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

APLICAÇÃO DO PROFILE UML MARTE NA MODELAGEM DE SERVIÇOS DE PROTOCOLOS DE COMUNICAÇÃO DE TEMPO REAL

DIEGO ALVES DA SILVA

Uberlândia - Minas Gerais
2014

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

DIEGO ALVES DA SILVA

APLICAÇÃO DO PROFILE UML MARTE NA MODELAGEM DE SERVIÇOS DE PROTOCOLOS DE COMUNICAÇÃO DE TEMPO REAL

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

Área de concentração: Engenharia de Software.

Orientador:
Prof. Dr. Michel dos Santos Soares

Co-orientador:
Prof. Dr. Pedro Frosi Rosa

Uberlândia, Minas Gerais
2014

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

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Aplicação do Profile UML MARTE na Modelagem de Serviços de Protocolos de Comunicação de Tempo Real**" por **Diego Alves da Silva** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 19 de Agosto de 2014

Orientador: _____
Prof. Dr. Michel dos Santos Soares
Universidade Federal de Uberlândia

Co-orientador: _____
Prof. Dr. Pedro Frosi Rosa
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Flavio de Oliveira Silva
Universidade Federal de Uberlândia

_____
Prof. Dr. Edson Alves de Oliveira Junior
Universidade Estadual de Maringá

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

Data: Agosto de 2014

Autor:	**Diego Alves da Silva**
Título:	**Aplicação do Profile UML MARTE na Modelagem de Serviços de Protocolos de Comunicação de Tempo Real**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

_____

Autor

Dedicatória

Dedico este trabalho aos meus pais Ecio Tarcisio Alves da Silva e Welbia Maria Alves da Silva que sempre priorizaram a minha educação permitindo que eu tenha chegado até aqui.

Agradecimentos

Agradeço...

A Deus, por minha vida.

A meus pais Ecio Tarcisio Alves da Silva e Welbia Maria Alves da Silva pela dedicação, confiança, apoio, carinho e amor incondicional em todos os momentos.

Aos meus amigos e pessoas queridas que tiveram paciência pelos momentos de ausência durante o período destinado aos longos estudos .

A Nágilla Regina Saraiva Vieira pela ajuda e paciência quanto a minha ausência devido a pesquisa.

E a todos que se mostraram companheiros e, diretamente ou indiretamente, contribuíram para a realização deste trabalho.

Principalmente aos professores Michel dos Santos Soares e Pedro Frosi Rosa pelo profissionalismo, apoio, paciência, amizade e orientação em todos os momentos da realização deste trabalho.

"Frequentemente é necessário mais coragem para ousar fazer certo do que temer fazer errado."

(Abraham Lincoln)

Resumo

No contexto de sistemas distribuídos e Internet, existe a aplicação de novas tecnologias e dispositivos para melhorar o suporte à mobilidade, segurança, qualidade de serviços e *multicast*. Existem diversas abordagens para a modelagem de sistemas distribuídos para a validação do comportamento. Entretanto, muitas delas não permitem representar requisitos necessários para um protocolo de comunicação e a arquitetura em que o mesmo está inserido. Além disso, as abordagens que possuem suporte para modelagem de requisitos de tempo real são de difícil leitura, interpretação e criação dos modelos por humanos. Desta forma, este trabalho tem como objetivo apresentar duas abordagens de modelagem de protocolos de comunicação, utilizando o profile MARTE/UML, que tenha suporte para os requisitos de tempo real e que possua uma leitura simplificada. No trabalho foram criadas duas abordagens de modelagem para protocolos de comunicação da arquitetura de Internet Entity Title Architecture (ETArch). Além disso, foi criado um analisador para a linguagem VSL, que é responsável por modelar as expressões de restrições de tempo real dos modelos da segunda abordagem. Para garantir a aplicabilidade do trabalho foi feito um questionário utilizando um modelo de aceitação de tecnologia, que apresentou um resultado positivo de aceitação por parte dos usuários. Em comparação com outras abordagens, as abordagens apresentadas no trabalho são vantajosas por permitirem tanto uma modelagem visual de alto nível quanto uma modelagem algébrica para representação das restrições, permitindo uma validação matemática.

Palavras chave: marte, uml, protocolos, modelagem

Abstract

In the context of the Internet and distributed systems, there is the application of new technologies and devices to enhance the support for mobility, security, quality of service and multicast. There are many approaches to the modeling of distributed systems to behavior validation. In addition, approaches which have support for modeling of real-time requirements are difficult to read, interpret and by creation of human models. In this way, this work aims to present an approach to modeling communication protocols using the MARTE/UML profile that supports the requirements of real-time and having a streamlined reading. In this work, two modeling approaches for communication protocols of the Internet architecture Entity Title Architecture (ETArch) were created. An approach using pure UML and the second the profile MARTE. In addition, a parser for VSL language model that accounts for the expression of real time constraints of the models of the second approach was created. In order to ensure the future applicability of the proposed approach, a questionnaire based on the Technology Acceptance Model was performed. According to this model, the proposed approach was positive for most users. Compared to other approaches, the one proposed in this work is advantageous by allowing both visual modeling as an algebraic model for the representation of constraints, allowing a mathematical validation.

Keywords: marte, uml, protocols, modeling

Sumário

Lista de Figuras **xix**

Lista de Tabelas **xxi**

Lista de Abreviaturas e Siglas **xxiii**

1 Introdução **27**
1.1 Estado da Arte e Respectivos Problemas 27
1.2 Justificativa 29
1.3 Objetivos do Trabalho 30
1.4 Metodologia 31
1.5 Visão Geral da Dissertação 32

2 Fundamentos Teóricos e Trabalhos Correlatos **35**
2.1 Processo de Pesquisa de Linguagens de Modelagem 36
2.2 Linguagens de Modelagem Consideradas 39
2.3 Unified Modeling Language (UML) 43
2.3.1 Diagrama de Sequência 45
2.3.2 Diagrama de Estrutura Composta e Diagrama de Classes 48
2.4 Modeling and Analysis of Real Time and Embedded systems (MARTE) . . 49
2.5 Value Specification Language (VSL) 54

3 Entity Title Architecture (ETArch) **57**
3.1 Especificação da Arquitetura ETArch 57
3.1.1 Principais Elementos da ETArch 58
3.1.2 Principais primitivas ETCP 59
3.1.3 Principais primitivas DTSCP 60
3.2 Modelagem Utilizando Autômatos 60

4 Modelagem de protocolos *Entity Title Architecture* (ETArch) baseada em UML **65**
4.1 Motivação 65

4.2 Modelagem usando UML 66
4.3 Considerações Finais 76
4.3.1 Diretrizes de Uso 76
4.3.2 Vantagens e Limitações 77

5 Especificação de Protocolos de Comunicação de Tempo Real com UML / MARTE 79
5.1 Motivação 79
5.2 Modelagem usando MARTE 80
5.3 Analisador da Linguagem VSL 85
5.3.1 ANother Tool for Language Recognition (ANTLR) 86
5.3.2 Análise da linguagem VSL 86
5.4 Considerações Finais 90
5.4.1 Diretrizes 90
5.4.2 Vantagens e Limitações 91

6 Resultados Obtidos e Avaliação 93
6.1 Avaliação 93
6.2 Resultados do Questionário 100
6.3 Avaliação Qualitativa 102
6.4 Considerações Finais 104

7 Conclusão e Trabalhos Futuros 105
7.1 Abordagens de Modelagem 105
7.2 Resultados do Questionário 107
7.3 Respostas às Questões de Pesquisa 107
7.4 Artigo Aceito 108
7.5 Trabalhos Futuros 108

Referências Bibliográficas 111

Lista de Figuras

2.1 Processo de Pesquisa 37
2.2 Uso das linguagens de Modelagem 39
2.3 Estrutura de Diagramas UML [Group 2007] 45
2.4 Exemplo de Diagrama de Sequência 46
2.5 Exemplo de Diagrama de Classes 49
2.6 Exemplo de Diagrama de Estrutura Composta 49
2.7 Estrutura do Profile MARTE [Graf et al. 2006] 50
2.8 Estrutura do Pacote NFP [Graf et al. 2006] 51
2.9 Exemplo de utilização profile MARTE [Graf et al. 2006] 55

3.1 Exemplo Arquitetura ETArch [de Oliveira Silva 2013] 59
3.2 Entity Register - Sender 61
3.3 Entity Register - Receiver 61
3.4 Workspace Attach - Sender 62
3.5 Workspace Attach - Receiver 62
3.6 Workspace Create - Receiver 63
3.7 Workspace Create - Sender 63
3.8 Workspace Lookup - Receiver 64
3.9 Workspace Lookup - Sender 64

4.1 Diagrama de Classes - Principais Elementos da ETArch 66
4.2 Diagrama de Estrutura Composta do DTSA 68
4.3 Diagrama de Estrutura Composta - *ResourceAdapters* 69
4.4 Diagrama de Estrutura Composta - *BuildingBlocks* 69
4.5 Diagrama de Estrutura Composta - Estrutura do DTSA 69
4.6 Diagrama de Sequência - *Entity Register* 70
4.7 Diagrama de Sequência - *Workspace Create* 71
4.8 Diagrama de Sequência - *Workspace Attach* 72
4.9 Diagrama de Sequência - *Workspace Lookup* 73
4.10 Diagrama de Sequência - *DTSA Register* 73
4.11 Diagrama de Sequência - *Entity Unregister* 74

4.12 Diagrama de Sequência - *Workspace Message* 74
4.13 Diagrama de Sequência - *Workspace Adivertise* 74
4.14 Diagrama de Sequência - *Workspace Delete* 75
4.15 Diagrama de Sequência - *Workspace Detach* 75
4.16 Exemplo - Ferramenta Papyrus 76

5.1 Diagrama de Classes - Principais elementos ETArch 81
5.2 Estrutura do DTSA 82
5.3 Estrutura de Service Building Blocks 82
5.4 Cenário proposto para a modelagem 83
5.5 Operação Entity Register 84
5.6 Operação Workspace Lookup 85
5.7 Analisador VSL 88
5.8 Abordagem de Modelagem utilizando MARTE 90

6.1 Serviço *Workspace Attach* Representado por Autômato 95
6.2 Serviço *Workspace Attach* Representado por Autômato 95
6.3 Diagrama de Classes dos Elementos ETArch - UML 96
6.4 Diagrama de Classes MARTE/UML 96
6.5 Diagrama Estrutura Composta UML 97
6.6 Diagrama Estrutura Composta MARTE/UML 97
6.7 Modelo DTSA - UML 97
6.8 Modelo DTSA - MARTE/UML 98
6.9 Diagrama de Sequência Workspace Lookup - UML 98
6.10 Diagrama de Sequência Workspace Lookup - MARTE/UML 99
6.11 Cenário de Rede - MARTE/UML 99

Lista de Tabelas

2.1 Resultados de Busca no IEEExplore 38
2.2 Resultados de Busca no ADL 38
2.3 Resultado do Processo de Pesquisa 39

6.1 Resultado da Avaliação de Usabilidade 101
6.2 Resultado da Avaliação de Facilidade de Uso 101
6.3 Resultado da Avaliação de Interesse na Aplicação 102

Lista de Abreviaturas e Siglas

ACM: Association for Computing Machinery
ADL: ACM Digital Library
AICT: Advanced International Conference on Telecommunications
ANTLR: ANother Tool for Language Recognition
API: Application Programming Interface
BNF: Backus Normal Form
CAPES: CoordenaÃ§Ã£o de AperfeiÃ§oamento de Pessoal de NÃvel Superior
CCS: Calculus of Communicating Systems
CmTPNs: Communicating Time Petri Nets
CPN: Redes de Petri Coloridas
CSP: Communicating Sequential Processes
DNS: Domain Name System
DTS: Domain Title Service
DTSA: Domain Title Service Agent
DTSCP: DTS Control Protocol
ETCP: Entity Title Control Protocol
EDT: Estelle Development Toolset
EFSM: Extended Finite State Machines
EMF: Eclipse Modeling Framework
ESTELLE: Extended State Transition Language
ETArch: Entity Title Architecture
FSM: Finite-State Machines
FSM4WSR: Formal Model for Verifiable Web Service Runtime
GCM: Generic Component Model
GPU: Graphics Processing Unit
GQAM: Generic Quantitative Analysis Modeling
GRM: General Resource Modeling
HDTV: High-Definition Television
IEEE: Institute of Electrical and Electronics Engineers
IP: Internet Protocol

IPv6: Internet Protocol version 6
ISO: International Organization for Standardization
HMIPv6: Hierarchical Mobile IPv6
HLAM: High-Level Application Modeling
HPA: Heterogeneous Protocol Automata
HRM: Hardware Resource Modeling
LOTOS: Language Of Temporal Ordering Specification
LTL: Linear Temporal Logic
MARTE: Modeling and Analysis of Real Time and Embedded systems
MDTSA: Master DTSA
MIH: Media Independent Handover
MITM: Man-in-the-Middle
NATO: North Atlantic Treaty Organization
NE: Network Element
NFP: Non Functional Properties
NoC: Network on chip
OMG: Object Management Group
OCL: Object Constraint Language
OSI: Open Systems Interconnection
PAM: Performance Analysis Modeling
PROMELA: Protocol Meta Language
QoS: Quality of Service
RAMP: Reliable Adaptive Multicast Protocol
RP: Redes de Petri
SAM: Schedulability Analysis Modeling
SDL: Specification and Description Language
SDN: Software-Defined Networking
SIP: Session Initiation Protocol
SPIN: Simple PROMELA Interpreter
SPT: Schedulability, Performance and Time
SRM: Software Resource Modeling
TAM: Technology Acceptance Model
TBONE: Testbed for Optical Networking
TCP: Transmission Control Protocol
TI: Tecnologia da InformaÃ§Ã£o
TKIP: Temporal Key Integrity Protocol
UML: Unified Modeling Language
UMTS: Universal Mobile Telecommunications System
VSL: Value Specification Language

XML: Extensible Markup Language
WiMAX: Worldwide Interoperability for Microwave Access
WLAN: Wireless Local Area Network
WSNs: Wireless Sensor Networks

Capítulo 1

Introdução

1.1 Estado da Arte e Respectivos Problemas

Sistemas de software estão cada dia mais presentes na vida das pessoas. Quando existe a necessidade de utilizar mais de um software para a execução de uma tarefa, é necessário garantir que tanto o funcionamento de cada um está correto, quanto o funcionamento da integração também está. Além disso, se os sistemas de software que estão integrados estiverem desacoplados em espaço, é necessário garantir que o funcionamento da forma de comunicação entre eles seja feita de forma correta.

O conceito de Computação Ubíqua consiste em tornar os sistemas de computação tão inseridos na vida dos indivíduos que os mesmos passam a se tornar imperceptíveis [Weiser 1999]. Para isto, é necessário garantir que o funcionamento de cada um dos sistemas tenha o menor número possível de falhas para não interromper o fluxo das tarefas.

Além do conceito de Computação Ubíqua, o conceito de Internet das Coisas [Huang e Li 2010] está cada vez mais presente na vida dos indivíduos. A Internet das Coisas consiste em integrar e adicionar inteligência até nas coisas mais simples, como aparelhos domésticos, usando sensores e conexão de rede.

O cenário atual da computação contempla todos os conceitos apresentados anteriormente [Bari et al. 2013]. Um software executa diversas tarefas críticas para a vida humana. Desta forma, é necessário garantir o funcionamento dos mesmos, para isto, pode-se criar uma simulação do comportamento do software ou validar os mesmos de forma matemática.

Neste trabalho serão definidos os conceitos de software e sistema, e o uso dos mesmos deve seguir a definição aqui presente. Software deve ser coeso, ou seja, executar apenas a tarefa proposta e deve garantir que a mesma seja executada corretamente. Um sistema tem o conceito mais amplo que o software, desta forma, será considerado como o objeto de estudo, ou seja, um conjunto com um ou mais sistemas de software integrados com o objetivo de solucionar o problema do domínio [Sommerville 2006].

O conceito de tempo real, em sistemas de computação, está relacionado com a exatidão no cadenciamento da execução das tarefas. Ou seja, um software que executa de forma rápida não necessariamente é considerado de tempo real. Para que isto ocorra, as operações devem seguir restrições que definem o limite superior e inferior de um determinado recurso necessário para a execução, podendo ser tempo, gasto de energia, combustível, entre outros [Selic e Limited 1996].

Além de sistemas de tempo real, o conceito de sistemas críticos também é significativo para o contexto, os mesmos demandam grande preocupação em técnicas de modelagem que permitam a representação correta de todos os requisitos. Os sistemas críticos, de acordo com Ian Sommerville [Sommerville 2010], podem ser divididos em dois tipos básicos.

Quando o termo sistema de tempo real é utilizado, o conceito deve ser aplicado a todos os elementos do conjunto de sistemas de software que fazem parte do sistema. Ao inserir uma restrição de tempo em um sistema, todo o processo em que a mesma está envolvida deve ser considerado. Como exemplo, se um sistema de comunicação entre dois pontos possui uma restrição de tempo de resposta, tanto as restrições do remetente, quanto o meio de transmissão devem ser levados em consideração para o cálculo.

Quando mais de um software é utilizado de forma integrada é necessário padronizar a forma de comunicação dos mesmos, usando um protocolo. Um protocolo de comunicação consiste em todas as regras, formatos e procedimentos que são acordados entre dois indivíduos comunicantes. O protocolo formaliza a interação padronizando o uso de um canal de comunicação. Um protocolo pode conter definições sobre os métodos utilizados para: início e término da troca de dados, sincronização de emissores e receptores, detecção e correção de erros de transmissão, formatação e codificação de dados [Holzmann 1991a].

O padrão de protocolo de comunicação utilizado para a Internet é o *Transmission Control Protocol* (TCP)/ *Internet Protocol* (IP). Mas este modelo pode não ser adequado para a execução de aplicações que necessitam de um rigoroso controle de qualidade de comunicação, principalmente em relação a restrições de tempo, havendo a necessidade da criação de um protocolo de comunicação especializado para o problema [Day e Zimmermann 1983] [Park e Shiratori 1994]. O desenvolvimento de um protocolo de comunicação deve levar em consideração cinco aspectos principais [Holzmann 1991a]:

- O serviço provido pelo protocolo;
- As suposições a respeito do ambiente no qual o protocolo será executado;
- O vocabulário das mensagens utilizadas para implementar o protocolo;
- A codificação de cada mensagem do vocabulário;
- As regras que garantem a consistência da troca de mensagens.

Desta forma, para que um protocolo de comunicação de tempo real seja desenvolvido é necessário modelar todas as propriedades envolvidas nos cinco aspectos principais.

Para que os serviços de um protocolo de comunicação aconteçam são necesssários equipamentos provedores dos mesmos. A modelagem de um protocolo de comunicação deve ser capaz de representar todos os aspectos do mesmo e os elementos envolvidos para o serviço.

A modelagem é fundamental para a Engenharia de Software [Ludewig 2003], garantindo que todos os conceitos apresentados anteriormente possam ser representados e validados antes da construção do software. Há uma lacuna entre as necessidades e limitações de um sistema de tempo real que, geralmente, são expressos em linguagem natural pelas partes interessadas e as especificações necessárias para realmente construir software. A modelagem pode preencher esta lacuna, melhorando a comunicação entre equipes e diminuindo significativamente ambiguidades de linguagem natural.

Os modelos são criados seguindo uma metodologia de modelagem que define a estrutura, limitação e organização dos mesmos.

A partir de um modelo do software é possível entender as questões relevantes abstraindo as questões não relevantes. De acordo com Selic [Selic 2003], um modelo, na engenharia, deve possuir abstração, nível de entendimento, precisão, previsão e baixo custo.

A partir dos conceitos apresentados, os grandes problemas identificados estão relacionados à necessidade de garantir as propriedades dos sistemas de tempo real, bem como a integração das propriedades. Usando como base o conceito de modelagem apresentado, o trabalho mostra a modelagem de protocolos de comunicação de tempo real, representando as características em níveis suficientemente necessários para o entendimento.

1.2 Justificativa

Os sistemas de tempo real, apresentam necessidade de uma modelagem precisa das propriedades não funcionais, principalmente propriedades de tempo, para garantir que o orquestramento das tarefas seja eficiente e o funcionamento aconteça como esperado. As necessidades de soluções de tais problemas compõem a justificativa de realização do trabalho.

A abordagem de modelagem de software deve ser capaz de representar todos os aspectos da arquitetura do mesmo. Arquitetura de sistemas de tempo real possuem restrições de utilização de recursos que devem ser garantidas, para isto é necessário validar se realmente é possível implementar as mesmas e se elas serão implementadas de forma que não existam ambiguidades e inconsistências, surgindo a necessidade de uma abordagem de modelagem que permita a representação dos requisitos não funcionais.

Além de garantir a capacidade de representação de restrições em uma abordagem de modelagem, é necessário que tais representações sejam feitas usando mecanismos que permitam um fácil entendimento do usuário final. Mesmo que um modelo seja capaz de

representar tanto requisitos funcionais, não funcionais e componentes, é necessário que o mesmo seja entendido por todos os membros da equipe do software, seja em uma visão próxima da implementação ou em uma arquitetura de alto nível.

Os sistemas distribuídos são conectados por meio de uma rede, para a comunicação entre eles é necessário um protocolo de comunicação, o mesmo deve ser modelado utilizando os conceitos de modelagem de tempo real.

As abordagens de modelagem de protocolos de comunicação encontradas na literatura normalmente possuem foco em apenas uma necessidade, seja ela representação de restrições de forma a ser validada ou representação de um modelo de fácil leitura. No contexto de modelagem de protocolos de comunicação existe a necessidade de uma abordagem que permita representar elementos da rede em forma visual. Além disso, representar comportamento e restrições.

Com necessidades tão específicas de modelagem, as abordagens existentes podem não ser suficientes, sendo necessária a criação de abordagens para a representação das propriedades não funcionais em protocolos de comunicação.

1.3 Objetivos do Trabalho

Diante da importância em modelar as propriedades de um protocolo de tempo real para que seja possível garantir as propriedades estruturais e de comportamento, este trabalho tem como objetivo apresentar duas abordagens de utilização da linguagem de modelagem UML e o profile MARTE para representar as propriedades relevantes de protocolos de comunicação, como restrições de tempo, canais de comunicação e fluxo de dados interno de um componente, usando a definição de um conjunto de diagramas, estereótipos e uma linguagem de especificação de restrições.

Assim, objetiva-se obter soluções para as seguintes questões de pesquisa:

- Q1 - Quais são as principais abordagens de modelagem aplicadas em protocolos de comunicação?

  A resposta desta questão tem grande importância para a elaboração de um referencial teórico, garantindo o escopo do trabalho. A questão é respondida no Capítulo 2, onde são apresentadas e definidas as abordagens de modelagem utilizadas para a especificação de protocolos de comunicação.

- Q2 - As linguagens predominantes são visuais ou algébricas?

  Com base nesta resposta é possível definir qual o melhor caminho a ser seguido para a abordagem. Se uma linguagem visual possuir o maior número de trabalhos no contexto, esta forma deve ser priorizada, tendo como fato que sua aplicação foi mais aceita. A resposta para esta questão pode ser vista a cada descrição das abordagens, apresentada no Capítulo 2.

- Q3 - Como representar propriedades não funcionais nas linguagens de modelagem?

  Este aspecto consiste em um diferencial para uma linguagem de modelagem, pois, muitas vezes os requisitos não funcionais abordam propriedades quantitativas e devem ser representadas utilizando definições precisas. Esta questão é respondida no Capítulo 4 usando a linguagem UML e no Capítulo 5 usando o *profile* MARTE.

- Q4 - Como modelar aspectos temporais de protocolos de comunicação?

  Os sistemas de tempo real são dependentes de uma definição correta e não ambígua de restrições temporais. Esta questão é respondida no Capítulo 4 usando a linguagem UML e os observadores de tempo e no Capítulo 5 usando os estereótipos e expressões do *profile* MARTE.

- Q5 - Como garantir a aceitação da abordagem apresentada por parte do usuário?

  O efetivo aceite e uso da tecnologia são preocupações da abordagem apresentada no trabalho, para garantir esta propriedade um modelo de aceitação tecnológico é apresentado no Capítulo 6.

Com o propósito de desenvolver uma abordagem de modelagem que permita representar os aspectos necessários de um protocolo de comunicação de tempo real, os objetivos específicos deste trabalho são:

- definir um conjunto de estereótipos suficientes para representar elementos da arquitetura de Internet ETArch;
- utilizar expressões na linguagem VSL para a representação de propriedades não funcionais;
- definir diagramas que representem a estrutura e comportamento dos elementos da arquitetura de Internet ETArch;
- criar um analisador léxico e sintático para a linguagem Value Specification Language (VSL);
- utilizar um modelo de aceitação de tecnologia para validar a abordagem.

1.4 Metodologia

A metodologia utilizada para este trabalho consiste em realizar uma revisão bibliográfica sobre o conteúdo do mesmo, realizar a implementação de um software de análise léxica e sintática da linguagem VSL e realizar um estudo de caso utilizando o software criado para analisar sintaticamente as expressões adicionadas a um modelo criado com o *profile* MARTE.

A revisão bibliográfica segue uma linha de busca focada nas linguagens de modelagem de software que foram aplicadas no contexto de protocolos de comunicação, como LOTOS,

Redes de Petri, entre outras. Além disso, deve-se observar as publicações onde ferramentas que interpretam modelos gerados nas linguagens e realizam análise do mesmo, com geração de código.

Além de realizar a busca das linguagens, uma comparação entre elas será feita, mostrando os pontos fracos e fortes de cada uma em relação a abordagem de utilização somente da linguagem UML e da utilização do *profile* MARTE/UML, utilizando como parâmetro de comparação a aplicação da linguagem para modelar restrições em diagramas, que é o principal foco da VSL.

Para a construção da ferramenta, a primeira tarefa a ser feita é a busca de ambiguidades da linguagem VSL, ou seja, casos em que a ferramenta de verificação da linguagem VSL não consiga verificar o modelo de forma automática. Esta busca será feita por meio da análise da especificação da linguagem VSL [Cuccuru1 2011]. Caso alguma ambiguidade seja encontrada, a linguagem VSL poderá ser restringida, e será usado apenas um subconjunto da mesma. Após a análise da linguagem VSL quanto a possibilidade de ambiguidade, será implementada a ferramenta. Inicialmente foi feita a análise léxica seguida pela análise sintática. Com isso, a árvore sintática será gerada e a ferramenta indicará se as anotações presentes no modelo estão corretas sintaticamente. Após a construção da ferramenta é necessário que ela seja validada. Existem várias formas de validação de um estudo, neste caso, será utilizado um estudo empírico usando um estudo de caso [Ramananandro 2008].

Pelo estudo de caso é possível obter resultados mais consistentes sobre a ferramenta. Como estudo de caso, será utilizado um protocolo de redes de computadores. Este será modelado utilizando as abordagens propostas.

A avaliação do trabalho será feita utilizando o modelo de aceitação de tecnologia Technology Acceptance Model (TAM) [Davis 1989] que é feito usando um questionário com questões focadas na aceitação e proposta de uso da tecnologia para indivíduos que são o foco de aplicação da abordagem.

1.5 Visão Geral da Dissertação

O presente trabalho está dividido em sete capítulos.

No Capítulo 2 é apresentado o processo utilizado para criar o referencial teórico utilizado no trabalho, bem como apresentar a definição das linguagens de modelagem escolhidas durante o processo.

O Capítulo 3 apresenta a arquitetura de Internet ETArch utilizada para o estudo de caso da abordagem de modelagem criada. Além disso, a modelagem dos serviço dos protocolos é feita usando autômatos.

No Capítulo 4 é apresentada a primeira abordagem, onde são utilizados modelos UML para a modelagem dos serviços dos protocolos de comunicação da ETArch.

No Capítulo 5 é apresentada uma abordagem que tem como objetivo complementar os aspectos de modelagem que não foram satisfeitos na abordagem do Capítulo 3, usando o *profile* MARTE e a linguagem VSL.

O Capítulo 6 tem o objetivo de verificar a usabilidade das abordagens, este capítulo realiza uma avaliação da mesma usando um questionário. O questionário foi definido por um modelo de aceitação de tecnologia.

O Capítulo 7 apresenta a conclusão do trabalho, mostrando os resultados obtidos, as vantagens e limitações da pesquisa.

Capítulo 2

Fundamentos Teóricos e Trabalhos Correlatos

Existem vários fatores que podem contribuir para que uma linguagem única não exista, entre eles, necessidades de domínios específicos, mudanças de paradigmas, sistemas com diferentes níveis de tolerância a falhas e diferentes criticidades [Fraser e Mancl 2008].

Sistemas distribuídos, muitas vezes, dependem de uma modelagem específica, pois é necessário representar tanto uma visão arquitetural de alto nível, quanto integração de componentes (software e hardware), restrições de tempo, processamento, gasto de energia, entre outros. Ao longo do tempo, diversas linguagens de modelagem foram criadas ou utilizadas para modelar protocolos de comunicação. Entre elas, *Specification and Description Language* (SDL) [Belina e Hogrefe 1989], *Language Of Temporal Ordering Specification* (LOTOS) [IS8807 1988], *Extended State Transition Language* (Estelle) [Budkowski et al. 1989], *Finite-State Machines* (FSM) [Li et al. 2005], Redes de Petri [von Bochmann 1978], *Protocol Meta Language* (PROMELA) [Holzmann 1991b], entre outras.

O presente capítulo aborda os trabalhos que formam o fundamento teórico sobre o contexto de sistemas distribuídos. O mesmo é dividido em cinco seções.

A Seção 2.1 apresenta o processo de pesquisa para a escolha das linguagens de modelagem aplicadas na modelagem de protocolos de comunicação e os resultados do processo de pesquisa utilizado para a seleção das linguagens de modelagem, os resultados são apresentados divididos pelas fontes onde foram encontrados.

A Seção 2.2 apresenta a descrição de cada uma das linguagens de modelagem encontradas no processo de pesquisa, bem como exemplos de utilização das mesmas. Os exemplos são mais focados na modelagem de protocolos de comunicação.

A linguagem UML foi descrita na Seção 2.3 com os detalhes necessários para utilização no trabalho.

A Seção 2.4 descreve o *profile* MARTE, apresentando a estrutura básica de forma sucinta, com maior foco nos pacotes que serão utilizados na abordagem de modelagem presente no trabalho.

A Seção 2.5 apresenta a descrição da linguagem *Value Specification Language* (VSL), sua estrutura e um exemplo de uso.

2.1 Processo de Pesquisa de Linguagens de Modelagem

Devido à grande quantidade de linguagens de modelagem aplicadas no contexto de sistemas distribuídos, foi traçado um processo de pesquisa para definir as linguagens que seriam abordadas na pesquisa. O processo foi dividido em etapas e sub-etapas. Ao final, os resultados do processo são os trabalhos que servirão como arcabouço teórico para a presente pesquisa.

A primeira etapa consiste em uma busca pelos termos: modelagem de protocolos de comunicação, protocolos de comunicação e protocolos de comunicação de tempo real. Como próxima etapa, foi feita a filtragem dos resultados, levando em consideração o número de citações, referências e compatibilidade com o tema. O número de citações é uma forma de definir a importância da publicação no contexto acadêmico. As referências servem como base para identificar as publicações de definição conceitual das linguagens utilizadas no presente trabalho. A compatibilidade com o tema está relacionada à estrutura da linguagem escolhida, priorizando as linguagens que possuem capacidade de representação de protocolos de comunicação.

A primeira etapa gera como resultado a lista de linguagens escolhidas para o trabalho juntamente com as publicações clássicas das mesmas. A segunda etapa consiste na busca de publicações a respeito das linguagens escolhidas, SDL [Belina e Hogrefe 1989], LOTOS [IS8807 1988] [Chunmei et al. 2010] [Ganea et al. 2012], Estelle [Budkowski et al. 1989], FSM [von Bochmann 1978], PROMELA e Redes de Petri [Li et al. 2005] [Lee e Baik 2008] [Simonak et al. 2009]. As fontes de busca foram *IEEExplore* e *ACM Digital Library* (ADL).

A Fig.2.1 apresenta o processo utilizado para a escolha das linguagens que serão definidas ao longo do referencial teórico. A organização dos trabalhos foi feita com base no conteúdo do mesmo. O processo de busca de trabalhos em cada uma das linguagens escolhidas possui três etapas básicas:

1. Utilizar termos de busca relacionando a linguagem escolhida à modelagem de protocolos, como: “LOTOS protocol modeling "e “protocol modeling ";
2. Filtrar trabalhos publicados após o ano de 2008, que consiste em artigos com até seis anos de publicação. Esta data tem como objetivo filtrar abordagens que ainda possam ser aplicadas atualmente, principalmente com relação às ferramentas de modelagem;
3. Verificar a existência de termos de interesse no título e no *abstract*. De acordo com uma análise dos resultados das primeiras buscas, foi definido que ao utilizar a

fonte *ACM Digital Library* apenas os 80 primeiros resultados, caso existam, terão o *abstract* analisados, pois, após este índice, o número de resultados de interesse torna-se irrelevante.

Ao final deve-se encontrar as publicações de definição das linguagens, trabalhos que tenham o ano anterior ao definido no filtro mas que são relevantes e trabalhos necessários para o entendimento de publicações do grupo de interesse.

A Tabela 2.1 apresenta a quantidade de publicações encontradas no repositório *IEEEExplore* para cada uma das linguagens de interesse. Os resultados são apresentados em três colunas, as mesmas representam os filtros de busca que foram descritos anteriormente.

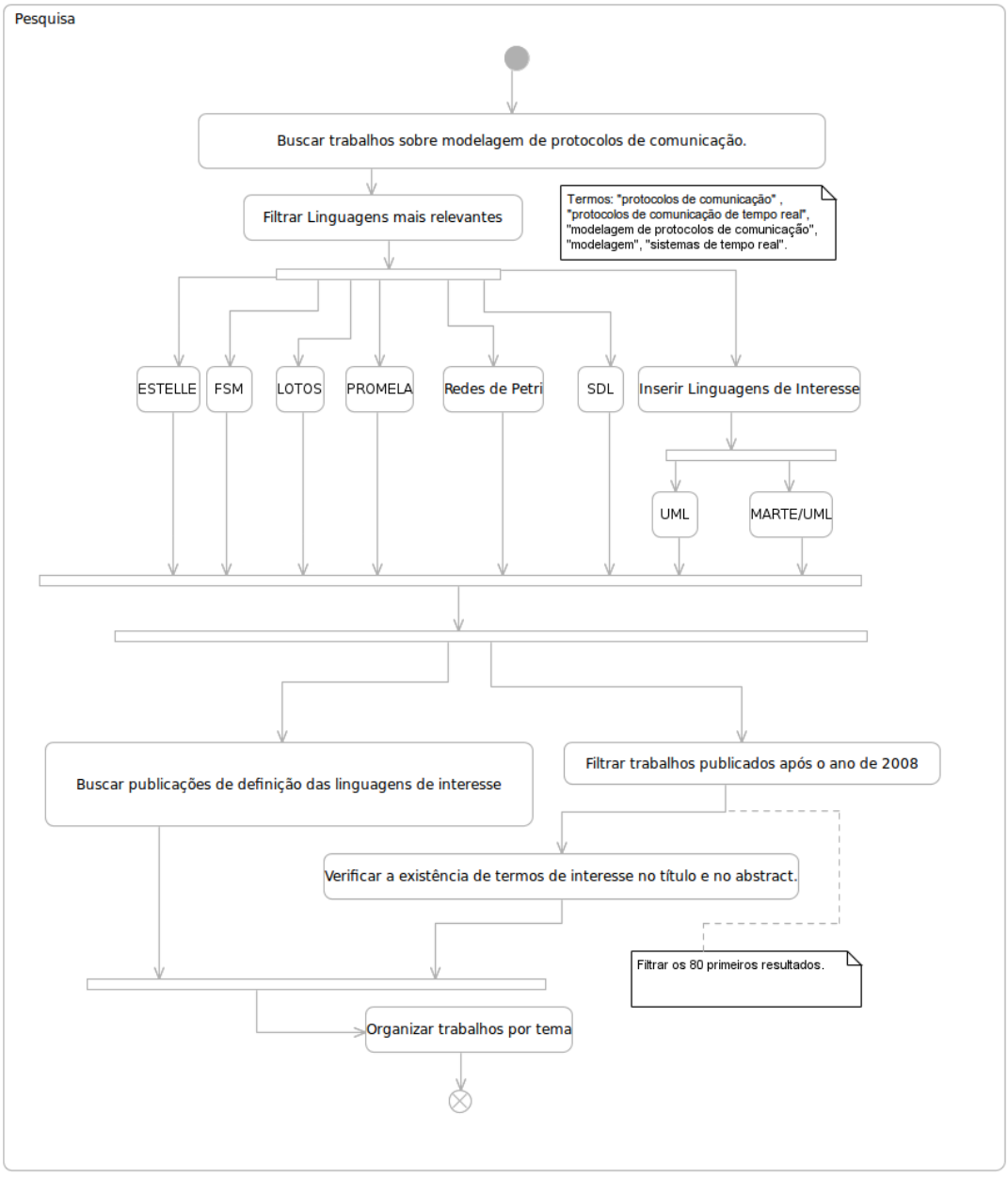

Figura 2.1: Processo de Pesquisa

Tabela 2.1: Resultados de Busca no IEEExplore

Linguagem de Interesse	1º Etapa - IEEExplore	2º Etapa - IEEExplore	3º Etapa - IEEExplore
Estelle	40	4	3
FSM	90	28	8
LOTOS	52	8	5
UML/MARTE	6	6	6
PROMELA	21	9	7
Redes de Petri	327	110	14
SDL	67	12	8
UML	102	44	9

O segundo filtro, que leva em consideração as publicações mais recentes, em comparação com o primeiro, mostra o interesse da aplicação da respectiva linguagem na modelagem de sistemas distribuídos no período de tempo analisado.

Tabela 2.2: Resultados de Busca no ADL

Linguagem de Interesse	1º Etapa - ADL	2º Etapa - ADL	3º Etapa - ADL
Estelle	187	87	2
FSM	1032	562	3
LOTOS	499	149	9
UML/MARTE	188	6	6
PROMELA	468	237	1
Redes de Petri	3407	1653	7
SDL	828	303	7
UML	4897	2610	9

A Tabela 2.2 apresenta o resultado da busca de publicações no repositório *ACM Digital Library*. É importante ressaltar a grande queda na quantidade de publicações entre a segunda e a terceira etapa. Isto ocorreu porque a busca levou em consideração apenas as 80 publicações com maior nível de relevância ao termo de busca. Este número foi baseado em uma observação empírica no resultado de busca de quatro linguagens, onde a partir do índice 60, os resultados eram em sua maioria descartados por falta de compatibilidade com o tema desejado. Para evitar que resultados relevantes sejam ignorados, uma margem de erro de 20 resultados foi adicionada.

A Tabela 2.3 apresenta o resultado final da busca, juntamente com a execução da tarefa final, que consiste em adicionar manualmente as publicações de definição da linguagem de interesse, bem como as principais aplicações e avaliações. Desta forma, o número de publicações presentes na coluna *Adicionados* representa os trabalhos relevantes que foram adicionados ao trabalho, porém, não se encaixaram na primeira etapa de busca.

A Fig. 2.2 apresenta um gráfico da aplicação das linguagens de modelagem em sistemas distribuídos no período de tempo filtrado. Como o processo de busca possui um filtro para publicações com o ano maior ou igual a 2008, pode-se notar que as linguagens UML e

Tabela 2.3: Resultado do Processo de Pesquisa

Linguagem de Interesse	Total IEExplore	Total ADL	Adicionados	Total
Estelle	3	2	1	6
FSM	8	3	8	19
LOTOS	5	9	1	15
UML/MARTE	6	6	3	15
PROMELA	7	1	4	12
Redes de Petri	14	7	9	30
SDL	8	7	2	17
UML	9	9	9	25
Total				139

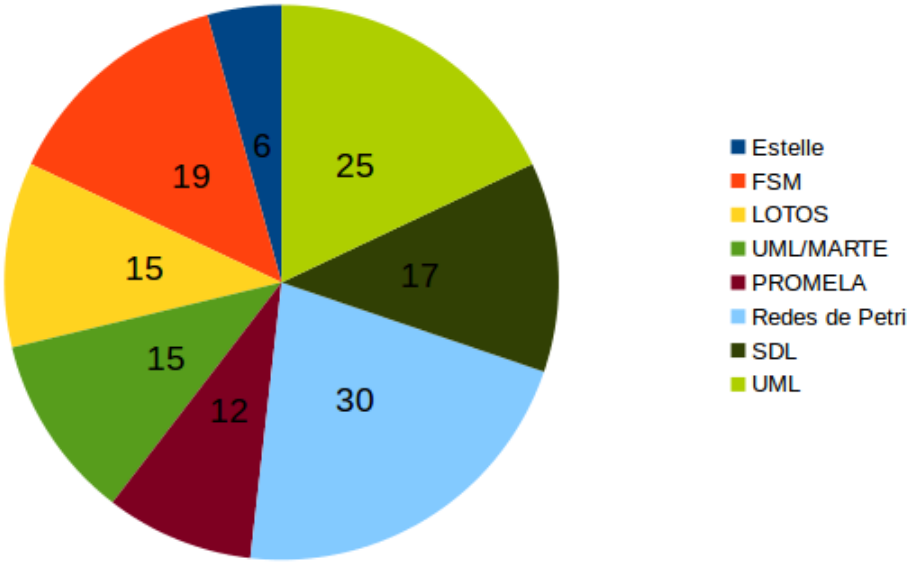

Figura 2.2: Uso das linguagens de Modelagem

Redes de Petri se destacam como mais citadas. Este dado mostra o crescente interesse pelo uso da linguagem UML e do *profile* MARTE.

A partir dos resultados encontrados, um universo de pesquisa foi formado e o referencial teórico de linguagens de modelagem aplicadas a sistemas distribuídos passa a possuir uma linha de pesquisa a ser seguida.

2.2 Linguagens de Modelagem Consideradas

A partir do arcabouço teórico que foi definido, é necessário buscar as publicações que serão mais relevantes para definir as linguagens e trabalhos que serão utilizados como alicerce para a construção da pesquisa. As linguagens de modelagem escolhidas serão utilizadas para uma futura comparação com as linguagens de interesse. É necessário

definir: a estrutura básica, elementos e exemplos de utilização das linguagens escolhidas em projetos relacionadas a modelagem de protocolos de tempo real e sistemas embarcados.

Ao longo desta seção, as linguagens de modelagem são descritas, tanto definição quanto exemplos de uso, tomando como base os trabalhos encontrados no processo de pesquisa.

Finite-State Machines (FSM)

Uma FSM [Gurp e Bosch 1999] é um modelo matemático para a representação de uma máquina de estados abstrata. Com FSM é possível modelar diversos tipos de problemas, entre eles, protocolos de comunicação.

A modelagem de protocolos de comunicação utilizando FSM consiste em dividir o sistema em componentes comunicantes em que cada componente pode ser uma máquina de estados. Uma vantagem em utilizar a representação por FSM está na verificação de propriedades do modelo. A principal limitação está na quantidade de estados que representam operações entre os componentes.

As máquinas de estado foram aplicadas para resolver problemas de filtragem das variáveis de entrada, algumas propriedades complexas que não podem ser descritas na lógica temporal e verificar as propriedades relacionadas do módulo de controle de fluxo de dados, superando as limitações dos métodos tradicionais [Hua et al. 2012].

As máquinas de estado finitas foram utilizadas para modelar diversos sistemas distribuídos, como controle de Internet para análise de reconfiguração de pilhas de protocolos [Chen et al. 2010] e IP reutilizável [He et al. 2009].

Timed Automata [Alur e Dill 1994] foi aplicado para modelar protocolos de comunicação, o mesmo consiste em uma abordagem de máquinas de estados com tratamento de tempo e clock. A ferramenta que permite a modelagem e validação de especificações em Timed Automata é chamada de UPPAAL [Wu et al. 2009] [Al-Bataineh et al. 2012]. Em [Limal et al. 2007], o autor apresenta uma abordagem com o objetivo de validar o gerenciamento de redundância de Ethernet PowerLink High Availability utilizando a ferramenta UPPAAL e Timed Automata.

Estelle

A linguagem Estelle [Budkowski et al. 1989] é baseada em uma máquina de estados finitos estendida (EFSM) e na linguagem Pascal, com modificações focadas na modelagem de protocolos de comunicação. Um modelo na linguagem Estelle possui módulos, transições, canais e estruturação. Isto permite que um sistema seja visto como um conjunto de componentes comunicantes, os módulos podem possuir variáveis, transições e pontos de interação formando sub-módulos que podem se comunicar através dos canais.

Em [Lei et al. 2007], a linguagem Estelle foi utilizada para modelar a mobilidade IPV6. A modelagem da mobilidade e gerenciamento da vida útil foram pontos críticos

na especificação, no entanto, a Estelle foi capaz de especificar a mobilidade com a capacidade de conexão e desconexão dinâmicas. O modelo foi validado e simulado, usando a ferramenta Estelle Development Toolset (EDT).

Como a linguagem Estelle foi criada com base na FSM, ela possui limitações para representar de forma expressiva as propriedades temporais de sistemas distribuídos. Para lidar com este problema, foi proposta por [Lai e Tsang 2007] uma extensão da linguagem para lidar com estas limitações, a Time-Estelle. Para que seja possível a verificação da especificação escrita em Time-Estelle, a mesma foi transformada para CmTPNs [Bucci e Vicario 1995] que consiste em uma Rede de Petri com suporte a representação de tempo.

Como estudo de casos, a linguagem Time-Estelle foi utilizada para a modelagem do protocolo Reliable Adaptive Multicast Protocol (RAMP) [Koifman e Zabele 1996] que consiste em um protocolo aprimorado para o uso de mais de uma rede *gigabit* para Testbed for Optical Networking (TBONE). O modelo foi transformado para CmTPNs com o objetivo de ser validado [Lai e Tsang 2007].

A linguagem Estelle foi utilizada para a criação de outros métodos de modelagem, como o FSM4WSR [Li et al. 2011], que foi criado a partir da mesma com o objetivo de gerar automaticamente a implementação do protocolo *Web Service* e construir um *Web Service* verificável em tempo de execução, chamado *XServices SODLRuntime* [Li et al. 2011].

Language Of Temporal Ordering Specification (LOTOS)

LOTOS [IS8807 1988] é em uma linguagem de modelagem formal que foi publicada como norma ISO 8807:1989 para a especificação de sistemas distribuídos. A linguagem consiste, em uma parte, de um processo algébrico de Calculus of Communicating Systems (CCS) e Communicating Sequential Processes (CSP), e a outra parte que é baseada no tipo abstrato de dados da linguagem ACT ONE [de Meer et al. 1992].

LOTOS permite a criação de várias formas de transformação e validação de protocolos de comunicação. Existem alguns exemplos de serviços de protocolos implementados em LOTOS [Khendek et al. 1989] [Kant et al. 1996] [Kapus-Kolar 1999], mas todas as abordagens compartilham do mesmo problema, a complexidade dos modelos. Como o modelo é criado nas primeiras fases do projeto, a complexidade no mesmo pode dificultar a construção de uma especificação completa do sistema [Bolognesi e Brinksma 1987].

A possibilidade de modelagem de propriedades como exclusão mútua, predicados lógicos e processos permite que a linguagem LOTOS seja aplicada em diversos contextos, como a modelagem de composição de *Web Services* [Zhao et al. 2012].

A linguagem LOTOS pode ser utilizada como meio para a geração de artefatos, podendo ser código executável ou modelos. Além disso, outras especificações podem ser transformadas na linguagem LOTOS para verificações de propriedades. Em [Salaun et al. 2012] foi feita a codificação de diagramas de colaboração na linguagem LOTOS, desta

forma, foi possível verificar as propriedades temporais de *Web Services*.

Protocol Meta Language (PROMELA)

PROMELA é uma linguagem formal e algébrica de modelagem, criada para especificar processos ou protocolos. A linguagem possui diversos recursos que permitem representar a comunicação entre componentes, como canais de comunicação. A semântica da PROMELA é definida em um modelo que deve conter pelo menos um processo, pode conter variáveis, canais e um motor semântico. O motor semântico de um modelo é responsável por definir como as ações dos processos são escalonadas de acordo com o tempo [Holzmann 1991b].

Para a verificação formal de especificações PROMELA, pode ser usado o Simple PROMELA Interpreter (SPIN), que permite a interpretação do modelo criado. A execução dos processos definidos é feita por etapas, sendo que em cada etapa um *statement* é selecionado de forma arbitrária, sendo possível definir se o *statement* é executável ou não.

Para protocolos robustos, com QoS, diversas primitivas e recursos contribuem na confiabilidade do mesmo, assim, o modelo pode ficar extenso e tão complexo quanto a implementação [Holzmann 1991b]. Além disso, uma visão arquitetural de alto nível não é possível de forma direta.

Levando em consideração a grande capacidade do interpretador SPIN, vários trabalhos utilizam a transformação de outras linguagens em PROMELA para que a especificação seja validada com o SPIN, como em [Palaniveloo e Sowmya 2011], em que a metodologia Network on Chip (NoC) [Borrione et al. 2007] é modelada utilizando Heterogeneous Protocol Automata (HPA) e, depois, a especificação é transformada manualmente para PROMELA para que seja validada com o interpretador SPIN. Desta forma, com apenas um modelo foi possível validar todas as propriedades desejadas da metodologia NoC.

O Worldwide Interoperability for Microwave Access (WiMAX) [Martin e Westall 2006] é um padrão para oferecer banda larga sem fio, mas, assim como outros padrões de Internet sem fio, a WiMax é vulnerável a ataques, como o Man-in-the-Middle. A linguagem PROMELA foi utilizada para a modelagem dos protocolos de segurança da tecnologia WiMAX, neste trabalho, ela foi utilizada juntamente com Linear Temporal Logic (LTL) [Komu et al. 2012].

Redes de Petri

Redes de Petri (RP) são uma linguagem gráfica de modelagem criada em 1962 e largamente utilizada para a modelagem de protocolos de comunicação [Saleh 1996] [Murata 1989].

As Redes de Petri proveem formas para a modelagem de protocolos de comunicação, suportam diretamente a modelagem de concorrência, compartilhamento de recursos e

eventos assíncronos. A falta de composicionalidade é a principal limitação encontrada nos modelos criados utilizando RP [Anisimov e Koutny 1995], [Lakos et al. 1995].

Juntamente com a transmissão de dados sem fio de forma aberta, surgiu uma preocupação com a segurança dos dados, para isto, foram criados padrões de segurança, como IEEE 802.11 [O'Hara e Petrick 1999]. O protocolo de segurança e criptografia *Temporal Key Integrity Protocol* (TKIP) foi modelado utilizando RP para analisar as lacunas de segurança a ataques [Min e Siyu 2010].

As RP em sua forma mais básica não permitem a modelagem de restrições temporais [El-Fakih et al. 2000]. Para lidar com a limitação na modelagem de tempo foram propostas as Redes de Petri com Suporte a Modelagem Temporal (TPR) [Masri et al. 2008]. Mas, apesar de permitir a modelagem temporal, as TPR não permitem a modelagem de restrições em diferentes bases temporais, criação de expressões estatísticas e expressões de tempo complexas, ou seja, que possuem operadores que podem lidar com estruturas de dados como intervalos, tuplas, escolhas e coleções utilizando uma transformação direta entre bases temporais.

Além das RP com suporte a modelagem temporal, existem as Redes de Petri Coloridas (CPN) [Sun et al. 2010] que aumentam a capacidade de representação de uma RP básica adicionando diferenciação nos tipos dos *tokens*, que podem representar estruturas de dados. As CPN foram aplicadas em diversos trabalhos de especificação de protocolos de comunicação, como na modelagem e análise de transações do protocolo Session Initiation Protocol (SIP) [Bai et al. 2011] e do gerenciamento hierárquico mobile de IPv6 (HMIPv6) [Sun et al. 2010].

Specification and Description Language (SDL)

SDL é uma linguagem gráfica orientada a objetos que foi criada com o objetivo de modelar sistemas distribuídos. A semântica SDL é baseada em máquinas de estado abstratas.

O desenvolvimento de padrões para comunicação sem fio como IEEE 802.11 e Universal Mobile Telecommunications System (UMTS) [Samukic 1998] deve garantir que propriedades de tempo sejam descritas corretamente e não sejam ambíguas, para isto, em [Nogueira et al. 2010], a linguagem SDL foi utilizada para especificar e validar aspectos de integração dos padrões UMTS e 802.11 de Internet.

2.3 Unified Modeling Language (UML)

A linguagem de modelagem UML [Booch et al. 2005] é utilizada como notação padrão para a indústria de software [Lange e Chaudron 2004]. Existem várias soluções que utilizam a modelagem nesta linguagem como base [Jaragh e Saleh 2001] [Sekaran 2001]

[Bagnato et al. 2013]. A UML é uma linguagem extensível, ou seja, permite a criação de novos perfis que utilizem as estruturas de dados básicas, diagramas e tipos. Os mecanismos de extensibilidade permitem personalizar e estender a linguagem UML, adicionando novos blocos de construção, criando novas propriedades e especificando nova semântica, a fim de tornar a linguagem adequada para cada domínio de problemas. Existem três mecanismos comuns de extensibilidade que são definidos pela UML: estereótipos, *Tagged Values* e restrições.

Estereótipos permitem ampliar o vocabulário da UML, de modo que se torna possível criar novos elementos do modelo derivando dos elementos já existentes, mas que podem possuir propriedades específicas que são adequadas para um problema de domínio específico. Os estereótipos também permitem a criação de novos símbolos gráficos [Group 2007].

Tagged Values são propriedades para especificar pares de chave valor dos elementos do modelo, onde as palavras-chave são atributos, e ainda, permitem estender as propriedades de um bloco de construção UML para criar novas informações na especificação do elemento. Eles podem ser definidos para os elementos do modelo existentes para estereótipos individuais. Um *tagged value* não é igual a um atributo de classe. Em vez disso, ele pode ser considerado como um meta-dado, já que seu valor se aplica ao próprio elemento e não às suas instâncias [Group 2007].

As restrições são propriedades para especificar a semântica e/ou as condições que devem ser realizadas em todos os momentos para os elementos de um modelo. Elas permitem que se estenda a semântica de um bloco de construção UML adicionando novas regras ou modificando as já existentes [Group 2007].

Na linguagem UML existe a definição de vários diagramas. O objetivo dos diagramas UML é cobrir a maior parte possível do processo de desenvolvimento de software, desde a definição dos requisitos e cenários de execução até a implantação.

A sintaxe da UML é muito extensa, com diversos tipos de dados, estereótipos e diagramas. Uma diferença da linguagem UML para outras linguagens de modelagem é a presença de vários tipos de diagramas para representar diversos níveis de abstração ou contexto. Neste trabalho não serão utilizados todos os diagramas da linguagem UML, o trabalho está focado nos diagramas de Sequência, Estrutura Composta e Classe.

Um modelo é constituído de um *Elemento* e os descendentes de um *Elemento* devem prover semântica suficiente para a representação dele. Todo *Elemento* pode possuir outros *Elementos*, caso ele seja removido do modelo, todos os *Elementos* dele também devem ser removidos do mesmo.

Um *Elemento* pode possuir um comentário, porém, o comentário não deve adicionar semântica ao elemento. Um *Relacionamento* consiste em uma relação entre dois elementos. Desta forma, todo *Relacionamento* deve possuir um *Elemento* de origem e um de destino.

Os *Templates* são elementos de referência que são parametrizados por outros elementos

do modelo. Um *Namespace* provê um contexto para que *Elementos* possam ser referenciados a partir de um nome. A partir de um *Namespace* é possível fazer a importação de elementos de outros *Namespaces*.

Um *Tipo* representa um conjunto de valores. Um elemento que possui um *Tipo* é restringido a representar apenas valores de tal conjunto de valores. A representação de um elemento com tipo pode ser feita utilizando cardinalidade, esta propriedade permite definir um intervalo não negativo da quantidade de elementos do mesmo tipo.

Uma *Restrição* é uma validação que indica que uma restrição deve ser satisfeita para que exista uma realização correta de um modelo. A restrição é anexada a um conjunto de elementos para representar uma semântica adicional nos mesmos.

A dependência entre elementos implica que a semântica do elemento dependente não é completa sem a semântica do elemento que fornece as propriedades semânticas.

Cada um dos elementos apresentados anteriormente são utilizados para criar uma especificação na linguagem UML e todos eles possuem uma sintaxe bem definida que pode ser encontrada na especificação da linguagem UML [Group 2007].

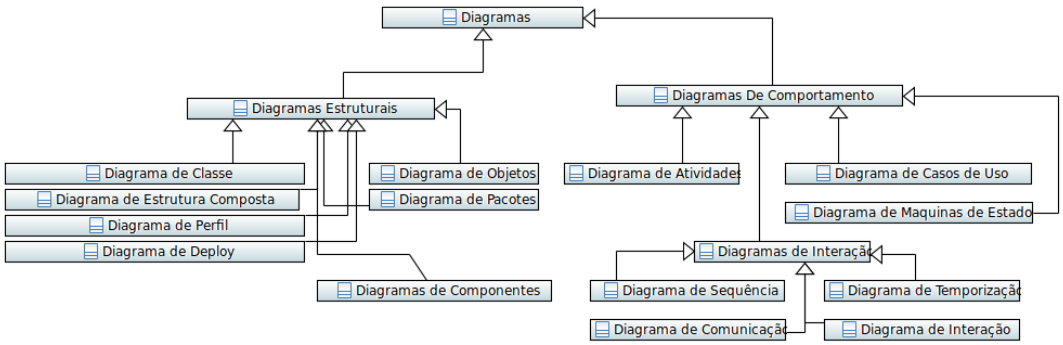

Figura 2.3: Estrutura de Diagramas UML [Group 2007]

A Fig. 2.3 apresenta a estrutura de Diagramas da Linguagem UML. Para a modelagem de um sistema de domínio específico é possível a escolha de um conjunto de diagramas que consiga representar todo cenário desejado. Levando em consideração a necessidade de modelagem de estruturas usando o paradigma de orientação a objetos, o Diagrama de Classe foi escolhido. Para a representação da estrutura em forma de módulos e o fluxo de dados, o Diagrama de Estrutura Composta foi escolhido. Devido a necessidade de representação de comportamento em ordem cronológica e adição de restrições temporais, o Diagrama de Sequência foi escolhido para o trabalho.

2.3.1 Diagrama de Sequência

Interações são utilizadas para obter melhor controle de uma situação de interação para um indivíduo ou um grupo que precisa alcançar um entendimento comum da situação.

Na modelagem de protocolos, as interações também podem ser usadas durante a fase de projeto mais detalhada, onde a comunicação entre os serviços deve ser configurada de acordo com protocolos formais. Quando o teste é realizado, os vestígios do sistema podem ser descritos como interações e comparados com os das fases anteriores.

Interações são mecanismos comuns para descrição de sistemas que devem ser interpretados com diferentes níveis de detalhes. Usando os diagramas de interação é possível descrever o comportamento do sistema em nível de troca de mensagem e definições de fluxo de execução de elementos conectados. O diagrama de interação que melhor representará a ação necessária para a modelagem dos serviços de um protocolo é o Diagrama de Sequência, pois nele é possível relacionar os classificadores com o comportamento, criando e gerenciando instâncias e comunicação entre as mesmas.

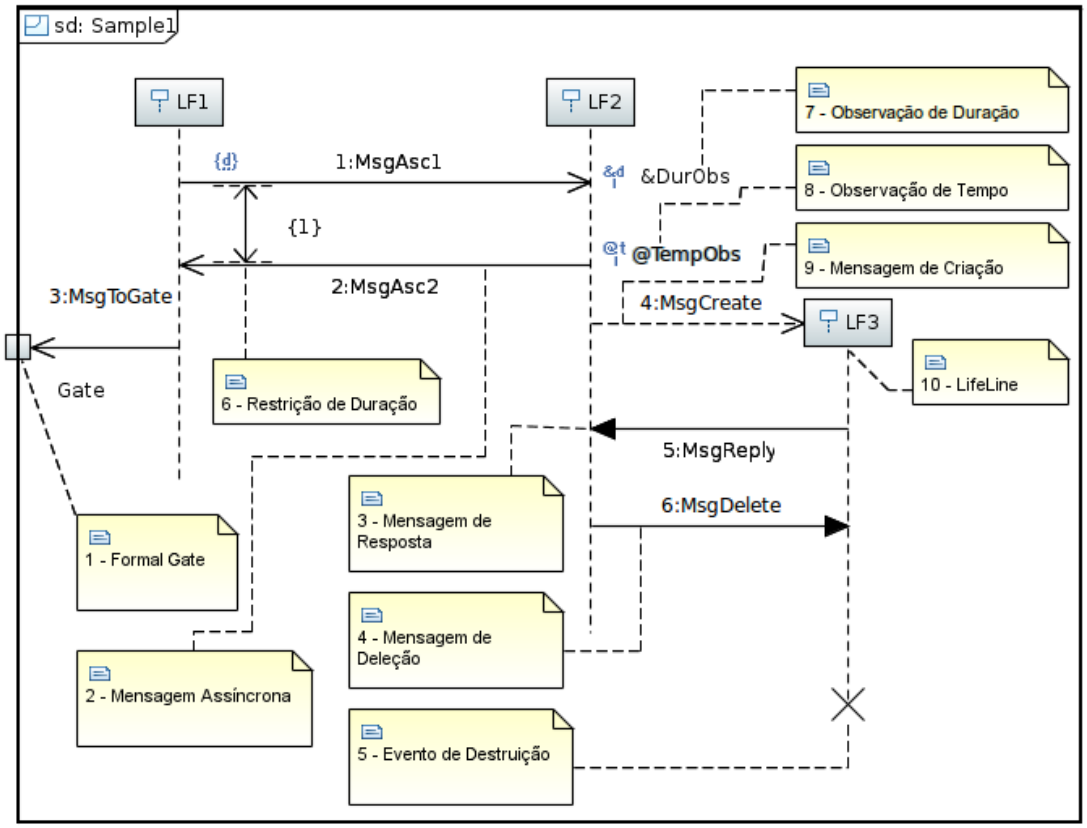

Figura 2.4: Exemplo de Diagrama de Sequência

A Fig. 2.4 apresenta um exemplo de Diagrama de Sequência, nos comentários estão descritos os elementos e os mesmos são referenciados usando o número presente na Fig. 2.4:

- Fig. 2.4 - 1: Representa um *Formal Gate* que consiste em um ponto de fim da mensagem no frame. O *Formal Gate* pode ser utilizado para conectar mais de uma Interação;

- Fig. 2.4 - 2: Representa uma mensagem assíncrona, podendo ser um sinal ou chamada assíncrona, desta forma, os elementos ficam desacoplados na ordem temporal, pois a próxima mensagem não será necessariamente uma resposta;
- Fig. 2.4 - 3: Representa uma mensagem de resposta, para que ela exista é necessário que uma mensagem síncrona tenha sido feita anteriormente no sentido contrário;
- Fig. 2.4 - 4: Representa uma mensagem de destruição que consiste na designação do término de uma *LifeLine*;
- Fig. 2.4 - 5: Representa a destruição da instância representada pela *LifeLine*;
- Fig. 2.4 - 6: Representa uma restrição de duração, ou seja, um intervalo de mensagens entre dois elementos;
- Fig. 2.4 - 7: Representa uma observação de duração, relativa entre dois elementos;
- Fig. 2.4 - 8: Representa uma observação de tempo para a entrada ou saída de um elemento específico, retornando o tempo em que o evento ocorre;
- Fig. 2.4 - 9: Representa a designação da criação de um elemento do tipo *LifeLine*;
- Fig. 2.4 - 10: Representa uma *LifeLine* que consiste na linha do tempo do elemento representada pela mesma.

Os *Combined Fragments* consistem em formas de modificação na execução dos elementos nele contidos, podendo alterar o fluxo, definir ordem e número de execuções. Os *Combined Fragments* mais relevantes para a modelagem de protocolos de comunicação são:

- *Alt:* Representa que o comportamento é uma escolha de elementos, um dos elementos deve ser escolhido, para isto, o elemento deve possuir uma expressão lógica de avaliação;
- *Opt:* Representa uma escolha de execução ou não do fragmento. A semântica deste fragmento é semelhante ao *Alt*, porém, não existem dois fragmentos;
- *Break:* Este fragmento possui uma validação e permite a interrupção da execução a partir da mesma;
- *Par:* Consiste na junção de comportamentos intercalando operandos, a ordem de execução não deve gerar efeitos colaterais na execução;
- *Seq e Strict:* Representam uma ordem fraca e forte, respectivamente, na ordem da sequência de execução de cada operador;
- *Loop:* Consiste na repetição dos eventos do fragmento até que a condição de parada seja suprida.

2.3.2 Diagrama de Estrutura Composta e Diagrama de Classes

O Diagrama de Classes é usado para definir interfaces, classes e associações. Interfaces representam um conjunto de operações de serviços públicos. As classes podem usar interfaces.

Diagramas de Estrutura Composta são usados para definir classes de estruturas internas e portas de comunicação, onde uma porta oferece ou exige um serviço.

Como os elementos básicos de ambos os diagramas, classe e estrutura composta, são os mesmos, a definição será feita em conjunto. Os elementos são chamados de *Structured Classifiers*.

A estrutura de um *Structured Classifiers* é formada por *ConnectableElement*, que consiste em uma meta-classe abstrata. Em um modelo, cada participante representa um *ConnectableElement* e eles podem ser conectados por *Connectors*. Um conector pode ou não possuir um tipo.

Um classificador consiste em uma extensão de um *Structured Classifiers* com a possibilidade de possuir uma ou mais estruturas *Ports*. A estrutura *Ports* representa um ponto de interação com o qual um classificador se comunica com o ambiente.

Uma classe (*Class*) é um tipo de classificador que é caracterizado por propriedades, operações, portas e conectores. Uma instância de uma classe é chamada de objeto. Os objetos podem possuir atributos que também são uma classe e também dão origem a outros objetos. Quando um objeto é destruído, todos os objetos que representam as classes dos atributos internos também são destruídos recursivamente.

Uma associação especifica uma relação semântica que pode ocorrer entre instâncias de um determinado tipo. Uma associação possui pelo menos dois elementos, sendo que uma extremidade define a dependência de um tipo ao outro, porém, uma associação pode possuir dois elementos do mesmo tipo nas extremidades.

O conceito de *Component* aborda a área de desenvolvimento baseando-se em componentes, onde um componente é modelado ao longo do ciclo de vida de desenvolvimento e sucessivamente refinado na implantação e em tempo de execução. Um componente pode ser considerado como um sistema ou um subsistema.

O conceito de colaboração aplicado aos diagramas estruturais consiste na aplicação de *design patterns* para apresentar o comportamento coletivo dos elementos de um sistema.

A Fig. 2.5 apresenta um diagrama de exemplo com os principais elementos do Diagrama de Classes que serão utilizados ao longo do trabalho. Nele estão presentes desde elementos estruturais como classes e componentes, até elementos para refinamento de classificadores, como: *DataType*, *Enumeration* e *PrimitiveType*.

A Fig. 2.6 apresenta um diagrama de exemplo com os principais elementos do Diagrama de Estrutura Composta que serão utilizados ao longo do trabalho.

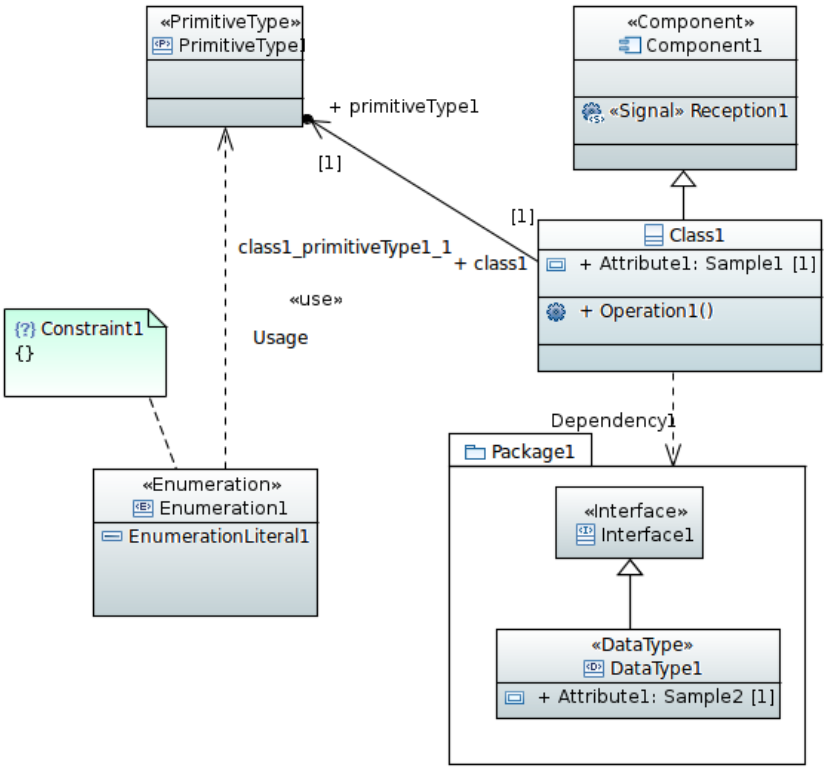

Figura 2.5: Exemplo de Diagrama de Classes

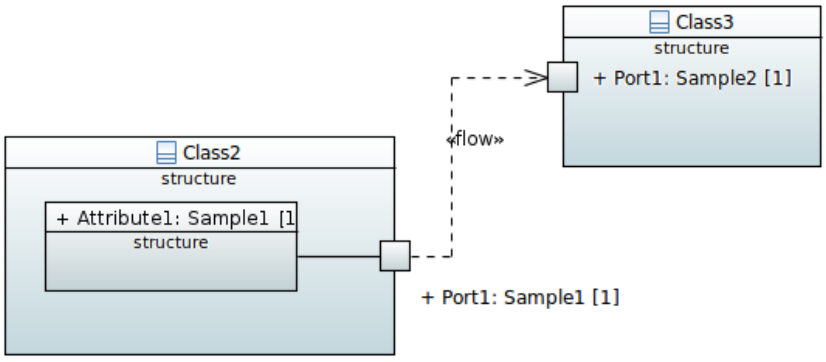

Figura 2.6: Exemplo de Diagrama de Estrutura Composta

2.4 Modeling and Analysis of Real Time and Embedded systems (MARTE)

O *profile* MARTE utiliza estereótipos para estender a linguagem UML e permite a modelagem de cenários em diferentes níveis de abstração. Entre eles, software, hardware, restrições de tempo, modelagem de recursos e alocação de software em hardware. Os estereótipos do *profile* podem ser aplicados no modelo criando anotações em classificadores, atribuindo valores nos atributos de classificadores ou adicionando restrições anotadas no modelo para atribuir valor nas propriedades.

Tanto a modelagem de software quanto hardware podem ser feitas de forma independente. Quando existe a necessidade de representar a execução de um software em um hardware, ambos modelados no *profile* MARTE, o modelo de alocação pode ser utilizado. Esta modelagem tem o objetivo de representar a junção de ambos os modelos.

A modelagem de tempo se tornou mais robusta no *profile* MARTE quando comparado com o profile SPT [Graf et al. 2006]. Com o *profile* MARTE é possível representar diretamente relacionamentos entre diferentes tipos de dados que possuem a mesma base de medida. Para completar os modelos, é possível anotá-los com expressões em uma linguagem de restrição. Esta linguagem é chamada de Value Specification Language (VSL).

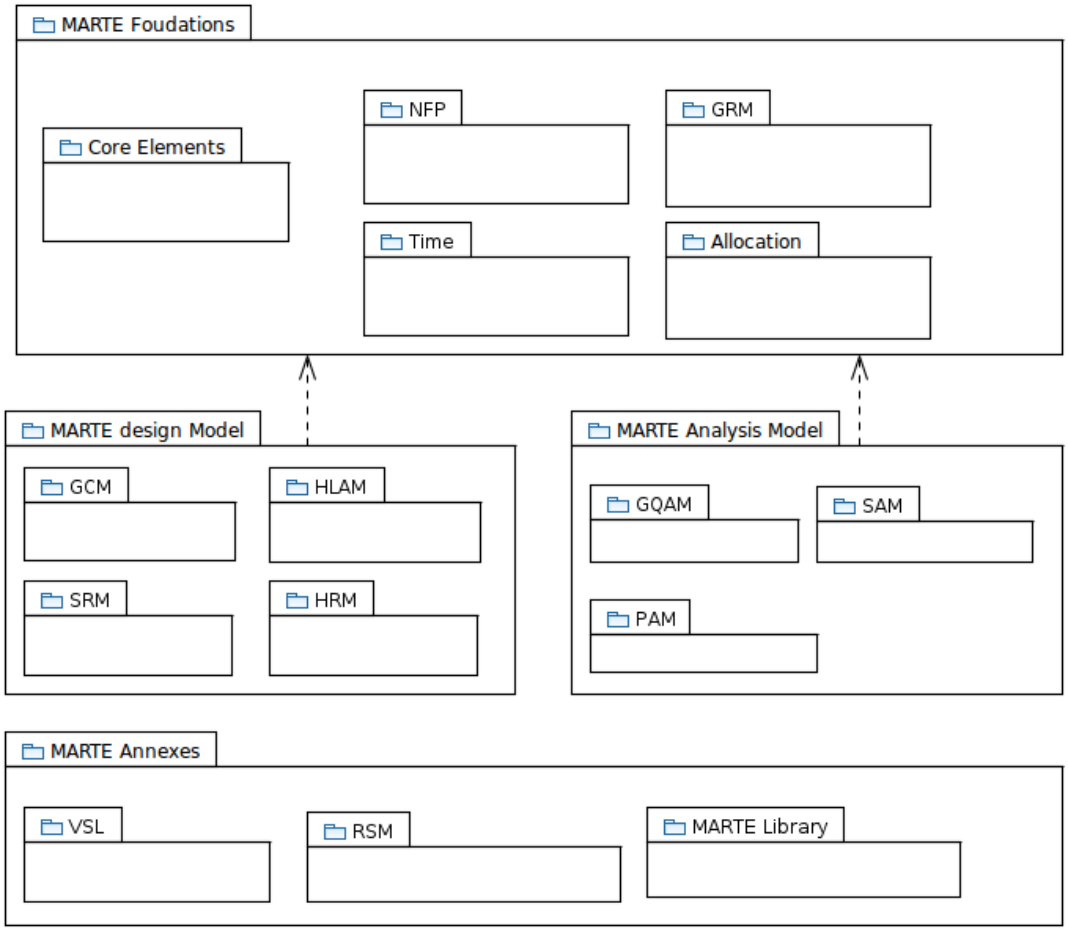

Figura 2.7: Estrutura do Profile MARTE [Graf et al. 2006]

A Fig. 2.7 apresenta a estrutura do *profile* MARTE com os pacotes que o constituem e as dependências entre eles. Considerando a grande quantidade de recursos no *profile* MARTE, o processo de especificação de um sistema não necessita utilizar todos os elementos, desta forma, a primeira preocupação na modelagem utilizando o mesmo consiste na escolha correta dos elementos que conseguem representar todas as funcionalidades desejadas do software, utilizando o menor número de elementos possível.

O *profile* é dividido em quatro pacotes: *Foundations*, *Design Model*, *Analysis Model* e *Annexes*. O pacote *Foundations* possui todos os elementos básicos do *profile*, e o mesmo é dividido em cinco sub-pacotes: *Core Elements*, *Non Functional Properties (NFP)*, *Time*, *General Resource Modeling (GRM)*, *Allocation*.

O pacote *Core Elements* consiste na base para todos os outros pacotes e outras definições do profile, nele estão representados os estereótipos que serão utilizados tanto para a criação de um modelo de análise quanto um modelo de *design*.

O pacote *Non Functional Properties (NFP)* define como são estruturadas as propriedades não funcionais. Em grande parte dos casos, este pacote é utilizado em conjunto com recursos mais específicos, como a linguagem VSL. Ele é utilizado para a definição da estrutura de adição de restrições não funcionais aos modelos. Nele existe a definição de tipos utilizados para estes tipos de restrição. Além disso, são definidos diversos estereótipos que adicionam semântica ao modelo.

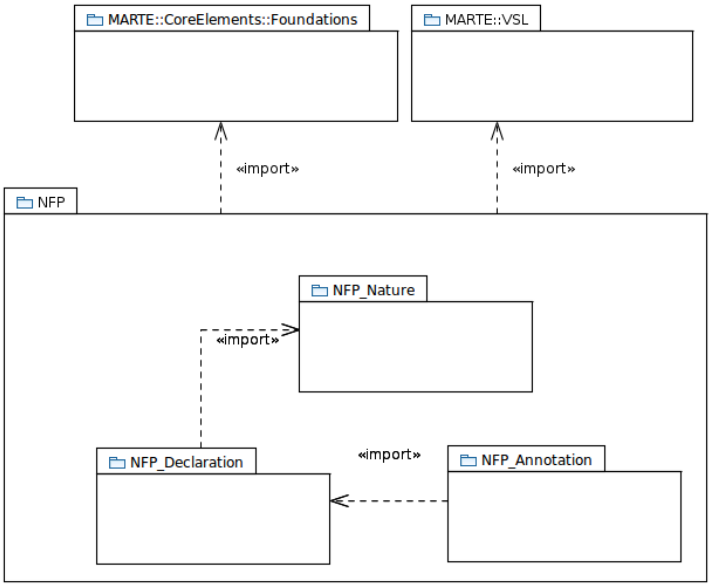

Figura 2.8: Estrutura do Pacote NFP [Graf et al. 2006]

A Fig. 2.8 apresenta a estrutura do pacote NFP, o mesmo é composto de três pacotes que dividem todos os elementos. O pacote *NFP Nature* representa todas as definições de medidas e unidades e relação entre bases de representação presentes no pacote NFP [Graf et al. 2006].

Uma das formas de adicionar semântica a um modelo é usando anotações, as propriedades não funcionais são criadas a partir de restrições anotadas em modelos. Para isto, no pacote *NFP Annotations* existe a definição de como as anotações devem ser feitas e integradas com a linguagem VSL [Graf et al. 2006].

As propriedades não funcionais de um sistema muitas vezes possuem tipos específicos, desta forma, o pacote *NFP Declaration* tem o objetivo de qualificar os tipos de dados que são utilizados pelo pacote NFP.

O pacote *Time*, apresentado na Fig. 2.7, possui as definições de modelagem de tempo do profile MARTE. Este pacote possui um framework para a modelagem apropriada de sistemas de tempo real e embarcados. Estes sistemas levam em consideração a cardinalidade de tempo (atraso, duração).

O objetivo do pacote *General Resource Modeling (GRM)* é oferecer conceitos que são necessários para modelar uma plataforma genérica para a execução de aplicações de tempo real e embarcados. Os elementos do pacote GRM são importantes para definições semânticas do papel que um elemento representa na arquitetura do sistema. A modelagem de recursos genérica inclui as características que lidam com a modelagem de plataformas de execução em diferentes níveis de detalhe, que podem ser definidos de acordo com o cenário descrito e a modelagem de hardware e software. Além disso, este pacote fornece construções de modelagem fundamentais que são posteriormente refinadas para apoiar o projeto (SRM e HRM), bem como a análise de modelos (GQAM, SAM e PAM) [Graf et al. 2006].

Os recursos que melhor representam o cenário de sistemas distribuídos de tempo real são:

- *ClockResource*, representa um hardware ou software que é capaz de seguir uma demanda de tempo prefixada. Os serviços e o mecanismo utilizados por um *ClockResource* podem ser refinados de acordo com o necessário;
- *CommunicationEndPoint*, atua como um terminal de conexão para um meio de comunicação, e isto é caracterizado pelo tamanho do pacote aceitado pelo endpoint. Um *CommunicationEndPoint* pode enviar ou receber dados, bem como um serviço de comunicação capaz de desencadear uma atividade na aplicação. O tamanho do pacote é definido por um atributo contido no elemento com o estereótipo;
- *CommunicationMedia*, representa o meio pelo qual os dados são transmitidos, em sistemas distribuídos também poderia ser chamado de canal de comunicação;
- *ComputingResource*, representa um dispositivo físico ou virtual capaz de armazenar e executar processamento;
- *DeviceResource*, representa um dispositivo externo que pode requerer serviços específicos da plataforma para uso ou gerenciamento. É suposto que o comportamento interno do dispositivo não é relevante para a parte do modelo considerada;
- *StorageResource*, representa um recurso capaz de armazenar memória, sua capacidade é expressada em número de elementos. O tamanho de um elemento deve ser representado em bits. O clock determina a velocidade de acesso a um elemento na memória.

Para o *profile* MARTE, o conceito de *Allocation* consiste em uma associação entre uma aplicação especificada utilizando o *profile* MARTE e uma plataforma de execução especifi-

cada utilizando o profile MARTE. Elementos da aplicação podem ser quaisquer elementos UML com aspectos estruturais e comportamentais adequados para a modelagem de uma aplicação. Uma plataforma de execução é representada como um conjunto de recursos ligados, onde cada recurso fornece serviços para apoiar a execução da aplicação. Assim, os recursos são, basicamente, elementos estruturais, enquanto os serviços são elementos em vez de comportamento. O pacote *Allocation* possui recursos para que seja possível realizar a ligação entre software e plataforma de execução.

O pacote *Design Model*, por sua vez, pode ser dividido em quatro pacotes que definem componentes adicionais, de software e hardware, para a modelagem de sistemas de tempo real e embarcados: Generic Component Model (GCM) , High-Level Application Modeling (HLAM), Software Resource Modeling (SRM) e Hardware Resource Modeling (HRM).

O pacote GCM é uma abstração da estrutura de classes da UML. Este pacote possui um denominador comum entre vários componentes de modelo. O objetivo deste pacote é fornecer ao profile MARTE um modelo mais geral possível, que não está vinculado à semântica de execução de domínios específicos, em que as características de tempo real não necessitam ser aplicadas.

O objetivo do pacote HLAM é fornecer conceitos de modelagem de alto nível para tratar modelagem de tempo real e recursos embarcados. Diferentemente dos domínios habituais de software, o desenvolvimento de sistemas de tempo real requer uma forma de modelagem com características quantitativas, e as mesmas estão relacionadas ao comportamento, comunicação e simultaneidade.

O pacote SRM permite realizar a descrição de uma forma unificada de software multitarefa para integrar explicitamente suporte a execução no fluxo do projeto. O pacote SRM não define um novo padrão de API multitarefa, ele apenas fornece artefatos de modelagem para descrever a API desejada.

O pacote HRM fornece mecanismos para modelar o hardware de sistemas embarcados.

Para a realização de análises quantitativas de software utiliza-se o modelo de análise do profile MARTE, que é divido em três subpacotes: Generic Quantitative Analysis Modeling (GQAM), Schedulability Analysis Modeling (SAM) e Performance Analysis Modeling (PAM).

O pacote GQAM inclui domínios especializados em que a análise é baseada no comportamento do software, tais como desempenho, escalonabilidade, energia, memória, confiabilidade, disponibilidade e segurança. Embora os domínios de análise constituem linhas semânticas diferentes, eles compartilham conceitos básicos.

O pacote SAM é destinado especificamente para a análise de escalonabilidade. Quando se lida com sistemas de tempo real, a influência da programação sobre o escalonamento e o desempenho é fundamental para calcular limites garantidos em tempos de resposta e cargas de processamento de recursos.

O pacote PAM inclui a análise de parâmetros de entrada, tais como análise de sensi-

bilidade, que explora um espaço de parâmetros para encontrar parâmetros operacionais ideais ou para identificar a carga de trabalho do sistema. A análise de sensibilidade pode também incluir cenários alternativos, plataformas, implementações físicas e configurações.

Para este trabalho, os pacotes mais relevantes são NFP, GRM e a linguagem Value Specification Language (VSL), que está presente nos Anexos, pois estes elementos serão utilizados, juntamente com a linguagem UML, para a modelagem de serviços de protocolo de comunicação.

Além de toda a estrutura apresentada anteriormente, o *profile* MARTE ainda possui anexos que são utilizados para definições complementares da linguagem, entre eles a linguagem VSL se destaca. A VSL é uma linguagem para a definição de restrições no modelo.

2.5 Value Specification Language (VSL)

A estrutura da linguagem VSL é dividida em cinco pacotes: *DataTypes*, *LiteralValues*, *Expressions*, *TimeExpressions* e *CompositeValues*.

Todos os tipos de dados definidos na linguagem UML e no profile MARTE podem ser acessados pela linguagem VSL usando a navegação de pacotes. O pacote *DataTypes* descreve os conceitos que definem as extensões de tipo de dados para UML, tipos de dados primitivos e enumeração, incluindo novas especializações para tipos de dados compostos e subtipos. Um *DataType* possui uma instância que determina seu valor. Um conjunto de instâncias de um determinado *DataType* consiste em um conjunto de valores distintos, caracterizados por propriedades e operações sobre os valores.

No pacote *LiteralValues* estão definidos valores constantes literais de diferentes tipos primitivos. Além dos literais UML, este pacote distingue, entre outros, literais reais.

As definições de expressão na linguagem VSL são encontradas no pacote *Expressions*, uma expressão consiste em um nó de uma árvore de expressões. O objetivo das expressões é derivar valores de outros valores ou expressões. Como VSL é uma linguagem tipada, as variáveis presentes nas expressões devem possuir um tipo definido, e as mesmas devem ser declaradas no contexto da expressão.

O pacote *TimeExpressions* possui recursos para tornar a especificação de tempo da linguagem UML mais expressiva.

A linguagem VSL possui a definição abstrata de tipos de dados que são derivados de outros tipos, tuplas, coleções, escolhas e intervalos. O pacote *CompositeValues* possui as definições de formação dos tipos compostos.

A Fig. 2.9 mostra um exemplo de uso dos pacotes NFP, GRM e linguagem VSL. O *Controller* é um classificador e C1 uma instância deste classificador. Na Fig. 2.9 existem duas expressões na linguagem VSL. A utilização da linguagem VSL é feita usando anotações nos modelos, desta forma, é possível inserir uma semântica sem ambiguidade

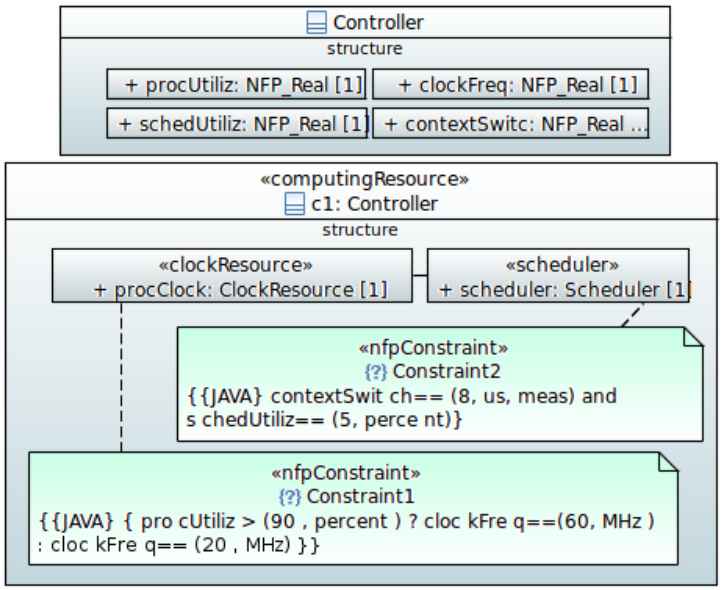

Figura 2.9: Exemplo de utilização profile MARTE [Graf et al. 2006]

e validável aos comentários. Com a linguagem VSL é possível criar validações temporais, processamento, consumo de energia, além de ser possível definir funções estatísticas como *jitter*, que é uma variação estatística de atraso, ou seja, pode ser definida como a variação do atraso entre acontecimentos sucessivos.

Na Fig. 2.9 é apresentado um exemplo da linguagem em que são utilizados operadores lógicos, operadores condicionais, atribuição de valores e uso de diferentes tipos de dados, como: *percent* e *MHz*. Na linguagem VSL a definição de um tipo pode ser construída de forma derivada. Desta forma, um tipo de dado que é derivado a partir de outro possui uma função de transformação, a mesma será utilizada em caso de comparação de valores em tipos diferentes.

Capítulo 3

Entity Title Architecture (ETArch)

Neste trabalho é utilizado como estudo de caso a modelagem de serviços da arquitetura de Internet ETarch. A mesma é descrita neste capítulo, em que seus principais componentes e serviços são definidos.

O modelo de referência para sistemas de comunicação, *International Organization for Standardization* (ISO)/ *Open Systems Interconnection* (OSI), possui diversos problemas [Day e Zimmermann 1983] [Park e Shiratori 1994]. Estes problemas estão relacionados à camada de rede, principalmente nos protocolos *Transmission Control Protocol* (TCP)/ *Internet Protocol* (IP). O modelo consiste basicamente em cinco camadas: Aplicação, Transporte, Rede, Enlace e Física. Cada camada é responsável por assegurar uma parte do serviço, mas não é possível garantir o serviço da camada anterior. Grande parte dos serviços são feitos na camada de aplicação, mas alguns poderiam ser feitos em camadas sobre o núcleo da rede, como as camadas de transporte e rede [Santos et al. 2011].

O presente capítulo é dividido em duas seções. A Seção 3.1 apresenta uma especificação da arquitetura ETArch, mostrando os principais elementos e seus respectivos comportamentos. A Seção 3.2 apresenta a modelagem dos serviços representados pelas primitivas dos protocolos da ETArch.

3.1 Especificação da Arquitetura ETArch

A *Entity Title Architecture* (ETArch) [de Oliveira Silva et al. 2012] é uma arquitetura de Internet que utiliza o conceito de *clean slate*, onde os esquemas de nomeação e endereçamento são baseados em uma designação independente de topologia que identifica uma entidade, chamada de *Title*, e na definição de um canal que reúne várias entidades de comunicação, chamado de *Workspace*.

O componente principal desta arquitetura é o *Domain Title Service* (DTS), que lida com todos os aspectos de controle da rede. O DTS é composto por *Domain Title Service Agents* (DTSAs), que mantêm informações sobre entidades registradas no domínio e *workspaces* que estão inscritos no mesmo, tendo com objetivo configurar os dispositivos

da rede para implementar os *workspaces* e permitir que os dados possam chegar a cada entidade inscrita no *workspace*.

Usando a ETArch, comunicações são manuseadas pelo *workspace*. Portanto, ETArch inerentemente permite o suporte integrado a multicast e mobilidade dentro do *workspace*, que pode ser visto como um barramento lógico interligando várias instâncias da entidade (por exemplo, um serviço, um sensor, um smartphone, um *host* ou mesmo um processo). Seu comportamento é inspirado na tecnologia multicast, onde os dados são enviados uma vez por uma fonte ao *workspace* e todas as entidades inscritas receberão os mesmos.

A operação da ETArch em que uma entidade centralizada é responsável pelo comportamento do plano de envio, consiste no conceito de *Software-Defined Networking* (SDN) [Open Networking Foundation 2012], implementado na ETArch pelo padrão *OpenFlow* [McKeown et al. 2008], que é uma versão de SDN disponível em produtos comerciais utilizados em vários projetos de pesquisa. Ele separa o plano de dados do plano de controle de rede permitindo uma entidade separada, controlador OpenFlow, gerenciar em baixo nível o plano de dados, configurando a tabela de endereçamento dos *switches* usando uma aplicação orientada a serviços. Isto habilita a reconfiguração dos *switches* durante a execução, permitindo um controle dinâmico da rede [Kim e Feamster 2013] e permitindo criar o conceito de comunicação orientada a *workspace*.

Considerando o modelo de Internet ETArch, a rede é composta por vários DTSAs que são configurados utilizando um modelo em árvore. Quando um *workspace* é requisitado por uma entidade e ele não existe no respectivo DTS, o mesmo pede a informação para o DTS de nível superior, e assim por diante, como acontece na estrutura DNS utilizada atualmente [de Oliveira Silva et al. 2011].

Para suportar os conceitos apresentados, a ETArch define protocolos no plano de dados e de controle. No plano de controle, a abordagem de sinalização fornece os serviços relacionados com o ciclo de vida das entidades e *workspaces*, tais como registrar uma entidade no *Domain Title Service* (DTS) ou criar um *workspace*, registrar e retirar entidades de um determinado *workspace*.

O *Entity Title Control Protocol* (ETCP) é responsável pela comunicação entre a entidade e o DTSA, enquanto *DTS Control Protocol* (DTSCP) é responsável pela comunicação entre DTSAs no DTS. Neste trabalho, os serviços dos protocolos ETCP e DTSCP são modelados como estudo de casos.

3.1.1 Principais Elementos da ETArch

A Fig. 3.1 apresenta um exemplo de um cenário da arquitetura ETArch com a presença dos elementos em alto nível. A linha tracejada indica um *workspace*, onde estão conectados o *User1* e *User2*.

Os *workspaces* podem ser de dois tipos: Workspaces de Controle, que tem como ob-

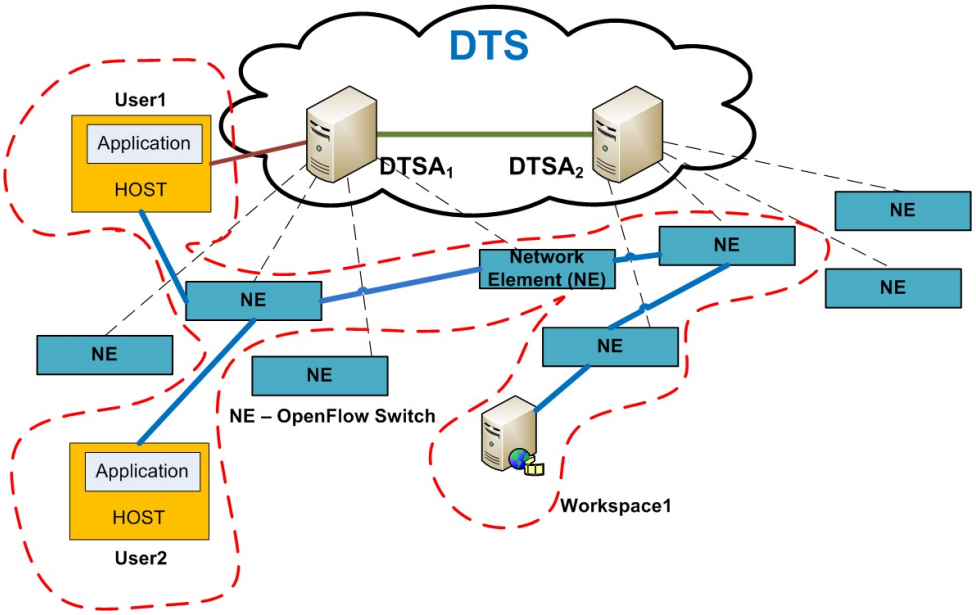

Figura 3.1: Exemplo Arquitetura ETArch [de Oliveira Silva 2013]

jetivo gerenciar, manter e controlar o ambiente distribuído do DTS, bem como gerenciar o ciclo de vida de entidades; e os Workspaces de Dados, que tem o objetivo de suportar a troca de primitivas de envio de dados entre Aplicações (Entidades) de propósito geral, tais como, Video Conference, HDTV, entre outras.

O Domain Title Service (DTS) é responsável pelas operações relacionadas à resolução de Títulos, gerenciamento do ciclo de vida das Entidades em ambiente distribuído e manutenção da relação entre Entidades e Títulos. Além disso, a base de dados das entidades, títulos e *workspaces* está presente no DTS. O mesmo é composto de Domain Title Service Agents (DTSA). O DTSA é responsável por implementar todas as operações de gerenciamento de entidades do domínio em que ele está presente.

Os DTSAs são conectados pelo Network Element (NE), como *switches*, que podem ser exclusivos para o Plano de Controle ou compartilhados com o Plano de Dados.

Um conjunto de DTSAs pode ser especificado para representar um domínio, como uma região geográfica. Para isto, é necessário definir um DTSA com responsabilidades adicionais, chamado de Master DTSA (MDTSA). O MDTSA é a interface de contato entre dois ou mais domínios, cujos *Workspaces de Controle* podem ser públicos ou privados. Se um domínio demandar apenas um DTSA, então este fará também o papel de MDTSA. O MDTSA possui informações sobre o grafo de interconexão de DTSAs do domínio [de Oliveira Silva 2013].

3.1.2 Principais primitivas ETCP

A comunicação no ETCP é feita usando primitivas, cada uma delas designa um comportamento ao DTSA. Existem seis primitivas principais [de Oliveira Silva 2013]:

- ENTITY_REGISTER: Registra uma entidade em um DTS. Para ser registrada, uma entidade deve apresentar seu título, capacidades e necessidades de comunicação. Para comunicar-se a entidade deve primeiro registrar-se;
- WORKSPACE_CREATE: Cria um *workspace* localmente no DTSA. Se o *workspace* tem um acesso público após a criação bem sucedida, o DTSA vai anunciar o mesmo usando a inserção de uma entrada do *workspace* no banco de dados;
- WORKSPACE_ATTACH: Anexa uma entidade em um *workspace*. Para realizar este processo, o DTSA obterá todos os elementos de rede e irá configurá-los para estender o *workspace*. Se o DTSA tem as informações sobre o *workspace* utilizando o protocolo DTSCP, ele irá enviar uma primitiva WORKSPACE_LOOKUP;
- ENTITY_UNREGISTER: Remove uma entidade de um DTS;
- WORKSPACE_DETACH: Remove uma entidade de um determinado *workspace*;
- WORKSPACE_DELETE: Apaga um *workspace* do DTSA e executa operações para limpar os dados referentes a destruição.

3.1.3 Principais primitivas DTSCP

As primitivas do DTSCP tem como objetivo definir a comunicação interna em um DTS. Existem três primitivas principais [de Oliveira Silva 2013]:

- WORKSPACE_LOOKUP: Enviado por um DTSA para outros DTSAs com o objetivo de encontrar um *workspace*. Esta primitiva percorre a estrutura de árvore até que o *workspace* seja encontrado ou atinja um limite de tempo;
- WORKSPACE_ADVERTISE: Insere, apaga ou atualiza o banco de dados que armazena as informações de *workspace*. A operação recebe o nível indicando a visibilidade do *workspace*. O DTSA armazenado no banco de dados deve ser do mesmo nível ou um DTSA *Master* do nível logo abaixo;
- DTS_MESSAGE: Habilita a comunicação entre diferentes DTSAs dentro de um DTS. Se o DTSA de origem conhece o caminho até o DTSA de destino, o caminho deverá ser passado pela mensagem. Caso contrário, a mensagem será passada pelos elementos de rede até algum elemento que possua o caminho de destino seja encontrado. Se o DTSA *Master* de um nível não conseguir resolver o caminho, a mensagem irá falhar.

3.2 Modelagem Utilizando Autômatos

A definição dos serviços na forma visual foi feita utilizando Autômatos Finitos. Os serviços são definidos sempre em duas visões, uma representando o processo executado

para a entidade que envia a requisição e outra representando a entidade responsável por tratar a requisição.

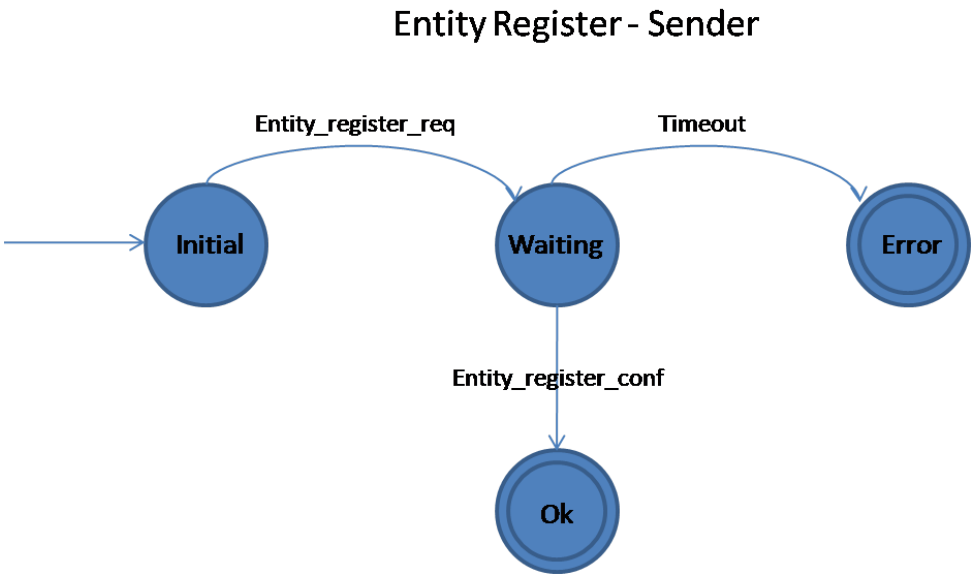

Figura 3.2: Entity Register - Sender

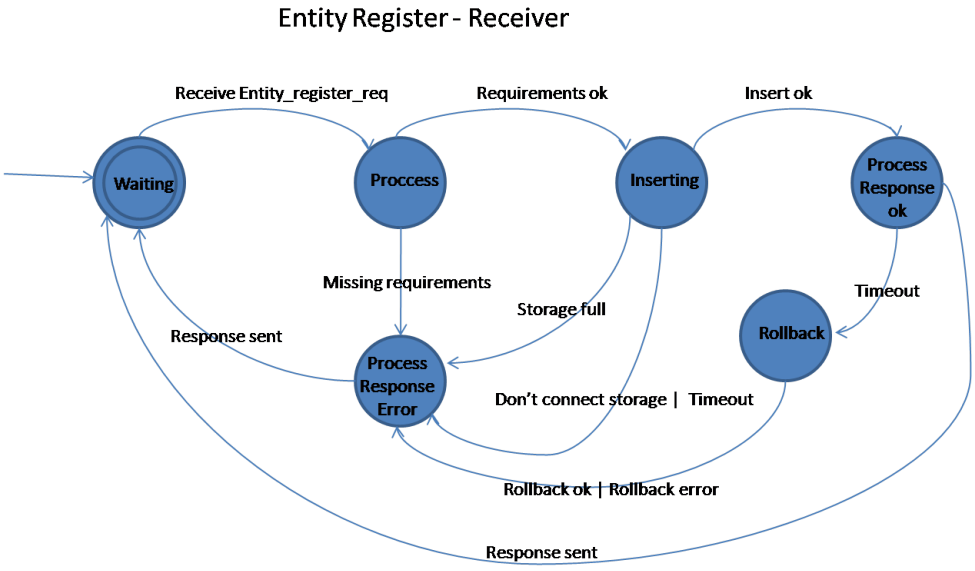

Figura 3.3: Entity Register - Receiver

As Figs. 3.2 e 3.3 apresentam os autômatos do serviço de registro de entidades. A entidade que faz a requisição, apesar de possuir um processo interno, necessita apenas aguardar a resposta da entidade que recebeu a requisição. Neste caso, o DTSA receberá a requisição e deverá executar serviços de inserção do título da entidade na base de dados. Caso ocorra um erro, é necessário realizar operações para desfazer as alterações realizadas pela operação.

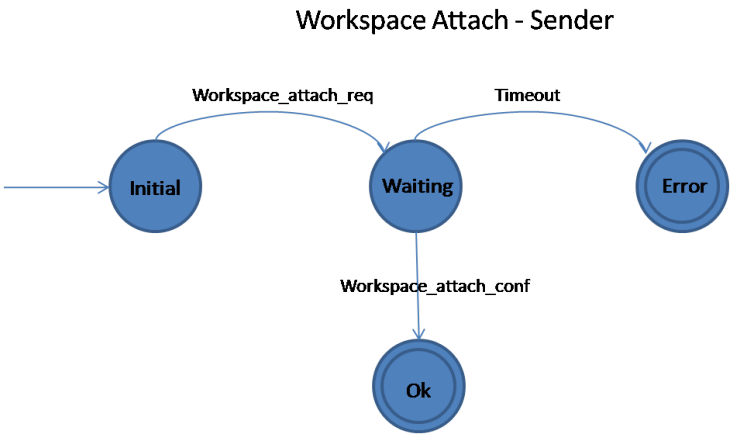

Figura 3.4: Workspace Attach - Sender

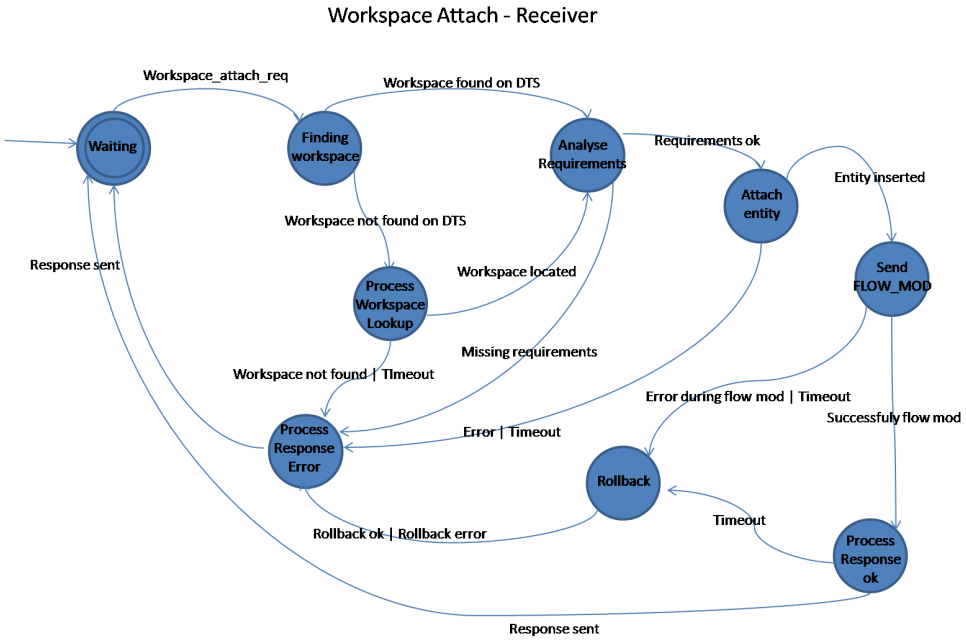

Figura 3.5: Workspace Attach - Receiver

As Figs. 3.4 e 3.5 apresentam o serviço de registro de uma entidade a um *workspace*. Esta operação é executada em várias etapas, em cada uma delas é necessário garantir que caso ocorra algum erro, uma operação de restauração será executada. Para que a entidade consiga executar esta operação é necessário que a mesma possua as capacidades e requisitos definidos no *workspace*, caso contrário, uma mensagem de erro será enviada para a entidade que realizou a requisição.

A distribuição de informações na arquitetura ETArch está diretamente relacionada com a criação de um *workspace*, como é mostrado nas Figs. 3.6 e 3.7. Quando uma entidade deseja transmitir dados, como vídeo e voz, e um conjunto de entidades deseja receber tais dados, um *workspace* é criado e a informação das entidades trafegará entre ele. Para esta operação, caso algum erro ocorra em qualquer etapa, é necessário executar

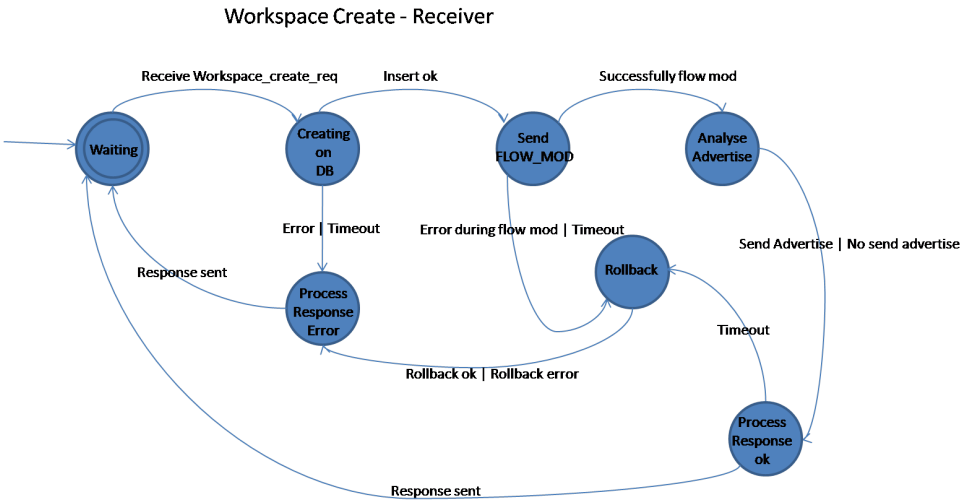

Figura 3.6: Workspace Create - Receiver

a restauração de todas as operações realizadas anteriormente.

Ao criar um *workspace*, a entidade poderá transmitir informações para todos os elementos do mesmo. É necessário que uma entidade entre em um *workspace* para receber informações do mesmo, desta forma, para encontrar o *workspace* é necessário realizar uma busca, que poderá ser executada em vários níveis de DTSAs. As Figs. 3.8 e 3.9 mostram o processo da busca por um *workspace*. É importante ressaltar a operação de laço que existe caso o nível atual não seja encontrado, pois é necessário que o título do *workspace* seja procurado em DTSAs de níveis superiores.

A representação dos serviços usando autômatos consegue definir todo o fluxo das operações, desta forma, é possível utilizar algoritmos sobre autômatos para garantir propriedades, simplificar estados, entre outros. Como é apresentado na Fig. 3.8 uma operação que ocorre em forma de um laço pode executar quantas vezes necessárias até que aconteça um *timeout*. Mas, a operação de *timeout* não é representada de forma significativa

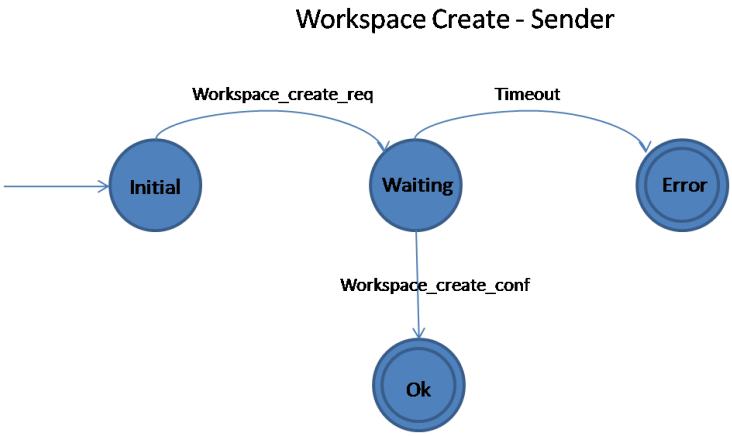

Figura 3.7: Workspace Create - Sender

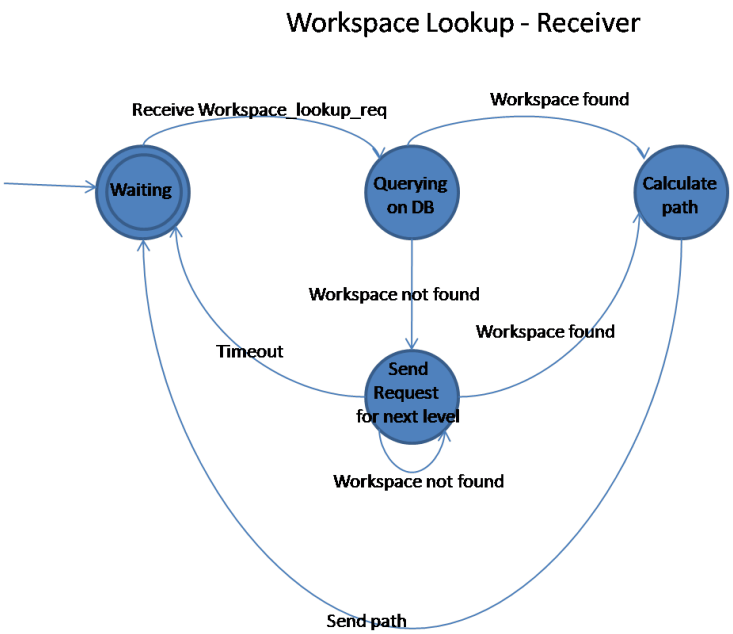

Figura 3.8: Workspace Lookup - Receiver

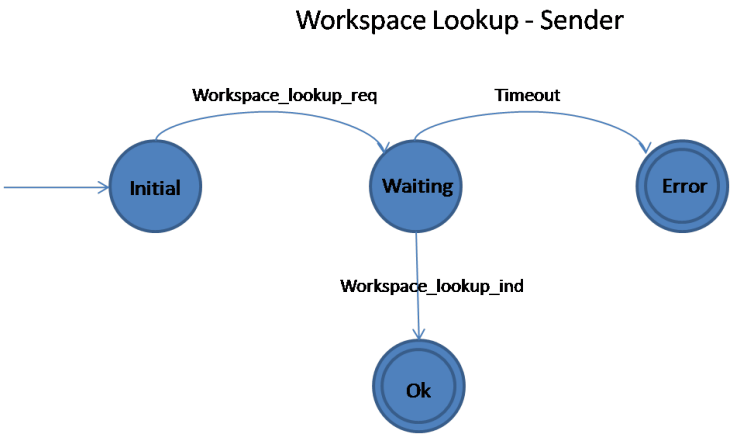

Figura 3.9: Workspace Lookup - Sender

nesta abordagem de modelagem, assim como a operação de busca das informações em um DTSA de nível superior. Portanto, existe a necessidade de um modelo com maior capacidade de representação.

Capítulo 4

Modelagem de protocolos *Entity Title Architecture* (ETArch) baseada em UML

A partir do estudo dos elementos e do comportamento da arquitetura de Internet ETArch é possível definir os pontos que necessitam de modelagem. A escolha dos diagramas foi feita utilizando uma análise humana das necessidades de modelagem dos protocolos ETCP e DTSCP, que consistem em permitir a modelagem estrutural e comportamental, juntamente com o papel dos diagramas definidos na especificação da UML.

No presente capítulo é apresentada a abordagem para a modelagem dos elementos da arquitetura ETArch. O capítulo é dividido em três seções. A Seção 4.1 apresenta o objetivo traçado para a modelagem dos serviços. A Seção 4.2 apresenta os modelos criados para representar os serviços dos protocolos ETCP e DTSCP. Ao final, na Seção 4.3 são apresentados os resultados, destacando as vantagens e limitações da abordagem.

4.1 Motivação

Para que uma especificação consiga representar os pontos mais relevantes da arquitetura, é necessário que o modelo possua mais de um nível de abstração [Selic 2003]. Para representar atributos, operações e relacionamentos entre elementos, o modelo deve se aproximar do nível de implementação. A segunda necessidade está relacionada a modelagem da estrutura dos elementos de uma forma abstrata, de modo que seja possível decompor os mesmos em módulos, sendo necessária a definição de um fluxo de dados entre cada módulo. É necessário modelar as operações entre os elementos, que devem seguir as definições dos modelos com maior nível de abstração, como o fluxo de dados.

Como a ETArch deve ser capaz de lidar principalmente com transmissão de voz e vídeo, as propriedades de tempo e recursos devem ser garantidas durante uma operação.

Para sanar as questões apresentadas, uma abordagem de uso da linguagem UML para a modelagem dos serviços dos protocolos de comunicação da arquitetura ETArch, com os respectivos elementos envolvidos, foi proposta.

A linguagem de modelagem escolhida para este trabalho é a UML, usando os Diagramas Estruturais e Comportamentais.

4.2 Modelagem usando UML

A modelagem de protocolos da arquitetura ETArch utilizando a linguagem UML visa apresentar a estrutura dos elementos, comportamentos e restrições de tempo em um alto nível de abstração. Para isto, os elementos são modelados utilizando os diagramas de Classes, Sequência e Estrutura Composta.

O Diagrama de Classes é responsável por definir os classificadores. Com este diagrama, é possível definir atributos, métodos, regras de visibilidade dos elementos internos e relacionamento entre classificadores. O Diagrama de Classes é importante na modelagem de protocolos para definir os tipos utilizados, as estruturas de dados, definir os elementos e como eles estão relacionados entre si. Este recurso tem uma grande contribuição na modelagem do comportamento do protocolo.

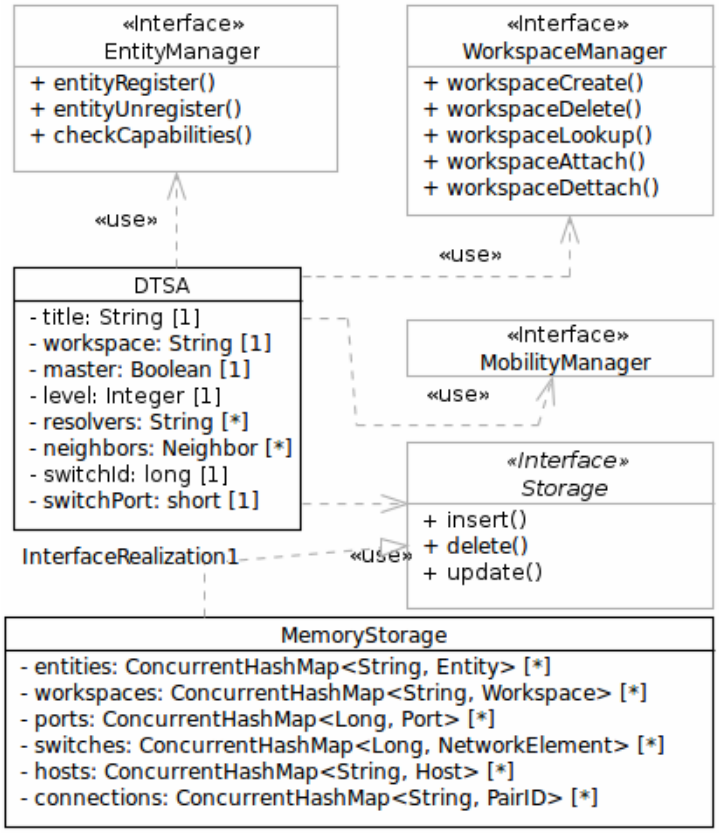

Figura 4.1: Diagrama de Classes - Principais Elementos da ETArch

A Fig. 4.1 apresenta as classes mais importantes da arquitetura ETArch. As classes DTSA e MemoryStorage mostram a representação de atributos e as outras classes do diagrama são definições simples de elementos utilizados. Os elementos mostrados na Fig. 4.1 estão relacionados com os serviços dos protocolos de comunicação da ETArch. Esta definição tem como objetivo mostrar o nível de abstração representado pelo Diagrama de Classes, que é a representação de atributos e definição de mensagens, não incidindo sobre a estrutura interna do elemento, o que seria modelado usando o Diagrama de máquina de estados.

O Diagrama de Estrutura Composta permite definir uma visão detalhada da estrutura de um classificador, o relacionamento entre atributos, interfaces de entrada e saída e fluxo de dados. Na arquitetura ETarch, o DTSA é um dos elementos relevantes, desta forma, ele foi definido utilizando o Diagrama de Estrutura Composta. Como uma entidade pode ser qualquer tipo de dispositivo e o comportamento interno da mesma não é relevante para a arquitetura, então, a estrutura interna do mesmo não foi modelada.

A Fig. 4.2 apresenta o Diagrama de Estrutura Composta do elemento DTSA, que consiste no elemento central da ETArch. O modelo representado na Fig. 4.2 apresenta todo o fluxo de dados do DTSA, desde a entrada das requisições pelos canais de comunicação, *wireless* e *ethernet*, até a resposta.

Para melhor visualização, a definição do DTSA foi dividida em três figuras, para isto, o DTSA foi dividido em dois módulos, o *ResourceAdapters* que consiste no controle de fluxo, que são definidos como bases de comunicação dos protocolos e *Building Block* que consiste em um elemento responsável pelo controle e armazenamento de dados do protocolo.

Na Fig. 4.3 é apresentada a estrutura do elemento que representa a padronização de mensagens dos protocolos ETArch. Na arquitetura, este protocolo será usado para enviar mensagens até que um elemento responsável trate e transforme a mensagem no formato do protocolo ETCP ou DTSCP. Neste caso, o elemento responsável é chamado *NE conector*.

Na ETArch são utilizados dois protocolos que padronizam a comunicação: OpenFlow [Sonkoly et al. 2012] e Media Independent Handover (MIH) [Griffith et al. 2010]. Requisições que são de origem *Ethernet* são padronizadas usando o protocolo OpenFlow e as requisições originadas de um canal *Wireless* são padronizadas pelo MIH.

Como não existe diferença no fluxo de entrada e saída, a distinção foi feita no modelo de fluxo de dados dos elementos para melhor entendimento da semântica. A implementação de elementos deve representar o conceito apresentado na estrutura modelada. Tal como o elemento mostrado na Fig. 4.3 que é responsável por intermediar mensagens na solicitação e resposta, todas as requisições destinadas ao DTSA devem passar por este módulo.

O módulo *Building Blocks* é descrito na Fig. 4.4. Há quatro elementos internos neste módulo. O *WorkspaceManager* é responsável por todas as operações relacionadas ao *workspace*, como criar, anexar, desanexar e excluir elementos. O *EntityManager* trata os requisitos da entidade, como registrar e cancelar o registro. O *MobilityManager* é

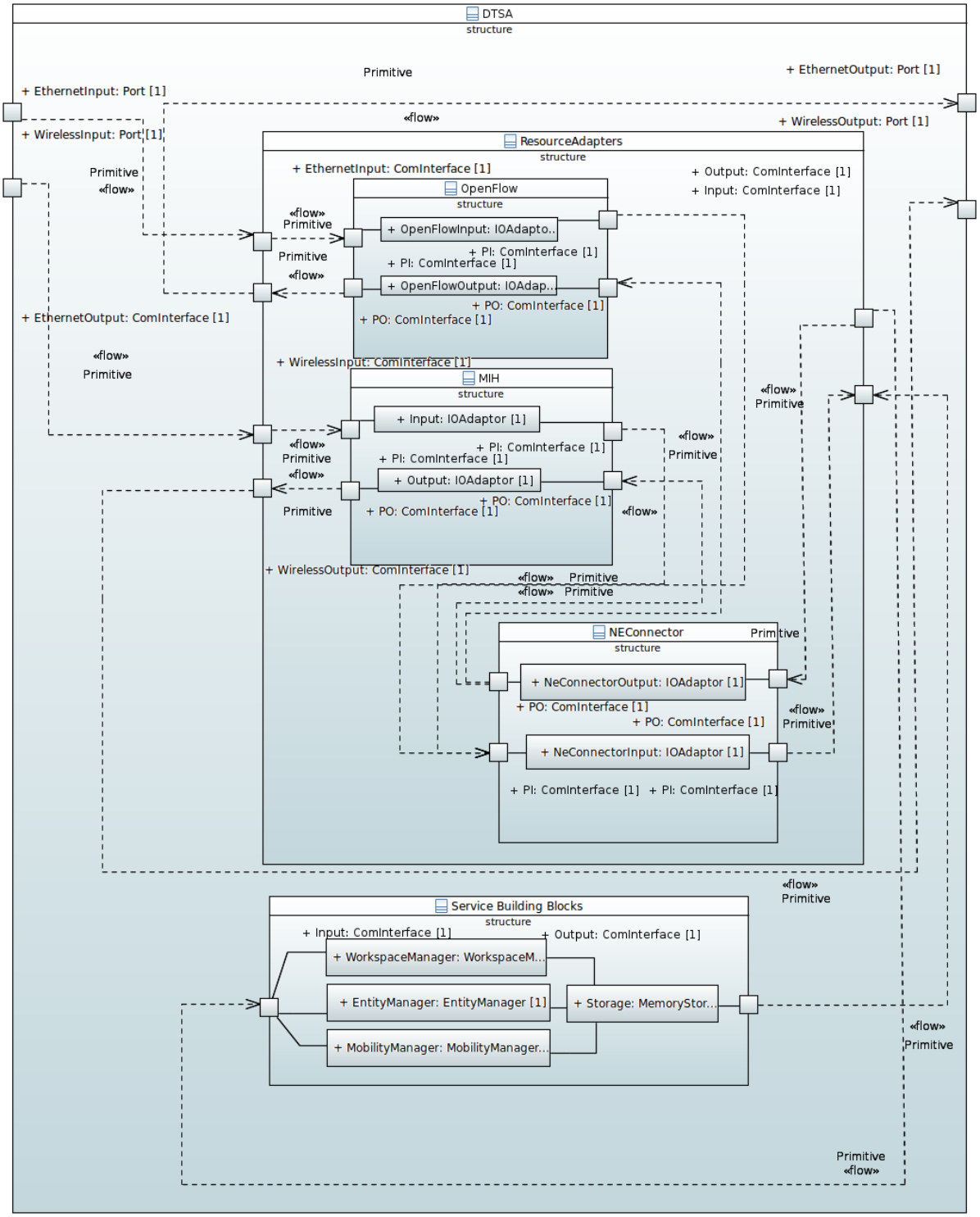

Figura 4.2: Diagrama de Estrutura Composta do DTSA

responsável pelas operações de mobilidade, como operações de *handover*. O *Storage* é uma estrutura genérica de armazenamento para representar uma base de dados, e todas as operações do módulo *Building Blocks* necessitam alterar a base de dados.

A Fig. 4.5 representa o relacionamento em alto nível de abstração entre os modelos apresentados nas Figs. 4.3 e 4.4. A requisição de uma entidade deve seguir o compor-

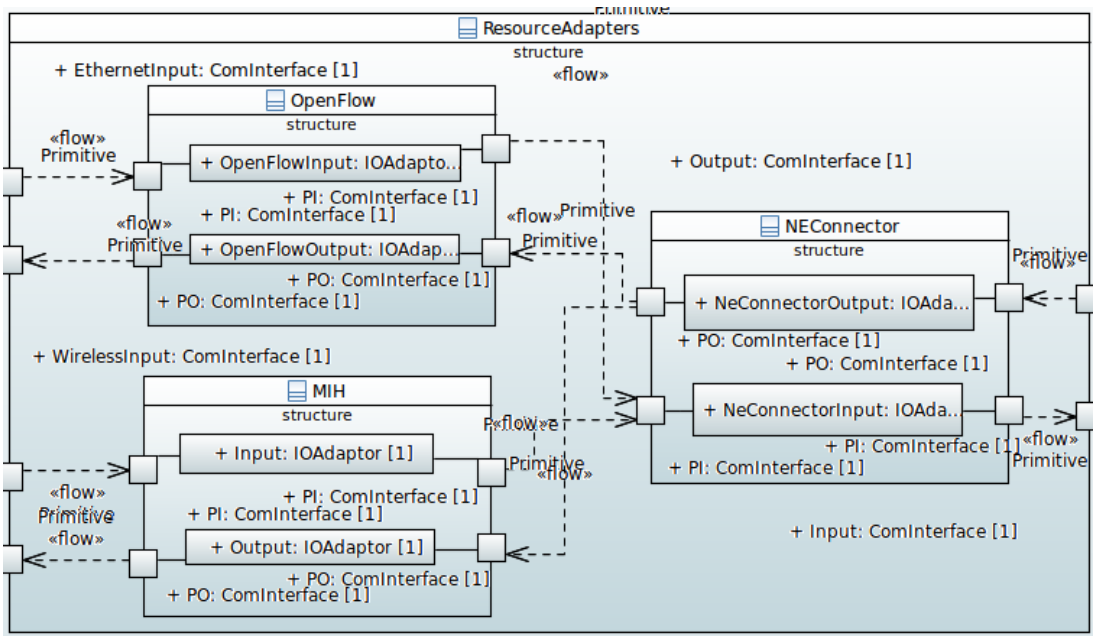

Figura 4.3: Diagrama de Estrutura Composta - *ResourceAdapters*

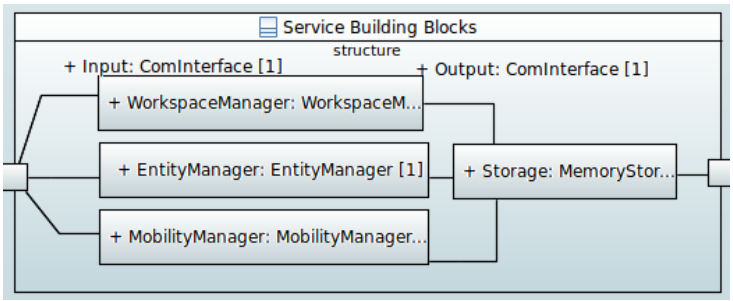

Figura 4.4: Diagrama de Estrutura Composta - *BuildingBlocks*

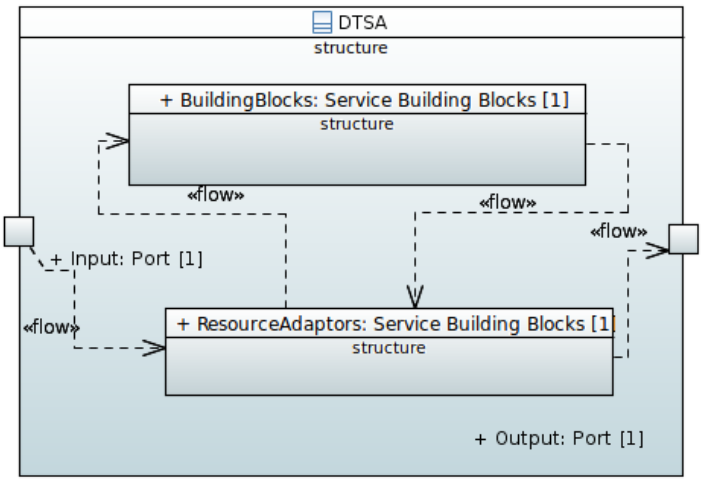

Figura 4.5: Diagrama de Estrutura Composta - Estrutura do DTSA

tamento definido na estrutura do DTSA. Em cada módulo existem diversos comportamentos. Os comportamentos mais representativos neste trabalho são: *Entity Register*, *Workspace Create*, *Workspace Attach* e *Workspace Lookup*.

A definição do comportamento dos serviços relevantes é realizada utilizando Diagramas de Sequência. Para melhorar a visualização dos parâmetros adicionados na mensagem, o nome dos mesmos foi adicionado no identificador das mensagens.

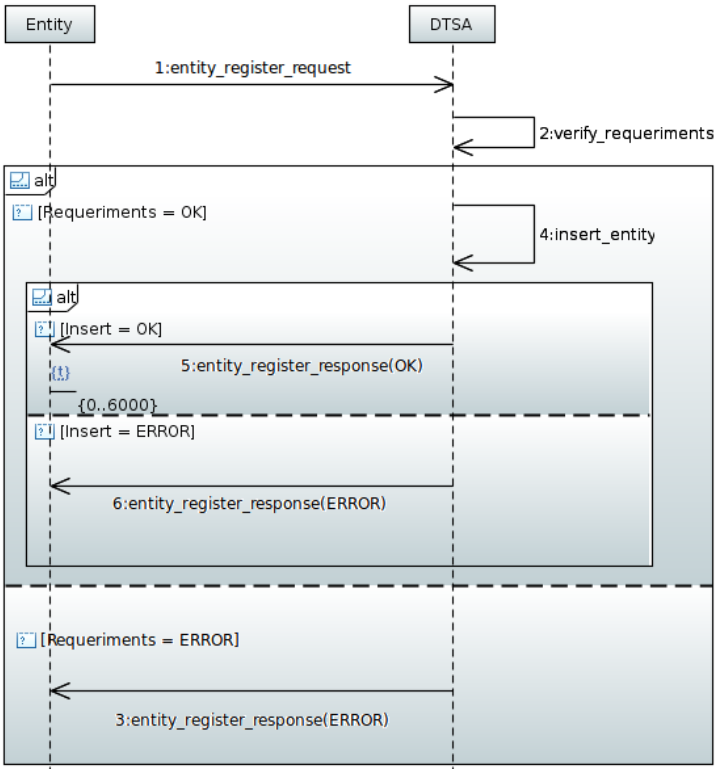

Figura 4.6: Diagrama de Sequência - *Entity Register*

A Fig. 4.6 mostra o Diagrama de Sequência de um serviço de registro de uma entidade em um DTSA. Neste diagrama, o canal de comunicação é abstraído de tal maneira que, quando uma chamada tiver como destino um DTSA, a mensagem do dispositivo deverá passar através da estrutura de fluxo descrita nos *Resource Adaptors*. Os principais recursos utilizados neste diagrama são as restrições de tempo e os fragmentos combinados do tipo *alt*. Os predicados entre colchetes significam o resultado da operação chamada.

A Fig. 4.7 representa o fluxo de mensagens da criação de um *workspace*. Para executar esta operação, tanto os elementos dos módulos *Resource Adaptors* quanto *Building Blocks* são utilizados. Quando um *workspace* é criado, as informações do mesmo devem ser salvas no *Storage* para que ele possa ser utilizado por outros elementos da rede.

As Figs. 4.8 e 4.9 estão relacionadas. A operação *Workspace Lookup* é um sub-processo da operação *Workspace Attach*. A referência entre os diagramas não é apresentada nas figuras para que a compreensão seja mais simples, mas a mesma é representada em nível de ferramentas. Nesta operação, as restrições de tempo da operação *Workspace Lookup*

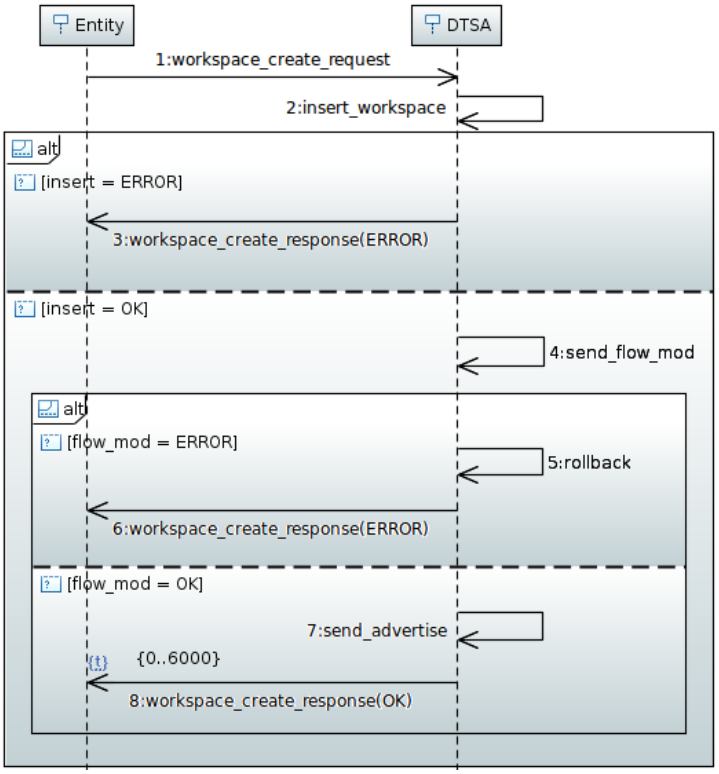

Figura 4.7: Diagrama de Sequência - *Workspace Create*

devem ser consideradas no *Workspace Attach*.

Um DTSA deve estar sob a área de influência de um ou mais MDTSAs. Para isto, é necessário que o DTSA registre-se junto ao MDTSA. Um DTSA deve registrar-se assim que é iniciado ou caso ocorra alguma alteração em seu arquivo de configuração. A Fig. 4.10 apresenta o processo realizado por um DTSA para se registrar.

O serviço de cancelamento de registro de uma entidade deve ser feito pelo DTSA. Antes de efetuar o cancelamento, a entidade deve ser removida de todos os Workspaces que está anexada, desta forma, a operação *Workspace Detach* será executada. Além disso, todas as outras entidades que estão a ela ligadas também devem ter o seu registro cancelado. A Fig. 4.11 apresenta um simples exemplo do cancelamento de registro, sem estender a quantidade de níveis de uma rede.

Em certas situações é necessário que um DTSA comunique-se com outro DTSA. Estas interações necessitam de troca de informações entre DTSAs. Para isto, o serviço *DTS Message* é utilizado. A Fig. 4.12 mostra o Diagrama de Sequência desta operação. Para que a mesma ocorra, é necessário que exista um elemento de origem e um elemento de destino para a mensagem.

Em sua criação, um *Workspace* está presente somente no contexto do DTSA que o contém. Para que seja visível e acessível por outros DTSAs é necessário publicá-lo. A publicação consiste em atualizar o *Workspace Database* de forma que a informação seja mantida pelo MDTSA. Para isto, a Fig. 4.13 descreve os passos para que o evento seja

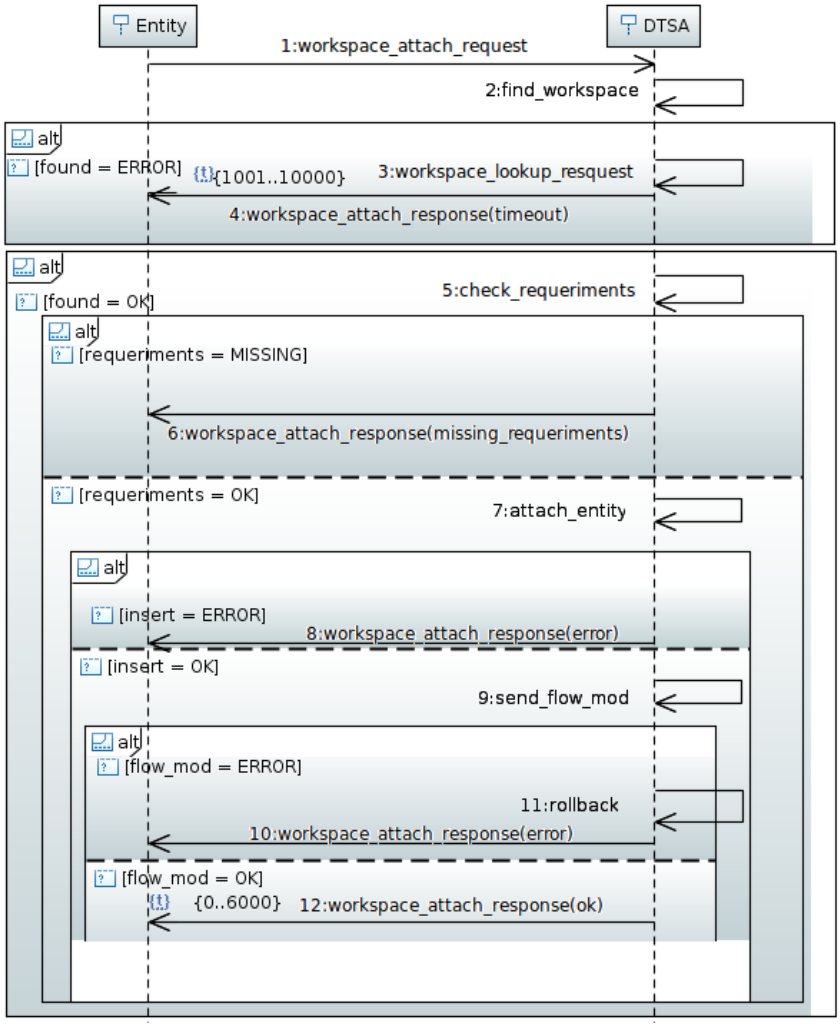

Figura 4.8: Diagrama de Sequência - *Workspace Attach*

publicado usando um Diagrama de Sequência.

Uma entidade pode tanto criar um *Workspace* quanto destruí-lo. A Fig. 4.14 apresenta o processo de destruição de um *Workspace*. Neste processo, o *Workspace Manager* removerá todas as entidades que estão anexadas a este *Workspace* e, além disso, retornará a lista de elementos de rede que serão afetados por esta operação no âmbito do DTSA onde, inicialmente, o *Workspace* foi criado.

Uma entidade pode se registrar em um *Workspace* e pode se desligar do mesmo. Caso uma entidade deseje desligar-se de um *Workspace*, a mesma deve solicitar este serviço ao DTS. Após esta operação, é necessário atualizar a lista de entidades que estão conectadas no *Workspace*, como é mostrado na Fig. 4.15.

Ao propor uma abordagem de modelagem de protocolos de comunicação, é intuitivo que seja tomado como base abordagens semelhantes a outras já existentes na literatura, como máquinas de estados. Quando existe uma mudança na linguagem de modelagem, ainda é possível buscar esta semelhança, como o Diagrama de Máquina de Estados da linguagem UML.

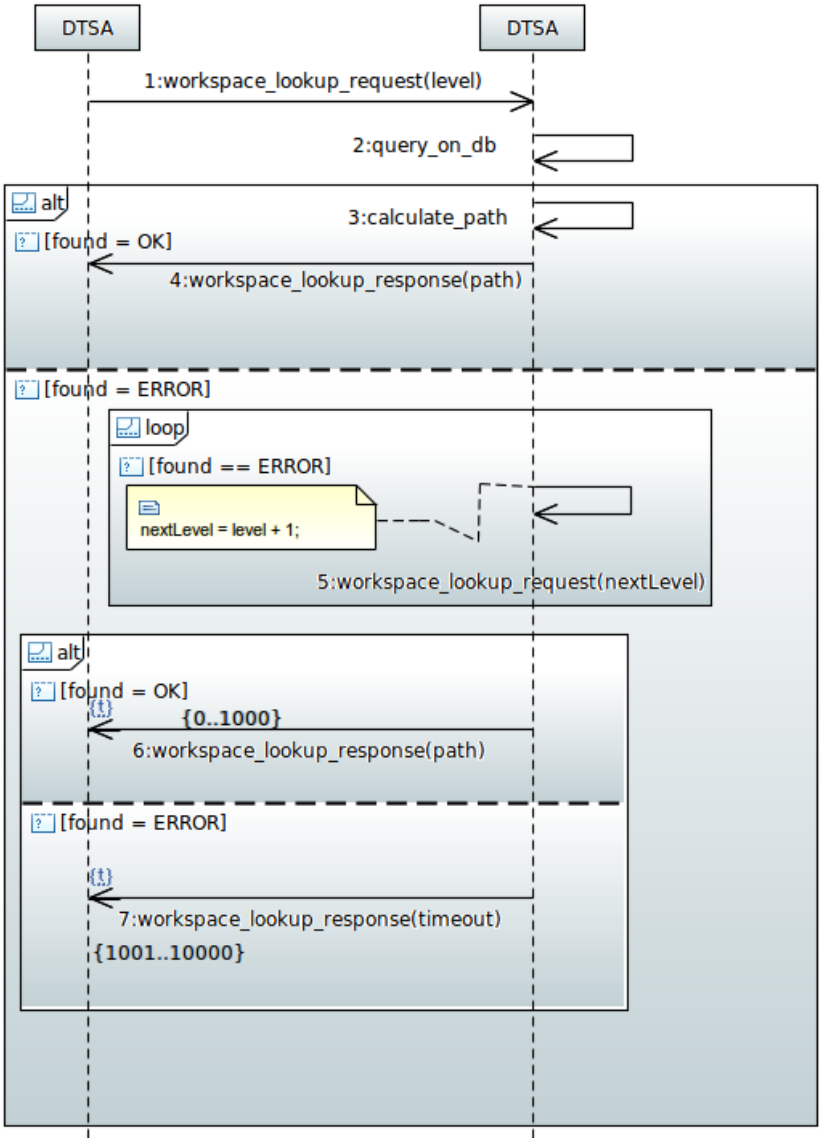

Figura 4.9: Diagrama de Sequência - *Workspace Lookup*

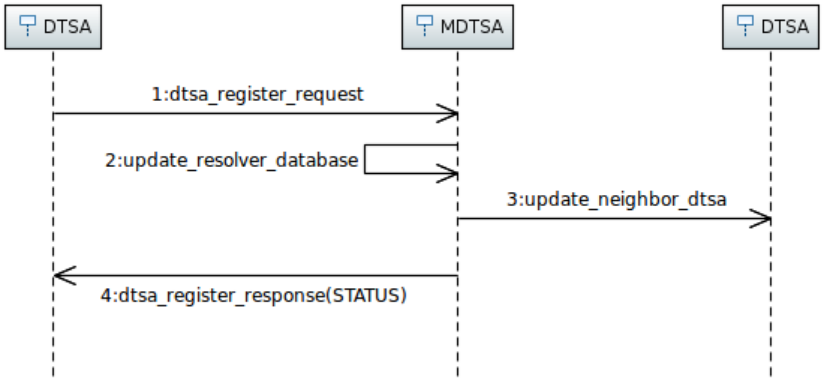

Figura 4.10: Diagrama de Sequência - *DTSA Register*

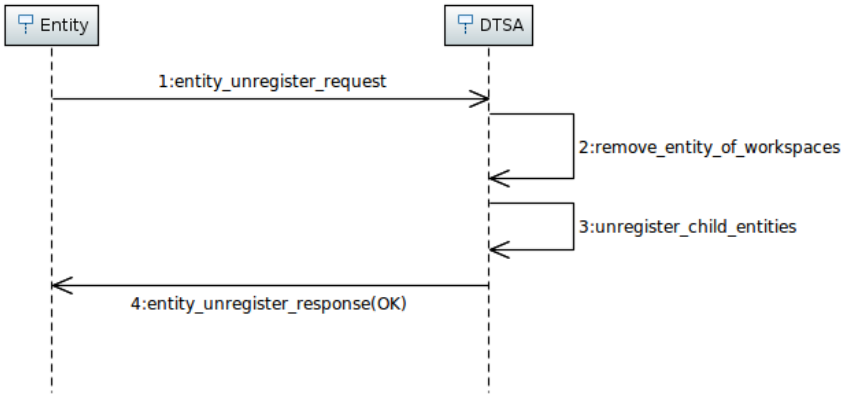

Figura 4.11: Diagrama de Sequência - *Entity Unregister*

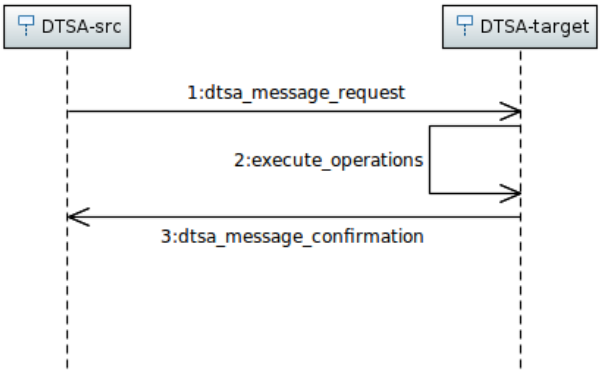

Figura 4.12: Diagrama de Sequência - *Workspace Message*

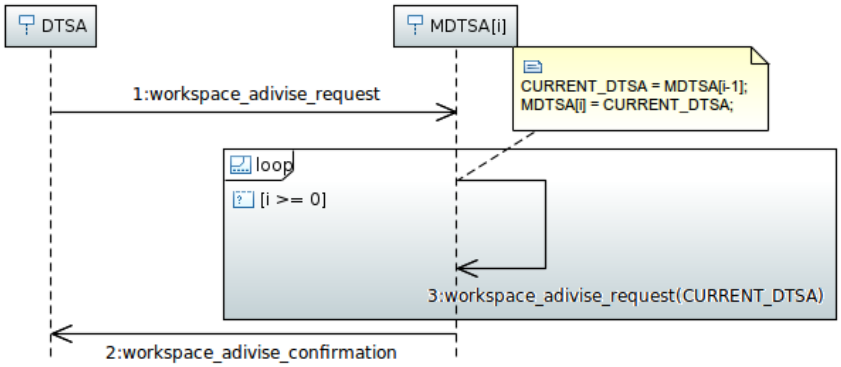

Figura 4.13: Diagrama de Sequência - *Workspace Adivertise*

Este trabalho é a primeira etapa de um caminho que visa criar um escopo formalizável da linguagem UML e profiles UML que permitam enriquecer a modelagem de protocolos de comunicação. Ou seja, definir um conjunto de elementos em que seja possível aplicar uma regra de transformação a um método validável, ou até mesmo criar tal método. Certamente, será necessário utilizar recursos de transformação de modelos, entre linguagens de modelagem diferentes. O Diagrama de Sequência, apesar de ser uma abordagem

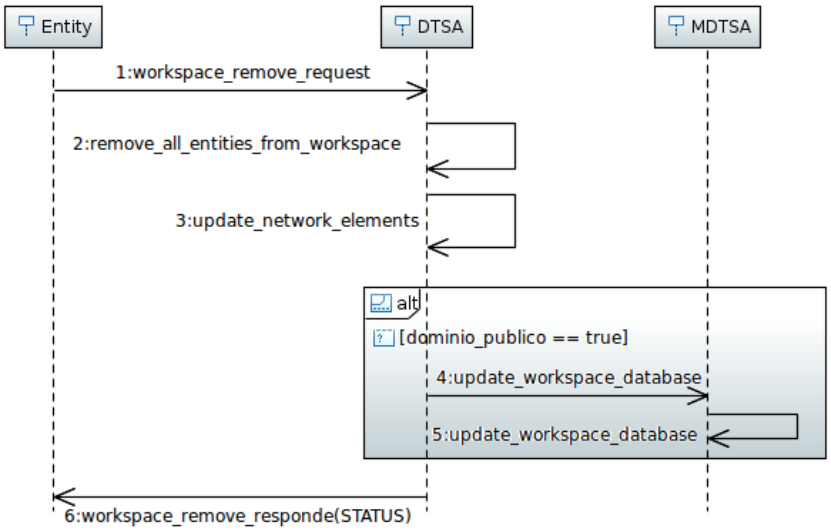

Figura 4.14: Diagrama de Sequência - *Workspace Delete*

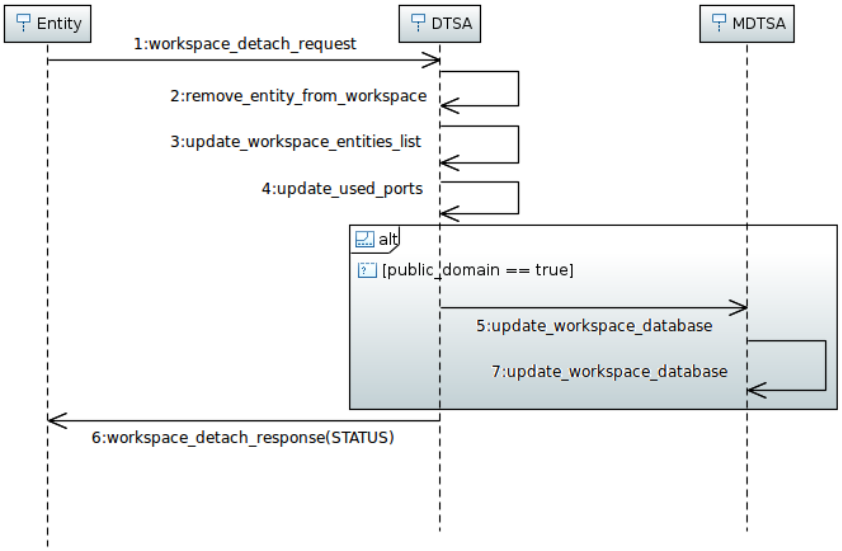

Figura 4.15: Diagrama de Sequência - *Workspace Detach*

pouco explorada neste contexto, é visualmente mais representativa que outros diagramas da UML. Para uma validação formal dos Diagramas de Sequência, vislumbramos a utilização de abordagens como a transformação em Redes de Petri [Soares e Vrancken 2008].

Ainda na perspectiva de transformação de modelos, a abordagem de modelagem de comportamento dos serviços de protocolos de comunicação usando Diagramas de Sequência é possível utilizar técnicas de sincronização entre Diagramas de Sequência e outros diagramas [Whittle e Schumann 2000] [Grønmo e Møller-Pedersen 2010].

A modelagem do protocolo ETArch, assim como a linguagem UML, segue duas vertentes, a primeira estrutural, que consiste na utilização dos diagramas de Classes e Estrutura Composta, a segunda, usando os Diagramas de Sequência, os elementos necessários são:

LifeLine, mensagens síncronas e assíncronas, fragmentos combinados (*alt* e *loop*), observação de tempo e duração.

A modelagem na linguagem UML, recorrentemente, é feita utilizando uma ferramenta de suporte para facilitar a manutenção de modelos. Neste trabalho, a modelagem foi feita utilizando a ferramenta Papyrus [Dubois et al. 2009].

Papyrus é uma ferramenta de modelagem criada utilizando a ferramenta Eclipse. Ela tem o objetivo de proporcionar um ambiente integrado para que o usuário possa editar qualquer tipo de modelo que utilize o Eclipse Modeling Framework (EMF). Esta ferramenta suporta tanto o meta-dado da linguagem UML quanto de seus *profiles*, tais como SysML e MARTE.

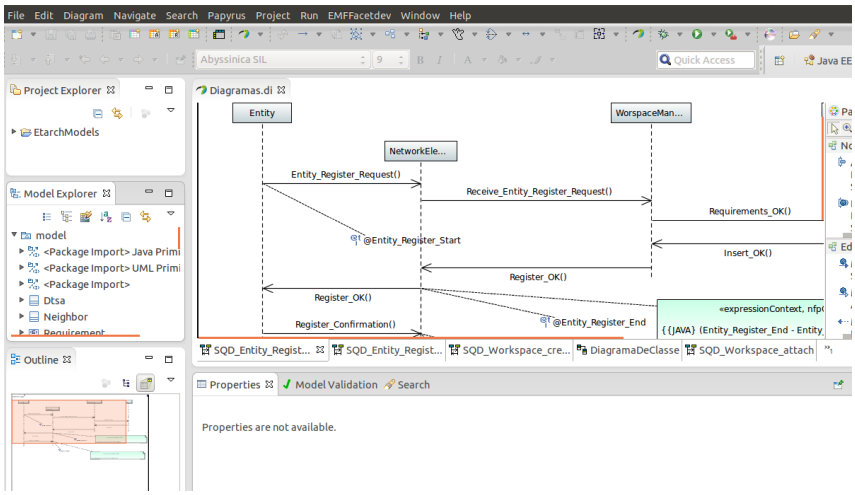

Figura 4.16: Exemplo - Ferramenta Papyrus

A Fig. 4.16 mostra o exemplo de utilização da ferramenta Papyrus com Diagramas de Sequência. O modelo é persistido utilizando um arquivo no formato XML, seguindo o padrão definido pela OMG. Como a ferramenta Papyrus é desenvolvida em Java, é possível executá-la em qualquer plataforma de sistema operacional, tornando o modelo ainda mais flexível. Uma limitação desta abordagem está no controle de versão dos modelos, pois, como todos os modelos de um projeto são feitos em apenas um arquivo, a utilização em ambientes distribuídos pode causar conflitos.

4.3 Considerações Finais

4.3.1 Diretrizes de Uso

A abordagem apresentada no presente capítulo pode ser sumarizada em diretrizes, as mesmas definem quando utilizar cada um dos diagramas e como modelar elementos específicos.

Para a modelagem da estrutura de elementos da arquitetura, como o DTSA, o Diagrama de Estrutura Composta deve ser utilizado, pois a partir dele é possível definir componentes internos em diferentes níveis de abstração. Então, o Diagrama de Estrutura Composta pode ser usado repetidas vezes permitindo que uma modelagem inicie em uma definição de alto nível e se torne mais específica e detalhada até o nível de abstração desejado. Além disso, todos os elementos que necessitam de uma definição de fluxo de dados devem utilizar o Diagrama de Estrutura Composta para a definição do mesmo.

Os atributos que não são tratados como componentes internos devem ser modelados utilizando o Diagrama de Classes, como listas, indexadores, entre outros. Este diagrama permite que o modelo represente em um nível de abstração bastante próximo da implementação orientada a objetos, tornando-se muito representativo na fase de implementação do respectivo software. Os tipos de dados utilizados ao longo da definição que não sejam nativos da UML devem ser definidos utilizando o Diagrama de Classes.

Os serviços providos pelo protocolo de comunicação devem ser modelados utilizando o Diagrama de Sequência. As *Lifelines* são representações dos elementos definidas no Diagrama de Estrutura Composta, e o fluxo de informação do Diagrama de Sequência deve seguir o mesmo que é definido no Diagrama de Estrutura Composta. As restrições de tempo são definidas utilizando a observação de tempo da linguagem UML. A unidade de tempo adotada foi o milisegundo.

4.3.2 Vantagens e Limitações

Esta abordagem mostrou a modelagem dos protocolos de comunicação DTSCP e ETCP na linguagem UML. Foi possível representar toda a estrutura dos elementos envolvidos nos serviços providos pelos protocolos presentes na arquitetura ETArch. Foi possível modelar o comportamento quanto a comunicação entre os componentes e restrições de tempo entre cada uma das chamadas. O comportamento poderia ser melhor representado com expressões estatísticas para a definição das guardas dos *Combined Fragments*.

A linguagem UML possui muitos recursos para modelar componentes que ajudam na representação de uma visão arquitetural de alto nível. Entretanto, a linguagem não provê uma definição formal para os modelos, sendo necessária uma intervenção externa baseada em outras abordagens.

A modelagem de comportamento não pode representar alguns tipos de restrição, como restrições de largura de banda e processamento. Além disso, não existe um recurso para a modelagem de hardware e sistemas embarcados, que seria utilizada para a modelagem de entidades, como sensores.

As restrições de tempo da linguagem UML permitem a definição de diferentes tipos para um limite máximo e mínimo de tempo, entretanto, não é possível criar um relacionamento entre as unidades de medida.

O uso dos *Combined Fragments* é limitado a um valor estático para predicados. Ou seja, não é possível basear o valor dos mesmos em uma expressão lógica, para que a mesma possua dinamicidade baseada em mais de um contexto. Mesmo que uma expressão seja adicionada no campo, a linguagem UML não possui um mecanismo de definição da expressão, o que a torna não verificável. Isto é considerado como uma grande limitação, pois um protocolo de comunicação com suporte a operações de tempo real, muitas vezes, precisa lidar com expressões com um grande número de operações e verificações.

Outra limitação da linguagem UML para definir protocolos de comunicação está relacionada na definição de cenários, considerando que neste caso é necessário lidar com mais de um fluxo de informações. Por um lado, esta possibilidade é uma vantagem, pois é possível definir um fluxo de dados genérico que será alterado em um nível de abstração mais específico. Mas, por outro, torna-se possível a definição de fluxos de dados não coincidentes. Por exemplo, na definição do Diagrama de Estrutura Composta, o fluxo de dados é do atributo de A a B, e no Diagrama de Sequência é possível definir uma mensagem de B para A, ferindo a definição anterior. Como não existe um método de formalização de composição de diagramas UML, torna-se impossível tal validação.

A especificação da linguagem UML é muito extensa, são sete Diagramas Estruturais e sete Diagramas Comportamentais. Cada diagrama possui um papel no projeto e isto foi levado em consideração na escolha dos diagramas utilizados no trabalho. Como existe um fator humano para a escolha dos diagramas, isto pode ser considerado uma ameaça ao trabalho.

Capítulo 5

Especificação de Protocolos de Comunicação de Tempo Real com UML / MARTE

Neste capítulo é apresentada uma abordagem de aplicação do *profile* MARTE para a modelagem de protocolos de comunicação de tempo real. A abordagem apresentada no Capítulo 4 mostra a aplicação da linguagem UML no contexto de modelagem de protocolos de comunicação. Entretanto, foram apresentadas limitações da linguagem na modelagem de restrições de tempo, hardware, representação de memória e semântica de elementos no modelo.

A abordagem deste capítulo visa atacar as limitações apresentadas no Capítulo 4 para a modelagem de serviços dos protocolos ETCP e DTSCP da arquitetura de Internet ETArch. A grande vantagem da mesma está relacionada com a facilidade de leitura dos modelos gerados. Os modelos definem tanto a parte estrutural quanto comportamental de um protocolo de comunicação.

Este capítulo é dividido em cinco seções. A Seção 5.1 apresenta a motivação para a criação da abordagem, bem como o objetivo da mesma. A Seção 5.2 apresenta os modelos criados, os conceitos e os serviços do protocolo. Além da utilização de uma ferramenta para a modelagem dos serviços, foi necessária a criação de um analisador para expressões, este conteúdo é apresentado na Seção 5.3. Ao final, são apresentadas as considerações finais sobre a abordagem, mostrando as vantagens e limitações da mesma, na Seção 5.4.

5.1 Motivação

Algumas propostas para resolver os problemas da linguagem UML, em relação à modelagem de software em tempo real, foram criadas. Uma tentativa foi o *profile Schedulability, Performance and Time* (SPT), que provê um mecanismo para anotação de um conjunto

pré-definido de estereótipos e *tagged values* [Xu et al. 2003] [Bennett e Field 2004]. No entanto, essa abordagem não permite novas definições de propriedades não funcionais pelo usuário ou para diferentes campos especializados. Portanto, o *Object Management Group* (OMG) especificou um novo *profile* da UML para permitir a modelagem e análise de sistemas de tempo real e embarcados.

O *profile* MARTE (*Modeling and Analysis of Real-Time and Embedded Systems*) [Graf et al. 2006], desde a sua criação, tem sido aplicado em muitos domínios, como na modelagem de sistemas robóticos [Demathieu et al. 2008], linhas de produtos [Belategi et al. 2010], geração de código executável a GPU (*Graphics Processing Unit*) [Rodrigues et al. 2011], ambientes industriais [Shousha et al. 2012] e modelagem de ambiente [Iqbal et al. 2012]. No entanto, abordagens com foco na aplicação de MARTE/UML para a modelagem de protocolos de comunicação não foram encontradas na literatura.

Visando aumentar o poder de representação de semântica e restrições dos modelos criados no Capítulo 4, este trabalho apresenta uma abordagem para criar modelos flexíveis, reutilizáveis e conectáveis de protocolos de comunicação de tempo real, utilizando o *profile* MARTE/UML. Como não foram encontradas propostas de abordagens com foco na aplicação de MARTE/UML juntos para projetar protocolos de comunicação de tempo real, para demonstrar seus pontos fortes, mostramos a modelagem de serviços dos protocolos da arquitetura ETArch. Essa modelagem consiste em definir os elementos do *profile* MARTE que são aplicados para modelar serviços do ETArch Control Protocol (ETCP) usando: estereótipos, definição de normas e escopo. A abordagem é capaz de modelar serviços do protocolo ECTP com a finalidade de definir um alcance formal da modelagem.

Todos os modelos criados foram feitos utilizando a ferramenta Papyrus com o *plugin* dos meta-dados do *profile* MARTE. As expressões feitas na linguagem VSL utilizadas nos modelos foram sintaticamente validadas por um software externo criado pela equipe de pesquisa.

5.2 Modelagem usando MARTE

Na presente seção são apresentados os artefatos da arbodagem, contendo os modelos usando MARTE dos serviços dos protocolos ETArch bem como os elementos envolvidos nos protocolos. Entretanto, como o objetivo deste trabalho é complementar a abordagem do Capítulo 4 com a utilização de recursos do *profile* MARTE, são apresentados os diagramas que sofreram modificações com a inserção do *profile*.

Para que o *profile* MARTE pudesse ser aplicado nos modelos já existentes, foram necessárias modificações nos mesmos, acrescentando suporte aos tipos de dados do MARTE, como no Diagrama de Classes. Além disso, requisitos não funcionais foram inseridos no escopo de modelagem da arquitetura ETArch, entre eles expressões para o controle de capacidade do *Storage*. Expressões que antes foram feitas utilizando comentários,

nesta abordagem foram modeladas com expressões na linguagem VSL, como no serviço de *Workspace Lookup*. Desta forma, o objetivo da abordagem utilizando o *profile* MARTE se tornou mais amplo que o objetivo da abordagem apresentada no Capítulo 4.

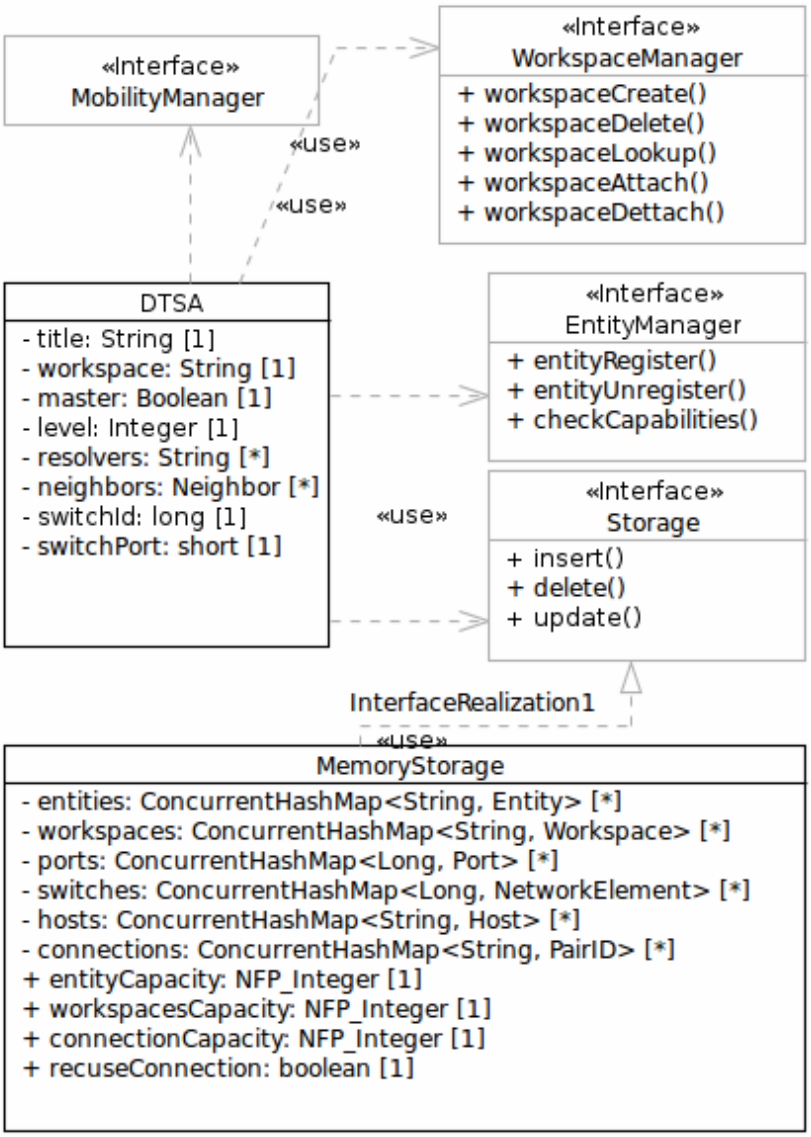

Figura 5.1: Diagrama de Classes - Principais elementos ETArch

A Fig. 5.1 apresenta o Diagrama de Classes dos principais elementos da ETArch. Neste nível de abstração, os elementos são apresentados, bem como as suas propriedades e a relação para com outros elementos. Este nível abstração de modelos fornece uma grande base técnica para a implementação do elemento. A principal alteração neste diagrama foi a adição dos atributos no *MemoryStorage* para a representação da capacidade do mesmo para o armazenamento de entidades, *workspaces* e conexões. Os atributos adicionados utilizam o tipo *NFP_Integer* que é pelo *profile* MARTE.

A Fig. 5.2 descreve o fluxo de dados da estrutura interna de um DTSA de forma

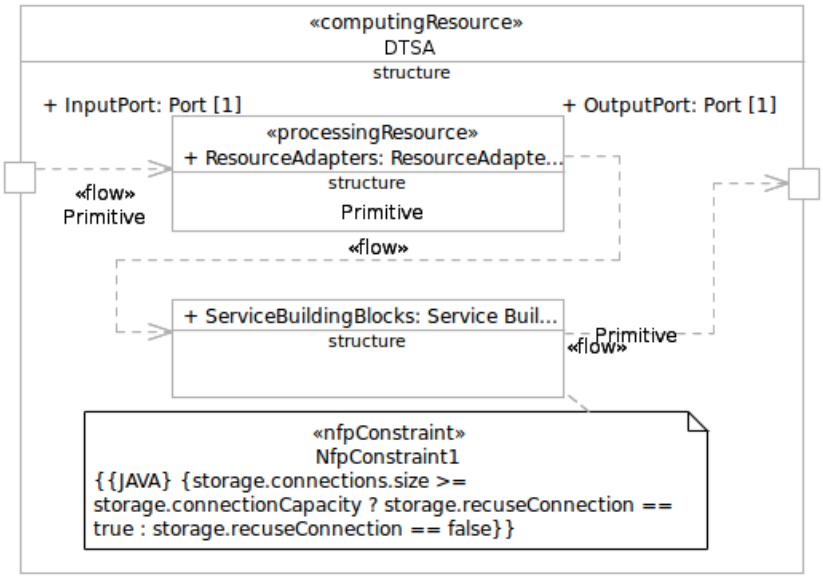

Figura 5.2: Estrutura do DTSA

abstrata, mostrando exemplos de classificadores internos. A estrutura interna pode ser dividida em dois módulos que fazem referência a um classificador em outro diagrama. É importante mencionar que existe uma restrição de propriedades não funcionais no elemento *Service Building Blocks*, a mesma pode alterar o fluxo da conexão se a capacidade é excedida.

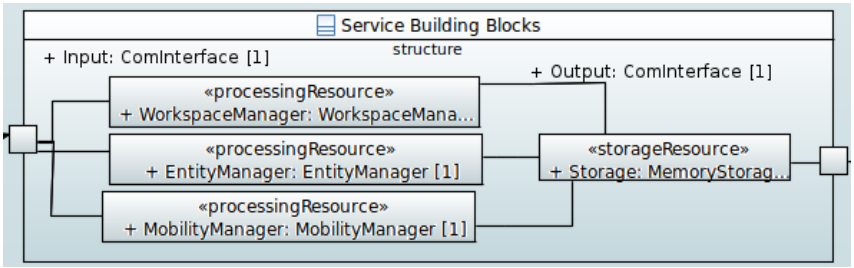

Figura 5.3: Estrutura de Service Building Blocks

A Fig. 5.3 apresenta uma estrutura de um módulo DTSA, chamada de *Service Building Blocks*. Este módulo contêm os gerenciadores de entidade, *workspace*, mobilidade e persistência. A estrutura possui duas interfaces de comunicação, *input* e *output*. No entanto, estas portas não fazem qualquer referência a uma ligação comum de Internet, isto significa que estes módulos não são necessariamente remotos. Neste nível de abstração, os estereótipos *processingResource* e *storageResource* são utilizados para definir a semântica dos elementos.

A Fig. 3.1 apresenta um exemplo de um cenário da arquitetura ETArch que contempla a necessidade de uso de todas os serviços, ou primitivas, dos protocolos ETCP e DTSCP. A modelagem de cenários da arquitetura em questão é tão importante quanto

a modelagem do comportamento dos elementos de forma individual. O pacote *Foundations* do *profile* MARTE apresenta um conceito de grande utilidade para a modelagem de cenários. Um elemento MARTE pode ser de dois tipos, um classificador ou uma instância. Um classificador consiste em um elemento com o objetivo de realizar uma definição abstrata de estruturas, restrições e atributos. O segundo tipo de elemento consiste em uma instância de um determinado classificador, a mesma define as propriedades não definidas no classificador e atribui valores aos atributos e restrições. Além disso, é possível criar classificadores a partir de classificadores, desta forma, ao se obter uma instância do classificador filho, as propriedades serão herdadas de ambos os classificadores.

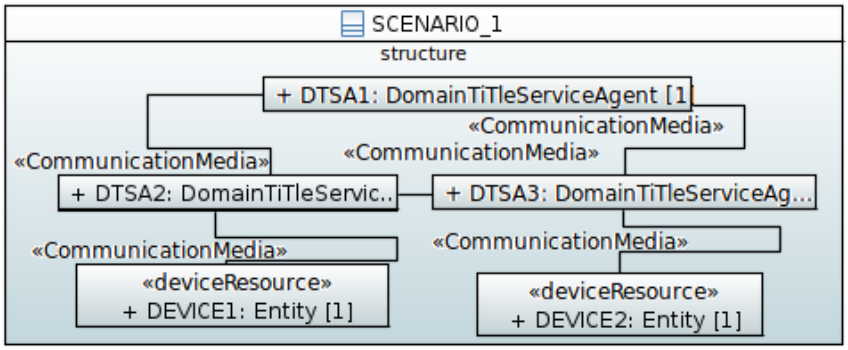

Figura 5.4: Cenário proposto para a modelagem

A Fig. 5.4, assim como a Fig. 3.1, apresenta um cenário em que os serviços são executados, mas esta especificação utiliza as definições da linguagem UML e do *profile* MARTE. Neste cenário, há três DTSAs ligados de forma hierárquica e duas entidades ligadas nos DTSAs. Todos os elementos deste cenário são conectados usando o estereótipo *CommunicationMedia*, que representa um canal de comunicação. Isso permitirá a definição de propriedades como tamanho da primitiva transitada e velocidade de transferência. As propriedades deste estereótipo podem influenciar diretamente na limitação de tempo dos serviços que utilizam a conexão entre esses elementos.

A modelagem do comportamento dos protocolos de comunicação sofre intervenção do cenário em que os elementos estão presentes, para isto, os Diagramas de Sequência foram criados utilizando uma referência do *LifeLine* para o respectivo elemento no cenário descrito na Fig. 5.4. Além disso, a partir desta referência, a criação de um método de formalização deve levá-la em consideração para uma possível validação do modelo.

A Fig. 5.5 apresenta o Diagrama de Sequência do serviço *Entity Register* e faz referência aos elementos definidos na Fig. 5.4. A diferença entre a definição utilizando o *profile* é o uso de expressões VSL para verificar a viabilidade de conexões.

Na situação em que a entidade deseja anexar-se a um *Workspace* e o DTSA onde esta entidade está registrada não possuir informações a respeito deste *Workspace*, ele utilizará o serviço *Workspace Lookup* para encontrar as informações desejadas, ou seja, o caminho para o *Workspace* desejado. Inicialmente, o DTSA envia uma requisição ao MDTSA em

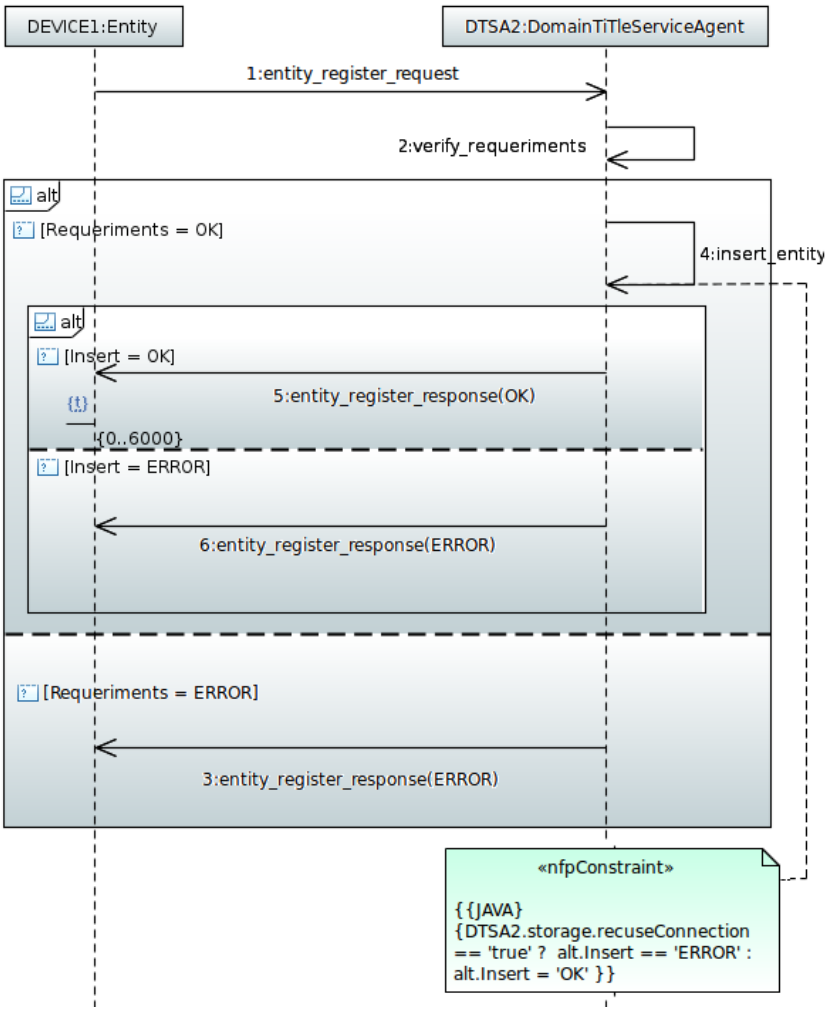

Figura 5.5: Operação Entity Register

que está registrado. Ao receber a requisição do tipo *Workspace Lookup*, o MDTSA irá criar um evento, o mesmo será capturado pelo gerenciador de *Workspaces*, que é responsável por realizar a busca pelo *Workspace* desejado. Caso encontre o *Workspace*, então o mesmo realizará um cálculo para encontrar o caminho de menor custo. Este caminho é composto de uma lista de DTSAs que permite ao DTSA que originou a requisição chegar até o *Workspace* desejado.

A Fig. 5.6 apresenta o Diagrama de Sequência do serviço *Workspace Lookup*, da definição de duas expressões na linguagem VSL. As expressões na linguagem VSL tem como objetivo representar a semântica de chamada do MDTSA.

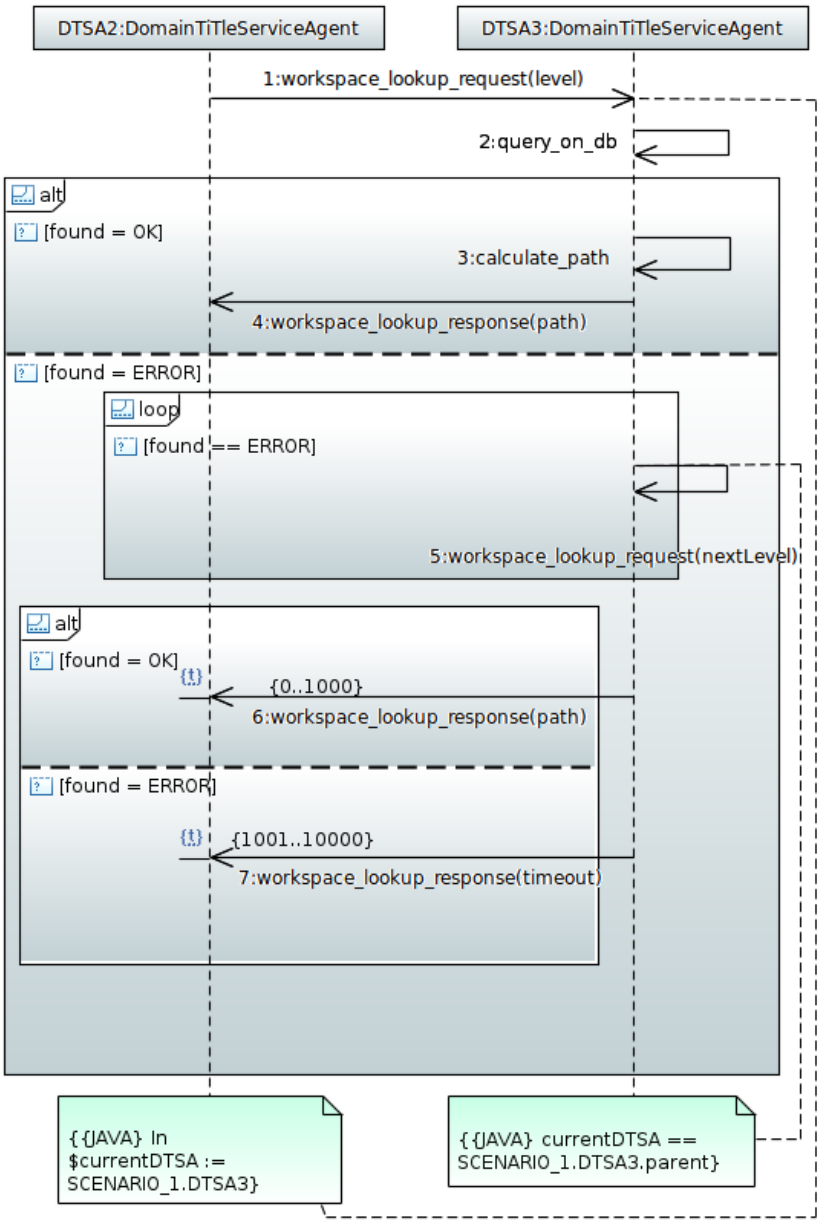

Figura 5.6: Operação Workspace Lookup

5.3 Analisador da Linguagem VSL

Na ferramenta Papyrus existem dois analisadores com suporte a expressões VSL, o primeiro é o mesmo utilizado para expressões *Object Constraint Language* (OCL), o segundo é adicionado juntamente com o *plugin* do *profile* MARTE. Porém, a validação de ambos não foi satisfatória para o trabalho, pois a apresentação do resultado da análise das expressões de acordo com a gramática VSL não é apresentada ao final do processo.

Desta forma, foi utilizado um editor genérico da ferramenta Papyrus e as expressões foram validadas sintaticamente pela ferramenta criada.

Para a criação do analisador léxico e sintático foi utilizada a ferramenta ANTLR [Parr 2013]. A mesma foi escolhida levando em consideração outros analisadores presentes na

ferramenta Papyrus que também utilizam a ferramenta ANTLR para a criação de seus respectivos analisadores sintáticos.

5.3.1 ANother Tool for Language Recognition (ANTLR)

A ANTLR [Parr 2013] é uma ferramenta para criação de tradutores, analisadores léxicos e sintáticos. Um tradutor mapeia cada sentença de entrada de uma linguagem em uma sentença de saída. Para realizar o mapeamento, o tradutor executa o código fornecido, operando sobre os símbolos de entrada e emite a saída. Um tradutor deve executar ações diferentes para sentenças diferentes, o que significa que ele deve ser capaz de reconhecer várias sentenças, porém a função de reconhecimento deverá ser bijetora, para garantir a não ambiguidade.

A ANTLR foi escrita em Java. A mesma possui duas formas básicas de execução: execução visual, onde existe uma interface gráfica da ferramenta ANTLR, que permite analisar a sentença, e execução usando chamada de biblioteca, que deverá ser feita por outra aplicação.

Para que a ferramenta reconheça a gramática inserida, é necessário que a mesma siga a sintaxe de definição da API, no seguinte padrão:

```
expr: \wedge('+' expr expr)
    |  \wedge('*' expr expr)
    | INT;
```

A ferramenta faz a leitura da definição da gramática que está presente em um arquivo externo e cria os arquivos necessários do analisador na linguagem Java. Um arquivo de definição de gramática reconhecido pela ferramenta deve ser do tipo ".g4". Além da definição básica da gramática, é possível definir uma série de opções para customização do analisador.

5.3.2 Análise da linguagem VSL

O primeiro passo para a criação do analisador da linguagem VSL foi a extração da definição da gramática que estava presente na especificação do profile MARTE. A mesma não estava no padrão definido pelo ANTLR, desta forma, foi necessário realizar uma normalização da definição.

- Exemplo de definição na especificação MARTE:

  ```
  <value-specification> ::= <literal>
  ```

```
| <enum-specification>
| <interval>
| <collection>
| <tuple>
| <choice>
| <expression>
| <time-expression>
| <obs-call-expression>
```

- Normalização para a sintaxe do ANTLR:

```
value_specification
: literal
| enum_specification
| interval
| collection
| tuple
| choice
| expression
| time_expression
| obs_call_expression
;
```

Apesar de bem definida, a definição da linguagem VSL na especificação MARTE necessitou ser adaptada para que fosse possível a definição, como na sentença *namespace*:

- Definição VSL:

```
<body-text> ::=  {Qualquer sequência de caracteres}
<namespace> ::= [ <body-text> '.' <namespace>]
<namespace> ::= [ <body-text> <namespace> '.']
```

- Definição ANTLR:

```
$namespace_scope
:  ( namespace Scope_Separator )+
;
namespace_expr
: Identifier namespace_scope?
```

```
;
namespace
: Identifier
;
```

O *token Identifier* é definido pela ferramenta ANTLR e o mesmo representa qualquer sequência de caracteres válida. As expressões foram validadas sintaticamente pelo analisador externo e adicionadas no modelo pelo editor genérico. O editor genérico da ferramenta Papyrus adiciona o prefixo "JAVA" nas restrições.

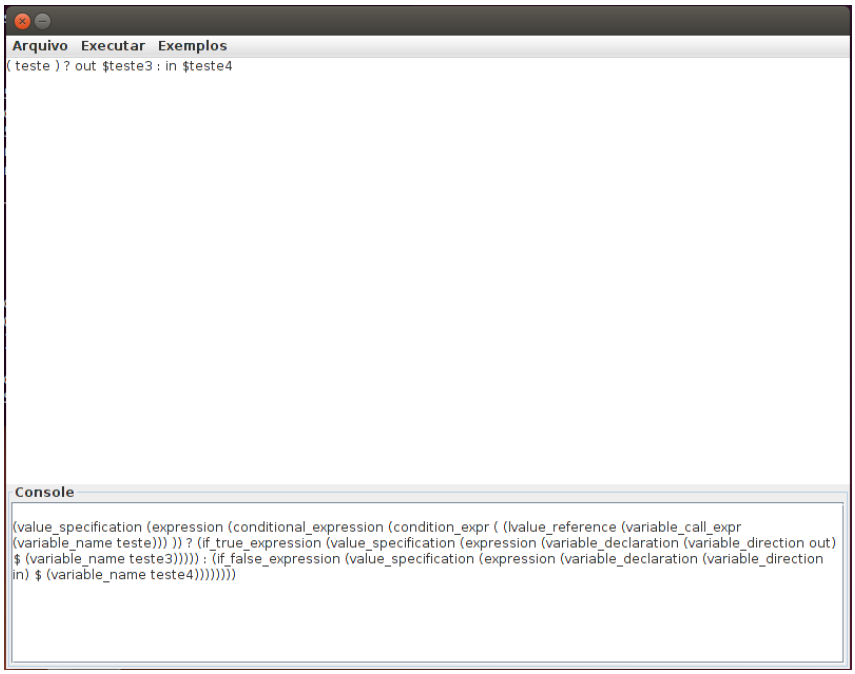

Figura 5.7: Analisador VSL

A Fig. 5.7 mostra um exemplo de utilização do validador para uma expressão condicional. Em caso de sucesso, o validador apresenta como resultado a árvore sintática baseada na definição da gramática presente no arquivo. Caso a expressão não esteja correta, o motivo da falha é apresentado no console.

A análise léxica consiste em transformar os símbolos de entrada em identificadores e verificar se os mesmos estão no alfabeto, ou seja, na definição da linguagem. A análise sintática consiste em verificar se a sequência de entrada está de acordo com as regras de formação do alfabeto. Como exemplo, será apresentada a definição de uma expressão condicional. Uma expressão condicional na linguagem VSL deverá possuir o formato:

```
conditional_expression
```

```
: condition_expr Conditional_Symbol
        if_true_expression
        Branch_Symbol if_false_expression
;

Conditional_Symbol
: '?'
;

condition_expr
: Left_Paren lvalue_reference Right_Paren
;

if_true_expression
: value_specification
;

if_false_expression
: value_specification
;
```

Desta forma, a partir de derivações é possível criar uma da seguinte formas:

```
( teste ) ? out $teste3 : in $teste4
```

A derivação da expressão pela ferramenta ANTLR terá como resultado uma árvore sintática no formato de texto.

```
(value_specification (expression (conditional_expression
    (condition_expr
     ((lvalue_reference
        (variable_call_expr (variable_name teste))) ))
        ? (if_true_expression
          (value_specification
           (expression (variable_declaration (variable_direction out)
             $ (variable_name teste3)))))
        : (if_false_expression
```

```
(value_specification
 (expression (variable_declaration (variable_direction in)
   $ (variable_name teste4)))))))
```

Com a utilização de um analisador próprio, a liberdade de customização do mesmo é aumentada. O analisador poderá não somente realizar a análise léxica e sintática, mas também poderá gerar artefatos de implementação. Para uma geração de artefatos mais efetiva, os modelos também devem ser analisados e levados em consideração.

Os modelos na ferramenta Papyrus são persistidos em arquivos XML, que seguem o padrão definido pela OMG. A ferramenta permite a criação de *plugins*, desta forma, é possível vislumbrar um *plugin* para a ferramenta Papyrus que permita a validação estrutural e comportamental de arquiteturas de Internet que utilizem as diretrizes apresentadas na abordagem.

5.4 Considerações Finais

5.4.1 Diretrizes

As diretrizes de uso da abordagem apresentada no presente capítulo podem ser aplicadas em conjunto com as diretrizes apresentadas no Capítulo 4.

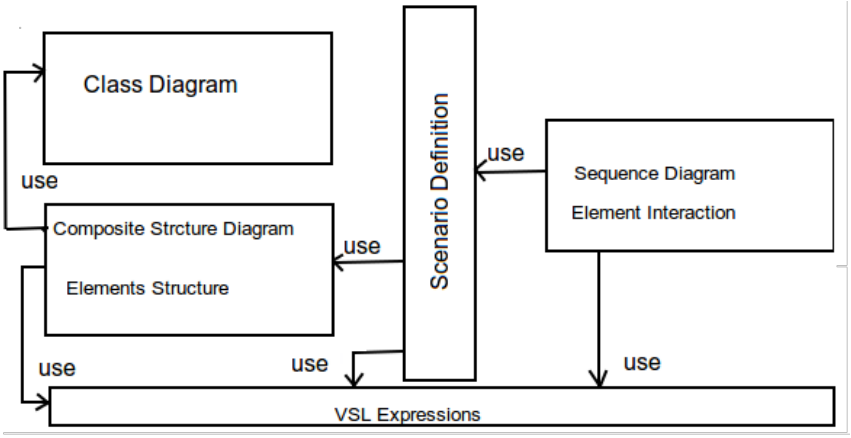

Figura 5.8: Abordagem de Modelagem utilizando MARTE

A Fig. 5.8 apresenta o diagrama com o papel de cada nível de abstração na abordagem. O relacionamento do tipo *use* tem como objetivo representar a dependência entre os diagramas. Na criação de um método para a formalização da abordagem, a definição das dependências se torna essencial para definir uma precedência de validação. Como exemplo, os Diagramas de Sequência utilizam recursos na definição de cenários, que por sua vez utilizam a definição de elementos feita com Diagrama de Estrutura Composta. Desta

forma, uma definição feita no Diagrama de Estrutura Composta terá maior precedência que o Diagrama de Sequência.

A abordagem apresenta quatro níveis de abstração: definição de atributos dos elementos, fluxos de dados, cenários e comportamentos. Os cenários podem ser aninhados utilizando o conceito de classificador e instância do pacote *MARTE Foundations*. As expressões VSL atuam transversalmente no modelo porque elas estão presentes em vários níveis de abstração.

Os canais de comunicação devem ser definidos obrigatoriamente com a atribuição do estereótipo *CommunicationMedia*. Quando dispositivos possuem apenas uma interação de comunicação com elementos da arquitetura, deve-se aplicar o estereótipo *DeviceResource*, desta forma, mesmo que exista uma modelagem do comportamento interno do dispositivo, a mesma não pode ser levada em consideração no contexto de protocolos.

Os elementos que possuem armazenamento de dados possuem o estereótipo *StorageResource*, desta forma, existe uma representação padronizada da capacidade.

As expressões VSL são importantes na validação de dados, tempo e mudança de valor nos elementos. Este trabalho mostra o uso da linguagem VSL para melhorar a semântica do modelo, a mesma pode ser aplicada em outros cenários de protocolos de comunicação, tanto plano de controle como plano de dados.

5.4.2 Vantagens e Limitações

Neste trabalho, uma abordagem de aplicação do *profile* MARTE juntamente com a linguagem UML para a modelagem de protocolos de comunicação de tempo real é proposta. Poucas abordagens foram propostas com foco na aplicação de MARTE e UML juntas para modelagem de protocolos de tempo real. Com a aplicação do *profile*, é possível criar modelos com muitos níveis de abstração e, em seguida, aumentar a robustez do modelo usando expressões algébricas. As expressões são validadas por um software desenvolvido para verificar expressões VSL em modelos UML/MARTE.

A principal vantagem desta abordagem é que o modelo criado tem tanto uma visão comportamental quanto uma visão estrutural da arquitetura ETArch. Além disso, realiza a validação algébrica inserida nos modelos por meio de anotações com a linguagem VSL. Portanto, a equipe de desenvolvimento pôde projetar modelos robustos que são úteis tanto nas fases iniciais do projeto para orientar a implementação, como nas últimas fases para validar a arquitetura implementada de um protocolo de comunicação.

Os estereótipos MARTE acrescentaram a semântica necessária para a representação de elementos de rede, canais de comunicação e elementos de armazenamento. Este recurso contribui para a padronização das propriedades dos protocolos de comunicação, tornando a aplicação de um método de validação mais simples, não sendo mais necessária a padronização de estruturas, como um canal de comunicação. Além disso, o estereótipo

Device Resource permite criar elementos que não necessitam de uma validação para o comportamento interno.

Uma limitação do *profile* MARTE é a curva de aprendizam, porque a mesma tem uma definição muito extensa e o processo de escolha dos estereótipos exige a leitura de toda a especificação MARTE e identificação do contexto em que cada definição pode ser utilizada.

Esta abordagem consiste na primeira etapa da definição dos elementos MARTE/UML para que seja possível aplicar um método de validação no contexto de modelagem de protocolos. Desta forma, é possível aumentar as validações com expressões VSL no modelo e aumentar a eficácia da validação do domínio de modelagem de protocolo. Ao especificar arquitetura ETArch e seus protocolos em UML/MARTE, é possível para os analistas modelar e validar os módulos no software de forma integrada com a arquitetura.

Capítulo 6

Resultados Obtidos e Avaliação

Neste capítulo é apresentada a avaliação da abordagem de modelagem de serviços de protocolos de tempo real com o *profile* MARTE apresentado no Capítulo 5. A avaliação qualitativa consiste em apresentar a abordagem usando um guia e receber o retorno de indivíduos usando um questionário definido pelo modelo de aceitação *Technology Acceptance Model* (TAM).

O capítulo está dividido em quatro seções, a presente seção que apresenta a introdução ao tema. A Seção 6.1 tem como objetivo apresentar o modelo de aceitação. A Seção 6.2 apresenta os resultados do questionário. A Seção 6.3 apresenta a avaliação qualitativa do trabalho baseada em comentários dos usuários. A Seção 6.4 apresenta a conclusão obtida a partir do retorno dos indivíduos.

6.1 Avaliação

Muitos modelos foram propostos para explicar e prever a utilização de uma tecnologia. Um modelo amplamente empregado de adoção de projetos de Tecnologia da Informação (TI) é o TAM [Venkatesh e Bala 2008]. O modelo sugere que quando os usuários são apresentados com uma nova tecnologia, uma série de fatores influenciam a sua decisão sobre como e quando eles vão usá-la. Ele postula que a intenção comportamental dos indivíduos para usar uma tecnologia é determinada por duas crenças: utilidade percebida, definida como o grau em que uma pessoa acredita que o uso de uma tecnologia irá melhorar o seu desempenho no trabalho, e percepção de facilidade de uso, definida como o grau para o qual uma pessoa acredita que a utilização de um sistema específico estará livre de esforço. Além disso, a vontade do indivíduo de aprender sobre a tecnologia também será utilizada como parâmetro do questionário.

Para que os usuários pudessem responder o questionário, foi criado um documento contendo um guia com os objetivos de apresentar os conhecimentos básicos sobre o *profile* MARTE e a arquitetura ETArch, e ainda, apresentar os modelos criados utilizando Autômatos, UML e MARTE/UML. O texto do guia foi modificado apenas o necessário

para a formatação.

Profile MARTE

O profile MARTE/UML permite a modelagem de vários cenários em diferentes níveis de abstração. Entre eles, modelagem de software, hardware e alocação de recursos. Além disso, ele permite a criação da análise quantitativa e de desempenho do software.

Tanto a modelagem de software quanto a modelagem de hardware podem ser feitas de forma independente. Quando há a necessidade de representar a execução de um software por um hardware, ambos modelados no profile MARTE, utiliza-se a modelagem de alocação.

ETArch

A Entity Title Architecture (ETArch) é uma arquitetura de rede *clean slate*, onde esquemas de nomeação e endereçamento são baseados em uma designação independente de topologia que identifica uma entidade, chamada de título, e na definição de um canal que reúne várias entidades de comunicação, chamado Workspace.

Um componente fundamental dessa arquitetura é o Domain Title Service (DTS), que trata de todos os aspectos da rede de controle. O DTS é composto por Domain Title Service Agents (DTSAs), que mantêm informações sobre entidades registradas no domínio e os espaços de trabalho que são subscritos, com o objetivo de configurar os dispositivos de rede para implementar os *workspaces* e para permitir que os dados para chegar a todas as entidades subscritas.

São apresentados os modelos dos protocolos ETCP e DTSCP utilizando Autômatos, UML e MARTE/UML.

Modelagem Utilizando Autômatos

A grande diferença entre a Fig. 6.5 e Fig. 6.6 consiste nos estereótipos utilizados. Quando utilizamos o estereótipo *processingResource*, o elemento passa a possuir uma semântica que indica que o mesmo executa um processamento que é atômico para o contexto, podendo até ter processadores externos. O estereótipo *storageResource* consiste em um elemento que armazena dados, o mesmo possui um atributo para descrever a capacidade do mesmo.

A definição da semântica dos elementos aumentou a robustez do modelo, definindo melhor a visão arquitetural.

Além da estrutura interna do DTSA, a Fig. 6.8 apresenta uma expressão que valida a capacidade de armazenamento de elementos do *Storage* presente dentro do objeto *ServiceBuildingBlocks* que é definido na Fig. 6.6. Desta forma, todas as instâncias do elemento DTSA (*Domain Title Service Agent*) possuem esta restrição.

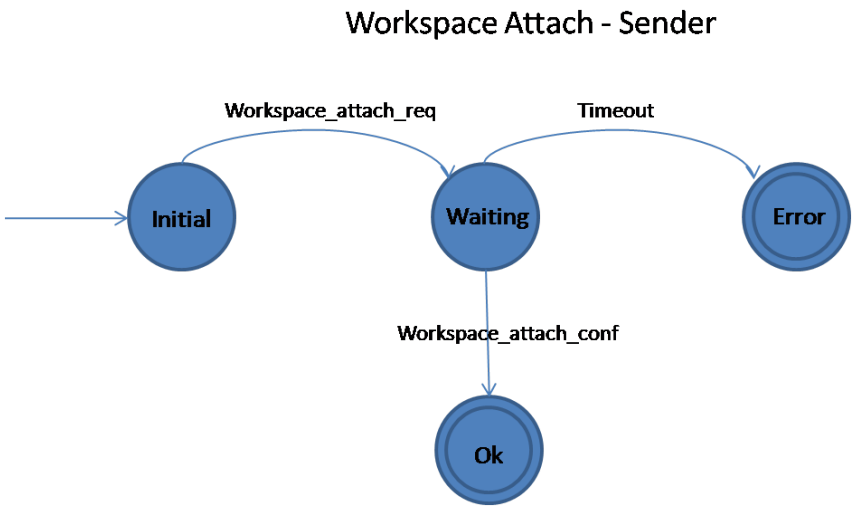

Figura 6.1: Serviço *Workspace Attach* Representado por Autômato

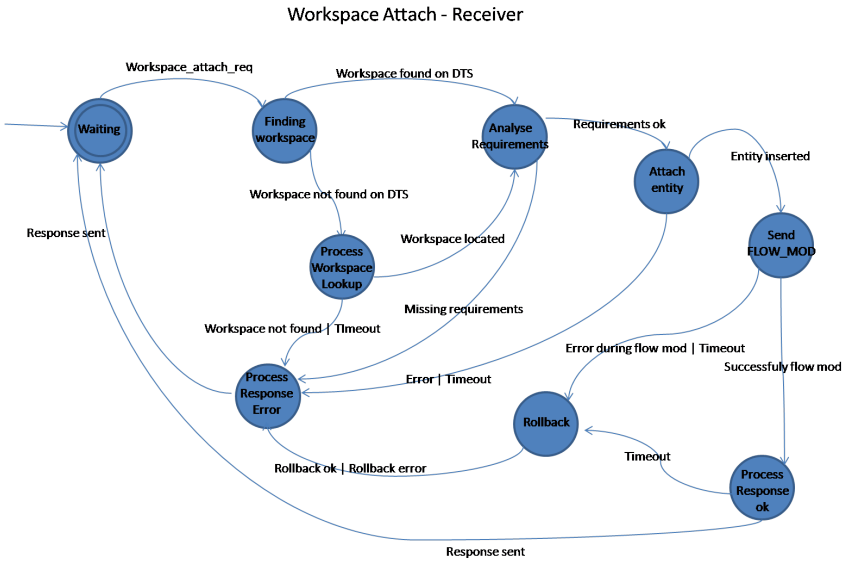

Figura 6.2: Serviço *Workspace Attach* Representado por Autômato

A Fig. 6.8 mostra uma expressão que instancia uma variável e outra que faz uma atribuição de valores. A atribuição da Fig. 6.10 formaliza a demonstrada na Fig. 6.8.

Quando utilizamos os estereótipos significa que o elemento possui uma semântica e pode possuir propriedades. O estereótipo *ComunicationMedia* possui a semântica que representa transmissão de dados, ou seja, se um elemento possui este estereótipo ele será responsável por transferir dados. Este elemento possui propriedades como tamanho da primitiva transitada, velocidade de transmissão e política de transmissão. O estereótipo *DeviceResource* consiste em um dispositivo que interage com o modelo, mas que o processamento interno não deverá ser considerado. Esta semântica foi utilizada para enriquecer os modelos.

As expressões da linguagem VSL utilizadas neste trabalho possuem a função de inserir

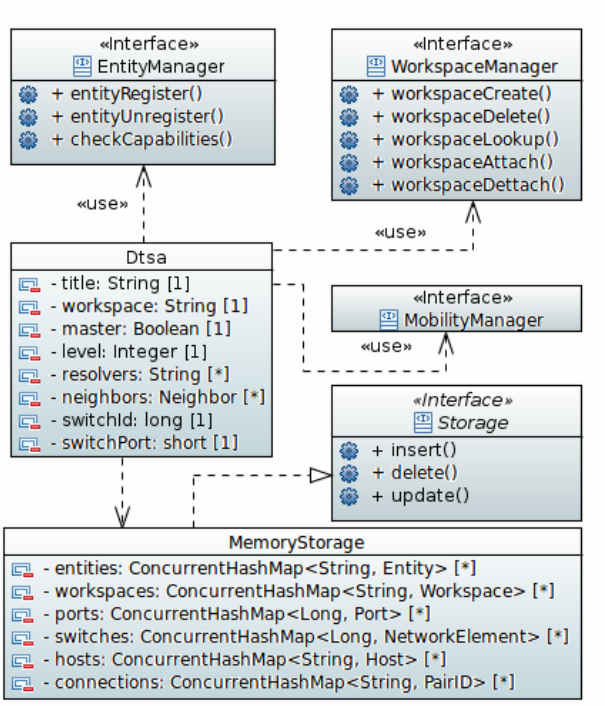

Figura 6.3: Diagrama de Classes dos Elementos ETArch - UML

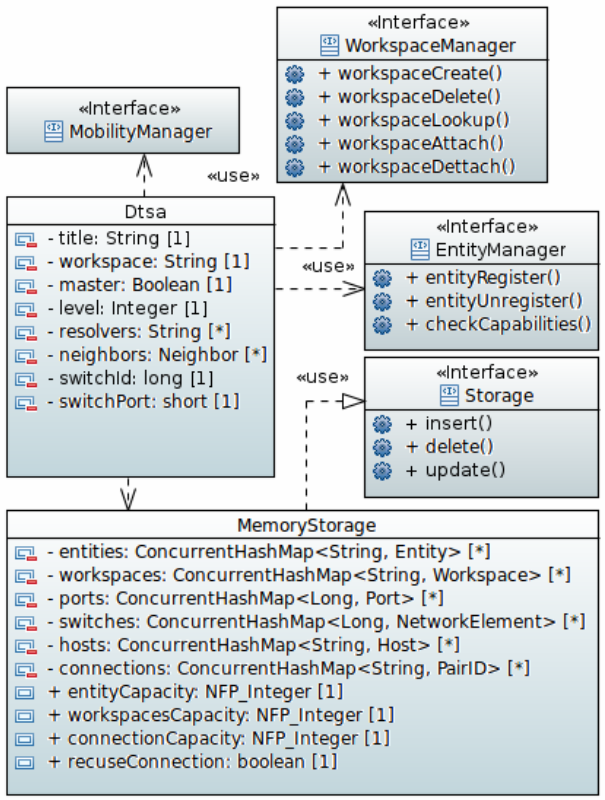

Figura 6.4: Diagrama de Classes MARTE/UML

restrições no modelo.

Exemplos de outras expressões VSL:

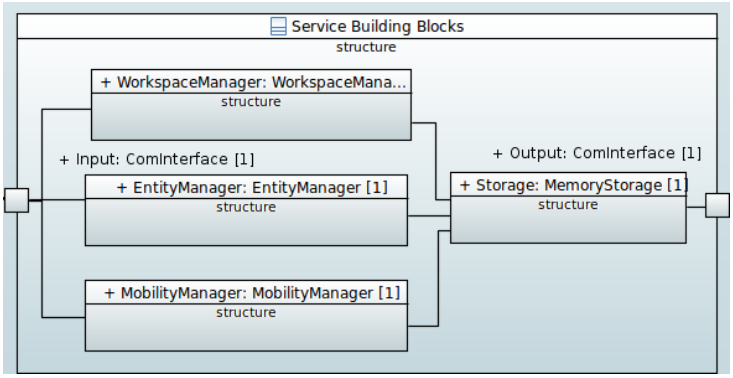

Figura 6.5: Diagrama Estrutura Composta UML

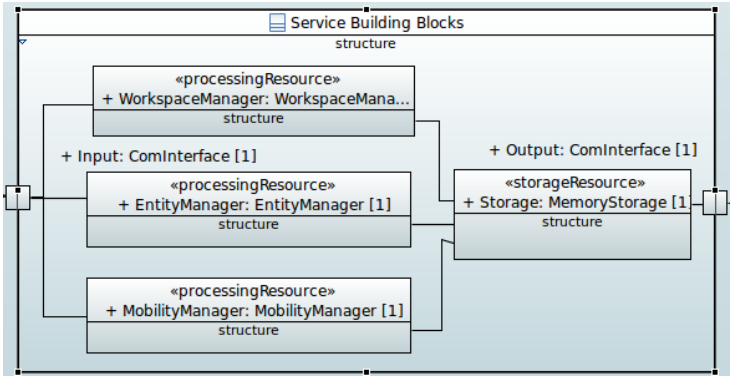

Figura 6.6: Diagrama Estrutura Composta MARTE/UML

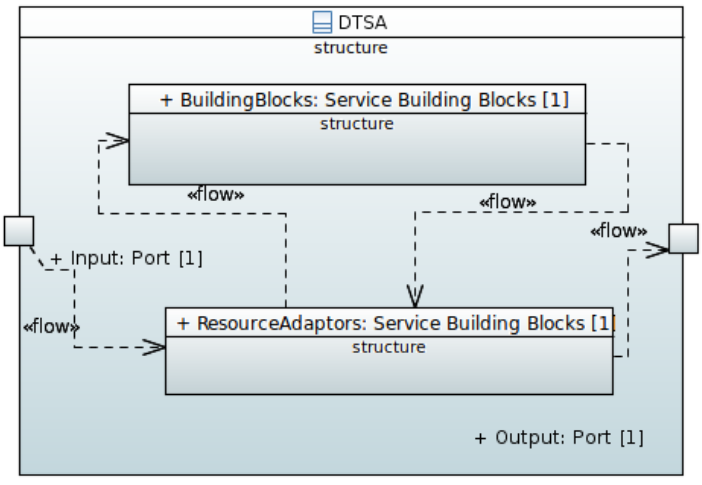

Figura 6.7: Modelo DTSA - UML

- constraint1 = { (t 0[i+1] - t0[i]) > (10 0 , ms) }

- constraint2 = { (t 3 w hen data< 5 .0) < t 2+(30, ms) }

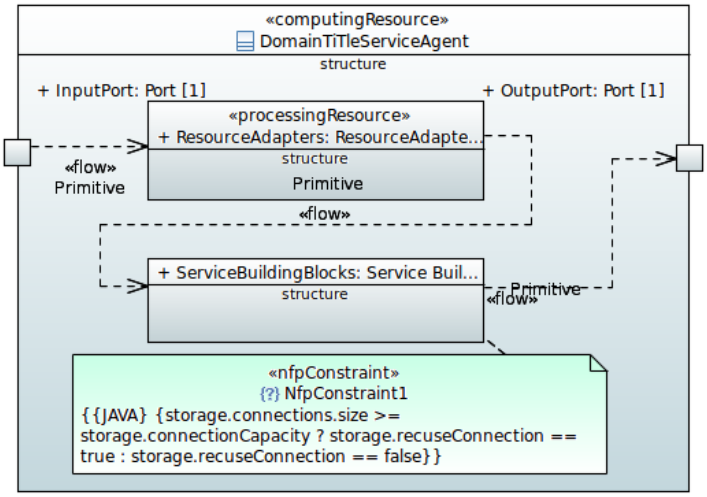

Figura 6.8: Modelo DTSA - MARTE/UML

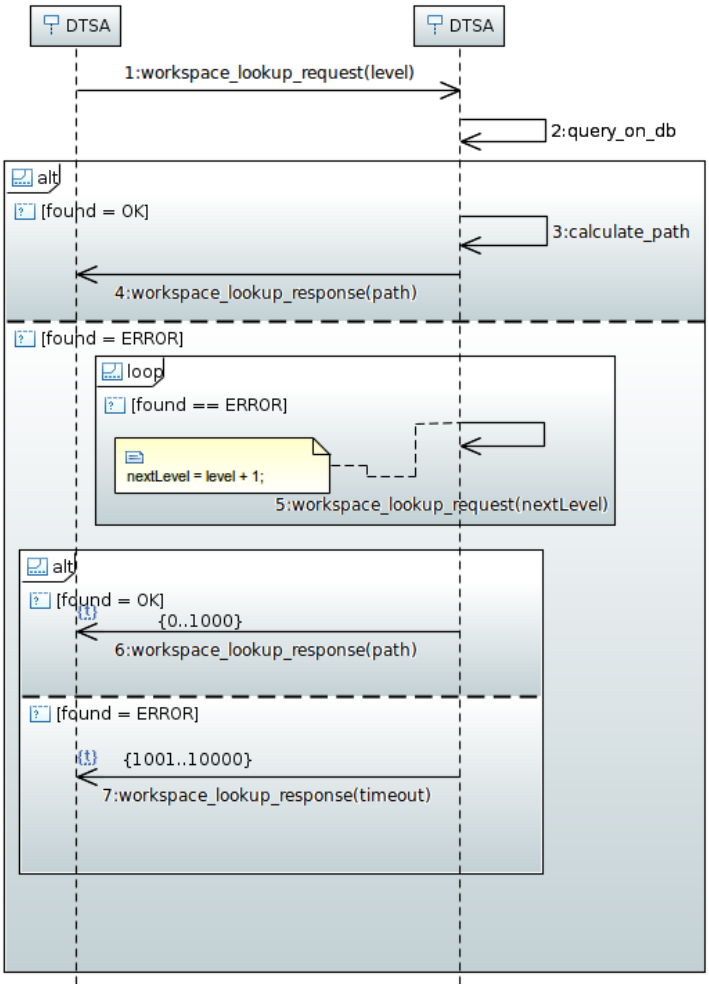

Figura 6.9: Diagrama de Sequência Workspace Lookup - UML

```
- procUtiliz   >  (90 , percent ) ?
   clockFreq == (60 , MHz) : clockFreq ==( 20 , M H z)
```

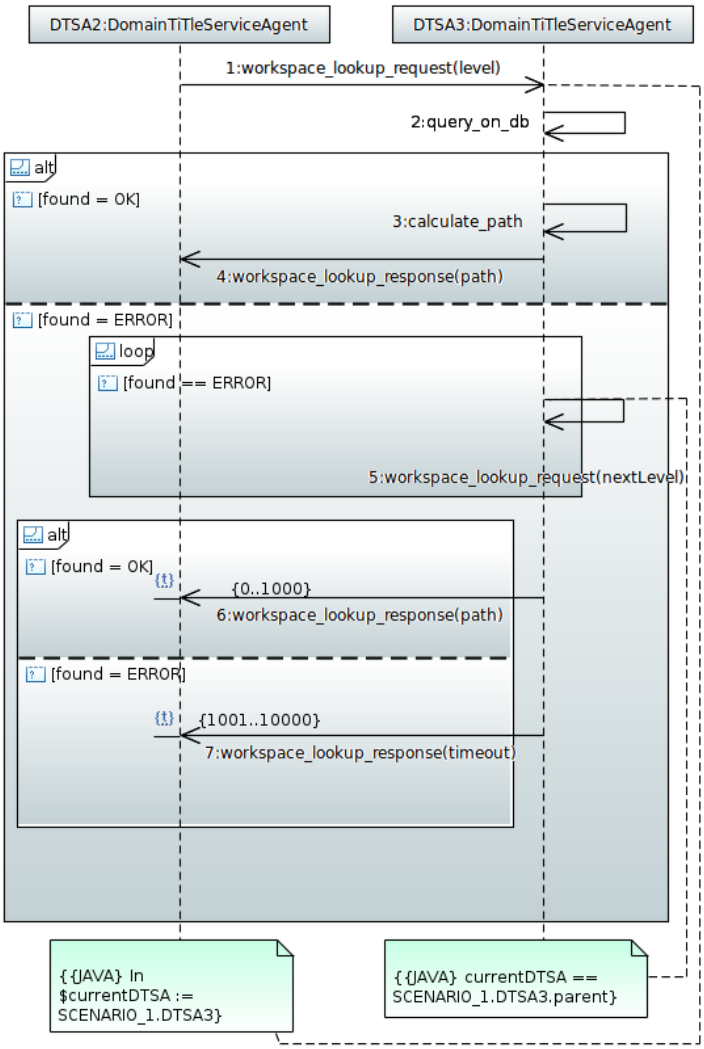

Figura 6.10: Diagrama de Sequência Workspace Lookup - MARTE/UML

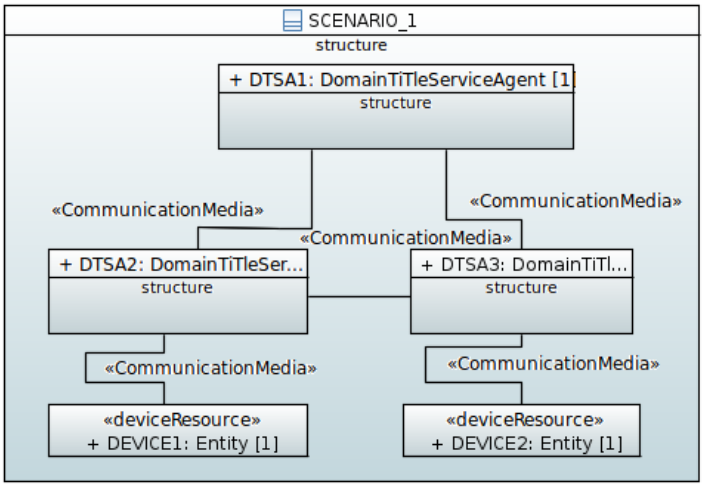

Figura 6.11: Cenário de Rede - MARTE/UML

6.2 Resultados do Questionário

O questionário criado utilizando a metodologia TAM possui 14 questões. A quantidade de questões foi sumarizada de forma que o ato de responder as questões não se tornasse oneroso para o indivíduo de interesse. Além disso, seria necessário realizar uma entrevista com os indivíduos que responderam o questionário, como não seria possível reunir com cada um deles, uma nova questão "comentários" foi adicionada no questionário. Desta forma, o usuário teve a liberdade de criticar, elogiar e sugerir modificações no trabalho.

As questões possuem cinco respostas possíveis, segundo escala Likert, de 1 a 5. Com os seguintes significados:

- 1 - Discordo totalmente;
- 2 - Discordo;
- 3 - Neutro;
- 4 - Concordo;
- 5 - Concordo totalmente.

A escolha dos indivíduos para realizar a pesquisa foi feita levando em consideração os parâmetros: nível acadêmico e experiência profissional na indústria de software. Foram escolhidos indivíduos com experiência na indústria de software e com os respectivos níveis acadêmicos: cursando graduação, cursando mestrado e mestrado completo. Ao final, foram enviados um total de 14 questionários, destes, com 11 retornando respostas.

As tabelas de resultado possuem nove colunas, sendo a primeira de identificação da linguagem, as próximas cinco representando os valores das respostas, a coluna com o identificador "m" significa a média aritmética dos resultados para a respectiva questão, o identificador "sd" consiste no desvio padrão da amostra e o identificador "pos" representa o número de resultados positivos da questão. Neste caso, os resultados com valores 4 ou 5.

A Tabela 6.1 apresenta o resultado das questões relacionadas com a usabilidade da tecnologia. Neste ponto, a tecnologia de interesse é comparada com outras. A média aritmética de cada uma das questões está acima do neutro, ou seja, um resultado positivo. Além disso, o número de resultados positivos está sempre acima da metade. Desta forma, estas questões forneceram indícios que, de acordo com os indivíduos da pesquisa, o *profile* MARTE é mais representativo que as outras tecnologias apresentadas.

A Tabela 6.2 tem como objetivo sumarizar a pesquisa quanto a facilidade de uso do *profile* MARTE. Neste caso, as médias aritméticas se aproximam de um valor que representa neutro. Para este resultado cria-se uma hipótese, que consiste no fato dos

Tabela 6.1: Resultado da Avaliação de Usabilidade

Questão	1	2	3	4	5	m	sd	pos
1 - A funcionalidade dos serviços do protocolo modeladas utilizando UML/MARTE são mais representativas do que as representadas pelos Automatos.	0	1	3	5	2	3,72	0,90	7
2- Os diagramas que utilizam a semântica do profile MARTE são mais representativos que diagramas que utilizam UML em sua forma básica.	0	0	0	8	3	4,27	0,46	11
3 - Estereótipos que possuem e permitem atribuir valor para atributos, como *communicationMedia*, agregam propriedades relevantes para o modelo.	0	0	2	4	5	4,27	0,78	9
4 - Estereótipos que definem apenas semântica e não possuem atributos, como *device* agregam informações relevantes para o modelo.	0	0	3	6	2	3,90	0,70	8
5 - Anotações com validação sintática, definidas na linguagem VSL, são mais relevantes que as anotações comuns da UML.	0	1	3	3	4	3,90	1,04	7
6 - As validações feitas pelas expressões na linguagem VSL adicionam semântica relevante ao modelo.	0	0	4	5	2	3,81	0,75	7

Tabela 6.2: Resultado da Avaliação de Facilidade de Uso

Questão	1	2	3	4	5	m	s	pos
7 - A utilização de estereótipos e definição de valores para os estereótipos é facil de usar.	0	2	5	4	0	3,18	0,75	4
8 - As expressões VSL são fáceis de usar.	0	3	5	2	1	3,09	0,94	3
9 - Os cenários definidos são fáceis de reproduzir.	0	3	4	4	0	3,09	0,83	4
10 - O relacionamento entre vários níveis de abstração, como acontece entre cenários, estrutura de um DTSA e diagrama de sequência é de simples implementação.	0	1	4	6	0	3,45	0,68	6
11 - A estrutura do DTSA modelada utilizando MARTE é de facil entendimento.	0	0	2	6	3	4,09	0,70	9

indivíduos possuírem um conhecimento prévio da linguagem UML. O entendimento dos modelos fica mais simples, porém quando são utilizados recursos MARTE, a curva de aprendizado é maior.

A maior crítica quanto a facilidade de uso da tecnologia pode ser resolvida com a aplicação de um treinamento prévio no contexto em que a abordagem de modelagem será aplicada.

A Tabela 6.3 apresenta os resultados da pesquisa sobre o interesse na utilização do *profile* MARTE na indústria de software. Todos os resultados foram, em média, positivos.

Tabela 6.3: Resultado da Avaliação de Interesse na Aplicação

Questão	1	2	3	4	5	m	s	pos
12 - Eu usaria o profile MARTE em um projeto real.	0	1	3	6	0	3,63	0,80	6
13 - A relação de custo benefício entre aprendizado e uso do profile MARTE em um projeto real é lucrativa.	1	0	7	3	0	3,3	0,48	3
14 - Eu estou interessado em saber saber mais sobre o profile MARTE.	0	0	4	4	3	3,90	0,83	7

6.3 Avaliação Qualitativa

Os comentários realizados no questionário são discutidos com o objetivo de confirmar a hipótese de que a utilização do *profile* MARTE na modelagem de protocolos possui mais pontos positivos que negativos.

- "*Acredito que os modelos produzidos utilizando o profile MARTE ficaram de fácil entendimento para usuários que conhecem UML e conhecem o uso de Estereótipos. Como sugestão, avaliar o custo benefício de aprender sobre UML e/ou profile MARTE a usar a notação de automatos para a representação. As expressões na linguagem VSL são complicadas a primeiro momento (talvez de difícil aprendizado?), mas após seu entendimento agrega valores importantes ao modelo.*"

  Este comentário reforça a escolha da linguagem base para a abordagem, pois os modelos criados a partir de *profiles* UML serão de mais fácil compreensão que modelos criados em uma nova linguagem de modelagem.

- "*O uso dos estereótipos de fato agregam valor para o entendimento do diagrama. Achei interessante a possibilidade de realizar uma análise de desempenho do software através do Profile Marte.*"

- " *Pontos negativos: As figuras 9 e 10 não foram tão ilustrativas (ou tão óbvias) quanto as outras. Talvez o guia devesse explicar com mais detalhes o exemplo. Pontos positivos: Os esteriótipos e suas respectivas descrições foram efetivas para o entendimento dos exemplos com a aplicação dos mesmos. As restrições em VSL são de fácil entendimento. (Talvez seja importante saber que eu tenho um conhecimento prévio de OCL).*"

- "*Tenho pouca experiência com modelagens em geral, devido a isso, é possível que tenha tido uma visão errada de alguns pontos, e outros tinha muito pouca informação para responder algo diferente de "Neutro".*

- "*Ponto negativo: ao comparar os diagramas de sequência/classe (sem e com uso do Marte) com o autômato, pude perceber que o autômato dá um ideia melhor sobre os estados que o sistema fica. Talvez isso ocorra pq existem outros diagramas da*

UML direcionados retratar o estado (como o Diagrama de Estado) e nos exemplos apresentados no PDF o Marte foi usado apenas em diagramas de classe e sequência. Outro ponto negativo, é que grande parte dos desenvolvedores não sabem nada ou sabem apenas um pouco de UML, o que pode ser um empecilho para que eles usem Marte + UML. Pontos positivos: Achei interessante o uso dos esteriótipos dos modelos UML, pois é uma forma de incrementar a UML padrão, colocando informações adicionais que a mesma não proporciona. O uso da linaguagem VSL também é bem interessante e com certeza os diagramas que possuem expressões em VSL ajudam o desenvolvedor a implementar o código-fonte correspondente ao modelo, pois a sintaxe da VSL parece muito com a da uma linguagem de programação genérica como C++. Como trabalho futuro sugiro explorar o profile Marte nos outros diagramas da UML (além de classe e sequência). "

Este comentário mostra que um desenvolvedor familiarizado com linguagens de programação e UML é capaz de entender o modelo criado. Além disso, é apresentada uma sugestão para um trabalho futuro, a utilização de Diagramas de Estado ou a sincronização entre outros diagramas já existentes na abordagem.

- "*Muito interessante a possibilidade de se modelar hardware mais o vsl achei de difícil compreensão*"
- "*Pontos positivos: utilização do DeviceResource, o que para o mundo de redes e até mesmo de telecomunicações pode ser bem aproveitado. Pontos negativos: ainda falta algo que de fato valide o protocolo, MARTE ainda parece apenas modelar.*"
- "*O Profile MARTE é uma ferramenta de grande utilidade para a industria de software e hardware.*"
- "*Em relação a representatividade, o profile MARTE realmente amplia o horizonte na modelagem, ou seja, você consegue representar mais detalhes na modelagem, facilitando o entendimento da mesma. Porém, precisa de mais tempo de estudo para entender completamente a mesma, em comparação com os modelos mais simples como o UML. Estudos futuros poderiam focar na lucratividade mencionada na pergunta 13, estudos empiricos verificando o uso do profile MARTE e a relação com a facilidade e necessidade de uso, e lucratividade ,se foi útil ou não nesses 3 fatores.*"
- "*Da mesma forma que a UML não é semanticamente completa, ela não obriga o desenvolvedor a ter profundo conhecimento da linguagem para um rápido entendimento. Marte parece ser mais completo semanticamente, mas o fato de ser mais completo e com semântica mais forte tráz também maior complexidade no entendimento e utilização em larga escala, fato que, na minha opinião, ajudou a UML a ser largamente utilizada no mundo, ou seja, por sua simplicidade.*"
- "*A utilização dos estereótipos adiciona semântica relevante a um software ou sis-*

tema."

6.4 Considerações Finais

Neste capítulo foi apresentada a avaliação do trabalho realizado, com o objetivo de obter uma posição dos indivíduos que serão público alvo da aplicação da abordagem em modelagem de protocolos de comunicação de tempo real. Os resultados dos questionários apresentaram sempre uma média considerada positiva, pois é maior que o valor neutro para a questão.

Uma crítica ao trabalho está relacionada com a validação do modelo escrito utilizando MARTE, porém métodos de transformação de modelos podem ser utilizados para a transformação de elementos das abordagens em elementos de linguagens validáveis, como Redes de Petri e PROMELA. O caminho comum, muitas vezes, para a criação de uma abordagem, seria focar no problema. Mas, pelo que foi observado em várias outras linguagens de modelagem, o foco na solução pode tornar a linguagem não agradável para o usuário. Neste trabalho, o foco está em avaliar as abordagens quanto a um modelo de aceitação, antes que um plano de validação seja desenvolvido.

A questão 14 pode influenciar em outros resultados, pois, com maior conhecimento do *profile*, as possibilidades de uso aumentam, tornando a relação de custo benefício mais favorável à utilização do *profile*.

Os resultados do questionáio fornecem indícios de que o uso das abordagens são relevantes para a modelagem de protocolos de comunicação. Entretanto, existem ameaças à validade da pesquisa. Os indivíduos que responderam o questionário podem, por coincidência, possuir preferências por uma linguagem de modelagem específica. Para evitar esta ameaça, a escolha dos indivíduos seguiu uma aleatorieadade de empresas trabalhadas, projetos acadêmicos e idades. A pesquisa pode ser alterada caso indivíduos não tenham conhecimento de modelagem ou protocolos de comunicação. Para isto, além de uma pesquisa prévia sobre o conhecimento do indivíduo, foi apresentado um guia para o mesmo, a fim de realizar um pequeno treinamento prévio.

Capítulo 7

Conclusão e Trabalhos Futuros

Para que o presente trabalho fosse realizado, foram necessárias várias etapas, descritas a cada capítulo. O Capítulo 1 apresentou as questões a serem respondidas, uma introdução ao contexto abordado e as motivações para o trabalho. O primeiro passo para solucionar as questões do Capítulo 1 consiste em realizar uma pesquisa e gerar um referencial teórico para o trabalho, isto foi realizado no Capítulo 2. Para aplicar a abordagem proposta no trabalho, uma arquitetura de Internet do Futuro, ETArch, foi utilizada, a mesma foi definida no Capítulo 3. O Capítulo 4 apresenta a abordagem de utilização da linguagem UML para a modelagem dos serviços dos protocolos de comunicação da ETArch. O Capítulo 5 complementa a abordagem do Capítulo 4 usando o *profile* MARTE. No Capítulo 6 a avaliação das abordagens em relação aos usuário é feita.

O presente capítulo tem como objetivo apresentar as conclusões obtidas a partir do trabalho, bem como apresentar as ações que darão continuidade ao mesmo, sendo dividido em seis seções, a presente seção tem como objetivo introduzir o tema e apresentar o conteúdo das próximas seções.

A Seção 7.1 apresenta a conclusão a respeito das abordagens de modelagem criadas no trabalho, listando vantagens e limitações das mesmas.

A Seção 7.2 sumariza os resultados obtidos no questionário que é apresentado no Capítulo 6.

A Seção 7.3 apresenta a respostas das questões de pesquisa feitas no Capítulo 1.

A Seção 7.4 apresenta o resultado de submissão de artigo acadêmico realizado no trabalho.

A Seção 7.5 apresenta os trabalhos futuros que darão continuidade ao trabalho.

7.1 Abordagens de Modelagem

Ao final do trabalho foram criadas duas abordagens de modelagem para protocolos de comunicação. A primeira utilizando apenas a linguagem UML e a segunda utilizando o *profile* MARTE.

A primeira abordagem, de uso da linguagem UML, conseguiu representar os requisitos necessários da arquitetura ETArch, como a estrutura do DTSA e os serviços dos protocolos. Esta abordagem possui diversas vantagens quando comparadas a outras, como Autômatos. A linguagem UML possui uma representação nativa para orientação a objetos, isto é de muita importância, pois a implementação dos sistemas de software presentes nos equipamentos utilizam o mesmo paradigma de programação. O Diagrama de Estrutura Composta permite diversos níveis de abstração ilimitados, tornando possível a modelagem de estruturas complexas definindo cada módulo da mesma separadamente. Além disso, o Diagrama de Sequência permite representar o comportamento entre os elementos, desta forma, é possível especificar os serviços do protocolo.

Entretanto, a primeira abordagem não se mostrou suficiente para representar as propriedades de tempo real, como consumo de recursos. As restrições adicionadas ao modelo, em forma de comentários, são apenas informativas, pois, como não possuem nenhuma definição sintática, as mesmas podem conter ambiguidades, não sendo úteis para a especificação.

A segunda abordagem, de uso do *profile* MARTE, conseguiu solucionar os problemas apresentados na primeira. Utilizando os estereótipos MARTE foi possível definir elementos capazes de representar recursos, que podem ser controlados usando restrições. As restrições do modelo passaram a ser feitas em uma linguagem de restrições, chamada VSL. Como a mesma possui uma sintáxe definida, é possível criar especificações sem ambiguidades, desta forma, as restrições passam a ter grande utilização para a definição da semântica do modelo. Uma grande vantagem na utilização desta abordagem está na divisão de necessidades, ou seja, as partes do sistema que necessitam de um nível de abstração mais alto podem ser modeladas utilizando Diagramas de Classe ou Estrutura Composta, as partes que necessitam de expressões algébricas para a sua definição podem ser especificadas utilizando a linguagem VSL.

Inicialmente, o foco da segunda abordagem é definir os estereótipos que permitem a especificação dos elementos relevantes da arquitetura ETArch. Após essa definição, é possível pensar em uma formalização dos elementos selecionados, bem como a integração entre elementos, diagramas e restrições, formalmente.

Desta forma, um ponto negativo em ambas as abordagens é a falta de um método definido para a formalização das mesmas, permitindo uma validação automatizada. Algumas abordagens de transformação de modelos se mostraram capazes de permitir tal formalização.

Ambas as abordagens possuem uma grande vantagem quanto a utilização, pois a ferramenta de modelagem Papyrus que permite a modelagem tanto em UML quanto em MARTE é de código aberto, podendo ser evoluída pela comunidade.

7.2 Resultados do Questionário

As respostas dos questionários contribuíram para a formação de um caminho a ser seguido para os trabalhos futuros. As respostas foram dentro das expectativas. Como as respostas obtiveram uma média positiva, pode-se considerar que o trabalho está seguindo o caminho correto.

A partir do questionário foi possível filtrar diversos indícios. Os indivíduos que já utilizam a linguagem UML terão a curva de aprendizado menor, tornando-se fluentes na abordagem. Como as restrições escritas na linguagem VSL são semelhantes as linguagens de programação C ou Java, os indivíduos que possuírem o conhecimento básico em uma delas conseguirão um entendimento suficiente do modelo para aplicar na implementação.

A comparação entre os modelos criados utilizando a linguagem UML e os Autômatos foi favorável à linguagem UML. Desta forma, mesmo que seja necessário utilizar Autômatos para a modelagem, a abordagem de transformação automática de modelos deve ser pensada.

Portanto, usando o questionário, foi possível concluir que tanto indivíduos que lidam com protocolos de comunicação quanto linguagem de modelagem consideraram a abordagem relevante.

Os comentários feitos pelos indivíduos que responderam o questionário permitiram obter uma visão mais ampla da percepção dos usuários. Além disso, os comentários de sugestão são muito úteis, pois mostram o caminho que os mesmos desejam que a abordagem continue.

7.3 Respostas às Questões de Pesquisa

No Capítulo 1 foram propostas diversas questões, na presente seção as questões serão respondidas. As seguintes questões foram apresentadas:

- *Q1 - Quais são as principais abordagens de modelagem aplicadas em protocolos de comunicação?*

  De acordo com o processo de pesquisa apresentado no Capítulo 1, as abordagens mais relevantes aplicadas na modelagem de protocolos de comunicação são: *Specification and Description Language* (SDL), *Language Of Temporal Ordering Specification* (LOTOS), *Extended State Transition Language* (Estelle), *Finite-State Machines* (FSM), *Redes de Petri e Protocol Meta Language* (PROMELA).

- *Q2 - As linguagens predominantes são visuais ou algébricas?*

  No referencial teórico houve uma predominância de linguagens visuais, porém, com validação matemática, como Redes de Petri. As linguagens puramente algébricas são apenas PROMELA e Estelle. A abordagem do *profile* MARTE possui tanto a

abordagem visual, com os elementos UML e estereótipos, quanto abordagem algébrica, pela utilização da linguagem VSL.

- *Q3 - Como representar propriedades não funcionais nas linguagens de modelagem?*

  Na abordagem de utilização do *profile* MARTE as propriedades não funcionais são representadas por expressões na linguagem VSL e pelas definições presentes no pacote NFP, como *NFP_Annotation* para definir as anotações criadas nos modelos.

- *Q4 - Como modelar aspectos temporais de protocolos de comunicação?*

  A modelagem dos aspectos temporais pode ser feita de duas formas, na primeira abordagem usando os observadores de tempo da linguagem UML, na segunda abordagem usando os observadores juntamente com a linguagem VSL, para criar restrições temporais baseadas em expressões.

- *Q5 - Como garantir a aceitação da abordagem apresentada por parte do usuário?*

  A aceitação por parte do usuário obteve indícios positivos de acordo com o modelo TAM, definido no Capítulo 6, que realiza um questionário baseado nas abordagens do trabalho.

Ao longo do trabalho é possível obter todas as respostas para as questões apresentadas, as mesmas foram respondidas de forma breve e objetiva, focando na organização do texto.

7.4 Artigo Aceito

A abordagem apresentada no Capítulo 4 [Alves da Silva et al. 2014] foi submetida na conferência AICT [Iaria 2014] com o intuito de ser validada. A mesma possui qualificação nível B1 de acordo com o nível de qualificação definido pela CAPES, o Qualis [Capes 2014]. O trabalho foi aceito e apresentado na conferência.

7.5 Trabalhos Futuros

O presente trabalho consiste apenas em uma de várias etapas necessárias para a criação de uma abordagem de validação de uma especificação de protocolos de comunicação de tempo real. Desta forma, foram criados planos para a continuação da criação da abordagem. Esta seção apresenta os próximos passos da pesquisa:

- Submissão da abordagem de utilização do *profile* MARTE para uma conferência, a publicação está escrita e em fase de refinamento;
- Análise de novos diagramas UML para a aplicação na modelagem de protocolos de comunicação, com foco no Diagrama de Estados;

- Levantamento e posterior aplicação das abordagens de validação de modelos, para que as mesmas sejam aplicadas nos elementos presentes na abordagem criada;
- Melhoria da ferramenta de modelagem Papyrus de forma que suporte as validações dos modelos gerados e inserção da ferramenta de validação VSL na ferramenta Papyrus;
- Elaboração de novo questionário para validação e apresentação da abordagem completa a pesquisadores e alunos de pós-graduação.

Após a realização de todas as etapas descritas anteriormente, é esperado que o trabalho possa ser concluído em sua totalidade e uma nova abordagem de modelagem e validação de protocolos de comunicação de tempo real seja concluída, e que a mesma se torne um padrão para a modelagem de protocolos de comunicação em todos os aspectos necessários.

Referências Bibliográficas

[Al-Bataineh et al. 2012] Al-Bataineh, O., French, T., e Woodings, T. (2012). Formal Modeling and Analysis of a Distributed Transaction Protocol in UPPAAL. In *19th International Symposium on Temporal Representation and Reasoning (TIME)*, pp. 65–72.

[Alur e Dill 1994] Alur, R. e Dill, D. L. (1994). A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235.

[Alves da Silva et al. 2014] Alves da Silva, D., de Oliveira Silva, F., Vieira de Souza Neto, N., e Rosa, P. (2014). UML-based Modeling Entity Title Architecture (ETArch) Protocols. In *IARIA Advanced International Conference on Telecommunications - AICT*.

[Anisimov e Koutny 1995] Anisimov, N. A. e Koutny, M. (1995). On Compositionality and Petri nets in Protocol Engineering. In Dembinski, P. e Sredniawa, M. (editores), *PSTV*, volume 38 de *IFIP Conference Proceedings*, pp. 71–86. Chapman & Hall.

[Bagnato et al. 2013] Bagnato, A., Sadovykh, A., Brosse, E., e Vos, T. E. (2013). The OMG UML Testing Profile in Use–An Industrial Case Study for the Future Internet Testing. *15th European Conference on Software Maintenance and Reengineering*, 1:457–460.

[Bai et al. 2011] Bai, Y., Ye, X., e Ma, Y. (2011). Formal Modeling and Analysis of SIP Using Colored Petri Nets. In *7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2011*, pp. 1–5.

[Bari et al. 2013] Bari, N., Mani, G., e Berkovich, S. (2013). Internet of Things as a Methodological Concept. In *Computing for Geospatial Research and Application (COM.Geo), 2013 Fourth International Conference on*, pp. 48–55.

[Belategi et al. 2010] Belategi, L., Sagardui, G., e Etxeberria, L. (2010). MARTE Mechanisms to Model Variability When Analyzing Embedded Software Product Lines. In *14th Proceedings of the International Conference on Software Product Lines*, pp. 466–470.

[Belina e Hogrefe 1989] Belina, F. e Hogrefe, D. (1989). The CCITT-specification and Description Language SDL. *Comput. Netw. ISDN Syst.*, 16(4):311–341.

[Bennett e Field 2004] Bennett, A. J. e Field, A. J. (2004). Performance Engineering with the UML Profile for Schedulability, Performance and Time: a Case Study. In *12th Annual International Symposium on the IEEE Computer Society's*, pp. 67–75.

[Bolognesi e Brinksma 1987] Bolognesi, T. e Brinksma, E. (1987). Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59.

[Booch et al. 2005] Booch, G., Rumbaugh, J., e Jacobson, I. (2005). *The Unified Modeling Language User Guide, (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional.

[Borrione et al. 2007] Borrione, D., Helmy, A., Pierre, L., e Schmaltz, J. (2007). A Generic Model for Formally Verifying NoC Communication Architectures: A Case Study. In *First International Symposium on Networks-on-Chip, 2007. NOCS 2007.*, pp. 127–136.

[Bucci e Vicario 1995] Bucci, G. e Vicario, E. (1995). Compositional Validation of Time-critical Systems Using Communicating Time Petri Nets. *Software Engineering, IEEE Transactions on*, 21(12):969–992.

[Budkowski et al. 1989] Budkowski, S., Dembinski, P., e Diaz, M. (1989). Information Processing Systems - Open Systems Interconnection - Estelle: A Formal Description Technique Based on an Extended State Transition Model.

[Capes 2014] Capes (2014). http://qualis.capes.gov.br.

[Chen et al. 2010] Chen, H., Zhou, C., Ma, J., e Qing, Y. (2010). FSM Model and Analysis for Reconfigurable Protocol Stack in Networked Control System. In *8th IEEE International Conference on Control and Automation (ICCA), 2010*, pp. 779–784.

[Chunmei et al. 2010] Chunmei, H., Yulan, Z., Xiaoqing, G., Chenguang, Z., e Rengaowa, S. (2010). The Formal Description of the Protocol and the Design and Implementation of Automatically Generating Protocol Test Suite. In *International Conference on Computational Intelligence and Software Engineering (CiSE), 2010*, pp. 1–4.

[Cuccuru1 2011] Cuccuru1, A. (2011). Methodological Guidelines on the Usage of MARTE VSL for Specification of Time Constraints. In *2nd Workshop on Model Based Engineering for Embedded Systems Design.*

[Davis 1989] Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Q.*, 13(3):319–340.

[Day e Zimmermann 1983] Day, J. e Zimmermann, H. (1983). The OSI Reference Model. *Proceedings of the IEEE*, 71(12):1334–1340.

[de Meer et al. 1992] de Meer, J., Roth, R., e Vuong, S. (1992). Introduction to Algebraic Specifications Based on the Language ACT ONE. *Computer Networks and ISDN Systems*, 23(5):363–392.

[de Oliveira Silva 2013] de Oliveira Silva, F. (2013). *Endereçamento por Título: Uma Forma de Encaminhamento Multicast Para a Próxima Geração de Redes de Computadores.* PhD thesis, Universidade de São Paulo.

[de Oliveira Silva et al. 2011] de Oliveira Silva, F., Goncalves, M., de Souza Pereira, J., Pasquini, R., Rosa, P., e Kofuji, S. (2011). Domain Title Service for Future Internet Networks. In *Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF), SBRC 2011.*

[de Oliveira Silva et al. 2012] de Oliveira Silva, F., Goncalves, M., de Souza Pereira, J., Pasquini, R., Rosa, P., e Kofuji, S. (2012). On the Analysis of Multicast Traffic Over the Entity Title Architecture. In *18th IEEE International Conference on Networks (ICON)*, pp. 30–35.

[Demathieu et al. 2008] Demathieu, S., Thomas, F., Andre, C., Gerard, S., e Terrier, T. (2008). First Experiments Using the UML Profile for MARTE. In *11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp. 50–57. IEEE Computer Society.

[Dubois et al. 2009] Dubois, H., Lakhal, F., e Gerard, S. (2009). The Papyrus Tool as an Eclipse UML2-modeling Environment for Requirements. In *Second International Workshop on Managing Requirements Knowledge (MARK), 2009*, pp. 85–88.

[El-Fakih et al. 2000] El-Fakih, K., Yamaguchi, H., v. Bochmann, G., e Higashino, T. (2000). Protocol Re-synthesis Based on Extended Petri Nets.

[Fraser e Mancl 2008] Fraser, S. e Mancl, D. (2008). No Silver Bullet: Software Engineering Reloaded. *Software, IEEE*, 25(1):91–94.

[Ganea et al. 2012] Ganea, O., Pop, F., Dobre, C., e Cristea, V. (2012). Specification and Validation of a Real-Time Simple Parallel Kernel for Dependable Distributed Systems. In *Third International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), 2012*, pp. 320–325.

[Graf et al. 2006] Graf, S., Gérard, S., Haugen, O., Ober, I., e Selic, B. (2006). Modeling and Analysis of Real-time and Embedded Systems. In *Proceedings of the 2005 International Conference on Satellite Events at the Models*, MoDELS'05, pp. 58–66.

[Griffith et al. 2010] Griffith, D., Rouil, R., e Golmie, N. (2010). Performance Metrics for IEEE 802.21 Media Independent Handover (MIH) Signaling. *Wireless Personal Communications*, 52(3):537–567.

[Grønmo e Møller-Pedersen 2010] Grønmo, R. e Møller-Pedersen, B. (2010). From Sequence Diagrams to State Machines by Graph Transformation. In *Proceedings of the Third International Conference on Theory and Practice of Model Transformations*, ICMT'10, pp. 93–107, Málaga, Spain, Berlin, Heidelberg. Springer-Verlag.

[Group 2007] Group, O. M. (2007). OMG Unified Modeling Language (OMG UML), Infrastructure, V2.5. Technical report, Group, Object M.

[Gurp e Bosch 1999] Gurp, J. V. e Bosch, J. (1999). On the Implementation of Finite State Machines. In *In Proceedings of the 3rd Annual IASTED International Conference Software Engineering and Applications, IASTED/Acta*, pp. 172–178. Press.

[He et al. 2009] He, S., Zhang, D., Wang, Y., e Guo, D. (2009). Comparing of Reusable IP Design Method with RL and FSM. In *3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, 2009.*, pp. 355–358.

[Holzmann 1991a] Holzmann, G. J. (1991a). *Design and Validation of Computer Protocols*. Prentice-Hall, Inc.

[Holzmann 1991b] Holzmann, G. J. (1991b). *Design and Validation of Computer Protocols*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Hua et al. 2012] Hua, W., Li, X., Guan, Y., Shi, Z., Dong, L., e Zhang, J. (2012). Formal Verification for SpaceWire Communication Protocol Based on Environment State Machine. In *8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM), 2012.*, pp. 1–4.

[Huang e Li 2010] Huang, Y. e Li, G. (2010). Descriptive models for Internet of Things. In *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on*, pp. 483–486.

[Iaria 2014] Iaria (2014). http://www.iaria.org/conferences2014/AICT14.html.

[Iqbal et al. 2012] Iqbal, M., Ali, S., Yue, T., e Briand, L. (2012). Experiences of Applying UML/MARTE on Three Industrial Projects. In *Model Driven Engineering Languages and Systems*, pp. 642–658. Springer Berlin Heidelberg.

[IS8807 1988] IS8807 (1988). ISO- Information Processing Systems - Open Systems Interconnection- LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behavior.

[Jaragh e Saleh 2001] Jaragh, M. e Saleh, I. (2001). Protocols Modeling using the Unified Modeling Language. In *Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology*, volume 1, pp. 69–73.

[Kant et al. 1996] Kant, C., Higashino, T., e Bochmann, G. V. (1996). Deriving Protocol Specifications from Service Specifications Written in LOTOS. *Distributed Computing*, 10(1):29–47.

[Kapus-Kolar 1999] Kapus-Kolar, M. (1999). Comments on Deriving Protocol Specifications from Service Specifications Written in LOTOS. *Distributed Computing*, 12:175–177.

[Khendek et al. 1989] Khendek, F., von Bochmann, G., e Kant, C. (1989). New Results on Deriving Protocol Specifications from Service Specifications. *SIGCOMM Computer Communication Review*, 19(4):136–145.

[Kim e Feamster 2013] Kim, H. e Feamster, N. (2013). Improving Network Management with Software Defined Networking. *IEEE Communications Magazine*, 51(2):114–119.

[Koifman e Zabele 1996] Koifman, A. e Zabele, S. (1996). RAMP: A Reliable Adaptive Multicast Protocol. In *Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, INFOCOM 1996.*, volume 3, pp. 1442–1451.

[Komu et al. 2012] Komu, B., Mzyece, M., e Djouani, K. (2012). SPIN-Based Verification of Authentication Protocols in WiMAX Networks. In *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, pp. 1–5.

[Lai e Tsang 2007] Lai, R. e Tsang, T. (2007). Timed Verification of the Reliable Adaptive Multicast Protocol. *Journal of Systems and Software*, 80:224 – 239.

[Lakos et al. 1995] Lakos, C., Lamp, J., Keen, C., e Marriott, B. (1995). Modelling Network Protocols with Object Petri Nets. In *Proceedings of Workshop on Petri Nets Applied to Protocols*, pp. 31–42. Springer-Verlag.

[Lange e Chaudron 2004] Lange, C. e Chaudron, M. (2004). An Empirical Assessment of Completeness in UML Designs. In *Proceedings Conference of Empirical Assessment in Software Engineering*, pp. 111–121.

[Lee e Baik 2008] Lee, D. e Baik, J. (2008). QoS Protocol Verification using Petri-Net for Seamless Mobility in a Ubiquitous Environment: A Case Study. In *International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS*, pp. 617–622.

[Lei et al. 2007] Lei, X., Kim, Y. S., e Lai, R. (2007). Modelling and Simulating IPv6 Mobility. In *7th IEEE International Conference on Computer and Information Technology, 2007. CIT 2007.*

[Li et al. 2005] Li, D., Cui, Y., Xu, K., e Wu, J. (2005). Improvement of Multicast Routing Protocol Using Petri Nets. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, volume 3642 de *Lecture Notes in Computer Science*, pp. 634–643. Springer Berlin Heidelberg.

[Li et al. 2011] Li, Z., Ma, D., Zhao, Y., Li, J., e Yang, Q. (2011). FSM4WSR: A Formal Model for Verifiable Web Service Runtime. In *Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific.*

[Limal et al. 2007] Limal, S., Potier, S., Denis, B., e Lesage, J. (2007). Formal Verification of Redundant Media Extension of Ethernet PowerLink. In *IEEE Conference on Emerging Technologies and Factory Automation*, pp. 1045–1052.

[Ludewig 2003] Ludewig, J. (2003). Models in Software Engineering - An introduction. *Software and Systems Modeling*, 2:5–14.

[Martin e Westall 2006] Martin, J. e Westall, J. (2006). Evaluating IEEE 802.16 Broadband Wireless as a Communications Infrastructure for Public Safety Activities.

[Masri et al. 2008] Masri, A., Bourdeaud'huy, T., e Toguyeni, A. (2008). Network Protocol Modeling: A Time Petri Net Modular Approach. In *16th International Conference on Software, Telecommunications and Computer Networks*, pp. 274–278.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., e Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *Special Interest Group on data Communications (SIGCOMM) Computer Communication Review*, 38:69–74.

[Min e Siyu 2010] Min, H. e Siyu, Z. (2010). Analysis of Cryptographic Protocol About Wireless LAN Base on Petri Net. In *International Conference on Computer Application and System Modeling (ICCASM), 2010*, volume 8, pp. V8–267–V8–269.

[Murata 1989] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

[Nogueira et al. 2010] Nogueira, A., Ramos, L., de Castro, M., e Andrade, R. (2010). Validating Mobility Management Solutions for Interworking UMTS and IEEE 802.11 Networks. In *Telecommunications (ICT), 2010 IEEE 17th International Conference on*, pp. 581–588.

[O'Hara e Petrick 1999] O'Hara, B. e Petrick, A. (1999). *The IEEE 802.11 Handbook: A Designer's Companion.* Standards Information Network IEEE Press.

[Open Networking Foundation 2012] Open Networking Foundation (2012). Software-Defined Networking: The New Norm for Networks.

[Palaniveloo e Sowmya 2011] Palaniveloo, V. e Sowmya, A. (2011). Application of Formal Methods for System-Level Verification of Network on Chip. In *Computer Society Annual Symposium on VLSI (ISVLSI), 2011 IEEE*, pp. 162–169.

[Park e Shiratori 1994] Park, S.-S. e Shiratori, N. (1994). Distributed Systems Management Based On OSI Environment: Problems, Solutions, and Their Evaluation. In *IEEE 13th Annual International Phoenix Conference on Computers and Communications*.

[Parr 2013] Parr, T. (2013). *The Definitive ANTLR 4 Reference*. The Programatic Programmers.

[Ramananandro 2008] Ramananandro, T. (2008). Mondex, An Electronic Purse: Specification and Refinement Checks with the Alloy Model-finding Method. *Formal Aspects of Computing*, 20:21–39.

[Rodrigues et al. 2011] Rodrigues, A. W. D. O., Guyomarc'h, F., e Dekeyser, J.-L. (2011). A Modeling Approach based on UML/MARTE for GPU Architecture. *The Computing Research Repository (CoRR)*, abs:1105–4424.

[Salaun et al. 2012] Salaun, G., Bultan, T., e Roohi, N. (2012). Realizability of Choreographies Using Process Algebra Encodings. *IEEE Transactions on Services Computing, 2012*, 5(3):290–304.

[Saleh 1996] Saleh, K. (1996). Synthesis of Communications Protocols: An Annotated Bibliography. *SIGCOMM Computer Communication Review*, 26:40–59.

[Samukic 1998] Samukic, A. (1998). UMTS Universal Mobile Telecommunications System: Development of Standards for the Third Generation. *Vehicular Technology, IEEE Transactions on*, 47(4):1099–1104.

[Santos et al. 2011] Santos, E. D. S., Pereira, F. S. F., de Souza Pereira, J. H., Theodoro, L. C., Rosa, P. F., e Kofuji, S. T. (2011). Meeting Services and Networks in the Future Internet. In *Future Internet Assembly*, volume 6656, pp. 339 – 350. Springer.

[Sekaran 2001] Sekaran, K. (2001). Development of a Link Layer Protocol using UML. In *International Conference on Computer Networks and Mobile Computing*, pp. 309–315.

[Selic 2003] Selic, B. (2003). The Pragmatics of Model-driven Development. *Software, IEEE*, 20(5):19–25.

[Selic e Limited 1996] Selic, B. e Limited, O. (1996). Real-Time Object-Oriented Modeling (ROOM). In *Proceedings of the 2Nd IEEE Real-Time Technology and Applications Symposium (RTAS '96)*. IEEE Computer Society.

[Shousha et al. 2012] Shousha, M., Briand, L. C., e Labiche, Y. (2012). A UML/MARTE Model Analysis Method for Uncovering Scenarios Leading to Starvation and Deadlocks in Concurrent Systems. *IEEE Transactions on Software Engineering*, 38(2):354–374.

[Simonak et al. 2009] Simonak, S., Hudak, S., e Korecko, S. (2009). Protocol Specification and Verification Using Process Algebra and Petri Nets. In *International Conference on Computational Intelligence, Modelling and Simulation, 2009. CSSim '09.*, pp. 110–114.

[Soares e Vrancken 2008] Soares, M. S. e Vrancken, J. (2008). A Metamodeling Approach to Transform UML 2.0 Sequence Diagrams to Petri Nets. In *Proceedings of the IASTED International Conference on Software Engineering*, pp. 159–164.

[Sommerville 2006] Sommerville, I. (2006). *Software Engineering: (Update) (8th Edition) (International Computer Science)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Sommerville 2010] Sommerville, I. (2010). *Software Engineering (9th Edition)*. Pearson Addison Wesley.

[Sonkoly et al. 2012] Sonkoly, B., Gulyas, A., Nemeth, F., Czentye, J., Kurucz, K., Novak, B., e Vaszkun, G. (2012). On QoS Support to Ofelia and OpenFlow. In *European Workshop on Software Defined Networking (EWSDN)*, pp. 109–113.

[Sun et al. 2010] Sun, T., Ye, X., Liu, J., e Yang, M. (2010). Formal Modeling and Analysis of HMIPv6 Using Colored Petri Nets. In *Communications and Mobile Computing (CMC), 2010 International Conference on*, volume 1, pp. 462–466.

[Venkatesh e Bala 2008] Venkatesh, V. e Bala, H. (2008). Technology Acceptance Model 3 and a Research Agenda on Interventions. *Decision Sciences*, 39:273–315.

[von Bochmann 1978] von Bochmann, G. (1978). Finite State Description of Communication Protocols. *Computer Networks*, 2:361–372.

[Weiser 1999] Weiser, M. (1999). The Computer for the 21st Century. *SIGMOBILE Mobility Computer Communication Review*, 3:3–11.

[Whittle e Schumann 2000] Whittle, J. e Schumann, J. (2000). Generating Statechart Designs from Scenarios. In *Proceedings of the 2000 International Conference on Software Engineering, 2000.*, pp. 314–323.

[Wu et al. 2009] Wu, X., Ling, H., e Dong, Y. (2009). On Modeling and Verifying of Application Protocols of TTCAN in Flight-Control System with UPPAAL. In *International Conference on Embedded Software and Systems*, pp. 572–577.

[Xu et al. 2003] Xu, J., Woodside, M., e Petriu, D. (2003). Performance Analysis of a Software Design Using the UML Profile for Schedulability, Performance and Time. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pp. 291–310.

[Zhao et al. 2012] Zhao, H., Wang, W., Sun, J., e Wei, Y. (2012). Research on Formal Modeling and Verification of BPEL-based Web Service Composition. In *11th International Conference on Computer and Information Science (ICIS), 2012 IEEE/ACIS*, pp. 631–636.