se, estatisticamente, que os dados não seguem uma distribuição normal (ESTATCAMP, 2013).

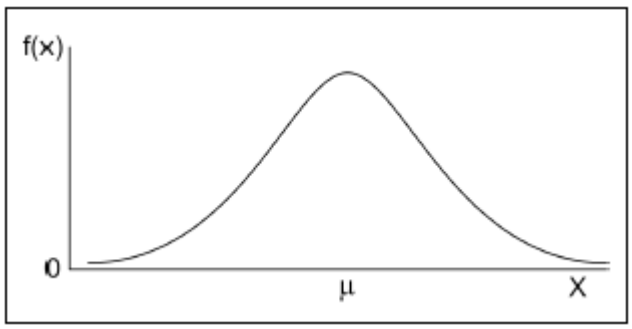

Figura 5-1 Representação gráfica de uma distribuição normal de dados

Em geral, as ferramentas de estatística utilizam um gráfico de probabilidade (*Q-Q Plot*) (ESTATCAMP, 2013) para representar graficamente a distribuição de um conjunto de dados. Nesse gráfico, quando os dados se posicionam ao redor da linha diagonal, considera-se que os dados seguem uma distribuição normal. Na Figura 5-2 são mostrados dois exemplos de gráficos de probabilidade, um com distribuição normal e outro não normal.

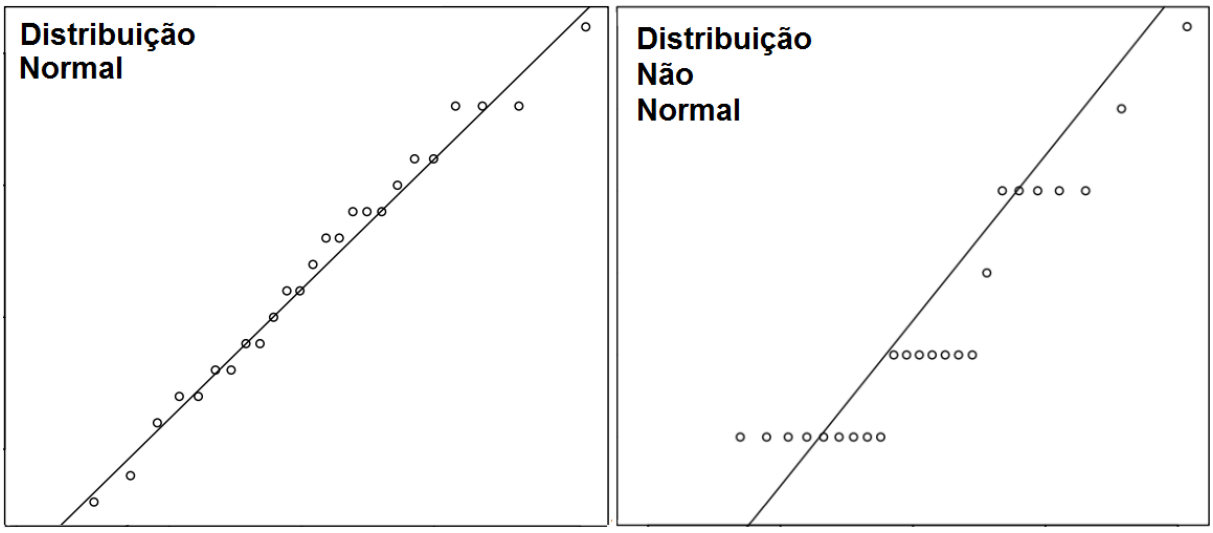

Figura 5-2 Exemplos de gráficos de probabilidade

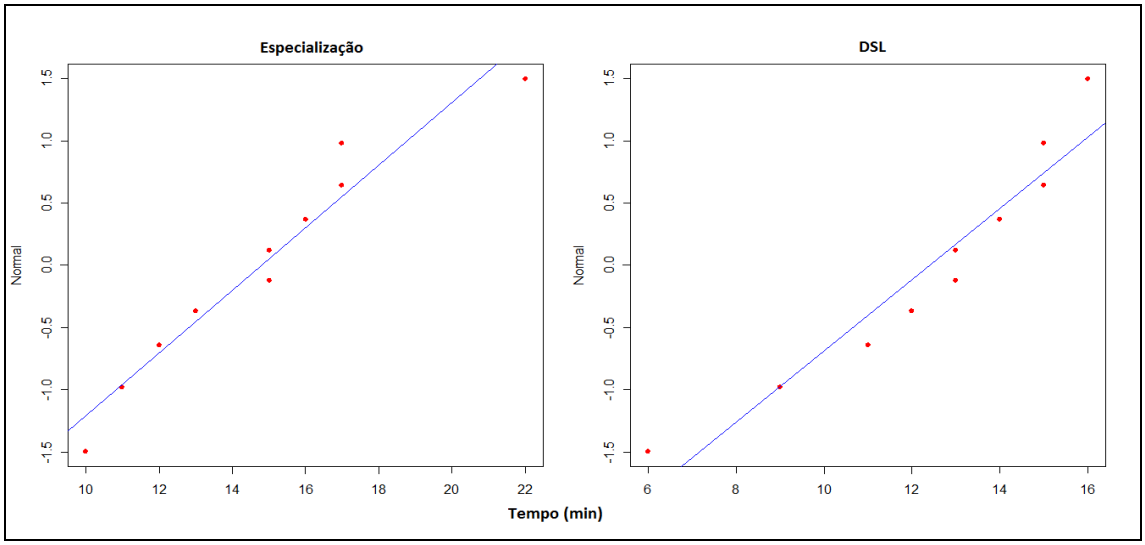

Figura 5-3 Gráficos resultantes do teste de normalidade sobre os dados do tempo gasto no desenvolvimento das aplicações

O P-valor resultante do teste *Shapiro-Wilk* sobre os dados do tempo gasto no desenvolvimento das aplicações foi 0,6726 utilizando a especialização da classe e 0,3606 utilizando a DSL. Portanto, como o P-valor dos dois testes foi superior a 0,05, pode-se afirmar, com nível de confiança de 95%, que os dados do tempo gasto no desenvolvimento das aplicações seguem uma distribuição normal. Isso pode ser verificado nos gráficos mostrados na Figura 5-3, nos quais os dados estão distribuídos sobre a reta.

O teste que pode ser aplicado somente quando ambos os conjuntos de dados seguem uma distribuição normal é o *Paried T-Test*. Se o P-valor (resultado) do *Paried T-Test* for menor que 0,05, significa que a chance dos dois conjuntos de dados serem estatisticamente semelhantes é menor do que 5%. Portanto, nesse caso, a hipótese nula deve ser rejeitada (ESTATCAMP, 2013).

Como os dados de tempo coletados no experimento estão normalizados, o *Paried T-Test* foi aplicado para verificar as hipóteses da Q1 do experimento (Seção 5.2.1.2). O resultado desse teste foi um P-valor = 0,1474. Dessa forma, como 0,1474 é maior que 0,05, com nível de confiança de 95%, no conjunto dos dados não há diferença entre o tempo gasto para desenvolver as aplicações utilizando a especialização de classes ou utilizando a DSL. Portanto, a hipótese nula ($H1_0$) deve ser aceita.

Número de Problemas

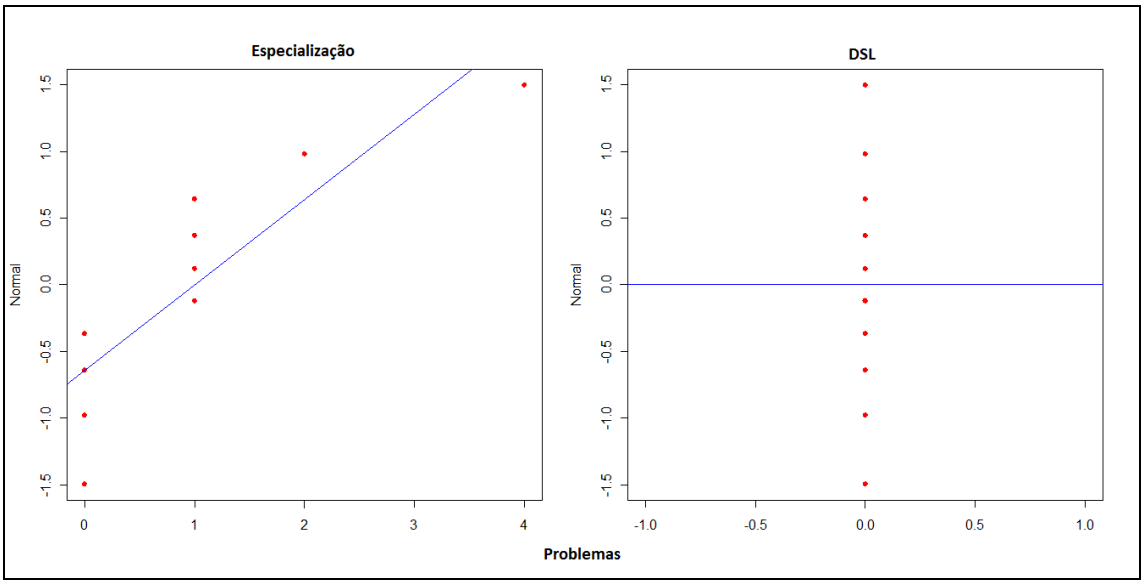

Figura 5-4 Gráficos resultantes do teste de normalidade sobre os dados do número de problemas no código-fonte das aplicações

O P-valor resultante do teste *Shapiro-Wilk* sobre os dados do número de problemas no código-fonte foi 0,0068 utilizando a especialização de classes e não se pode calcular o P-valor com a utilização da DSL, pois todos os dados são iguais a zero. Portanto, como o P-valor para a especialização de classe foi inferior a 0,05, pode-se afirmar, com nível de confiança de 95%, que os dados obtidos relacionados ao número de problemas no código-fonte não seguem uma distribuição normal. Isso também pode ser verificado nos gráficos da Figura 5-4, pois os dados obtidos relacionados ao número de problemas no reúso com a DSL não estão distribuídos sobre a reta.

O teste que pode ser aplicado somente quando ao menos um desses conjuntos não segue uma distribuição normal é o teste *Wilcoxon Signed Rank*. Assim como no *Paried T-Test*, se o P-valor (resultado) do teste *Wilcoxon Signed Rank* for menor que 0,05 significa que a chance dos dois conjuntos de dados serem estatisticamente semelhantes é menor do que 5%. Portanto, a hipótese nula deve ser rejeitada (ESTATCAMP, 2013).

Como os dados não estão normalizados, o teste *Wilcoxon Signed-Rank* foi aplicado sobre esses dados para verificar as hipóteses da Q2 do experimento (Seção 5.2.1.2). O resultado desse teste foi um P-valor = 0,0235 < 0,05, então, com nível de confiança de 95%, existem evidências de diferença entre o número de problemas nas

aplicações desenvolvidas utilizando a especialização de classes e utilizando a DSL. Portanto, a hipótese nula ($H2_0$) foi refutada e a $H2_1$ foi aceita, pois $p_{DSL} < p_{Especialização}$.

5.2.4 Ameaças à Validade do Experimento

Algumas ameaças à validade desse experimento podem ser consideradas. Pode ser argumentado que a abordagem foi testada com poucos participantes. Porém, esses participantes são desenvolvedores, que trabalham em uma empresa real, que participam direta ou indiretamente do desenvolvimento do sistema de gerenciamento integrado de redes. Portanto, são participantes que estão aptos a realizar o experimento e fazer comentários pertinentes com relação às duas abordagens aplicadas.

5.3 Considerações Finais

Neste capítulo foi apresentado o experimento realizado para avaliar a DSL desenvolvida neste projeto de mestrado. Esse experimento foi previamente planejado com base nos objetivos pretendidos, nos recursos a serem utilizados, nos participantes e nas variáveis. Os testes estatísticos foram aplicados para obter maior confiabilidade nos resultados obtidos.

O objetivo do experimento foi o de analisar se o uso da DSL para especificar uma nova tecnologia de rede implica em redução do esforço quando comparado com a implementação realizada por meio da especialização das classes para fazer a mesma função no sistema. Além disso, se existe redução do número de problemas encontrados utilizando a DSL comparado com a especialização das classes.

Na análise descritiva dos dados, pode-se dizer que houve redução no tempo gasto pelos participantes na especificação da nova tecnologia de rede quando foi utilizada a DSL em comparação com a especialização das classes. Aproximadamente, 54,41% do tempo total do experimento foram gastos com a especialização da classe e 45,59% com a DSL. Contudo, quando foi realizada a análise estatística sobre os dados coletados com o experimento para verificar as hipóteses da Q1, utilizando o *Paried T-Test*, o resultado foi um P-valor = 0,1474, ou seja, que não há diferença entre o tempo gasto para desenvolver as aplicações utilizando a especialização das classes ou utilizando a DSL. Portanto, a hipótese nula ($H1_0$) foi aceita.

Ao analisar o tempo despendido individualmente por cada um dos participantes do experimento para desenvolver a aplicação, observa-se que o tempo gasto com a especialização de classes foi maior ou igual ao tempo quando a DSL foi utilizada. Uma possível causa para esse resultado pode ser a diferença do nível de conhecimento dos participantes do experimento obtido no Formulário de Caracterização de Participante (Apêndice A). Além disso, pode ser que foram considerados poucos participantes para o experimento. Outro dado observado foi que, para os participantes que têm a função de implementadores, embora tenha ocorrido redução do tempo, este foi menor se comparado com os demais participantes, cujas funções são diferentes no desenvolvimento do sistema. Infere-se que uma causa para esse resultado seja a familiaridade que os implementadores têm com o código enquanto que os demais não possuem a mesma experiência.

Estatisticamente, não pode ser considerada a hipótese ($H1_1$), mas individualmente é relevante o resultado. Pois, o tempo despendido com a aplicação utilizando a DSL foi menor ou igual se comparado com a utilização da especialização das classes, enfatizado pelos comentários dos participantes relatados no Formulário de Coleta de Dados (Apêndice A) que a DSL apresenta facilidade de uso e visão ampla da nova tecnologia de rede.

Com relação aos erros, a ocorrência dos mesmos foi observada nas aplicações desenvolvidas utilizando especialização das classes por erro de digitação das especificações da nova tecnologia de rede e erro de código Java. Os comentários dos participantes relatados no Formulário de Coleta de Dados (Apêndice A) com relação ao número de problemas ressaltam a facilidade do uso da DSL comparada com a necessidade de escrever o código Java das classes derivadas. O modelo criado com a utilização da DSL proporciona uma visão mais ampla da nova tecnologia de rede, enquanto que para escrever as classes derivadas é necessário conhecer a linguagem de programação Java, para implementar os blocos de código para cada grupo de configuração e seus respectivos atributos.

A análise estatística sobre os dados coletados com o experimento para verificar as hipóteses da Q2 do experimento, utilizando o *Wilcoxon Signed-Rank*, o resultado foi um P-valor = 0,0235, ou seja, que há diferença entre o número de problemas nas aplicações desenvolvidas utilizando a especialização de classes e utilizando a DSL. Portanto, a hipótese $H2_1$ foi aceita, ou seja, $p_{DSL} < p_{Especialização}$.

Capítulo 6

CONCLUSÃO

6.1 Resultados Obtidos

O objetivo deste projeto de mestrado foi o de elaborar um apoio computacional para facilitar a especificação de uma nova tecnologia de rede em um sistema de gerenciamento integrado de redes desenvolvido em uma empresa real. A maneira usada para alcançar esse objetivo foi aplicar os conceitos de MDD, com a criação de uma DSL, que substitui a especialização de classes utilizada no desenvolvimento deste sistema. A DSL foi desenvolvida utilizando a ferramenta *open-source* EMF (ECLIPSE MODELING FRAMEWORK, 2013) para especificar o metamodelo, que contém as regras de negócio para especificar uma nova tecnologia de rede, e a ferramenta Acceleo (ACCELEO, 2013), para transformar o modelo em código Java por meio de *templates*. Ambas as ferramentas dispõem de recursos para a geração de *plug-ins* que facilitam o uso da DSL.

Um experimento foi conduzido com dez participantes, que trabalham na empresa real e que conhecem o sistema de gerenciamento integrado de redes, para avaliar a eficiência e o número de erros encontrados para especificar uma nova tecnologia de rede utilizando a DSL desenvolvida e a especialização de classes, sendo este último o processo de desenvolvimento utilizado atualmente pelos desenvolvedores deste sistema na empresa real.

O resultado obtido no experimento com relação à eficiência, quando feita a análise descritiva, indicou que houve uma redução no tempo gasto pelos participantes na especificação da nova tecnologia de rede quando foi utilizada a DSL

em comparação com a especialização das classes. Porém, devido ao resultado do P-valor do *Paried T-Test*, estatisticamente, não foi possível concluir que o uso da DSL permitiu um desenvolvimento mais rápido e com menor custo, se comparado com a utilização da especialização das classes.

O resultado obtido no experimento com relação ao número de erros na especificação da nova tecnologia de rede indicou que eles ocorreram somente quando foi utilizada a especialização de classes, ou por erro de digitação ou erro de código Java.

Os relatos dos participantes do experimento enfatizaram a facilidade do uso da DSL, tanto na especificação da nova tecnologia de rede e na ampla visão de requisito que a DSL proporcionou, quanto na geração do código Java a partir do modelo especificado sem ter a preocupação com a escrita do código Java.

6.2 Contribuições da Proposta

As contribuições deste projeto de mestrado são:

- Construção da DSL para a especificação de uma nova tecnologia de rede para a gerência de configuração do sistema de gerenciamento integrado de redes. A DSL desenvolvida pode ser adicionada ao projeto de demonstração do sistema de gerenciamento integrado de redes. Este sistema pode mostrar a descoberta de um elemento de rede de uma nova tecnologia que o sistema ainda não gerencia, mas que o cliente possui na sua rede de telecomunicações;
- Maior facilidade para descobrir um elemento de uma nova tecnologia de rede: o uso da DSL desenvolvida facilita a especificação de uma nova tecnologia de rede e os novos elementos desta tecnologia podem ser descobertos e passam a ser gerenciados pelo sistema de gerenciamento integrado de redes;
- Protege os desenvolvedores da linguagem de programação da plataforma: o desenvolvedor faz a especificação da nova tecnologia de rede por meio do modelo da DSL e solicita a geração do código Java a partir do modelo especificado. Dessa forma, o desenvolvedor pode se dedicar mais às

regras de negócio e não se preocupar diretamente com o código-fonte e com as complexidades e as particularidades requeridas para fazer a implementação do código para uma determinada plataforma;

- Desenvolvimento com menos número de erros: devido à geração automática do código Java a partir do modelo, a inserção de erros de digitação no código é praticamente inexistente;
- Validação dos modelos das aplicações: os modelos criados nas aplicações são construídos conforme o metamodelo que contém as regras de negócio do domínio. Dessa forma, as regras de negócio são validadas já na criação dos modelos;
- Maior confiabilidade nas aplicações desenvolvidas: a possibilidade de existirem erros no código Java gerado a partir do modelo é menor do que quando implementado manualmente, pois os *templates* utilizados na geração do código foram previamente testados;
- Apresentação de novos conceitos de desenvolvimento de software à equipe de desenvolvimento de software: o desenvolvimento da DSL contribuiu para apresentar para a equipe de desenvolvimento de software conceitos de MDD, novas ferramentas e enfatizar a importância de conhecer o domínio de negócio para desenvolver um sistema com grande potencial de reúso.

6.3 Limitações da Proposta

Para aplicar os conceitos de MDD e desenvolver a DSL, foi selecionada a função específica de tecnologia que compõe as funções da gerência de configuração do sistema de gerenciamento integrado de redes de uma empresa real. Este sistema possui muitas funções (comuns e específicas) e está em contínua evolução em razão das solicitações de outras funções advindas de vários clientes. Assim sendo, foi necessário estabelecer um *baseline* do sistema sobre o qual este projeto de mestrado foi desenvolvido. Foi escolhida uma versão do sistema que ainda podia ser executada utilizando o banco de dados MySQL, pois a versão mais

atual do sistema é executada somente utilizando o banco de dados Oracle. O escopo foi delimitado na especificação de uma nova tecnologia de rede.

Dessa forma, a aplicação dos conceitos de MDD por meio do desenvolvimento da DSL não abrangeu todo o sistema de gerenciamento integrado de redes. Porém, essa experiência agregou de forma positiva apontando uma possibilidade de desenvolver DSL para conjuntos de funções afins de um sistema em produção. Além disso, com o experimento foi possível observar algumas vantagens da utilização da DSL e, perceber a necessidade e a importância da constante atualização do conhecimento dos desenvolvedores com relação às novas tecnologias

Ainda com a utilização da DSL, que aumenta o nível de abstração para a especificação dos requisitos, o desenvolvimento necessita fortemente dos conhecimentos do especialista de redes para especificar novas tecnologias de rede.

Para a utilização da DSL, desenvolvida com ferramentas da plataforma Eclipse, é necessário que o desenvolvedor conheça o ambiente de desenvolvimento Eclipse.

6.4 Trabalhos Relacionados

Santosh e Madanagopal, 2009, propuseram um framework Agente Proxy Genérico para facilitar o gerenciamento de elementos heterogêneos de rede. Este framework é composto por dois componentes, um agente proxy SNMP e um dispositivo de mediação. O dispositivo de mediação é um módulo de software separado que se comunica com os elementos da rede convertendo as requisições SNMP em mensagens de protocolo proprietário e vice-versa. Um agente proxy realiza a tradução das requisições SNMP em requisições não SNMP e vice-versa. O framework Agente Proxy Genérico SNMP opera com qualquer tipo de elemento de rede (NE) de diferentes fornecedores, equipado com os seus protocolos proprietários. Ele também fornece a flexibilidade de adicioná-los na rede. O framework desenvolvido assegura uma comunicação contínua e gerenciamento de vários NEs heterogêneos. O framework mantém a lista de MIBs e permite que essa lista e seus OIDs se registrem quando necessário no agente proxy.

Ma-kun Guo et al., 2010, propuseram uma espécie de sistema de gerenciamento de rede baseada em Extensible Markup Language (XML), que tem a vantagem de ser plataforma livre, ter eficiência e flexibilidade para resolver problemas relacionados com a gestão de configuração e processos de desenvolvimento eficiente de aplicativos. Esta abordagem propõe um tipo de modelo de transformação de MIB SNMP para uma visão XML e um sistema de gerenciamento de rede comum. A tecnologia do sistema pode ser usada como um sistema de gerenciamento de rede e ser uma referência de desenvolvimento de software. Com o rápido desenvolvimento da rede, escalabilidade SNMP, eficiência, segurança e outras questões, fazem da tecnologia de gerenciamento de rede baseado em XML é uma oportunidade para o desenvolvimento.

Nossa abordagem foi desenvolvida em um sistema de gerenciamento inegrado de redes de uma empresa real. Este sistema tem como objetivo gerenciar diferentes tipos de redes e tecnologias, tais como: IP, IP/MPLS (*Multiprotocol Label Switching*), WiMAX, GPON, redes de transporte (WDM, SDH (*Synchronous Digital Hierarchy)*, PDH (Pl*esiochronous Digital Hierarchy)*, modems ópticos e SHDSL (*Symmetric High-Speed Digital Subscriber Line*)) e de infra-estrutura de telecomunicações de plantas. Para comunicar-se com os elementos de rede de diferentes tecnologias e fornecedores, por meio de SNMP ou sistema de gerência proprietária, o sistema de gerenciamento integrado de redes tem mediadores responsáveis para converter diferentes protocolos de comunicação em um comum, considerado padrão pelo sistema. Usando abordagem DSL é possível especificar novas tecnologias de rede em um alto nível de abstração, ou seja, em um modelo. A vantagem é a geração automática do código a fim de que os elementos de uma nova tecnologia possam ser gerenciados pelo sistema. Depois disso, a gerência de falhas é capaz de executar as regras de correlação de alarmes, independentemente da tecnologia, para encontrar a causa raiz do problema de rede, e desta forma ajudar a reparar a rede, a fim de torná-la operacional novamente.

6.5 Trabalhos Futuros

O desenvolvimento da DSL se restringiu a algumas funções da gerência de configuração. Dessa forma, uma proposta de trabalhos futuros seria analisar as funções do sistema de gerenciamento integrado de redes e propor o desenvolvimento de outras DSL até que todo o sistema fosse migrado para desenvolvimento orientado a modelos.

O desenvolvimento da DSL poderia ser feito utilizando outros recursos das ferramentas relacionadas à modelagem e às tecnologias MDD para deixar a DSL com uma interface mais gráfica.

REFERÊNCIAS

ACCELEO. 2013. Disponível em: <http://www.eclipse.org/acceleo/>. Acesso em: 23 de maio de 2014.

ACCELEO. Acceleo User Tutorial. 2006. Disponível em: http://www.acceleo.org/doc/obeo/en/acceleo-2.6-user-tutorial.pdf. Acesso em: 20 de maio de 2014.

ACCELEO. Acceleo Reference. 2006. Disponível em: http://www.acceleo.org/doc/obeo/en/acceleo-2.6-reference.pdf. Acesso em: 20 de maio de 2014.

ACCELEO. Disponível em: https://wiki.eclipse.org/Acceleo/Getting_Started. Acesso em: 20 de maio de 2014.

ACCELEO. How to create an Eclipse menu executing Acceleo trasnformations. Disponível em: http://www.youtube.com/watch?v=GO0pC3p0iFg. Acesso em: 20 de maio de 2014.

AMBLER, S. W. Agile model driven development is good enough. IEEE Software, v. 20, n. 5, p. 71–73, 2003.

ANT. 2009. Disponível em: <http;//ant.apache.org/>. Acesso em maio de 2014.

ANTKIEWICZ, M.; CZARNECKI, K.; STEPHAN, M. Engineering of Framework-Specific Modeling Languages. IEEE Transactions on Software Engineering, v. 35, n. 6, p. 795-823, Dezembro 2009.

ANTKIEWICZ, M.; CZARNECKI, K. Framework-specific modeling languages with round-trip engineering. In: NIERSTRASZ, O. et al. (Ed.). Model Driven Engineering

Languages and Systems (MoDELS 2006). [S.l.]: Springer Berlin / Heidelberg, 2006. (Lecture Notes in Computer Science, v. 4199/2006), p. 692–706.

AUGUST, J. H. Joint Application Design**:** the group session approach to systems design. Englewood Cliffs, N. J.: Prentice-Hall, 1991.

BAHNOT, V; ROMAN, A; TRASK, B; CORPORATION, P. Using Domain-Specific Modeling to Develop Software Defined Radio. In: OOPSLA on Domain-Specific Modeling, 5th, 2005, San Diego, Proceedings… 2005, p. 33–42.

BAUER, Veronika. Facts and Fallacies of Reuse in Practice. Technische Universität München, Germany. Disponível em: http://www4.in.tum.de/~bauerv/docs/bauer2013docsym.pdf, 2013. Acesso em: 20 de junho de 2014.

BÉZIVIN, J. On the Unification Power of Models, Software and System Modeling. SoSym Journal, 4(2); 171-188, 2005. Disponível em: http://www.sciences.univnantes.fr/lina/atl/www/papers/OnTheUnificationPowerOfModels.pdf.

BUDINSKY, F. Eclipse Modelling Framework: Developer's Guide. Addison-Wesley, 2003.

CASTRO, Maria Cristina Felippeto. Sistemas Wireless e Padrões. 2009. Pontifícia Universidade Católica do Rio Grande do Sul, Departamento de Engenharia elétrica, Rio Grande do Sul.

CLEAVELAND, J. C. Building application generators. IEEE Software, v. 7, n. 1, p. 25–33, 1988.

CLEMENTS, P.; NORTHROP, L. Software Product Lines: Practices and Patterns. [S.l.]: Addison-Wesley, 2002.

CMMi. Disponível em: http://www.sei.cmu.edu/library/assets/whitepapers/cmmi-dev_1-2_portuguese.pdf, 2009. Acesso em: 22 de maio de 2014.

CPqD. CPqD Gerência Integrada de Telecom. Disponível em: <https://www.cpqd.com.br/solucoes-e-produtos/197-cpqd-gerencia-integrada-telecom.html>, 2009. Acesso em: 22 de maio de 2014.

CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; DAVIN, J. 1990. RFC 1157 - A Simple Network Management Protocol (SNMP)

CUADRADO, J.; MOLINA, J. A Model-Based Approach to Families of Embedded Domain-Specific Languages. IEEE Transactions on Software Engineering, v. 35, n. 6, p. 825-840, Dezembro 2009.

D'ÁVILA, César Kyn. LTE: Long Term Evolution – Arquitetura Básica e Acesso Múltiplo. 2009. Disponível em: http://www.cedet.com.br/index.php?/Tutoriais/Telecom/lte-long-term-evolution-arquitetura-basica-e-acessomultiplo.html.

DEURSEN, A. v.; KLINT, P.; VISSER, J. Domain-specific languages: An annotated bibliography. SIGPLAN Notices - ACM Press, v. 35, n. 6, p. 26–36, 2000.

DJUKIC, V.; LUKOVIC, I.; POPOVIC, A. Domain-Specific Modeling in Document Engineering. In: Federated Conference on Computer Science and information Systems – FedCSIS. Szczecin, Polônia, 2011. Proceedings… Washington: IEEE Computer Society 2010. p. 817-824.

DURELLI, R. S. Uma abordagem apoiada por linguagens específicas de domínio para criação de linhas de produtos de software embarcado, UFSCar, 2011.

ECLIPSE. Eclipse Modeling Project. 2013. Disponível em: <http://www.eclipse.org/modeling/>. Acesso em: 20 de maio de 2014.

ECLIPSEIDE. Eclipse. 2013. Disponível em: <http://www.eclipse.org/>. Acesso em: 20 de maio de 2014.

EMF. Eclipse Modeling Framework. 2013. Disponível em: <http://www.eclipse.org/modeling/emf/>. Acesso em; 20 de maio de 2014.

ESTATCAMP (2013). Portal Action. Disponível em HTTP://www.portalaction.com.br. Acesso em; 20 de junho de 2014.

FILHO, Wilson de Pádua Paula. Engenharia de software Fundamentos, Métodos e Padrões. Rio de Janeiro, RJ. Ed LTC, 2001.

FMT. Feature Modeling Tool. 2013. Disponível em: <http://www.microsoft.com/visualstudio/en-us/>. Acesso em: 23 de maio de 2014.

FMP. Feature Modeling Plug-in for Eclipse. Disponível em: http://sourceforge.net/projects/fmp/. Acesso em: 23 de maio de 2014.

FOWLER, M. Language Workbenches: The Killer-App for Domain Specific Languages? [S.l.]: martinfowler.com, 2005. Disponível em: <http://www.martinfowler.com/articles/languageWorkbench.html>. Acesso em: 14 de junho de 2014.

FOWLER, M. et al. Refactoring: Improving the Design of Existing Code. [S.l.]: Addison Wesley, 1999.

FRAKES,W. B.; ISODA, S. Success factors of systematic software reuse. IEEE Software, v. 11, n. 01, p. 14–19, 1994.

FRANCE, R.; RUMPE, B. Model-Driven Development of Complex Software: A Research Roadmap. In: 29th INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING – FUTURE OF SOFTWARE ENGINEERING – ICSE. Minneapolis, USA, 2007. Proceedings… Washington: IEEE Computer Society 2007.p.37-54.

GARCÍA-MAGARIÑO, Iván *et al.*. Guideline for the definition of EMF metamodels using na Entity-Relationship approach. Information and Software Technology, Volume 51, Issue 8 (August 2009), p. 1217-1230. 2009. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0950584909000251>.

GEF. Generic Modeling Environment. GME, 2013. Disponível em: < http://www.eclipse.org/gef/>. Acesso em: 18 de maio de 2014.

GME. Generic Modeling Environment. GME, 2013. Disponível em: <http://www.isis.vanderbilt.edu/projects/gme>. Acesso em: 18 de maio de 2014.

GMF. Generic Modeling Framework. GMF, 2013. Disponível em: < http://wiki.eclipse.org/Graphical_Modeling_Framework>. Acesso em: 18 de maio de 2014.

GRISS, M. Making software reuse work at hewlett-packard. IEEE Software, v. 12, n. 01, p.105–107, 1995.

GRISS, M.; FAVARO, J.; D'ALESSANDRO, M. Integrating feature modeling with the RSEB. In: The Fifty International Conference on Software Reuse (ICSR). Victoria, Canada: IEEE/CS Press, 1998. p. 76–85.

GRONBACK, Richard C. Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Professional, 2009.

HARNEDY, S. 1997. Total SNMP: exploring the Simple Network Management Protocol", 2º edição, Prentice Hall.

HARRISON, N. B.; AVGERIOU, P.; ZDUN, U. Using Patterns to Capture Architectural Decisions. IEEE Software, v.24, n. 4, p. 38-45, Julho 2007.

HUTCHINSON, J; WHITTLE, J.; ROUNCCEFIELD, M.; KRISTOFFERSEN, S. Empirical Assessment of MDE in Industry. In: *33rd International Conference on*

Software Engineering – ICSE. INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING – ICSE. Honolulu, HI, EUA, 2011. Proceedings…New York: ACM. p. 471-480.

JACOBSON, I. The Unified Software Development Process. Addison-Wesley, 1999.

JACOBSON, I.; GRISS, M.; JONSSON, P. Reuse-driven Software Engineering Business (RSEB). [S.l.]: Addison-Wesley, 1997.

KENDALL, K.E. Kendall, J.E. Kendall; Systems Analysis and Design, Prentice Hall, 1992.

KENT, S. Model Driven Engineering. Proceedings of IFM2002, LNCS 2335 (pp. 286-298). Springer. 2002.

KLEPPE, A.; WARMER, J.; BAST, W. MDA Explained: The Model Driven Architecture: Practice and Promise. 1. ed. Addison-Wesley Longman Publishing, 2003. 192 p.

KRUEGER, C. Software reuse. ACM Computing Surveys, v. 24, n. 02, p. 131–183, 1992.

KUROSE, James F. ROSS, Keith W. Redes de Computadores e a Internet. Addision Wesley, 2010.

LANGE, C. F. J.; CHAUDRON, M. R. V. Managing model quality in UML-based software development. In: *STEP '05: Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice*. Washington, DC, USA: IEEE Computer Society, 2005. p. 7–16. ISBN 0-7695-2639-X.

LEFFINGWELL, Dean, Don Widrig: Managing Software Requirements - A Use Case Approach, Second Edition 2003.

LINDEN, F. V. D.; SCHMID, K.; ROMMES, E. Software Product Lines In Action: The Best Industrial Practice In Product Line Engineering. [S.l.]: Springer, 2007.

LUCRÉDIO, D.; ALMEIDA, E. S. d.; PRADO, A. F. d. A survey on software components search and retrieval. In: STEINMETZ, R.; MAUTHE, A. (Ed.). 30th IEEE EUROMICRO Conference, Component-Based Software Engineering Track. Rennes - France: IEEE/CS Press, 2004. p. 152–159.

LUCRÉDIO, D. Uma Abordagem Orientada a Modelos para Reutilização de Software. 2009. Tese (Doutorado em Ciência da Computação), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2009.

LUCRÉDIO, D.; FORTES, R. P. M.; ALMEIDA, E. S.; MEIRA, S. L. Designing Domain Architectures for Model-Driven Engineering. In: IV SIMPÓSIO BRASILEIRO DE COMPONENTES, ARQUITETURAS E REÚSO DE SOFTWARE – SBCARS. Salvador, Brasil, 2010. Anais... Salvador: UFBA 2010. p. 100-109.

MAGALHÃES, A. P.; Andrade, A.; MACIEL, R. P. MTP; Model Transformation Profile. In: VII SIMPÓSIO BRASILEIRO DE COMPONENTES, ARQUITETURAS E REUTILIZAÇÃO DE SOFTWARE – SBCARS. Brasília, Brasil, 2013. Anais... Brasília. UNB 2013. P. 125-134.

MA-KUN, G., YI-MIN, Y., MIN, W., QI, Y., (2010). Research and Implementation of Network Management System Based on XML View. In: *International Conference on Logistics Engineering and Intelligent Transportation Systems (LEITS)*, IEEE.

MAURO, D., SCHMIDT, K., (2001). Essencial SNMP. O`Reilly.

MCILROY, M. D. ’Mass produced’ software components. In: NATO Software Engineering Conference. [S.l.: s.n.], 1968. p. 138–155.

McCLOGHRIE; ROSE, M. 1990. RFC 1156 - Management Information Base for Network Management of TCP/IP-based internets

McCLOGHRIE; ROSE, M. 1991. RFC 1213 - Management Information Base for Network Management of TCP/IP-based internets: MIB-II

MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T. Model-Driven Development. IEEE Software, v.20, n.5, p. 14-18, Setembro 2003.

MERNIK, M.; HEERING, J.; SLOANE, A. M. When and How to Develop Domain-Specific Languages. ACM Computing Surveys, v.19, n. 7 p. 316-344, 2005.

MILI, H. et al. Reuse-Based Software Engineering: Techniques, Organization, and Controls. [S.l.]: John Wiley & Sons, 2002.

MOORE, B. et al. Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework. [S.l.]: ibm.com/redbooks, 2004.

MORRIS, S. Network Management, MIBs and MPLS: Principles, Design and Implementation. Prentice Hall, 2003.

NEIGHBORS, J. M. Software Construction Using Components. Tese (Ph.D. Thesis) - University of California at Irvine, 1980.

OLIVEIRA, I. P.; RAMOS, L. A.; GUEDES, R. M. Concepção de uma Ferramenta Parser para Extração de dados do metamodelo ECore. In: X ENCONTRO ANUAL DE COMPUTAÇÃO – EnAComp 2013. Disponível em: http://www.aptor.com.br/enacomp2013/pdf/34.pdf. Acesso em: 20 de junho de 2014.

OLIVEIRA, J. B. Uma Abordagem Baseada em Engenharia Dirigida por Modelos para Suportar o Teste de Sistemas de Software na Plataforma de Computação em Nuvem. 2012. Dissertação de Mestrado em Engenharia da Eletricidade, Universidade Federal do Maranhão, 2012. Disponível em: http://www.tedebc.ufma.br/tde_arquivos/10/TDE-2013-04-03T163947Z-759/Publico/Dissertacao%20Jessica%20Bassani.pdf. Acesso em: 20 de junho de 2014.

OMG. Object Management Group. Disponível em: www.omg.org. Acesso em: 18 de maio de 2014.

OMG. MDA Guide Version 1.0.1. [S.l.], 2003. Disponível em: <http://www.omg.org>. Acesso em: 14 de junho de 2014.

OMG. MOF 2.0 Query / Views / Transformations Final Adopted Specification. [S.l.], 2005. Disponível em: <http://www.omg.org>. Acesso em: 14 de junho de 2014.

OMG. Catalog of UML Profiles Specifications. 2006. Disponível em: <http://www.omg.org>. Acesso em: 14 de junho de 2014.

OMG. Meta Object Facility Core Specification Version 2.0. [S.l.], 2006. Disponível em: <http://www.omg.org>. Acesso em: 14 de junho de 2014.

OMG. XML Metadata Interchange (XMI) Specification. [S.l.], 2006. Disponível em: <http://www.omg.org>. Acesso em: 14 de junho de 2014.

OMG. Unified Modeling Language (OMG UML) Infrastructure. [S.l.], 2007. Disponível em: <http://www.omg.org>. Acesso em: 14 de junho de 2014.

OMG M2T. MOF Model to Text Transformation Language. 2008. Disponível em: <http://www.omg.org>. Acesso em 14 de junho de 2014.

PARNAS, D. L. On the design and development of program families. IEEE Transactions on Software Engineering, v. 2, n. 1, p. 1–9, mar. 1976.

PETERS, James F.; Pedrycz, Wiltold. Engenharia de Software: Teoria e Prática. Rio de Janeiro: Campus, 2001.

PIERS, W. and BRUNELIERE, H. (2012). ATL Concepts. Disponível em: <http://wiki.eclipse.org/ATL/Concepts>. Acesso em: 10 de junho de 2014.

PINHEIRO, José M. S. Redes Ópticas de Alto Desempenho. Disponível em: http://www.projetoderedes.com.br/artigos/artigo_redes_opticas_alto_desempenho.php . Acesso em: 22 de maio 2013.

PRESSMAN, R. S. Software Engineering: A Practitioner's Approach. 7th ed. New York: McGraw-Hill, 2011.

RUMPE, B.; SCHINDLER, M.; VÖLKEL, S.; WEISEMÖLLER, I. Generative Software Development. *In: 23nd ACM/IEEE International Conference on Software Engineering.* Cape Town, África do Sul, 2010. Proceedings… Washington: IEEE Computer Society 2010. p. 473-474.

SAMETINGER, J. Software Engineering with Reusable Components. Berlin Heidelberg: Springer-Verlag, 1997.

SANCHEZ, William Penha. "PON: Redes Ópticas de Acesso de Baixo". Disponível em: <http://www.teleco.com.br/tutoriais/tutorialpon/default.asp>. Acesso em: 22 de maio de 2014.

SARASA-CABEZUELO, A.; TEMPRADO-BATTAD, B.; CEREZO, D. R.; SIERRA, J. Building XML-Driven Application Generators with Compiler Construction Tools. Computer Science and Information Systems, v. 9, n. 2, p. 485-504, Junho 2012.

SCHMIDT, D. Guest Editor's Introduction; Model-Driven Engineering. IEEE Computer, v.39, n. 6, p. 25-31, Fevereiro 2006.

SANTOSH, S. CHAVAN and MADANAGOPAL, R. 2009. Generic SNMP Proxy Agent Framework for Management of Heterogeneous Network Elements. In: *First International Conference on COMmunication Systems And NETworks*. Pages 331-336. IEEE Press Piscataway, NJ, USA.

SILVA, A. G. P. Uma Abordagem Dirigida por Modelos para Desenvolvimento de Middlewares Auto-Adaptativos para Transmissão de Fluxo de Dados Baseado em

Restrições de QoS. 2010. Dissertação de Mestrado em Sistemas e Computação, Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, 2010.

SILVA, André G. P. et al. A Model for the Transmission of Multimedia Data Flow Based on the Use of Shared Memory Mechanism. *In: 15 International Symposium on Frontiers of Information Systems and Network Applications (FINA 2009),* 2009, Bradford. International Conference on Advanced Information Networking and Applications Workshops, 2009. WAINA '09, 2009. p. 84-89.

SIMOS, M. et al. Organization Domain Modeling (ODM) Guidebook Version 2.0. [S.l.], 1996. STARS Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defense Systems, Manassas VA.

SOMMERVILLE, Ian. Engenharia de Software. 9ª ed. São Paulo: Pearson Addison Wesley, 2011.

SORTICA, Eduardo. Redes de Telecomunicações, TMN e Gerência Integrada de Redes e Serviços. 3IT, Rio de Janeiro: 1999.

STARS. Software Technology for Adaptable Reliable Systems (STARS). The Reuse-Oriented Software Evolution (ROSE). [S.l.], 1993. Unisys STARS Technical Report,Advanced Research Projects Agency (ARPA) STARS Technology Center, 801 N. Randolph St. Suite 400, Arlington VA 22203.

StArt. State of the Art through Systematic Review, v. 1.06.2. UFSCar. Disponível em: lapes.dc.ufscar.br.

SZYPERSKI, C. Component Software: Beyond Object-Oriented Programming. [S.l.]: Addison Wesley, 1999.

SZYPERSKI, C.; GRUNTZ, D.; MURER, S. Component Software: Beyond Object-Oriented Programming - Second Edition. [S.l.]: Addison Wesley / ACM Press, 2002.

TANENBAUM, Andrew S. Redes de Computadores. Pratice Hall, 2011.

THOMAS, D. MDA: Revenge of the modelers or uml utopia? IEEE Software, v. 21, n. 3, p. 15–17, 2004.

VISUAL. VISUAL STUDIO 2010 PRODUCTS. 2013. Disponível em: <http://www.microsoft.com/visualstudio/en-us/products/2010-editions>. Acesso em: 18 de maio de 2014.

VOGEL, L. (2008). Eclipse EMF. Disponível em: <http://www.vogella.com/articles/EclipseEMF/article.html>. Acesso em: 19 de maio de 2014.

WOHLIN, C. et al. Experimentation in Software Engineering: An Introduction. [S.l.]: Kluwer Academic Publishers, 2000.

XTEND. 2009. Disponível em: http://wiki.eclipse.org. Acesso em: 20 de junho de 2014.

Apêndice A

FORMULÁRIOS E ROTEIRO DO EXPERIMENTO

Formulário de Caracterização de Participante

Nome:

1. Conhecimento sobre Linguagens de programação:

Já programou com quais linguagens de programação?_____

Você já trabalhou por pelo menos 1 ano com alguma dessas linguagens? ( ) Sim ( ) Não

2. Conhecimento sobre Java:

( ) Básico – uso de recursos comuns às linguagens de programação orientadas a objetos.

( ) Intermediário – Interface gráfica, estruturas de dados, genéricos.

( ) Avançado – programação web.

3. Conhecimento sobre Padrões de Software:

( ) Nunca utilizei.

( ) Já estudei vários padrões teoricamente, mas trabalhei pouco com eles na prática.

( ) Trabalho(ei) com padrões em aulas, projetos ou no desenvolvimento de sistemas.

4. Conhecimento sobre Frameworks:

( ) Nunca utilizei.

( ) Já estudei vários frameworks teoricamente, mas trabalhei pouco com eles na prática.

( ) Trabalho(ei) com frameworks em aulas, projetos ou no desenvolvimento de sistemas.

5. Conhecimento sobre MDD (marque uma opção em cada linha):

Entendi ( )pouco ( )bem sobre a teoria de MDD.

Entendi ( )pouco ( )bem sobre metamodelagem.

Entendi ( )pouco ( )bem sobre transformações.

6. Sobre DSL

( )Não Sei ( )Sei gerar código a partir de um modelo de uma aplicação.

( )Não Sei ( )Sei instalar a DSL do framework no Eclipse e modelar uma aplicação com ela.

( )Não Sei ( )Sei criar os modelos GMF para a DSL do framework.

Formulário de Coleta de Dados

DADOS	
NOME	
ATIVIDADE	Primeira ( ) Segunda ( )
GRUPO	A ( ) B ( )
DOMINIO	Base Station WiMax ( ) TCO RDB ( )
FERRAMENTA	Especialização das Classes ( ) DSL ( )

TEMPOS	**Início**	**Fim**
Implementação		
Execução do Sistema		

INTERRUPÇÕES	**Início**	**Fim**

OBSERVAÇÕES:
Problemas e dificuldades encontradas durante o experimento:

COMENTÁRIOS:
Pontos positivos e negativos do uso da abordagem utilizada (Ex: problemas na interface, facilidade de uso, eficiência, etc.):

Roteiro para a atividade utilizando a DSL: criar modelo para a especificação de uma nova tecnologia de rede

1) No *Windows Explorer*, ir ao diretório C:\Rosangela\Mestrado\Ferramentas\juno\eclipse
2) Dar *double click* em Eclipse para iniciar o Eclipse Modeling
3) Clicar no projeto Experimento, com o botão direito e escolher *New→Other...Example EMF Model Creation Wizards → Gi Model*, clicar no botão *Next*.
4) *File name*: NOME_DO_PROJETO_QUE_VC_QUISER.gi
5) Clicar em *Next*
6) *Model Object*: selecionar a opção Gi Configuração
7) *XML Encoding*: selecionar a opção UTF-8
8) Clicar em *Finish*
9) Abrir a pasta platform:/resource/Experimento/ NOME_DO_PROJETO_QUE_VC_QUISER.gi
10) Aparecerá GI Configuracao
11) Clicar no GI Configuracao
12) Na *view Properties* (janela que aparecerá no meio inferior da tela), digitar um *Value* para o Nome. Pode ser o nome que você quiser. Sugestão configuração.
13) Clicar com o botão direito em GI Configuracao que está na árvore para aparecer o menu de contexto, clicar em *New Child → Tecnologia*
14) Aparecerá Tecnologia na árvore.
15) Clicar na Tecnologia na árvore.
16) Na *view Properties* (janela que aparecerá no meio inferior da tela), digitar os valores para Identificação, Modelo, Nome e Tipo. Esses valores são os especificados no documento de requisitos para a atividade.
17) Clicar com o botão direito em Tecnologia que está na árvore para aparecer o menu de contexto, clicar em *New Child → Grupo Configuracao Campo*
18) Aparecerá Grupo Configuracao Campo na árvore.
19) Clicar em Grupo Configuracao Campo na árvore.
20) Na *view Properties* (janela que aparecerá no meio inferior da tela), digitar o valor para o nome do grupo configuração campo. Esse valor está especificado no documento de requisitos para a atividade.
21) Clicar com o botão direito em Grupo Configuracao Campo que está na árvore para aparecer o menu de contexto, clicar em *New Child → Atributo*
22) Aparecerá Atributo na árvore.
23) Clicar em Atributo na árvore.
24) Na *view Properties* (janela que aparecerá no meio inferior da tela), digitar os valores para o nome e a identificação do atributo. Esses valores estão especificados no documento de requisitos para a atividade.
25) ATENÇÃO: Para colocar outros grupos e atributos os passos são os mesmo descritos acima. Atenção para quando for grupo de campo ou grupo de tabela. Todos os valores estão especificados nos requisitos para a atividade.

26) Quando terminar de colocar todos os valores especificados nos requisitos para a atividade. SALVE o modelo, clicando no ícone  (*Save all*).
27) No *Project Explorer*, clicar no modelo criado NOME_DO_PROJETO_QUE_VC_DEU.gi no projeto Experimento.
28) Clicar com o botão direito para aparecer o menu de contexto. Clicar em *Acceleo Model to Text → Generate Codigo Java*.
29) No projeto Experimento, será criado um diretório chamado *src-gen* com os dois arquivos: DiscoveryNOME_DA_TECNOLOGIA.java e DiscoveryNOME_DA_TECNOLOGIASession.java
30) Selecione esses dois arquivos e os copie.
31) Ir para o Eclipse EE, no projeto DiscoveryBS-ejb, DiscoveryBS-ejb\src\com\cpqd\im\discovery\cisco
32) Colar os dois arquivos aqui.
33) Para executar o sistema e verificar o resultado, vá ao *Windows Explorer* no diretório c:\IM
34) Dê *double click* no arquivo *MySqlServer* para iniciar o servidor de banco de dados.
35) O simulador estará sendo executado em uma máquina na mesma rede onde está o computador que está executando o IM.
36) No Eclipse EE, na janela *Servers* (fica do lado esquerdo, inferior) clique o *ear*, clique com o botão direito para ser exibida o menu de contexto e clique na opção *Full Publish*.
37) Clique em *JBoss 7.1 Runtime Server* e clique sobre o ícone  para iniciar o servidor do IM.
38) Abrir um navegador e digitar a url http://localhost:8080/im-web/ <enter>
39) O IM irá apresentar a janela inicial de *login*. Digitar *root* para usuário e *public* para senha.
40) Será exibida a janela de *dashboard*.
41) Clicar no menu Administração->Rede->Descoberta
42) Clicar no botão Incluir.
43) Endereços IP: informar o IP que está nos requisitos para a atividade.
44) Protocolo: Clicar em *SNMP Default*
45) Clicar no botão Confirmar
46) Clicar na nova linha inserida na lista de configurações de descoberta
47) Clicar no botão Executar
48) Verificar que ocorrerá uma alteração no campo Data última execução.
49) Clicar no menu Inventário->IP->Nó
50) Verificar que o ip do nó descoberto aparecerá na lista de nós.
51) Clicar no IP nó descoberto
52) Clicar no botão Visualizar
53) Visualizar as informações gerais do nó na aba Geral
54) Visualizar as informações da MIB na aba MIB
55) Visualizar as informações de extensão na aba Extensões. É nessa aba que serão exibidos os grupos e os campos informados pelo usuário para o novo elemento.

Apêndice B

PLUG-INS PARA A DSL

Plug-in para o metamodelo da DSL

EMF proporciona meios para a criação de *plug-in* para facilitar a utilização da DSL especificada. Após a especificação do metamodelo em Ecore, a ferramenta EMF permite a criação de editores baseados nos modelos construídos. Um conjunto de classes Java é gerado a partir dos metamodelos, tornando possível a visualização e edição da linguagem de modelagem correspondente (GARCÍA-MARGARIÑO *et al.*, 2009). Nesta seção é apresentado o passo a passo para a criação desse *plug-in*.

Para fazer a transformação do metamodelo em código Java, utilizando a ferramenta EMF, deve-se clicar com o botão direito do *mouse* sobre o nome do projeto Ecore (no caso, *gi.ecore*), que especifica o metamodelo, e selecionar a opção Eugênia→*Generate EMF GenModel*, como ilustrado na Figura B-1. O projeto com a extensão *genmodel* é gerado no mesmo diretório do modelo Ecore.

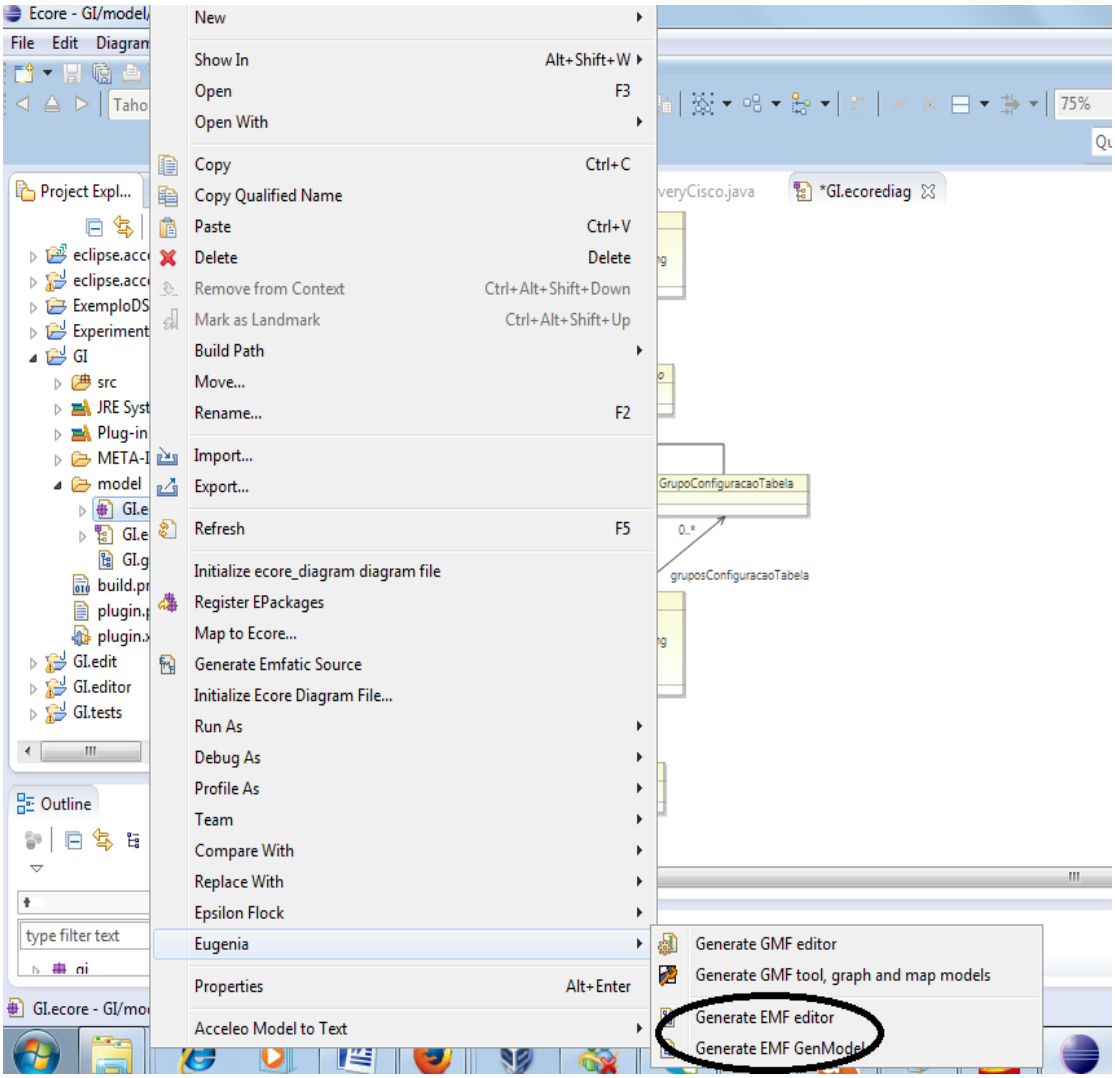

Figura B-1: Tela para gerar o GenModel e editor

O EMF também disponibiliza a geração automática de outros dois projetos chamados *.edit* e *.editor* responsáveis por classes de interface gráfica e geração de *plug-in* para o modelo Ecore. O projeto *.edit* contém as classes e as propriedades que permitem que o modelo Ecore seja exibido em visões padrões (JFace). O projeto *.editor* contém as classes com as informações necessárias para construir um editor de *plug-in* completo para o modelo Ecore com persistência em XMI.

Para gerar esses dois projetos, deve-se clicar com o botão direito do *mouse* sobre o projeto Ecore (no caso, *gi.ecore*) que especifica o metamodelo e selecionar a opção Eugênia→*Generate EMF editor*, como ilustrado na Figura B-1.

Para exportar esses dois projetos como *plug-in* do Eclipse, deve-se clicar com o botão direito do *mouse* sobre o projeto *genmodel* (no caso, *gi.genmodel*) e solicitar a exportação do *plug-in* (*Export*...), como ilustrado na Figura B-2.

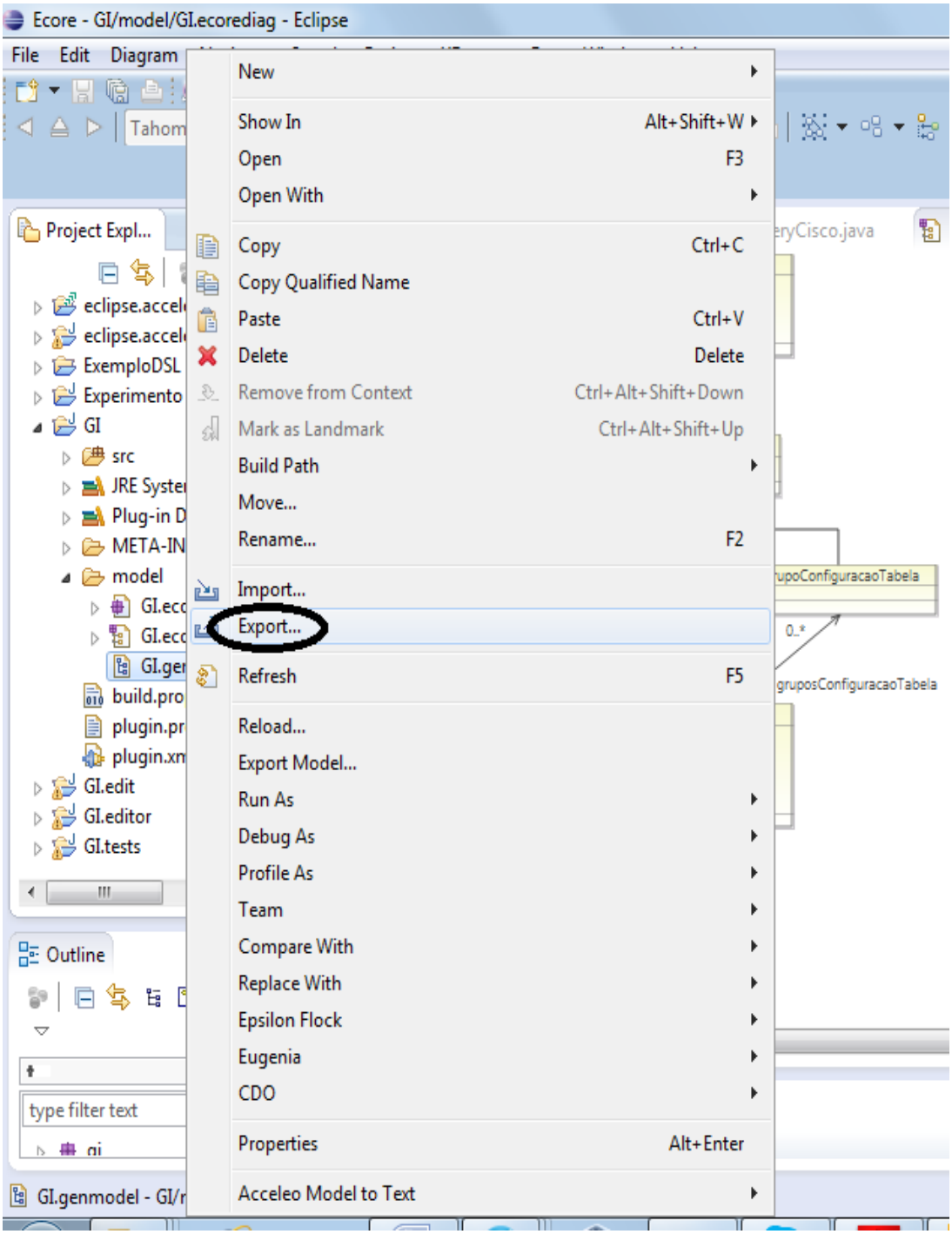

Figura B-2: Tela para solicitar a exportação dos *plug-ins Edit e Editor*

Selecionar a opção *Plug-in Development→Deployable plug-ins and fragments* e clicar em *Next*, como ilustrado na Figura B-3.

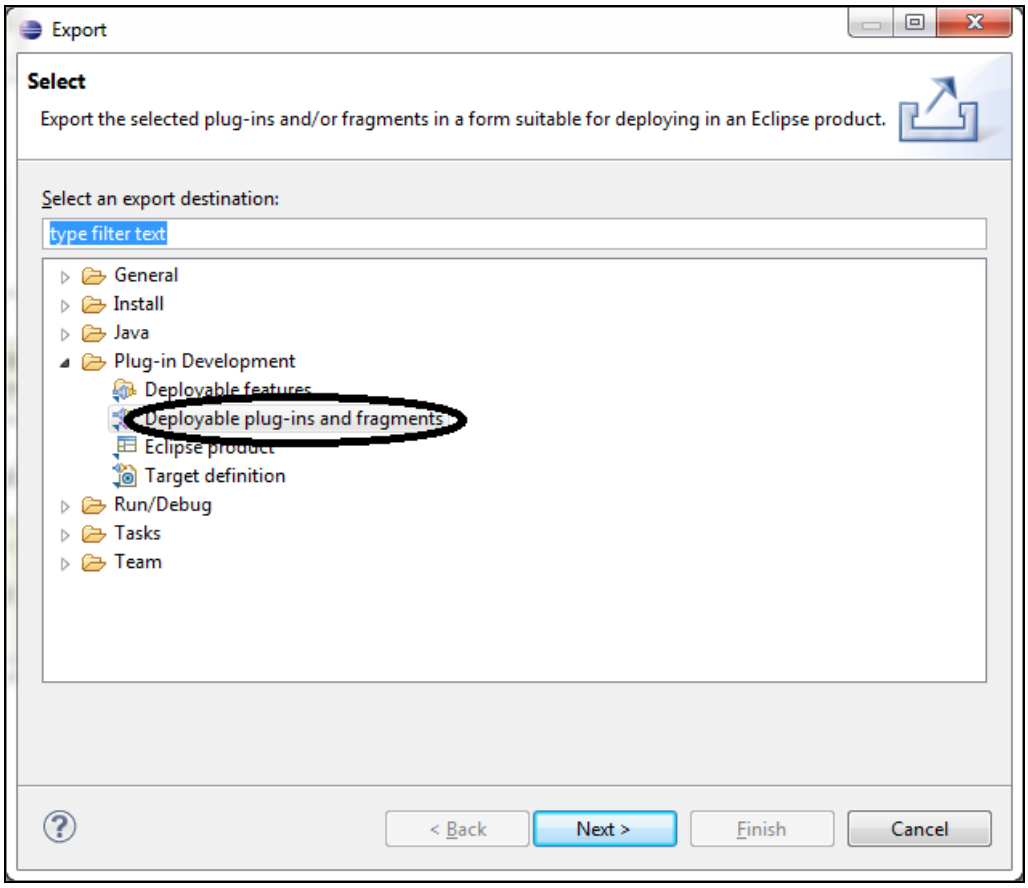

Figura B-3: Tela para selecionar *Deployable plug-ins and fragments*

Selecionar os projetos que serão exportados e que farão parte do *plug-in* do Eclipse, no caso, o *Edit* e o *Editor* (GI, *GI.edit* e *GI.editor*), e clicar em *Finish*, como ilustrado na Figura B-4.

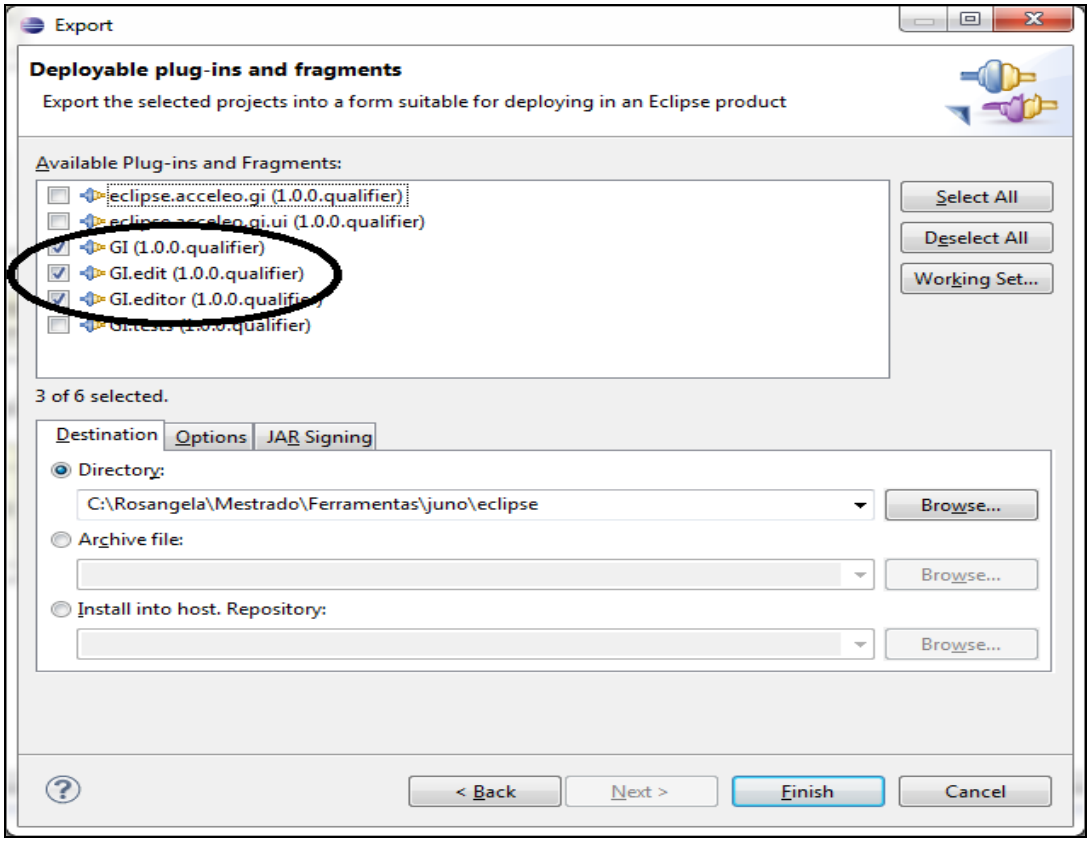

Figura B-4: Tela para selecionar os projetos GI, edit e editor

O Eclipse irá gerar os *plug-ins* no diretório de *plug-ins* do Eclipse (\eclipse\plugins).

Para o Eclipse reconhecer os *plug-ins* e carregá-los, deve-se reiniciar o Eclipse. Para isto, deve-se clicar em *File→Restart*, como ilustrado na Figura B-5.

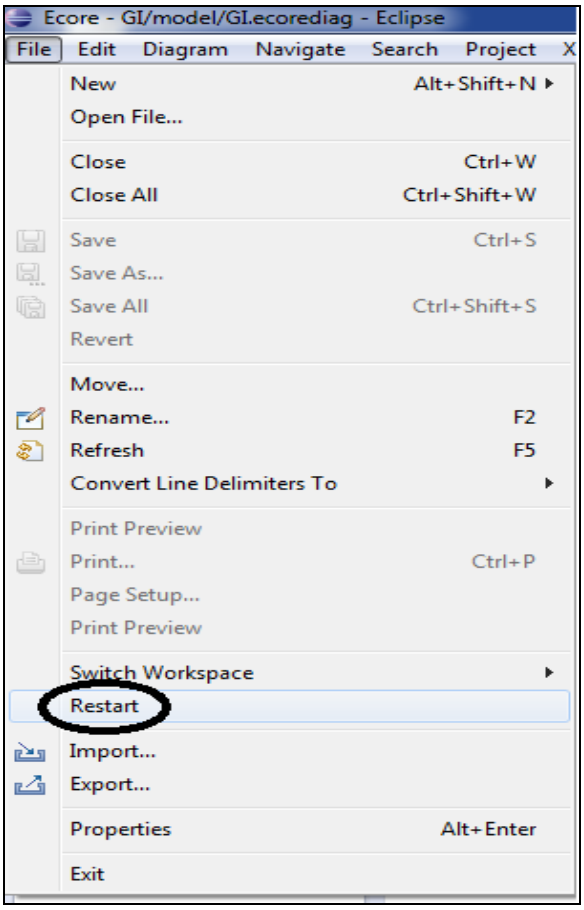

Figura B-5: Tela para solicitar *Restart* do Eclipse

O *plug-in* exportado é carregado no Eclipse, tornando-se um projeto que contém o metamodelo do domínio de negócio que pode ser usado pelo usuário para realizar a especificação de uma nova tecnologia de rede que passará a ser gerenciada pelo sistema de gerenciamento integrado de redes.

Criação de *templates* utilizando Acceleo

A perspectiva do Acceleo fornece alguns itens de menu que tem visões e ações dedicadas. Para acessar a perspectiva do Acceleo, deve-se selecionar *Windows→Open Perspective→Other*... e selecionar Acceleo no *popup*.

Para criar um projeto Acceleo, deve-se clicar em *File→New→Acceleo Project*, como ilustrado na Figura B-6.

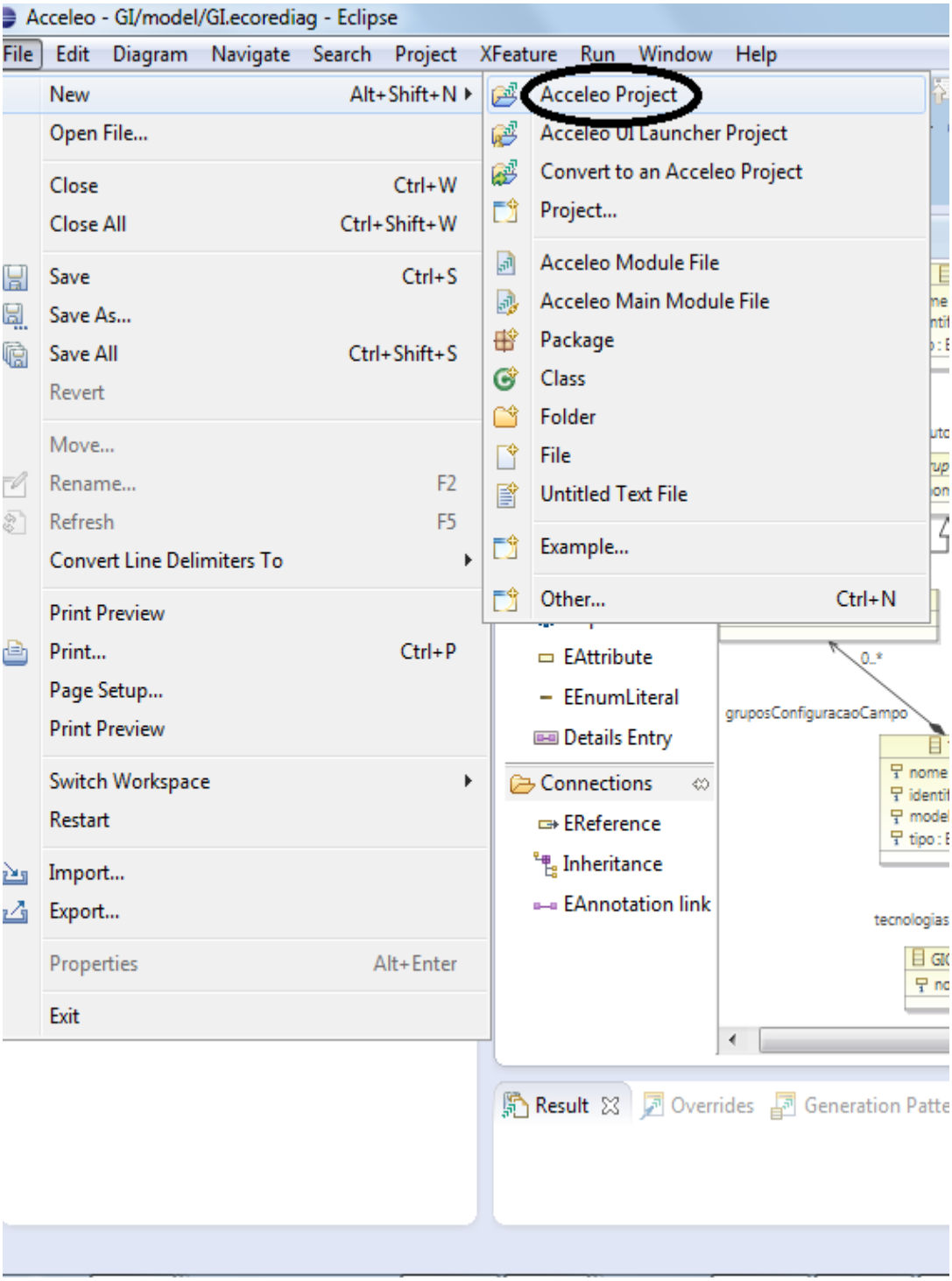

Figura B-6: Tela para criar um novo projeto Acceleo

Deve-se escolher o nome para o novo projeto e clicar em *Next*, como ilustrado na Figura B-7.

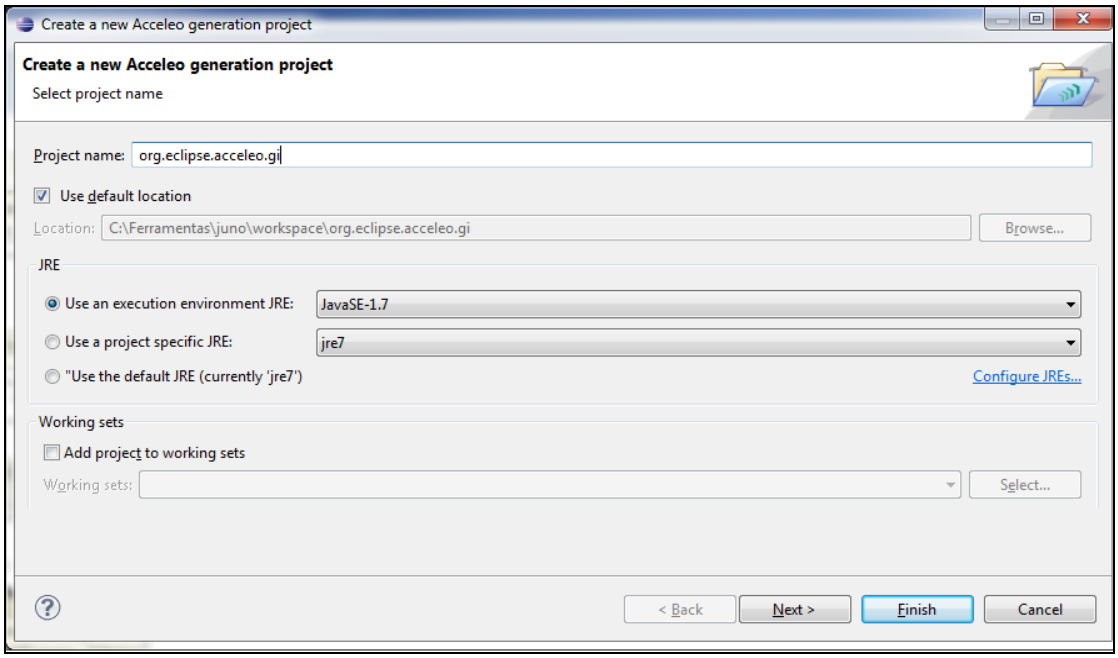

Figura B-7: Tela para nomear novo projeto Acceleo

A próxima janela *wizard* apresentada, ilustrada na Figura B-8, permite que seja iniciado o projeto criando um ou vários arquivos de módulos Acceleo. Para isso deve-se:

- Selecionar o diretório no qual o novo arquivo de módulo será criado;
- Preencher o nome do módulo;
- Selecionar o metamodelo do qual a geração do módulo fará a obtenção dos tipos;
- Escolher a metaclasse que será usada para gerar o arquivo. E, por fim, clicar em *Finish*.

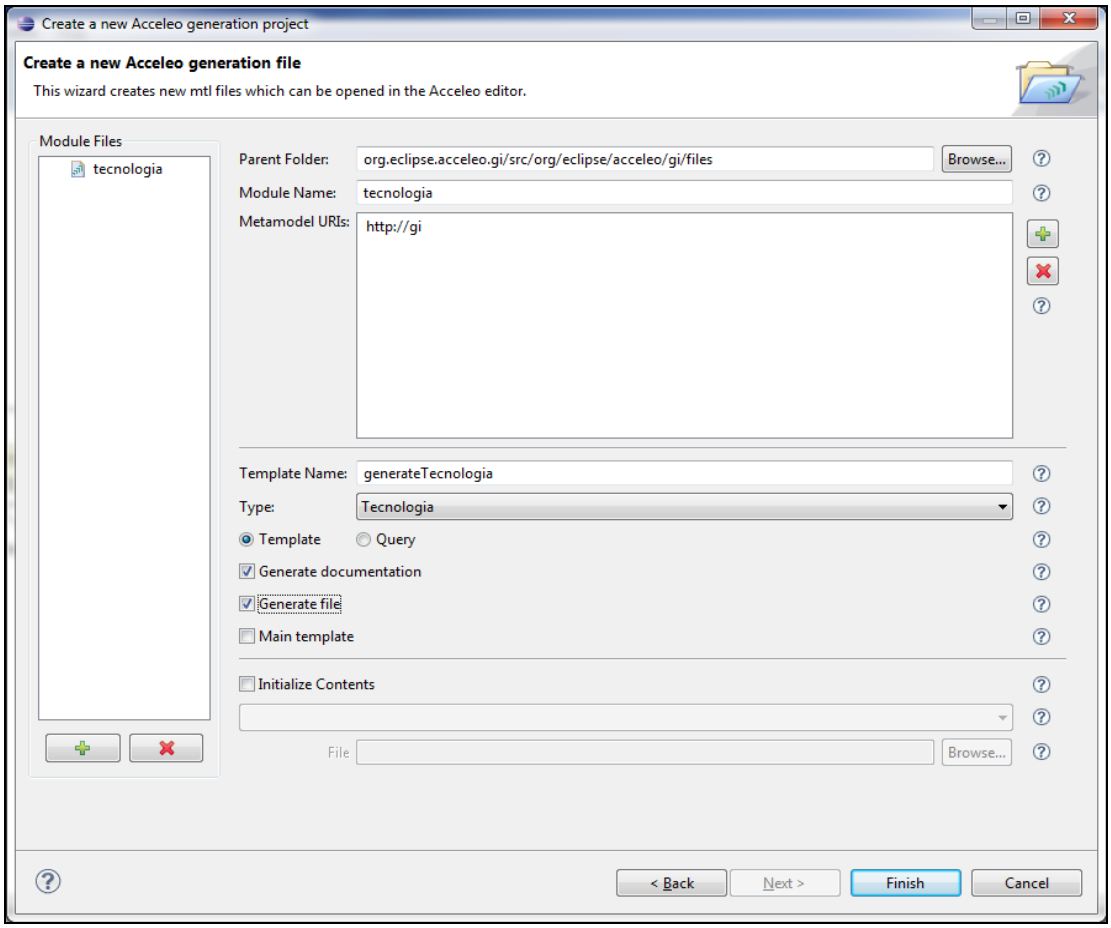

Figura B-8: Tela para criar arquivos de módulos do Acceleo

Ao clicar em *Finish*, será gerado automaticamente o arquivo do módulo, como ilustrado na Figura B-9.

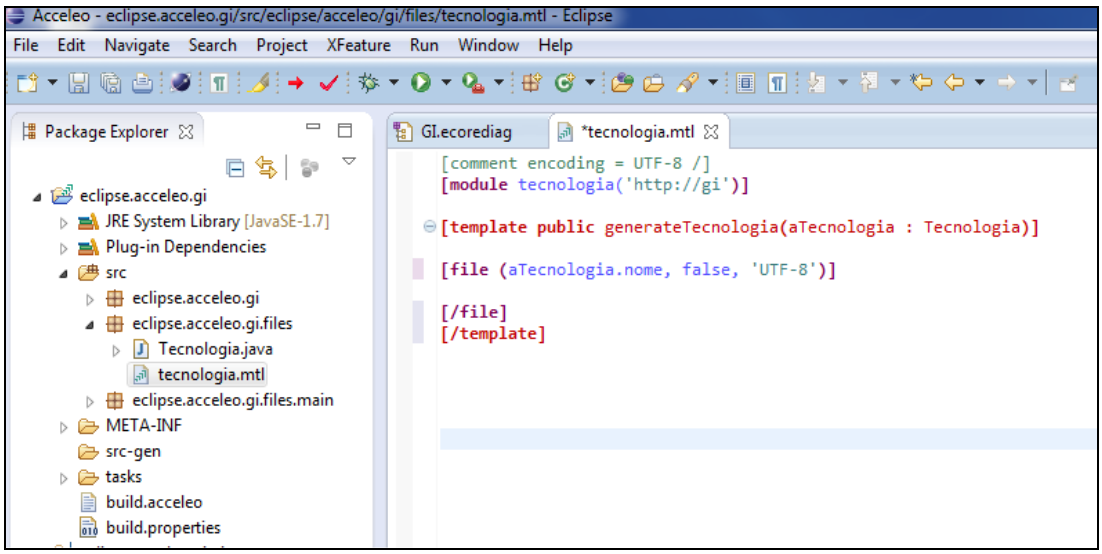

Figura B-9: Tela do arquivo de módulo do Acceleo gerado automaticamente

O editor do Acceleo fornece várias facilidades nas suas funções:

- Sintaxe em destaque;
- Assistente de conteúdo;
- Detecção de erro;
- Rápidas soluções (*ctrl+shift+1*);
- Contorno dinâmico;
- Rápido esboço (*ctrl* + *O*);
- Código dobrável;
- Declaração aberta (ctrl + clique esquerdo ou F3 na seleção);
- Pesquisa referências *(+shift ctrl* + *G*).

Para criar outros arquivos de módulo, deve-se clicar sobre o nome do pacote e selecionar *New→Acceleo Module File*, como ilustrado na Figura B-10.

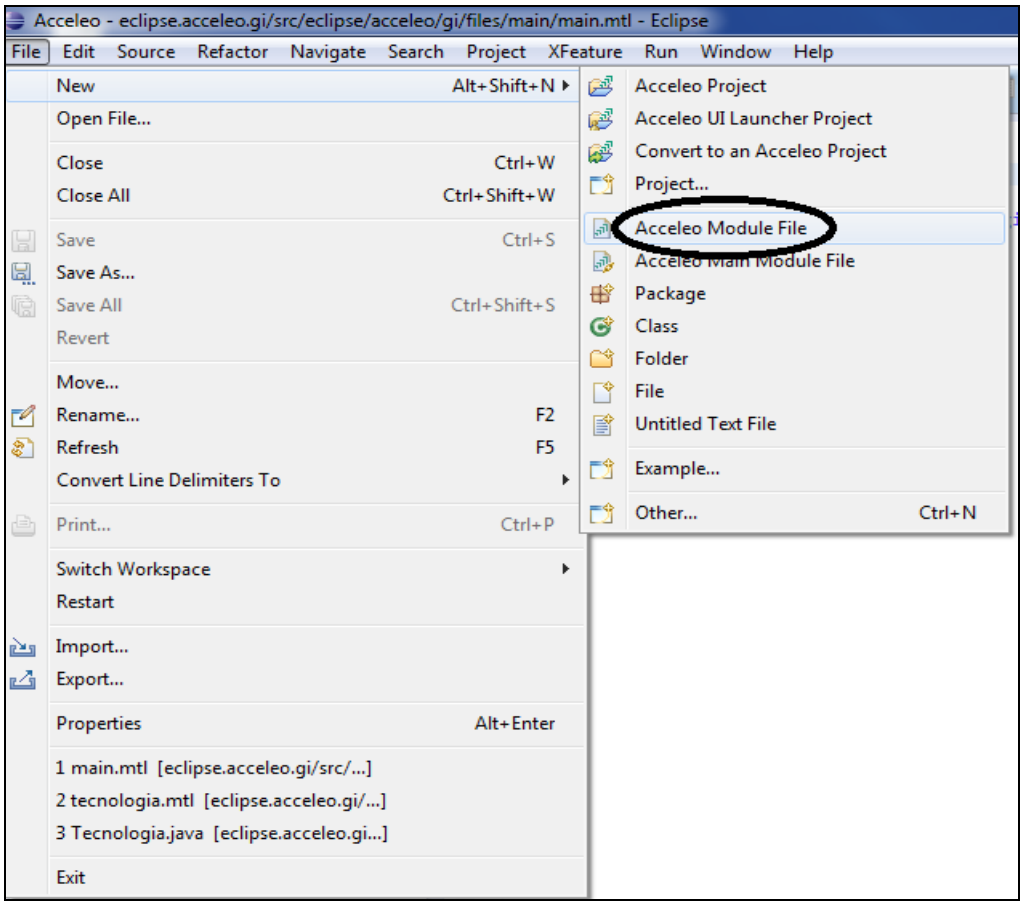

Figura B-10: Tela para criar novos arquivos de módulos do Acceleo

Quando o projeto Acceleo tem vários módulos, é recomendado que não tenha vários módulos com a anotação *@main*, pois isso dificulta o fluxo de criação de todos os arquivos. Dessa forma, deve ser criado um módulo principal que será responsável pela criação de todos os outros módulos. Geralmente, este módulo *main* é colocado no pacote *main*. Para criar o módulo *main,* deve-se clicar sobre o nome do pacote e selecionar *New→Acceleo Module File*. A janela *wizard* apresentada está ilustrada na Figura B-11.

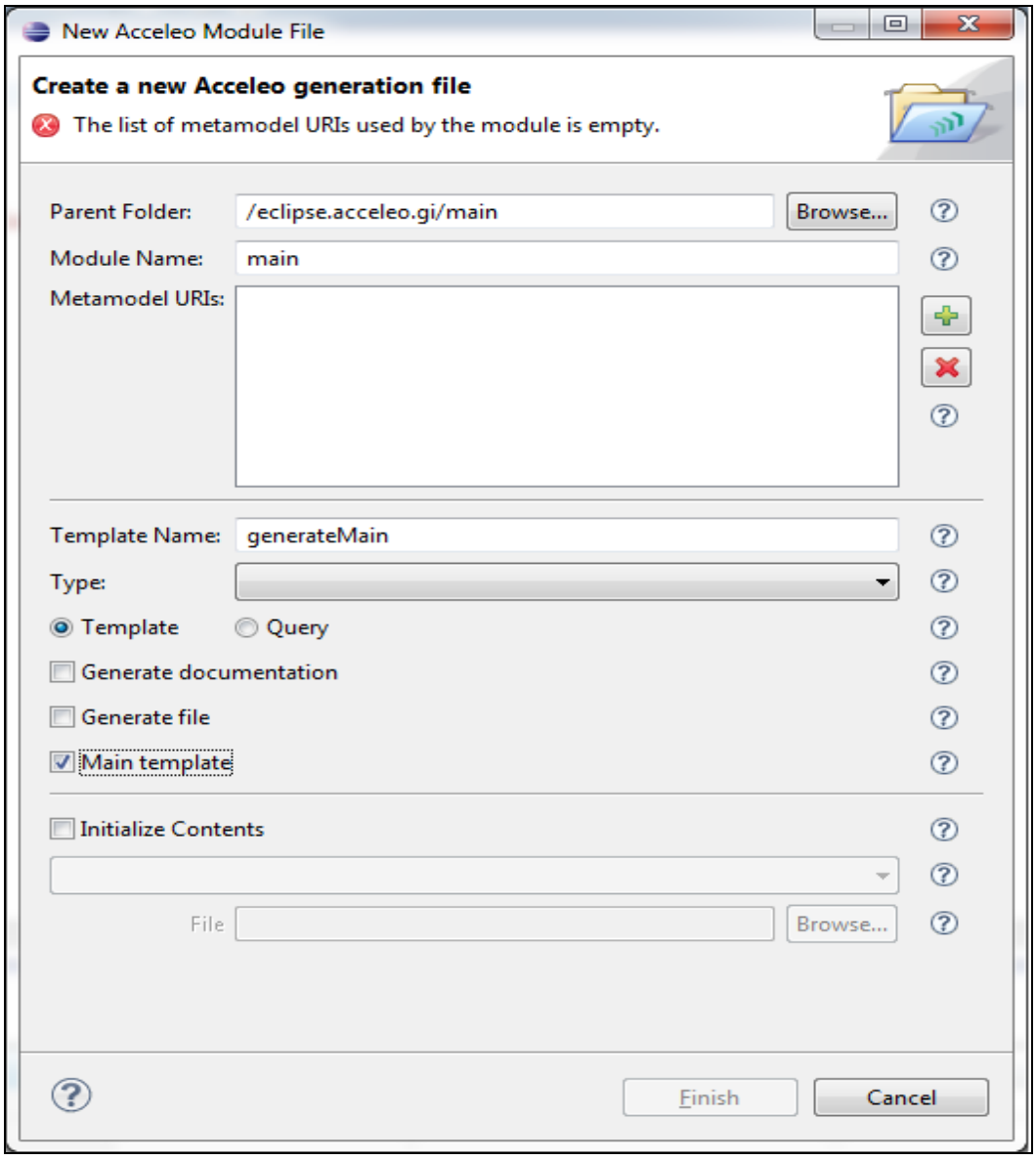

Figura B-11: Tela para criar o módulo *main* no Acceleo

Nesta janela deve ser colocado "*main*" no final do nome do pacote no campo *Parent Folder,* o nome do módulo deve ser *main*, o nome do *template* deve ser "*generateMain"*, a opção *Main template* deve ser selecionada. Clicar em *Finish*.

O pacote que contém o módulo com a anotação *@main* deve ser exportado para que outros *plug-ins* do Eclipse possam utilizá-lo. Para isto, o arquivo *META-INF/MANIFEST.MF* do projeto deve ser aberto. Na pasta *Runtime*, clicar em *Add* e selecionar o nome do pacote que contém o *main*, conforme ilustrado na Figura B-12.

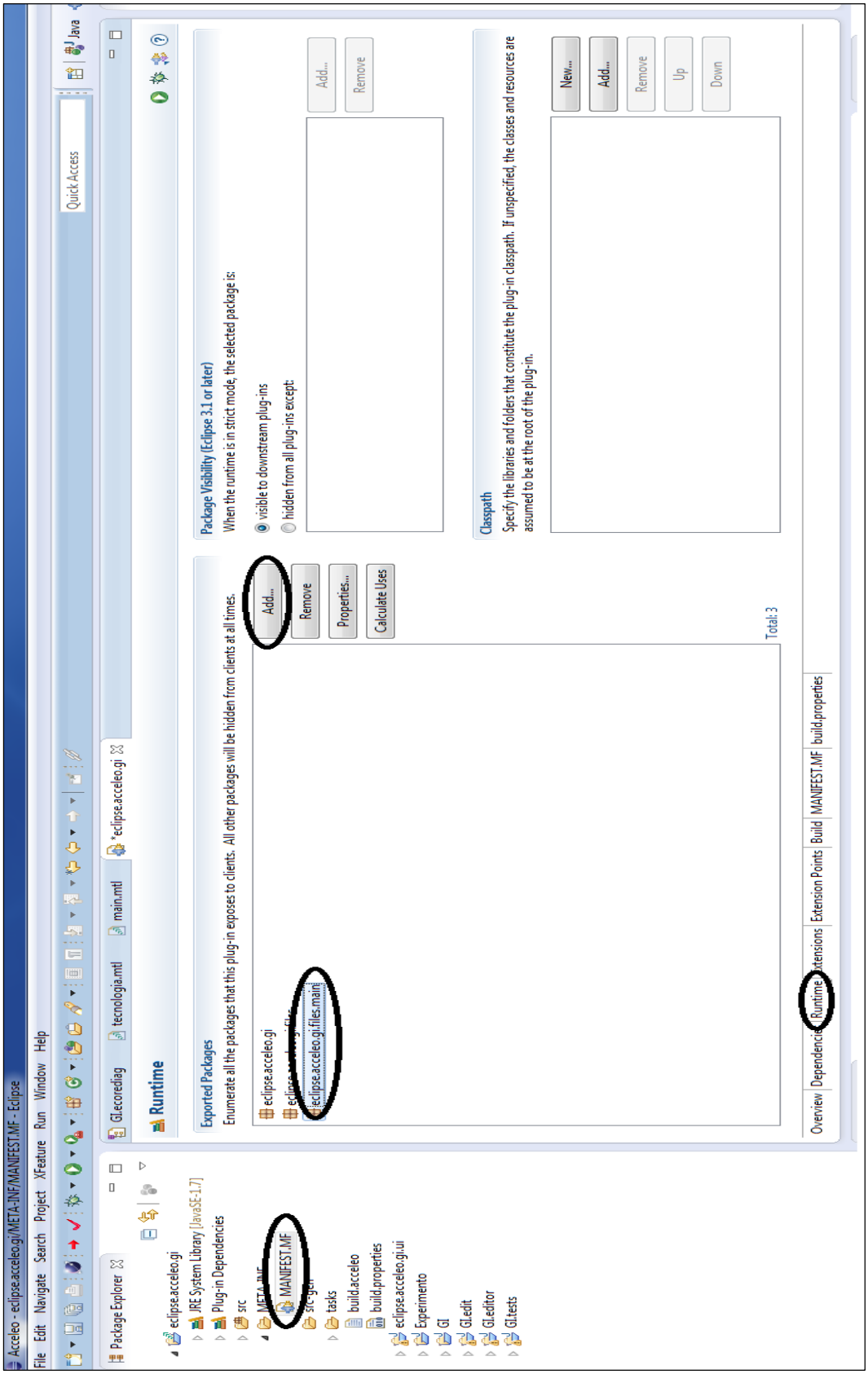

Figura B-12: Tela para exportar o módulo *main* por meio do arquivo MANIFEST

Plug-in para a geração do código

A ferramenta Acceleo permite a geração de uma interface gráfica para o usuário solicitar a geração de código a partir da especificação do modelo. Após a geração de todos os módulos finalizados, inclusive com a edição dos *templates*, pode ser criado um *wizard* para gerar o código do modelo a partir do Eclipse. Para criar esta facilidade foi utilizado o *wizard New Acceleo UI Project*. Este *wizard* cria um novo projeto Eclipse que permite solicitar a geração de código por meio de uma ação de clicar com o botão direito sobre o modelo que será transformado em código. Para criar este projeto, deve-se selecionar o projeto Acceleo e selecionar *New→Acceleo UI Launcher Project*, como ilustrado na Figura B-13.

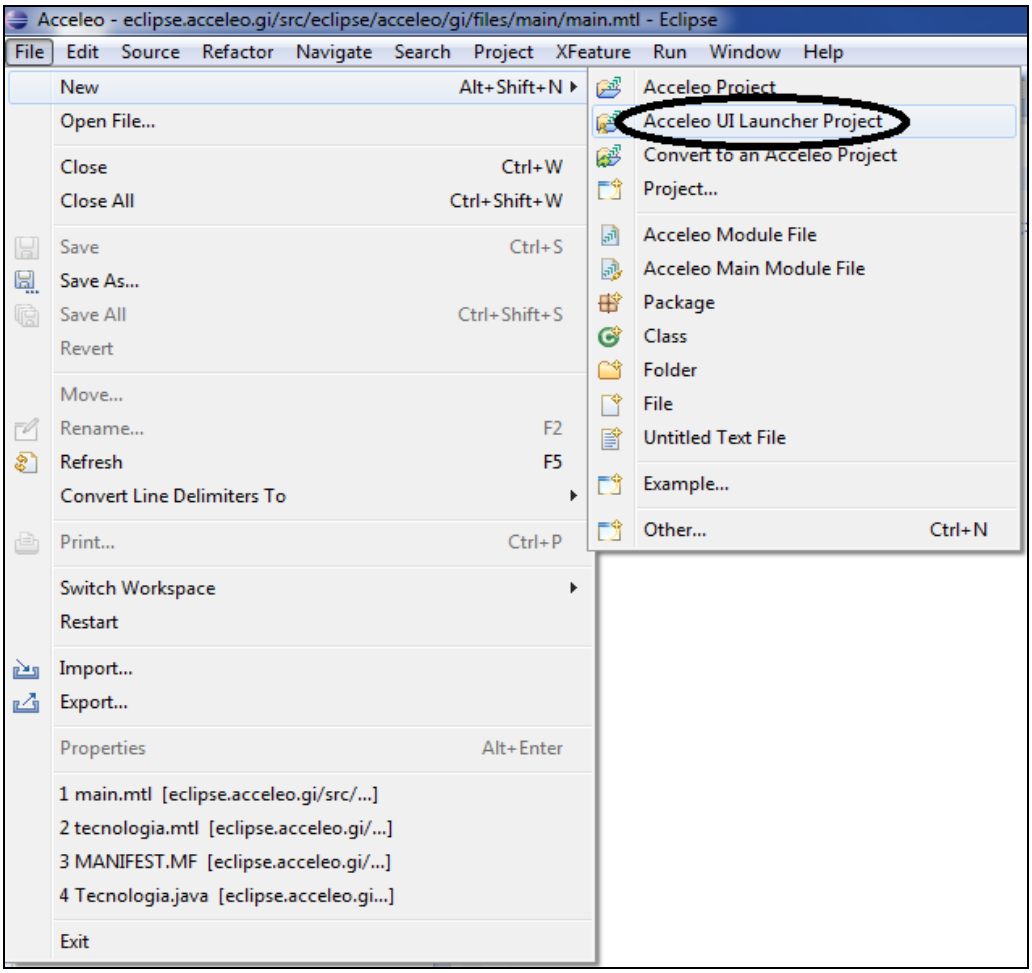

Figura B-13: Tela para criar um *Acceleo UI Launcher Project*

Uma janela *wizard* será apresentada, como ilustrada na Figura B-14. Deve-se preencher o nome do projeto e clicar em *Next*.

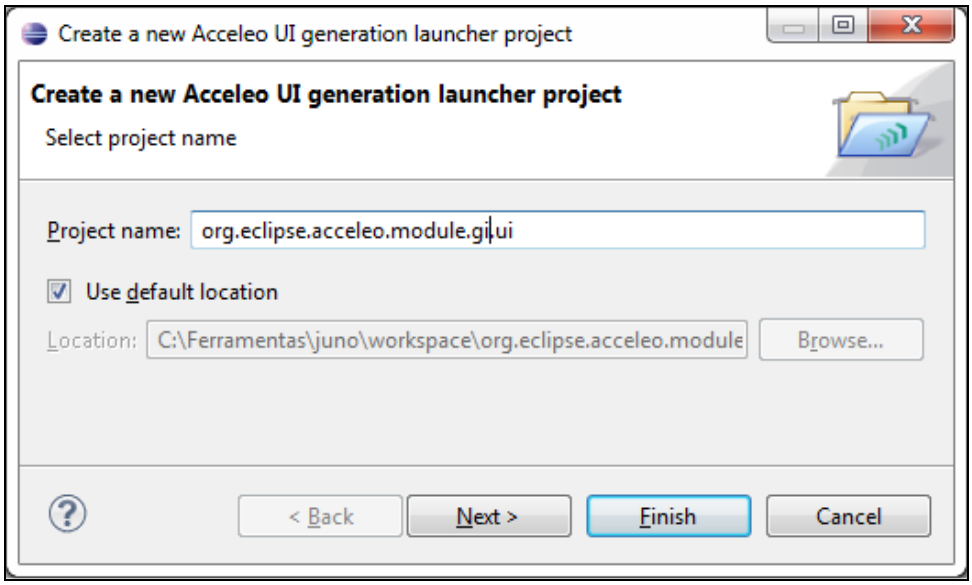

Figura B-14: Tela para nomear o *Acceleo UI Launcher Project*

Uma janela *wizard* será apresentada, como ilustrada na Figura B-15. Deve-se selecionar o projeto gerador e clicar em *Next*.

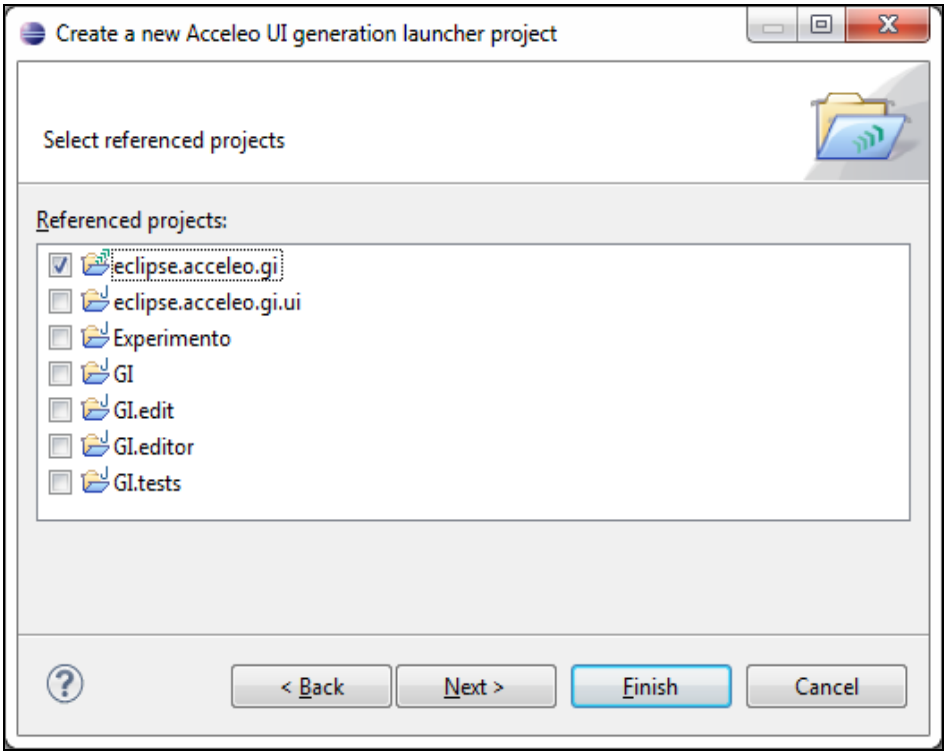

Figura B-15: Tela para selecionar o projeto gerador do *Acceleo UI Launcher Project*

Uma janela *wizard* será apresentada, como ilustrada na Figura B-16. Deve-se informar o filtro para os nomes de arquivos dos modelos e o diretório no qual os arquivos Java gerados serão colocados e clicar em *Finish*.

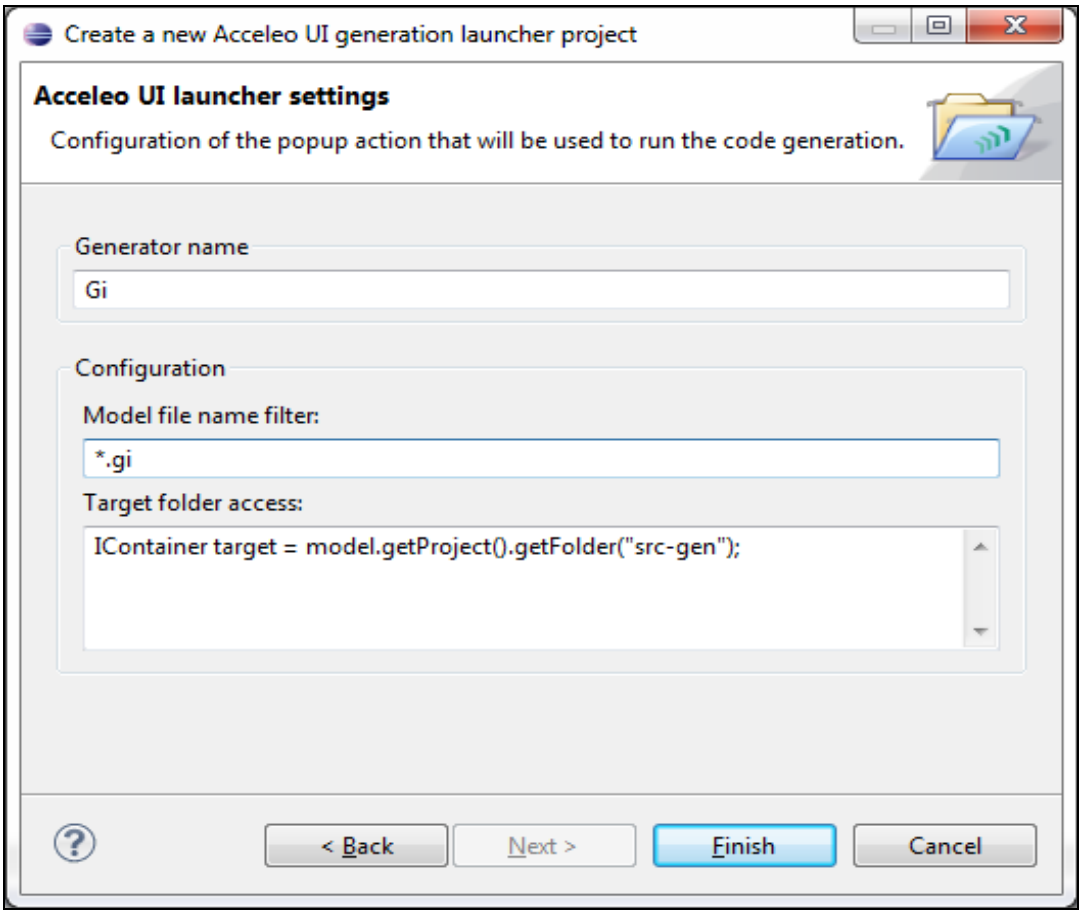

Figura B-16: Tela para configurar o *Acceleo UI Launcher Project*

O *wizard* criará um novo *plug-in* para o Eclipse com todo o código necessário para disponibilizar uma nova ação, que poderá ser selecionada pelo usuário, a partir do modelo especificado, para gerar o código do mesmo.

Para fazer a exportação desse *wizard* para ser reconhecido como um *plug-in* do Eclipse, deve-se selecionar ao mesmo tempo dois projetos no *Project Explorer* do Eclipse, o projeto do modelo e o projeto do UI, clicar com o botão direito do *mouse* e selecionar a opção *Export*.... , conforme ilustrado na Figura B-17.

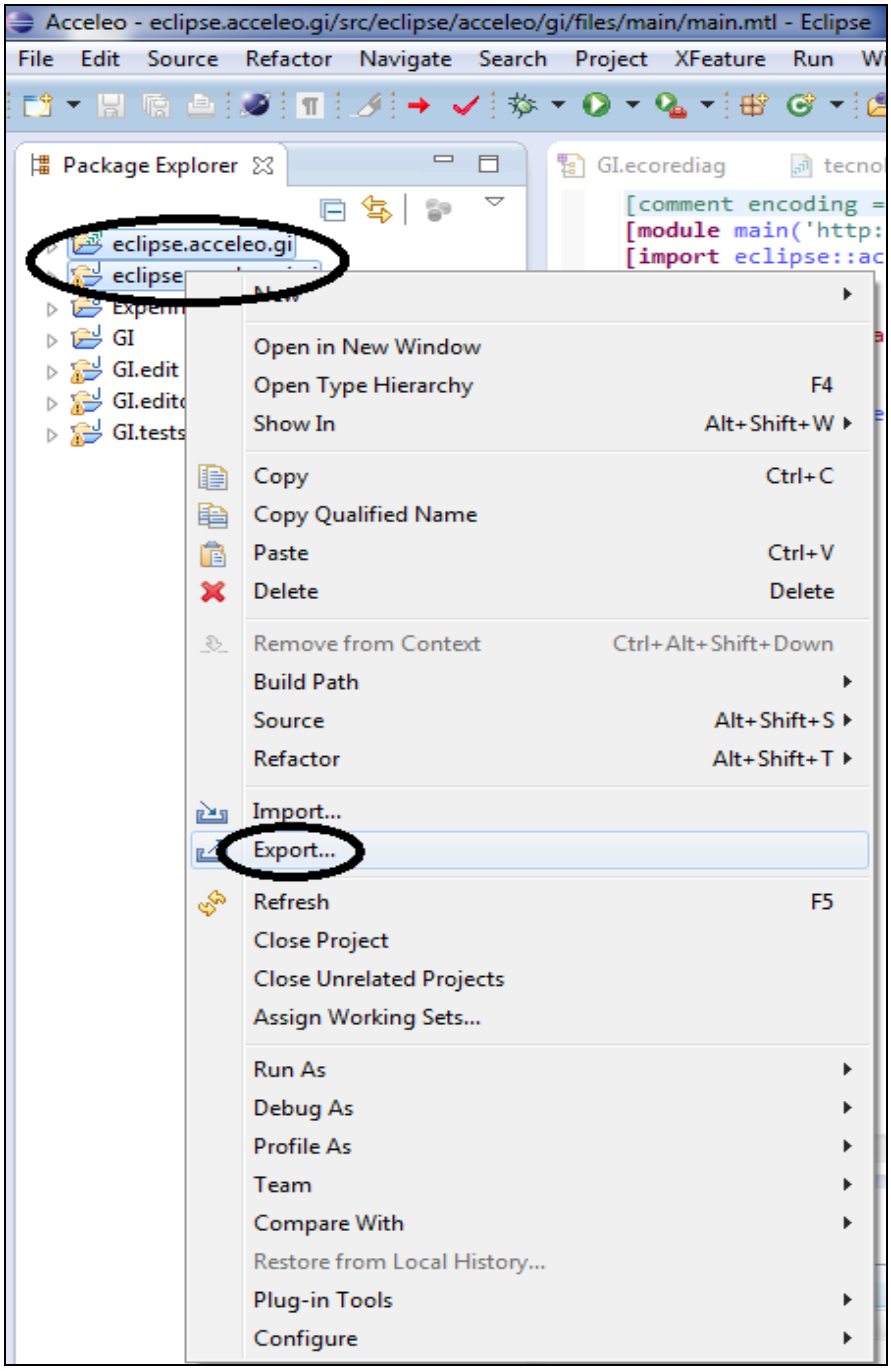

Figura B-17: Tela para exportar o *Acceleo UI Launcher Project*

Na janela do *Export*, deve-se selecionar a opção *Plug-in Development→Deployable plug-ins and fragments* e clicar em *Next*, como ilustrado na Figura B-18.

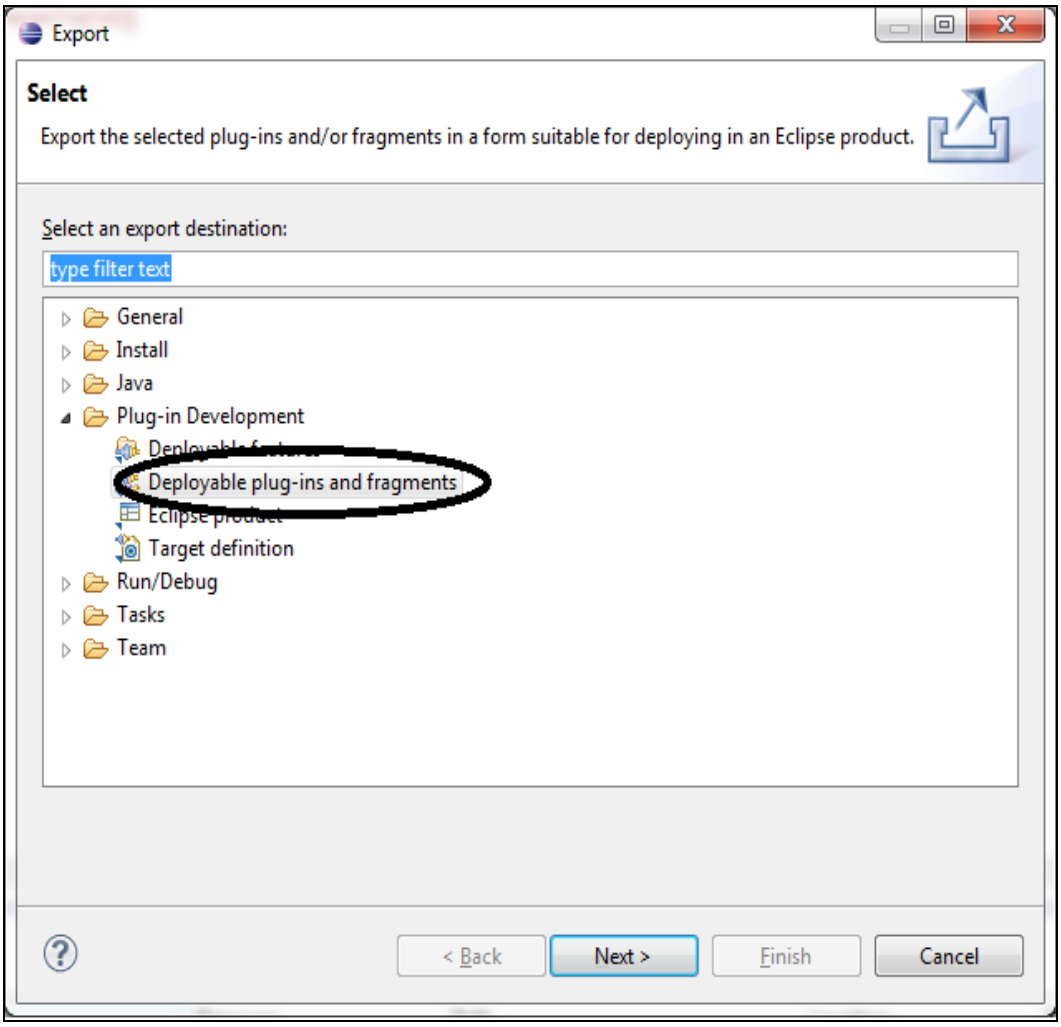

Figura B-18: Tela para selecionar *Deployable plug-ins and fragments*

Selecionar os dois módulos para o *plug-in* e clicar em *Finish*, como ilustrado na Figura B-19.

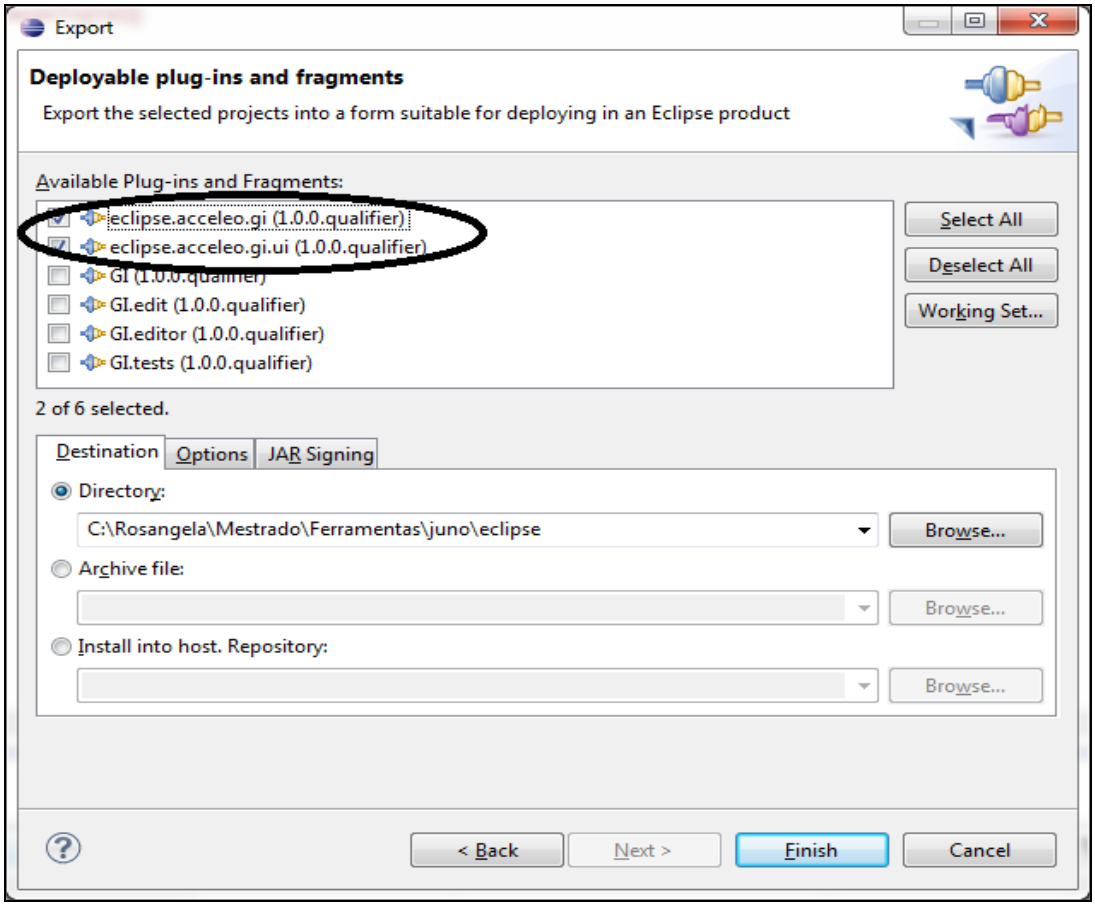

Figura B-19: Tela para selecionar os projetos para *Deployable plug-ins and fragments*

O Eclipse irá gerar os *plug-ins* no diretório de *plug-ins* do Eclipse (\eclipse\plugins).

Para o Eclipse reconhecer os *plug-ins* e carregá-los, deve-se reiniciar o Eclipse. Para isto deve-se clicar em *File→Restart,* conforme ilustrado Figura B-20.

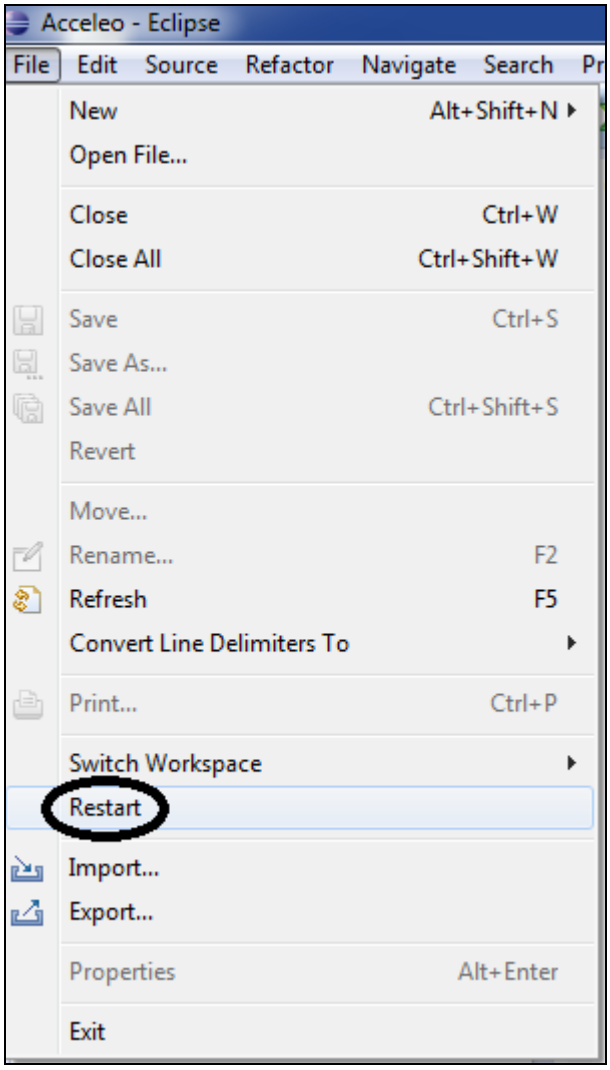

Figura B-20: Tela para solicitar *Restart* do Eclipse

Na Figura B-21 é ilustrado o resultado desse *wizard*.

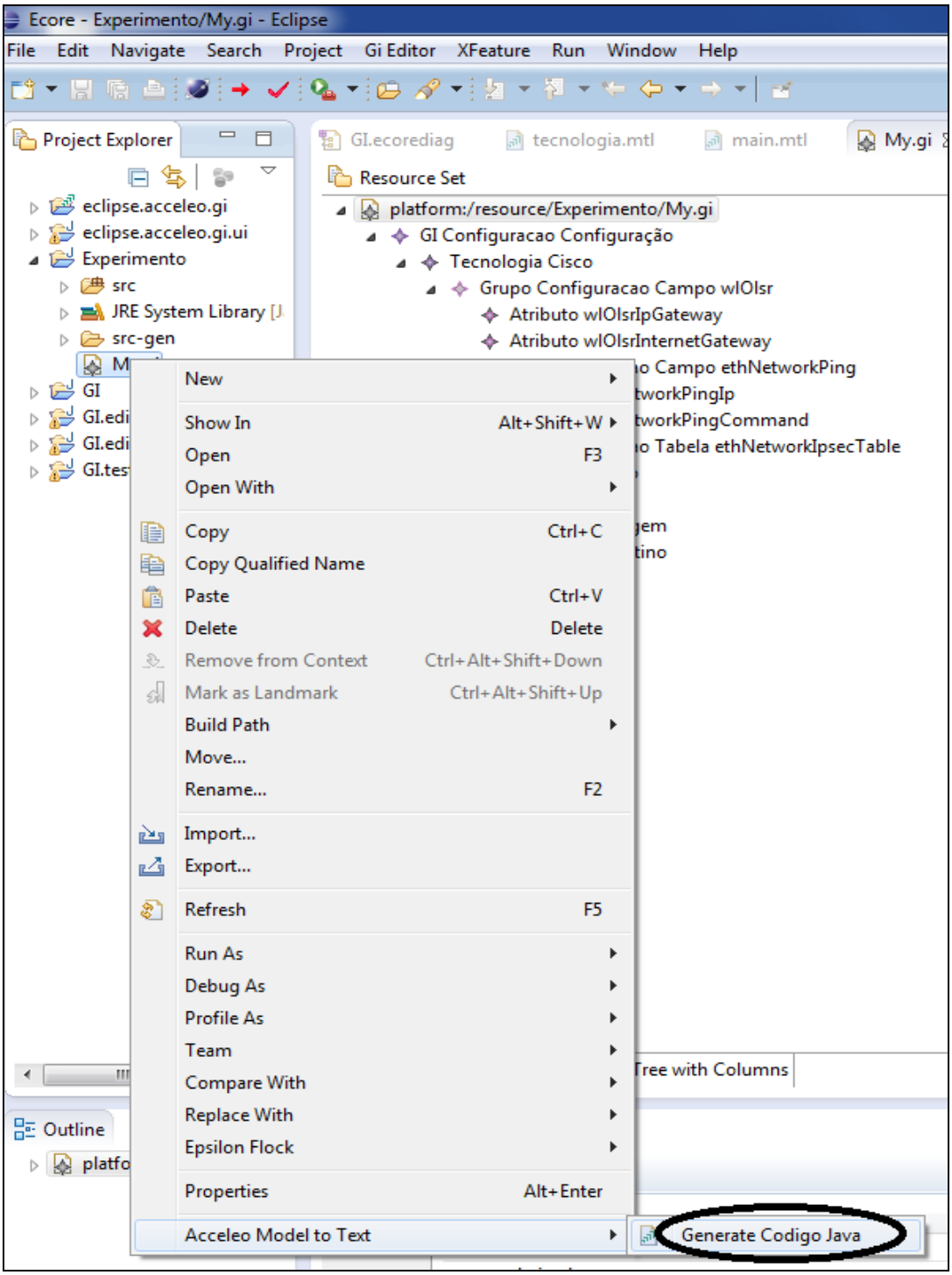

Figura B-21: Tela com o resultado do *wizard*

Ao solicitar a geração do código Java por meio do botão da ação *Acceleo Model to Text → Generate Codigo Java*, os arquivos, que foram especificados nos *templates*, serão gerados no diretório especificado, com as informações definidas no modelo, como ilustrado na Figura B-22.

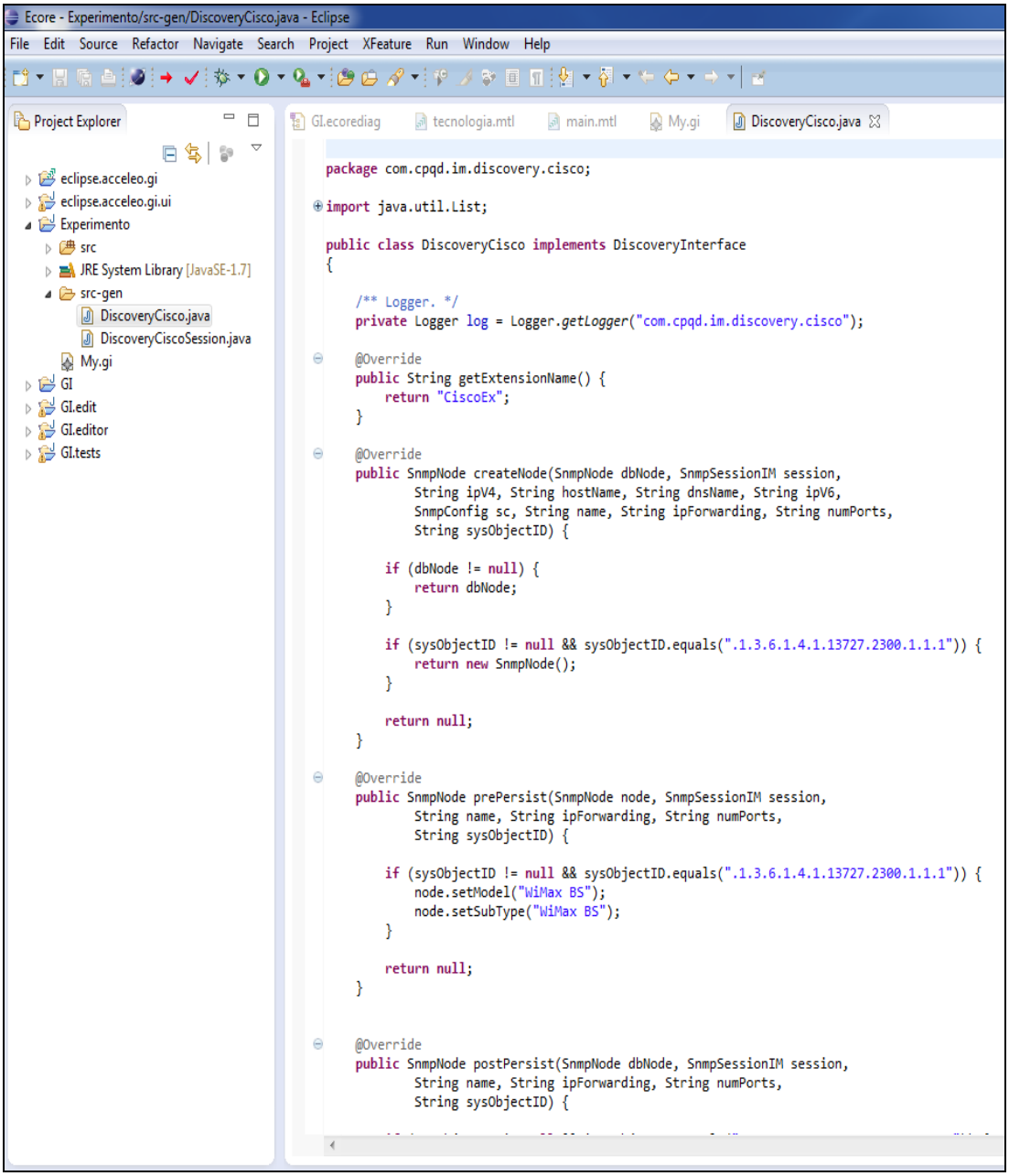

Figura B-22: Tela com o arquivo gerado por meio do *wizard*

Utilização da DSL desenvolvida

Para utilizar a DSL desenvolvida para especificar uma nova tecnologia de rede, deve-se criar um novo projeto no Eclipse que servirá de diretório pai para um novo projeto de modelo, o qual será especificado conforme o metamodelo que contém as regras de negócio do domínio para especificar uma nova tecnologia de rede. Quando é criado um novo projeto de modelo, somente os objetos que pertencem ao seu metamodelo são exibidos em uma *Combo Box* para que seja selecionado o objeto raiz (*container*) especificado no metamodelo. É a partir desse objeto raiz que todos os outros objetos do metamodelo são criados. No caso do metamodelo da especificação de uma nova tecnologia de rede, o objeto raiz é o GIConfiguração. Após esse passo, o novo projeto do modelo é criado e exibido no *Project Explorer* do Eclipse, e uma árvore hierárquica do modelo iniciando com o objeto raiz é exibida. Nessa árvore, é possível configurar o objeto raiz e adicionar seus filhos conforme as regras de negócio para especificar uma nova tecnologia de rede. À medida que se seleciona o objeto folha e clica com o botão direito do *mouse* sobre ele, um menu de contexto chamado *New Child* é exibido para que o objeto filho seja especificado. E assim por diante até que seja finalizada a especificação da nova tecnologia de rede. As propriedades de cada objeto do metamodelo devem ser preenchidas com a especificação da nova tecnologia de rede. Após a finalização da especificação da nova tecnologia de rede, pode ser solicitada a geração do código-fonte correspondente ao modelo especificado. O código é gerado e colocado no diretório que foi definido na construção da DSL, nesse caso src-gen, que se encontra sob o diretório pai do projeto de modelo. Esse passo a passo assim como as ilustrações das janelas do Eclipse são apresentadas a seguir.

Para criar um novo projeto no Eclipse, deve-se selecionar (File→New→Project...), como ilustrado na Figura B-23. Uma janela chamada *New Project* é exibida, deve-se selecionar *General→Project* e clicar em *Next* como ilustrado na Figura B-24. Na próxima janela exibida, ilustrada na Figura B-25, devem ser fornecidos um nome para o projeto e o diretório de sua localização e clicar em

Finish. No exemplo, o projeto se chamará “Experimentos” e ficará no diretório c:\Ferramentas\juno\workspace\Experimentos.

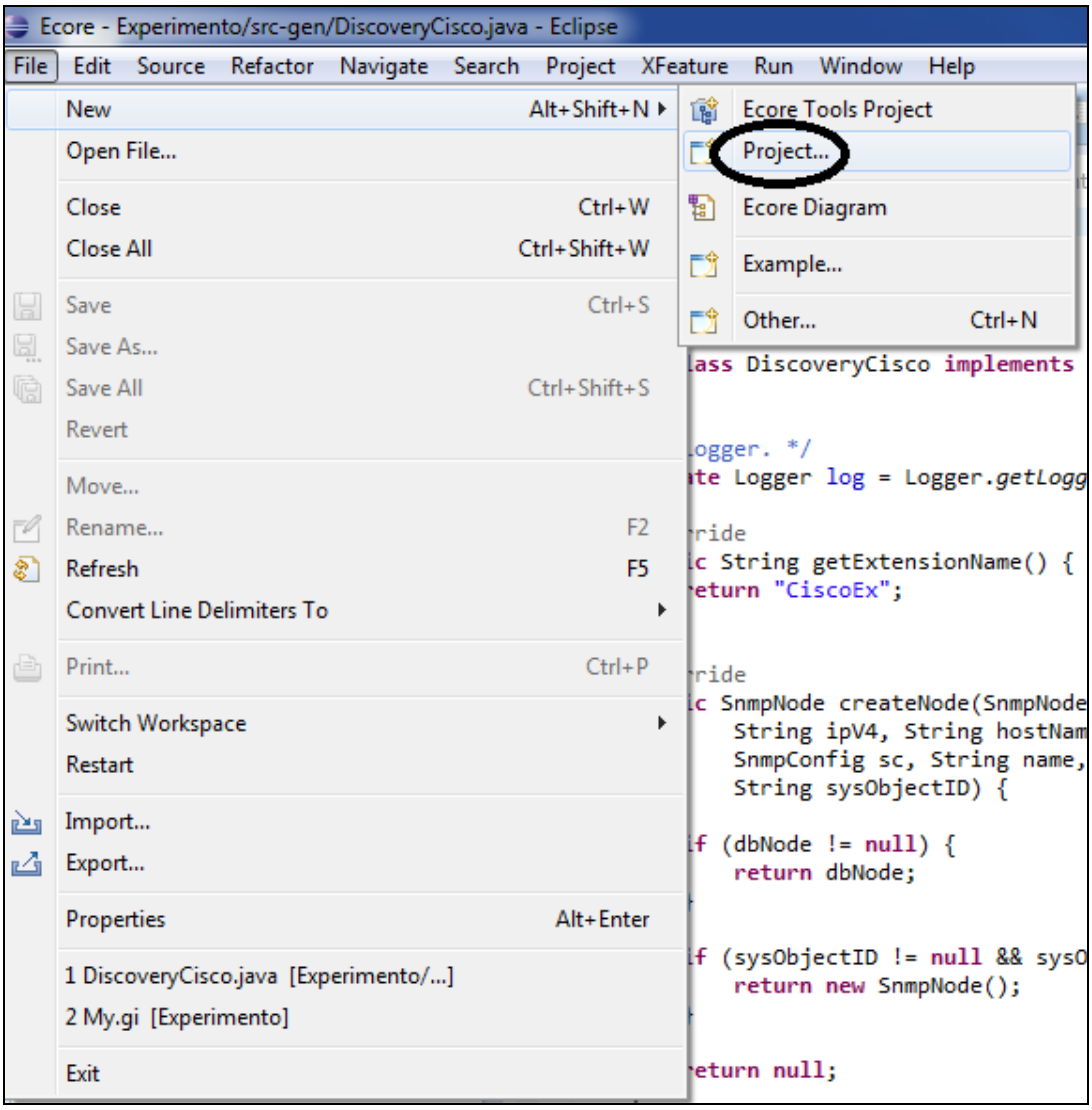

Figura B-23: Tela para criar um novo projeto Eclipse

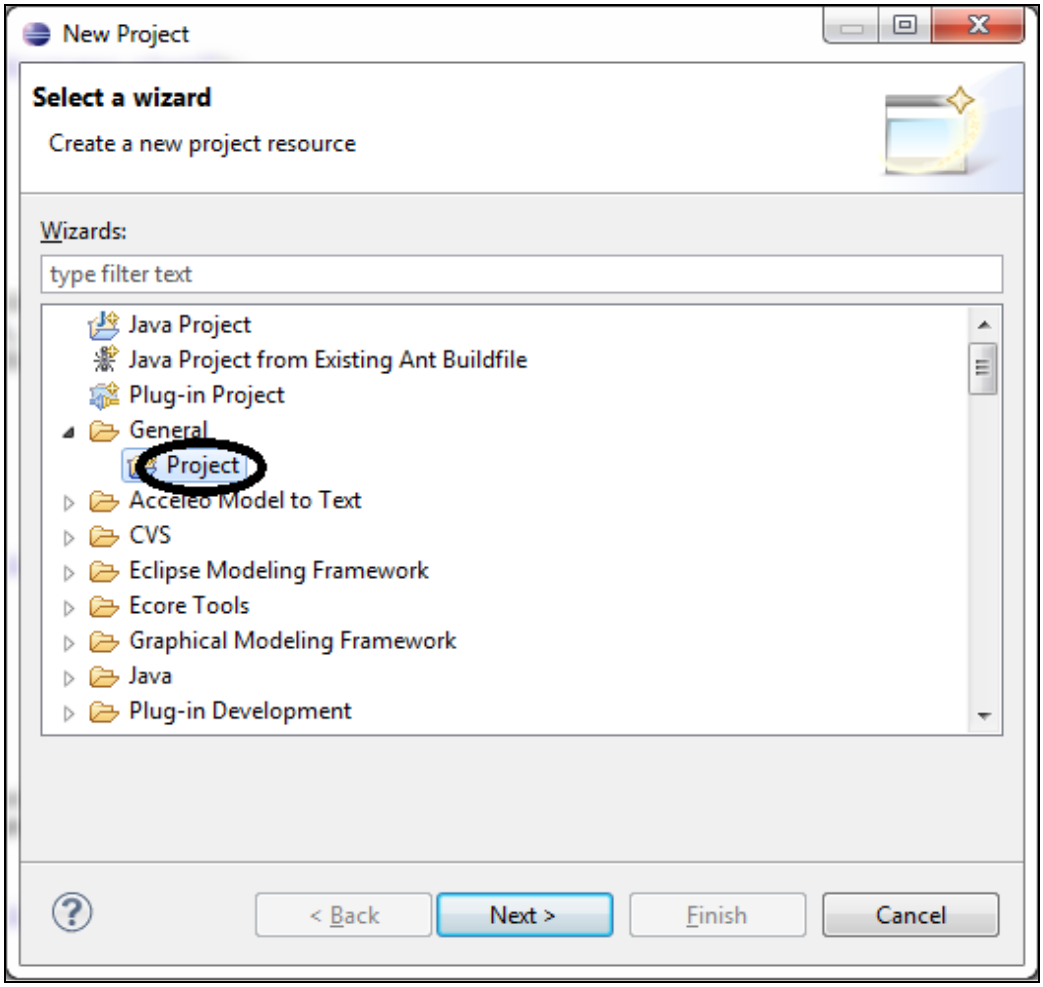

Figura B-24: Tela para selecionar *General Project*

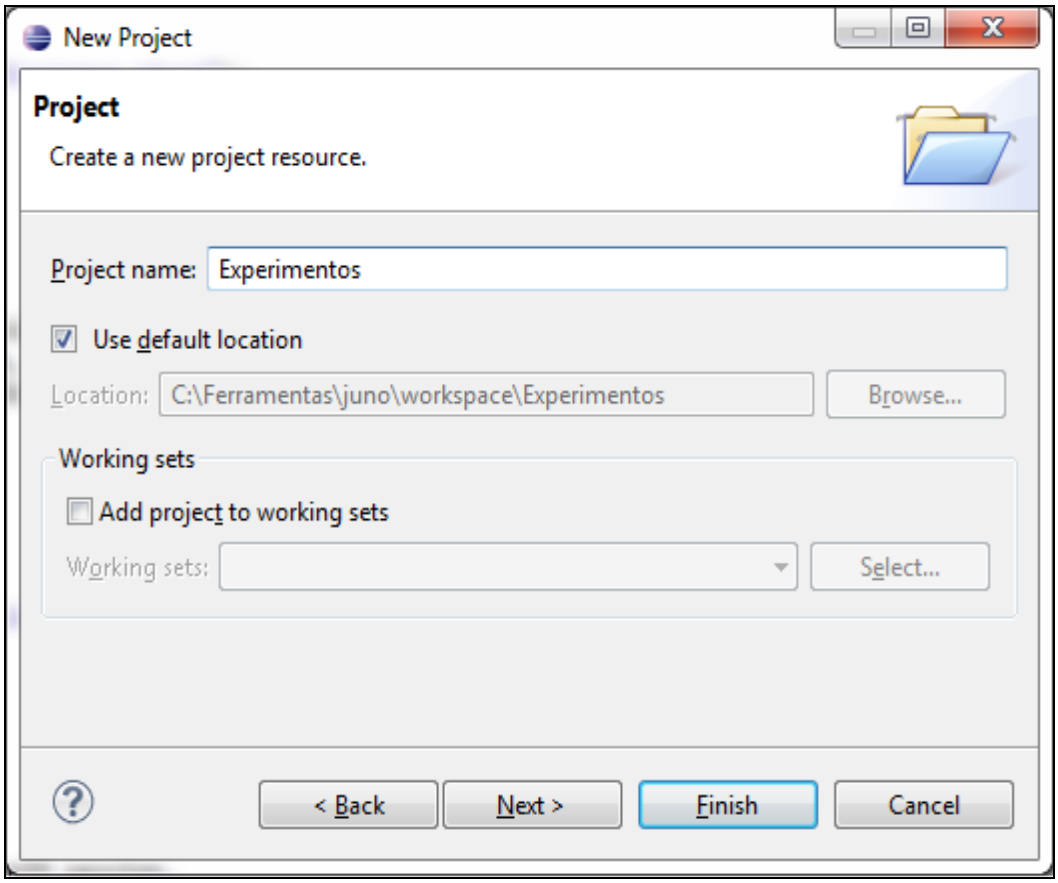

Figura B-25: Tela para nomear o *General Project*

O próximo passo é a criação de um novo projeto para um modelo de gerência de configuração. Para isto, deve-se clicar em *File→New→Other*..., como ilustrado na Figura B-26.

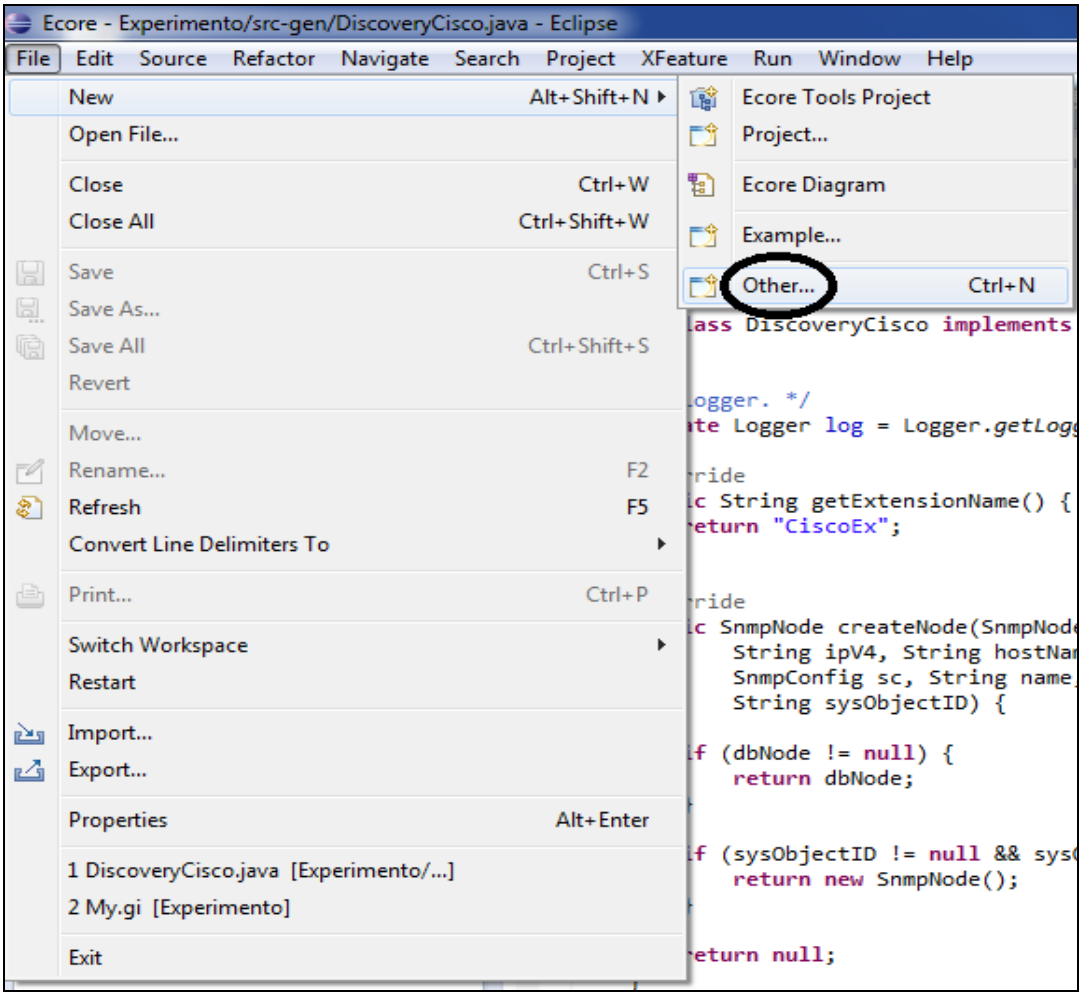

Figura B-26: Tela para criar um outro projeto

Em seguida, na janela *wizard* apresentada, deve-se selecionar a opção *Exemple EMF Model Creation Wizards* → *Gi Model* e clicar em *Next*, como ilustrado na Figura B-27, para a criação de um novo projeto de modelo GI.

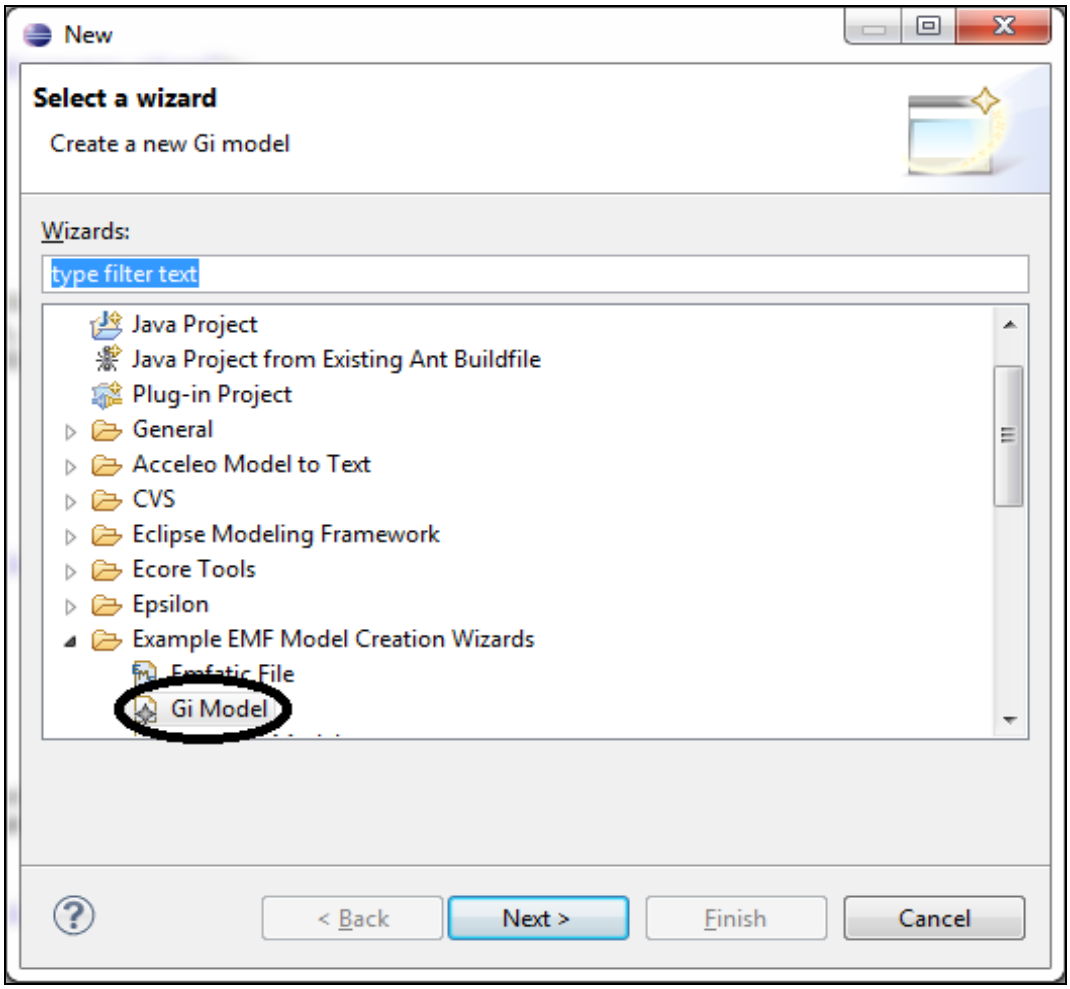

Figura B-27: Tela para criar um projeto de modelo Gi

A próxima janela *wizard* apresentada é a ilustrada na Figura B-28, na qual devem ser fornecidos um nome para o diretório pai para o projeto de modelo de gerência de configuração e um nome para o projeto de modelo. Em seguida, deve=se clicar em *Next*.

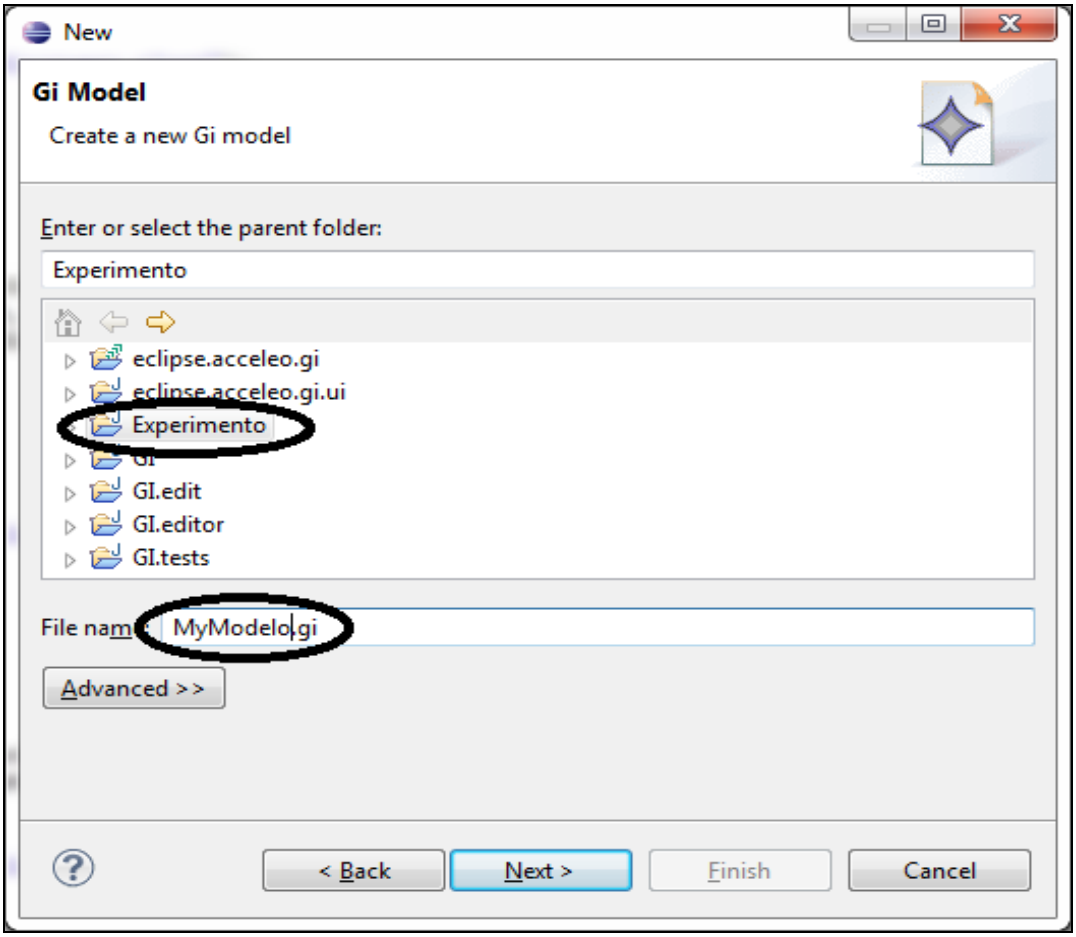

Figura B-28: Tela para nomear o projeto de modelo Gi

Na próxima janela *wizard* apresentada, selecionar na C*ombo Box* do *Model Object* a opção “GI Configuração”, pois é a partir deste objeto que as regras de negócio do domínio da gerência de configuração se iniciam. Em seguida, deve-se clicar em *Finish,* como ilustrado na Figura B-29, e será exibida uma tela com o novo projeto para o modelo de gerência de configuração criado, como ilustrado na Figura B-30.

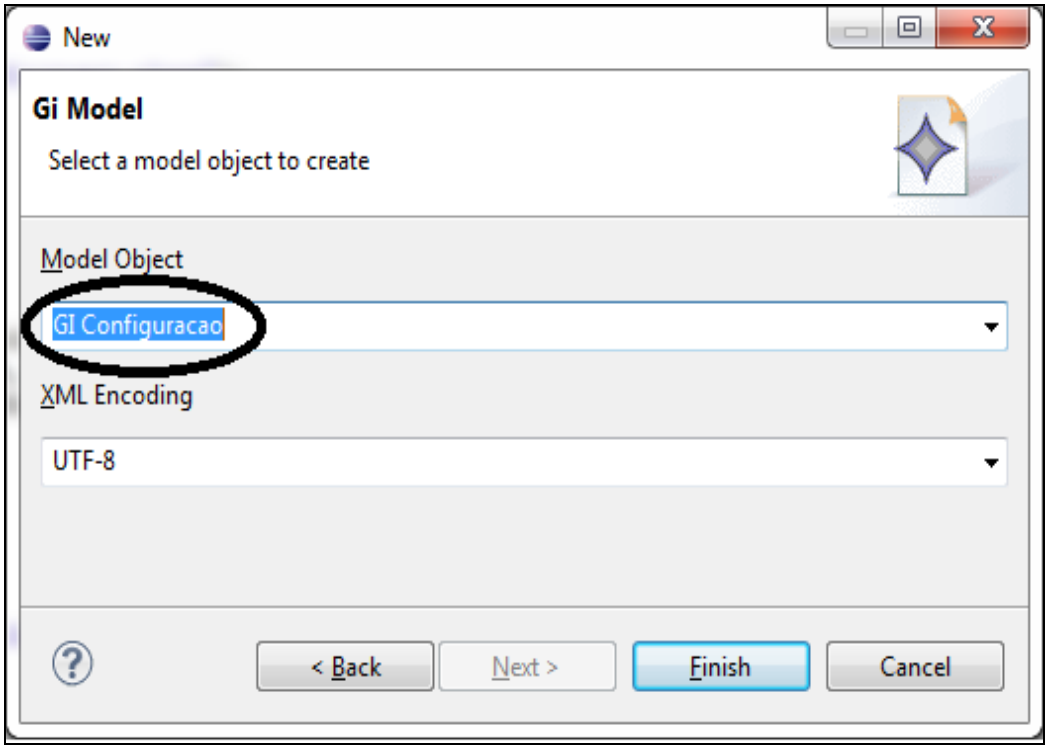

Figura B-29: Tela para selecionar um objeto do modelo Gi

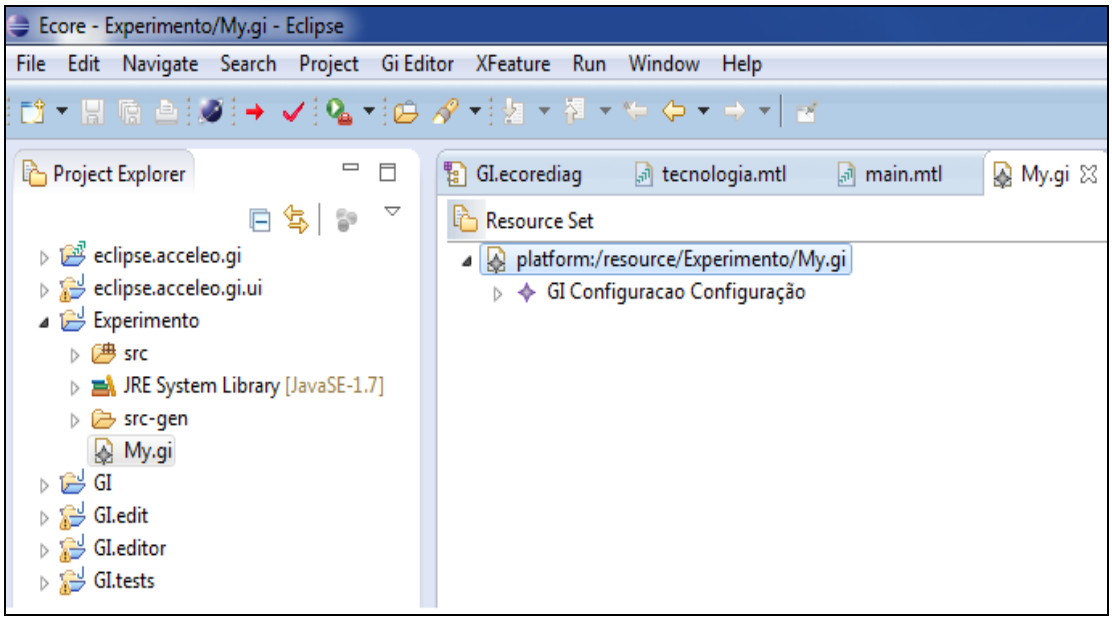

Figura B-30: Tela com o novo projeto do modelo Gi criado

No novo projeto de modelo de gerência de configuração, deve ser especificada uma nova tecnologia de rede que passará a ser gerenciada pelo sistema de gerenciamento integrado de redes conforme as regras de negócio definidas no metamodelo.

A primeira especificação deve ser o nome da configuração, ou seja, deve ser preenchido o nome do GI Configuração. Para isto, deve-se clicar no GI Configuração e a visão de propriedades (*Properties*) se abre e exibe duas colunas: uma chamada *Property* e a outra *Value*. Na coluna *Property* aparece a propriedade Nome e na coluna *Value* deve ser informado o nome para o GI Configuração, como ilustrado na Figura B-31.

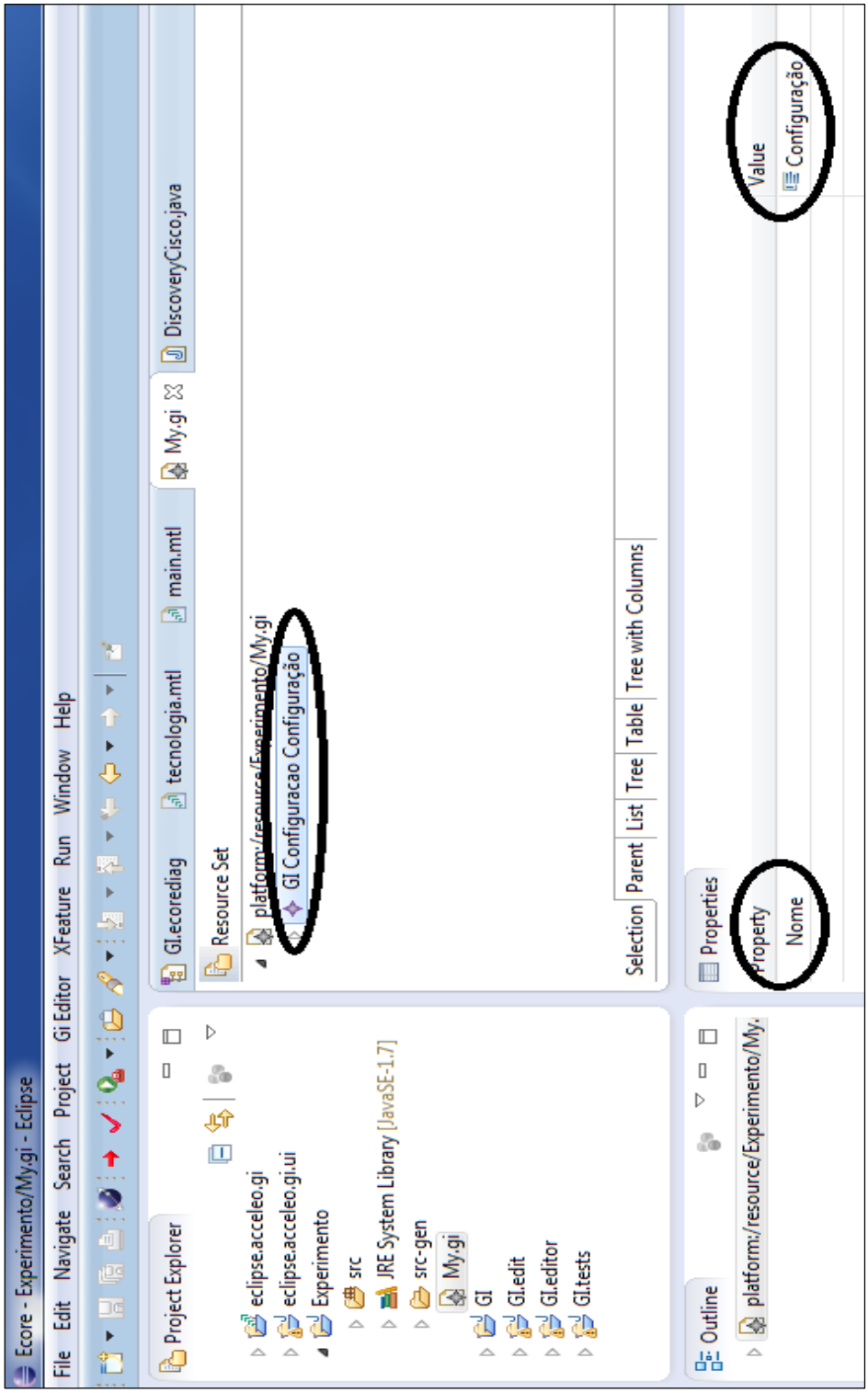

Figura B-31: Tela para especificar o objeto Gi Configuração

Depois de especificado o GI Configuração, deve-se especificar a nova tecnologia que passará a ser gerenciada pelo sistema de gerenciamento integrado de redes. Para isto, deve-se clicar com o botão direito do *mouse* sobre o GI Configuração e selecionar *New Child*→Tecnologia, como ilustrado na Figura B-32.

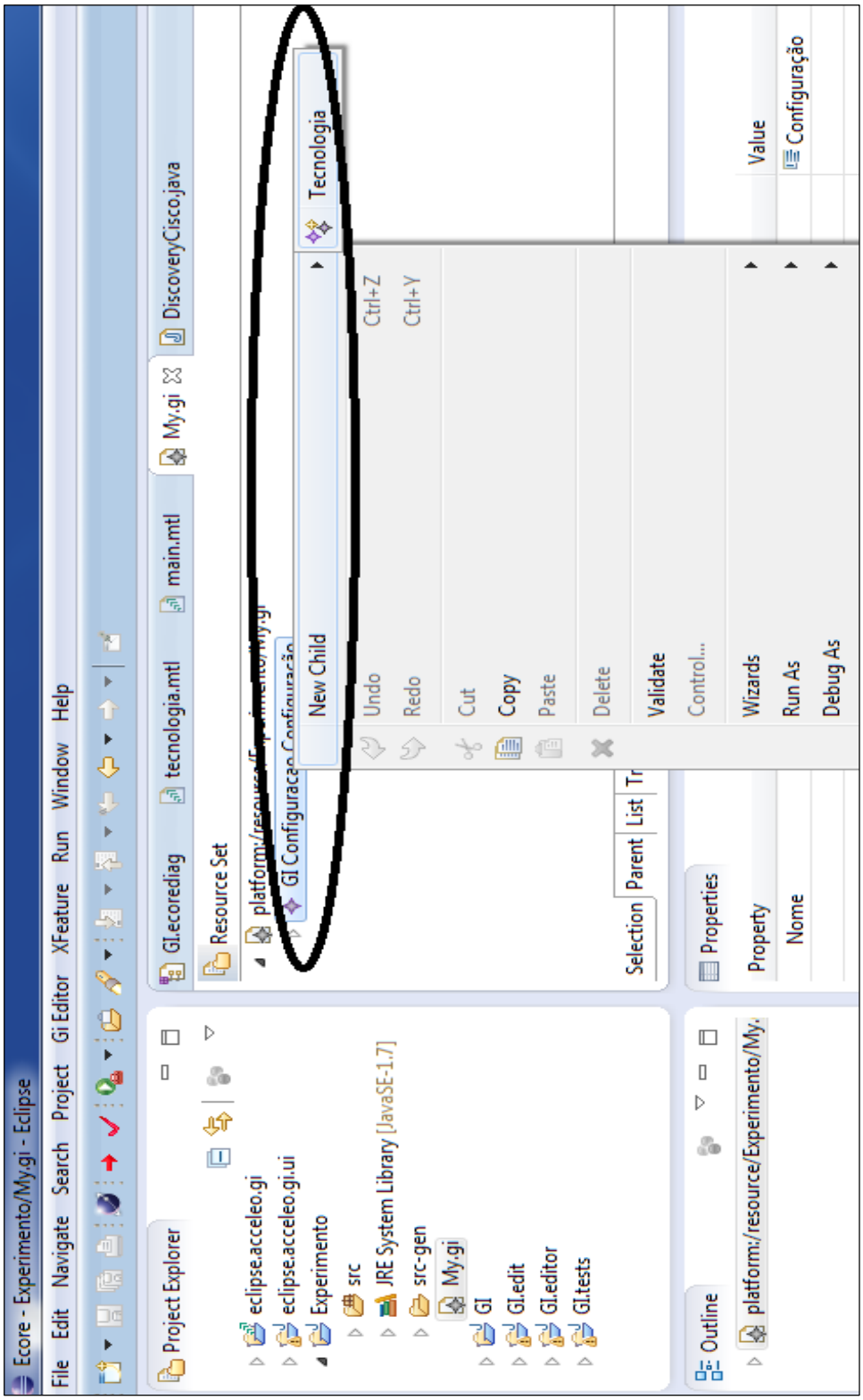

Figura B-32: Tela para selecionar o objeto tecnologia

A tecnologia aparece hierarquicamente abaixo de GI Configuração. A visão de propriedades (*Properties*) exibe duas colunas: uma chamada *Property* e a outra

Value. Na coluna *Property* aparecem as propriedades: Identificação, Nome, Modelo e Tipo. Na coluna *Value* deve ser informado os valores para Identificação, Nome, Modelo e Tipo da Tecnologia, como ilustrado na Figura B-33. Essas informações dependem do especialista da rede, pois são informações intrínsecas da tecnologia.

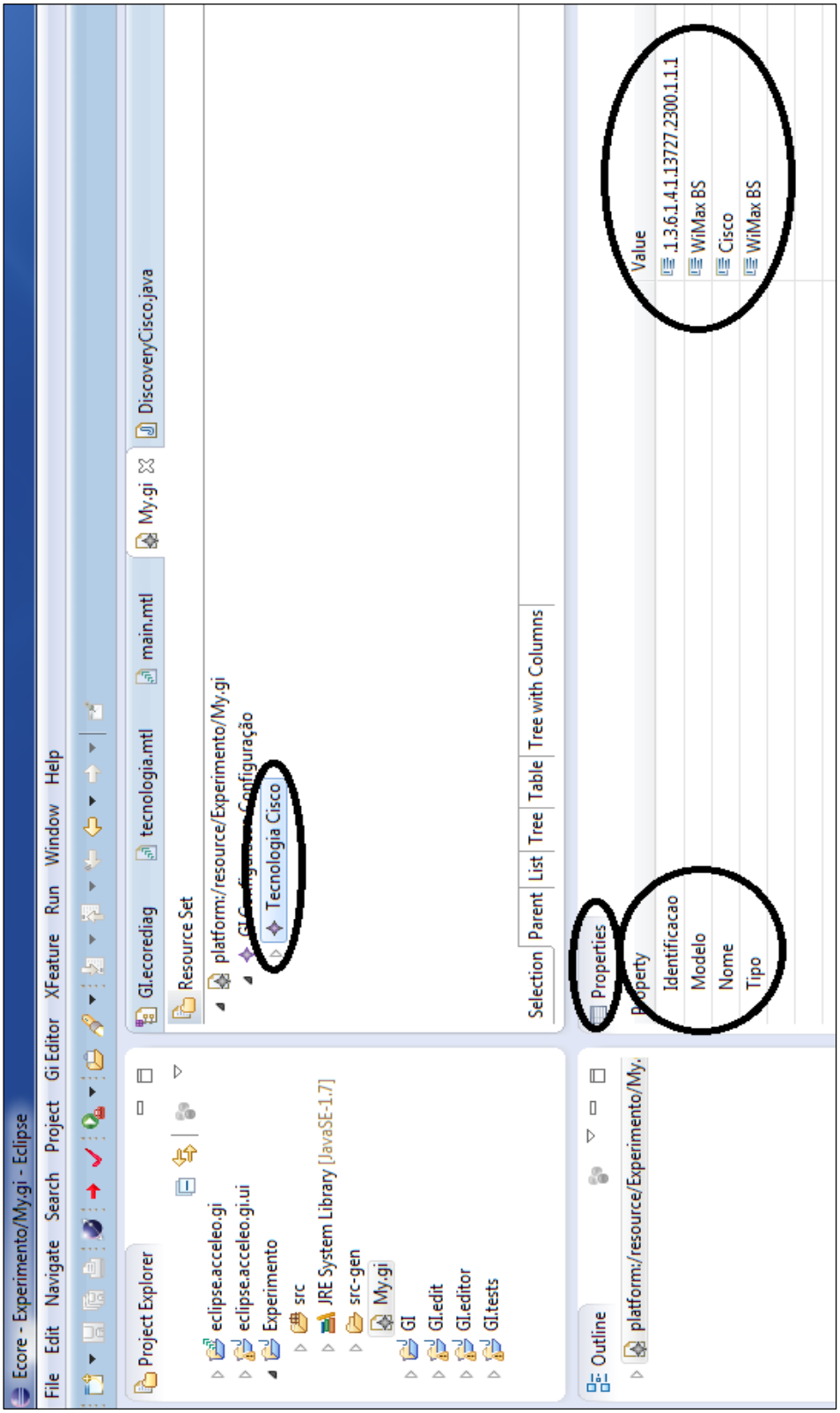

Figura B-33: Tela para especificar a tecnologia

Depois de especificada a Tecnologia, deve-se especificar os grupos de configuração. Existem dois grupos de configuração um que especifica campos e

outro que especifica tabela. A forma de realizar a especificação de ambos os grupos é semelhante. Para isto, deve-se clicar com o botão direito do mouse sobre a Tecnologia e selecionar *New Child*→GrupoConfiguraçãoCampo ou selecionar *New Child*→GrupoConfiguraçãoTabela, como ilustrado na Figura B-34.

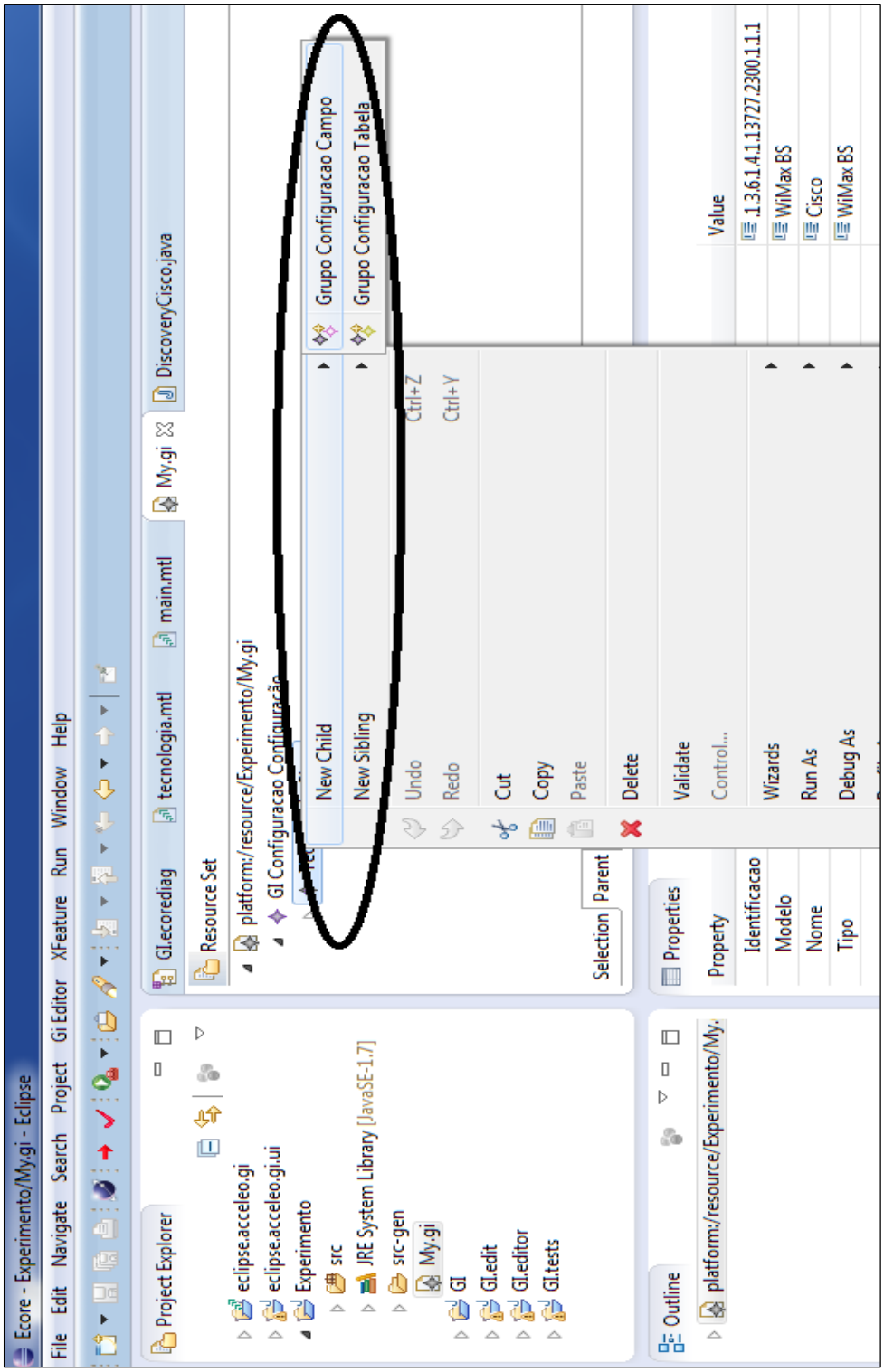

Figura B-34: Tela para selecionar o objeto Grupo de Configuração

O grupo de configuração selecionado aparece hierarquicamente abaixo de Tecnologia. A visão de propriedades (*Properties*) exibe duas colunas: uma chamada *Property* e a outra *Value*. Na coluna *Property* aparece a propriedade Nome. Na coluna *Value* deve ser informado o nome do grupo de configuração, como ilustrado

na Figura B-35. Essa informação depende do especialista da rede, pois é uma informação intrínseca da tecnologia. O nome do grupo de configuração deve ser exatamente igual ao que está definido na MIB dos elementos desta tecnologia.

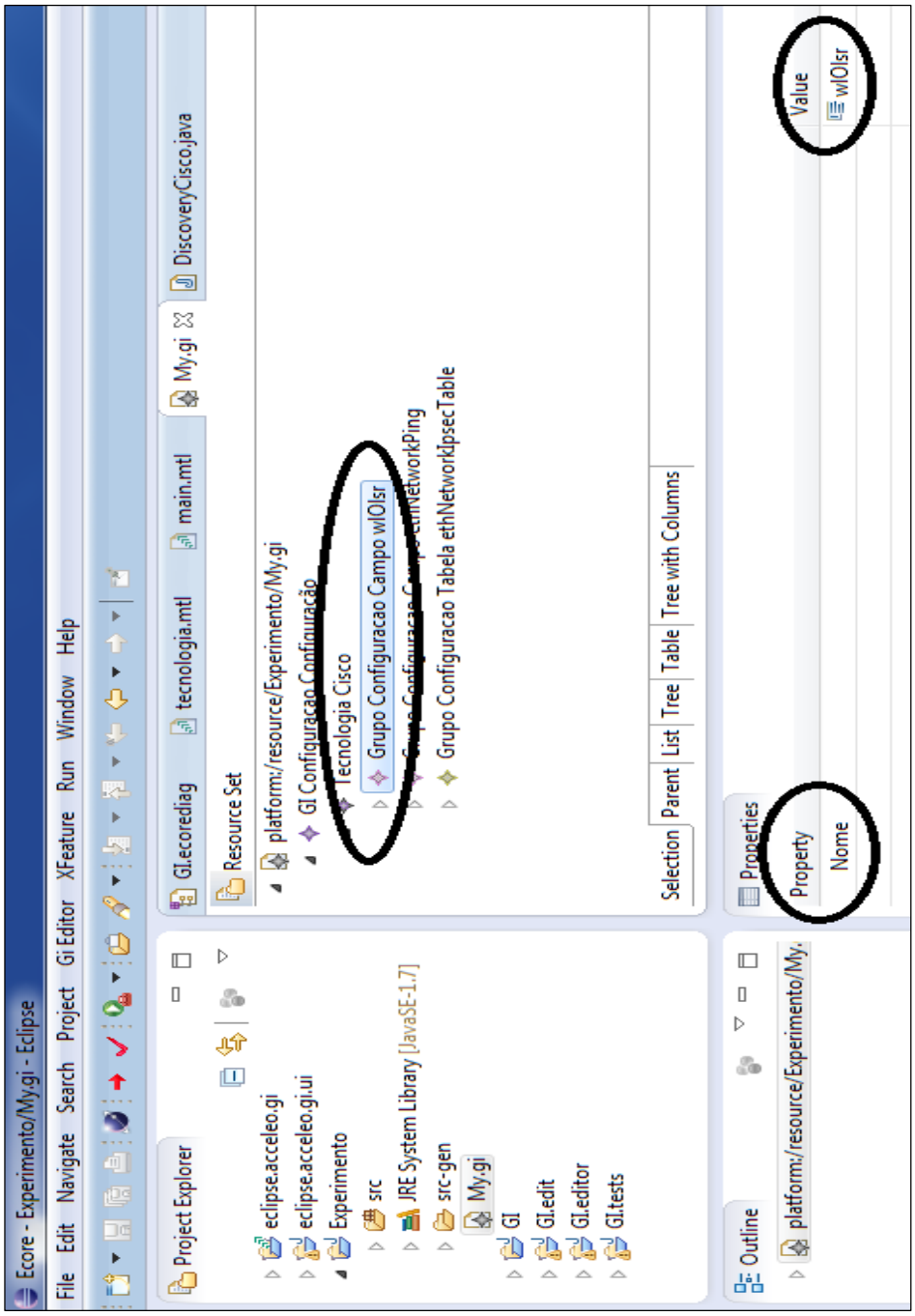

Figura B-35: Tela para especificar o Grupo de Configuração

Depois de especificado o nome do grupo de configuração, deve-se especificar os atributos deste grupo. Para isto, deve-se clicar com o botão direito do mouse

sobre o grupo de configuração e selecionar *New Child*→Atributo, como ilustrado na Figura B-36.

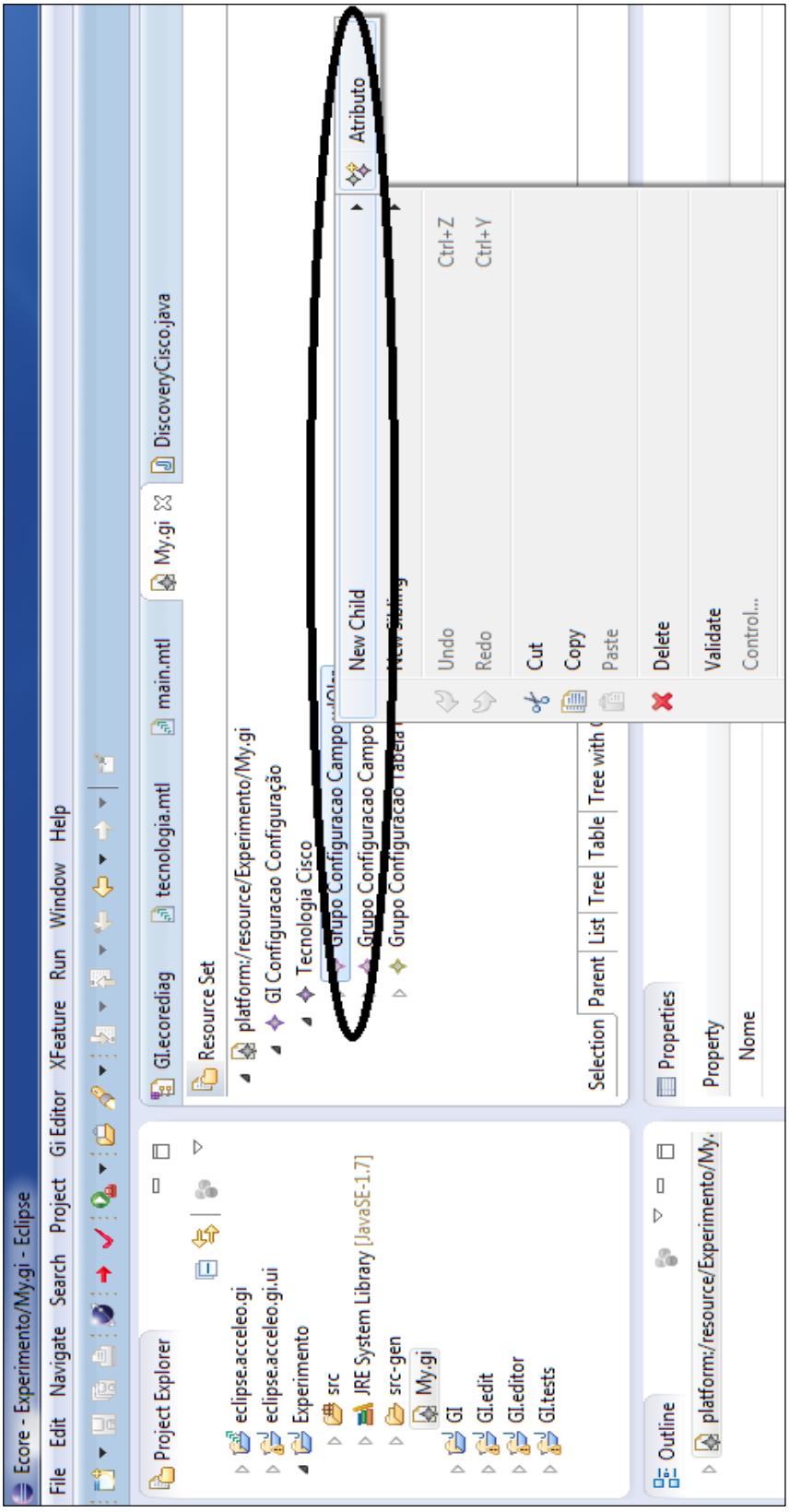

Figura B-36: Tela para selecionar o objeto Atributo

O Atributo selecionado aparece hierarquicamente abaixo do grupo de configuração. A visão de propriedades (*Properties*) exibe duas colunas: uma chamada *Property* e a outra *Value*. Na coluna *Property* aparecem as propriedades Nome e Identificação. Na coluna *Value* devem ser informados os valores para Nome e paraIdentificação do Atributo, como ilustrado na Figura B-37. Essa informação depende do especialista da rede, pois é uma informação intrínseca do atributo. Tanto o nome quanto a identificação do atributo devem ser exatamente iguais ao que estão definidos na MIB dos elementos desta tecnologia.

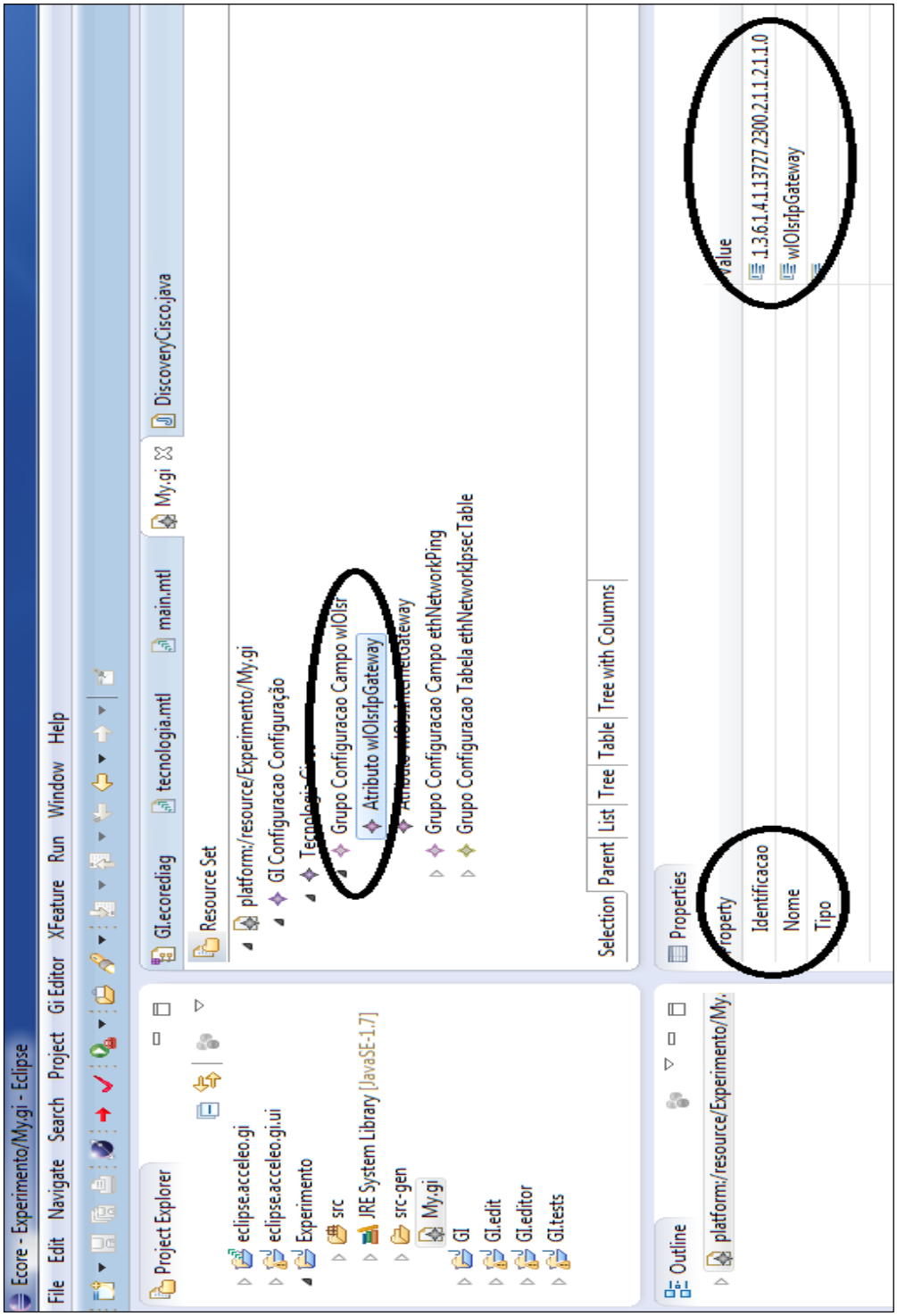

Figura B-37: Tela para especificar o Atributo

Conforme a especificação das regras de negócio no metamodelo, uma configuração poderá ter várias tecnologias. Cada tecnologia poderá ter vários grupos de configuração, sejam eles de campos ou tabelas, e cada grupo de configuração

poderá ter vários atributos. A maneira como essa hierarquia é estabelecida e a forma como os valores são definidos para cada propriedade são semelhantes.

A Figura B-38 ilustra a árvore da DSL após a especificação do modelo de gerência de configuração com todos os requisitos definidos.

Figura B-38: Tela com o modelo de gerência de configuração completo

Na Figura B-39 é ilustrada a opção (*Acceleo Model to Text → Generate Gera Codigo Java)* que deve ser selecionada para a geração de código Java, após a finalização da especificação do modelo de gerência de configuração.

Os arquivos com o código Java definidos nos *templates* serão gerados no diretório especificado, como ilustrado na Figura B-40.

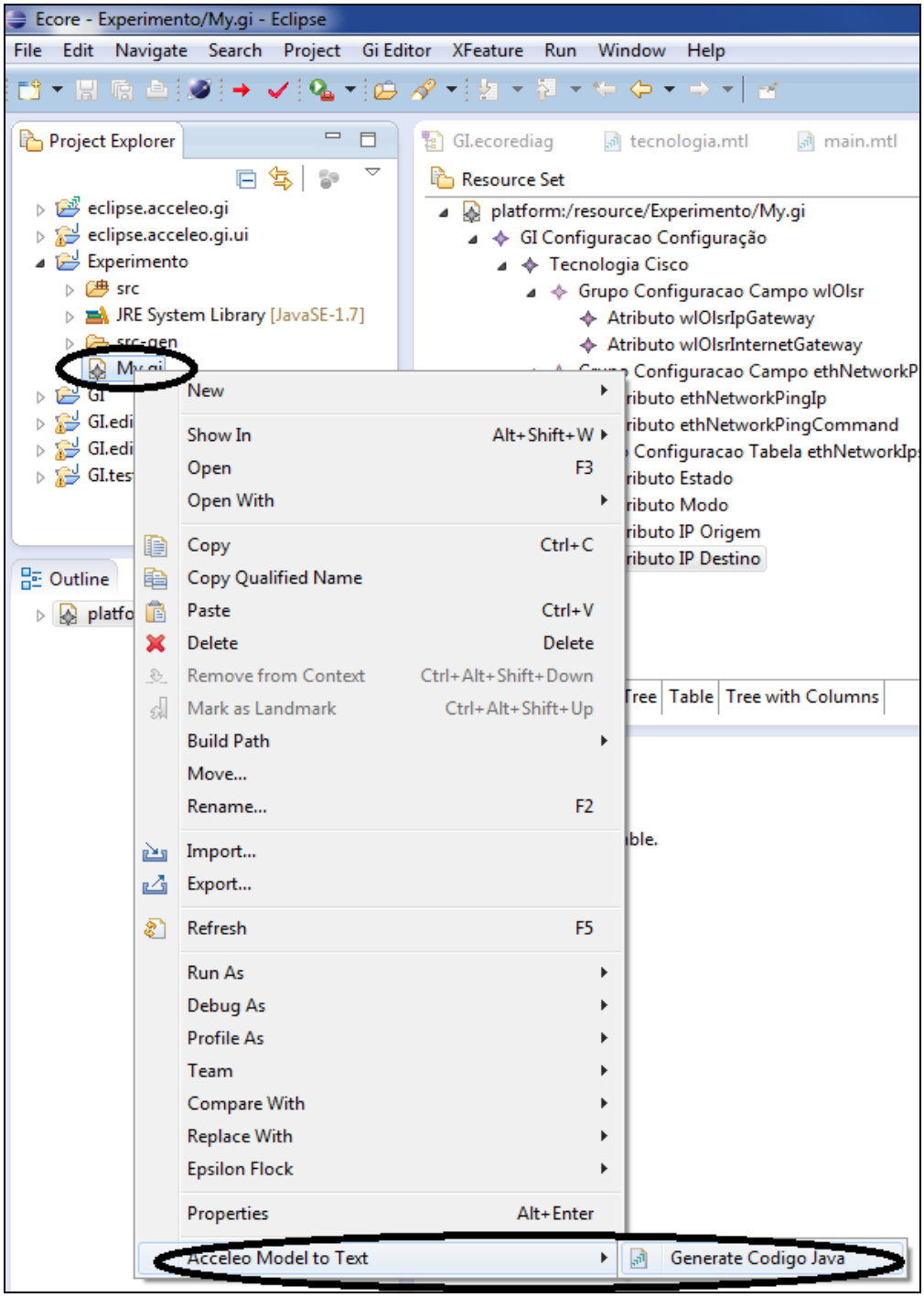

Figura B-39: Tela para solicitar a geração de código

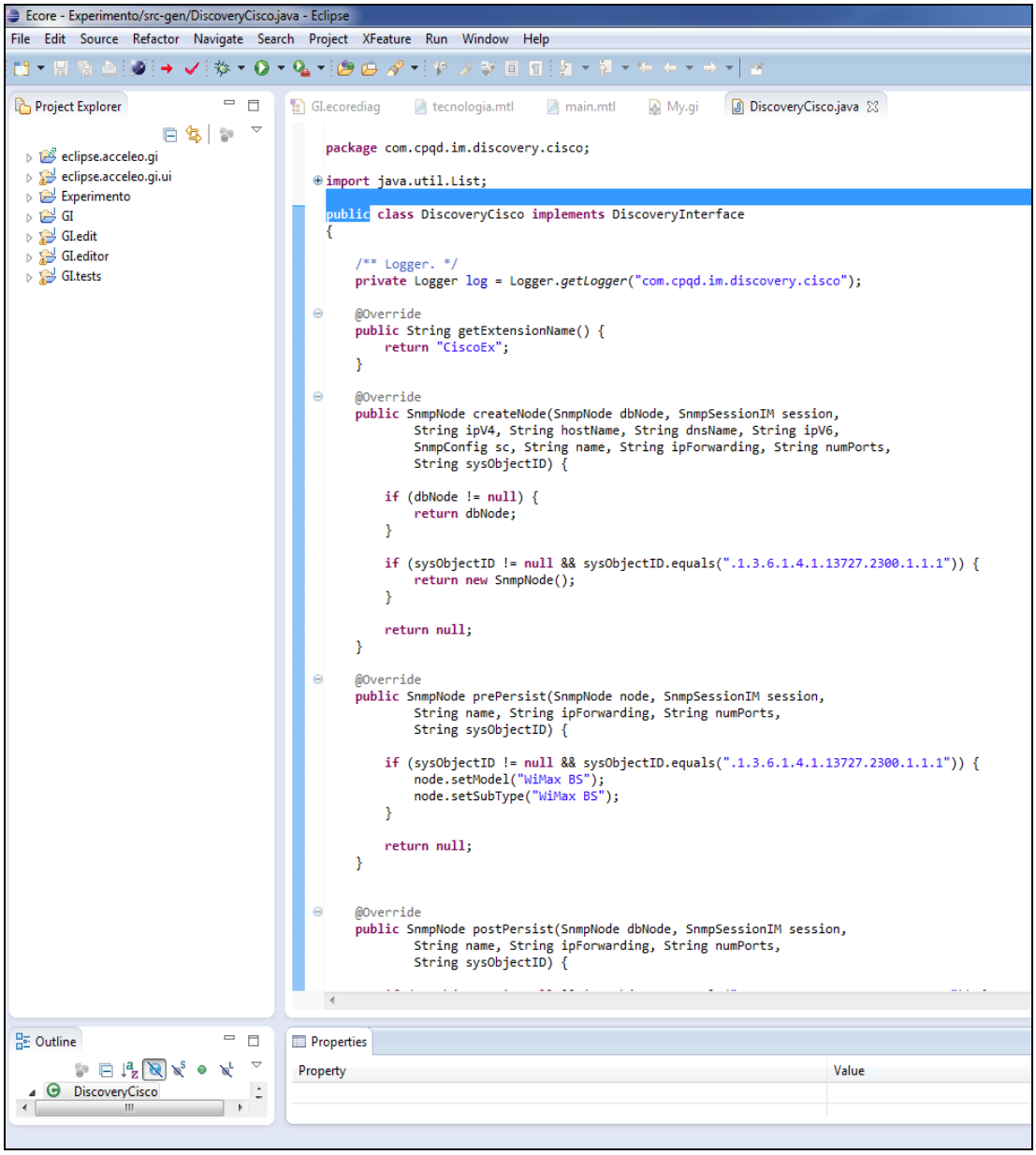

Figura B-40: Tela com o código gerado para o modelo