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

MODELAGEM DE REQUISITOS DE SOFTWARE DE TEMPO-REAL USANDO SYSML E MARTE

FABÍOLA GONÇALVES COELHO RIBEIRO

Uberlândia - Minas Gerais
2013

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

FABÍOLA GONÇALVES COELHO RIBEIRO

MODELAGEM DE REQUISITOS DE SOFTWARE DE TEMPO-REAL USANDO SYSML E MARTE

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

Uberlândia, Minas Gerais
2013

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 "**Modelagem de Requisitos de Software de Tempo-Real usando SysML e MARTE**" por **Fabíola Gonçalves Coelho Ribeiro** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 8 de Março de 2013

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

Banca Examinadora:

_____
Prof. Dr. Stéphane Julia
Universidade Federal de Uberlândia

_____
Prof. Dra. Itana Maria de Souza Gimenes
Universidade Estadual de Maringá

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

Data: Março de 2013

Autor:	**Fabíola Gonçalves Coelho Ribeiro**
Título:	**Modelagem de Requisitos de Software de Tempo-Real usando SysML e MARTE**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

Dedicatória

Dedico a este trabalho a minha família. Em especial, a você Maria Fernanda o presente mais lindo que Deus poderia me dar. Você, meu amor, trouxe luz para meus olhos, serenidade para minha vida e o amor que eu jamais poderia imaginar existir.

Agradecimentos

Agradecer é uma maneira de compartilhar a nossa alegria e gratidão com aqueles que tanto nos apoiaram e ficaram do nosso lado. Sendo assim, quero realizar meus mais sinceros agradecimentos:

Àquele que representa um amigo, um irmão, um sábio tutor, fonte de orientação, que traz tranquilidade e sabedoria sempre que necessário. A você, querido orientador, que acreditou em mim e com isso fez-me capaz. Obrigada pela paciência, dedicação e apoio que sempre me dedicou.

Ao Centro de Ensino Superior de Catalão, na pessoa do professor Paulo Antônio de Lima, instituição esta que está presente em minha vida de forma direta e que tem participado de minha formação acadêmica e profissional, dando-me sempre apoio e respaldo.

À Universidade Federal de Goiás e a Universidade Federal de Uberlândia, instituições que me deram oportunidades para vivenciar e aprender, no transcorrer da pesquisa, a maravilhosa arte da docência. Sendo sempre, na figura da coordenação e de seu corpo docente, acolhedora aos meus horários e restrições e ainda, assim, dando-me oportunidade de aprender e crescer.

Ao Dr. Lacordaire pelo incentivo, apoio, amizade e por tempos atrás dar-me a oportunidade de conhecer a pesquisa e docência, incentivando-me a trilhar o caminho por onde tenho andado.

À minha mãe, maravilhosa, Luzia Gonçalves, que sempre esteve ao meu lado e, mesmo que silenciosamente, na calmaria de seu olhar, deu-me a força que muitas das vezes me faltou. O seu exemplo, diário, de caráter e honestidade provê a melhor formação que qualquer indivíduo poderia ter. Obrigada por estar onde não pude estar e por suprir minhas inúmeras ausências sendo mãe e avó de minha pequena. Amo-te.

Meu querido pai e minha mãe-drasta, Eleuza, que carinhosamente me recebiam e acolhiam em seu convívio com carinho e amor. Eleuza, querida, obrigada pelos conselhos, apoio e por ser várias vezes o esteio que tem sustentando a felicidade de meu querido pai. Meu querido pai, Jairo Bernardes, obrigado pelo exemplo de vida e pelo amor, que, mesmo a sua maneira, percebo em cada abraço e carinho que trocamos.

A você, Geovanna Gonçalves, que, independente da situação, foi meu escudo e minha maior defensora. Amo-te muito.

Leandro Rezende, como agradecê-lo? Entre tantos tropeços e dificuldades partilhados, tantas vezes que não pude ser ou estar da forma que devia e que eras merecedor, mas você esteve ao meu lado dando-me seu apoio, carinho, proteção e amor. Obrigada!

Nívea Rodrigues, tantos adjetivos e agradecimentos caberiam aqui. Sempre tão guerreira e protetora. Obrigada pela constante preocupação em não deixar meus sonhos morrer

e por me fazer ter equilíbrio e despreocupação com aquilo que tenho de melhor e muitas vezes cumprindo meu papel na ausência destes longos dias.

Dona Angêla Rezende e João Rezende, pessoas tão lindas. Na simplicidade de seus atos e na grandeza de seus corações me conquistaram como filha. E isso tornou-me privilegiada entre tanto outros. Obrigada pelo incentivo, carinho e compreensão.

Meus queridos, amados e doces amigos. Como agradecer-lhes?

Renata de Oliveira você meu ombro amigo e sensato. Quantas memórias partilhadas, já nem lhe tenho, simplesmente, como amiga ou comadre e sim como parte inseparável de minha vida. Obrigada pela sua amizade de sempre e por acolher-me em momentos de dúvida e solidão.

Clénia Rios, és especial e nunca se ausentou, sendo minha justa defensora e minha irmã de lutas. É um exemplo de determinação, de responsabilidade e caráter. Obrigada por todas as conversas e conselhos.

Dayse Silveira, quantas noites, quantos diálogos... Você traz-me serenidade e mesmo com nossa complexidade aparente tenho descoberto, com você, o mundo de outras formas e reformulado velhos conceitos.

Kênia Santos que compartilhou comigo conquistas, espaços, saudades, sonhos e, muitas vezes, o árduo dia-a-dia. Obrigada pela amizade, companhia e auxílio.

Ana Paula, minha querida, você com sua meiguice e sabedoria sempre com uma palavra de incentivo e amizade participou de meu caminhar e deixou a sua marca. Como sou agraciada pelos inúmeros momentos que pudemos rir e partilhar.

Wanély Aires como sou feliz perto de ti, quantas conversas, sorrisos, choros e alegrias. Você, pessoinha de fibra e determinação, mas que soube ser tão dócil e amiga quando precisei.

Leiliane Rezende, Joyce França e Cícero Lima, muito obrigada, pela amizade, pelo carinho e pelas inúmeras vezes que partilharam seu tempo comigo auxiliando-me nas disciplinas e compartilhando diversos momentos juntos. Vocês são lindos presentes que este projeto me trouxe.

César Santos, Thiago Elias, João Paulo Mendes, meus amigos. De forma direta participam de minha vida e de meu caminhar. Agradeço pelo carinho, estímulo e apoio que a mim dedicaram.

À Maria Fernanda. Quando olho para você e me vejo em seus olhos estremeço-me. Entre tantas coisas, por não estar acompanhando você em cada sorriso, em seus primeiros passos ou tropeços, em suas graças. Mas saiba, meu grande amor, que é por você, pelo seu afago e abraço, a cada retorno, todas as minhas conquistas. Desde um levantar em dia nublado a conclusão deste projeto. Agradeço por, mesmo em sua tão pouca idade e, muitas vezes, não sabendo expressar diretamente, ter me compreendido, ter tolerado os longos dias e noites de estudo em que me separei de ti, por ter esperado e me amado. Obrigada minha querida por sempre dar-me forças para continuar.

A todos que cercaram minha vida com apoio durante esta jornada eu também dedico este trabalho. Obrigada!

“ Nada há de te fazer parar quando segues o caminho que faz teu coração vibrar.”

Carolina Salcides.

Resumo

A especificação, análise e *design* de sistemas de tempo-real são atividades altamente dependentes de uma efetiva compreensão do domínio de aplicação e descrevem, em grande parte, a representação de seus diferentes requisitos. A utilização de abordagens dirigidas a modelos, para o desenvolvimento de sistemas de tempo-real, têm ganhado força em diversos campos do desenvolvimento de software e tende a contribuir para minimizar a complexidade inerente ao desenvolvimento destes sistemas. A UML têm sido utilizada intensamente nos últimos anos para modelagem de requisitos de tempo-real. No entanto, a UML por si só não consegue representar completamente as diversas características pertinentes a estes sistemas. Uma das grandes vantagens da UML é a sua ampla capacidade de extensão permitindo a criação de *profiles* específicos. Assim, este projeto irá explorar o uso dos *profiles* SysML e MARTE para modelagem de requisitos de sistemas de tempo-real para um sistema de controle de tráfego. O objetivo principal é utilizar os *profiles* SysML e MARTE da UML para modelagem, representação, classificação, documentação e, ainda, para definir critérios de rastreabilidade de requisitos de softwares de tempo-real. A abordagem proposta mostrou-se relevante no contexto da especificação, análise, classificação e modelagem de requisitos para sistemas de tempo-real. O diagrama de requisitos da SysML mostra explicitamente os vários tipos de relacionamentos entre diferentes requisitos, aumentando o espectro de compreensão e definição dos requisitos de um sistema de tempo-real. O *profile* MARTE complementou a precisão do cenário com anotações não-funcionais bem formadas, em geral para requisitos temporais. Os conceitos de SysML e MARTE, articulados no diagrama Requisitos da SysML são altamente complementares cobrindo muitos dos propósitos necessários a especificação de requisitos para sistemas de tempo-real.

Palavras chave: desenvolvimento de software, engenharia de requisitos, modelagem de software, sysml, marte.

Abstract

The specification, analysis and design of real-time systems (RTS) are activities that are highly dependent on an effective understanding of the application domain and on the completeness of the representation of their primordial requirements. The use of model-based approaches for the development of RTS systems tend to contribute to minimizing the complexity of the system development process. The UML has been strongly used in recent years for modeling real-time software. However, UML alone does not represent with completeness important characteristics of these systems, such as timing requirements. UML is a language that has several extensions capabilities enabling the creation of specific profiles. This work will explore the use of UML profiles SysML and MARTE to modeling software requirements of RTS for the control of road traffic. The main objective is to demonstrate the application of SysML with MARTE stereotypes, which enables the modeling, representation, classification, documentation and, also, for defined criteria of traceability of real-time software requirements. The proposed approach was shown to be relevant in the context of the specification, analysis, classification and modeling requirements for real-time systems. The SysML Requirements diagram shows explicitly the various types of relationships between different requirements, increasing the spectrum of understanding and defining the requirements of a real-time system. The MARTE profile complemented the precision of scenario with functional non annotations well formed, in general timing requirements. The concepts of SysML and MARTE, articulated in the SysML Requirements diagram are highly complementary covering many of the purposes of specifying requirements for real-time systems.

Keywords: software development, requirements engineering, software modeling, sysml, marte.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ATMS	Sistemas Avançados de Gerenciamento de Tráfego
ATIS	Sistemas Avançados de Informação ao Viajante
AVCS	Sistemas Avançados de Controle de Veículos
ARTS	Sistemas Avançados de Transporte Rural
APTS	Sistemas Avançados de Transporte Público
BR	Brasil
INCOSE	International Council on Systems Engineering
MARTE	Modelagem e Análise de Sistemas de Tempo Real e Embarcados
MG	Minas gerais
NFP	Propriedades Não Funcionais
OMG	Object Management Group
RTMS	Sistema de Controle de Tráfego de Ruas
SysML	Linguagem de Modelagem de Sistemas
SRRT	Tabela de Rastreamento de Requisitos SysML
UML	Linguagem de Modelagem Unificada
UTCS	Sistema de Controle de Tráfego Urbano

Capítulo 1

Introdução

Este capítulo tem como finalidade apresentar os conceitos sobre Desenvolvimento de Software e Engenharia de Requisitos, Sistemas de Tempo-Real e Modelagem de Sistemas de Tempo-Real, os quais fundamentaram esta pesquisa e são elucidados, respectivamente, nas Seções 1.1, 1.2 e 1.3. Na Seção 1.4 é realizada a revisão de literatura referente a Sistemas de Tempo-Real. Os trabalhos relacionados à pesquisa em questão são abordados na Seção 1.5. Na Seção 1.6, as Metas e Objetivos da Pesquisa são esclarecidos. O Problema de Estudo que fundamentou esta pesquisa é definido na Seção 1.7. A Metodologia de Pesquisa e suas Questões, Instrumentos e Estratégias de Pesquisa são apresentados na Seção 1.8. Finalmente, na Seção 1.9 é apresentado o esquema geral deste trabalho.

1.1 Introdução ao Problema de Estudo

A importância atual dada ao papel do software na sociedade, bem como a ciência de que o mesmo pode fazer diferença competitiva aos processos de negócio e às diversas atividades corriqueiras das pessoas, confirma as previsões de [Naisbitt 1988] quanto à transformação de uma sociedade industrial em uma "sociedade de informação".

O desenvolvimento de software abrange um conjunto de atividades, processos, ferramentas e metodologias aplicáveis ao desenvolvimento de sistemas informatizados (softwares) eficientes e que atendam plenamente suas necessidades [Sommerville 2011]. É uma atividade difícil e complexa, em que a criatividade e o rigor devem ser balanceados [Nuseibeh e Easterbrook 2000]. É bem por isso que, em diversos trabalhos acadêmicos, é discutida a complexidade relacionada ao desenvolvimento, implantação e manutenção de software [Brooks 1987] [Berry 2004] [Parviainen et al. 2005] [Booch et al. 2007].

Como praticamente todas as aplicações, sejam organizacionais, institucionais, financeiras e de entretenimento, são de alguma forma dependentes de softwares, em maior ou menor grau de complexidade, devem ser propostos modelos que permitam gerenciar, produzir, manter e prover qualidade a estas aplicações [Sommerville 2011].

A engenharia de software define um conjunto de etapas, geralmente interativas, que

estabelecem desde os processos da engenharia de requisitos aos processos de projeto, desenvolvimento, validação e verificação e evolução do sistema em estudo. Como destacado em [Sommerville 2011], a engenharia de software se preocupa com todos os aspectos da produção de software, desde os estágios iniciais de especificação do software à manutenção do mesmo, depois que foi implantado/entregue.

Os processos de análise e a especificação de requisitos são atividades que, se bem realizadas, contribuem significativamente e decisivamente para o desenvolvimento satisfatório de software [Cabral e Sampaio 2008]. Por se tratarem de atividades de grande importância no ciclo de vida do software, e que se relacionam diretamente com a qualidade do produto a ser desenvolvido, a Engenharia de Requisitos precisa ser devidamente planejada. Sendo assim, deve ser aplicada de forma abrangente para assegurar que o conjunto completo das necessidades e requisitos dos usuários sejam capturados e, posteriormente, transformados em um conjunto validado de requisitos em todo o ciclo de vida [Carrillo e Nicolas 2011].

A demanda na sociedade por um maior número de aplicações informatizadas e o grau de automação e complexidade dos modernos sistemas de software no mercado, geralmente sistemas distribuídos e concorrentes, torna as atividades de Engenharia de Requisitos tanto mais importantes quanto difíceis. De acordo com [Ghezzi et al. 2003], a complexidade do desenvolvimento de software pode ser tratada por meio de estratégias de decomposição (dividir para conquistar) e, ainda, técnicas tais como ocultamento de informações, decomposição funcional, modularização e uso de abstração e modelagem no processo de desenvolvimento de software.

1.2 Conceitos de Sistemas de Tempo-Real

A característica principal dos sistemas de tempo-real (STR) é que eles têm por obrigação responder a eventos dentro de um período pré-definido e restrito de tempo. Este tipo de sistema tem por objetivo monitorar condições do ambiente e prover respostas a diferentes entradas monitoradas [Laplante 2006].

Os STR têm sido estudados intensivamente pois, além das qualidades genéricas dos softwares de propósito geral, são caracterizados por quão bem eles satisfazem os requisitos de tempo de resposta e de segurança, os quais estão diretamente relacionados ao critério de corretude e confiabilidade do sistema. Prover segurança e confiabilidade a STR são missões primárias do projeto deste tipo de sistema, pois objetivam garantir que o processo seja executado sem causar riscos inaceitáveis [Burns e Wellings 2001]. Os STR são tipicamente sistemas embarcados que interagem diretamente com equipamentos físicos, monitores, sensores e pessoas, sendo comumente encontrados em muitas áreas, tais como plantas industriais, controle de operações críticas, de telecomunicações, de transporte, militar e de saúde [Everett e Honiden 1995].

Os sistemas de tempo-real, como destacado anteriormente, devem responder a estímu-

los gerados externamente dentro de um período finito e especificável [Everett e Honiden 1995]. Portanto, esforços devem ser despendidos para analisar, projetar e validar esse tipo de sistema, almejando assim prover maior confiabilidade e segurança aos mesmos.

Um dos principais desafios ao processo de desenvolvimento de software é a especificação de requisitos, pois sua corretude e qualidade impactarão diretamente no desenvolvimento do projeto, influenciando em atrasos no cronograma, aumento de custos e possivelmente em um software de má qualidade. Para o desenvolvimento de STR isto também é uma realidade, uma vez que qualquer falha nestes softwares pode acarretar sérios danos (humanos/materiais). Como destacado por [Zhu et al. 2012], a análise e o projeto de requisitos funcionais tem recebido muita atenção dado os vários métodos de engenharia de software propostos para sua representação. No entanto, como já abordado, os STR são frequentemente caracterizados por um conjunto de propriedades não-funcionais que devem ser atendidas. Assim, as diversas propriedades não-funcionais dos STR devem receber maior atenção ao longo dos processos de especificação, documentação, análise, projeto, implementação, testes e manutenção de requisitos destes sistemas.

1.3 Contextualização da Modelagem de Software para Sistemas de Tempo-Real

A modelagem de software utiliza combinações gráficas e textuais para expressar componentes de um sistema, objetivando, assim, melhorar a compreensão do software e, principalmente, aumentar sua completude, corretude e consistência [Bezerra 2006]. É relatado, em [Booch et al. 2007], que a construção de modelos torna possível destacar ou enfatizar certas características críticas de um sistema em detrimento de outras não tão importantes, contribuindo para sua compreensão e análise.

A modelagem de aplicações de tempo-real não é um processo trivial, uma vez que o projeto das mesmas deverá conter, na maioria dos casos, uma análise do comportamento dos elementos distribuídos em rede e a disposição dos diversos componentes do sistema. Em geral, os modelos elaborados devem contemplar elementos físicos e lógicos como, por exemplo, os sensores e atuadores, que estão aptos, respectivamente, a captar estímulos gerados externamente e respondê-los dentro de um intervalo de tempo finito e especificável.

A modelagem de STR é um processo complexo que depende cada vez mais de uma efetiva interação entre disciplinas, tais como mecânica, eletrônica e engenharia de software. O uso de modelos no desenvolvimento de STR está ganhando força por abordar estas questões [Espinoza et al. 2009], e, ainda, por permitir representar e compreender a complexidade inerente a um software. O uso de modelos permite que os projetistas de diferentes disciplinas possam compartilhar conhecimentos, facilita a compreensão do *design*, diminui custos na medida em que riscos potenciais podem ser descobertos e tratados

desde as fases iniciais do desenvolvimento e, também, permite avaliar o nível do sistema na busca de maior qualidade e confiabilidade.

O uso de técnicas de modelagem e especificação de requisitos são imprescindíveis ao processo de desenvolvimento de sistemas de tempo-real, uma vez que, por meio destas, a documentação dos sistemas pode ser criada, servindo como base para o seu desenvolvimento. No entanto, se a linguagem de modelagem não for bem definida, compreendida ou utilizada, inviabilizará a construção de modelos expressivos e completos o suficiente para serem usados nas etapas posteriores, e erros e/ou inconsistências podem ser introduzidos.

1.4 Revisão da Literatura Correlata sobre Modelagem de Sistemas de Tempo-Real

Um grande desafio da sociedade atual é o desenvolvimento satisfatório de STR, respeitando restrições como custos e prazos e, ainda, manter e evoluir esses sistemas. Este desafio está associado a um outro importante: o desenvolvimento de princípios, métodos, algoritmos e ferramentas confiáveis e bem fundamentados para programação e engenharia confiável, segura, com custo efetivo, eficaz e eficiente para os STR em todo o seu ciclo de vida [Pettersson 1999] [Soares 2010].

Para auxiliar na completude e corretude das funções e restrições de tempo específicas e inerentes ao desenvolvimento de softwares de tempo-real, descritas no documento de requisitos, diferentes métodos, técnicas e linguagens de modelagem foram propostos ao longo do tempo. Podem ser citados os métodos formais [Lee 2002] [Laplante 2006] [Burns e Wellings 2001], a UML [OMG 2011b] (Seção 2.5) e os *profiles* UML, foco deste trabalho, SysML (Seção 2.7) e MARTE (Seção 2.8).

As linguagens formais utilizam especificações lógicas e matemáticas para representar os componentes do sistema e assegurar-lhes a consistência na especificação [Adrial 2007]. Diferentes métodos formais têm sido propostos como, por exemplo, as máquinas de estado finitos, os *Statecharts*, as Redes de Petri (RdP) e a Linguagem *Z*. Dada a importância destes métodos para os processos de engenharia de software, alguns trabalhos que descrevem sua aplicabilidade, no contexto do desenvolvimento dos STR, serão brevemente abordados.

No trabalho [Cheng e Krishnakumar 1993], é mostrada uma extensão da máquina de estados finitos (FSM) para modelagem de circuitos integrados. O método proposto transforma automaticamente descrições de circuitos em alto-nível (por exemplo VHDL) em um circuito em máquina de estado finito estendida e, também, formula um método automático para geração de vetores funcionais para circuitos sequenciais. No trabalho [Lin et al. 2008], as FSM são utilizadas para rastrear e gerenciar/capturar mudanças em uma especificação de requisitos.

Em [Jo et al. 2009], trabalhou-se na aplicação de uma abordagem híbrida para incorporar a linguagem formal *Z* e "*Statemate MAGNUM*" para especificação de requisitos formais em sistemas de sinalização ferroviária. Segundo os autores, a proposta justifica-se pela criação de um modelo que contribui para a integridade das especificações de requisitos via modelos formais e, também, por melhorar a aplicabilidade real do modelo. As FSM são utilizadas em [Billington et al. 2010] para representar o comportamento de um sistema robótico.

No trabalho de Harel [Harel 1987], é apresentada uma extensão às máquinas de estados para representar especificações e *design* de sistemas complexos e de tempo-real. Tal extensão foi nomeada *Statecharts*, com três elementos essenciais adicionados, incorporando noções de concorrência, hierarquia e comunicação ao diagrama de máquina de estados.

Os *Statecharts* são amplamente empregados para modelagem de software. O trabalho [Jokela e Lindberg 1990] usa os *Statecharts* para análise de requisitos e para criação de modelos orientados ao usuário. Os modelos elaborados servem para especificar a modelagem completa do software e os modelos de usuário. No trabalho [Kang e Ko 1996], é proposta uma nova abordagem para verificação de propriedades de segurança para sistemas de tempo-real em que os *Statecharts* são empregados para especificar o comportamento do sistema. Em [Eshuis et al. 2002], é descrito o uso da abordagem Orientada a Objetos para requisitos e *design* de software, com *Statecharts*. Em [Burmester e Giese 2003], os autores propuseram uma extensão aos *Statecharts* de tempo-real para a ferramenta Fujaba, com intuito de superar as limitações dos *Statecharts* e apoiar uma semântica bem definida baseada em automâtos temporizados e síntese de código. O trabalho [Moura e Guedes 2012] propõe uma metodologia para modelagem e avaliação de sistemas de controle de produção industrial, com a representação de operações sequenciais, paralelas e temporizadas.

As Redes de Petri [Murata 1989] são outro método formal usado para especificar as operações a serem realizadas em um ambiente de multiprocessamento ou multi-tarefa, sendo aplicáveis a diversos tipos de STR. Embora as Redes de Petri possuam uma rigorosa base matemática, elas também podem ser descritas graficamente [Murata 1989].

Uma extensão das Redes de Petri originais, nomeada como Redes de Petri Temporizadas Coloridas, é utilizada em [Tchako et al. 1994]. Neste trabalho, as Redes de Petri são aplicadas no contexto de sistemas de produção para representar restrições de produtividade e flexibilidade e como requisitos de controle de atividades e de recursos são modelados.

Em [Elkoutbi et al. 2002], os autores apresentaram um processo de engenharia de requisitos para sistemas de tempo-real que visa uma especificação formal do sistema, usando Redes de Petri Temporizadas (TPN). Nesta abordagem, a primeira atividade é a elaboração do diagrama de casos de uso. Em seguida, um cenário correspondente, em forma de diagrama de sequência, é criado para cada caso de uso. Posteriormente, a

especificação do cenário é derivada em uma rede de Petri temporizada. Finalmente, as TPNs parciais são agrupadas para obter uma TPN integrada e, em seguida, as demais verificações são executadas. Em [Liu et al. 2009], uma extensão às Redes de Petri é aplicada no contexto do ciclo de fabricação de semicondutores.

Segundo [Soares e Vrancken 2012], o comportamento dinâmico de um grupo de sinais de controle de tráfego em uma rede de intersecções foi modelado por meio de uma extensão às Redes de Petri. Neste trabalho, uma Rede de Petri é estendida através da associação de intervalos de tempo a lugares e pela definição de um componente cujo comportamento é também baseado nessas redes. Características úteis para tratar complexidade, modularidade e modelagem de restrições de tempo-real foram adicionadas.

Outro método formal bastante conhecido é a linguagem *Z*. A linguagem *Z* [Spivey 1998] descreve um código de especificação formal, baseada na teoria dos conjuntos, cálculos lambda e em cálculo de predicados de primeira ordem. *Z* foi desenvolvida por *Jean-Raymond Abrial*, com a ajuda de *Steve Schuman* e *Bertrand Meyen*, no grupo pesquisa em programação na Universidade de *Oxford*.

A metodologia Clepsidra, citada em [Ciaccia et al. 1997], utiliza a notação *Z* para documentar requisitos de software. Em [Fidge et al. 1997], é desenvolvido o método *Quartzo*, uma variante da notação *Z*, para especificação de requisitos de software de tempo-real.

Em [Sengupta e Bhattacharya 2006], é apresentada, para a especificação de requisitos de software, uma abordagem expansível do diagrama de casos de uso expresso em notação *Z*, e, posteriormente, representada através do diagrama entidade-relacionamento. No trabalho [Yan e Tang 2008] é descrito um estudo de caso em sistemas de controle de tráfego rodoviário em que é proposta uma abordagem de modelagem e verificação formal, que utiliza a Rede Autômata Temporizada (TAN) e diagramas de sequência UML, para sistemas de tempo-real.

Além das linguagens formais, outras linguagens semi-formais tais como a UML, têm sido utilizadas como ferramentas para especificação, análise e representação de requisitos de sistemas de tempo-real. A UML foi originalmente desenvolvida para criar modelos de sistemas de software para um grande número de domínios diferentes como, por exemplo, processos de negócios [Selic 2007], engenharia de sistemas [Briand et al. 2003] e sistemas de tempo-real [Lee 2002]. A UML [OMG 2011b] é atualmente a linguagem padrão para visualizar, especificar, construir e documentar os artefatos de software de um sistema. Alguns trabalhos discorrem sobre o uso da UML no âmbito da engenharia de requisitos.

Um diagrama bastante conhecido e que tem sido utilizado para a modelagem de requisitos é o diagrama de caso de uso. Mesmo antes da UML surgir como a principal linguagem de modelagem da Engenharia de Software, os casos de uso já eram uma prática comum para representar graficamente os requisitos funcionais em outras metodologias, tais como os modelos de Rumbaugh (método OMT) e Booch (método Booch) [Lee 2002] [Jacobson

2004]. Muitos trabalhos relatam sobre o uso do diagrama de casos de uso [OMG 2011b] nos processos de engenharia de requisitos.

Em [Somé 2006] é proposta uma abordagem para modelagem de requisitos que utiliza o diagrama de casos de uso e provê uma formalização para os mesmos, em que é criada uma forma restrita de linguagem natural para descrição de casos de uso e, com isso, a derivação e simulação de uma especificação executável dos casos de uso. No trabalho [Xu et al. 2011], é proposta uma abordagem para modelagem de requisitos de software em três níveis de representação do diagrama de casos de uso, nos quais, para sua formalização, são utilizados conceitos da Lógica de Hoare, regras de composição e Redes de Petri.

Por não ter uma semântica bem definida, o diagrama de caso de uso pode levar a diferenças na interpretação dos cenários e atores [Xu et al. 2011]. Este diagrama pode ser mal utilizado quando muitos detalhes são adicionados, o que pode incorretamente transformar o diagrama em fluxogramas ou torná-lo difícil de compreender [Firesmith 2004]. Segundo [Espinoza et al. 2009], os casos de uso são frequentemente criticados por um número de limitações. Casos de uso são aplicados principalmente para modelar requisitos funcionais, não sendo muito aplicáveis para modelos não-funcionais [Soares e Vrancken 2008]. [Soares 2011] destaca que os relacionamentos entre requisitos e as várias partes da arquitetura que satisfazem estes requisitos são difíceis de rastrear.

Outros trabalhos aplicam os demais diagramas da UML para processos da engenharia de requisitos. O trabalho [Bianco et al. 2002] tem por intenção empregar a UML para a especificação de sistemas de tempo-real. Os autores utilizam a UML com uma extensão conveniente do diagrama de *Statecharts* UML para representar a especificação do comportamento de tempo-real.

Em [Douglass e Evangelist 2000], o diagrama de sequência é utilizado, em um estudo de caso, para representar aspectos comportamentais do sistema. Como contribuição, os autores concluem que este diagrama não consegue representar satisfatoriamente restrições de tempo tendo em vista que não são diagramas temporais e sim comportamentais e expressam, na modelagem, apenas a ordem cronológica de uma interação. No artigo [Glinz 2000], propõem-se algumas soluções para as deficiências da UML e cita-se o uso de estereótipos como um mecanismo que pode ser utilizado de modo a prover refinamentos ou redefinições nos elementos da linguagem, sem (diretamente) modificar o metamodelo da linguagem.

Em [Von et al. 2002], a UML é empregada em uma abordagem para a engenharia de requisitos, baseada em modelos com o objetivo de facilitar o processo de especificação de requisitos de sistemas embarcados automotivos. Para tal fim, utiliza-se o conjunto de notações da UML para criar o metamodelo, que é um subconjunto da UML, nomeado como Linguagem de Modelagem Automotiva (AML). No trabalho [Charles et al. 2007], a UML é utilizada para representação de Modelos de Computação e Comunicação em tempo-real para sistemas embarcados (MoCCS), fornecendo meios para a construção de

modelos de tempo sofisticados.

No trabalho [Chicote et al. 2007], é abordada a especificação de requisitos dirigida a modelos, enfatizando como tradicionais processos de especificação, baseados em documentos, podem ser transformados em um processo de modelagem de requisitos. É proposto um metamodelo de requisitos chamado REMM (Metamodelo de Engenharia de Requisitos) que inclui elementos que devem aparecer em um modelo de requisitos (requisitos, *stakeholders*, casos de testes), juntamente com seus possíveis relacionamentos.

Em [Helming et al. 2010], é proposta a URML (Linguagem de Modelagem de Requisitos Unificada), uma linguagem que integra, a partir da UML, aspectos como modelagem de objetivos, de recursos e análise de riscos em um modelo homogêneo e compreensível. A URML define uma linguagem para expressar, validar e relacionar requisitos, facilitando, assim, a compreensão das interdependências entre os requisitos.

O artigo [Cóté 2011] apresenta um método para especificação de requisitos de software com uso da UML e mostra sua aplicação para a engenharia de requisitos evolutiva em que novos requisitos são analisados no contexto de um conjunto de requisitos já fornecidos. A UML é utilizada por meio de adição de estereótipos que suportam a evolução do software. Neste trabalho, os estereótipos são complementados pelas restrições e consultas expressas a partir da OCL (*Object Constraint Language*) [Hamie 1998].

São amplamente descritas na literatura as deficiências da UML e as propriedades não bem representáveis pela mesma [Douglass e Evangelist 2000] [Selic 2007]. Segundo [Bianco et al. 2002], as restrições temporais e as atividades realizáveis concorrente ou paralelamente não são bem expressadas na UML, apresentando dificuldades para representar com corretude e completude as propriedades dos sistemas de tempo-real. Em [Weilkiens 2008], é relatado que a UML, apesar de possuir modelos para representar projetos de sistemas de tempo-real, por si só não consegue lidar com completude com as especificações de softwares de tempo-real como restrições temporais, controle, sincronização e paralelismo de processos.

Uma das características mais importantes da UML é a sua capacidade de extensibilidade, facilitando sua extensão em *profiles* que suportam a modelagem de propriedades de tempo-real [Lee 2002]. *Profiles* são usados para estender a UML com elementos específicos do domínio para que a linguagem não seja sobrecarregada com características que não são necessárias em todas ou pelo menos muitas áreas de modelagem de sistemas [Martin 2002].

O *profile* SPT (*Scheduling, Performance and Time*) [OMG 2005] foi criado para suprir algumas lacunas da UML em áreas chaves de interesse para modelagem de sistemas de tempo-real [Bennett e Field 2004]. O *profile* QoS [OMG 2008], adotado pela OMG em 2004 para Qualidade de Serviço, fornece ao usuário facilidades para definir maior variedade de requisitos com foco principalmente em escalonabilidade e análise de desempenho [Cabral e Sampaio 2008]. O *profile* UML-RT [Rational Software 2010] estende os conceitos básicos

da UML para facilitar a construção de complexos sistemas de tempo-real. Este *profile* é um padrão industrial, com foco principal na especificação da arquitetura de sistemas de tempo-real e de protocolos [Flynn e Hamlet 2006].

O *profile* SPT introduz uma noção quantificável de tempo e recursos, agregando aos elementos do modelo informações quantitativas relacionadas a tempo, prazos, desempenho e análise de escalonabilidade [Charles et al. 2007]. No trabalho [Xu et al. 2003], o *profile* SPT é utilizado para descrever requisitos não-funcionais ligados ao desempenho de um sistema. Os autores realizam, neste trabalho, a análise e a representação dos requisitos não-funcionais, como tempo de resposta, capacidade e escalabilidade.

Em [Thramboulidis 2004], o *profile* SPT é usado no cenário de controle e automação, sendo empregado em um ambiente cuja abordagem dirigida a modelos é adotada. O SPT foi proposto com objetivo de resolver problemas existentes na UML, citados anteriormente, e ser um padrão para modelagem de software de tempo-real. No entanto, em [Silvestre e Soares 2011] é relatado que o SPT possui alguns problemas, incluindo a dificuldade para modelar tempo lógico e a modelagem de sofware baseada em componentes.

Podem ainda ser citados dois importantes *profiles* da UML que são a SysML [OMG 2010] e o MARTE [OMG 2011a]. Devido à sua importância para o presente trabalho tais *profiles* serão abordados separadamente e em maior nível de detalhe no Capítulo 2.

Como já abordado, é importante que os requisitos sejam bem explorados, elicitados e documentados para nortear de forma efetiva e com clareza o processo de desenvolvimento. Por esta razão, e para minimizar as dificuldades para a modelagem de sistemas de tempo-real, propõem-se, neste trabalho, utilizar o *profile* SysML (descrito na Seção 2.7), que estende a UML e introduz estereótipos para diversas restrições não abordadas na UML e adiciona novos diagramas. Pretende-se, também, utilizar os estereótipos do *profile* MARTE (descrito na Seção 2.8) em conjunto com a SysML para representar requisitos de sistemas de tempo-real contribuindo para sua análise, representação, classificação e mapeamento de rastreabilidade.

1.5 Trabalhos Relacionados a SysML e MARTE

A linguagem SysML [OMG 2010] permite a modelagem de propósito geral para diversos tipos de aplicações em engenharia de sistemas, possibilitando a especificação, análise, projeto, verificação e validação de sistemas complexos. A SysML é apoiada pela OMG (*Object Management Group*) e pela INCOSE (*International Council on Systems Engineering*) [OMG 2010] e é considerada uma evolução da UML 2.0, podendo ser aplicada a sistemas que incluem hardware, software, informações, processos, pessoas e instalações.

Diversos trabalhos discorrem sobre o uso da SysML para modelagem de sistemas. Dentre estes trabalhos, alguns autores propõem a aplicação da SysML e do diagrama de requisitos da SysML no domínio de sistemas de tempo-real.

Nesse contexto, em [Soares e Vrancken 2007] mostra-se a aplicação do diagrama de requisitos da SysML para a especificação de requisitos de sistema. No trabalho demonstrou-se que a modelagem de requisitos por meio de diagramas da SysML pode ser útil para representar explicitamente as várias maneiras que os requisitos podem ser relacionados uns aos outros, e que, a partir dos modelos de requisitos criados com SysML, é mais fácil construir diagramas de caso de uso que representem os requisitos do sistema. [Linhares et al. 2007] apresenta uma abordagem para combinar os diagramas da SysML com Redes de Petri para representar uma maneira formal para modelagem e verificação de sistemas. Este trabalho consiste em primeiro modelar o sistema utilizando o diagrama de blocos e o diagrama de requisitos da SysML e, em seguida, a modelagem detalhada de um requisito é realizada, utilizando as Redes de Petri.

O trabalho [Soares e Vrancken 2008] refere-se à aplicação de uma abordagem para Engenharia de Requisitos dirigida a modelos baseados no diagrama de requisitos da SysML, diagrama de casos de uso da SysML e na tabela de requisitos da SysML, com foco nas atividades de documentação e análise de requisitos. Neste trabalho, foi proposta a aplicação da SysML para especificar e modelar requisitos de usuário para um sistema de gerenciamento de tráfego rodoviário. Em [David et al. 2010] é proposto um método para unificar e facilitar o desenvolvimento de sistemas com requisitos de segurança crítica, ligando a fase de projeto funcional, com uso da SysML, com técnicas de confiabilidade comumente utilizadas.

No artigo [Soares et al. 2011], é proposto um *framework* para documentação e análise de requisitos em projetos de sistemas intensivos de software. A estrutura é inspirada no estilo em camadas para arquitetura de software para mostrar a necessidade da especificação de requisitos em diferentes níveis de detalhe (requisitos de usuário, de sistema e de implementação). Neste trabalho, os requisitos do usuário são expressos em linguagem natural e modelados por meio do diagrama de requisitos da SysML. O autor conclui que o diagrama de requisitos da SysML pode modelar os requisitos funcionais e não-funcionais, a hierarquia entre os requisitos, e outras relações, como a dependência entre os requisitos e a reutilização destes.

Em [Li et al. 2011], a SysML é empregada para a análise de requisitos para o processo de suporte e manutenção de aeronaves militares através do diagrama de requisitos da SysML. O processo para o sistema de suporte e manutenção envolvido é descrito estático e dinamicamente por meio do diagrama de blocos interno, diagrama de atividades e diagrama de sequência da SysML. Segundo o autor, os modelos criados têm boa escalabilidade, podem ser usados para o estudo aprofundado do processo de suporte de manutenção e podem, também, ser utilizados como parte de níveis elevados de simulação de sistemas militares de aeronaves.

Outro *profile* criado para modelagem e análise de sistemas de tempo-real e embarcados é o MARTE [OMG 2011a]. O *profile* MARTE (*Modeling and Analysis of Real-Time*

and Embedded Systems) [OMG 2011a] estende e adiciona capacidades à UML para modelagem e análise de sistemas de tempo-real e sistemas embarcados, fornecendo suporte para especificação, projeto, verificação e validação de sistemas. O *profile* MARTE tem por intenção substituir o *profile* UML SPT [Lee 2002] [Kumar e Jasperneite 2010]. Este *profile* apresenta características de domínio de STR que são específicas e relevantes para a modelagem de sistemas de tempo-real, possibilitando expressar propriedades não-funcionais, plataformas de execução, alocação de recursos e noções quantificáveis de tempo [OMG 2011a].

Por ser um *profile* desenvolvido recentemente e com pesquisas ainda incipientes, alguns trabalhos recentes têm sido propostos para explorar o uso e o poder de expressidade deste *profile*. Podem ser destacados o trabalho [Demathieu et al. 2008], em que os autores relatam a primeira experiência de uso de MARTE em um STR. É realizado, neste trabalho, um estudo de caso em um sistema de desenvolvimento robótico Josegil [Josefil 2013], usando MARTE com o objetivo de avaliar a aplicabilidade do *profile* aos sistemas e práticas atuais da engenharia de software, informar sobre os estágios de adoção/utilização do *profile*, bem como descrever como utilizar alguns construtores MARTE e explicar sobre a sua adequação.

O trabalho [Espinoza et al. 2008] define a TADL (Timing Augmented Description Language) para ser utilizada no projeto como complemento aos modelos AUTOSAR [Voget 2010]. Neste projeto, MARTE é utilizado como uma possível entrada para a TADL em que as construções MARTE são utilizadas semanticamente, pois MARTE fornece uma base forte para análise de temporização e, sintaticamente, por definir uma linguagem de modelagem para muitos aspectos comuns de tempo para definição de requisitos no contexto do *framework* TIMMO [Espinoza et al. 2008]. Por fim, MARTE é aplicado em um estudo de caso no domínio de um projeto automobilístico.

O uso do MARTE é citado na literatura para desenvolvimento de linhas de produto [Belategi et al. 2010]. Em [Zaki e Jawawi 2011], o *profile* MARTE é utilizado para compor uma metodologia de modelagem para engenharia de requisitos que satisfaça às necessidades de modelagem e representação de sistemas de tempo-real. [Arpinen et al. 2012] apresenta uma extensão ao *profile* MARTE para modelagem de aspectos de gerenciamento dinâmico de energia. Assim, por meio da extensão do *profile* MARTE, nomeada como *profile* DPM, permite-se a modelagem de complexas políticas de DPM, tais como: parâmetros de QoS, utilização de componentes, entre outras.

MARTE é usado em [Ebeid et al. 2012] para extração dos requisitos de comunicação (especificação *front-end*) para especificação de aplicações distribuídas em rede. Neste trabalho, as informações, as restrições no *design* e os elementos de comunicação são modeladas por meio do MARTE e a abordagem elaborada é aplicada em um estudo de caso relacionado com automação de edifícios.

Em [Iqbal et al. 2012], os autores mostram o uso de MARTE em três problemas

industriais distintos: modelagem de arquitetura e configuração em larga escala de sistemas de controle integrados altamente configuráveis (Sistemas de Produção Submarina), modelagem baseada em testes de robustez em sistemas de comunicação intensiva (Sistema de Conferência da Cisco Systems) e modelagem baseada em simuladores de ambiente para geração em larga escala de RTEs para teste (Sistema Marinho de Aquisição Sísmica da WesternGEco). Os autores centram em reportar as experiências empregadas para resolver os problemas relacionados à aplicação de MARTE nestes projetos. A contribuição do trabalho é um conjunto de considerações/guias sobre como aplicar o MARTE em contextos industriais, ajudando assim a reduzir o *gap* entre padrões de modelagem e necessidades do mercado. É realizada, ainda, uma análise minuciosa do domínio, descrevendo quais construtores, pacotes e estereótipos MARTE podem ser utilizados para distinguir os diferentes conceitos do domínio.

Em [Silvestre e Soares 2012a], o *profile* MARTE é utilizado em um estudo de caso para sistemas de tempo-real. Os autores elaboram um *framework* comparativo do *profile* MARTE com outros *profiles* para STR e realizam, também, uma extensiva análise das vantagens e desvantagens da utilização do MARTE. O trabalho [Silvestre e Soares 2012b] utiliza o MARTE para modelar um sistema de controle de sinais de trânsito. O MARTE é utilizado, neste trabalho, em conjunto com a UML, complementando alguns aspectos que historicamente são considerados deficientes, incluindo modelagem de tempo, recursos e processos. Os autores concluem, a partir de um estudo de caso, que os modelos elaborados em seu trabalho, apesar de serem mais complexos, podem ser mais expressivos que os modelados no *profile* SPT para sistemas de tempo-real.

No trabalho [Albinet et al. 2007], MARTE, SysML e EAST-ADL [ITEA 2004] foram utilizados para definir a metodologia nomeada como MeMVaTEx para rastreabilidade e classificação de requisitos. Esta metodologia é aplicada em um estudo de caso em sistemas automotivos. Os *profiles* MARTE e SysML são utilizados em [Mura et al. 2008], cujo foco é apresentar algumas orientações sobre como usar a SysML e o *profile* MARTE para identificar pontos de *design* que preencham as restrições de tempo de um STR.

Foram encontrados dois trabalhos que se assemelham a proposta deste trabalho de pesquisa. Nos trabalhos [Quadri et al. 2012a] e [Quadri et al. 2012b] foi descrito o projeto MADES *FP7 EU* que objetivou o desenvolvimento de uma metodologia dirigida a modelos que evolui a partir das práticas atuais de desenvolvimento de sistemas embarcados e de tempo-real. O projeto MADES é aplicado, neste trabalho, no domínio das indústrias de aviação e de vigilância. Uma das maiores contribuições deste trabalho é a descrição de uma completa metodologia baseada no uso combinado e refinado de SysML/MARTE para o *design*, validação, simulação e geração automática de código, enquanto integra aspectos tais como reuso de componentes de STR.

A metodologia MADES foca em um efetivo subconjunto de diagramas e elementos dos *profiles* SysML e MARTE para expressar diferentes aspectos relacionados a um STR

como, por exemplo, usando o diagrama de requisitos MADES, que integra conceitos de requisitos da SysML, para modelar requisitos de sistema; o diagrama de especificação funcional de blocos, que usa o conceito de blocos SysML, para realizar especificação funcional; o diagrama de especificação funcional refinado, que usa componentes MARTE, para cada componente em bloco SysML gerado na especificação funcional e outros conceitos da SysML/MARTE para especificação de propriedades não-funcionais, especificação de hardware, modelagem de tempo, entre outros [Quadri et al. 2012a] [Quadri et al. 2012b].

No trabalho [Gomez et al. 2012], propõe-se uma abordagem multi-visão baseada em MARTE e SysML para modelagem de visões de consumo de energia e seus relacionamentos com outros aspectos funcionais/não-funcionais, estruturais/comportamentais. Nesta abordagem, cada domínio pode ser tratado separadamente através de diferentes visões, mantendo fortes ligações dos elementos do modelo para com as outras visões. Para tanto, o *profile* MARTE é utilizado para definir o modelo de arquitetura do hardware e utiliza-se, nesse contexto, o pacote de propriedades não-funcionais MARTE para definição de propriedades como energia, voltagem e frequência. Já a SysML é utilizada para especificar, por meio do modelo paramétrico, as equações que definem relações matemáticas entre interesses não-funcionais de diferentes visões.

As abordagens propostas citadas anteriormente diferem da proposta neste trabalho em diversos aspectos.

O projeto MADES [Quadri et al. 2012a] contempla uma ampla metodologia para o desenvolvimento de sistemas embarcados e de tempo-real, sendo que esta metodologia não é específica para engenharia de requisitos de STR, mas para diferentes partes de seu processo de desenvolvimento. A metodologia MADES possui um conjunto distinto de diagramas MADES para representar, em diferentes etapas, a especificação de requisitos de sistema. Podem ser citados os diagramas MADES para a especificação de comportamento inicial do sistema, a especificação funcional do sistema, a especificação funcional refinada do sistema, a especificação de hardware/software do sistema, a especificação detalhada de hardware/software do sistema e, ainda, a especificação de alocação do sistema.

Os diagramas MADES, anteriormente citados, são construídos a partir da extensão de diagramas SysML e, em especial, os diagramas MADES para especificação de requisitos não contemplam, como explicitado no parágrafo anterior, a abordagem proposta neste trabalho que é mostrar o uso e a aplicação do diagrama de requisitos da SysML em conjunto com elementos do *profile* MARTE. No MADES é utilizado o diagrama de blocos da SysML para os processos de especificação funcional e refinada de requisitos.

A abordagem proposta por [Gomez et al. 2012] relata o uso do diagrama paramétrico da SysML e de pacotes MARTE para formular a abordagem multi-visão definida para a arquitetura do sistema. Neste trabalho, o foco está em representar propriedades arquiteturais do projeto, utilizando o pacote NFP MARTE e o diagrama paramétrico da SysML. A abordagem proposta em [Gomez et al. 2012], difere-se da abordagem proposta

na pesquisa em questão que foca em atividades da engenharia de requisitos e utiliza os pacotes *Foundations*, NFP e *Time* de MARTE em conjunto com diagrama de requisitos da SysML.

Assim, os conceitos e diagramas da SysML/MARTE complementam a metodologia/abordagem criada em ambos os trabalhos e não utilizados de forma agrupada como é proposto neste projeto de pesquisa. Isto é, o diagrama de requisitos é estendido, nesta pesquisa, e sua semântica enriquecida com construtores do *profile* MARTE.

De acordo com pesquisa bibliográfica realizada até o momento, não existem trabalhos na literatura que utilizam e demonstrem a aplicabilidade do *profile* SysML com estereótipos MARTE de forma agrupada (isto é, agregados no mesmo modelo) para modelagem de requisitos de software de tempo-real. Apesar da existência de inúmeras abordagens para modelagem de requisitos de software e para a modelagem de requisitos específicos de tempo-real, como apresentado nesta pesquisa, a Engenharia de Requisitos para STR ainda carece de metodologias representativas, para seu desenvolvimento e gerenciamento, que sejam expressivas, completas, corretas e padronizadas. Conforme abordado por [Espinoza et al. 2009], apesar de um número cada vez maior de *profiles* serem construídos em muitos domínios para a concepção de alguns tipos de sistemas, um único *profile* pode não ser adequado para cobrir todos os aspectos requeridos, como os aspectos multidisciplinares no domínio de sistemas embarcados.

Deste modo, são relevantes as pesquisas em linguagens de modelagem de requisitos para STR. Logo, o ponto chave desta pesquisa está em usar efetivamente estes dois *profiles* para modelagem de requisitos de sistemas de tempo-real, realizando uma análise dos possíveis benefícios apresentados pelo uso dos mesmos.

1.6 Metas e Objetivos da Pesquisa

O objetivo principal deste trabalho é contribuir para a modelagem de requisitos de software de tempo-real através da utilização dos *profiles* SysML e MARTE da UML e aplicar uma ferramenta para a modelagem de representações gráficas para este tipo de software. Diversas abordagens são propostas para elicitação, análise e *design* de STR. No entanto, o foco deste estudo está em prover, por meio da utilização de *profiles* e construtores adequados ao domínio de STR, uma extensão apropriada para representar com completude e consistência requisitos característicos a STR. De forma geral, este trabalho contempla os seguintes objetivos específicos:

- Realizar o estudo de ferramentas de modelagem de requisitos de software e suas características;
- Propor um modelo, combinando diagrama de requisitos da SysML com estereótipos MARTE para especificação de requisitos de tempo-real;

- Aplicar os diagramas e estereótipos dos *profiles* SysML e MARTE como linguagem de modelagem de requisitos de tempo-real;
- Avaliar a combinação de SysML e MARTE para rastrear requisitos;
- Utilizar de forma integrada as linguagens de modelagem UML, SysML e MARTE;
- Produzir uma ferramenta de modelagem de software que represente requisitos de software de tempo-real por meio da SysML com estereótipos MARTE;
- Estudar a aplicação do modelo proposto em um estudo de caso a partir de um conjunto de requisitos de software de tempo-real.

1.7 Definição do Problema de Estudo

Com o objetivo de desenvolver STR, as tarefas de modelagem devem cobrir diferentes fases, tais como análise de requisitos, projeto de arquitetura e *design*. Outras fases, como implementação, teste e integração, são consequências diretas das fases de modelagem [Soares 2010].

Na Figura 1.1, as principais atividades do processo de engenharia de requisitos são apresentadas. Como pode ser observado, entre as preocupações da engenharia de requisitos estão a negociação de requisitos, a elicitação, a manutenção, a validação e a priorização de requisitos. Todavia, o foco de estudo deste trabalho são as atividades, destacadas na Figura 1.1, de análise de requisitos, rastreamento e documentação/especificação de requisitos.

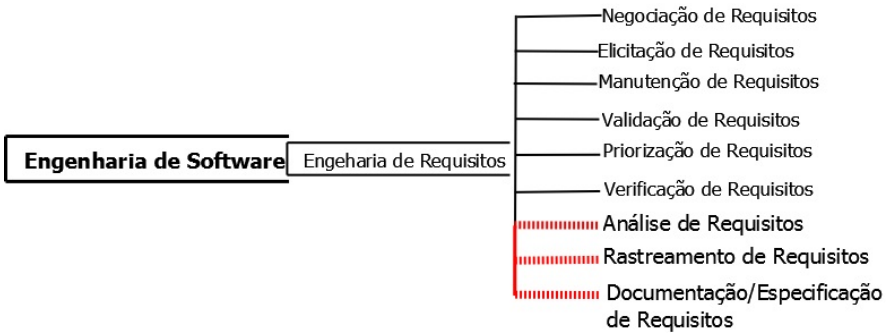

Figura 1.1: Escopo do Trabalho.

Este trabalho objetiva propor melhorias no processo de desenvolvimento de software de tempo-real, mais especificamente, na modelagem dos requisitos deste tipo de sistema. Para tanto, são estudadas técnicas e linguagens/ferramentas de modelagem que permitam modelar corretamente as propriedades intrínsecas a este tipo de aplicação como, por exemplo, requisitos de tempo, desempenho e paralelismo.

1.8 Metodologia

Para cumprir o objetivo da pesquisa, serão abordadas nesta Seção as questões, a abordagem e os instrumentos de pesquisa utilizados.

1.8.1 Questões de Pesquisa

Segundo [Shaw 2003], um bom trabalho de pesquisa deve responder a uma série de perguntas tais como: *Quais perguntas foram respondidas*? *Qual, exatamente, foi a contribuição do trabalho*? *Quais são os resultados*?

[Shaw 2003] destaca que existem vários tipos de questões de pesquisa e perguntas intimamente relacionadas a essas questões em engenharia de software. Sendo assim, este trabalho tem como tipo de pesquisa **"*design*, avaliação ou análise de uma instância em particular"**. As perguntas de pesquisa formuladas neste trabalho seguem a seguinte semântica: "Quão bom é X", "Qual é a (melhor) análise, *design*, implementação, manutenção ou adaptação para a aplicação X" e/ou "Como comparar X com Y?".

Este trabalho está fundamentado na hipótese de que "A combinação de estereótipos MARTE e o diagrama de requisitos da SysML agrega valor à modelagem de requisitos de software de tempo-real e torna-a mais expressiva". Para avaliar essa hipótese, as seguintes perguntas de pesquisa são propostas:

RQ1 - Como o diagrama de requisitos da SysML pode representar com consistência a classificação e o rastreamento de requisitos dos STR?

Esta questão é respondida no Capítulo 4. A resposta a esta pergunta é um metamodelo que adiciona ao diagrama de requisitos da SysML propriedades que permitem representar com completude características como classificação e rastreamento dos requisitos de sistemas de tempo-real.

RQ2 - Como integrar SysML e MARTE para especificação de requisitos de software de tempo-real?

Esta questão é respondida no Capítulo 5, uma vez que está relacionada à extensão do diagrama de requisitos da SysML e de seus relacionamentos, em que uma nova proposta para especificação de requisitos é elaborada, considerando questões (como classificação e agrupamento) e atributos chave (como temporização) para documentação de requisitos de sistemas de tempo-real.

RQ3 - De que modo o uso do *profile* SysML com estereótipos MARTE pode contribuir para tornar mais expressiva a especificação de requisitos de STR?

Esta questão é respondida no Capítulo 5 a partir do estudo de caso retratado para especificação de requisitos de sistemas de tempo-real. Com o metamodelo proposto e sua aplicação e comparação (Capítulo 6) a outras abordagens já existentes, para especificação de requisitos de STR, é possível fundamentar a contribuição deste trabalho no âmbito da

engenharia de requisitos e mais especificamente aos seus processos de análise, especificação e documentação.

1.8.2 Estratégia de Pesquisa

Dois paradigmas cobrem grande parte da pesquisa em áreas de Sistemas de Informação: o paradigma de ciência comportamental e o paradigma de ciência do *design*. O primeiro busca desenvolver e verificar as teorias que explicam ou predizem o comportamento humano e organizacional [Hevner et al. 2004]. Já o paradigma do *design* é ciência que procura entender os limites das capacidades humanas e organizacionais, criando artefatos novos e inovadores [Hevner et al. 2004].

Em termos de objetivo, o paradigma da ciência comportamental procura encontrar "o que é verdade", enquanto o paradigma da ciência projeto busca criar "o que é eficaz".

Como a maioria dos sistemas atualmente desenvolvidos são sistemas sociotécnicos [Sommerville 2011], sendo implementados dentro de um contexto organizacional, humano e/ou social, cabe ao paradigma comportamental estudar e predizer os requisitos e restrições relacionados a este contexto. Logo, o objetivo principal do paradigma de ciência comportamental é desenvolver e justificar teorias que explicam ou predizem fenômenos organizacionais e humanos envolvidos em processos de análise, *design*, implementação, gerenciamento e uso de sistemas de informação [Hevner et al. 2004].

No paradigma de ciência de *design*, o conhecimento e compreensão do domínio de um problema e sua solução são alcançados na construção e aplicação do artefato projetado [McKay e Marshall 2008]. Este paradigma caracteriza-se como um paradigma de resolução de problemas, pois destina-se a criar inovações que definem ideias, práticas, capacidades técnicas e produtos através da qual a análise, projeto, implementação, gerenciamento e uso de sistemas ocorram de uma maneira eficaz. Os artefatos, produtos do paradigma de *design*, não estão isentos das leis naturais ou teorias comportamentais. Ao contrário, sua criação se baseia em núcleos de teorias já existentes que são aplicadas, testadas, modificadas e estendidas através da experiência, criatividade, intuição e capacidade de resolução de problemas dos pesquisadores [Hevner et al. 2004].

Importante é, pois, evidenciar que a ciência do *design* é a estratégia de pesquisa adotada neste trabalho, atendendo ao objetivo desta pesquisa que é o de criar e avaliar um artefato para modelagem de requisitos de software de tempo-real. Segundo [Hevner et al. 2004] e [McKay e Marshall 2008], um artefato pode referir a instanciações, construções, modelos e métodos utilizados em um processo de desenvolvimento ou utilização de um sistema.

Neste trabalho, os artefatos a serem criados são construções e modelos que serão representados, nos Capítulos 4 e 5, de maneira semiformal a partir de extensões dos *profiles* UML. Os modelos criados serão aplicados em estudo de caso propostos (Capítulos 4 e 5).

1.8.3 Instrumentos de Pesquisa

Conforme destacado em [Easterbrook et al. 2007], um método é um conjunto de princípios organizadores em torno do qual os dados empíricos são recolhidos e analisados. Uma variedade de métodos pode ser aplicado a qualquer problema de pesquisa e é, muitas vezes, necessário utilizar uma combinação de métodos para compreender completamente o problema e conduzir a questões pertinentes e não-refutáveis, em relação aos critérios de pesquisa. Para cada método que será utilizado neste trabalho, será descrito, a seguir, a corrente filosófica ao qual o mesmo está relacionado:

- Positivismo - Esta corrente filosófica afirma que todo conhecimento deve ser baseado em inferência lógica a partir de um conjunto de fatos observáveis básicos. Positivistas preferem métodos que começam com teorias precisas a partir das quais hipóteses verificáveis podem ser extraídas e testadas isoladamente. Portanto, o positivismo é mais associado com o experimento controlado e, por conseguinte, estudos de caso e *surveys* também são frequentemente realizados [Easterbrook et al. 2007].
- Construtivismo - Também conhecido como interpretativismo, rejeita a ideia de que o conhecimento científico pode ser separado de seu contexto humano. Para os construtivistas, a ciência é o processo de busca de teorias locais que emergem e explicam os dados e está estreitamente associado com etnografias, embora os construtivistas frequentemente usem estudos de caso exploratórios e *surveys*, para fundamentar o processo de pesquisa [Easterbrook et al. 2007].

O conhecimento gerado a partir da realização deste trabalho dar-se-á de forma qualitativa, tendo em vista que o mesmo será galgado de experiências, estudos e observações de como as extensões de linguagens de modelagem podem contribuir favoravelmente para a modelagem de requisitos de software críticos e, consequentemente, para seu desenvolvimento.

A técnica de Estudo de Caso será utilizada para validar o trabalho em questão e responder às perguntas de pesquisa. Em geral, a estratégia de pesquisa por estudo de caso é utilizada quando as questões propostas ao estudo são do tipo "como" e "de que modo" [Runeson e Host 2009]. Uma variedade de diferentes fontes de dados são tipicamente utilizadas neste método, como dados qualitativos, incluindo entrevistas e observações, já que oferecem percepções ricas sobre o caso de estudo [Yin 2008].

Os estudos de casos são apropriados para vários objetivos de pesquisa e podem ser empregados em contextos de descrição (tentativa de responder a questionamentos do tipo "o que", "onde", "como"), de explicação (tentativa de exclarecer perguntas do tipo "por que"), de previsão (inclui previsões de estado de curto/longo prazo, comportamentos, eventos) e de controle de processos (inclui tentativas de influenciar atitudes, cognições e comportamentos para um caso particular) [Woodside e Wilson 2003] [Yin 2008].

Em relação à técnica para coleta de dados, para este trabalho pretende-se utilizar abordagens de Métodos Mistos, melhor detalhadas em [Easterbrook et al. 2007]. Dessas, serão empregadas técnicas de coleta de dados por meio de revisões de literatura relacionadas à pesquisa em questão para que, posteriormente, a análise associada a dados qualitativos possa ser realizada. Neste contexto, pretende-se avaliar, qualitativamente, as extensões de métodos de modelagem de software aplicados no contexto de software de tempo-real.

Nas pesquisas qualitativas, a compreensão da pesquisa ocorre segundo a perspectiva dos participantes da situação estudada mostrando, como resultado, a análise e a interpretação dos fenômenos estudados. A pesquisa qualitativa é formulada e direcionada ao longo de seu desenvolvimento e permite explicar os resultados da investigação de uma forma coerente [Hazzan e Dubinsky 2007].

A abordagem qualitativa, utilizada nesta pesquisa, objetiva a construção de um arcabouço teórico que emerge da análise dos dados coletados durante a pesquisa e permite explicar os resultados da investigação de uma forma coerente, sendo, portanto, especialmente adequada à investigação de tópicos que não tenham sido previamente pesquisados [Davy 2009].

1.9 Visão Geral da Dissertação

Este trabalho contribui para a pesquisa e prática em Engenharia de Software, especialmente, como apresentado na Figura 1.1, para a subárea da Engenharia de Software conhecida como Engenharia de Requisitos. É proposta também uma extensão de *profiles* UML - a SysML e o MARTE - e sua integração em um metamodelo para documentação e análise de requisitos de sistemas de tempo-real. A extensão é aplicada em um estudo de caso para um Sistema de Controle de Tráfego.

O texto desta dissertação está organizado nos seguintes Capítulos.

No Capítulo 2, é realizada a fundamentação teórica sobre a pesquisa proposta descrevendo conceitos da engenharia de requisitos, desenvolvimento e modelagem de requisitos, sistemas de tempo-real e conceitos pertinentes à modelagem de requisitos de sistemas de tempo-real. Neste Capítulo, o *profile* SysML e o *profile* MARTE são caracterizados detalhamente, bem como são expostas as informações pertinentes ao diagrama de requisitos da SysML e seus relacionamentos e, também, os elementos de domínio do *profile* MARTE são descritos.

No Capítulo 3, os conceitos relacionados aos Sistemas de Transporte Inteligentes relevantes ao estudo de caso, apresentado nos capítulos posteriores, são apresentados. O foco principal deste capítulo é conceitualizar sobre os sistemas de controle de tráfego e os possíveis requisitos para desenvolvimento de estrátegias de controle destes sistemas.

No Capítulo 4, tem-se como principal objetivo apresentar o metamodelo que estende a SysML, adicionando-lhe características relevantes para especificação de requisitos de

sistemas de tempo-real. A ferramenta para suporte ao metamodelo criado e suas características são descritas e as tabelas da SysML, para prover rastreabilidade de requisitos em diferentes níveis de abstração, também são apresentadas. Finalmente, é realizada a aplicação do metamodelo elaborado em um estudo de caso para um Sistema de Controle de Tráfego.

No Capítulo 5, os conceitos sobre o metamodelo, que combina SysML e MARTE para análise, documentação e especificação de requisitos de tempo-real, são abordados. O conjunto de estereótipos e de conceitos relacionados para definir os conceitos de domínio do *profile* MARTE são apresentados e aplicados em um estudo de caso. O foco deste Capítulo está em apresentar uma abordagem pertinente para a documentação e modelagem de requisitos de STR.

Por fim, no Capítulo 6 são apresentadas as conclusões referentes ao trabalho e alguns desafios que viabilizam trabalhos e pesquisas futuras.

Capítulo 2

Referencial Teórico

No presente capítulo serão abordados conceitos fundamentais do processo de Engenharia de Requisitos e de métodos de modelagem de software. Na Seção 2.1 é apresentada a introdução à Engenharia de Requisitos e alguns conceitos relevantes relacionados aos sistemas de tempo-real são descritos na Seção 2.3. A modelagem de requisitos é abordada na Seção 2.2 e, especificamente, a modelagem de requisitos de sistemas de tempo-real é caracterizada na Seção 2.4. Os conceitos pertinentes à UML e sua capacidade de extensão são descritos na Seção 2.6. Na Seção 2.7 é apresentada a fundamentação teórica do *profile* SysML em que os conceitos principais, os diagramas propostos e o metamodelo para o diagrama de requisitos da SysML são abordados. Na Seção 2.8 o *profile* MARTE é descrito e uma breve caracterização de todos os seus pacotes é realizada. Finalmente, na Seção 2.9, os pacotes *CoreElements*, *Time* e *NFPs* são detalhados.

2.1 Conceitos de Engenharia de Requisitos

De acordo com [Kotonya e Sommerville 1998] e [Sommerville 2011], a Engenharia de Requisitos é uma abordagem disciplinada e sistemática para elicitar, especificar, analisar, confirmar, validar e gerenciar requisitos, considerando usuários, objetivos e necessidades técnicas, econômicas e de negócio. Em [David et al. 2010], a engenharia de requisitos é caracterizada como a etapa inicial da atividade de desenvolvimento de software em que os requisitos do cliente são extraídos e documentados, sendo comumente utilizada para apoiar o processo de definição, compreensão e representação dos requisitos de um software.

A Engenharia de Requisitos é caracterizada, em diferentes bibliografias, como o processo mais crítico e complexo do ciclo de desenvolvimento de software. A principal razão é que o processo de engenharia de requisitos tem impacto dominante sobre as capacidades do produto resultante [Ghezzi et al. 2003]. Além disso, a engenharia de requisitos representa "o processo no qual o conjunto mais diversificado das necessidades de um produto é concebido a partir do conjunto mais diverso de partes interessadas" [Parviainen et al. 2005]. Estas duas razões reafirmam tanto as exigências complexas quanto críticas para a

engenharia de requisitos [IEEE 1990].

"Um requisito bem-formado", ou seja, claramente estabelecido, é uma declaração das funcionalidades do sistema (capacidades) que deve ser atendida para satisfazer às necessidades de um cliente ou para atingir o objetivo de um cliente em relação a um serviço. Os requisitos são qualificados por condições mensuráveis e são limitados por restrições [IEEE 1990]. Conforme [Saiedian e Dale 2000] e [Sommerville 2011], os requisitos podem ser categorizados em :

- **Requisitos de usuário**: definição geral dos requisitos do usuário, em linguagem natural e/ou com diagramas, descrevendo quais serviços e restrições o sistema deve abranger;
- **Requisitos de Sistema**: definem, detalhadamente, as funções, os serviços e as restrições operacionais do sistema. Os requisitos do sistema acrescentam detalhes a cada requisito de usuário, definindo exatamente o que deve ser implementado.
- **Requisitos de Domínio**: são requisitos provenientes do domínio da aplicação do sistema e que refletem as características e restrições desse domínio.

Estes requisitos podem, ainda, ser classificados em dois tipos:

- **Requisitos Funcionais**: declarações de serviços/funcionalidades que o sistema deve fornecer, podendo estabelecer o que o sistema não deve fazer;
- **Requisitos Não-Funcionais**: restrições sobre os serviços ou as funções oferecidas pelo sistema, como as restrições de tempo e, também, sobre o processo de desenvolvimento, desempenho do produto e padrões utilizados. Aplicam-se geralmente ao sistema como um todo e não a características ou serviços individuais de sistemas.

De maneira geral, como descrito em [IEEE 1998], uma especificação de requisitos de software deve fornecer, para os clientes, fornecedores e demais pessoas interessadas, benefícios específicos, tais como:

- Estabelecer a base para um acordo entre os clientes e os desenvolvedores sobre o que o produto de software realizará, pois a partir da descrição completa das funções a serem executadas pelo software especificado é possível que usuários potenciais determinem se a especificação atende ou não as suas necessidades;
- Reduzir o esforço de desenvolvimento;
- Fornecer uma base para estimar custos e cronogramas;
- Fornecer uma base para validação e verificação;
- Servir de base para aprimorar o documento de requisitos.

Um grande número de problemas no desenvolvimento de software pode surgir devido à má especificação de requisitos como atrasos na entrega, custos maiores do que a estimativa original, cliente e usuário final insatisfeitos. Além disso, o sistema pode não ser confiável e pode haver defeitos no sistema como um todo [David et al. 2010]. De acordo com [Ramingwong 2009], muitos problemas associados com o desenvolvimento de software estão diretamente relacionados à elicitação e à análise de requisitos. Tais problemas podem ser causados por má adequação da técnica utilizada para especificação [Kausar et al. 2010], problemas de negociação e/ou baixo envolvimento dos *stakeholders* e demora para realizar a atividade de especificação [Chua et al. 2010]. Segundo [Apshvalka et al. 2009], podem ser considerados como entraves e desafios a uma especificação com qualidade os requisitos incorretos e a subjetividade envolvida na interpretação dos requisitos.

O processo de engenharia de requisitos é muitas vezes tratado como uma atividade demorada, burocrática e contratual e, nesse sentido, observa-se que muitos sistemas entregues não atendem aos requisitos de seus clientes devido, pelo menos em parte, a uma prática de engenharia de requisitos ineficaz [Nuseibeh e Easterbrook 2000].

Em [Boehm 1973], é relatada a importância da corretude em especificações de requisitos e do uso de técnicas, metodologias, abordagens e ferramentas para auxiliar a especificação das propriedades do sistema em questão. [Nuseibeh e Easterbrook 1989] [Lee 2002] [Soares e Vrancken 2007] destacam a importância de utilização, no ciclo de desenvolvimento de projetos de STR, de métodos de especificação de requisitos.

Quando os requisitos são especificados para o software de um STR, uma série de pontos de vista do sistema deve ser fornecido, cada um fornecendo uma perspectiva diferente em suas necessidades totais [Nuseibeh e Easterbrook 1989]. Com isso, tem-se que a análise e a especificação de requisitos são atividades fundamentais ao processo de desenvolvimento de STR e influeciam diretamente o desenvolvimento satisfatório destes sistemas [Cabral e Sampaio 2008]. Por se tratarem de atividades de grande importância no ciclo de vida do software e que se relacionam diretamente com a qualidade do produto a ser desenvolvido, a Engenharia de Requisitos precisa ser devidamente planejada. Sendo assim, deve ser aplicada de uma forma abrangente para assegurar que um conjunto completo das necessidades e requisitos dos usuários sejam capturados e transformados em um conjunto validado de requisitos em todo o ciclo de vida [Carrillo e Nicolas 2011].

2.2 Conceitos de Modelagem de Requisitos

A modelagem é fundamental para os processos da engenharia de requisitos [Booch et al. 1999] [Sommerville 2011]. Existem diversas abordagens para modelagem de requisitos que, basicamente, podem ser classificadas como: representações gráficas, representações textuais ou uma combinação de ambas. Tais representações podem, ainda, compor métodos formais, semi-formais ou informações para especificação de requisitos. Em geral,

os métodos e linguagens de elicitação de requisitos e as ferramentas que lhes apoiam são classificados em três categorias gerais: orientada a objeto, orientada a processos e comportamentais [Carrillo e Nicolas 2011].

- As abordagens **Orientadas à Objeto** organizam os requisitos em termos de objetos do mundo real, representando os atributos e os serviços realizados por esses objetos;
- As abordagens **Baseadas em Processo** organizam os requisitos em hierarquias de funções que se comunicam através de fluxos de dados;
- As abordagens **Comportamentais** descrevem o comportamento observável do sistema em termos de alguma noção abstrata (como cálculo de predicados), funções matemáticas ou máquinas de estado.

O grau com que tais ferramentas e métodos são úteis/aplicáveis na preparação da especificação de requisitos do software depende do tamanho e da complexidade do sistema [Arif et al. 2009]. Os modelos criados devem ser legíveis e representativos, pois as várias partes interessadas têm de compreendê-los [Carrillo e Nicolas 2011].

Na Seção 2.1 foram listadas as características desejáveis para elaborar a especificação de requisitos de software com qualidade. A seguir são listadas algumas características, especificadas a partir da [IEEE 1998], que as metodologias e linguagens de modelagem de requisitos devem tratar. Vale destacar que tais propriedades são válidas para especificação de requisitos em qualquer uma das categorias citadas anteriormente.

- Corretude e Completude;
- Verificabilidade e Modificabilidade;
- Consistência;
- Diminuir/Tratar Ambiguidade;
- Gerar Representações em Conformidade;
- Rastreabilidade;
- Identificar os Tipos de Requisitos;
- Mostrar Prioridade entre os Requisitos;
- Representar Requisitos Não-Funcionais;
- Modelagem Gráfica;
- Legibilidade aos Seres Humanos;
- Gerar *Ranking* dos Requisitos por Estabilidade ou Importância;
- A necessidade de apoiar simultaneamente diferentes níveis de precisão;
- A necessidade de representar múltiplas visões diferentes, mas mutuamente consistentes para certos elementos do modelo.

2.3 Características Relacionadas aos Sistemas de Tempo-Real

A principal característica dos STR é que requisitos funcionais podem estar associados a restrições sejam elas temporais, de segurança e/ou de desempenho. Como estes sistemas são usualmente reativos (o sistema deve reagir a cada estímulo do ambiente) e concorrentes (operam em paralelo), o atendimento a um requisito precisa considerar, também, a satisfação dos requisitos de tempo em um ambiente concorrente em que eventos assíncronos podem ser lançados a qualquer instante [Kavi e Yang 1992] [Pettersson 1999].

Os STR cobrem uma ampla gama de sistemas como, por exemplo, os controladores de automóveis, de telecomunicação industrial, de controle de tráfego aéreo, de controle ferroviário e de controle de tráfego [Welch et al. 1998]. Outros exemplos podem incluir os sistemas médicos, os sistemas nucleares e os sistemas de controle de processos [Liu 2000].

Os STR podem ser classificados em três classes primárias [Kirner e Davis 1996] [Laplante 2006]: Sistemas de tempo-real "*hard*", sistemas de tempo-real "*soft*" e sistemas "*firm*".

Os STR *soft* são aqueles em que os tempos de resposta são importantes, mas o sistema funcionará aceitavelmente se algumas restrições temporais forem, ocasionalmente, perdidas, isto é, os casos em que o desempenho do sistema estiver degradado não acarretarão grandes perdas devido ao não cumprimento das restrições de tempo de resposta [Pettersson 1999]. Podem ser citados como exemplos de sistemas *soft* os de aplicações multimídia, os sistemas interativos e os de reservas [Laplante 2006].

Os STR *hard* (rígidos) são aqueles em que é imperativo que as respostas ocorram de acordo com a restrições de tempo especificadas nos requisitos. Assim, o não cumprimento das restrições de temporização em tais aplicações pode levar a degradações intoleráveis no sistema e, em alguns casos, resultar em perdas catastróficas. Segundo [Pettersson 1999], estes sistemas devem reagir sempre que solicitado, ao contrário de STR *soft* que, eventualmente, podem não conseguir satisfazer as suas necessidades de tempo. Podem ser citados como exemplos de sistemas *hard* os de controle da aeronaves, de controle de usinas nuclear, de controle de tráfego, entre outros [Laplante 2006].

Segundo [Laplante 2006], os STR *firm* podem se comportar como sistemas *hard* ou como STR *soft*. Em um STR *firm*, pequenos atrasos não levam à falha total do sistema (STR *soft*), mas atrasos maiores podem levar a uma falha global e perigosa do sistema (STR *hard*). Podem ser citados como exemplos destes sistemas os de controle de caixa eletrônico e os de controle de transações online.

Em geral, um STR é composto por três elementos principais: o hardware, o sistema operacional de tempo-real (RTOS) e o software [Giese 2003]. O hardware consiste de um ou múltiplos controladores e recursos operacionais tais como memória, sensores e atuadores. O RTOS é a interface entre o hardware e o software e fornece operações para

acesso aos recursos. Em cada operação do sistema, todas as tarefas executam concorrentemente e tentam acessar os controladores e recursos simultaneamente. Logo, o RTOS deve fornecer gerenciadores de tarefas para regular tais acessos. O software é o componente lógico, desenvolvido com critérios de desempenho, segurança e confiabilidade, geralmente embarcado, que determina as funções possíveis ao sistema de tempo-real.

O desenvolvimento de STR de alta qualidade depende de sua correta especificação de requisitos que inclui análise e especificação de suas restrições temporais [Kirner e Davis 1996]. Por conseguinte, a especificação de requisitos de sistemas de tempo-real exige um suporte completo e eficaz para definir, expressar e validar todos os parâmetros temporais e restrições de tempo relacionados com a aplicação que está sendo desenvolvida.

2.4 Propostas para Modelagem de Requisitos de Software de Tempo-Real

Para minimizar a complexidade dos STR são utilizados modelos que de forma gráfica ou textual auxiliam na compreensão e representação destes sistemas. Segundo [Saiedian e Dale 2000], os modelos permitem que os *designers* de diferentes disciplinas possam compartilhar conhecimento e facilitam a compreensão e avaliação do *design* do sistema. Para a criação de modelos para STR é necessário modelar não só a estrutura e o comportamento do software, mas também seus recursos físicos e lógicos, sua infraestrutura e os aspectos temporais [Liu 2000]. Essas características distinguem claramente a diferença entre esforço aplicado e abordagens necessárias para modelagem de sistemas de tempo-real e para sistemas de propósito geral [Gomaa 2001].

De maneira geral, os engenheiros de STR tendem a utilizar uma ou a combinação das seguintes abordagens para modelagem e especificação de requisitos de sistemas com características de tempo-real:

- Decomposição de processos *top-down* ou análise estruturada;
- Abordagens Orientadas a Objeto;
- Métodos Formais;
- Linguagens de descrição de programas (PDL) ou pseudocódigo;

Representações informais não são suficientemente precisas e expressivas para modelagem de STR, nas quais o formalismo dos diversos requisitos funcionais e não-funcionais devem ser representados. É arriscado prescrever uma técnica preferida, porque é bem conhecido que não há "bala de prata" em Engenharia de Software, ainda mais quando se trata de especificação e *design* de software [Brooks 1987]. No entanto, independentemente da abordagem, a modelagem de STR deve incorporar as seguintes práticas:

- Usar abordagens de modelagem e técnicas consistentes ao longo da especificação, por exemplo, decomposição *top-down*, estruturada ou abordagens formais e/ou Orientadas a Objetos;
- Utilizar níveis de abstração consistentes dentro dos modelos e em conformidade entre os níveis de refinamento dos modelos;
- Modelos de requisitos não-funcionais devem ser tratados como uma parte dos modelos de especificação, em particular, a modelagem de propriedades de tempo;
- Omitir atribuições de hardware e software na especificação (aspectos de projeto).

Várias abordagens para modelagem de software têm sido criadas nas últimas décadas para modelagem de STR. Podem ser citadas a análise estruturada com extensões para tempo-real, as abordagens formais e as abordagens Orientadas a Objetos como sendo as três abordagens principais [Burns e Wellings 2001] [Laplante 2006]. Nas seções posteriores os principais métodos semiformais utilizados no contexto de modelagem de software de tempo-real são descritos.

2.5 UML para Modelagem de Requisitos de Software de Tempo-Real

A UML foi originalmente desenvolvida para criar modelos de sistemas de software de propósito geral [Booch et al. 2007]. A palavra unificada representa a afirmação de que a linguagem pode ser usada para sistemas de software, como também para um grande número de domínios diferentes do desenvolvimento de software, incluindo processos de negócios e produtos de software [Briand et al. 2003].

A UML utiliza mecanismos de extensão controlados (estereótipos), permitindo adaptá-los a um domínio específico e define um conjunto de anotações para expressar a modelagem de sistemas. Nesse sentido, há diferentes visões na UML que são representadas por vários diagramas: o de caso de uso, de classes, de objetos, de sequência, de comunicação, de estados, de componentes e de implantação [Booch et al. 2007].

A UML, apesar de possuir diversos modelos para representar projetos de sistemas de tempo-real, por si só não consegue lidar com completude com as especificações desses softwares. As restrições temporais, controle, sincronização e paralelismo de processos [Booch et al. 2007] constituem algumas dessas complexidades.

Segundo [Bennett e Field 2004], a UML representa a estrutura do sistema e seu comportamento em muitos níveis diferentes de abstração. No entanto, o tempo é raramente parte da modelagem comportamental, uma vez que o mesmo é essencialmente de duração indeterminada. Em mecanismos de avaliação dinâmica ou em aspectos estruturais do projeto, o tempo também não é modelado. Logo, segundo [Bennett e Field 2004], esta

propriedade não é bem representada na modelagem UML. [Silvestre e Soares 2011] destaca que a UML apresenta dificuldade em expressar as propriedades não-funcionais do sistema, requisitos frequentemente importantes para aplicações de tempo-real e, ainda, que a UML carece de mecanismos para validação e verificação de softwares.

Modernos STR têm certas características que demandam novas abordagens para sua especificação, *design* e implementação. A modelagem de tais sistemas requer um conjunto de notações heterogêneas que refletem: tempo contínuo, concorrência, fluxo de controle, eventos discretos e/ou reativos. Neste contexto, várias propostas para tratar os problemas da UML em relação à modelagem de softwares de tempo-real foram criadas. Diversos *profiles* que estendem a UML e adicionam elementos que modelam requisitos de tempo, de sistema e propriedades não-funcionais foram criados: o SPT [OMG 2005], a SysML [OMG 2010], o QoS [OMG 2008] e o MARTE [OMG 2011a].

A conceitualização relacionada ao termo *profiles* e aos *profiles* SysML e MARTE, devido à sua importância para o presente trabalho, serão abordados separadamente e em maior nível de detalhe nas seções subsequentes.

2.6 Profiles UML para Modelagem de Sistemas de Tempo-Real

Desde o início de seu projeto, a UML foi designada a ser personalizável, como uma família potencial de linguagens [OMG 2011b]. Sua definição inclui muitas variáveis semânticas e também fornece construtores especiais na linguagem para refinamento. Tais construtores, estereótipos, valores rotulados e *tags*, são utilizados para definir linguagens de modelagem de domínio específico (DSML) baseadas em UML. Uma vez que capturam conceitos específicos do domínio, eles tipicamente são usados em conjunto com outros estereótipos específicos do mesmo domínio. Isto levou à concepção dos nomeados *profiles* UML que contém uma coleção de estereótipos relacionados.

Os *profiles* geralmente baseiam-se em apenas um subconjunto do metamodelo UML (em oposição ao metamodelo completo), resultando em DSMLs mais simples e compactas [Selic 2007]. Como vantagem da utilização dos *profiles*, pode ser citado o fato de que os *profiles* são baseados na especialização da semântica e de conceitos gerais da UML e podem, portanto, tirar proveito das ferramentas UML e *expertise* já existentes [Espinoza et al. 2009].

Diversos *profiles*, como o SPT, a SysML, o QoS e o MARTE, que estendem a UML e adicionam elementos que modelam requisitos de tempo, de sistema e de propriedades não-funcionais foram criados ao longo do tempo.

Os *profiles* SysML e MARTE, foco deste trabalho, serão detalhados nas seções posteriores de modo a elucidar seus conceitos fundamentais e possibilitar a criação de um

framework de modelagem de requisitos de software de tempo-real que combine os *profiles* SysML e MARTE.

2.7 Caracterização da SysML

A linguagem SysML [OMG 2010] permite a modelagem de propósito geral para diversos tipos de aplicações em engenharia de sistemas, possibilitando a especificação, análise, projeto, verificação e validação de sistemas complexos.

A SysML baseia-se na UML e reutiliza algumas das suas construções. A diferença básica é que SysML foi construída para apoiar a engenharia de sistemas, o que significa que algumas construções específicas, orientadas a software, não necessárias para a modelagem de sistemas, foram evitadas em seu metamodelo. A SysML permite a criação de diversos estereótipos aplicáveis em diferentes domínios como: elicitação e validação de requisitos, especificação de arquitetura do sistema, projeto de sistemas e verificação e validação de uma vasta gama de sistemas complexos [OMG 2010]. Como pode ser observado na Figura 2.1, a SysML herda alguns dos diagramas do metamodelo UML e também propõe diagramas adicionais.

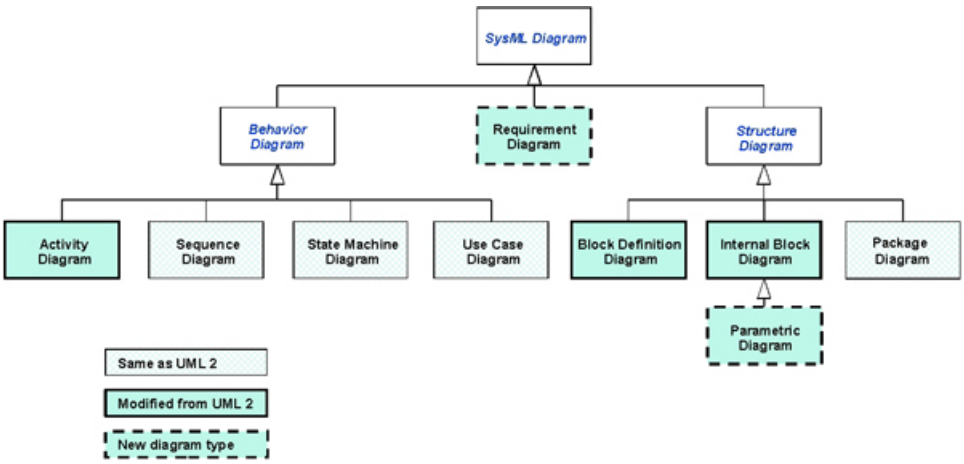

Figura 2.1: *Profile* SysML [OMG 2010].

Na Figura 2.1 foi acrescentada uma legenda para representar os diagramas que são herdados e utilizados, na SysML, sem nenhuma alteração em seu metamodelo. Alguns diagramas foram herdados e novos conceitos adicionados para a modelagem. Como pode ser observado na Figura 2.1, novos diagramas, como o paramétrico e o de requisitos, foram acrescentados no metamodelo da SysML. De um modo geral, podem ser citadas como contribuições gerais da SysML em relação a UML os seguintes aspectos:

- **Blocos e Fluxos**: Descrição de blocos e diagramas de blocos interno da SysML permitem a especificação de interações mais genéricas e fenômenos que existem apenas em sistemas de software;

- **Comportamento**: Embora a maioria das construções de comportamento em SysML sejam semelhantes a UML (interações, máquinas de estado, atividades e casos de uso), a SysML refina alguns deles para a modelagem de sistemas contínuos e probabilidades no diagramas de atividades;
- **Modelagem de Requisitos**: A SysML fornece uma facilidade explícita para modelagem de requisitos de sistema juntamente com sua rastreabilidade em relação à evolução da arquitetura. Isto pode ser especificado em seu formato gráfico ou tabular;
- **Paramétricos**: O diagrama paramétrico permite aos usuários da SysML representar, de uma maneira gráfica, relacionamentos analíticos e restrições, geralmente descritas por equações matemáticas. O diagrama paramétrico é um mecanismo para integrar modelos de *design* SysML com análise de engenharia.

2.7.1 Diagrama de Requisitos da SysML

Para possibilitar a modelagem de requisitos, a SysML, como mostrado na Figura 2.2, fornece um novo diagrama que viabiliza a modelagem de requisitos e seus relacionamentos (Figura 2.3) com outros elementos do modelo. Na especificação da SysML [OMG 2010], um requisito é definido de acordo com a Figura 2.2.

O diagrama de requisitos da SysML mostra explicitamente os vários tipos de relacionamentos entre diferentes requisitos. Os construtores de modelagem do diagrama de requisitos da SysML são destinados a fornecer um ponto de interligação entre modelos de arquitetura e as ferramentas de gerenciamento de requisitos tradicionais.

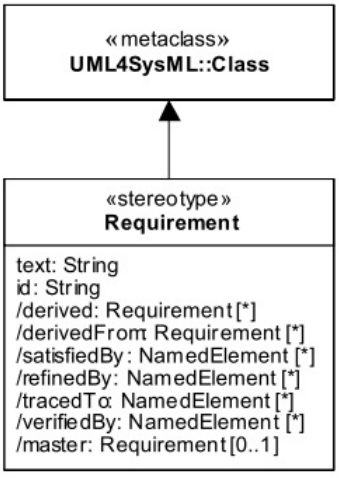

Figura 2.2: Modelo de Requisitos da SysML [OMG 2010].

No diagrama de requisitos um requisito é representado por um estereótipo << *Requirement* >> do elemento de classe. Sendo assim, um requisito é uma classe estereotipada.

Isto significa que um requisito tem a semântica de uma classe estendida por uma semântica específica e as propriedades/atributos do estereótipo. Os atributos do modelo de requisitos da SysML possuem os seguintes significados:

- text: String

  Uma string simples que inclui a descrição textual do requisito;

- id: String

  Uma string simples destinada a ser a identificação única do requisito;

- /derived: Requirement [*]

  Indica o requisito que foi derivado (cliente) a partir desse requisito (fornecedor);

- /derivedFrom: Requirement [*]

  Indica a partir de quais requisitos (fornecedores) esse requisito foi derivado (cliente);

- /satisfiedBy: NamedElement [*]

  Indica os elementos (clientes) que satisfazem esse requisito (fornecedor);

- /refinedBy: NamedElement [*]

  Indica os elementos (clientes) que refinam esse requisito (fornecedor);

- /tracedTo: NamedElement [*]

  Indica todos os elementos (fornecedores) que são rastreáveis com esse requisito (cliente);

- /verifiedBy: NamedElement [*]

  Indica todos os elementos (clientes) que verificam esse requisito (fornecedor);

- /master: Requirement [0..1]

  Essa é uma propriedade derivada que lista o requisito mestre para esse requisito escravo. O atributo mestre é derivado a partir do fornecedor que possui uma dependência *Copy* (cópia) com o requisito escravo.

2.7.2 Relacionamentos do Diagrama de Requisitos da SysML

O diagrama de requisitos da SysML oferece diferentes maneiras para representar os possíveis relacionamentos entre requisitos. Os relacionamentos podem ser observados na Figura 2.3 e uma breve descrição sobre cada um deles é realizada a seguir:

- **DeriveReqt**: O relacionamento *deriveReqt* é representado pelo estereótipo << *deriveReqt* >>. Ele descreve que um requisito (cliente) foi derivado a partir de um outro requisito (fornecedor). Portanto, este relacionamento mostra quando um requisito resulta de outro como, por exemplo, em casos em que um requisito do sistema

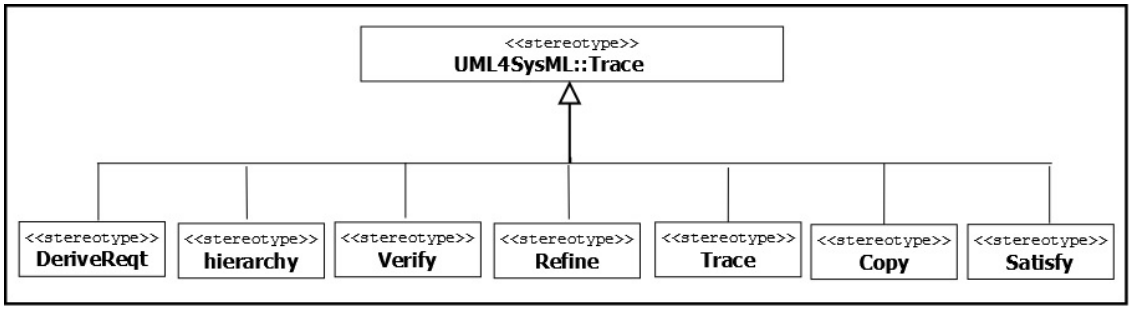

Figura 2.3: Relacionamentos SysML [OMG 2010].

pode ser derivado a partir de uma necessidade de negócio, ou requisitos mais específicos podem ser derivados a partir de um requisito de sistema. O relacionamento *deriveReqt* pode ser observado na Figura 2.4.

Figura 2.4: Relacionamento *deriveReqt* [OMG 2010].

- **Hierarchy**: O relacionamento *Hierarchy* (hierárquico), na Figura 2.5, descreve um requisito que é derivado hierarquicamente de outro requisito. Este relacionamento permite relacionar requisitos em diferentes níveis hierárquicos, possibilitando representar a hierarquia de requisitos mais gerais com seus respectivos requisitos específicos (mais detalhados). Por exemplo, requisitos de negócio, em alto nível, podem ser gradualmente decompostos em requisitos mais detalhados, formando assim uma hierarquia.

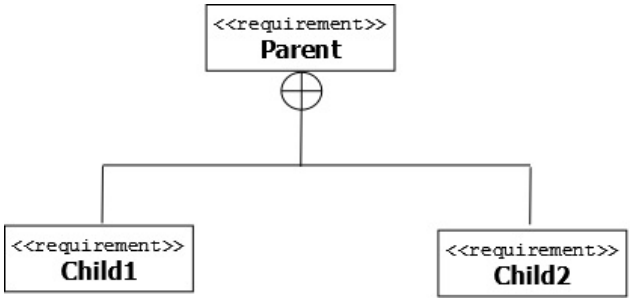

Figura 2.5: Relacionamento *hierarchy* [OMG 2010].

- **Verify**: Um modelo de teste geralmente define um grande número de casos de teste que checam se os requisitos são implementados corretamente no sistema. Com isso, o relacionamento *Verify* (verificar), apresentado na Figura 2.6, objetiva mostrar a conexão de um caso de teste com o requisito que é verificado por este caso de teste.
- **Refine**: O relacionamento *Refine* (refinar) especifica que um elemento do modelo descreve as propriedades de um requisito em mais detalhes. Isso significa que um

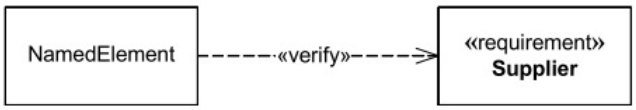

Figura 2.6: Relacionamento *verify* [OMG 2010].

caso de uso pode ser refinado por um ou mais requisitos funcionais. O relacionamento *Refine* é mostrado na Figura 2.7.

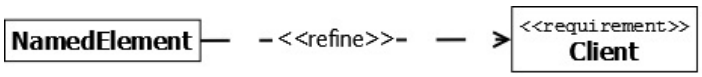

Figura 2.7: Relacionamento *refine* [OMG 2010].

- **Trace**: O relacionamento *Trace* (rastreamento) é um relacionamento de propósito geral entre um requisito e qualquer outro elemento do modelo. O relacionamento *trace* é geral e sua semântica descreve um relacionamento genérico por razões de rastreabilidade apenas. Ele expressa que os elementos do modelo (clientes) têm um relacionamento com um requisito (fornecedor), mas sem especificar essa relação em mais detalhes. A representação deste pode ser visualizada na Figura 2.8.

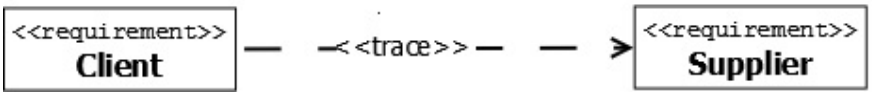

Figura 2.8: Relacionamento *trace* [OMG 2010].

- **Copy**: O relacionamento *Copy* (cópia) descreve um requisito que é uma cópia de outro requisito e mantém um relacionamento de mestre/escravo entre os dois elementos. Este relacionamento aplica-se quando há uma necessidade na modelagem de requisitos de reutilização de um requisito particular em outro contexto. O relacionamento *Copy* pode ser visualizado na Figura 2.9.

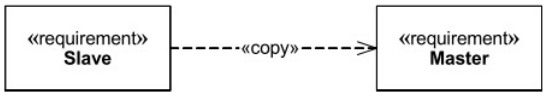

Figura 2.9: Relacionamento *copy* [OMG 2010].

- **Satisfy**: Cada elemento do *design* do modelo tem a finalidade de satisfazer um requisito do sistema. Assim, o relacionamento *Satisfy* (satisfazer) descreve que um elemento do *design* executa/satisfaz um requisito particular. O relacionamento *Satisfy* pode ser observado na Figura 2.10.

O foco desses relacionamentos é descrever como os requisitos relacionam-se uns aos outros e, também, os relacionamentos entre os diversos elementos do modelo, tais como os casos de uso. Estes são de grande importância para a especificação de requisitos, pois,

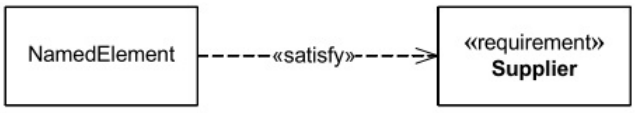

Figura 2.10: Relacionamento *satisfy* [OMG 2010].

além de descreverem os contatos entre requisitos do sistema e os diversos elementos do modelo, promove o gerenciamento de mudanças e evolução e, ainda, gerenciamento de rastreabilidade dos requisitos especificados.

2.8 Caracterização do *Profile* MARTE

MARTE [OMG 2011a] é um *profile* UML para modelagem e análise de sistemas de tempo-real e embarcados que fornece um amplo suporte para especificação, *design*, verificação e validação de sistemas complexos. Conforme fundamentando em [OMG 2011a], o *profile* MARTE tem como vantagens o fornecimento de uma maneira comum de modelar aspectos de hardware e software, a melhoria de comunicação entre desenvolvedores, a padronização e interoperabilidade entre desenvolvedores de ferramentas e, ainda, o fornecimento de construtores que permitam a criação de modelos que podem descrever aspectos quantitativos, temporais e restritivos dos sistemas.

A arquitetura do *profile* MARTE é apresentada na Figura 2.11. Ela consiste em três pacotes principais nomeados como **Pacote de Fundamentos MARTE** (*pacote MARTE Foundations*) que tem por objetivo definir conceitos fundamentais para sistemas embarcados de tempo-real, o **Modelo de *Design* MARTE** (*pacote MARTE Design Model*) que fornece suporte necessário para realizar uma especificação detalhada de um projeto de sistemas embarcados de tempo-real e o **Modelo de Análise MARTE** (*pacote MARTE Analysis Model*) que oferece conceitos para verificação e apresenta diversos domínios em que a análise é realizada, baseando-se em diversos comportamentos do software tais como desempenho, escalonabilidade, consumo de energia, memória, confiabilidade, disponibilidade e segurança [OMG 2011a].

Como pode ser observado na Figura 2.11, o *profile* MARTE possui, para cada um dos pacotes descritos anteriormente, outros subpacotes. O **Pacote de Fundamentos MARTE** e seus subpacotes são descritos a seguir.

- **Elementos Principais** (*Core Elements*): Este sub-pacote possui os elementos básicos para a modelagem comportamental e para a representação semântica de seu tempo de execução;
- **Propriedades não-funcionais** (*Non-Functional Properties Modeling - NFPs*): Este sub-pacote oferece caminhos para especificar as propriedades não-funcionais dos sistemas de tempo-real, como o uso de memória e consumo de energia. Ele tam-

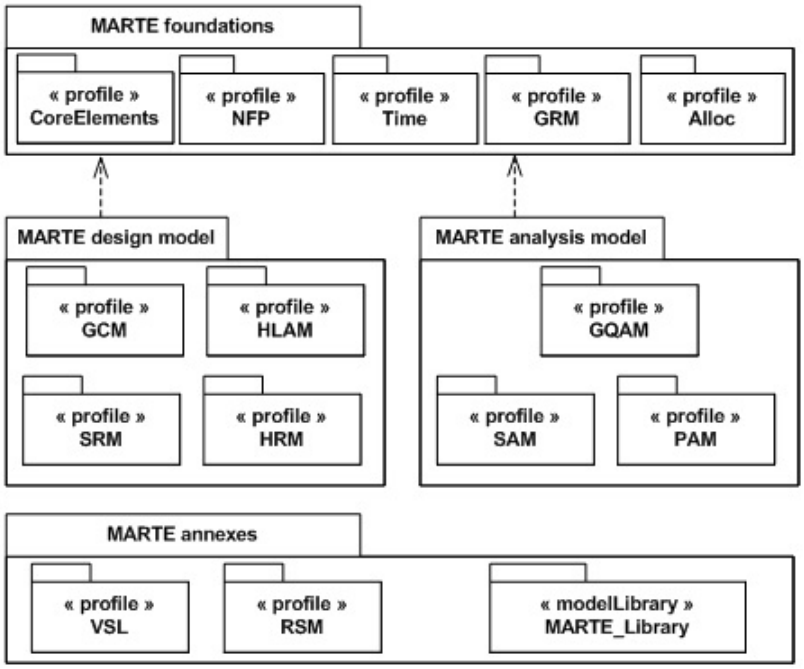

Figura 2.11: *Profile* MARTE [OMG 2011a].

bém explica como as NFPs podem ser ligadas aos elementos do modelo, fornecendo assim a capacidade de encapsular ricas anotações dentro de valores não-funcionais;

- **Modelagem de Tempo** (*Time Modeling* - *Time*): Este pacote permite a modelagem de tempo e estruturas relacionadas com o tempo;
- **Modelagem de Recursos Genéricos** (*Generic Resource Modeling* - *GRM*): Este sub-pacote fornece todos os estereótipos necessários e valores marcados para representar recursos como, por exemplo, meios de comunicação, recursos de computação e recursos de armazenamento. Além disso, inclui recursos que são necessários para lidar com a modelagem de plataformas de execução com diferentes níveis de abstração e modelagem. O pacote GRM junto com o pacote de modelagem de tempo pode ser usado para especificar restrições de tempo e, quando utilizado com o pacote NFP, pode ser usado para especificar a qualidade dos serviços;
- **Modelo de alocação** (*Allocation Modeling* - *Alloc*): o *profile* MARTE permite aos projetistas modelar os aplicativos e as plataformas de execução. Um elemento de aplicação em MARTE pode ser um serviço, a computação ou uma função de sistema operacional de tempo-real (RTOS). Uma plataforma de execução é um conjunto de recursos ligados representando a arquitetura de hardware. Ela consiste em $<< HW - Recursos >>$ como recursos de armazenamento (RAM, ROM ou Cache), dispositivos de comunicação (barramento e dispositivos I/O) e recursos de computação (processador, acelerador de hardware).

Os conceitos definidos no pacote de fundamentos são refinados, para propósito de projeto, nos dois pacotes seguintes: o *MARTE Design Model* (Modelo de Projeto MARTE)

e no *MARTE Analysis Model* (Modelo de Análise MARTE), fornecendo suporte a modelagem e análise de sistemas de tempo-real.

O Pacote de Fundamentos MARTE e a descrição detalhada de seus subpacotes *CoreElements*, *NFP* e *Time* são importantes no âmbito deste trabalho, pois possibilitam, entre outros aspectos, a análise e modelagem dos requisitos não-funcionais e de tempo dos STR.

2.9 Estrutura do *Profile* MARTE

Os pacotes apresentados nas subseções a seguir fazem parte do **Pacote de Fundamentos MARTE** e serão descritos em um nível de detalhe maior dada a sua utilização no metamodelo proposto neste trabalho.

2.9.1 SubPacote *CoreElements*

Os conceitos definidos em *CoreElements* servem como uma base geral para a descrição da maioria dos elementos do restante desta especificação. Tais conceitos formam um conjunto abrangente de conceitos úteis para definir outros mais elaborados e que são utilizados para construir as cláusulas (pacotes) subsequentes da especificação MARTE. Conforme apresentado na Figura 2.12, o pacote *CoreElements* é dividido em dois subpacotes principais.

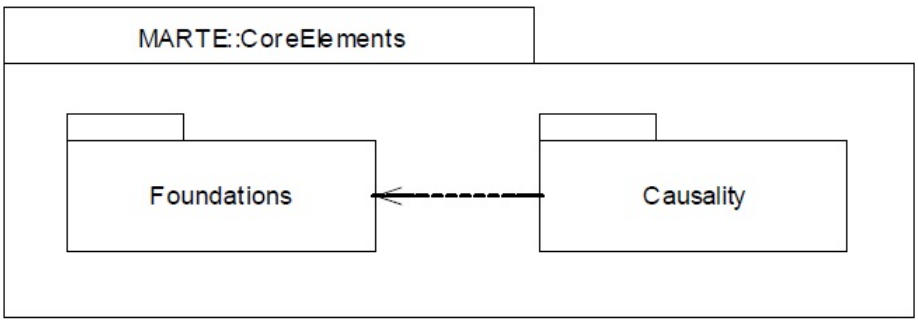

Figura 2.12: Dependência entre Pacotes para o Pacote CoreElements [OMG 2011a].

O pacote *Foundations* é empregado para representar elementos classificadores de *design*-tempo, como as classes/tipos e os elementos de instância de tempo de execução que são criados a partir dos classificadores. Os modelos de domínio apresentados no pacote *Foundations* fornecem um conjunto consistente de elementos de modelagem e formam um grande metamodelo que cobre todas as exigências impostas pela modelagem de propriedades não-funcionais. Os valores das propriedades não-funcionais podem ser anotados em qualquer elemento do modelo, neste caso, por meio de um *AnnotatedElements*.

O objetivo do pacote *Causality* é fornecer uma visão de alto nível da semântica de tempo de execução para os elementos de modelagem, adequados aos STR, e que serão utilizados, se necessário, para destacar os vários elementos cobertos e especializados nos mo-

delos de domínio da especificação MARTE. O pacote *Causality* é utilizado como base para qualquer modelo dinâmico com descrição associada ao *profile* MARTE, sendo composto por quatro subpacotes internos: o *CommonBehavior*, o *RunTimeContext*, o *Invocation* e o *Communication*.

O Pacote *CommonBehavior* é formado pelo Modelo *Basic Behavior* e pelo Modelo *Modal Behavior*. O **Modelo *Basic Behavior*** define os principais conceitos para descrever o comportamento básico dos elementos do modelo e representa uma base conceitual para extensões adicionais necessárias a sistemas de tempo-real e embarcados. O **Modelo *Modal Behavior*** apresenta um novo tipo de comportamento relacionado à modelagem de tempo para sistemas de segurança crítica. Um *Modal Behavior* aborda a noção de modo operacional e pode representar um estado do sistema que é gerenciado por mecanismos de reconfiguração de acordo com as condições de falha (questões de tolerância a falhas de um sistema de gerenciamento) ou uma fase de uma operação de sistema. O modelo *Modal Behavior* especifica um classificador *Mode* que identifica um segmento operacional dentro do sistema de execução que é caracterizado por uma determinada configuração. Um *ModeBehavior* (*Causality:: ModalBehavior:: BehavioredClassifier*) especifica um conjunto de modos mutuamente exclusivos, ou seja, apenas um modo pode ser ativado em um dado instante de tempo e, também, um *BehavioredClassifier* que pode estar ativo em zero ou mais modos operacionais. A dinâmica dos modos é representada por transições definidas por *ModeTransitions* que descreve o sistema modelado sob a comutação de modos.

O Pacote ***RunTimeContext*** descreve mecanismos de especificação de comportamento para os elementos do domínio e representa as ocorrências de eventos observáveis resultantes da execução de uma possível situação de execução comportamental.

O Pacote ***Invocation*** apresenta os conceitos de domínio relacionados a eventos que podem causar a execução de um comportamento. Os *Events* podem ocorrer a partir da invocação direta de um comportamento que ocorre através de uma ação ou de uma ocorrência de disparo (*trigger* - que representa uma invocação indireta de um comportamento) ou através de uma chamada de operação.

O Pacote ***Communication*** acrescenta a infraestrutura de comunicação entre as instâncias do classificador e especifica a semântica geral de comunicação entre as unidades concorrentes.

2.9.2 SubPacote *Non-functional Properties Modeling* (NFPs)

O subpacote NFP descreve o modelo de domínio para a especificação e a representação de propriedades não-funcionais (NFPs) e descreve como NFPs podem ser ligadas a elementos de modelagem UML.

O *framework* de modelagem NFP fornece a capacidade para descrever vários tipos de valores relacionados a quantidades físicas, como tempo, massa e energia. Estes valores

são usados para descrever a arquitetura, comportamento e propriedades, por meio de elementos do modelo, de um modelo para um sistema de computação.

No contexto das abordagens *Model-Driven Development* (MDD) para sistemas de tempo-real, a modelagem das NFPs é de fundamental relevância e implica em uma série de decisões de projeto. O *framework* de anotação NFP tem diversos aspectos que são agrupados em três subpacotes fundamentais, nomeados como *NFP_Nature*, *NFP_Anotation* e *NFP_Declaration*. O subpacote NFP pode ser visualizado na Figura 2.13.

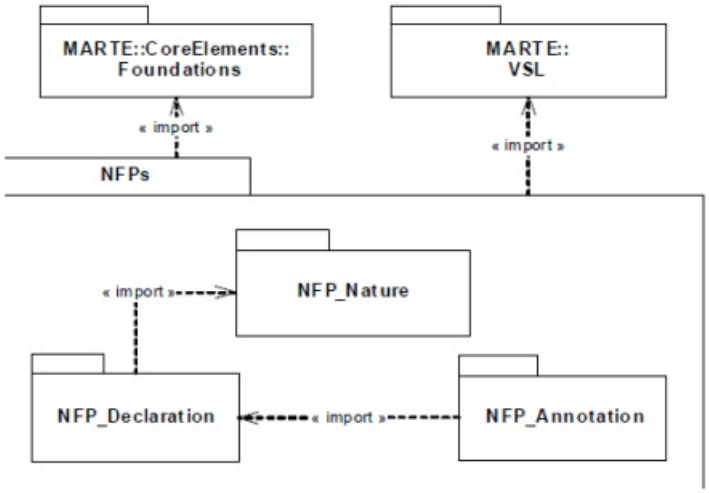

Figura 2.13: Estrutura e Dependência do Pacote de Modelagem NFPs [OMG 2011a].

O Pacote ***NFP_Nature*** possibilita descrever informações qualitativas e/ou quantitativas de uma propriedade não-funcional. As *QuantitativeNFPs* são propriedades NFPs mensuráveis, podendo ser caracterizadas por um conjunto de *SampleRealizations* (representam um conjunto de valores que ocorrem para uma *QuantitativeNFP*) e *Measures* (definida por uma função estatística, por exemplo, média, max, min, que caracteriza o conjunto de realizações da amostra). As *QualitativeNFP* referem-se a características inerentes ou distintivas que não podem ser medidas diretamente. Geralmente, uma *QualitativeNFP* assume um valor a partir de uma lista de valores permitidos, em que cada valor identifica uma possível alternativa.

O Pacote ***NFP_Anotation*** descreve como adicionar características/conceitos de interesse específicos em uma anotação NFP (expressa em UML), por meio de extensões com semântica adicional, a mecanismos de modelagem padrão, permitindo que restrições possam ser associadas a elementos do modelo. Uma *NFP_Constraint* é uma condição (expressão booleana) sobre as propriedades não-funcionais associadas aos elementos do modelo. Em geral, *NFP_Constraints* são afirmações que indicam restrições que devem ser satisfeitas por um sistema de tempo-real e qualificam as restrições NFP pela **natureza requerida** (os valores especificados na *NFP_Constraint* indicam o nível mínimo quantitativo ou qualitativo que os elementos restringidos demandam), **natureza oferecida**

(estabelecem o espaço de valores NFPs que podem apoiar um elemento do modelo) ou **natureza contratada** (definem expressões condicionais que especificam as relações entre requisições e as ofertas de valores não-funcionais) por meio da enumeração *ConstraintKind*.

Um elemento anotado (*AnnotatedElement*) descreve alguns de seus aspectos não-funcionais (aqueles que estão diretamente relacionados aos interesses da anotação) por meio de anotações de valor NFP. Estas anotações, especificadas por meio dos construtores do pacote ***Declaration***, são especificadas pelo projetista nos modelos e anexadas aos elementos do modelo. Assim, o pacote ***Declaration*** destina-se a qualificar e atribuir tipos de dados estendidos aos valores NFP.

2.9.3 SubPacote *Time*

O subpacote *Time* descreve um *framework* geral para a representação de tempo e de conceitos/mecanismos apropriados para modelagem de aspectos de sistemas de tempo-real. Este subpacote descreve diversas características inerentes aos STR como atrasos, duração de eventos, *clocks*, tempo cronométrico e modelos de tempo lógico.

O tempo pode ser diferentemente percebido nas diferentes fases (modelagem, *design*, análise de desempenho, análise de escalonabilidade, implementação, entre outros) do desenvolvimento de um sistema de tempo-real. Como definido a seguir, existem três classes principais de abstração de tempo utilizadas para representar os fluxos de comportamento.

- **Causal/Temporal**: estes modelos preocupam-se apenas com instruções de precedência/dependência;
- **Cronometrado/Síncrona**: esta classe de abstração de tempo acrescenta uma noção de simultaneidade e divide a escala de tempo em uma sucessão de instantes discretos;
- **Física/tempo-real**: esta classe de abstração de tempo representa a modelagem precisa de valores de tempo-real de duração sendo utilizadas para agendamento de questões em sistemas críticos.

Na modelagem de sistemas de tempo-real o tempo não deve ser considerado como um modelo externo, tendo em vista que tempo e comportamento são fortemente acoplados. O modelo de domínio *Time* identifica conceitos que se relacionam com tempo e comportamento, enriquecendo a visão do pacote *Causality* com referências explícitas ao tempo. Para capturar a influência do tempo sobre os comportamentos, os objetos, as execuções de comportamento e as ocorrências de eventos podem referir explicitamente a *clocks*, que são considerados acessadores para a estrutura do tempo.

Os principais conceitos tratados no subpacote de domínio *Time* são representados esquematicamente na Figura 2.14 sendo divididos em quatro grupos distintos e podem

representar os seguintes conceitos: **conceitos para modelar uma forma simples de tempo estruturado** como um conjunto totalmente ordenado de instantes pertencentes a uma base de tempo (como mostrado na Figura 2.14 preocupação de *TimeStructure*), **conceitos para modelagem de bases de tempo múltiplas** (preocupação de *TimeStructure* como representado na Figura 2.14), **conceitos de acesso à estrutura de tempo** incluindo *clocks* e valores de tempo (preocupação de *TimeAccess* e *TimeValueSpecification* Figura 2.14) e **conceitos para modelar entidades ligadas ao tempo** (preocupação de *TimeUsage*).

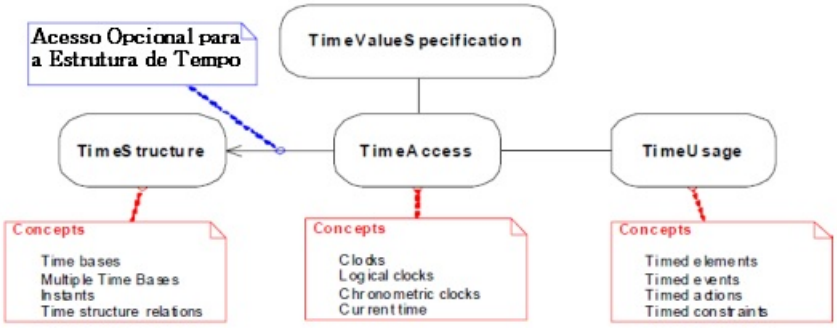

Figura 2.14: Visão Geral dos Interesses do SubPacote *Time* [OMG 2011a].

O modelo de domínio do subpacote *Time* pode ser observado na Figura 2.15 e é formado pelos pacotes ***BasicTimeModels*** (fornece uma visão estrutural do tempo como um conjunto ordenado de instantes), ***MultipleTimeModels*** (fornece uma estrutura de tempo composta por uma ou muitas bases de tempo), ***TimeAccesses*** (introduz os conceitos para representar as estruturas de acesso ao tempo lógico ou cronométrico (*clock*), conceitos para definir valores de instantes de tempo (*TimeValue*) e, ainda, conceitos relacionados a *clocks* ligados ao tempo físico (*ChronometricClocks*) e ***TimeRelatedEntities*** (descreve como relacionar eventos, ações e mensagens ao tempo). Neste pacote, o conceito de *TimedElement* é definido para modelar a associação de um elemento do modelo a um conjunto não vazio de *clocks*).

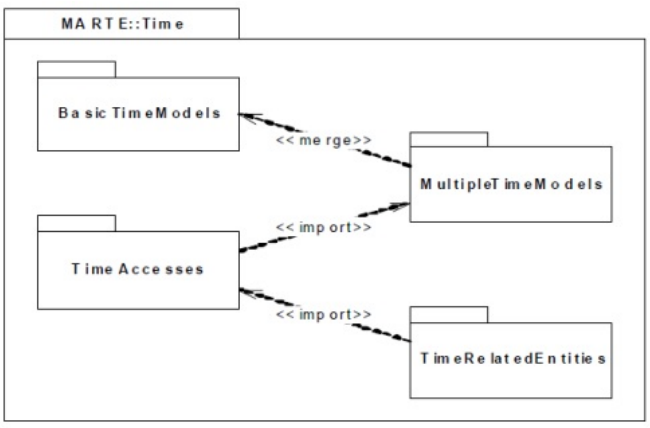

Figura 2.15: Estrutura do Modelo de Domínio de *Time* [OMG 2011a].

Os conceitos de domínio apresentados nas seções anteriores são estendidos para serem utilizados de acordo com as características a serem representadas na modelagem e com a semântica pretendida. Os conceitos definidos nesta visão de domínio são concretizados em elementos de modelagem MARTE, com semântica explícita e direta ou indiretamente associadas a sistemas de tempo-real, e apresentados detalhadamente no Capítulo 3.

Capítulo 3

Requisitos para o Estudo de Caso em Sistemas de Controle de Tráfego

No presente capítulo descreve-se, nas Seções 3.1 e 3.2, o domínio do problema em que se aplica o estudo de caso proposto nos capítulos posteriores. As políticas e estratégias de controle para sistemas de controle de intersecções são abordadas na Seção 3.3. Este capítulo fundamentará, também, um conjunto consistente de requisitos para um Sistema de Controle de Tráfego Urbano que é formado por um conjunto de requisitos em alto-nível (Seção 3.4), requisitos funcionais (Seção 3.5) e requisitos não-funcionais (Seção 3.6).

3.1 Contextualização dos Sistemas Inteligentes de Tráfego

Rodovias foram originalmente concebidas como de acesso ilimitado, de fluxo livre e instalações com pouca necessidade de controle de tráfego. O rápido crescimento do volume de tráfego e o congestionamento das rodovias resultaram na necessidade de projetos para garantir segurança e fluidez aos condutores e pedestres que utilizam o sistema de tráfego [Klein 2006].

Uma alternativa à construção de rodovias novas e com alto custo de construção é a implementação de estratégias que promovam uma utilização mais eficiente das vias atuais. Essas estratégias são encontradas em Sistemas de Transporte Inteligentes (ITS) das rodovias e em programas de trânsito que têm entre seus objetivos reduzir o tempo de viagem, diminuir atrasos e congestionamentos, melhorar a segurança e reduzir as emissões poluentes.

Os ITS tem por objetivo a otimização do transporte em geral, o que envolve a otimização da estrutura de transporte de pessoas e bens. Os ITS fazem uso de uma vasta gama de tecnologias para a criação de meios de comunicação, veículos e vias mais inteligentes aumentando, assim, a fluidez no transporte de pessoas e mercadorias [Gordon 2003].

Existe uma gama de aplicações no âmbito de ITS que podem ser divididas em seis categorias [Sussman 2005].

- Sistemas Avançados de Gerenciamento de Tráfego (ATMS);
- Sistemas Avançados de Informação ao Viajante (ATIS);
- Sistemas Avançados de Controle de Veículos (AVCS);
- Sistemas de Operações de Veículos Comerciais;
- Sistemas Avançados de Transporte Público (APTS);
- Sistemas Avançados de Transporte Rural (ARTS).

O estudo de caso proposto neste trabalho abordará questões relacionadas à especificação de requisitos de Sistemas de Gerenciamento de Tráfego (ATMS) e, por essa razão, estes sistemas serão detalhados a seguir.

3.2 Sistemas de Controle de Tráfego Urbano

Na intersecção de duas ou mais vias sem o controle de tráfego, os motoristas deverão decidir por eles mesmos quando devem seguir ou quando devem esperar. Eles apenas se orientam pelas regras de direção, pelas leis do estado e, ainda, sua cortesia e cooperação. Para intersecções em que o fluxo de tráfego é menor e as condições são favoráveis, os motoristas não têm problemas em atribuir seu próprio direito de passagem [Klein 2006]. No entanto, na medida em que o fluxo torna-se maior, os motoristas se deparam com muitas decisões a fazer a fim de atribuir, com segurança, seu direito de passagem. Logo, o uso de regras e normas de tráfego pode ajudar os motoristas em uma série de decisões.

A forma mais simples de controle de tráfego é a utilização de um semáforo para sinalizar aos motoristas, em uma das vias, para que os mesmos conheçam a movimentação correta. Isto é mais adequado para vias com baixo volume de tráfego. Com aumento do tráfego, os motoristas devem ser parados para que tenham tempo para realizar complexas decisões sobre o direito de passagem. Uma intersecção de duas vias com sinal de parada na abordagem secundária serve a este próposito.

Eventualmente, o tráfego será construído a tal ponto que a atribuição do direito de passagem não é o único problema e a movimentação do tráfego eficiente torna-se uma preocupação primária também.

A partir das constatações anteriores, nota-se que o gerenciamento de ruas é uma importante atividade, uma vez que o uso de diversos elementos para controle, fiscalização e vigilância tendem a garantir maior fluidez para o tráfego e menores gargalos, filas e/ou congestionamentos que caracterizam grandes problemas do tráfego.

Os sistemas de controle de sinais de trânsito são sistemas de tempo-real que se enquadram na categoria de sistemas intensivos de software (*Software Intensive Systems*) e,

dada a sua complexidade, muitas são as dificuldades encontradas em seu desenvolvimento e manutenção. Segundo [Roess et al. 2004], os sistemas controladores de sinais de trânsito têm como características serem sistemas heterogêneos, legados, com incompatibilidade de hardware e software, de manutenção cara e com constantes alterações nas políticas de trânsito.

Podem ser citados como elementos que pertencem à infraestrutura de gerenciamento de ruas os pontos/*loops* de detecção (equipamentos eletrônicos instalados sob a estrada), veículos com indicadores (que podem, através de dispositivos previamente instalados, sinalizar condições de viagem), controladores de tráfego (que possuem a lógica necessária para gerir o fluxo de tráfego sob diversas condições), semáforos de tráfego, entre outros [Gden 1994].

A criação de modelos para representar as funcionalidades destes sistemas e também para compreender diferentes aspectos deste domínio é importante, uma vez que permitem diminuir a complexidade inerente aos mesmos, destaca características relevantes e fornece diferentes visões relacionadas aos requisitos do projeto para diversos *stakeholders*. Com isso, contribui para expressar confiavelmente os diversos requisitos dos controladores na especificação.

Este trabalho de pesquisa atuará, neste sentido, propondo modelos que reflitam adequadamente o dinamismo, criticidade, concorrência e outros critérios relacionados à especificação, classificação e documentação de requisitos de um Sistema de Controle de Tráfego Urbano (RTMS). O estudo de caso foi aplicado em um Sistema de Controle de Tráfego Urbano em que pretende-se contribuir para representação das diversas características relacionadas ao escopo e à especificação dos sistemas de controle de trânsito.

3.3 Políticas de Sinais de Trânsito

Os controladores de sinais de tráfego para intersecções podem ser agrupados em dois tipos de estratégias [Administration 1996]. As estratégias individuais para intersecções e estratégias para grupos de intersecções. As estratégias para intersecções individuais são classificadas em **Controladores de tempo-fixo**, **Controladores Atuados** e **Controladores Adaptativos**.

Os Controladores de tempo-fixo utilizam temporizadores simples com política de tempo fixo e seguem um ciclo fixo independente da demanda. O direito de passagem é atribuído com base em um período de tempo fixo e pré-determinado ou determinado a partir de dados históricos e empregados em todos os intervalos de exibição de sinal. Portanto, o controle de tempo fixo é geralmente ineficiente para controlar intersecções propícias a mudanças contínuas na demanda [Roess et al. 2004].

Controladores Atuados usam planos flexíveis, já pré-estabelecidos, em que o deslocamento e as durações de fase podem ser prontamente alterados em resposta a percepção

das alterações no tráfego. Os controladores de tráfego atuados variam seu tempo de verde, baseado na demanda da intersecção, conforme medido em detectores instalados na rua. Estes detectores variam em tecnologia, mas os mais comuns são os detectores de *loop* indutivo [Klein 2006].

Nos controladores atuados, cada fase pode ser servida ou ignorada. A decisão de pular para a próxima fase ocorre quando não há carro no detector ou quando a fase anterior torna-se amarela. Se a fase é servida, ela é servida por um período mínimo chamado de tempo mínimo de verde ou iniciação, sendo estendida, se necessário, até o tempo máximo de verde disponível para a fase. O tempo de extensão serve para estender o tempo de verde por um pequeno período (extensão da fase verde) para manter o sinal verde enquanto pelotões (*platoon*, em inglês) de carros estão presentes na intersecção. O tempo de verde termina se não ocorrer mais atuações ou quando o tempo máximo de verde é atingido (e assim a fase fica amarela).

Na Figura 3.1 são mostrados os três parâmetros de tempo para o modo de controle atuado: o tempo mínimo de fase verde, a extensão da fase verde e o tempo máximo da fase verde.

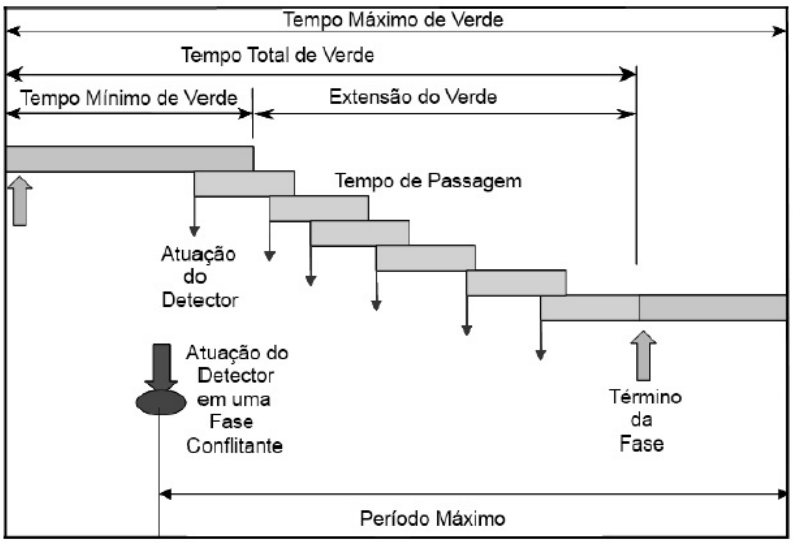

Figura 3.1: Operação em uma Fase de Modo Atuado. Adaptada de [Silvestre e Soares 2012a].

Um Controle Atuado pode ser de três tipos: **volume-densidade**, **atuado** ou **semi-atuado** [Roess et al. 2004]. No controle semi-atuado, a via principal funciona em modo não atuado de forma que a fase verde esteja sempre presente, a menos que uma atuação em uma via secundária seja recebida. Portanto, os sensores são necessários apenas para as vias secundárias sendo apropriados para locais onde existam distâncias maiores entre as intersecções e, principalmente, em situações em que o volume das vias secundárias é relativamente baixo. Já o controle totalmente atuado opera com detecção de tráfego em todas as ruas da intersecção e para todas as fases do sinal. É a estratégia de controle

amplamente aplicada para intersecções isoladas sendo útil quando o volume de tráfego é esporádico e variante, tornando necessário a modificação do plano semafórico. Finalmente, no tipo volume-densidade, as informações de tráfego são registradas e guardadas para serem usadas, posteriormente, para calcular/recalcular a duração do tempo mínimo da fase verde baseado na demanda atual.

Os Controladores Adaptativos são considerados sistemas distribuídos que executam em intersecções locais, usando controladores de tráfego avançados que recalculam predições e otimizações para o fluxo de tráfego em tempo-real [Gordon 2003]. Estes sistemas não têm ciclo, *splits* e deslocamentos no sentido clássico. São caracterizados por medir as condições de tráfego e desenvolver uma nova temporização para o sinal em tempo-real, semelhantes aos controladores de tempo atuado, mas eles podem alterar, por meio de um algoritmo adaptativo, mais parâmetros do que apenas a duração do tempo de verde. A otimização do controlador é baseada na alocação do tempo em tempo-real, com os objetivos de prover máximo desempenho ou mínimos atrasos e paradas de veículos.

Em relação às estratégias para intersecções em grupo tem-se que, no Controle sem Coordenação (Intersecção Isolada), o fluxo de tráfego é controlado sem considerar o funcionamento de sinais de trânsito adjacentes. Assim, cada controlador opera de forma descoordenada, isto é, independente do funcionamento dos controladores dos semáforos das outras intersecções.

O Controle Interligado ou em Rede caracteriza a extensão mais ampla dos controles, pois considera o desempenho e a configuração de um conjunto de intersecções semafóricas de uma determinada área. Os objetivos dos planos de temporização do sistema em rede são de proporcionar progressão em rede, de modo a transportar volumes de tráfego mais pesados, para maximizar a capacidade da rede e para minimizar o atraso. Duração e divisão do ciclo e os parâmetros do plano de compensação de tempo são variados para refletir as condições de tráfego atual. Este modo de operação é eficaz quando não existe sentido de tráfego dominante na rede. Qualquer um dos tipos de controladores (de tempo fixo, atuado ou adaptativo) pode ser utilizado para configuração de controles interligados.

3.4 Características Gerais do Estudo de Caso

O *Institute of Transportation Engineers* (ITE) composto por membros da indústria, de vários estados e cidades, da Administração de Rodovias Federais e do NEMA desenvolveu o conceito para o *Advanced Transportation Controller* (ATC) criando um padrão funcional para controladores de ruas e de rodovias. Um exemplo de controlador de tráfego pode ser visualizado na Figura 3.2.

O controlador é projetado para controlar ou processar dados de um ou mais dispositivos das rodovias. Ele funciona como um computador de propósito geral, com um sistema operacional de tempo-real. Com base nestas especificações, foram derivados um conjunto

Figura 3.2: Exemplo de Controlador para 170E [Administration 1996].

de requisitos que descrevem as capacidades de controle do sistema controlador de trânsito.

Segundo [Gordon 2003], os processos da engenharia de sistemas envolvidos na concepção de sistemas de controle de sinalização de tráfego começam com a identificação dos requisitos para atender às seguintes questões: especificação das funções do controlador, especificação e gerenciamento do projeto do sistema, definição das operações, estabelecimentos dos critérios de logística e avaliação. Assim, o foco deste trabalho é aplicar modelos para descrever as funções relacionadas ao sistema de controle de intersecções de tráfego que aborde parte das questões levantadas por [Gordon 2003], como funções e operações do controlador, critérios de logística da operação e controle da intersecção e especificação dos requisitos relacionados.

A partir de pesquisas em diferentes bibliografias, [Gden 1994], [Gordon 2003] [Laplante 2006], [Roess et al. 2004], [Sussman 2005], [Klein 2006], [Soares e Vrancken 2007] e [Silvestre e Soares 2012b], foi elaborado um documento que define diversos requisitos para o projeto de um Sistema de Gerenciamento de Tráfego com controle de intersecções de quatro vias.

Esta especificação é destinada para o uso de usuários finais bem como desenvolvedores de software. Os requisitos estão dispostos, na especificação, como descrito a seguir:

- Quatorze requisitos de propósito geral e rotulados com TM (de *Traffic Management*);
- Sessenta e um requisitos funcionais apresentados na Seção 3.5 e rotulados como TMFR (de *Functional Requirements of Traffic Management*);
- Sessenta e dois requisitos não-funcionais descritos na Seção 3.6 e rotulados como TMNFR (*Non Functional Requirements of Traffic Management*).

Os requisitos em alto nível de abstração para um Sistema de Controle de Tráfego são apresentados em seguida.

- TM1 - O sistema deve controlar o padrão de tráfego de veículos na intersecção;
- TM2 - O sistema deve permitir a sincronização de semáforos;
- TM3 - O sistema pode coletar todos os tipos de informações das vias com o objetivo de avaliar convenientemente estes dados;

- TM4 - O sistema deve permitir a gestão do histórico de tráfego;
- TM5 - O sistema deve permitir o controle adaptativo de horários da intersecção em resposta ao fluxo de tráfego;
- TM6 - O sistema deve permitir o controle atuado da intersecção em resposta ao fluxo de tráfego;
- TM7 - O sistema deve permitir o modo de preempção de emergência, ou seja, a movimentação preferencial de veículos de emergência;
- TM8 - O sistema deve permitir o controle da intersecção presencial em resposta aos comandos de acionamento manual;
- TM9 - O sistema deve permitir o controle de intersecção em resposta aos comandos de substituição remotos;
- TM10 - O sistema deve permitir a gestão de incidentes;
- TM11 - O sistema de controle da intersecção deve ser capaz de interagir com o painel de controle do software (Interface Externa);
- TM12 - O sistema deve permitir a operação automática dos semáforos;
- TM13 - O sistema utilizará interface TCP/IP SNMP para comunicação inter sistema;
- TM14 - O sistema pode gerar dados estatísticos para apoio a tomada de decisões.

3.5 Conjunto de Requisitos Funcionais para o Estudo de Caso

Nesta seção os requisitos que descrevem as propriedades funcionais para o Sistema de Controle de Tráfego são apresentados.

ID	Nome Requisito
TMFR1.1	O sistema deve exibir a indicação comandada para uma via.
TMFR1.2	O sistema deve detectar a presença de veículos.
TMFR1.3	O sistema deve avaliar as requisições de serviços de veículos.
TMFR1.4	O sistema deve gerar eventos de solicitação de serviços do veículo (modo atuado).
TMFR1.5	O sistema deve manter as estatísticas de contagem de veículo (modo fixo, atuado e adaptativo).
TMFR1.6	O sistema deve indicar os status do loop de detecção.
TMFR1.7	O sistema deve determinar a indicação atualmente exibida.

Tabela 3.1: Requisitos Funcionais para TM1

ID	Nome Requisito
TMFR2.1	O sistema deve receber informações de fluxos das vias adjacentes.
TMFR2.2	O sistema deve processar informações recebidas dos controladores de cada uma das vias.
TMFR2.3	O sistema deve enviar mensagens de controle de sincronização para outros controladores.

Tabela 3.2: Requisitos Funcionais para TM2

ID	Nome Requisito
TMFR3.1	O sistema deve perceber a entrada de veículos na abordagem.
TMFR3.2	O sistema deve verificar a presença de veículos.
TMFR3.3	O sistema deve controlar os padrões de tráfego associado a cada via.
TMFR3.4	O sistema deve manipular o loop de detecção de Entrada e Saída de veículos.
TMFR3.5	O sistema deve rastrear a contagem de veículos já capturada nas vias.
TMFR 3.6	O sistema deve prover interface de comunicação com sensores e atuadores.
TMFR 3.7	O sistema deve captar a indicação atual de trânsito baseado em sinais de sensores.
TMFR 3.8	O sistema deve armazenar informações das vias.
TMFR3.9	O sistema deve definir a contagem de veículos passantes através das vias.
TMFR3.10	O sistema deve reiniciar a contagem de veículos.

Tabela 3.3: Requisitos Funcionais para TM3

ID	Nome Requisito
TMFR4.1	O sistema deve armazenar o histórico de registros de tráfego.
TMFR4.2	O sistema deve recuperar o histórico registros de tráfego.
TMFR4.3	O sistema deve limpar o histórico de tráfego em resposta a um comando remoto.
TMFR4.4	O sistema deve limpar o histórico de tráfego em resposta a um comando local.

Tabela 3.4: Requisitos Funcionais para TM4

ID	Nome Requisito
TMFR5.1	O sistema deve captar informações das vias (detecção de volume).
TMFR5.2	O sistema deve armazenar as informações de tráfego enviadas pelo loop de detecção de veículos.
TMFR5.3	O sistema deve processar informações de tráfego.
TMFR5.4	O sistema deve disparar novo estado em tempo hábil para reprogramação dos controladores da intersecção.
TMFR5.5	O sistema deve manter as estatísticas de contagem de veículos.

Tabela 3.5: Requisitos Funcionais para TM5

ID	Nome Requisito
TMFR6.1	O sistema deve receber requisições de passagem de veículos.
TMFR6.2	O sistema deve processar requisição de veículos.
TMFR6.3	O sistema deve enviar sinal de "ok" ou "not ok" para prosseguir veículos solicitantes.
TMFR6.4	O sistema deve armazenar informações temporais de tráfego.
TMFR6.5	O sistema deve estabelecer cenários em horários pré-estabelecidos.
TMFR6.6	O sistema deve controlar tempo de verde.
TMFR6.7	O sistema deve manter as estatísticas de contagem de veículos.
TMFR6.8	O sistema deve indicar que o loop de detecção de veículos está ocupado.
TMFR6.9	O sistema deve indicar que o loop de detecção de veículos não está ocupado.
TMFR6.10	O sistema deve limpar requisições já realizadas por veículos.

Tabela 3.6: Requisitos Funcionais para TM6

ID	Nome Requisito
TMFR7.1	O sistema deve receber requisições (preempção) de veículos de emergência.
TMFR7.2	O sistema deve decodificar requisições de preempção de veículos de emergência.
TMFR7.3	O sistema deve provocar mudança de modo apropriado.
TMFR7.4	O sistema deve validar requisições de preempção de veículos de emergência.
TMFR7.5	O sistema deve gerar eventos necessários para controlar a fase da intersecção.
TMFR7.6	O sistema deve manipular eventos para controlar a fase da intersecção (gerar solicitação de prioridade para o software).

Tabela 3.7: Requisitos Funcionais para TM7

ID	Nome Requisito
TMFR8.1	O sistema deve gerar eventos necessários para controlar de forma presencial a fase da intersecção.
TMFR8.2	O sistema deve manipular eventos para controlar a intersecção de forma presencial.

Tabela 3.8: Requisitos Funcionais para TM8

ID	Nome Requisito
TMFR9.1	O sistema deve atribuir parâmetros de tempo de ciclo calculados pelo *host* remoto.
TMFR9.2	O sistema deve recuperar parâmetros/dados de *status* (usados como entrada para o algoritmo calculador de ciclo do controle adaptativo).
TMFR9.3	O sistema deve provocar a mudança de modo apropriado.
TMFR9.4	O sistema deve gerar eventos requeridos para o controle da intersecção.
TMFR9.5	O sistema deve manipular eventos requeridos para a fase de controle da intersecção.

Tabela 3.9: Requisitos Funcionais para TM9

ID	Nome Requisito
TMFR 10.1	O sistema deve armazenar os logs de incidentes.
TMFR 10.2	O sistema deve recuperar os logs de incidentes.
TMFR10.3	O sistema deve limpar o log de um incidente em resposta a um comando de um *host* remoto.
TMFR 10.4	O sistema deve registrar condições/eventos geradores de erro.

Tabela 3.10: Requisitos Funcionais para TM10

ID	Nome Requisito
TMFR 11.1	O sistema deve receber mensagens pela rede.
TMFR 11.2	O sistema deve enviar mensagens pela rede.
TMFR11.3	O sistema deve transferir história de tráfego.
TMFR11.4	O sistema deve transferir dados de logs de incidente.

Tabela 3.11: Requisitos Funcionais para TM11

ID	Nome Requisito
TMFR12.1	O sistema deve alterar configurações de tempo dos controladores.
TMFR12.2	O sistema deve permitir alterar configurações de tempo dos controladores.

Tabela 3.12: Requisitos Funcionais para TM12

ID	Nome Requisito
TMFR13.1	O sistema deve receber eventos gerados pelos objetos do sistema.
TMFR13.2	O sistema deve disparar eventos para objetos apropriados.

Tabela 3.13: Requisitos Funcionais para TM13

ID	Nome Requisito
TMFR14.1	O sistema pode realizar mineração de dados armazenados.
TMFR14.2	O sistema pode gerar planilhas com informações pré determinadas.

Tabela 3.14: Requisitos Funcionais para TM14

3.6 Conjunto de Requisitos Não-Funcionais para o Estudo de Caso

Nesta seção os requisitos não-funcionais que descrevem diferentes características relacionadas aos Sistemas de Controle de Tráfego são detalhados.

ID	Nome Requisito
TMNFR1.1	O sistema deve controlar o padrão de tráfego na intersecção com máximo rendimento.
TMNFR1.2	O sistema deve controlar o padrão de tráfego de veículos na intersecção com máximo desempenho.
TMNFR1.3	O sistema deve realizar a avaliação das requisições de serviços de veículos em no máximo 100ms.
TMNFR1.6	O sistema deve indicar o *status* do loop de detecção em no mínimo 100ms e no máximo 150ms.

Tabela 3.15: Requisitos Não-Funcionais para TM1

ID	Nome Requisito
TMNFR2.1	As informações de fluxos das vias adjacentes devem ser recebidas em no máximo 100ms.
TMNFR2.2	O sistema deve processar as informações recebidas dos controladores de cada uma das vias em no mínimo 150ms e no máximo 200ms.
TMNFR2.3	O sistema deve enviar as mensagens de controle de sincronização para outros controladores em no máximo 100ms.
TMNFR2.4	O sistema deve realizar a sincronização dos semáforos de forma paralela com as demais operações na intersecção.
TMNFR2.5	O sistema deve aumentar a capacidade da via por meio de estrátegias de sincronização dos semáforos.

Tabela 3.16: Requisitos Não-Funcionais para TM2

ID	Nome Requisito
TMNFR3.1	O sistema deve captar a entrada de veículos em no máximo 10ms.
TMNFR3.2	O sistema deve realizar a verificação de presença de veículoem no máximo 10ms.
TMNFR3.5	O sistema deve rastrear a contagem de veículos já capturada nas vias em no máximo 100ms.
TMNFR3.8	O sistema deve armazenar as informações das vias em no máximo 200ms.
TMNFR3.9	O sistema deve definir a contagem de veículos passantes através das vias em no máximo 50ms.
TMNFR3.10	O sistema deve reiniciar a contagem de veículos em no máximo 100ms.
TMNFR3.11	As funções de comunicação entre os componentes do sistema devem ser apresentadas em uma interface de fácil usabilidade.
TMNFR3.12	O sistema deve armazenar a velocidade média dos carros passantes pelas vias que cortam a intersecção.
TMNFR3.13	O sistema deve prover segurança a pedestres e condutores que acessam a intersecção.
TMNFR3.14	O sistema deve realizar a avaliação dos dados e das intersecções com máximo desempenho.

Tabela 3.17: Requisitos Não-Funcionais para TM3

ID	Nome Requisito
TMNFR4.1	O sistema deve armazenar os registros de histórico em no máximo 1000ms.
TMNFR4.2	O sistema deve recuperar os registros de histórico em no máximo 200ms.
TMNFR4.3	O sistema deve limpar o histórico de tráfego em resposta a um comando remoto em no mínimo 200ms e no máximo 250ms.
TMNFR4.4	O sistema deve limpar o histórico de tráfego em resposta a um comando local em no máximo 200ms.
TMNFR4.5	O sistema deve armazenar as informações no histórico de registros de forma íntegra.

Tabela 3.18: Requisitos Não-Funcionais para TM4

ID	Nome Requisito
TMNFR5.1	O sistema deve captar as informações de detecção de volume das vias (detecção de volume) em no máximo 100ms.
TMNFR5.2	O armazenamento de informações de tráfego, enviadas pelo loop de detecção de veículo, deverá ser efetuado em no máximo 1000ms.
TMNFR5.3	O sistema deve processar as informações de tráfego em no mínimo 150ms e no máximo 200ms.
TMNFR5.4	O sistema deve disparar novo estado em tempo hábil para reprogramação dos controladores da intersecção em no máximo 100ms.
TMNFR5.5	O sistema deve alterar o novo estado dos atuadores em sincronia com os demais controladores da rede.
TMNFR5.6	Os controladores deverão receber seu estado de forma segura.
TMNFR 5.7	A detecção de volume deve ser realizada com máximo desempenho e confiabilidade.

Tabela 3.19: Requisitos Não-Funcionais para TM5

ID	Nome Requisito
TMNFR6.1	O sistema deve receber as requisições de passagem de veículos em no máximo 100ms.
TMNFR6.2	O sistema deve realizar o processamento das requisições de veículos em no máximo 100ms.
TMNFR6.3	O sistema deve enviar sinal de “ok” ou “not ok” para prosseguir para veículos solicitantes em no máximo 100ms.
TMNFR6.4	O sistema deve indicar que o loop de detecção de veículos está ocupado com em máximo 150ms.
TMNFR6.5	O sistema deve indicar que o loop de detecção de veículos não está ocupado em no máximo 150ms.
TMNFR6.6	O sistema deve checar as condições do loop de detecção com máximo desempenho.
TMNFR6.7	O sistema deve controlar o tempo de verde com alto desempenho de modo a garantir a fluidez do tráfego.

Tabela 3.20: Requisitos Não-Funcionais para TM6

ID	Nome Requisito
TMNFR7.1	O sistema deve receber as requisições (preempções) de veículos de emergência em no máximo 200ms.
TMNFR7.2	O sistema deve decodificar as requisições de preempção de veículos de emergência em no máximo 200ms.
TMNFR 7.3	O sistema deve provocar mudança de modo apropriado em no máximo 100ms.
TMNFR7.4	O sistema deve provocar mudança na prioridade de acesso à via de forma segura.

Tabela 3.21: Requisitos Não-Funcionais para TM7

ID	Nome Requisito
TMNFR8.1	O sistema deve gerar eventos necessários para controlar a fase de intersecção em no máximo 100ms.
TMNFR8.2	O controle manual do controlador não deve comprometer o desempenho operacional do mesmo.
TMNFR8.3	O sistema deve provocar mudança do modo de acesso à intersecção de forma segura.
TMNFR8.4	O sistema deve provocar mudança de modo de acesso com bom desempenho.

Tabela 3.22: Requisitos Não-Funcionais para TM8

ID	Nome Requisito
TMNFR9.1	O sistema deve atribuir parâmetros de tempo de ciclo calculados pelo host remoto em no máximo 100ms.
TMNFR 9.3	O sistema deve provocar mudança de modo apropriado em no máximo 100ms.
TMNFR9.4	O sistema deve provocar mudança de modo de acesso a intersecção de forma segura.
TMNFR9.5	O sistema deve provocar mudança de modo de acesso com bom desempenho.

Tabela 3.23: Requisitos Não-Funcionais para TM9

ID	Nome Requisito
TMNFR10.1	O sistema deve realizar o armazenamento dos logs de incidentes em no máximo 200ms.
TMNFR10.2	O sistema deve recuperar os logs de incidentes em no máximo 1000ms.
TMNFR10.3	O sistema deve limpar o log de um incidente em resposta a um comando de um host remoto em no máximo (200ms).
TMNFR10.4	O sistema deve registrar condições/eventos geradores de erros em no máximo 500ms.
TMNFR10.5	O sistema deve mostrar informações quantitativas em relação aos incidentes.

Tabela 3.24: Requisitos Não-Funcionais para TM10

ID	Nome Requisito
TMNFR11.1	O sistema deve receber mensagens da rede em no máximo 100ms.
TMNFR11.2	O sistema deve enviar mensagens pela rede em no máximo 100ms.

Tabela 3.25: Requisitos Não-Funcionais para TM11

ID	Nome Requisito
TMNFR12.1	O sistema deve alterar configurações de tempo dos controladores em no mínimo 20ms e no máximo 30ms.

Tabela 3.26: Requisitos Não-Funcionais para TM12

Capítulo 4

Modelagem e Rastreabilidade de Requisitos usando o Diagrama de Requisitos da SysML Estendido

Este capítulo baseia-se nos artigos "*A Metamodel for Tracing Requirements of Real-Time Systems*" [Ribeiro e Soares 2013c] e "*Application of an Extended SysML Requirements Diagram to Model Real-Time Control Systems*" [Ribeiro e Soares 2013a] e tem por objetivo estender o metamodelo do diagrama de requisitos da SysML, por meio da adição de novas propriedades e relacionamentos que permitam representar com completude as características dos requisitos de sistemas de tempo-real, documentar requisitos em diversos níveis de abstração e seus relacionamentos e, ainda, explicitar, já nas fases iniciais de especificação, a modelagem e a representação de questões de rastreabilidade entre requisitos. O metamodelo para representar requisitos em múltiplos níveis de abstração e classificação e para descrever o rastreamento entre requisitos é apresentado na Seção 4.1. Na Seção 4.2, os novos relacionamentos estendidos da SysML para fornecer a comunicação e a classificação de requisitos são propostos. A extensão proposta às tabelas da SysML são apresentadas na Seção 4.3. Finalmente, na Seção 4.4, é proposto o estudo de caso mostrando a aplicação da Tabela SysML estendida e do metamodelo criado em um conjunto de requisitos para um sistema de controle de tráfego.

4.1 Metamodelo SysML

Na Figura 4.1, é apresentado o metamodelo proposto neste trabalho para representar requisitos de STR e seus relacionamentos.

O diagrama de requisitos da SysML, descrito no Capítulo 2, é estendido com novas categorias de requisitos. Na Figura 4.1 são mostrados os novos requisitos estendidos sendo nomeados, respectivamente, como $<< ExtRequirement >>$, $<< RequirementFunc >>$,

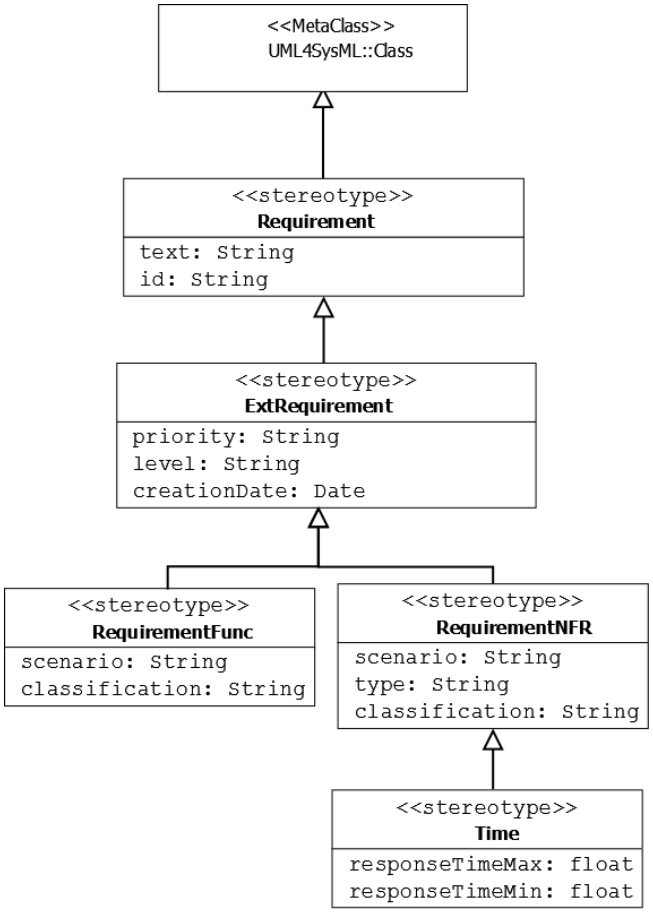

Figura 4.1: Metamodelo SysML Estendido.

$<< RequirementNFR >>$ e $<< Time >>$. O requisito estendido $<< ExtRequirement >>$ possibilita documentar requisitos gerais (mais abstratos) e funcionais. O requisito estereotipado como $<< RequirementFunc >>$ possibilita a modelagem e classificação de requisitos funcionais. O requisito estereotipado como $<< RequirementNFR >>$ possibilita a modelagem e classificação de requisitos não-funcionais. Finalmente, o requisito estendido representado por $<< Time >>$, que também descreve um requisito não-funcional, mas com atributos adicionais e específicos para modelar propriedades temporais de um requisito é apresentado.

O modelo estendido inclui alguns atributos adicionais ao modelo básico do diagrama de requisitos da SysML. O estereótipo $<< ExtRequirement >>$ tem como atributos adicionais *priority*, *level* e *creationDate*. O atributo *priority* objetiva definir a prioridade para um requisito em relação aos demais. É um atributo do tipo string que pode ter os seguintes valores atribuídos: *must* (deve), *should* (pode), *could* (poderia) e *won't* (não necessário). O atributo *level* representa, para fins de classificação e rastreamento, qual é o nível de abstração de um requisito. O atributo *level*, do tipo inteiro, pode assumir o valor 1 para referenciar requisitos de alto-nível, o valor 2 para referenciar requisitos funcionais, o valor 3 para requisitos não-funcionais e assim sucessivamente. O atributo *creationDate*, do tipo string, define a data de criação do requisito, sendo importante para definir uma

extensão de controle de um requisito conforme suas alterações ao longo da especificação.

O estereótipo << *RequirementFunc* >> possui dois novos atributos que são *scenario* e *classification*. O atributo *scenario*, do tipo string, apresenta o cenário ao qual o requisito em questão está interligado. O atributo *classification* classifica um requisito em funcional, não-funcional ou de domínio.

O estereótipo << *RequirementNFR* >> contém três atributos: *scenario*, *classification* e *type*. O atributo *type*, do tipo string, para referir já na modelagem o tipo relacionado ao requisito não-funcional como, por exemplo, segurança, desempenho, confiabilidade, entre outros. O estereótipo << *Time* >> tem como atributos *responsetTimeMin* e *responseTimeMax*, do tipo float, que representam restrições de tempo de um requisito que são descritas dentro do intervalo pré-definido pelos atributos.

O principal objetivo retratado, através da extensão proposta do diagrama de requisitos da SysML, é permitir a representação de requisitos de software em diferentes níveis de abstração e detalhe, a classificação e documentação destes requisitos e a representação de diversos atributos característicos aos STR.

4.2 Novos Relacionamentos entre Requisitos

A SysML permite, por meio de seus relacionamentos, decompor requisitos complexos em requisitos mais simples, formando uma hierarquia de requisitos relacionados entre si. Com estes relacionamentos, a complexidade de um sistema é tratada desde as fases iniciais do desenvolvimento através da decomposição de requisitos complexos.

O relacionamento *trace* da SysML é geral e sua semântica é ineficaz para representar a rastreabilidade na modelagem de requisitos. Nos relacionamentos já definidos pela SysML não são representadas relações de composição entre requisitos. Assim, foram propostos neste trabalho novos relacionamentos que estendem os relacionamentos básicos da SysML e, ainda, o uso das Tabelas da SysML (Seção 4.4.2) para representar interligações entre requisitos, em um mesmo nível de abstração e entre dois ou mais níveis, e a rastreabilidade de tais requisitos.

O modelo estendido é capaz de representar requisitos em diferentes níveis de abstração, seus relacionamentos e sincronismo entre requisitos. Os novos relacionamentos são apresentados na Figura 4.2.

O estereótipo << *aggregate* >>, representado na Figura 4.3, foi proposto com intuito de mostrar os requisitos explicitamente correlacionados (em um mesmo nível de classificação/hierarquia) com um requisito/função em um nível superior (isto é, quais requisitos compõem um requisito mais geral). Portanto, os requisitos que são relacionados por meio do relacionamento << *aggregate* >> estão fortemente relacionados e fornecem, conjuntamente, uma funcionalidade proposta/fornecida de requisito de mais alto nível. Este relacionamento permite representar a decomposição de requisitos e o seu agrupamento para

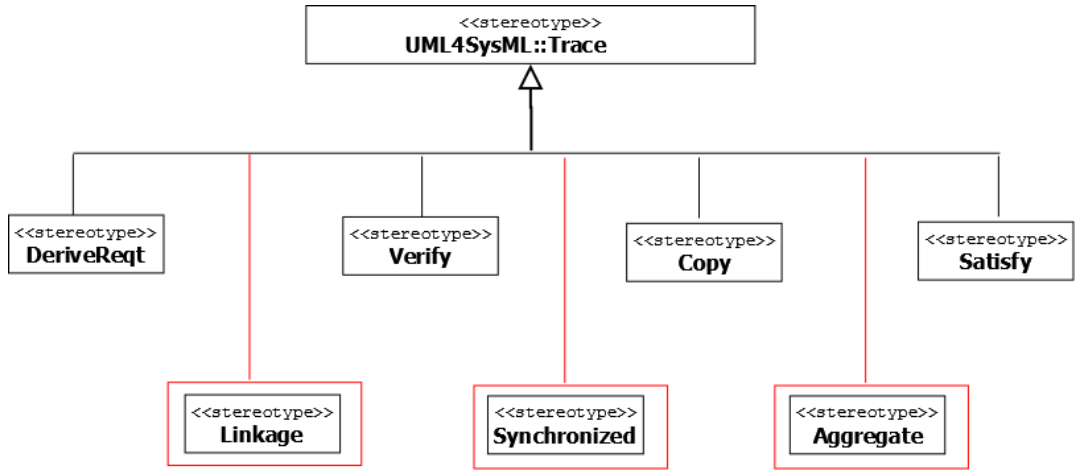

Figura 4.2: O Metamodelo de Relacionamento Estendido.

compor o subsistema representado. Sendo assim, o relacionamento << *aggregate* >> permite, para fins de análise e estudo, a modelagem das funcionalidades de um sistema de forma agrupada.

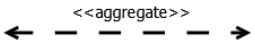

Figura 4.3: Relacionamento << *aggregate* >>.

O estereótipo << *linkage* >> representado por uma linha reta e um círculo com uma cruz interna em ambas as extremidades é proposto para melhorar a atividade de rastreamento de requisitos para os modelos de projeto. Este relacionamento mostra explicitamente as interligações de um requisito em qualquer nível da hierarquia. Assim, é possível, por exemplo, determinar a rastreabilidade de um requisito em alto nível para com seus requisitos funcionais e/ou não-funcionais. Sua representação gráfica é representada na Figura 4.4.

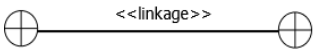

Figura 4.4: Relacionamento << *linkage* >>.

O estereótipo << *synchronized* >> (Figura 4.5) foi criado para ser utilizado por requisitos não-funcionais que podem descrever, além de restrições de desempenho, outros requisitos que devem ser processados/representados concorrentemente.

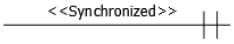

Figura 4.5: Relacionamento << *synchronized* >>.

4.3 Tabelas da SysML

A Tabela 4.1 representa uma extensão à tabela de requisitos da SysML nomeada como Tabela de Relacionamentos entre Requisitos da SysML (SRRT). Foram adicionados na

tabela os campos **ID**, **RequisitoNome**, **RelacionaCom**, **RelacionaComo** e **Tipo**.

O campo **ID** especifica o identificador do requisito em questão, o campo **RequisitoNome** descreve o nome do requisito, **RelacionaCom** especifica, caso haja, quais requisitos se relacionam com o requisito descrito, e como os requisitos são relacionados é descrito em **RelacionaComo** e, por fim, o tipo do requisito também pode ser descrito no campo **Tipo**.

ID	RequisitoNome	RelacionaCom	RelacionaComo	Tipo
-	-	-	-	-

Tabela 4.1: Tabela de Rastreabilidade de Requisitos da SysML

As Tabelas da SysML representam explicitamente as interligações entre os requisitos em cada nível de representação e entre os diferentes níveis na hierarquia. Além de melhorar a rastreabilidade dos requisitos, as Tabelas da SysML fornecem uma forma ágil para identificar e priorizar requisitos e, ainda, complementam a documentação dos requisitos do sistema ao fornecer a visão textual formatada (tabular) dos mesmos, o que possibilita a diminuição de ambiguidades e inconsistências da especificação.

4.4 Estudo de Caso Proposto

Nesta seção o metamodelo proposto, na Seção 4.1, será aplicado em um estudo de caso para um Sistema de Controle de Tráfego Urbano. O diagrama de requisitos da SysML estendido será utilizado para documentar e modelar um subconjunto de requisitos (descrito na Seção 4.4.1) de um sistema de controle de intersecções de tráfego.

4.4.1 Conjunto de Requisitos Considerados

Nesta seção objetiva um subconjunto de requisitos para um Sistema de Controle de Tráfego Urbano (RTMS) são descritos. Esses requisitos são derivados dos 137 requisitos propostos no documento de requisitos apresentados no Capítulo 3.

O subconjunto de requisitos de propósito geral para o RTMS é apresentado na Tabela 4.2, descrevendo, em linhas gerais, as características de um sistema de controle de intersecções de tráfego. A especificação proposta objetiva detalhar os seguintes requisitos:

- Controle de Tempo da Intersecção;
- Controle de Fluxo de Veículos;
- Configuração e Controle dos Controladores em Modo Atuado;
- Recebimento, Armazenamento e Processamento de Dados das Abordagens;
- Controle dos Planos Semafóricos;

ID	Requirement Name
TM1	O sistema deve controlar o padrão de tráfego de veículos na intersecção.
TM2	O sistema deve permitir a sincronização de semáforos.
TM3	O sistema pode coletar todos os tipos de informações das vias com o objetivo de avaliar convenientemente estes dados.
TM4	O sistema deve permitir a gestão do histórico de tráfego.
TM5	O sistema deve permitir o controle adaptativo de horários da intersecção em resposta ao fluxo de tráfego.
TM6	O sistema deve permitir o controle atuado da intersecção em resposta ao fluxo de tráfego.
TM7	O sistema deve permitir o modo de preempção de emergência, ou seja, permitir a movimentação preferencial de veículos de emergência.
TM8	O sistema deve permitir o controle da intersecção presencial em resposta aos comandos de acionamento manual.
TM9	O sistema deve permitir o controle da intersecção em resposta a comandos de substituição remotos.
TM10	O sistema deve permitir a gestão de incidentes.
TM11	O sistema de controle da intersecção deve ser capaz de interagir com o painel de controle do software.
TM12	O sistema deve permitir a operação automática dos semáforos.
TM13	O sistema utilizará interface TCP/IP SNMP para comunicação inter sistema.
TM14	O sistema pode gerar dados estatísticos para fornecer apoio a tomada de decisões.

Tabela 4.2: Requisitos em Alto Nível para um Sistema de Controle de Tráfego

- Controle do Tempo de Verde.

Os requisitos funcionais modelados neste estudo de caso são representados em linguagem natural na Tabela 5.2. A Tabela 5.3 descreve o subconjunto de requisitos não-funcionais utilizados no estudo de caso.

ID	Nome Requisito
TMFR6.1	O sistema deve receber requisições de passagem de veículos.
TMFR6.2	O sistema deve processar requisições de veículos.
TMFR6.3	O sistema deve enviar sinal de “ok” ou “not ok” para prosseguir veículos solicitantes.
TMFR6.4	O sistema deve armazenar informações temporais de tráfego.
TMFR6.5	O sistema deve estabelecer cenários em horários pré-estabelecidos.
TMFR6.6	O sistema deve controlar tempo de verde.
TMFR6.7	O sistema deve manter as estatísticas de contagem de veículos.
TMFR6.8	O sistema deve indicar que o loop de detecção de veículos está ocupado.
TMFR6.9	O sistema deve indicar que o loop de detecção de veículos não está ocupado.
TMFR6.10	O sistema deve limpar requisições já realizadas por veículos.

Tabela 4.3: Requisitos Funcionais para um Sistema de Controle de Tráfego

Os requisitos, em cada um dos diferentes níveis, foram apresentados em linguagem

ID	Nome Requisito
TMNFR6.1	O sistema deve receber as requisições de passagem de veículos em no máximo 100ms.
TMNFR6.2	O sistema deve realizar o processamento da requisição de veículos em no máximo 100ms.
TMNFR6.3	O sistema deve enviar sinal de "ok" ou "not ok" para prosseguir para veículos solicitantes em no máximo 100ms.
TMNFR6.4	O sistema deve indicar que o loop de detecção de veículos está ocupado em no máximo 150ms.
TMNFR6.5	O sistema deve indicar que o loop de detecção de veículos não está ocupado em no máximo 150ms.
TMNFR6.6	O sistema deve checar as condições do loop de detecção com máximo desempenho
TMNFR6.7	O sistema deve controlar o tempo de verde com alto desempenho de modo a garantir a fluidez do tráfego

Tabela 4.4: Requisitos Não-Funcionais para um Sistema de Controle de Tráfego

natural. No entanto, o foco deste trabalho é a utilização de modelos gráficos para representação, classificação e rastreamento dos requisitos do RTMS. A aplicação do metamodelo SysML para modelagem de requisitos é realizada e detalhada na Seção 4.5.

4.4.2 Tabela SysML Proposta

A Tabela 4.5 e a Tabela 4.6 representam os requisitos propostos para o estudo de caso por meio da SRRT, estendida na Seção 4.3. Como pode ser observado, os requisitos são dispostos na SRRT e estão identificados por seu ID ou nome e seus relacionamentos são explicitados. Os relacionamentos entre requisitos são listados a partir da aplicação do metamodelo no conjunto completo de requisitos modelado na Figura 4.8.

ID	RequisitoNome	RelacionaCom	RelacionaComo	Tipo
TM1	ControlarPadrão	TM5, TM6	hierarchy,aggregate,derive	Geral
TM5	ControleAdaptativo	nome	hierarchy,aggregate,derive	Geral
TM6	ControleAtuado	TMFR6.1 até TMFR6.10	linkage	Geral
TMFR6.1	receberReq	TMFR6.2 até TMFR6.10	aggregate	Funcional
TMFR6.1	receberReq	TMNFR6.1, TM6	linkage	Funcional
TMFR6.2	processarReq	TMFR6.1, TMFR6.3 até TMFR6.10	aggregate	Funcional
TMFR6.2	processarReq	TMNFR6.2, TM6	linkage	Funcional
TMFR6.3	enviarSinal	TMFR6.1, TMFR6.2, TMFR 6.3 até TMFR6.10	aggregate	Funcional
TMFR6.3	enviarSinal	TMNFR6.3, TMNFR6.7, TM6	linkage	Funcional
TMFR6.4	armazenarInfo	TMFR6.1 até TMFR6.3, TMFR6.5 até TMFR6.10	aggregate	Funcional
TMFR6.4	armazenarInfo	TM6	linkage	Funcional
TMFR6.5	estabelecerCen	TMFR6.1 até TMFR6.4, TM6 até TMFR6.10	aggregate	Funcional
TMFR6.5	estabelecerCen	TM6	linkage	Funcional
TMFR6.6	tempoVerde	TMFR6.1 até TMFR6.5, TM7 até TMFR6.10	aggregate	Funcional
TMFR6.6	tempoVerde	TMNFR6.6, TM6	linkage	Funcional
TMFR6.7	estatísticasCont	TMFR6.1 até TMFR6.6, TM8 até TMFR6.10	aggregate	Funcional
TMFR6.7	estatisticaCont	TM6	linkage	Funcional
TMFR6.8	loopOcupado	TMFR6.1 até TMFR6.7, TM9, TMFR6.10	aggregate	Funcional
TMFR6.8	loopOcupado	TMNFR6.4, TM6	linkage	Funcional
TMFR6.9	loopNoOcupado	TMFR6.1 até TMFR6.8, TMFR6.10	aggregate	Funcional
TMFR6.9	loopNoOcupado	TMNFR6.5, TM6	linkage	Funcional
TMFR6.10	limparReq	TMFR6.1 até TMFR6.9	aggregate	Funcional
TMFR6.10	limparReq	TM6	linkage	Funcional

Tabela 4.5: Representação de Rastreabilidade por meio do uso da SRRT

ID	RequisitoNome	RelacionaCom	RelacionaComo	Tipo
TMNFR6.1	reqPassagem	TMNFR6.2 até TMNFR6.7	aggregate	NFP
TMNFR6.1	reqPassagem	TMFR6.1	linkage	NFP
TMNFR6.2	procReqVeiculo	TMNFR6.1, TMNFR6.3 até TMNFR6.7	aggregate	NFP
TMNFR6.2	procReqVeiculo	TMFR6.2	linkage	NFP
TMNFR6.3	enviarSinalVei	TMNFR6.1, TMNFR6.2, TMNFR6.4 até TMNFR6.7	aggregate	NFP
TMNFR6.3	enviarSinalVei	TMFR6.3	linkage	NFP
TMNFR6.4	indicarOcup	TMNFR6.1 até TMNFR6.3, TMNFR6.5 até TMNFR6.7	aggregate	NFP
TMNFR6.4	indicarOcup	TMFR6.8	linkage	NFP
TMNFR6.5	indicarNoOcup	TMNFR6.1 até TMNFR6.4, TMNFR6.6,TMNFR6.7	aggregate	NFP
TMNFR6.5	indicarNoOcup	TMFR6.9	linkage	NFP
TMNFR6.6	checarLoop	TMNFR6.1 até TMNFR6.5, TMNFR6.7	aggregate	NFP
TMNFR6.6	checarLoop	TMFR6.6	linkage	NFP
TMNFR6.6	checarLoop	TMFR6.7	synchronized	NFP
TMNFR6.7	performanceTempoV	TMNFR6.1 até TMNFR6.6	aggregate	NFP
TMNFR6.7	performanceTempoV	TMFR6.6, TMFR6.3	linkage	NFP

Tabela 4.6: Representação de Rastreabilidade por meio do uso da SRRT

A representação proposta é util por representar de forma relativamente simples todos os requisitos de um sistema e como um requisito afeta outro requisito.

4.5 Diagrama de Requisitos da SysML Estendido para Classsificação, Rastreabilidade e Documentação

Como pode ser observado na Figura 4.6, os requisitos $TM5$, $TM6$, $TM8$, $TM9$ e $TM12$ são relacionados com o requisito $TM1$ por meio do relacionamento *derive*, sendo estereotipado com << *deriveReqt* >> e, também, pelo relacionamento *hierarchical*. O emprego do relacionamento *derive* justifica-se devido ao fato de que os requisitos mencionados anteriormente são derivados a partir do requisito que controla o padrão de tráfego de veículos na intersecção (neste caso $TM1$).

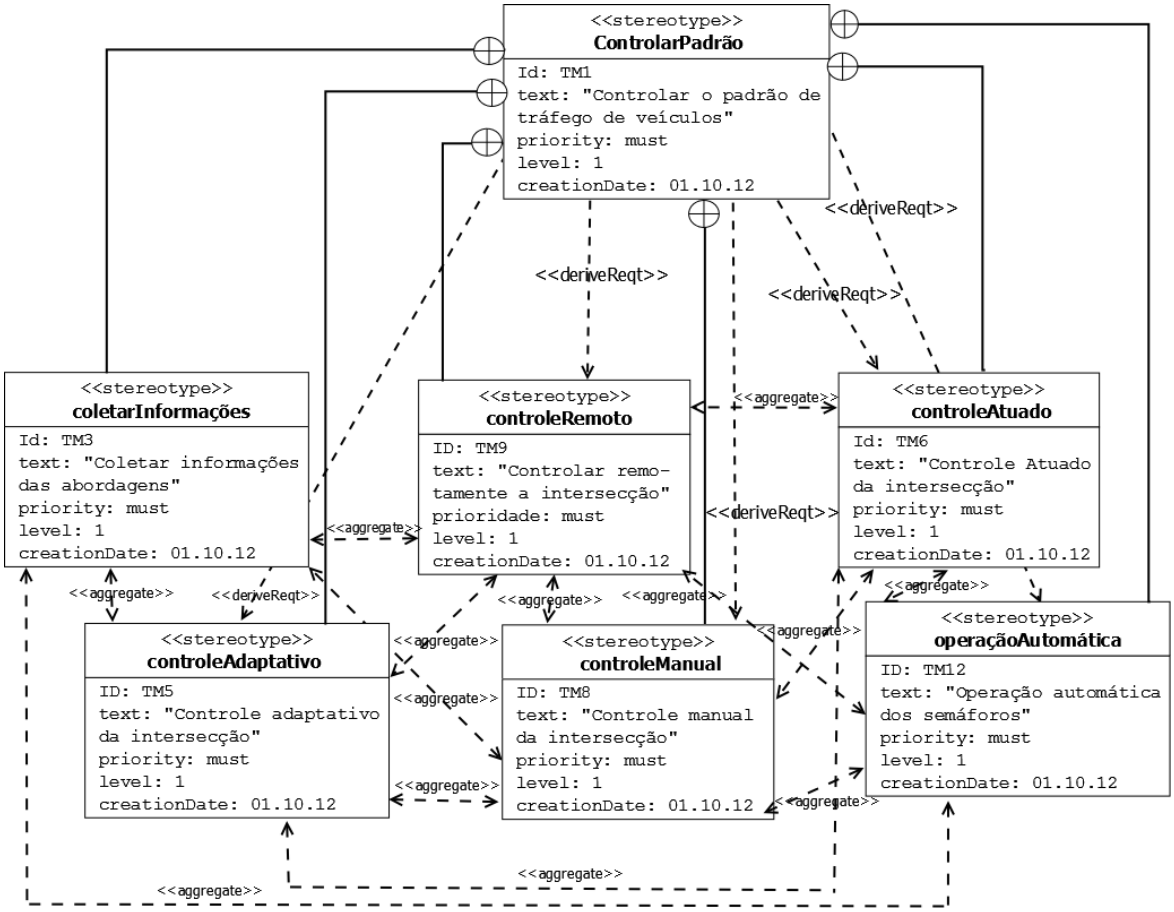

Figura 4.6: Requisitos em Alto Nível e seus Relacionamentos.

O relacionamento *hierarchical*, representado por um círculo com uma cruz interna, também mostrado na Figura 4.6, representa um forte relacionamento entre um requisito mais geral e requisitos mais específicos.

A aplicação do novo relacionamento criado juntamente com seu estereótipo << *aggregate* >> é mostrada na Figura 4.6. O relacionamento << *aggregate* >> expressa se dois

ou mais requisitos em um mesmo nível de abstração estão interligados entre si possivelmente para compor um módulo do sistema modelado (explicitado por um ou mais requisito (s) de mais alto nível).

Na Figura 4.7, os relacionamentos << *aggregate* >> entre requisitos em mesmo nível (no mesmo ramo da árvore de modelagem) foram suprimidos devido a questões de legibilidade. Assim, o pacote com o estereótipo << *aggregate* >> mostra que $TM3$, $TM5$, $TM6$, $TM8$, $TM9$ e $TM12$ relacionam-se ou estão agregados a função (ou grupo de funções) fornecida pelo requisito $TM1$ que controla o padrão de tráfego.

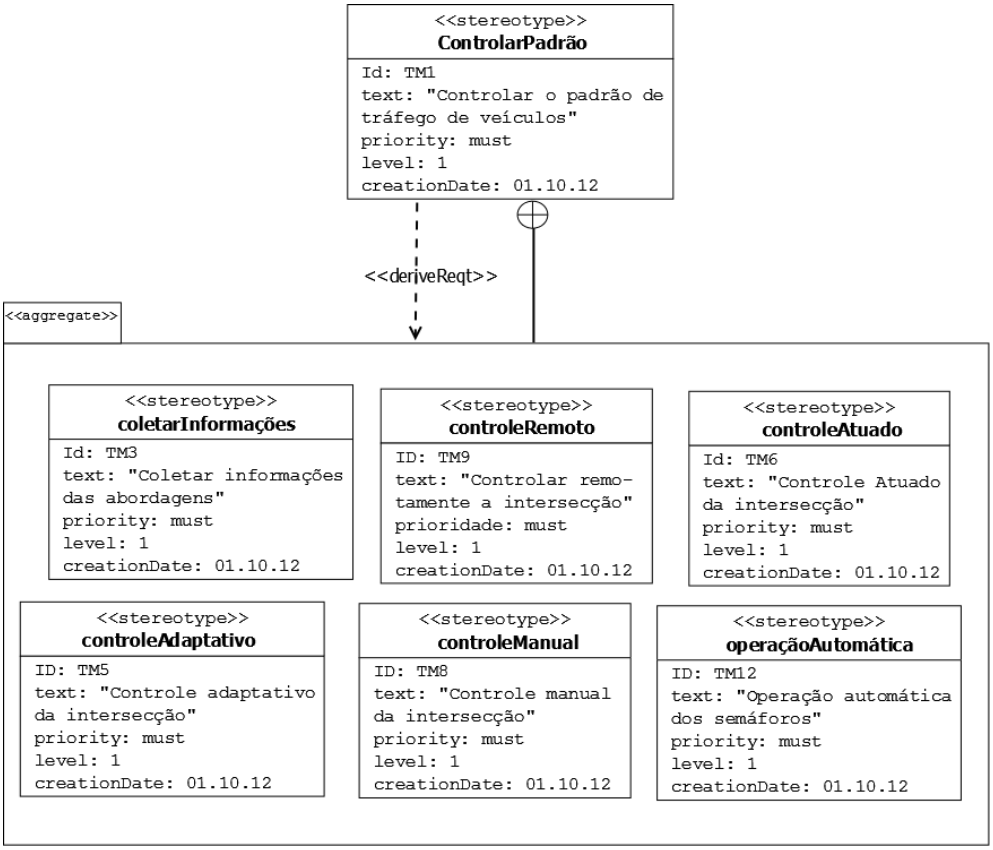

Figura 4.7: Requisitos em Alto Nível e seus Relacionamentos com Pacote.

A aplicação do metamodelo no estudo de caso é apresentada na Figura 4.8, em que é representado explicitamente o uso do relacionamento << *linkage* >> entre requisitos mais abstratos, funcionais e não-funcionais. É possível observar, através do novo relacionamento, quando um requisito em qualquer nível de abstração está interligado com outros requisitos em diferentes níveis da hierarquia como, por exemplo, a forte ligação entre o requisito de alto nível $TM6$ com o requisito TMNFR6.1 evidenciada pelo relacionamento << *linkage* >>.

Qualquer requisito que possua em suas extremidades o relacionamento << *linkage* >> indica possivelmente um ponto de conexão entre requisitos de mais alto nível ou, ainda, com requisitos menos abstratos.

Como pode ser observado na Figura 4.8, os requisitos em que estratégias de concorrên-

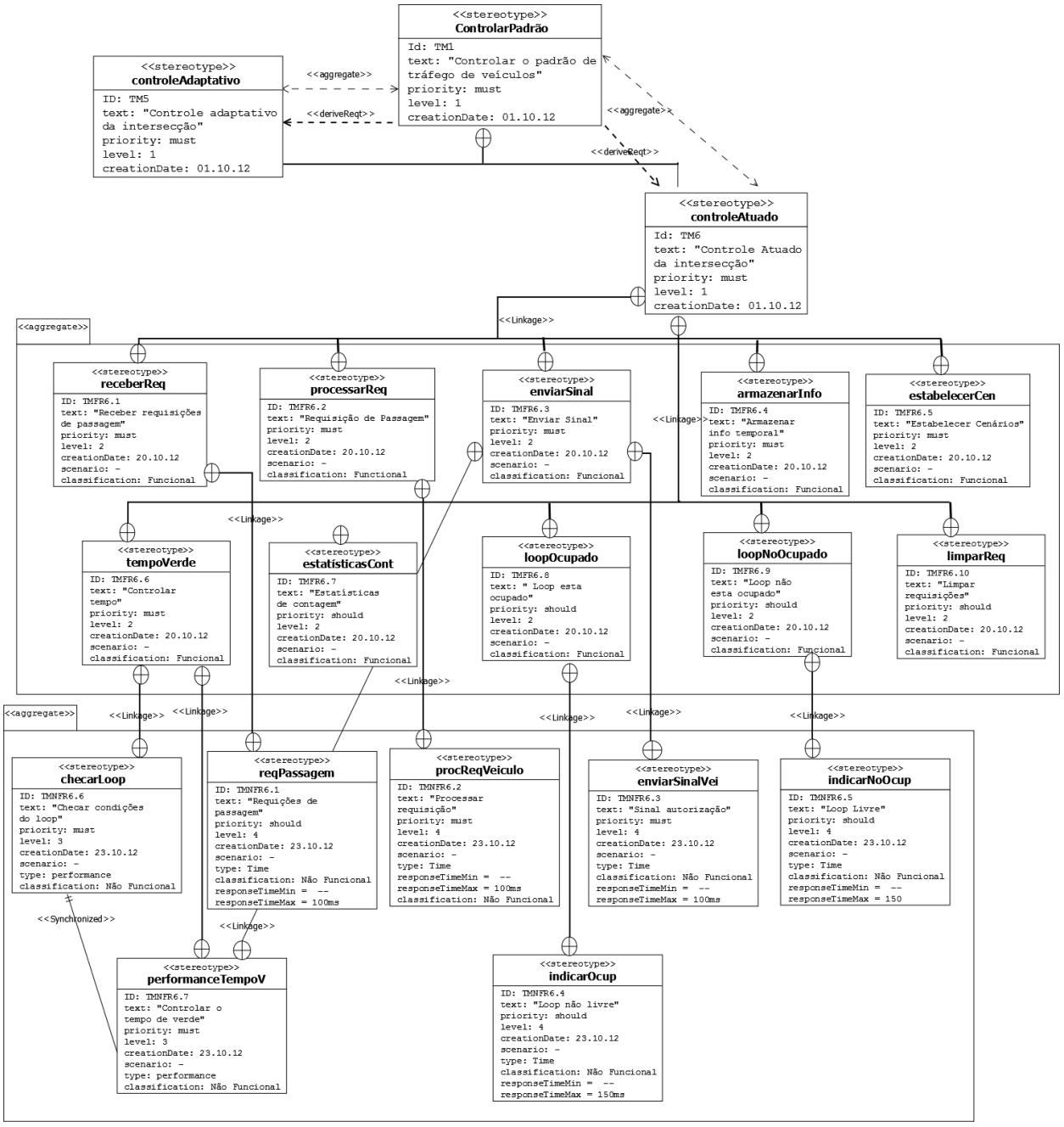

Figura 4.8: Aplicação do Metamodelo.

cia e paralelismo devem ser aplicadas, em seu desenvolvimento, podem ser representados com o relacionamento << *synchronized* >>.

Na Figura 4.9, a hierarquia entre requisitos é claramente expressa por meio do relacionamento << *linkage* >>. Assim, a árvore para explicitar a rastreabilidade entre requisitos pode ser modelada a partir deste relacionamento.

O relacionamento *linkage*, diferentemente do relacionamento *trace* original da SysML, fornece uma semântica bem definida, tendo em vista que permite observar claramente os níveis de hierarquia ou de rastreamento de um conjunto de requisitos. O rastreamento é mostrado pelas hiperligações de um requisito básico (requisito cliente), quer funcional ou

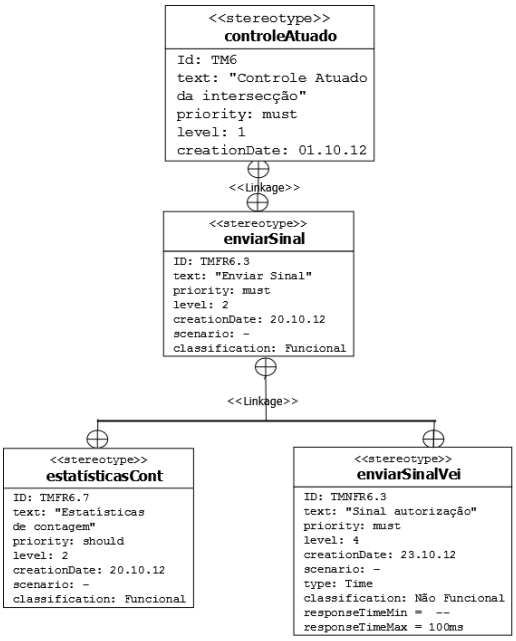

Figura 4.9: Rastreabilidade entre Requisitos em diferentes Níveis.

não-funcional, com os outros em níveis superiores (requisito fornecedor) ou, ainda, com requisitos em níveis inferiores. Também é possível, na modelagem realizada, observar a classificação e a organização de tais requisitos.

A vantagem destes novos relacionamentos e estereótipos é que a complexidade do *design* de software pode ser tratada a partir das fases iniciais do desenvolvimento pela classificação (em que os requisitos não-funcionais, por exemplo, são identificados e podem ser manipulados diretamente) e através do detalhamento de vários atributos relevantes para requisitos contidos no projeto de um sistema de tempo-real. Sendo assim, a complexidade envolvida no processo de *design* pode ser minimizada pela expressiva especificação, detalhamento e identificação de requisitos no processo de engenharia de requisitos.

4.6 Contribuições do Capítulo

A primeira contribuição é a aplicação de um diagrama específico para modelagem de requisitos e seus relacionamentos em diferentes níveis de detalhes. Os relacionamentos entre requisitos são apresentados no mesmo nível de abstração compondo, no documento de requisitos, uma hierarquia de requisitos e em diferentes níveis de detalhes. Esta contribuição é fornecida por meio do uso do diagrama de requisitos da SysML com novos relacionamentos para modelar requisitos.

O metamodelo do diagrama de requisitos original (ver Capítulo 2) é estendido com propriedades adicionais, representadas pelos novos atributos, com objetivo de expressar as propriedades inerentes aos STR. Extensões aos relacionamentos do diagrama de requisitos da SysML são realizadas com novos estereótipos que melhoram a rastreabilidade entre requisitos.

O relacionamento *trace* da SysML não apresenta explicitamente como os elementos modelados estão interligados sendo fraca a noção de rastreabilidade fornecida pelo mesmo. Neste sentido, os novos relacionamentos criados adicionam maior clareza nas relações entre requisitos do modelo e permite, também, expressar como os requisitos estão integrados e a quais outros elementos do modelo estão ligados.

Capítulo 5

Modelagem de Requisitos com SysML e MARTE

Neste capítulo, que se baseia no artigo "*An Approach for Modeling Real-Time Requirements with SysML and MARTE Stereotypes*" [Ribeiro e Soares 2013b], tem-se o objetivo de explorar o uso dos *profiles* SysML e MARTE em conjunto para a modelagem, classificação e documentação de requisitos de software de tempo-real e descrever a capacidade de MARTE e de seus construtores, para considerar diferentes aspectos inerentes a softwares de tempo-real. O metamodelo geral para modelagem SysML e MARTE é detalhado na Seção 5.1. Os conceitos da visão de domínio MARTE são concretizados em um conjunto de estereótipos relacionados, sendo estes explicados na Seção 5.2. As extensões realizadas na ferramenta de modelagem para suportar a modelagem de requisitos com SysML e MARTE são abordadas na Seção 5.3. Na Seção 5.4 é proposto o estudo de caso para modelagem e especificação, por meio do metamodelo criado, de um conjunto de requisitos de um sistema de tempo-real.

5.1 Proposta de Metamodelo com SysML e MARTE

A proposta neste capítulo é mostrar uma nova extensão da SysML, mais especificamente do diagrama de requisitos da SysML, com estereótipos MARTE para representar uma linguagem que possibilite a descrição e modelagem de requisitos de softwares de tempo-real. O metamodelo criado que contempla a combinação dos *profiles* SysML e MARTE é apresentado na Figura 5.1.

O diagrama de requisitos da SysML foi estendido para permitir novas representações para requisitos de software. Diversos novos atributos foram criados para o diagrama de requisitos com base em declarações contidas no padrão IEEE $830 - 1998$ que define características chaves para requisitos funcionais e não-funcionais [IEEE 1998].

Um requisito estendido (representado pelo estereótipo $<<$ *ExtRequirement* $>>$)

é proposto com oito atributos adicionais característicos/relevantes a todos os tipos de requisitos propostos para um sistema de tempo-real. O requisito representado por << *ExtRequirementNRF* >>, que é derivado do requisito anterior, é uma extensão para representar requisitos não-funcionais e possui três novos atributos. Três tipos de requisitos não-funcionais específicos também são propostos no metamodelo sendo representados, respectivamente, por << *Timing* >>, << *Performance* >> e << *Safety* >>.

Os atributos de << *Requirement* >> são **id** e **text**. O atributo **id** representa o identificador único do requisito e brevemente indica seu contexto e o atributo ***text*** é uma pequena caracterização do requisito.

Os novos atributos definidos para << *ExtRequirement* >> são : ***priority***, ***type***, ***abstractLevel***, ***constraint***, ***scenario***, ***creationDate***, ***modificationDate*** e **versionNumber**.

O atributo ***priority*** define a prioridade de um requisito em relação aos demais, indicando a ordem com que um requisito deve ser tratado. Podem ser atribuídos valores do tipo *string* a este requisito, incluindo prioridades do tipo *must* (deve), *should* (pode), *could* (poderia) e *won't* (não necessário).

O atributo ***type*** é utilizado para descrever características especiais de um requisito como, por exemplo, se o requisito define o comportamento do sistema, se representa algum estado especial do sistema, se esta relacionado a algum evento externo ou, ainda, se o requisito representa elementos temporizados (*clocks*). O atributo ***abstractLevel*** mostra, para fins de classificação e rastreamento, qual é o nível de abstração de um requisito. O atributo ***constraint***, permite mostrar os requisitos que possuem algum tipo de restrição. Este é um atributo do tipo booleano e, para os casos que o mesmo possuir valor de verdadeiro (*true*), o seu identificador (ID) e a descrição detalhada desta restrição estão contidos em uma tabela de restrições. **Scenario** é um atributo do tipo String que basicamente identifica o cenário ao qual o requisito está relacionado. ***CreationDate*** e ***modificationDate*** são atributos do tipo string que, respectivamente, descrevem a data de criação do requisito e a data de modificação do mesmo. Por fim, o atributo ***versionNumber*** permite determinar a data de criação/elaboração de um requisito. Os últimos três atributos descritos são importantes por definirem uma extensão de controle para verificação da integridade de um requisito a partir das mudanças realizadas ao longo da especificação.

A modelagem de requisitos por meio do diagrama de requisitos da SysML com estereótipos MARTE torna-se possível com a importação do pacote MARTE *Foundations* que tem por objetivo definir conceitos fundamentais para sistemas embarcados e de tempo-real e, mais especificamente, de seus subpacotes *CoreElements*, *Non-Functional Properties* e do subpacote *Time*.

O estereótipo << *RequirementFunc* >> é utilizado para descrição de requisitos funcionais de um software. Seu atributo *classification* classifica um requisito em funcional,

não-funcional ou de domínio.

O estereótipo << *RequirementNFR* >> é utilizado para descrição de requisitos não-funcionais de um software. Seus atributos são: ***externalFac***, ***cost***, ***levelQoS*** e *classification*. O atributo ***externalFac*** objetiva estabelecer se um requisito é dependente de algum fator externo para que o mesmo seja desenvolvido. É um atributo do tipo string que objetiva fornecer uma breve descrição do fator de dependência. O atributo ***cost*** permite estabelecer critérios de custos para satisfazer um requisito e que podem influenciar futuras decisões quanto à viabilidade de seu desenvolvimento. *Cost* é um atributo do tipo string que pode assumir os valores *High*, *Medium* e *Low*. ***LevelQoS*** mostra o nível de qualidade exigido para um requisito. ***LevelQoS*** é um atributo do tipo string que pode assumir os valores *High*, *Medium* e *Low*. O atributo *classification* possui a mesma semântica já descrita anteriormente.

O tipo << *Timing* >> para um requisito não-funcional fornece descrições temporais relacionadas ao requisito. Seus atributos são ***typeTime*** que podem possuir apenas dois valores para designar se este requisito caracteriza tempo físico ou tempo lógico, ***minResponseTime*** é um atributo do tipo *float* que representa o tempo mínimo exigido para este requisito e ***maxResponseTime*** determina o tempo máximo que pode ser atribuído ao requisito. Os dois últimos atributos descrevem restrições de tempo relacionadas ao requisito.

O tipo << *Performance* >> possui três atributos. O atributo do tipo *float* ***RespTime*** descreve o tempo máximo de resposta associado a um requisito. Seu valor permite estabelecer qual nível de desempenho deve ser associado ou garantido pelo requisito. O atributo ***capacityOp*** vincula a um requisito qual o possível número máximo de operações simultâneas que estão relacionadas a esta função em um dado período de tempo (exemplo, quantidade de relatórios gerados para armazenamento, operações por segundo, entre outros). O atributo ***recoveryTime*** representa o tempo máximo exigido para recuperação da(s) função(ões), relacionadas ao requisito modelado, em caso de falha.

O tipo << *Safety* >> de um requisito não funcional possui como atributos ***integrity***, ***acessLevel*** e ***limitedC***. O atributo do tipo string ***integrity*** identifica o nível de integridade que deve ser garantido para um requisito. Este pode ter os seguintes valores atribuídos: *High*, *Medium* e *Low*. Em casos que a integridade é demarcada como alta, já se estabelece, desde cedo, a necessidade de checar as variáveis críticas. ***AcessLevel*** permite, antecipadamente, estabelecer o nível de acesso de um *stakeholder* ou de um grupo de *stakeholders* a(s) função(es) do sistema. ***AcessLevel*** é um atributo do tipo string que pode assumir os valores *High*, *Medium* e *Low*. Finalmente, o atributo ***limitedC***, do tipo booleano, permite mostrar se a comunicação de um requisito deve ser limitada em relação às outras funções/módulos do sistema e/ou vice-versa. O atributo ***limitedC*** possibilita, desde as atividades de especificação, direcionar decisões quanto ao projeto da arquitetura do sistema.

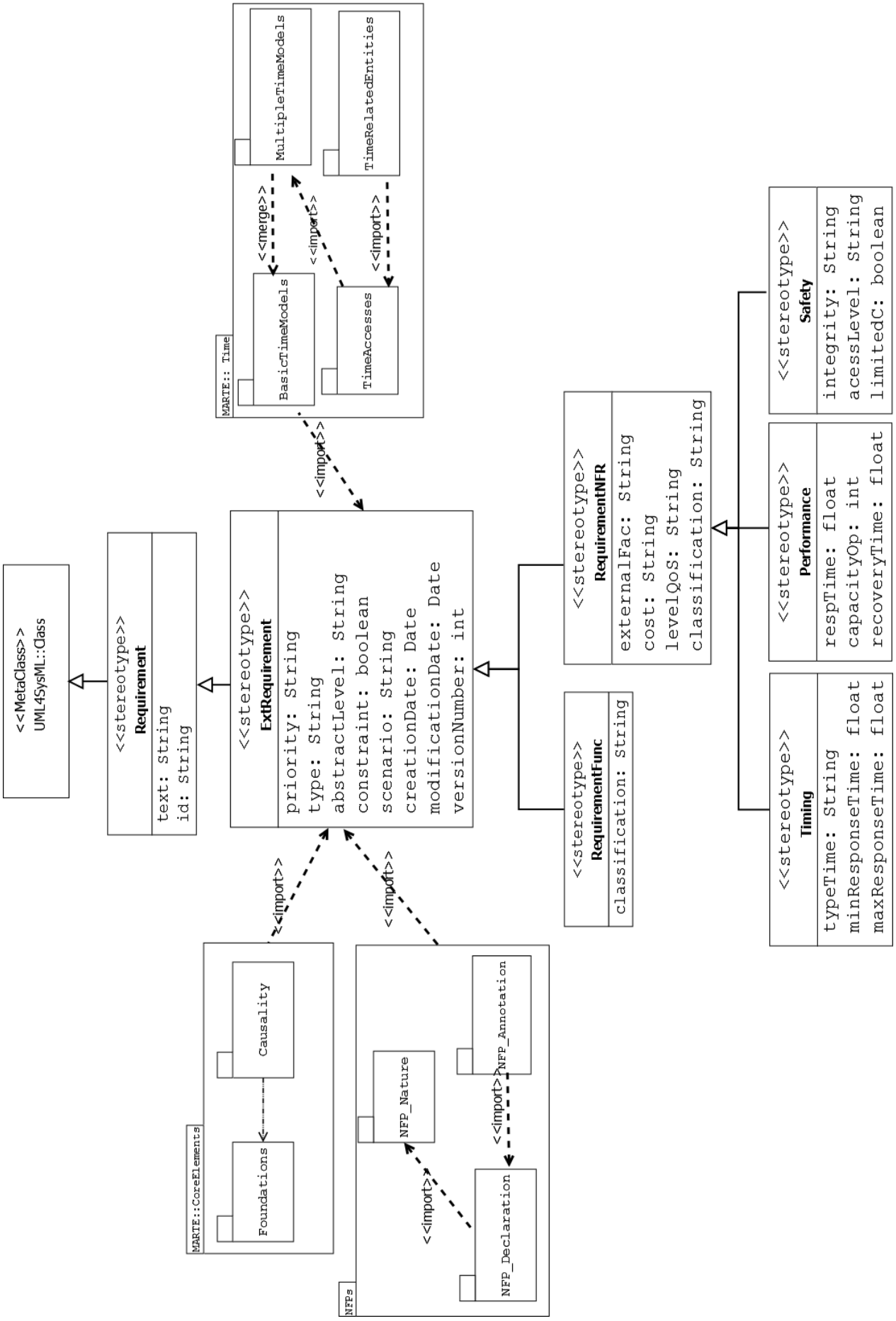

Figura 5.1: Metamodelo para Modelagem de SysML com MARTE.

5.2 Conjunto de Estereótipos MARTE

Na maioria dos casos, os conceitos definidos na visão domínio para os subpacotes do pacote *Marte Foundations*, em especial os subpacotes *CoreElements*, *Non Functional Proprieties* e *Time*, apresentados no Capítulo 2, são representados em MARTE por meio de um estereótipo que estende um elemento de modelagem UML concreto. Quando este for o caso, o classificador ou instância de natureza intrínseca de elemento UML anotado pode levar a identificar a natureza correspondente, a semântica ou variações concretas do conceito MARTE que deverá ser representado com a anotação.

SubPacote	Modelo de Domínio	Nome Estereótipo	Definição
Causality	Modal Behavior	Configuration	Representa a configuração do sistema podendo ser definida por um conjunto de elementos ativos do sistema (por exemplo, componentes de aplicação, componentes da plataforma, recursos de hardware), e/ou por um conjunto de parâmetros de operação (por exemplo, os parâmetros de QoS ou parâmetros funcionais).
Causality	Modal Behavior	Mode	Identifica um segmento operacional dentro do sistema de execução que é caracterizado por uma determinada configuração. Trabalhar em um determinado modo pode implicar que um conjunto de entidades do sistema são ativos durante aquele fragmento operacional.
Causality	Modal Behavior	ModeBehavior	Especifica um conjunto de modos mutuamente exclusivos. Sua dinâmica é representada por modos de conexão por meio de *ModeTransitions*.
Causality	Modal Behavior	ModeTransition	Descreve o sistema modelado sob modo de comutação. Um *ModeTransition* pode ser produzida em resposta a uma *Trigger*.

Tabela 5.1: Pacote MARTE *Foundations*: Estereótipos para *CoreElements*

As extensões da UML necessárias para apoiar os conceitos definidos na visão de

domínio MARTE e as descrições semânticas associadas são apresentadas no decorrer desta seção. Assim, o conjunto de extensões para apoiar *CoreElements* (Tabela 5.1), NFPs (Tabela 5.2) e *Time* (Tabela 5.3 e Tabela 5.4), para modelagem com UML, são organizados e abordados de acordo com o contexto da aplicação dos conceitos de domínio que se relacionam com a especificação de requisitos de STR.

O pacote *Time* define diversas anotações temporais para elementos do modelo, como *clocks*, elementos temporizados, especificação de valores de tempo, entre outros. As Tabelas 5.3 e 5.4 objetivam listar os conceitos concretos que permitem anotações de conceitos relacionados à estrutura de tempo, já elucidados no modelo de domínio do pacote *Time* (Capítulo 2) e também define conceitos que permitem reconhecer a adequação de cada estereótipo para ser utilizado como um elemento concreto na modelagem.

Modelo de Domínio	Nome Estereótipo	Definição
NFP_Anotation	Nfp_Constraint	Objetiva aplicar uma condição ou restrição aos elementos modelados. Especifica as restrições NFP que apoiam expressões textuais para especificar afirmações sobre a programação, desempenho e outras características dos sistemas de tempo-real e sua relação com os demais elementos de modelagem por meio de variáveis matemáticas, lógicas e expressões de tempo.
NFP_Declaration	Nfp	Destina-se a declarar, qualificar e atribuir tipos de dados estendidos aos valores *NFPs*.
NFP_Declaration	NfpType	Um tipo de *NFPs* é um tipo cujas instâncias são identificadas apenas por especificações de valor *NFPs*. Um tipo de Nfp contém atributos específicos para apoiar a modelagem de tipos da tupla *NFPs*.
NFP_Nature	Dimension	Estabelece um relacionamento entre uma quantidade e um conjunto de quantidades base em um dado sistema de quantidades.
NFP_Nature	Unit	É um qualificador de valores medidos em termos de quais magnitudes de outras quantidades (que tem a mesma dimensão física) podem ser declaradas.

Tabela 5.2: Pacote MARTE *Foundations*: Estereótipos para *Non- Functional Properties*

Modelo de Domínio	Nome Estereótipo	Definição
TimedRelatedEntities	TimedElement	Trata-se de estereótipo abstrato que deve ser utilizado para associar/referenciar um(s)*Clock*(s) a um elemento do modelo.
TimeAcess	Clock	Introduz um conceito geral de *clock*. É um elemento do modelo que representa uma instância de *clockType* dando acesso ao tempo. A propriedade *TimedStandarKind* é uma enumeração que referencia o sistema de tempo (físico/lógico)
«metaclass»	ClockType	É um classificador para *Clock*. Define atributos para especificar: a natureza de um valor de tempo (discreto/denso), o sistema de tempo (fisico/lógico), os tipos de unidades aceitas por um *clock*, o máximo valor e valor de deslocamento de um *Clock* e a resolução do *clock* associado.
TimeAcess	TimedValueSpecification	É a especificação de um conjunto de instâncias de *TimeValue*. A propriedade opcional *interpretation* pode forçar a interpretação do valor como uma duração ou especificação de um instante. Como um *TimedElement*, um *TimedValueSpecification* faz referências a *Clock*.
TimedRelatedEntities	TimedConstraint	Representa restrições impostas a qualquer valor de instante ou no valor de duração associado a elementos do modelo ligados a *clocks* de acordo com o valor dado ao atributo *interpretation* (instante literal, enumeração ou duração literal enumeração). Como um *TimedElement*, um *TimedConstraint* faz referências a*Clock*. É um estereótipo especializado de NFP_Modeling::NFP_Annotation::NfpConstraint
TimedRelatedEntities	ClockConstraint	Objetiva impor dependências entre *Clocks* ou entre tipos de *Clock*. Como um *TimedElement*, um *ClockConstraint* faz referêcias a *Clock*. É um estereótipo especializado de NFP_Modeling::NFP_Annotation::NfpConstraint

Tabela 5.3: Pacote MARTE *Foundations*: Estereótipos para *Time* I

Modelo de Domínio	Nome Estereótipo	Definição
TimedRelatedEntities:: TimedEventsModels	TimedEvent	Representa *Events* cujas ocorrências são explicitamente ligadas a *Clocks*. Seu atributo opcional *repetition* refere-se a um fator de repetição, isto é, o número de ocorrências sucessivas do *TimedElement*.
TimedRelatedEntities:: TimedProcessingModels	TimedProcessing	Representa atividades em que se têm conhecimento de seu tempo de início e término ou com uma duração conhecida e cujos instantes são explicitamente ligados a *clock*.
TimedRelatedEntities:: TimedObservation	TimedInstantObservation	Denota um instante no tempo associado com a ocorrência de um evento e observado, por meio de *obskind*, por um dado *clock*. Por ser uma especialização de *TimedElement* um *TimedInstantObservation* refere-se a *Clocks*.
TimedRelatedEntities:: TimedObservation	TimedDurationObservation	Denota algum intervalo de tempo associado com a execução, solicitação ou ocorrência de eventos observados em um ou dois relógios. O tipo preciso de evento pode ser determinado pelo seu atributo *obskind*. Por ser especialização de *TimedElement* um *TimedDurationObservation* refere-se a *Clocks*.
UML::Classes:: kernel::Namespace	TimedDomain	Referem-se a elementos do modelo que podem referir a *clocks* para expressar que seu comportamento depende do tempo.

Tabela 5.4: Pacote MARTE *Foundations*: Estereótipos para *Time*II

5.3 Descrição da Ferramenta de Apoio

O ArgoUML (disponível em [Argo]) é um software de diagramação de propósito geral para uso na análise e *design* de sistemas de software Orientado à Objetos. Sua interface pode ser observada na Figura 5.2.

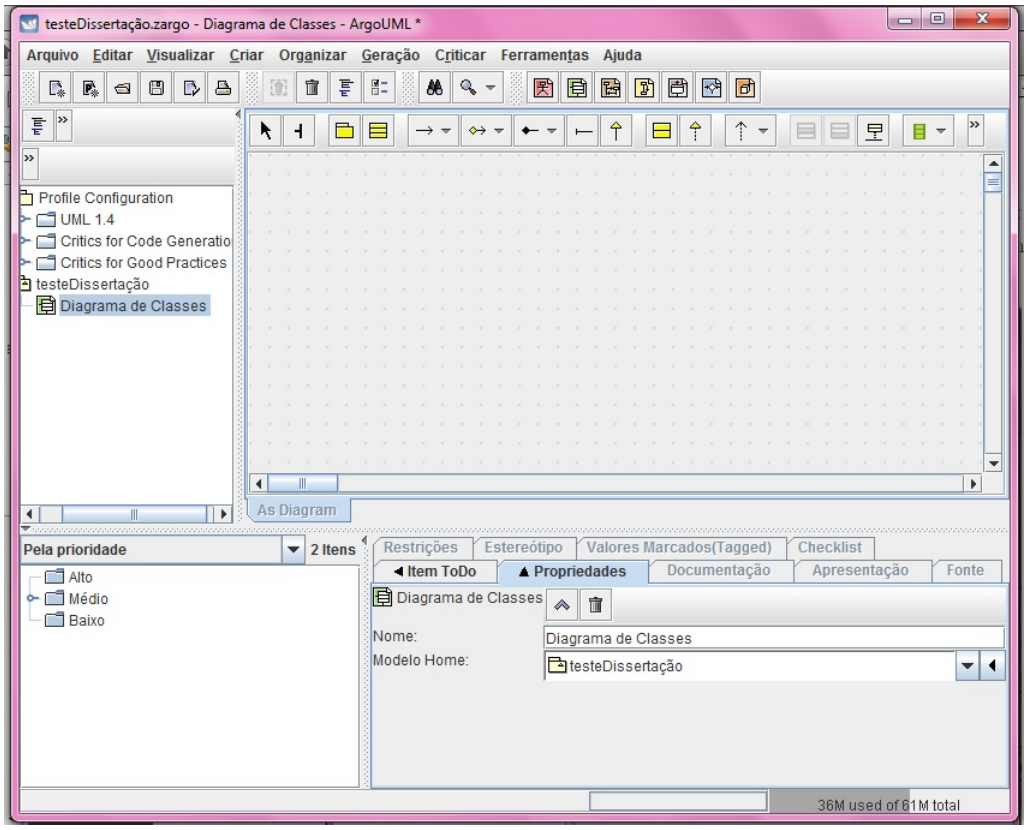

Figura 5.2: Interface da Ferramenta ArgoUML.

A funcionalidade do ArgoUML é dar suporte a edição na modelagem de diferentes tipos de diagramas, como por exemplo, diagramação de sistemas Orientado a Objetos. Trata-se de um software de código aberto e multi-plataforma (como por exemplo, GNU/Linux, diversas variações de Unix e Windows). É programa baseado em GTK+ (um kit de ferramentas multi-plataforma para a criação de interfaces gráficas) para criação dos diagramas e liberado sob a licença BSB (*Berkeley Software Distribution*).

O ArgoUML, atualmente na versão 0.26, foi desenvolvido utilizando a linguagem Java, sendo composto por aproximadamente 2.000 classes e 15.000 métodos [Argo]. Este software foi escolhido neste trabalho, após uma vasta pesquisa em diversas ferramentas de modelagem gráfica, entre outros fatores por ser desenvolvido na linguagem JAVA e, principalmente, por ser de código fonte livre.

Para facilitar a análise e compreensão do código da ferramenta ArgoUML possibilitando a alteração e adaptação do mesmo para a modelagem proposta neste trabalho foram aplicadas as técnicas de Software *Reconnaissance* e de análise da documentação do ArgoUML incluindo os tutorias disponíveis e exame do código fonte.

A técnica de Software *Reconnaissance* analisa rastros de execução do sistema para encontrar componentes do programa de interesse [Robbins et al. 2005]. Esta técnica facilita encontrar no programa os locais em que uma determinada função de interesse está implementada.

Para implementar as novas funções na ferramenta ArgoUML foi utilizada a técnica de clone de código. Os clones são uma porção do código-fonte reutilizada em outra parte de um programa. Neste contexto, foi realizada a adaptação do código fonte do ArgoUML, em especial do diagrama de classes (pacote *UMLClassDiagram*), para prover a modelagem do diagrama de requisitos da SysML.

As alterações na ferramenta ArgoUML, realçadas na Figura 5.3, foram realizadas adicionando um novo botão a barra de ferramentas do ArgoUML. Este botão gera um comando que é enviado para um controlador (CoreFactoryMDRImpl) que cria efetivamente o diagrama de requisitos da SysML no "palco" do ArgoUML. O diagrama de requisitos é basicamente uma extensão ao diagrama de classes, já passível de modelagem com ArgoUML, que possui o estereótipo << *Requirement* >> e que não exibe o container de operações.

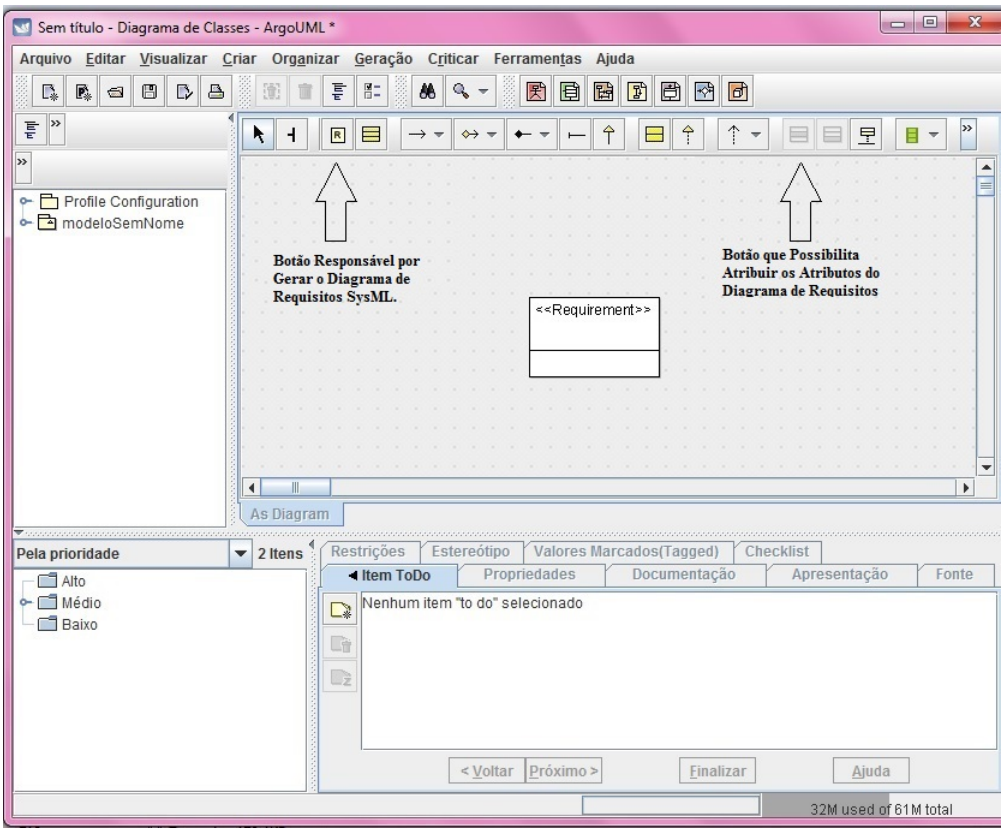

Figura 5.3: Interface da Ferramenta ArgoUML Estendida.

A Tabela 5.5 lista em termos gerais as principais modificações realizadas no código fonte do ArgoUML e descreve o propósito para cada alteração.

Os relacionamentos criados no metamodelo SysML estendido (Capitulo 4) não foram implementados nesta extensão, uma vez que o ArgoUML foi baseado num conjunto de

Elemento Modificado	Propósito
org.argouml.application.helpers. ResourceLoaderWrapper	Responsável por mapear os recursos do novo botão como uma string identificadora, texto de rótulo (label) e ícone.
org.argouml.model.mdr. CoreFactoryMDRImpl	Classe que recebe o comando lançado pela interface com o usuário, criando um diagrama de classe e realiza as operações necessárias para transformá-lo em um diagrama de requisitos.
org.argouml.uml.diagram. static_structure.ui.UMLClassDiagram	Adição dos comandos de criação do novo diagrama e adição do mesmo à interface gráfica.
org.argouml.model.mdr. MetaTypesMDRImpl	Classe central que mapeia os tipos usados no argo, modificada para mapear o novo tipo criado no sistema o Diagrama de Requisitos da SysML.
org.argouml.model.mdr. UmlFactoryMDRImpl	Classe do padrão *Factory* que recebe o comando de criação do diagrama de requisitos e chama a implementação concreta para a construção do mesmo.
org.argouml.uml.diagram.ui. FigCompartmentBox	Classe responsável por desenhar o diagrama de classes, que foi alterada para que quando o comando recebido seja de criação de um diagrama de requisito, não faça a chamada ao método que desenha o container de operações.
org.argouml.model.MetaTypes	Adição dos metodos de recuperação dos tipos.

Tabela 5.5: Modificações a Ferramenta ArgoUML

bibliotecas já predefinidas da UML. No código essas bibliotecas são importadas como org.omg.uml. Quaisquer alterações em termos de elementos gráficos devem ser feitas e/ou incluídas nesta biblioteca. Como esta biblioteca não encontra-se disponível para acesso em [Argo] não foi possível dentro do tempo disponível a pesquisa adicionar os novos relacionamentos e implementá-los na ferramenta ArgoUML estendida. No entanto, um e-mail foi enviado aos mantenedores da ferramenta para a disponibilização completa da biblioteca e conclusão das alterações.

5.4 Estudo de Caso Realizado

Esta seção objetiva avaliar a aplicabilidade do metamodelo proposto em um estudo de caso para um Sistema de Controle de Tráfego de Ruas. Inicialmente, os requisitos para possibilitar o controle adaptativo das interseções interconectadas e sincronizadas são apresentados. Esses requisitos descrevem o funcionamento de sinais de trânsito em rede. A Figura 5.4 retrata graficamente uma rodovia com n-interconexões dispostas e

conectadas. Posteriormente, os requisitos estabelecidos e os estereótipos, já classificados, são aplicados, por meio do metamodelo SysML e MARTE, para realizar a modelagem por meio da abordagem proposta.

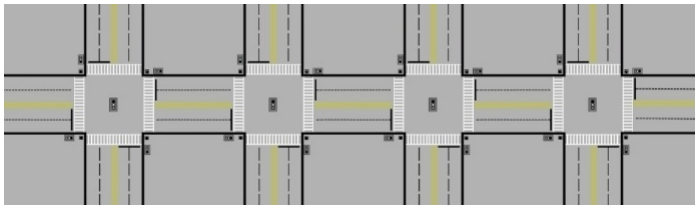

Figura 5.4: Intersecções em Rede de uma Rodovia.

5.4.1 Conjunto de Requisitos Considerados

Nesta seção, um subconjunto dos requisitos propostos no Capítulo 3, para um Sistema de Controle de Tráfego de Ruas (RTMS), é apresentado e posteriormente modelado e analisado. Tais requisitos são descritos utilizando a linguagem natural na Tabela 5.6 e representam, em linhas gerais, as características de um sistema de controle de intersecções de tráfego sincronizadas. O foco desses requisitos é determinar as capacidades de um controlador de tráfego adaptativo, bem como os requisitos funcionais e não-funcionais envolvidos no controle de tempo de verde dos semáforos e com a sincronização dos controladores em uma rede de interconexões.

Os requisitos da Tabela 5.6 representam, especificamente, um subconjunto de 20 requisitos derivados a partir dos 137 requisitos propostos no documento de requisitos elaborado no Capítulo 3.

5.4.2 Tabela SysML Proposta para Rastreamento de Cenários

O diagrama de casos de uso possibilita, por meio de uma linguagem simples, representar o comportamento externamente observável do sistema, bem como identificar os diferentes *stakeholders* que interagem com o sistema, ou seja, descreve os diversos cenários de uso do sistema. Na Figura 5.5, o diagrama de casos de uso modelando as funções principais do sistema proposto e as entidades externas que exercem influências em um cenário é apresentado.

É possível identificar a interconexão de um caso de uso com um elemento SysML, por meio do relacionamento *refine* (Figura 5.6). Este relacionamento fornece a capacidade para reduzir a ambiguidade entre requisitos, uma vez que o mesmo pode relacionar um cenário com seus requisitos mais específicos.

Na Figura 5.7 é apresentado um exemplo em que um caso de uso (nomeado como Sincronizar Intersecções) é refinado por um conjunto de requisitos descritos pelo diagrama

ID	NomeRequisito
TM1	O sistema deve controlar o padrão de tráfego de veículos na intersecção.
TM2	O sistema deve permitir a sincronização de semáforos.
TM3	O sistema pode coletar todos os tipos de informações das vias com o objetivo de avaliar convenientemente estes dados.
TM4	O sistema deve permitir a gestão do histórico de tráfego.
TM5	O sistema deve possuir o controle adaptativo da intersecção em resposta ao fluxo de tráfego.
TM6	O sistema deve coordenar o tempo de verde das intersecções de forma síncrona.
TM7	O sistema deve possuir o modo de preempção de emergência, ou seja, permitir a movimentação preferencial de veículos de emergência.
TM8	O sistema deve permitir o controle presencial da intersecção em resposta aos comandos de acionamento manual.
TM9	O sistema deve permitir o controle de intersecção em resposta a comandos de configuração remotos.
TM10	O sistema deve controlar os planos semafóricos das intersecções em rede.
TM11	O sistema de controle da intersecção deve ser capaz de interagir com o painel de controle do software.
TM12	O sistema deve calcular o atraso dos controladores das intersecções.
TM13	O sistema deve otimizar o fluxo de tráfego.
TM14	O sistema deve setar o modo/estado dos flags em resposta ao processamento atual do controlador da intersecção.
TM15	O sistema deve minimizar bloqueios ao longo da rodovia.
TM16	O sistema deve alterar de forma eficiente os planos para liberação dos pelotões de carros.
TM17	O sistema deve checar a demanda de tráfego com máxima precisão.
TM18	O sistema deve permitir a gestão de incidentes.
TM19	O sistema deve configurar o tempo da fase verde das intersecções.
TM20	O sistema deve sincronizar a fase de verde das intersecções precisamente. Com tempo mínimo de 50ms e tempo máximo de 150ms.

Tabela 5.6: Requisitos para um Sistema de Controle de Tráfego

de requisitos da SysML. A Tabela 5.7 descreve uma proposta para rastreamento de todos os cenários descritos na Figura 5.5 com seus requisitos relacionados.

A classificação proposta na Tabela 5.7 melhora a representatividade do relacionamento $<< refine >>$ da SysML. Assim, os relacionamentos entre requisitos são documentados desde o estabelecimento das funções iniciais do sistema e podem ser validados e rastreados ao longo do processo de desenvolvimento.

5.4.3 Aplicação do Metamodelo: SysML e MARTE para Documentação de Requisitos de Tempo-Real

Na Figura 5.8, o diagrama de requisitos da SysML estendido é utilizado com estereótipos MARTE para a modelagem de diferentes propriedades referentes aos requisitos

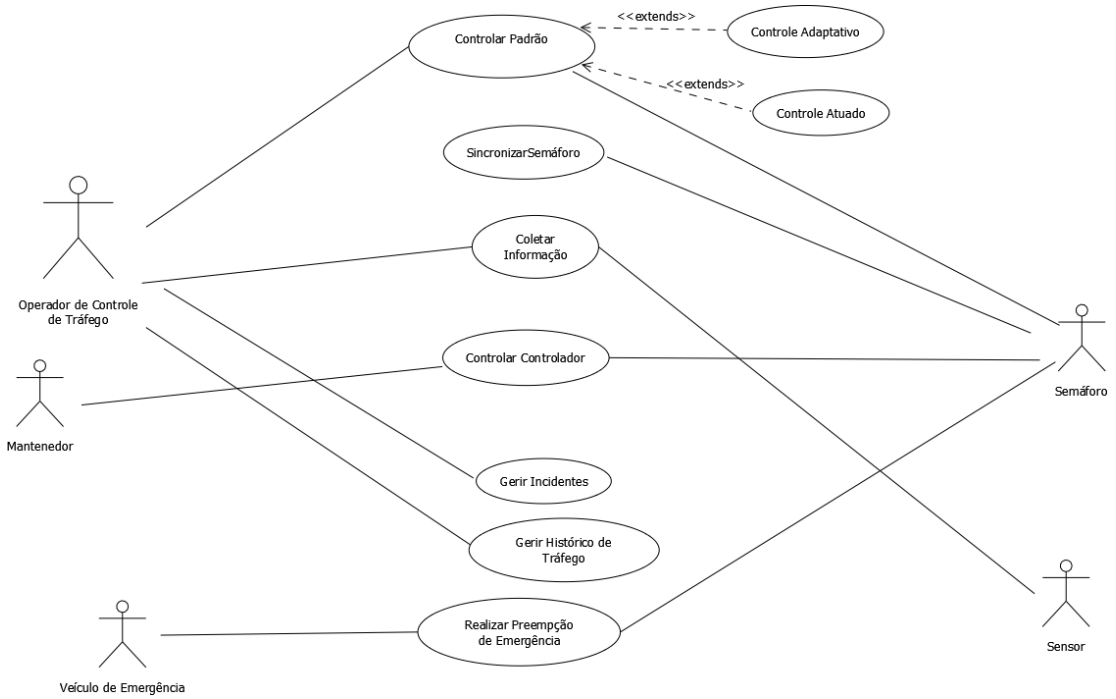

Figura 5.5: Cenários para o Sistema de Controle de Tráfego.

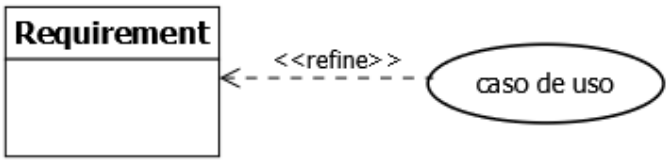

Figura 5.6: Relacionamento *Refine*.

Nome Caso de Uso	ID Cenário	Ator Relacionado	ID Requisito
controlarPadrão	Sc1	Operador	TM1, TM4, TM10, TM11, TM12, TM13, TM14, TM15,TM16, TM19, TM20
controleAtuado	Sc2	Semáforo	-
controleAdaptativo	Sc3	Semáforo	TM5
sincroSemaforo	Sc4	Semáforo	TM2, TM6, TM15, TM20
coletarInfo	Sc5	Operador, Sensor	TM3
controlarControlador	Sc6	Mantenedor, Semáforo	TM8, TM9, TM11
gerirInci	Sc7	Operador	TM18
gerirTráfego	Sc8	Operador	TM17
preempEmerg	Sc9	Veículo de Emergência, Semáforo	TM7

Tabela 5.7: Rastreamento entre Cenários

propostos para o sistema de controle de tráfego especificado na Tabela 5.6.

O metamodelo apresentado na Seção 5.1 é utilizado para modelar o subconjunto de

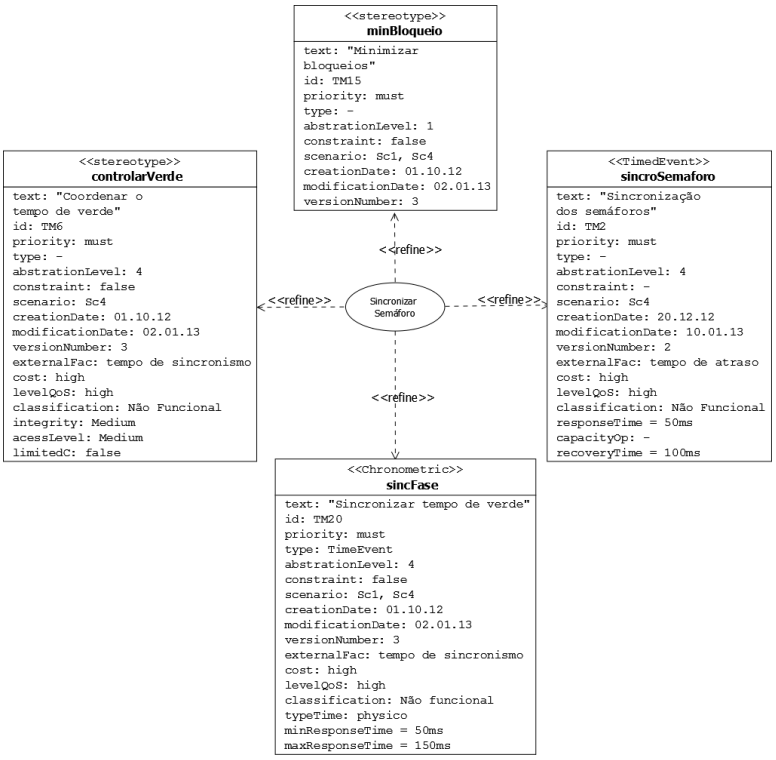

Figura 5.7: Caso de Uso Sincronização de Semáforos e sua Relação com Requisitos por meio do Relacionamento *Refine*.

requisitos listados. A modelagem proposta mostra como os requisitos característicos de sistemas de tempo-real podem ser modelados em diferentes níveis de abstração, como muitas propriedades inerentes a sistemas de tempo-real podem ser modeladas e representadas, bem como a classificação desses requisitos.

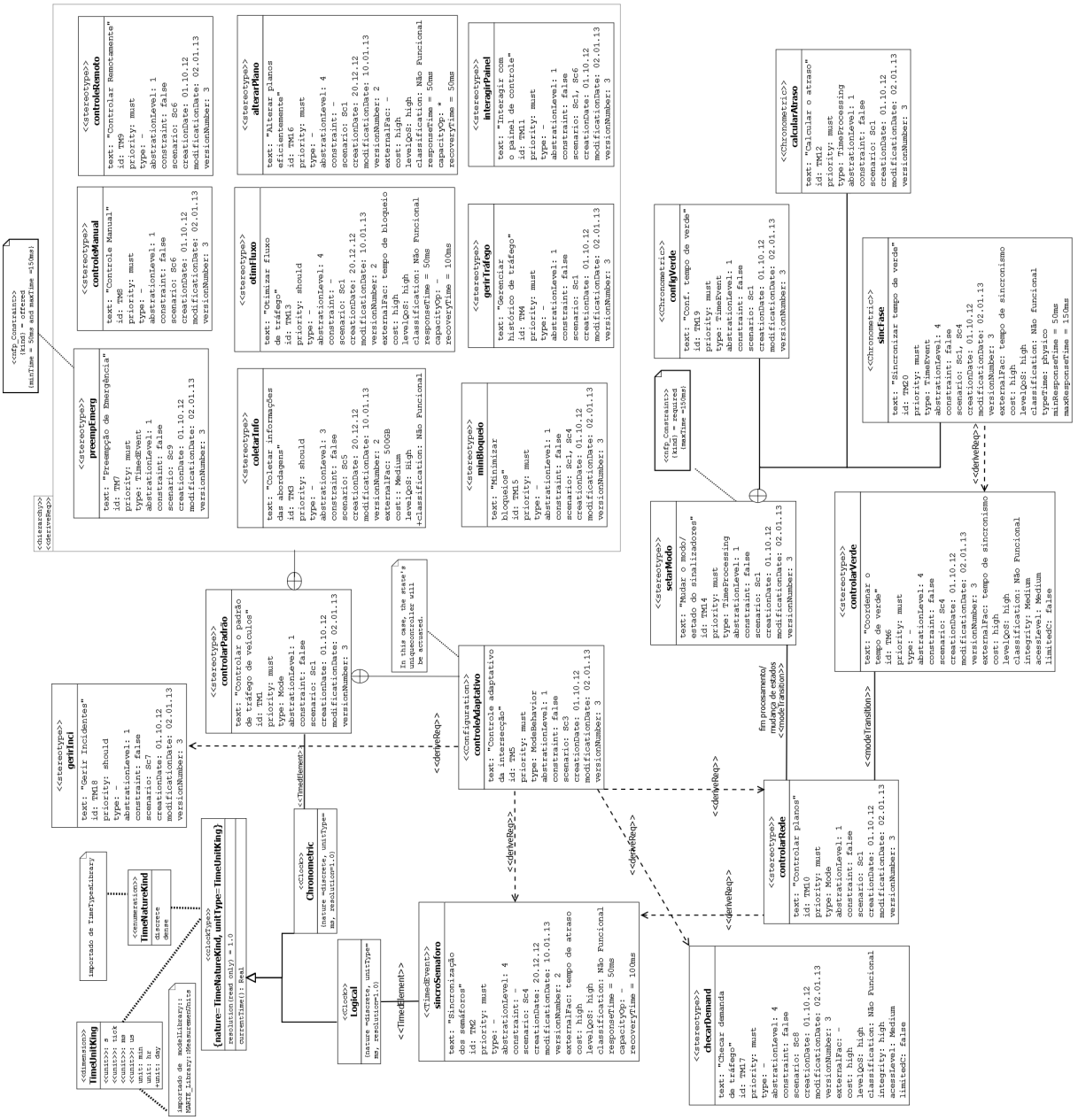

Figura 5.8: Modelo Final.

O requisito *TM1* possui como ***type*** o estereótipo MARTE *Mode* (de *Causality::-ModalBehavior*). Este requisito foi estereotipado desta maneira para destacar que o mesmo possui a lógica para representar e controlar um segmento operacional. Assim, *TM1* refere-se ao sistema de controle de tráfego que possui a configuração necessária para controlar todas as entidades/elementos ativos do fragmento operacional. Este requisito tem um relacionamento SysML << *hierarchy* >> e << *deriveReqt* >> com os requisitos elementares de um sistema de controle de tráfego, que são os requisitos *TM3*, *TM4*, *TM5* *TM7*, *TM8*, *TM9*, *TM11*, *TM15* e *TM17* e por questões de legibilidade da modelagem não foram representadas cada uma das associações (e sim diretamente ao seu pacote).

Vale destacar que o requisito *modePreemp* (*TM7*) tem uma restrição vinculada pelo estereótipo << $nfp\_Constraint$ >> (de *NFPs::Nfp_ Constraint*) aplicando uma restrição temporal a *TM7* para tentar "assegurar um bom desempenho" a um requisito crítico. Neste caso, a prioridade de passagem de veículos de emergência é indicada pelo atributo da restrição *kind = offered*, que representa o espaço de valores para apoiar/restringir este requisito (nesta modelagem, por se tratar de restrições simples opta-se por não utilizar a *Value Specification Language* (VLS) introduzida em MARTE para criar anotações em elementos do modelo). O requisto *TM7* tem o ***type*** *TimedEvent* (de *TimedRelatedEntities:: TimedEventsModels*), uma vez que o mesmo está relacionado a um evento cujo tempo de início e fim não são definidos a priori. No entanto, a decisão de atender a esta ocorrência está diretamente ligada ao *Clock*.

O requisito *TM1* tem um relacionamento estereotipado com << *TimedElement* >> (de *Time::TimedRelatedEntities*) com o *Clock Chronometric* (que é uma especialização de *ClockType*), ambos do pacote *Time::TimeAcess*. O estereótipo << *TimedElement* >> é importante, neste contexto, e deve ser utilizado sempre que for necessário associar um *Clock* a um elemento do modelo (neste caso o requisito *TM1*). O requisito *TM1* deve ser elaborado, respeitando diversas restrições temporais e, com isso, atender aos requisitos não-funcionais que definem critérios de confiabilidade e desempenho do sistema de controle de tráfego. Assim, é necessário criar um estereótipo << *ClockChronometric* >> que ao ser relacionado a *TM1*, por meio de << *TimedElement* >>, garante ao mesmo o acesso à estrutura de tempo. Os atributos de << *ClockChronometric* >> são ***nature***, uma enumeração de *TimeNatureKind::TimeTypesLibrary*, que possibilita especificar a natureza densa ou discreta de um valor de tempo (neste caso, o tempo é discreto), o atributo ***unitType***, valorado como *TimeUnitKing*, representa a dimensão da unidade de tempo (neste caso, a unidade é *ms*), associada ao relógio e é importada de *modelLibrary::MARTE_ Library::MeasuremenUnits*, e ***resolution*** que expressa a granulidade do *clock* (neste caso, por *default*, o valor atribuído é 1.0).

O modo de controle adaptativo é representado para o controlador descrito neste estudo de caso. O requisito *TM5* se relaciona hierarquicamente a *TM1* e define a configuração do controlador. Isto é explicitado pelo estereótipo << *Configuration* >> (

de *CoreElements:: Causality::ModalBehavior*). Este estereótipo mostra que a configuração (neste caso, modo de controle adaptativo) pode ser definida por um conjunto de elementos ativos do sistema. O atributo ***type*** do requisito *TM5* é *ModeBehavior* (de *CoreElements::Causality::ModalBehavior*), já que o modo de controle adaptativo configurado é único para o sistema de controle, ou seja, apenas um modo de controle é considerado nas intersecções deste estudo de caso.

O requisito *TM5* deriva *TM10*, que é responsável por controlar o plano semafórico das intersecções. O plano semafórico alterará, conforme o volume de tráfego e o respectivo processamento do mesmo. Assim, através da associação de *TM10* com *TM14* estereotipada com << *modeTransition* >> (de *CoreElements::Causality::ModalBehavior*), as transições entre os planos semafóricos, para um determinado estado no sinal (vermelho, verde, amarelo e verde estendido), são representadas.

O requisito *TM14* é do ***type*** *TimeProcessing* (de *Time::TimedRelatedEntities::TimedProcessing*). O ***type*** *TimeProcessing* é importante por salientar que a mudança/alteração dos estados dos sinalizadores é uma atividade em que há grande importância em conhecer/estabelecer o tempo de início e término para cada estado e, assim, criar estrátegias para garantir melhor desempenho quanto ao desenvolvimento deste requisito. A restrição de temporização para este requisito descreve, precisamente, que deve ser estabelecido um nível mínimo quantitativo (especificado por *kind = required*) de processamento (com no máximo $150ms$).

Os requisitos *TM19* e *TM20* são estereotipados com << *Chronometric* >> caracterizando-os como requisitos que referem-se ao tempo físico e são do ***type*** *TimedEvent* (de *TimedRelatedEntities:: TimedEventsModels*) por estarem associados a *TM14* (pelo relacionamento *hierarchy* da SysML) e por representarem *Events* específicos de mudança do estado dos sinalizadores. Por ser do ***type*** *TimedEvent*, suas ocorrências relacionam diretamento a um *Clock* (neste caso, um << *clockChronometric* >>.

O requisito *TM12*, do ***type*** *TimeProcessing* (devido ao fato de que seu tempo/instante de processamento refere-se explicitamente a *clocks*) refere-se à estrutura de tempo físico (e por essa razão é estereotipado como << *Chronometric* >>). É através deste requisito que o atraso (*offset*) é estabelecido entre o controlador de uma intersecção e das intersecções subsequentes.

Como pode ser observado, a maior parte desta especificação refere-se ao acesso a estrutura de tempo através de relógios físicos. No entanto, MARTE também permite modelagem de tempo lógico. Por esta razão, uma estrutura de tempo que especializa << *clockType* >> e define o tempo lógico é criada (<< *clockLogical* >>), sendo associada por meio de << *TimedElement* >> ao requisito *TM2* que é responsável por caracterizar a sincronização de semáforos em uma rede de intersecções. Esta relação torna-se necessária por definir que *TM2* não obedece um tempo físico pré estabelecido, pois *TM2* somente poderá ser definido por meio da leitura e da detecção do fluxo das abor-

dagens, processamento do volume de tráfego, estabelecimento das estratégias de controle de mudanças de planos e de diversos processamentos interligados. Logo, não é um tempo explicitamente físico e depende da lógica de processamento de vários outros elementos.

5.5 Contribuições do Capítulo

Com intuito de contribuir para as abordagens dirigidas a modelos para engenharia de requisitos de sistemas de tempo-real, este capítulo apresentou o uso combinado e de forma consistente dos *profiles* SysML e MARTE da UML para modelagem de STR.

A estrátegia de combinação fundamentou-se em criar uma metodologia de modelagem específica para requisitos de software que permita representar com completude e expressividade, através do metamodelo criado, a modelagem de requisitos em diferentes níveis de abstração e classificação e, ainda, as diferentes características de um STR, como requisitos temporais e requisitos de desempenho.

Assim, a combinação destes *profiles* na abordagem apresentada mostrou-se expressiva e completa para a representação de diversos requisitos. A SysML contribui com construtores para definir requisitos e as suas relações. Adicionalmente, MARTE completa a precisão do cenário com anotações não-funcionais bem formadas. Os conceitos da SysML e do MARTE, articulados no diagrama de requisitos da SysML, são altamente complementares, cobrindo muitos dos propósitos de especificação de STR.

Capítulo 6

Conclusão

Os sistemas de tempo-real têm sido estudados intensivamente, uma vez que a sociedade tem se tornado dependente destes sistemas e, consequentemente, de sua corretude e segurança. Como caracterizado no Capítulo 1, os sistemas de tempo-real possuem certas características que demandam abordagens específicas para o seu desenvolvimento. A especificação e modelagem destes sistemas requer um conjunto de notações heterogêneas que possibilitem avaliar questões de tempo, sincronização e dimensionamento destes softwares.

Para este domínio, que exige um alto nível de confiabilidade, são necessárias metodologias, métodos e ferramentas para lidar com os complexos problemas que surgem e para representar com completude e confiabilidade as características relacionadas a estes sistemas. Neste contexto, diversas linguagens de modelagem foram propostas e têm sido aplicadas no contexto das abordagens dirigida a modelos para sistemas de tempo-real. O uso de modelos no processo de desenvolvimento de sistemas de tempo-real permite representar o sistema sob diferentes perspectivas possibilitando na especificação do mesmo descrever os componentes de hardware e software, alocação destes componentes, bem como a possibilidade de integrar componentes heterogêneos no sistema.

No Capítulo 2, evidenciou-se que o uso de linguagens gráficas aumenta a inteligibilidade do sistema, uma vez que permite aos projetistas fornecer descrições de alto nível do sistema e facilmente ilustrar os conceitos internos dos componentes (hierarquia, conexões e dependências). Sendo assim, os *profiles* SysML e MARTE da UML (caracterizados no Capítulo 2) foram utilizados neste trabalho com intuito de contribuir para a modelagem de requisitos de softwares de tempo-real, através da extensão e/ou utilização conjunta destes *profiles*.

A aplicação da abordagem proposta foi realizada por meio da técnica de estudo de caso em um conjunto de requisitos para um Sistema de Controle de Sinais de Trânsito Urbano (RTMS). Os RTMS, apresentados no Capítulo 3, são sistemas críticos de software de tempo-real, em que vários elementos estão envolvidos, e são aplicados a uma infraestrutura crítica. O projeto desses sistemas deve abordar critérios de desempenho, confiabilidade e segurança, uma vez que vidas humanas estão diretamente envolvidas e que a infraestrutura

de transportes é fundamental para a economia de um país.

Neste trabalho, estratégias que contribuem para os processos de especificação de requisitos que visam identificar, classificar, descrever e relacionar requisitos de sistemas de tempo-real foram elaboradas.

A proposta elaborada no Capítulo 4 descreve extensões ao diagrama de requisitos da SysML, com novas propriedades, das Tabelas de Relacionamentos entre Requisitos (estendidas da Tabela de Requisitos da SysML) e de novos relacionamentos para o diagrama de requisitos da SysML, contribuindo, assim, para o rastreamento de requisitos em diferentes níveis de abstração. A introdução desta nova abordagem fornece uma ligação entre representações de requisitos em diferentes níveis de abstração e fundamenta a representação das relações de rastreabilidade e composição entre requisitos modelados.

Os *profiles* SysML e MARTE foram utilizados de forma conjunta, no Capítulo 5, para representar, através de diversos construtores e estereótipos nivelados a partir do *profile* MARTE e de seus subpacotes *CoreElements*, NFP e *Time*, uma estensão apropriada para representar com completude e consistência requisitos característicos a sistemas de tempo-real.

A completude na satisfação de requisitos em sistemas de tempo-real é fortemente dependente do acoplamento entre a função do sistema e o tempo. Portanto, conclui-se que os conceitos de SysML e MARTE, articulados por meio de seus diversos construtores são altamente complementares. A extensão proposta da SysML pode fornecer elementos para descrever requisitos de software e seus relacionamentos, gerenciamento de mudanças, evolução e gerenciamento de rastreabilidade. Em MARTE, há anotações de tempo que fornecem definições semânticas estreitamente relacionadas com o comportamento do sistema. Assim, o MARTE complementa a precisão da modelagem com anotações não-funcionais bem formadas importantes ao desenvolvimento de sistemas de tempo-real e que influenciam uma ampla série de decisões de projeto.

6.1 Avaliação Comparativa do Trabalaho

Foram classificados a partir da revisão de literatura, realizada neste trabalho, sobre o uso dos *profiles* SysML, MARTE e dos *profiles* SysML e MARTE conjuntamente 20 trabalhos no âmbito dos processos da engenharia de requisitos. Na Tabela 6.1, são listados o nome dos trabalhos relacionados, sua referência (para facilitar a busca de informações sobre o trabalho no Capítulo 1), o número de citações, segundo o *Google* Acadêmico, o ano de publicação do trabalho e, ainda, qual (is) *profile*(*s*) foi/foram utilizado(s) no trabalho em questão.

Nome Trabalho	Referência	N°Citações	Ano	Profile
Requirements Specification and Modeling through SysML	[Soares e Vrancken 2007]	16	2007	SysML
Introducing the Modeling and Verification Process in SysML	[Linhares et al. 2007]	13	2007	SysML
Model-Driven User Requirements Specification Using SysML	[Soares e Vrancken 2008]	30	2008	SysML
Reliability Study of Complex Physical Systems using SysML	[David et al. 2010]	22	2010	SysML
A Formal Model for the Requirements Diagrams of SysML	[Valles 2010]	3	2010	SysML
User Requirements Modeling and Analysis of Software-Intensive Systems	[Soares et al. 2011]	3	2011	SysML
Modeling Method of Military Aircraft Support Process based SysML	[Li et al. 2011]	0	2011	SysML
First Experiments Using the UML Profile for MARTE	[Demathieu et al. 2008]	42	2008	MARTE
Evaluating MARTE in an Industry-Driven Environment: TIMMO's Challenges for AUTOSAR Timing Modeling	[Espinoza et al. 2008]	7	2008	MARTE
MARTE Based Modeling Approach for Partial Dynamic Reconfigurable FPGAs	[Quadri 2008]	4	2008	MARTE
MARTE Mechanisms to Model Variability When Analyzing Embedded Software Product Lines	[Belategi et al. 2010]	7	2008	MARTE
Model-Based Methodology for Implementing MARTE in Embedded Real-Time Software	[Zaki e Jawawi 2011]	0	2011	MARTE
Refinement of UML/MARTE Models for the Design of Networked Embedded Systems	[Ebeid et al. 2012]	1	2012	MARTE
Experiences of Applying UML/MARTE on Three Industrial Projects	[Iqbal et al. 2012]	1	2012	MARTE
Modelagem de Software de Tempo Real utilizando o Profile MARTE da UML	[Silvestre e Soares 2012a]	0	2012	MARTE
Model-based Methodology for Requirements Traceability in Embedded Systems	[Albinet et al. 2007]	20	2007	SysML e MARTE
Model-based Design Space Exploration for RTES with SysML and MARTE	[Mura et al. 2008]	15	2008	SysML e MARTE
MADES FP7 EU project: Effective high level SysML/MARTE methodology for real-time and embedded avionics systems	[Quadri et al. 2012a]	3	2012	SysML e MARTE
Multi-view Power Modeling Based on UML, MARTE and SysML	[Gomez et al. 2012]	12	2012	SysML e MARTE

Tabela 6.1: Trabalhos Revisão de Literatura (Número de citações definido em 10/03/2013).

Destes 20 trabalhos, listados na Tabela 6.1, foram selecionados os trabalhos que abordavam estratégias de modelagem semelhantes à proposta neste trabalho, isto é, trabalhos que utilizassem em uma mesma metodologia de modelagem os *profiles* SysML e MARTE. Os trabalhos comparados à estratégia proposta neste trabalho são, respectivamente [Albinet et al. 2007], [Mura et al. 2008], [Quadri et al. 2012a] e [Gomez et al. 2012].

Um conjunto de critérios de comparação foi elaborado para avaliar as abordagens já descritas na revisão de literatura e a extensão do diagrama de requisitos da SysML com estereótipos MARTE para modelagem de requisitos de software. Os critérios de comparação, anteriormente citados, foram escolhidos com base nas especificações da [IEEE 1998] (detalhadas no Capítulo 2) para as metodologias e linguagens de modelagem de requisitos de software.

A comparação dos trabalhos listados anteriormente com a abordagem proposta neste trabalho é apresentada na Tabela 6.2. Os símbolos utilizados na Tabela 6.2 são explanados a seguir:

■ - significa que o critério avaliado é totalmente satisfeito;

⊡ - significa que o critério avaliado é parcialmente satisfeito;

□ - significa que o critério avaliado não é satisfeito.

Critérios de Comparação	**[Albinet et al. 2007]**	**[Mura et al. 2008]**	[Quadri et al. 2012a]	**[Gomez et al. 2012]**	Abordagem Proposta
Agrupamento de Requisitos	□	□	□	□	■
Classificação de Requisitos	■	□	⊡	⊡	■
Diagrama de Requisitos Específico	■	□	⊡	□	■
Extensibilidade do Modelo	□	□	□	■	■
Ferramentas CASE Disponíveis	■	⊡	■	⊡	⊡
Identificação de Conflitos	□	□	⊡	⊡	□
Modelagem de Requisitos Funcionais	⊡	⊡	■	⊡	■
Modelagem de Requisitos Não-Funcionais	⊡	⊡	⊡	■	⊡
Modelagem Gráfica	■	■	■	■	■
Modelagem de Tempo	⊡	⊡	⊡	⊡	■
Rastreabilidade com o Projeto	■	■	■	⊡	■
Rastreabilidade entre Requisitos	⊡	□	□	□	■
Relacionamentos entre requisitos	■	□	■	□	■
Relação com a UML	■	■	■	■	■
Representação de Restrições	⊡	□	⊡	⊡	■
Metodologia Específica para Requisitos	⊡	□	⊡	□	■

Tabela 6.2: Comparação entre abordagens para modelagem de requisitos.

Na Tabela 6.2 são apresentados os pontos fortes da abordagem proposta neste trabalho que utiliza os *profiles* SysML e MARTE conjuntamente para descrição de requisitos de sistemas de tempo-real.

Dentre os critérios relacionados podem ser destacados como pontos fortes deste trabalho os seguintes: o **agrupamento de requisitos** (através do novo relacionamento << *aggregate* >>), a **classificação de requisitos** (com a adição de um conjunto de atributos chaves ao diagrama de requisitos da SysML), a **modelagem expressiva de requisitos funcionais** e **parciais de requisitos não-funcionais** (nestes dois casos, através de extensão ao diagrama de requisitos da SysML e uso de construtores do *profile* MARTE), a **definição** (ou redefinição a partir de extensões dos relacionamentos da SysML e da extensão das Tabelas da SysML) **de questões de rastreabilidade** entre requisitos e entre requisitos e o projeto e, ainda, o **desenvolvimento de uma metodologia específica para requisitos**, o que é considerado uma vantagem, uma vez que especializa e define critérios específicos ao domínio descrito.

A ausência de um maior número de trabalhos que utilizem os *profiles* SysML e MARTE agregados influenciou fortemente no não atendimento ou no atendimento parcial de alguns critérios, pelos trabalhos listados, uma vez que ou são metodologias bem genéricas, ou objetivam representar questões de *design* ou, ainda, são direcionadas a visões arquiteturais.

Vale destacar que parte das abordagens comparadas possuem ferramentas *Case* disponíveis e geralmente utilizam ferramentas já reconhecidas no mercado. A ferramenta proposta neste trabalho cobre parcialmente os metamodelos criados uma vez que os relacionamentos não foram representados satisfatoriamente.

Apesar de a abordagem proposta ter utilizado os pacotes *CoreElements*, *Time* e NFP MARTE para modelagem de tempo-real, a representação de requisitos não-funcionais e a representação de tempo não foi coberta em completude. Isso se explica pelo fato de que mesmo sendo possível, na abordagem proposta, modelar restrições, *clocks*, definição de critérios de desempenho e segurança este trabalho ainda carece de maior nível de formalismo na modelagem de requisitos não-funcionais.

No entanto, como destacado por [Silvestre e Soares 2012a], "MARTE não têm sido empregado para modelagem de requisitos em alto nível de abstração". A abordagem proposta mostra a aplicabilidade do *profile* MARTE, com seus respectivos estereótipos e restrições, com o *profile* SysML para descrever requisitos em maior nível de detalhes e em diferentes níveis de abstração, classificá-los e rastreá-los com componentes do projeto.

Assim, os estereótipos MARTE deixam a noção de tempo mais clara que a forma apresentada com a UML. Na aplicação do metamodelo proposto, no Capítulo 4 e no Capítulo 5, é possível observar as contribuições deste trabalho para melhorar as visões para representação de requisitos. O metamodelo proposto é complementar aos demais métodos de modelagem de software e não os substitui.

Finalmente, é possível observar ao visualizar a Tabela 6.2 que a abordagem pro-

posta é pertinente no contexto da especificação de requisitos de sistemas de tempo-real e quando comparada às abordagens mais próximas representa um *framework* relevante e de propósito geral para modelagem de características inerentes ao domínio de sistemas de tempo-real.

6.2 Contribuições do Trabalho

Podem ser citadas algumas contribuições para a comunidade científica quanto à realização deste trabalho:

- Metamodelo que descreve critérios relevantes para elicitação, documentação, classificação, rastreamento e análise de requisitos em diferentes níveis de abstração.
- Metamodelo que engloba, pela primeira vez, segundo a revisão de literatura, os *profiles* SysML e MARTE da UML em um *framework* conjunto de modelagem de requisitos de sistemas de tempo-real, fornecendo, assim, um conjunto de informações sobre como representar diversas características de STR, como restrições, atrasos, desempenho, consumo de energia, confiabilidade, *clocks*, prazos ou uso de memória.
- O projeto em nível de requisitos de sistema usando SysML e MARTE, mostrando a complementaridade dos dois *profiles* UML e ao incentivar os *designers* de sistemas de tempo-real para desenvolver sistemas de forma eficiente através da utilização de abordagens dirigidas a modelos em estágios iniciais do projeto.
- Estensão da ferramenta de modelagem ArgoUML para representar a modelagem de um conjunto de requisitos de RTMS por meio dos metamodelos criados nesta pesquisa.

Esta pesquisa também resultou nos seguintes trabalhos em congressos científicos:

- " *A Metamodel for Tracing Requirements of Real-Time Systems*" [Ribeiro e Soares 2013c] aceito no *16th IEEE Computer Society Symposium on Object/Service-Oriented RealTime Distributed Computing* (ISORC 2013), evento classificado como *Qualis B1*.
- " *Application of an Extended SysML Requirements Diagram to Model Real-Time Control Systems*" aceito no *International Conference on Computational Science and Its Applications* (ICCSA 2013), evento classificado como *Qualis B1*.
- " *An Approach for Modeling Real-Time Requirements with SysML and MARTE Stereotypes*" [Ribeiro e Soares 2013b] aceito ao *15th International Conference on Enterprise Information Systems*, evento classificado como *Qualis B1*.

6.3 Trabalhos Futuros

No domínio dos sistemas de tempo-real e embarcados, pretende-se como trabalhos futuros:

- Estudar sistematicamente os demais pacotes MARTE para aplicá-los em uma metodologia completa que aborde não apenas os processos de especificação de requisitos de RTMS, mas todo o ciclo de vida dos Sistemas de Tempo-Real;
- Melhorar a ferramenta CASE estendida para suporte completo à modelagem;
- Aplicar o *profile* SysML e MARTE para descrição de requisitos mais específicos ou em outros estudos de caso;
- Formalizar a descrição de requisitos não-funcionais e propriedades temporais no modelo por meio da *Value Specification Language* (VLS);
- Realizar análises qualitativas da modelagem, com *stakeholders* da indústria de desenvolvimento de RTMS e mensurar possíveis dificuldades/vantagens de seu uso.

Referências Bibliográficas

[Administration 1996] Administration, F. H. (1996). Traffic Control Systems Handbook. Technical report, Departament of Transportation, Washington, DC, USA.

[Adrial 2007] Adrial, J. R. (2007). Formal Methods: Theory Becoming Practice. *Journal of Universal Computer Science*, 13(5):619 – 628.

[Albinet et al. 2007] Albinet, A., Boulanger, J., Dubois, H., Peraldi-frati, M., Sorel, Y., e Van, Q. (2007). Model-based Methodology for Requirements Traceability in Embedded Systems.

[Apshvalka et al. 2009] Apshvalka, D., Donina, D., e Kirikova, M. (2009). Understanding the Problems of Requirements Elicitation Process: A Human Perspective. In *Springer*, pp. 211 – 223.

[Argo] Argo. http://argouml.tigris.org/.

[Arif et al. 2009] Arif, S. U., Khan, Q., e Gahyyur, S. A. K. (2009). Requirements Engineering Processes, Tools/Technologies, & Methodologies. *International Journal of Reviews in Computing*, pp. 41– 56.

[Arpinen et al. 2012] Arpinen, T., Salminen, E., Hamalainen, T. D., e Hannikainen, M. (2012). MARTE Profile Extension for Mmodeling Dynamic Power Management of Embedded Systems. *Journal of Systems Architecture*, 58(5):209 – 219.

[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.

[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.

[Berry 2004] Berry, D. M. (2004). The Inevitable Pain of Software Development: Why There is No Silver Bullet. In *In RISSEF*, pp. 50 – 74. Springer.

[Bezerra 2006] Bezerra, B. (2006). *Princípios de Análise e Projeto de Sistemas com UML*. Campus/Elsevier.

[Bianco et al. 2002] Bianco, V. D., Lavazza, L., e Mauri, M. (2002). A Formalization of UML Statecharts for Real-Time Software Modeling. In *Integrate Design and Process Technoly (IDPT)*, pp. 1 – 8.

[Billington et al. 2010] Billington, D., Castro, V. E., Hexel, R., e Rock, A. (2010). Modelling Behaviour Requirements for Automatic Interpretation, Simulation and Deployment. In *Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472.

[Boehm 1973] Boehm, B. W. (1973). Seven Basic Principles of Software Engineering. *Journal of Systems and Software*, 3:3 – 24.

[Booch et al. 2007] Booch, G., Maksimchuk, R., Engle, M., Young, B., Conallen, J., e Houston, K. (2007). *Object-Oriented Analysis and Design with Applications*. Addison - Wesley Professional, Redwood City, CA, USA, 3th edition.

[Booch et al. 1999] Booch, G., Rumbaugh, J., e Jacobson, I. (1999). *The Unified Modeling Language Reference Mannual*. Addison - Wesley, Boston, MA, 1th edition.

[Briand et al. 2003] Briand, L. C., Labiche, Y., e O Sullivan, Y. (2003). Impact Analysis and Change Management of UML Models. *Proceedings of the International Conference on Software Maintenance (ICSM)*, pp. 256–265.

[Brooks 1987] Brooks, F. P. J. (1987). No Silver Bullet - Essence and Accident in Software Engineering. *Computer 20*, pp. 10–20.

[Burmester e Giese 2003] Burmester, S. e Giese, H. (2003). The Fujaba Real Time Statechart PlugIn. In *1th International Fujaba Days 2003*, p. 137.

[Burns e Wellings 2001] Burns, A. e Wellings, A. J. (2001). *Real-Time Systems and their Programming Languages: ADA 95, Real-Time Java, ad Real-Time POSIX*. Addison-Wesley Longman Publishing Co., Boston, MA, USA, 3th edition.

[Cabral e Sampaio 2008] Cabral, G. e Sampaio, A. (2008). Formal Specification Generation from Requirement Documents. *Electronic Notes in Theoretical Computer Science*, 195:171 – 188.

[Carrillo e Nicolas 2011] Carrillo, J. e Nicolas, J. (2011). Requirements Engineering Tools. *IEEE Software*, 28:86–91.

[Charles et al. 2007] Charles, A., Mallet, F., e Simone, R. (2007). Modeling Time(s). In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pp. 559–573.

[Cheng e Krishnakumar 1993] Cheng, K. T. e Krishnakumar, A. S. (1993). Automatic Functional Test Generation Using the Extended Finite State Machine Model. In *30th of the International Design Automation Conference (DAC)*, pp. 86 – 91.

[Chicote et al. 2007] Chicote, V., Moros, C., Moros, B., e Toval, A. (2007). REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. *Journal of Object Technology, Special Issue TOOLS EUROPE 2007*, 6(9).

[Chua et al. 2010] Chua, B. B., Bernardo, V., e Verner, J. M. (2010). Understanding the Use of Elicitation Approaches for Effective Requirements Gathering. In *Fifth International Conference on Software Engineering Advances (ICSEA)*, pp. 325 – 220.

[Ciaccia et al. 1997] Ciaccia, P., Ciancarini, P., e Penzo, W. (1997). Formal Requirements and Design Specifications: The Clepsydra Methodology. *International Journal of Software Engineering and Knowledge Engineering*, 7:1–42.

[Côté 2011] Côté, I., H. M. (2011). A UML Profile and Tool Support for Evolutionary Requirements Engineering. In *15th Software Maintenance and Reengineering (CSMR)*, pp. 161– 179.

[David et al. 2010] David, P., Idasiak, V., e Kratz, F. (2010). Reliability Study of Complex Physical Systems using SysML. *Reliability Engineering & System Safety*, 95:431 – 450.

[Davy 2009] Davy, D. Valecillos, C. (2009). Summary of a Literature Review of Qualitative Research in Technical Communication from 2003 to 2007. In *IEEE International Professional Communication Conference (IPCC)*, pp. 1 – 7.

[Demathieu et al. 2008] Demathieu, S., Thomas, F., André, C., Gérard, 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 (ISORC 2008)*, pp. 50–57. IEEE Computer Society.

[Douglass e Evangelist 2000] Douglass, B. P. e Evangelist, C. (2000). UML - The New Language for Real-Time and Embedded Systems. *Information Society Technologies*, pp. 1 – 5.

[Easterbrook et al. 2007] Easterbrook, S., Singer, J., Storey, M. A., e Damian, D. (2007). Selecting Empirical Methods for Software Engineering Research. *Guide to Advanced Empirical Software Engineering Springer*, 16:23–44.

[Ebeid et al. 2012] Ebeid, E., Fummi, F., Quaglia, D., e Stefanni, F. (2012). Refinement of UML/MARTE Models for the Design of Networked Embedded Systems. In *Europe Conference & Exhibition Design, Automation & Test*, pp. 1072 – 1077.

[Elkoutbi et al. 2002] Elkoutbi, M., Bennani, M., keller, R. K., e Boulmalf, M. (2002). Real-time System Specifications Based on UML Scenarios and Timed Petri Nets. In *IEEE Internationnal Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 362 – 367.

[Eshuis et al. 2002] Eshuis, R., Jansen, D. N., e Wieringa, R. (2002). Requirements-Level Semantics and Model Checking of Object-Oriented Statecharts. *Requirements engineering*, 7(4):243 – 263.

[Espinoza et al. 2009] Espinoza, H., Cancila, D., Selic, B., e Gerard, S. (2009). Challenges in Combining SysML and MARTE for Model - Based Design of Embedded Systems. In *5th European Conference (ECMDA-FA)*, pp. 98 – 113.

[Espinoza et al. 2008] Espinoza, H., Richter, K., e Gérard, S. (2008). Evaluating MARTE in an Industry-Driven Environment: TIMMOś Challenges for AUTOSAR Timing Modeling. In *Conf. on Design, Automation and Test in Europe (DATE), MARTE Workshop*, volume 07002.

[Everett e Honiden 1995] Everett, W. W. e Honiden, S. (1995). Reliability and Safety of Real-Time Systems. *IEEE Computer Society*, 3:13 – 16.

[Fidge et al. 1997] Fidge, C., Kearney, P., e Utting, M. (1997). A Formal Method for Building Concurrent Real-Time Software. *Software, IEEE*, 14:99–106.

[Firesmith 2004] Firesmith, D. (2004). Generating Complete, Unambigous, and Verifiable Requirements from Stories, Scenarios, and Use Cases. *Journal of Object Technology*, 3:27–39.

[Flynn e Hamlet 2006] Flynn, S. e Hamlet, D. (2006). On Formal Specification of Software Components and Systems. *Electronic Notes in Theoretical Computer Science*, 161:91–107.

[Gden 1994] Gden, K. W. (1994). Traffic Engineering Road Safety: A Practitioners Guide. Technical report, Departament of Civil Engineering Monash University, Monash University, Ausmlian.

[Ghezzi et al. 2003] Ghezzi, C., Jazayer, M., e Mandrioli, D. (2003). *Fundamentals of Software Engineering*. Prentice-Hall, Saddle River, NJ, USA.

[Giese 2003] Giese, H. andBurmester, S. (2003). Real - Time Statechart Semantics. Technical report tr-ri-03-239, University of Paderborn, Lehrstuhl fur Softwaretechnil, Germany.

[Glinz 2000] Glinz, M. (2000). Problems and Deficiencies of UML as a Requirements Specification Language. In *Tenth International Workshop on Software Specification and Design*, pp. 11–22.

[Gomaa 2001] Gomaa, H. (2001). Designing Concurrent, Distributed, and Real-Time Applications with UML. In *Proceedings of the 23rd International Conference on Software Engineering*, pp. 737 – 738, Toronto, Ontario, Canada, Washington, DC, USA. IEEE Computer Society.

[Gomez et al. 2012] Gomez, C., DeAntoni, J., e Mallet, F. (2012). Multi-view Power Modeling Based on UML, MARTE and SysML. In *EUROMICRO-SEAA*, pp. 17 – 20.

[Gordon 2003] Gordon, R. (2003). Systems Engineering Processes for Developing Traffic Signal Systems: A Synthesis of Highway Practice. Nchrp synthesis 307, Transportation Research Board, Washington, DC, USA.

[Hamie 1998] Hamie, A. (1998). Interpreting the Object Constraint Language. In *Software Engineering Conference*, pp. 288 – 295.

[Harel 1987] Harel, D. (1987). Statecharts: A visual Formalism for Complex Systems. In *Science of Computer Programming*, pp. 231– 274.

[Hazzan e Dubinsky 2007] Hazzan, O. e Dubinsky, Y. (2007). Qualitative Research in Software Engineering. *Frontier*, 4:6–17.

[Helming et al. 2010] Helming, J., Schneider, F., Haeger, M., Kaminski, C., Bruegge, B., e Berenbach, B. (2010). Towards a Unified Requirements Modeling Language. In *15th International Workshop on Requirements Engineering Visualization (REV)*, pp. 53–57.

[Hevner et al. 2004] Hevner, A. R., March, S. T., Park, J., e Ram, S. (2004). Design science in information systems research. *MIS Q.*, 28(1):75–105.

[IEEE 1990] IEEE (1990). *IEEE Standard Glossary of Software Engineering Terminology.*

[IEEE 1998] IEEE (1998). *IEEE Recommended Practice for Software Requirements Specifications.*

[Iqbal et al. 2012] Iqbal, M. Z. Z., Ali, S., e Yue, T. ad Briand, L. C. (2012). Experiences of Applying UML/MARTE on Three Industrial Projects.

[ITEA 2004] ITEA (2004). EAST-ADL: The EAST-EEA Architecture Description Language.

[Jacobson 2004] Jacobson, I. (2004). Use cases - Yesterday, Today, and Tomorrow. *Software and System Modeling*, 43(6):210–220.

[Jo et al. 2009] Jo, H. J., Hwang, J. G., e Yoon, Y. K. (2009). Formal Requirements Specification in Safety-Critical Railway Signaling System. In *Transmission & Distribution Conference & Exposition: Asia and Pacific*, pp. 1 – 4.

[Jokela e Lindberg 1990] Jokela, T. e Lindberg, K. (1990). Statecharts Based Requirements Analysis: Deriving User Oriented Models. In *Hardware and Software in System Engineering (EuroMicro)*, pp. 289 – 296.

[Josefil 2013] Josefil, C. (2013). Open Source Tool for Graphical UML 2 Modeling. www.ensieta.fr/mda/JOSEFIL.

[Kang e Ko 1996] Kang, K. C. e Ko, K. I. (1996). Formalization and Verification of Safety Properties of Statechart Specifications. In *Software Engineering Conference*, pp. 16 – 27.

[Kausar et al. 2010] Kausar, S., Tarif, S., Riaz, S., e Khanum, A. (2010). Guidelines for the selection of Elicitation Techniques. In *6th International Conference on Emerging Technologies (ICET)*, pp. 265 – 269.

[Kavi e Yang 1992] Kavi, M. K. e Yang, S. M. (1992). Real-Time Systems Design Methodologies: An Introduction and a Survey. *Journal of Systems and Software*, 18(1):85–99.

[Kirner e Davis 1996] Kirner, T. G. e Davis, A. M. (1996). Requirements Specification of Real-Time Systems: Temporal Parameters and Timing-Constraints. *Information & Software Technology*, 38(12):735 – 741.

[Klein 2006] Klein, L. A. (2006). *Traffic Detector Handbook.* Department of Transportation - Federal Highway Administration, USA, 3 edition.

[Kotonya e Sommerville 1998] Kotonya, G. e Sommerville, I., R. K. (1998). *Requirements Engineering - Processes and Techniques.* Wiley.

[Kumar e Jasperneite 2010] Kumar, B. e Jasperneite, J. (2010). UML Profiles for Modeling Real-Time Communication Protocols. *Journal of Object Technology*, 9:178–198.

[Laplante 2006] Laplante, P. (2006). *Real-time Systems Design & Analysis.* Wiley India Pvt. Limited, 3th edition.

[Lee 2002] Lee, D. T. (2002). Evaluating Real-Time Software Specification Languages. *Computer Standards & Interfaces*, 24(5):395–409.

[Li et al. 2011] Li, L., Ma, L., Wang, N., e Yang, Q. (2011). Modeling Method of Military Aircraft Support Process based SysML. In *International Conference in Reliability, Maintainability and Safety (ICRMS)*, pp. 1247 – 1251.

[Lin et al. 2008] Lin, L., Carter, J., e Poore, J. (2008). Using State Machines to Model and Manage Requirements Changes and Specification Changes. In *51th Midwest Symposium on Circuits and Systems (MWCAS)*, pp. 523 – 526.

[Linhares et al. 2007] Linhares, R. S., Oliveira, R. S., Farines, J. M., e Vernadat, F. (2007). Introducing the Modeling and Verification Process in SysML. In *Conference on Emerging Technologies and Factory Automation*, pp. 344 – 351.

[Liu et al. 2009] Liu, H., Jiang, Z., e Fung, R. Y. K. (2009). Performance Modeling, Real-Time dispatching and Simulation of Wafer Fabrication Systems Using Timed Extended Object-Oriented Petri Nets. *Computers and Industrial Engineering*, 56(1):121 – 137.

[Liu 2000] Liu, J. W. S. W. (2000). *Real-Time Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[Martin 2002] Martin, G. (2002). UML for Embedded Systems Specification and Design: Motivation and Overview. In *Design, Automation and Test in Europe Conference and Exhibition*, pp. 773–775.

[McKay e Marshall 2008] McKay, J. e Marshall, P. (2008). Foundations of Design Science in Information Systems. In *19th Australasian Conference on Information Systems (ACIS)*.

[Moura e Guedes 2012] Moura, R. S. e Guedes, L. A. (2012). Using Basic Statechart to Program Industrial Controllers. *Computer Standards & Interfaces*, 34(1):60 – 67.

[Mura et al. 2008] Mura, M., Murillo, L. G., e Prevostini, M. (2008). Model-Based Design Space Exploration for RTES with SysML and MARTE. In *Forum on Specification, Verification and Design Languages (FDL)*, pp. 203 – 208.

[Murata 1989] Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541 – 580.

[Naisbitt 1988] Naisbitt, J. (1988). Macrotendências. Dez Novas Orientações que Transformam as Nossas Vidas. *Lisboa: Editorial Presença*, p. 279.

[Nuseibeh e Easterbrook 1989] Nuseibeh, B. e Easterbrook, S. (1989). Distinctions between Requirements Specification and Design of Real Time Systems. In *Second International Conference on Software Engineering for Real Time Systems*, pp. 26 – 30.

[Nuseibeh e Easterbrook 2000] Nuseibeh, B. e Easterbrook, S. (2000). Requirements Engineering: A Roadmap. In *Conference on The Future of Software Engineering*, pp. 35 – 46.

[OMG 2011a] OMG, M. (2011a). *Modeling and Analysis of Real-Time and Embedded Systems (MARTE)- version 1.1*. Technical Report Formal/2011-06-02.

[OMG 2008] OMG, Q. (2008). *UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms - version 1.1*. versão 1.1.

[OMG 2005] OMG, S. (2005). *UML Profile for Schedulability, Performance and Time (SPT) - version 1.1.* Technical Report Formal/2005-07-05.

[OMG 2010] OMG, S. (2010). *Systems Modeling Language (SysML) Specification - version 1.1.*

[OMG 2011b] OMG, U. (2011b). *Linguagem de Modelagem Unificada - version 2.3.* versão 2.3.

[Parviainen et al. 2005] Parviainen, P., Tihinen, M., e van Solingen, R. (2005). Requirements Engineering: Dealing with the Complexity of Sociotechnical Systems Development. *Idea Group Inc.*

[Pettersson 1999] Pettersson, P. (1999). *Modelling and Verification of Real-Time Systems Using Timed Automata: Theory and Practice.* PhD thesis, Department of Computer Systems Uppsala University, Uppsala, Suécia.

[Quadri 2008] Quadri, I. R. (2008). MARTE Based Modeling Approach for Partial Dynamic Reconfigurable FPGAs. In *Workshop on Embedded Systems for Real-Time Multimedia*, pp. 47 – 52.

[Quadri et al. 2012a] Quadri, I. R., Brosse, E., Gray, I., Matragkas, N. D., Indrusiak, L. S., Rossi, M., e Bagnato, A. Sadovykh, A. (2012a). MADES FP7 EU project: Effective High Level SysML/MARTE Methodology for Real-Time and Embedded Avionics Systems. In *7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp. 1 – 8.

[Quadri et al. 2012b] Quadri, I. R., Soares, L., Gray, I., Indrusiak, L. S., e Bagnato, A. Sadovykh, A. (2012b). MADES: A SysML/MARTE High Level Methodology for Real-Time and Embedded Systems. In *2th Embedded Realtime Software and Systems*, pp. 1 – 10.

[Ramingwong 2009] Ramingwong, L. (2009). A Review of Requirements Enginnering Processes, Problems and Models. *International Journal of Engineering Science and Technology (IJEST)*, 4:2977 – 3002.

[Rational Software 2010] Rational Software, U.-R. (2010). *Rational UML Profile for Real-Time Systems.*

[Ribeiro e Soares 2013a] Ribeiro, F. G. C. e Soares, M. S. (2013a). Application of an Extended SysML Requirements Diagram to Model Real-Time Control Systems. In *International Conference on Computational Science and Its Applications (ICCSA).*

[Ribeiro e Soares 2013b] Ribeiro, F. G. C. e Soares, M. S. (2013b). An Approach for Modeling Real-Time Requirements with SysML and MARTE Stereotypes. In *15th International Conference on Enterprise Information Systems (ICEIS).*

[Ribeiro e Soares 2013c] Ribeiro, F. G. C. e Soares, M. S. (2013c). A Metamodel for Tracing Requirements of Real-Time Systems. In *16th IEEE Computer Society Symposium on Object/Service-Oriented RealTime Distributed Computing (ISORC).*

[Robbins et al. 2005] Robbins, J. E., Hilbert, D. M., e Redmiles, D. F. (2005). Theories, Methods and Tools in Program Comprehension: Past, Present and Future. In *13th International Workshop on Program Comprehension (IWPC)*, pp. 181 – 191.

[Roess et al. 2004] Roess, R., Prassas, E., e McShane, W. (2004). *Traffic Engineering*. Pearson/Prentice Hall, USA, 4 edition.

[Runeson e Host 2009] Runeson, P. e Host, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14:131–164.

[Saiedian e Dale 2000] Saiedian, H. e Dale, R. (2000). Requirements Engineering: Making the Connection between the Software Developer and Customer. *IEEE Software*, 42:419–428.

[Selic 2007] Selic, B. (2007). A Systematic Approach to Domain-Specific Language Design Using UML. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 07)*, pp. 2 – 9.

[Sengupta e Bhattacharya 2006] Sengupta, S. e Bhattacharya, S. (2006). Formalization of UML Use Case Diagram-A Z Notation Based Approach. In *International Conference on Computing & Informatics (ICOCI)*, pp. 1–6.

[Shaw 2003] Shaw, M. (2003). Writing Good Software Engineering Research Papers. In *25th International Conference on Software Engineering*, pp. 726 – 736.

[Silvestre e Soares 2011] Silvestre, E. A. e Soares, M. S. (2011). Multiple View Architecture Model for Distributed Real-Time Systems Using MARTE. In *20th International Conference on Information Systems Development (ISD 2011)*, pp. 98 – 113.

[Silvestre e Soares 2012a] Silvestre, E. A. e Soares, M. S. (2012a). *Modelagem de Software de Tempo Real utilizando o Profile MARTE da UML*. PhD thesis, Universidade Federal de Uberlandia, Uberlandia, Br.

[Silvestre e Soares 2012b] Silvestre, E. A. e Soares, M. S. (2012b). Modeling Road Traffic Signals Control using UML and the MARTE Profile. In *12th Computational Science and Its Applications(ICCSA 2012)*, pp. 1 – 15.

[Soares 2010] Soares, M. d. S. (2010). *Architecture-Driven Integration of Modeling Languages for the Design of Software-Intensive Systems*. PhD thesis, Deltf University of Technology, Delft, Holanda.

[Soares 2011] Soares, M. S. (2011). A Framework for Multi-Layered Requirements Documentation and Analysis. In *Computer Software and Applications Conference (COMPSAC)*, pp. 308 – 313.

[Soares e Vrancken 2007] Soares, M. S. e Vrancken, J. (2007). Requirements Specification and Modeling through SysML. In *International Conference on Systems, Man and Cybernetics*, pp. 1735–1740.

[Soares e Vrancken 2008] Soares, M. S. e Vrancken, J. (2008). Model-Driven User Requirements Specification Using SysML. *Journal Of Software*, 3:57 – 69.

[Soares et al. 2011] Soares, M. S., Vrancken, J., e Verbraeck, A. (2011). User Requirements Modeling and Analysis of Software-Intensive Systems. *The Journal of Systems and Software*, 84:328–339.

[Soares e Vrancken 2012] Soares, M. S. e Vrancken, J. L. M. (2012). A Modular Petri Net for Modeling and Scenario Analysis of a Network of Road Traffic Signals. *Control Engineering Practice*, 20(11):1183 – 1194.

[Somé 2006] Somé, S. S. (2006). Supporting use Case based Requirements Engineering. *Information and Software Technology*, 48:43–58.

[Sommerville 2011] Sommerville, I. (2011). *Engenharia de Software*. Pearson, São Paulo, 8th edition.

[Spivey 1998] Spivey, J. M. (1998). *The Z Notation: a Reference Manual*. Prentice-Hall, Oriel College, Oxford, 2th edition.

[Sussman 2005] Sussman, J. (2005). *Perspectives on Intelligent Transportation Systems (ITS)*. Springer.

[Tchako et al. 1994] Tchako, J. F. N., Beldjilali, B., Trentesaux, D., e Tahon, C. (1994). Modelling with Coloured Timed Petri Nets and Simulation of a Dynamic and Distributed Management System for a Manufacturing Cel. *International Journal of Computer Integrated Manufacturing*, 7(6):323 – 339.

[Thramboulidis 2004] Thramboulidis, K. (2004). Using UML in Control and Automation: A Model Driven Approach. In *International Conference on Industrial Informatics*, pp. 587 – 593.

[Valles 2010] Valles, F. B. (2010). A Formal Model for the Requirements Diagrams of SysML. In *IEEE Latin America Transactions*, pp. 259 – 269.

[Voget 2010] Voget, S. (2010). AUTOSAR and the Automotive Tool Chain. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 259 – 262, Dresden, Germany. European Design and Automation Association.

[Von et al. 2002] Von, B. M., Braun, P., e Schroder, C. (2002). Model Based Requirements Engineering for Embedded Software. In *10th IEEE International Requirements Engineering Conference (RE'02)*, p. 92.

[Weilkiens 2008] Weilkiens, T. (2008). *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. Morgan Kaufmann.

[Welch et al. 1998] Welch, L., Ravindran, B., Shirazi, B., e Bruggeman, C. (1998). Specification and Modeling of Dynamic, Distributed Real-Time Systems. In *19th IEEE Real-Time Systems Symposium*, pp. 72–81.

[Woodside e Wilson 2003] Woodside, A. G. e Wilson, E. J. (2003). Case Study Research Methods for Theory Building. *Journal of Business & Industrial Marketing*, 18:493 – 508.

[Xu et al. 2011] Xu, J., Li, T., Xie, Z., e Gao, T. (2011). Use Cases and Feedback in Functional Requirements Analysis. In *Information Technology, Computer Engineering and Management Sciences (ICM)*, volume 2, pp. 54–57.

[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.

[Yan e Tang 2008] Yan, F. e Tang, T. (2008). A Formal Modeling and Verification Approach for Real-Time System. In *World Congress on Intelligent Control and Automation (WCICA)*, pp. 204 – 208.

[Yin 2008] Yin, R. K. (2008). *Case Study Research: Design and Methods (Applied Social Research Methods)*. Sage Publications, 3th edition.

[Zaki e Jawawi 2011] Zaki, M. Z. M. e Jawawi, D. N. A. (2011). Model-Based Methodology for Implementing MARTE in Embedded Real-Time Software. In *IEEE Symposium on Computers & Informatics (ISCI)*, pp. 536 – 541.

[Zhu et al. 2012] Zhu, M. X., Luo, X. X., Chen, X. H., e Wu, D. D. (2012). A Non-Functional Requirements Tradeoff Model in Trustworthy Software Original Research Article. *Information Sciences*, 191:61–75.