MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2018/01.08.19.24-TDI

UM MÉTODO DE MODELAGEM DESCRITIVA DE SISTEMAS DE ENGENHARIA PARA POSSIBILITAR A GERAÇÃO AUTOMÁTICA DE REQUISITOS TEXTUAIS APLICADO A UM SATÉLITE

Wagner Schalch Mendes

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Geilson Loureiro, aprovada em 01 de março de 2018.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3QBJFHL>

INPE
São José dos Campos
2018

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2018/01.08.19.24-TDI

UM MÉTODO DE MODELAGEM DESCRITIVA DE SISTEMAS DE ENGENHARIA PARA POSSIBILITAR A GERAÇÃO AUTOMÁTICA DE REQUISITOS TEXTUAIS APLICADO A UM SATÉLITE

Wagner Schalch Mendes

Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Geilson Loureiro, aprovada em 01 de março de 2018.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3QBJFHL>

INPE
São José dos Campos
2018

Dados Internacionais de Catalogação na Publicação (CIP)

Mendes, Wagner Schalch.

M522m Um método de modelagem descritiva de sistemas de engenharia para possibilitar a geração automática de requisitos textuais aplicado a um satélite / Wagner Schalch Mendes. – São José dos Campos : INPE, 2018.

xxviii + 271 p. ; (sid.inpe.br/mtc-m21b/2018/01.08.19.24-TDI)

Tese (Doutorado em Engenharia e Tecnologia Espaciais/Engenharia e Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2018.

Orientador : Dr. Geilson Loureiro.

1. Automática. 2. Descritivo. 3. Engenharia. 4. Sistema. 5. SysML. I.Título.

CDU 629.7.01

Aluno (a): *Wagner Schaich Mendes*

Título: "UM MÉTODO DE MODELAGEM DESCRITIVA DE SISTEMAS DE ENGENHARIA PARA POSSIBILITAR A GERAÇÃO AUTOMÁTICA DE REQUISITOS TEXTUAIS APLICADO A UM SATÉLITE".

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Doutor(a)* em

Engenharia e Tecnologia Espaciais/Eng. Gerenc. de Sistemas Espaciais

Dr. Walter Abrahão dos Santos

Presidente / INPE / São José dos Campos - SP

Dr. Geilson Loureiro

Orientador(a) / INPE / São José dos Campos - SP

Dra. Ana Maria Ambrosio

Membro da Banca / INPE / São José dos Campos - SP

Dr. Álvaro Augusto Neto

Convidado(a) / ITA / São José dos Campos - SP

Dr. Luis Eduardo Vergueiro Loures da Costa

Convidado(a) / ITA / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 01 de março de 2018

“Faça o teu melhor, na condição que você tem, enquanto você não tem condições melhores, para fazer melhor ainda!”

Mario Sérgio Cortella

A minha família, minha esposa Fernanda, com sua incrível paciência e compreensão, e meus filhos Julia e Lucas, que ainda são muito pequenos para compreender a ausência do pai em alguns momentos de lazer. Que esta minha realização sirva como exemplo para mostrar que a educação é o único caminho para um futuro melhor.

AGRADECIMENTOS

Agradeço, primeiramente, ao meu orientador, o Prof. Dr. Geilson Loureiro, pelo apoio, compreensão e sabedoria, dedicados a mim e ao meu trabalho nestes longos cinco anos. Gostaria, também, de agradecer ao INPE pela oportunidade de voltar à escola e fazer uma das coisas que mais me dá prazer: aprender. Por último, mas não menos importante, quero agradecer aos meus gestores na Alstom Brasil, nominalmente, Ivan Queiroz, Aristóteles Kono, Leandro Aveiro e Suzana Capassi, que ao longo destes cinco anos de estudos me apoiaram e garantiram a minha continuidade nesta empreitada.

RESUMO

Práticas de descoberta de requisitos de sistemas ineficientes representam um problema generalizado na indústria. A utilização de linguagens naturais nas especificações de requisitos insere ambiguidades e problemas de interpretação. O uso de modelos descritivos para expressar requisitos vem sendo cada vez mais abordado na Engenharia de Sistemas. Entretanto, os métodos de modelagem existentes requerem um grande esforço e, mesmo após o modelo estar completo, os requisitos ainda precisarão ser escritos e sua consistência verificada. Esta tese provê ao engenheiro de sistemas um método de tradução de um modelo descritivo em SysML do sistema de interesse em requisitos textuais. Este método prescreve de maneira precisa os passos a serem seguidos pelo engenheiro de sistemas e os resultados esperados ao final de cada etapa. Mais ainda, esta tese descreve em detalhes o mecanismo utilizado na geração dos requisitos, a partir dos elementos e seus relacionamentos criados no modelo SysML do sistema de interesse. Um estudo de caso utilizando um sistema espacial já desenvolvido foi usado para praticar o método proposto e a conclusão resultante foi que os requisitos textuais gerados pelo mecanismo descrito possuem indicadores de qualidade, definidos no escopo deste trabalho, iguais ou superiores aos correspondentes indicadores de qualidade levantados para os requisitos originalmente escritos para o sistema escolhido.

Palavras-chave: Automática. Descritivo. Engenharia. Geração. Metodologia. Modelagem. Requisito. Sistema. SysML.

AN ENGINEERING SYSTEMS DESCRIPTIVE MODELING METHOD TOWARDS AUTOMATIC TEXTUAL REQUIREMENTS GENERATION APPLIED TO A SATELLITE

ABSTRACT

Inefficient requirements discovery practices represent a generalized problem in the industry. Using natural languages in the requirements specifications inserts ambiguities and interpretation problems. The usage of descriptive models to express requirements is lately becoming more and more common in Systems Engineering. However, existing modeling methods require great effort and, even after the model is complete, requirements need to be written and their consistency verified. This thesis provides to the systems engineer a method to translate a descriptive model in SysML of the system of interest into textual requirements. This method prescribes in a precise manner the steps to be taken by the systems engineer and the results expected at the end of each phase. Moreover, this thesis describes in details the mechanism used in the requirements generation, starting from the elements and their relationships created in the SysML model. A case study using an already developed spacial system was used to practice the proposed method and the resulting conclusion was that textual requirements generated by the described mechanism had quality indicators, defined in the scope of this work, equal or greater than the corresponding quality indicators raised for the requirements originally written for the selected system.

Keywords: Automatic. Descriptive. Engineering. Generation. Methodology. Modeling. Requirement. System. SysML.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AADL	*Architecture Analysis & Design Language*
ACT	*Activity Diagram*
AEC	*Association Entity Class*
AFCD	*Architecture Flow Context Diagram*
AFD	*Architecture Flow Diagram*
AICD	*Architecture Interconnect Context Diagram*
AID	*Architecture Interconnect Diagram*
AIS	*Architecture Interconnect Specification*
AIT	*Assembly, Integration and Test*
AMCD	*Architecture Message Context Diagram*
AMD	*Architecture Message Diagram*
AMS	*Architecture Module Specification*
ARP	*Aerospace Recommended Practice*
BDD	*Block Definition Diagram*
BPMN	*Business Process Model and Notation*
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CASE	*Computer-Aided Software Engineering*
CCD	*Control Context Diagram*
CFD	*Control Flow Diagram*
CSPEC	*Control Specification*
CRAM	Clube de Radioamadores de Americana
DCD	*Data Context Diagram*
DFD	*Data Flow Diagram*
DoD	*Department of Defense*
EBNF	*Extended Backus-Naur Form*
ECD	*Entity Class Diagram*
ECSS	*European Cooperation for Space Standardization*
EFFBD	*Enhanced FFBD*
EPS	*Electric Power Subsystem*
FFBD	*Function Flow Block Diagram*
HHP	Hatley-Hruschka-Pirbhai

IBD	*Internal Block Diagram*
ICAM	*Integrated Computer Aided Manufacturing*
IDEF	*ICAM Definition*
IDEF0	*IDEF for Function Modeling*
IEEE	*Institute of Electrical and Electronics Engineers*
INCOSE	*International Council on Systems Engineering*
INPE	Instituto Nacional de Pesquisas Espaciais
IQR	Indicador de Qualidade de Requisito
ISO	*International Organization for Standardization*
ITA	Instituto Tecnológico de Aeronáutica
JPL	*Jet Propulsion Laboratory*
LF	Linguagem Formal
LN	Linguagem Natural
MBSE	*Model Based Systems Engineering*
MD5	*Message-Digest algorithm 5*
MDA	*Model Driven Architecture*
MID	*Module Inheritance Diagram*
MOF	*Meta-Object Facility*
MOFM2T	*MOF Model to Text Transformation Language*
NASA	*U.S. National Aeronautics and Space Administration*
OBDH	*On-Board Data Handler Subsystem*
OMG	*Object Management Group™ (www.omg.org)*
OOSEM	*Object-Oriented Systems Engineering Method*
OPD	*Object-Process Diagram*
OPL	*Object-Process Language*
OPM	*Object-Process Methodology*
PMI	*Project Management Institute*
PSARE	*Process for System Architecture and Requirements Engineering*
PSPEC	*Process Specification*
RF	Radiofrequência
RUP	*Rational Unified Process*
SA	*State Analysis*
SAE	*Society of Automotive Engineers*
SD	*Sequence Diagram*

SDL	*System Definition Language*
SE	*Systems Engineering*
STM	*Structural and Thermal Subsystem*
SysML	*System Modeling Language*
TTC	*Telemetry, Tracking and Command Subsystem*
UC	*Use Case Diagram*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Este documento consiste na Tese de Doutorado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais, área de concentração em Engenharia e Gerenciamento de Sistemas Espaciais do Instituto Nacional de Pesquisas Espaciais (INPE), cujo título é “Um Método de Modelagem Descritiva de Sistemas de Engenharia para Possibilitar a Geração Automática de Requisitos Textuais Aplicado e um Satélite”.

1.1 Motivação

Práticas de requisitos ineficientes representam um problema generalizado na indústria. Uma abordagem mais disciplinada para desenvolvimento e gerenciamento de requisitos se faz necessária para melhorar as taxas de sucesso de projetos. Um índice de 53% do investimento da indústria em projetos de desenvolvimentos técnicos apresentam custos superiores aos previstos ou projetos mal sucedidos (YOUNG, 2004).

A utilização de linguagens naturais (e.g.: Inglês, Português, etc) nas especificações de requisitos insere, inevitavelmente, ambiguidades e problemas de interpretação. Além disto, por ser um processo humano, tais especificações estão sujeitas a omissões nos requisitos que podem acarretar em retrabalhos, só descobertos em fases finais do desenvolvimento do sistema, aumentando o custo inerente de correção.

Abordagens de análise de sistemas baseadas em análise estruturada e a *Unified Modeling Language* (OMG, 2015) são, em essência, a expressão dos requisitos para software e, em geral, estão conectadas a ferramentas *Computer-Aided Software Engineering* (CASE) que geram o código-fonte ou a estrutura do código-fonte automaticamente. Entretanto, isto se aplica mais especificamente ao desenvolvimento de software.

Os métodos de modelagem descritiva de sistemas existentes requerem grande esforço e, mesmo ao fim destas atividades, os requisitos ainda precisarão ser escritos e sua consistência verificada. Em boa parte das indústrias de hoje, a elaboração de um documento de requisitos de sistema é

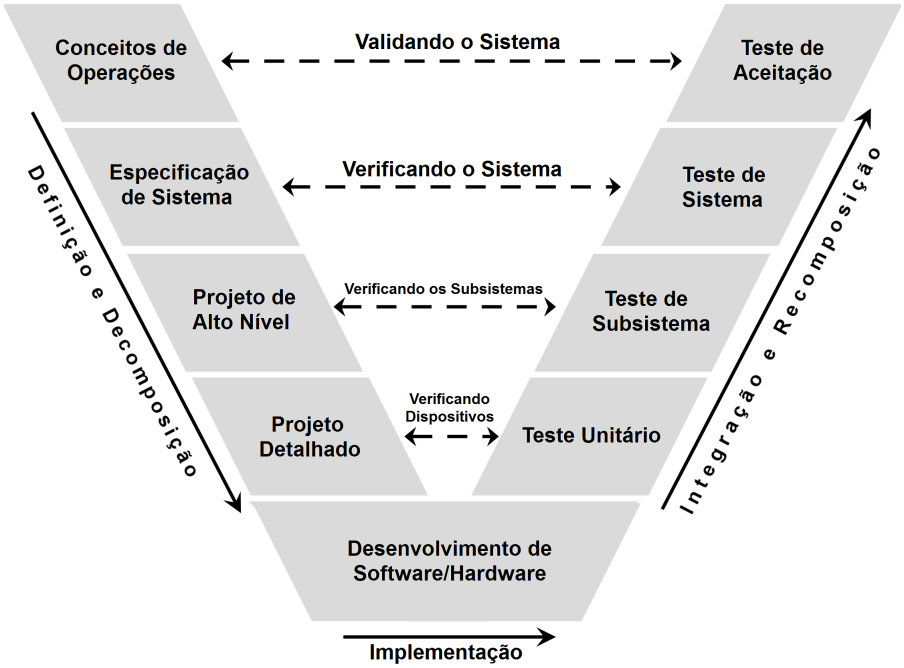
obrigatória e exigida por normas, ou da área de atuação, ou da própria empresa. São exemplos as áreas espacial, aeronáutica, ferroviária e automotiva, onde as normas adotadas ao redor do mundo exigem a elaboração de um documento contendo requisitos de sistema. A geração automática dos requisitos a partir do modelo descritivo do sistema de interesse economizaria as horas de engenharia utilizadas na elaboração de tais documentos.

Há métodos que fazem a conversão entre modelos descritivos e textos, abordados nos Capítulos 2 e 3, mas estes métodos ainda são incipientes e não se tornaram padrão no desenvolvimento de sistemas complexos.

1.2 Contextualização da Pesquisa

Segundo o *International Council on Systems Engineering* (INCOSE, 2015), vários modelos de ciclo de vida, tais como cascata, espiral, V e modelos de desenvolvimento ágil são úteis para se definir o início, o fim e as atividades apropriadas para os estágios do ciclo de vida de um sistema. Ainda segundo INCOSE (2015), o modelo V, visto na Figura 1.1 abaixo, é usado para focar na Engenharia de Sistemas, mais particularmente nos estágios de conceituação e desenvolvimento do sistema, ressaltando a necessidade de se criar planos de verificação durante o desenvolvimento dos requisitos. Segundo Shamieh (2011), especialistas no desenvolvimento de sistemas complexos vêm, a partir 1995, refinando este modelo V para melhor representar o processo de Engenharia de Sistemas.

Figura 1.1 – Modelo V da Engenharia de Sistemas.



Fonte: Adaptada de Shamieh (2011).

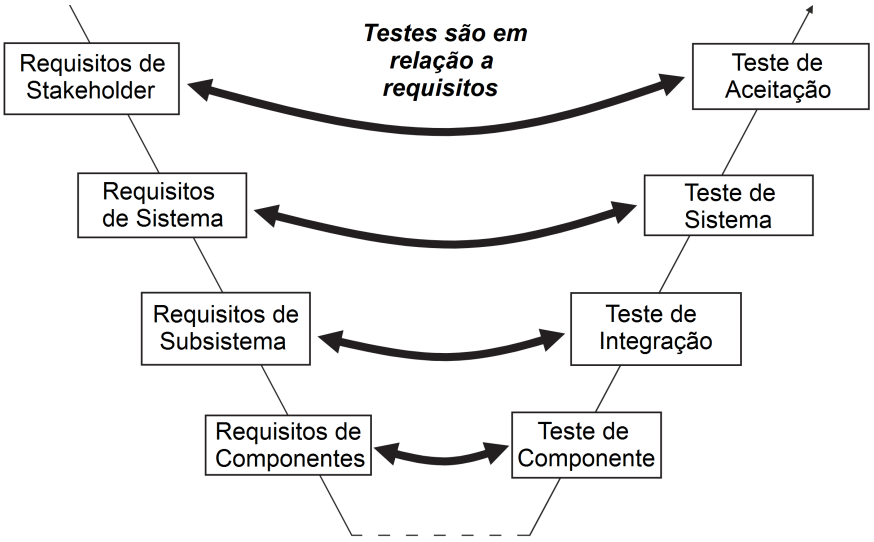
Na figura acima, da esquerda para a direita, de cima para baixo, têm-se os estágios do processo de Engenharia de Sistemas:

- Durante o estágio de Conceitos de Operação, definem-se as necessidades dos *stakeholders* e elicitam-se os correspondentes requisitos de *stakeholder* para representar as capacidades que o sistema deve prover e restrições que ele deve apresentar.
- Durante o estágio de Especificação do Sistema, definem-se os requisitos de sistema que atenderão aos requisitos de *stakeholder* acordados.
- Durante o estágio de Projeto de Alto Nível, define-se a arquitetura do sistema que incorpore os requisitos de sistema.
- Durante o estágio de Projeto Detalhado, define-se a arquitetura do sistema para o desenvolvimento de *software* e *hardware*.
- Durante o estágio de Desenvolvimento de *Software* e *Hardware*, acontecem os desenvolvimentos propriamente ditos, sejam eles desenvolvimentos internos à organização, externos ou até mesmo a compra de equipamentos e produtos de prateleira.

- Durante o estágio de Teste Unitário, aquilo que foi desenvolvido é verificado contra seus requisitos para garantir uma correta implementação de cada dispositivo individualmente.
- Durante o estágio de Teste de Subsistema, *software* e *hardware* são integrados e verificados contra os requisitos de Projeto de Alto Nível para garantir uma correta implementação das estruturas integradas (i.e. subsistemas).
- Durante o estágio de Teste de Sistema, os subsistemas são integrados e verificados contra os requisitos de sistema para garantir uma correta implementação do sistema com um todo.
- Durante o estágio de Teste de Aceitação, o sistema é validado contra os requisitos de *stakeholder* anteriormente elicitados.

Do ponto de vista da Engenharia de Requisitos, Hull et al. (2011) redesenhou o modelo V para representar no seu lado esquerdo os estágios do processo de derivação de requisitos. Esta releitura do modelo V é mostrada na Figura 1.2 abaixo.

Figura 1.2 – Modelo V da Engenharia de Requisitos.



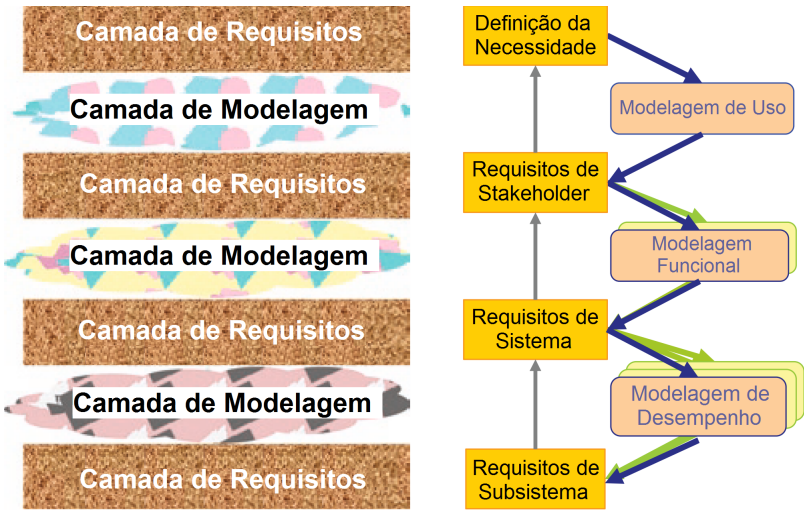
Fonte: Adaptada de Hull et al. (2011).

Na figura acima, no lado esquerdo deste modelo V têm-se os estágios do processo de derivação de requisitos. Começa-se pela elicitação dos requisitos de *stakeholder*, que serão utilizados posteriormente para a

aceitação do sistema. Destes, derivam-se os requisitos de sistema, que serão utilizados posteriormente para verificar se a implementação do sistema está correta. Então, derivam-se os requisitos de subsistema, que serão utilizados posteriormente para garantir que o sistema foi montado corretamente. Finalmente, derivam-se os requisitos de componente, que serão utilizados posteriormente para garantir que cada unidade do sistema foi desenvolvida corretamente. O lado direito deste modelo V é igual ao correspondente lado do modelo V apresentado na Figura 1.1.

Hull et al. (2011) vão além e discutem a utilização da modelagem na derivação de requisitos (i.e. lado esquerdo do modelo V da Figura 1.2), onde para se ir de uma camada de requisitos a outra (e.g.: de requisitos de *stakeholder* para requisitos de sistema), existe uma camada de modelagem que representa a atividade racional por trás da derivação. Na Figura 1.3 abaixo, mostra-se esta estrutura de camadas proposta pelos autores.

Figura 1.3 – Modelagem na derivação de requisitos.



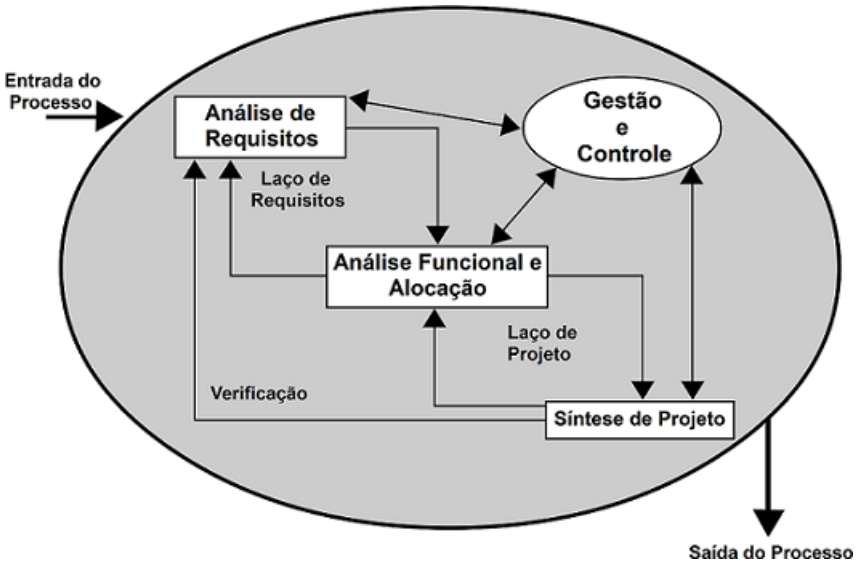
Fonte: Adaptada de Hull et al. (2011).

Do lado direito da figura acima, tem-se os estágios da derivação de requisitos, assim como apresentado na Figura 1.2. Tem-se, ainda, o tipo de modelagem utilizada de um estágio para o estágio subsequente, durante a derivação de requisitos: modelagem de uso para derivar requisitos de *stakeholder* a partir das necessidades identificadas, modelagem funcional para derivar requisitos de sistema a partir de requisitos de *stakeholder* e,

finalmente, modelagem de desempenho para derivar requisitos de subsistema a partir de requisitos de sistema. Do lado esquerdo da Figura 1.3, tem-se, simplesmente, uma representação das diferentes camadas que surgem no decorrer das atividades de derivação de requisitos.

Para se ir de um estágio para outro, tanto nos modelos V apresentados um na Figura 1.1 e outro na Figura 1.2, quanto nas atividades de modelagem para derivação de requisitos apresentadas na Figura 1.3, o *System Engineering Fundamentals* do *Department of Defense* (DoD, 2001) propõe um processo de alto nível de Engenharia de Sistemas, apresentado na Figura 1.4 abaixo.

Figura 1.4 – Processo de Alto-Nível de Engenharia de Sistemas.



Fonte: Adaptada de DoD (2001).

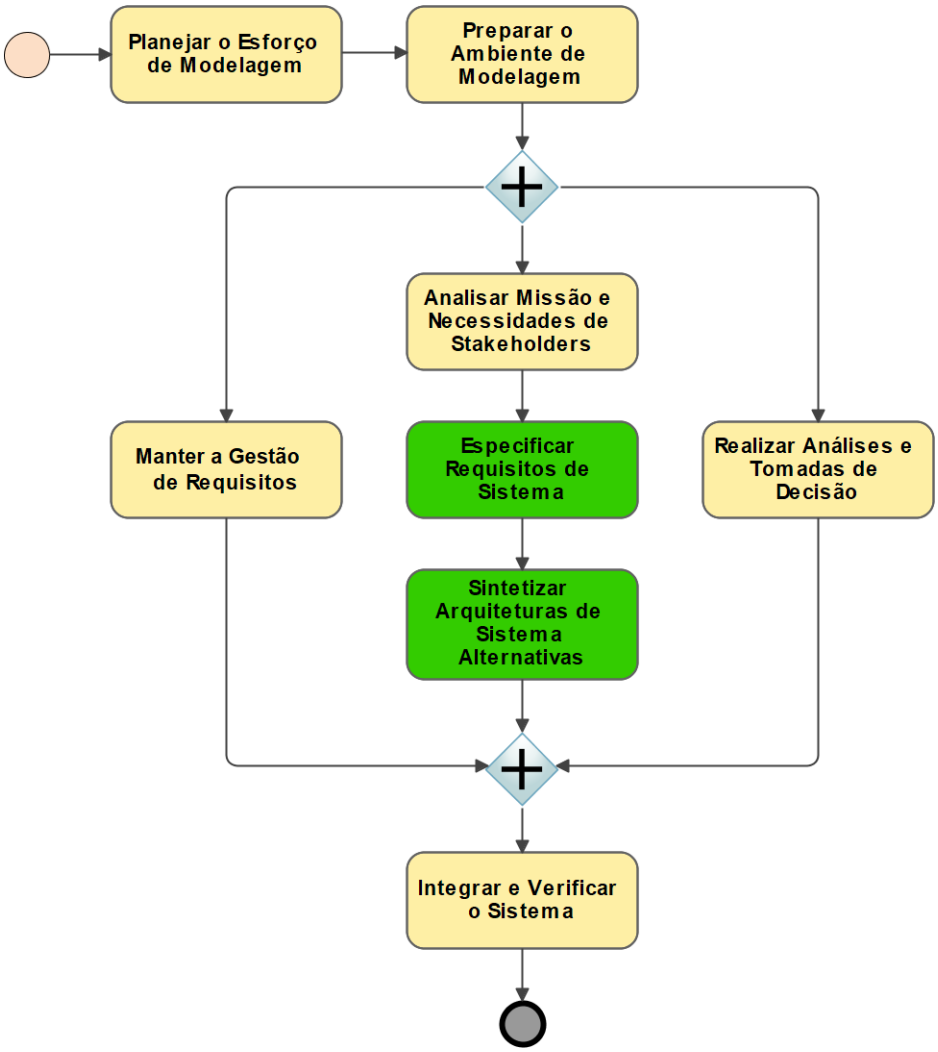
Da figura acima, tendo disponíveis as entradas para um determinado estágio do modelo V ou da derivação de requisitos, iniciam-se atividades interativas de análise de requisitos e análise funcional para se determinar as funcionalidades daquele estágio e, consequentemente, os requisitos que atenderão a estas entradas. Estes requisitos serão, por sua vez, refinados em requisitos mais detalhados até que sejam passíveis de alocação ao estágio subsequente, determinada através de atividades interativas de síntese de projeto e alocação funcional. Os resultados destas atividades, por sua vez, podem ser usados para realimentar o laço de análises funcional e

de requisitos, para adequação dos requisitos a mudanças de projeto. Outra realimentação destacada na Figura 1.4 é a de verificação, onde se garante que todas as entradas foram atendidas pela solução adotada. De maneira transversal a estas atividades, existe ainda, destacada na Figura 1.4, uma atividade de gestão e controle para propiciar a qualidade nas decisões, riscos, custos e prazos. Como saída deste processo, têm-se as entradas para o estágio subsequente no modelo V ou na derivação de requisitos.

No contexto desta pesquisa, que trata de um método de modelagem descritiva de sistemas para possibilitar a escrita automática de requisitos de sistema, foca-se nos estágios de Especificação de Sistema e Projeto de Alto Nível do modelo V apresentado anteriormente na Figura 1.1, correspondente aos estágios Requisitos de Sistema e Requisitos de Subsistema do modelo V apresentado na Figura 1.2, e que utiliza as modelagens funcional e de desempenho para derivação destes requisitos, assim como mostrado na Figura 1.3. Mais ainda, o método desenvolvido nesta pesquisa implementa as atividades de Análise Funcional e Alocação, e Síntese de Projeto do processo de alto nível de engenharia de sistemas mostrado na Figura 1.4.

Friedenthal e Oster (2017) propõem um processo para especificação e projeto de uma missão espacial, bem como do sistema espacial que realizará tal missão. As atividades deste processo preveem o uso dos princípios da MBSE, tendo a SysML como linguagem escolhida. Na Figura 1.5 abaixo, descreve-se este processo com auxílio da notação BPMN (OMG, 2013), notação esta que será utilizada no contexto desta Tese de Doutorado para descrever processos e seus respectivos subprocessos e atividades.

Figura 1.5 – Processo Proposto por Friedenthal e Oster (2017).



Fonte: Adaptada de Friedenthal e Oster (2017).

O processo proposto por Friedenthal e Oster (2017) visto na Figura 1.5 acima se ocupa desde a atividade de planejamento estratégico da modelagem do sistema de interesse até a atividade de integração e verificação do sistema. Além destas etapas, têm-se, ainda, as atividades de preparação do ambiente de modelagem, análise da missão espacial e necessidades dos *stakeholders*, especificação de requisitos de sistema e, finalmente, sintetização de alternativas para a arquitetura de sistema. Acontecendo paralelamente a estas atividades, têm-se as atividades de manutenção da gestão de requisitos e realização de análises para tomadas de decisão.

A metodologia proposta nesta Tese de Doutorado corresponde às atividades de especificação de requisitos de sistema e sintetização da arquitetura do sistema de interesse propostas por Friedenthal e Oster (2017), destacadas em verde na Figura 1.5 acima.

1.3 Objetivos, Premissas e Restrições

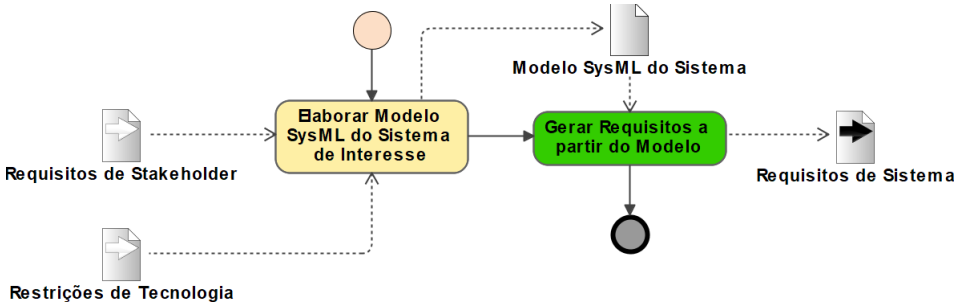
No contexto desta Tese de Doutorado, têm-se os seguintes objetivos geral e específicos:

Objetivo Geral. Tradução em requisitos textuais de um modelo descritivo de um sistema de engenharia, elaborado utilizando a SysML.

Objetivos Específicos. Apresentar uma metodologia de modelagem descritiva de sistemas de engenharia utilizando a SysML, aplicar esta metodologia em um sistema espacial já desenvolvido, comprar os requisitos textuais originais com aqueles traduzidos do modelo elaborado e, finalmente, propor uma discussão em torno da metodologia proposta e os resultados obtidos.

Para atingir estes objetivos, propõe-se, então, o processo de alto nível descrito na Figura 1.6 abaixo.

Figura 1.6 – Proposta para o Processo de Alto Nível.



Fonte: Produção do autor.

Pelo diagrama da figura acima, de posse dos requisitos de *stakeholder* e das restrições de tecnologia a serem levadas em conta, o engenheiro de sistema elabora o modelo SysML do sistema de interesse. Utiliza-se, então, um conjunto de regras que extrairão deste modelo os requisitos de sistema.

Para se determinar a relevância desta pesquisa, foi feito um levantamento estatístico em torno de alguns temas específicos à engenharia de sistemas. São eles: requisitos, mais especificamente, a escrita de bons requisitos; uso de modelos, mais especificamente, o uso de modelos para descrever sistemas complexos e altamente integrados; e, finalmente, a linguagem de modelagem de sistemas SysML. Tal estudo estatístico encontra-se detalhado no Capítulo 3.

A SysML foi escolhida como a linguagem de modelagem a ser utilizada por ser um padrão industrial (ISO, 2017), padrão este que consiste de uma linguagem visual de modelagem mantida pela OMG e usada para suportar a disciplina de engenharia de sistemas. Por não prescrever as atividades necessárias para elaboração do modelo de um sistema de interesse, além da linguagem, usou-se como base o método HHP (HATLEY et al., 2000). Este método implementa o processo chamado pelos próprios autores de *Process for System Architecture and Requirements Engineering* (PSARE), que visa a especificação de requisitos e sua posterior alocação aos elementos de arquitetura do sistema modelado, sempre com foco de se ter ao final um modelo completo e conciso, passível de ser entregue às subsequentes equipes do ciclo de vida do sistema em desenvolvimento. No Capítulo 6, apresentam-se discussões mais detalhadas sobre estas escolhas.

1.4 Estrutura

Esta Tese de Doutorado organiza-se da seguinte maneira. O Capítulo 1 fornece uma breve introdução, com a motivação, contextualização, objetivos, premissas e restrições da pesquisa. O Capítulo 2 contém a fundamentação teórica, provendo os principais conceitos e definições utilizados no trabalho e as principais linhas de pesquisa que permeiam o tema da tese. No Capítulo 3 apresentam-se um estudo estatístico para se determinar a relevância da pesquisa conduzida e uma revisão da literatura existente sobre temas semelhantes ao tema desta Tese de Doutorado. O Capítulo 4 desenvolve a metodologia para criação do modelo descritivo do sistema de interesse utilizando a SysML, bem como a extração automática dos requisitos de

sistema a partir deste modelo. O Capítulo 5 apresenta os resultados de um caso de estudo, onde a metodologia desenvolvida anteriormente no Capítulo 4 foi aplicada e os requisitos de sistema foram automaticamente extraídos do modelo. O Capítulo 6 fornece uma discussão sobre as diferentes escolhas e contribuições desta Tese de Doutorado e, finalmente, o Capítulo 7 contém a conclusão deste trabalho, reforçando o atendimento do objetivo da pesquisa e apresentando possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresentam-se os principais conceitos e definições utilizados no trabalho. Primeiramente, desenvolve-se uma discussão sobre requisitos, onde apresentam-se diferentes definições, destaca-se a importância de se escrever requisitos, elencam-se as principais características de um bom requisito e, finalmente, abordam-se métodos de como se deve escrever um requisito. Em seguida, apresenta-se a abordagem *Model Based Systems Engineering* (MBSE), associada a uma pesquisa sobre as diferentes metodologias existentes. Finalmente, apresentam-se os principais conceitos da System Modeling Language (OMG, 2017).

Após essas apresentações, passa-se, então, para uma discussão sobre as linhas de pesquisa que permeiam o trabalho desenvolvido. Em se tratando de um método para modelagem descritiva de um sistema que possibilite a geração automática de requisitos de sistema, deve-se cobrir duas linhas de pesquisa: metodologias de Engenharia de Sistemas baseadas em modelos descritivos e a conversão de modelos descritivos em texto.

A literatura apresenta distintas definições para o significado de ‘método’ e o significado de ‘metodologia’. Segundo Bloomberg e Schmelzer (2006, p. 174), metodologia é um conjunto de regras para alcançar um resultado desejado, por meio da aplicação de métodos para uma classe de problemas, ou seja, onde todos os problemas possuem algo em comum.

No contexto deste trabalho, tem-se:

Método ou Metodologia. Conjunto de regras que, ao serem aplicadas a um problema específico, produzem o resultado desejado.

Modelo descritivo. Uso de elementos definidos em uma linguagem de modelagem, relacionados entre si através de regras gramaticais nela estipuladas, criando, assim, estruturas semânticas que descrevem o sistema de interesse em termos de sua composição e comportamentos nos seus diversos cenários de utilização.

A interpretação destas estruturas semânticas e sua conversão em linguagem natural consiste na conversão de um modelo descritivo em texto.

2.1 Requisitos

Requisitos representam apenas um dos pilares do processo de Engenharia de Sistemas. Os requisitos elicitados a partir das necessidades dos *stakeholders* identificam cenários operacionais que, posteriormente, serão utilizados para a validação do projeto. Conforme o engenheiro de sistemas entende melhor o problema e particiona a solução a ser aplicada em componentes, requisitos de sistema são derivados dos requisitos de *stakeholders*. São estes requisitos de sistema que possibilitam o desenvolvimento da solução concebida e sua posterior verificação (BUEDE, 2009).

2.1.1 Definição de Requisito

Livros e normas contém, cada um da sua maneira, a definição de requisito mais adequada ao contexto abordado. Abaixo, apresentam-se algumas definições encontradas na literatura, que foram consideradas mais apropriadas para este trabalho. Em seguida, com base no que é comum das definições apresentadas, cria-se a definição de requisito utilizada neste trabalho.

European Cooperation for Space Standardization (ECSS, 2012) define requisito como sendo uma demanda documentada para ser cumprida.

U.S. National Aeronautics and Space Administration (NASA, 2007) define requisito com sendo uma necessidade, desejo, capacidade ou demanda acordada.

Aerospace Recommended Pratice (ARP, 2000) define requisito como sendo um elemento identificável de uma especificação de função que pode ser validada e cuja implementação pode ser verificada.

Institute of Electrical and Electronics Engineers (IEEE, 2005) define requisito como sendo uma sentença identificando uma característica operacional,

funcional ou de projeto, ou uma restrição de um produto ou processo, que é livre de ambiguidade, testável ou mensurável, e necessária para a aceitação do produto ou processo pelos clientes ou grupos internos de garantia da qualidade.

INCOSE (2015) define requisito como uma sentença identificando uma característica ou restrição de um sistema, produto ou processo, que é livre de ambiguidade, clara, única, consistente, concisa, verificável e necessária para a aceitação por parte dos *stakeholders*.

Alexander e Beus-Dukic (2009) definem requisito como uma condição verificável de um produto ou sistema, inicialmente falsa, que se deve tornar verdadeira com o desenvolvimento de um projeto.

Buede (2009) define requisito como uma sentença que restringe ou guia um projeto de um sistema de uma dada maneira que este sistema seja útil para um ou mais dos seus *stakeholders*.

Young (2004) define um requisito como sendo um atributo necessário em um sistema, uma sentença que identifica uma capacidade, característica, ou um fator de qualidade de um sistema para este ter valor ou utilidade para um cliente ou usuário.

Hatley et al. (2000) definem requisito como sendo ou uma capacidade requerida, combinada a um desempenho requerido ou uma restrição requerida.

De acordo com Schindel (2005), os requisitos de um sistema nada mais fazem do que descrever os relacionamentos entre as entradas e saídas deste sistema. Para o autor, esta ideia pode ser usada tanto para requisitos funcionais quanto para requisitos não-funcionais.

Já o *Project Management Institute* (PMI, 2013) define requisito como sendo uma condição ou capacidade, cuja presença em um produto, serviço ou resultado é exigida para satisfazer um contrato ou outra especificação formalmente imposta.

No contexto deste trabalho, tem-se:

Requisito. Capacidade, característica ou qualidade que o sistema de interesse deve apresentar, com o objetivo de satisfazer uma ou mais necessidades de um ou mais de seus *stakeholders*.

2.1.2 Importância dos Requisitos

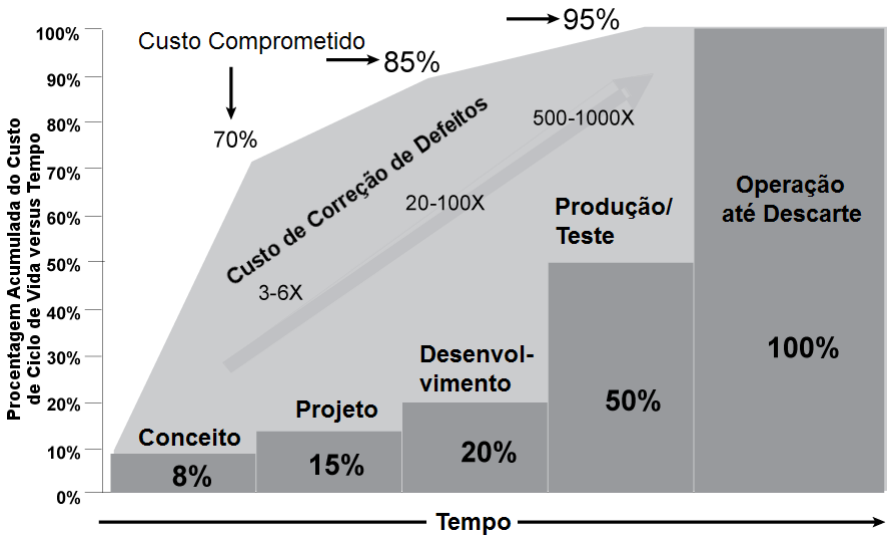
Requisitos fornecem a base para o esforço de desenvolvimento que está por vir. Somente após a especificação de requisitos estar consolidada, outras atividades técnicas, tais como projeto do sistema, desenvolvimento, testes e operação, podem efetivamente iniciar. Requisitos melhoram o entendimento do que os *stakeholders* realmente precisam, estabelecem uma sequência das entregas daquilo que está sendo construído, colaboram e estabelecem expectativas para o time de engenharia, ajudam no gerenciamento das expectativas dos *stakeholders*, melhoram a eficiência do processo de criação do sistema, justificam o desenvolvimento do sistema e estabelecem limites para o esforço de seu desenvolvimento (YOUNG, 2004).

Dados da NASA, a agência espacial norte-americana, evidenciados por Hooks e Farry (2001), mostram que seus projetos que utilizaram de 2% a 3% do orçamento total no processo de requisitos apresentaram um estouro de 80% a 200% no seu custo. Por outro lado, seus projetos que utilizaram de 8% a 14% do orçamento total do projeto no processo de requisitos apresentaram um estouro de 0% a 50% no seu custo. De acordo com o gráfico da Figura 2.1 abaixo, isto acontece pois a maior parte do custo do projeto é comprometida justamente na fase de concepção do sistema, onde os requisitos inciais são descobertos.

De acordo com Crosby (1979) qualidade significa, na sua essência, conformidade com requisitos e deve ser definida em termos mensuráveis e claramente estabelecidos para ajudar a organização a tomar ações baseadas em objetivos tangíveis, em vez de em palpites, experiência ou opiniões. Mais ainda, Deming (1982) aborda o tema da dificuldade de se atingir qualidade de um produto ou serviço por se tratar da tradução de necessidades futuras dos usuários em características mensuráveis (i.e. requisitos), para fazer com que o item projetado dê satisfação a um preço razoável. Já Juran e Gryna

(1988) definem qualidade como sendo a adequação ao uso, onde deve existir um equilíbrio entre as funcionalidades de um produto (i.e. requisitos de *stakeholder*) e em ser livre de deficiências. Nas três definições vistas acima, vê-se a correlação entre qualidade e requisitos corretamente estabelecidos.

Figura 2.1 – Custo do ciclo de vida versus tempo.



Fonte: Adaptada de Defense Acquisition University (1993, citado por INCOSE, 2015).

No gráfico da figura acima, tem-se o tempo no eixo horizontal e a porcentagem cumulativa do custo do ciclo de vida de um sistema no eixo vertical. Nota-se que, ao término da fase de concepção, apesar de um gasto de 8% do orçamento, já se comprometeu 70% deste. Ao término da fase de projeto, comprometeu-se 85% do orçamento com um gasto efetivo de 15%. Ao término do desenvolvimento, onde 90% do orçamento está comprometido, gastou-se 20% desta provisão. Ao término da fase de produção e testes, com um gasto de 50% do orçamento total, 100% desse valor está comprometido. A outra metade do orçamento é gasta nas demais fases do ciclo de vida do sistema, ou seja, desde a operação até o descarte.

Uma pesquisa conduzida pelo Standish Group mostrou as principais causas de falhas em projetos, resumidas na Tabela 2.1 abaixo.

Tabela 2.1 – Principais causas de falhas em projetos.

Causa de falhas em projetos	Porcentagem
Requisitos incompletos (*) (**)	13,1%
Falta de envolvimento do usuário (*)	12,4%
Falta de recursos	10,6%
Expectativas não realistas (*) (**)	9,9%
Falta de suporte da gerência	9,3%
Mudança nos requisitos e especificação (*)	8,7%
Falta de planejamento	8,1%
Requisito não mais necessário (*) (**)	7,5%

Fonte: Adaptada de Standish Group (1995, citado por Hull at al., 2011).

Aquelas causas que estão marcadas com (*) foram identificadas por Hull et al. (2011) como relacionadas ao processo de engenharia de requisitos, totalizando 51,6%, ou seja, um pouco mais da metade das causas identificadas pela pesquisa. Destacam-se, ainda, com (**) as causas diretamente relacionadas à escrita de requisitos.

Bons requisitos possuem características específicas e objetivas, amplamente abordadas na literatura, e são discutidas na seção seguinte.

2.1.3 Características de Bons Requisitos

Hooks (1993) define as seguintes características para um bom requisito: necessário, verificável, factível, conciso, simples e livre de ambiguidade. Já Buede (2009, p. 182) provê uma lista similar de características que devem ser observadas em um bom requisito: livre de ambiguidade, de fácil compreensão, correto, conciso, rastreado, rastreável, livre de implementação e verificável. Já Young (2004) estabelece critérios de como um bom requisito deve ser escrito. Nesta lista, além das características já apresentadas anteriormente, o autor acrescenta: completo, consistente, alocado, único, escrito usando uma construção padrão e atribuído a um identificador único.

Com uma abordagem oposta, Alexander e Stevens (2002) discutem sobre como requisitos não devem ser escritos e acabam, de maneira implícita, estabelecendo as mesmas características de um bom requisito já discutidas.

Halligan (2012), da mesma maneira, define tipos de defeitos que um requisito pode apresentar. Da forma como são colocados por este autor, estes defeitos também podem ser vistos como o oposto de características que um bom requisito deve apresentar, não destoando daquelas já vistas anteriormente.

Ryan e Wheatcraft (2017) estudaram um conjunto de trabalhos de outros autores com definições destas mesmas características para bons requisitos. Ao final, os autores propõem um conjunto de definições para estas características de bons requisitos. O que se percebe é que estes autores adicionaram um grau a mais de formalismo nas definições, mas que na essência continuam apontando para o mesmo norte das demais definições.

2.1.4 Como Escrever Requisitos

Alexander e Stevens (2002) discutem sobre a escrita de requisitos e defendem que não existe algo como o requisito perfeito. Tentar perseguir tal utopia só acarretará em gastos desnecessários de recursos. Deve-se, na verdade, buscar a qualidade do requisito, iniciando por um rascunho grosseiro que vai se transformando em um bom requisito ao longo do tempo, conforme as reais necessidades dos *stakeholders* são descobertas. Os autores definem, portanto, a anatomia de um requisito genérico, contendo um tipo de usuário (ou *stakeholder*), um verbo (ou ação), o objeto sob ação do verbo e uma frase adverbial qualificando o verbo, ou seja, dizendo quão bem a ação deve ser executada. São apresentadas, ainda, algumas boas práticas de como se escrever um requisito. Dentre elas, destacam-se duas: usar sentenças simples e diretas e um vocabulário limitado.

Hull et al. (2011) definem *boilerplates* (ou gabaritos) para requisitos como sendo frases prontas com lacunas para serem preenchidas com elementos dos requisitos, de forma a estabelecer um padrão para escrita de requisitos. Considerando requisitos de *stakeholders*, os autores definem, por exemplo, o seguinte gabarito: “O <tipo de *stakeholder*> deve ser capaz de <capacidade> com <desempenho> de <evento> enquanto <condição operacional>”. Considerando requisitos de sistema, os autores definem, por exemplo, os seguintes gabaritos: “O <sistema> deve <função> em não menos do que

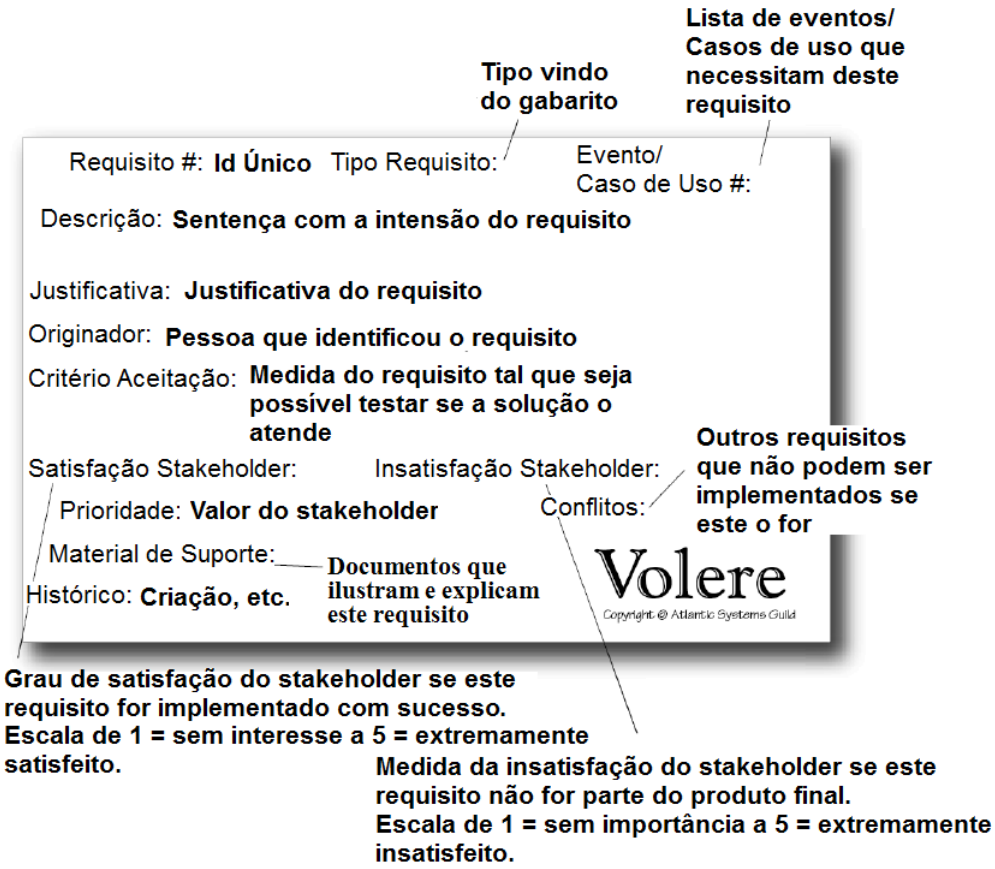
<quantidade> <objeto> enquanto <condição operacional>” e “O <sistema> deve <função> <objeto> todo <desempenho> <unidade>”. As lacunas estão identificadas com os caracteres ‘<’ e ‘>’ e contém uma dica de qual elemento do requisito deve preenchê-la. Não existe limite para se criar estes gabaritos e eles devem ser customizados para o contexto da organização ou projeto que os utilizarão (e.g.: gabaritos para requisitos funcionais, gabaritos para restrições de projeto, gabarito para restrições de desempenho, etc). Além da padronização na escrita de requisitos, estes gabaritos oferecem a possibilidade de automatizar sua geração, baseada nas definições dos gabaritos existentes e uma base de dados onde um requisito será composto por uma referência a um destes gabaritos e os elementos que preencherão as lacunas do gabarito referenciado. Isto torna possível, também, uma customização na geração dos requisitos, onde em certas ocasiões, algumas lacunas poderiam ser omitidas por questões de confidencialidade.

Halligan (1993) cria um modelo estrutural para requisitos, no qual os elementos de sintaxe da sentença são identificados. Esta estrutura contém os seguintes elementos: ator, condição, ação, objeto da ação, restrições da ação, refinamento ou origem do objeto e, finalmente, refinamento ou destino da ação. O ator representa o sujeito da sentença, a condição define quando a ação deve acontecer, a ação é representada por um verbo, as restrições da ação qualificam a ação, o objeto da ação é representado por um substantivo, o refinamento do objeto, a origem do objeto e o destino do objeto qualificam o objeto e, finalmente, o refinamento da ação também qualifica a ação. O autor estabelece, então, métricas de qualidade de um requisito baseadas na presença ou não destes elementos, levando em conta a possibilidade de que para um dado requisito, um dado elemento pode não ser aplicável.

Robertson e Robertson (2012) propõem o uso de um modelo para escrita de requisitos, comumente chamado de modelo Volere, que contém os atributos de um requisito que serão usados para a sua elicitação. Segundo os autores, cada atributo contribui para o entendimento e testabilidade do requisito. Na

Figura 2.2 abaixo, apresenta-se este modelo Volere em um possível formato para armazenar requisitos.

Figura 2.2 – Modelo Volere para escrita de requisitos.



Fonte: Adaptada de Robertson e Robertson (2012).

Na figura acima, tem-se a representação de um cartão, onde os atributos de um requisito são organizados. Da esquerda para direita, de cima para baixo, tem-se: identificador único do requisito, tipo do requisito (dependente da organização), eventos ou casos de uso relacionados, texto do requisito, justificativa da existência do requisito, originador do requisito, critério mensurável de aceitação, grau de satisfação dos *stakeholders* caso o requisito seja entregue, grau de insatisfação dos *stakeholders* caso o requisito não seja entregue, prioridade, conflitos com outros requisitos, materiais de suporte relacionados ao requisito (documentos, normas, etc) e histórico (criação, modificações, remoção, etc).

Carson (2015) identifica quatro tipos básicos de requisitos: funcional ou de desempenho, de projeto, ambiental e de adequabilidade. Requisitos

funcionais ou de desempenho definem os comportamentos funcionais do agente, o desempenho a eles associado, as condições em que isso ocorre, interfaces envolvidas e quaisquer entradas ou eventos que iniciam tais comportamentos. Requisitos de projeto limitam o espaço de soluções aceitáveis. Requisitos ambientais abordam condições naturais associadas aos comportamentos do agente. Finalmente, requisitos de adequabilidade abordam critérios de qualidade mais abrangentes, como por exemplo, restrições de segurança, disponibilidade, confiabilidade, etc.

Para cada um destes tipos de requisitos, Carson (2015) propõe o uso de um padrão de escrita contendo elementos que devem ser devidamente preenchidos quando o padrão for instanciado para um dado sistema. Abaixo, são apresentados estes padrões propostos pelo autor. Os elementos a serem preenchidos na instanciação do padrão estão em letras maiúsculas e negrito, e trechos entre colchetes são considerados opcionais.

– Requisitos Funcionais/Desempenho:

O **AGENTE** deve **FUNÇÃO** de acordo com **INTERFACE-SAÍDA** com **DESEMPENHO** [e **MOMENTO** dado **EVENTO-GATILHO** de acordo com **INTERFACE-ENTRADA**] enquanto na **CONDIÇÃO**.

– Requisitos de Projeto:

O **AGENTE** deve [exibir] **RESTRIÇÃO-PROJETO** [de acordo com **DESEMPENHO** enquanto na **CONDIÇÃO**].

– Requisitos Ambientais:

O **AGENTE** deve [exibir] **CARACTERÍSTICA** durante/depois de exposição ao **AMBIENTE** [por **DURAÇÃO**].

– Requisitos de Adequabilidade:

O **AGENTE** deve exibir **CARACTERÍSTICA** com **DESEMPENHO** enquanto na **CONDIÇÃO** [por **DURAÇÃO**].

Segundo o autor, o campo **AGENTE** representa a entidade que executa a função pretendida ou que apresenta a característica ou restrição requerida. O

campo **FUNÇÃO** representa o comportamento que o agente deve apresentar. O campo I**NTERFACE-SAÍDA** define o local onde a função deve ser observada e avaliada, com suas respectivas restrições. O campo **DESEMPENHO** representa um atributo mensurável da função ou de projeto que permita a determinação do sucesso ou falha durante atividades de verificação. O campo **CONDIÇÃO** descreve as circunstâncias nas quais o agente executa a ação ou função. O campo **EVENTO-GATILHO** define eventos observáveis na fronteira do agente e que iniciam funções. O campo **INTERFACE-ENTRADA** define o local onde o evento deve ser observado e avaliado, com suas respectivas restrições. O campo **MOMENTO** é uma declaração de quão logo ou por quanto tempo o comportamento deve ser observado, associado às condições de iniciação. O campo **RESTRIÇÃO-PROJETO** é uma afirmação de uma limitação do agente. O campo **CARACTERÍSTICA** descreve o agente enquanto aplicado a um ambiente. O campo **AMBIENTE** é uma descrição do ambiente ao qual o agente está exposto. Finalmente, o campo **DURAÇÃO** descreve uma medida temporal durante a qual uma condição ou um ambiente é aplicado ao agente.

Conclui-se, portanto, que requisitos são essenciais para o sucesso de um sistema, mas não basta escrever requisitos, deve-se escrever bons requisitos. Bons requisitos possuem características definidas e devem ser escritos de uma maneira estruturada e, preferencialmente, padronizada, contendo todos os elementos que garantam tais características.

2.2 A Disciplina da Engenharia de Sistemas

Com o aumento da complexidade e nível de integração dos sistemas, a abordagem de desenvolvimento baseada em componentes (onde cada parte do sistema é desenvolvida e otimizada independentemente da conceituação do todo) não estava mais alcançando os resultados desejados. Uma nova abordagem apareceu, então, com o objetivo de preencher esta lacuna: a Engenharia de Sistemas.

2.2.1 Definição de Engenharia de Sistemas

O INCOSE (2015) define Engenharia de Sistemas como sendo uma abordagem interdisciplinar que habilita a realização de sistemas de sucesso.

A ECSS (2009) define Engenharia de Sistemas como uma abordagem interdisciplinar governando todo o esforço técnico requerido para transformar um requisito em uma solução sistema.

A NASA (2007) define Engenharia de Sistemas como sendo um processo analítico pelo qual uma necessidade é transformada em um produto realizado e definitivo, capaz de suportar uma compatibilidade com todos os requisitos físicos e funcionais, e suportar os cenários operacionais em termos de confiabilidade, manutenibilidade, suportabilidade, e capacidade de serviços e descarte, enquanto mantendo desempenho e viabilidade.

O IEEE (2005) define Engenharia de Sistemas como o processo de derivar, evoluir e verificar uma solução balanceada ao longo do ciclo de vida do sistema com o objetivo de satisfazer as necessidades dos *stakeholders*.

Friedenthal et al. (2014) definem Engenharia de Sistemas como uma abordagem multidisciplinar para desenvolver soluções sistêmicas balanceadas no ciclo de vida, em resposta às diversas necessidades dos *stakeholders*.

Weilkiens (2007) define Engenharia de Sistemas como uma integração de todas as disciplinas em um processo de desenvolvimento estruturado, desde a concepção, passando pela produção e operação, chegando, finalmente, na retirada do sistema de operação para posterior descarte.

Holt e Perry (2013) definem Engenharia de Sistemas como uma abordagem multidisciplinar e de senso comum que habilita a realização de sistemas de sucesso.

Stevens et al. (1998) definem Engenharia de Sistemas como um processo de criar soluções eficazes e balanceadas para problemas, de um ponto de vista

holístico, e gerenciando a complexidade técnica dos desenvolvimentos resultantes.

Jenney (2010) define Engenharia de Sistemas como sendo o conjunto de quatro tarefas executadas dentro do escopo de desenvolvimento de produto: planejar o programa, definir o sistema, selecionar o projeto preferido e verificar o desempenho técnico.

No contexto deste trabalho, tem-se:

Engenharia de Sistemas. Abordagem multidisciplinar com o objetivo de conceber uma solução sistema balanceada em todo o ciclo de vida do sistema e que satisfaça as necessidades dos seus *stakeholders*.

2.2.2 As Abordagens para a Engenharia de Sistemas

Tradicionalmente, projetos de grande porte têm empregado a Engenharia de Sistemas com uma abordagem baseada em documentos (FRIEDENTHAL et al., 2014). Esta abordagem é caracterizada pela geração de especificações e documentos de projeto textuais para serem compartilhados entre clientes, usuários, desenvolvedores, montadores, integradores e testadores.

Requisitos de sistemas são expressos nestes documentos. A ênfase da Engenharia de Sistemas está em propiciar o controle desta documentação. A rastreabilidade dos requisitos consiste em mapeá-los para os diferentes níveis da hierarquia de especificações. Garantir a consistência de todo este conjunto de documentos torna-se extremamente difícil e custoso. Outra dificuldade é a informação espalhada em diversos documentos, tornando seu acesso e manutenção extremamente complicados (DELLIGATTI, 2014).

Para minimizar estas dificuldades, várias disciplinas da engenharia utilizam a modelagem como abordagem para projeto, garantindo assim, a consistência nas informações que servem como entrada para as atividades técnicas de produção. Na Engenharia Civil, plantas elaboradas durante a fase de arquitetura (e.g.: planta hidráulica, baixa, elétrica, estrutural, etc) nada mais são do que diferentes visões de um modelo do empreendimento que futuramente será construído. Na Engenharia Mecânica, modelos 2D e 3D e

de elementos finitos ajudam no dimensionamento e construção das estruturas mecânicas. Na Engenharia Elétrica, análises de circuitos são facilmente realizadas por meio de ferramentas computacionais onde cada componente eletrônico é representado por meio de seu modelo matemático e o funcionamento do circuito como um todo pode, então, ser simulado. Na Engenharia de Software, várias técnicas existentes propiciam a criação de modelos de software a serem desenvolvidos. Desde análise estruturada até métodos baseados em orientação a objetos com o uso, por exemplo, da UML, podem ser usados para se produzir informações necessárias a diferentes equipes de desenvolvimento de software.

2.2.3 Definição de Modelo

Segundo Hatley et al. (2000), um modelo é uma abstração que ressalta somente alguns aspectos de um sistema real para propiciar sua análise mais detalhada. Ainda segundo estes autores, um modelo, além de propiciar um melhor gerenciamento da complexidade do mundo real através da sua divisão em pedaços mais fáceis de serem compreendidos, possui um objetivo (i.e. questão a ser respondida) e um ponto de vista (i.e. ponto de vista de um ou mais *stakeholders*).

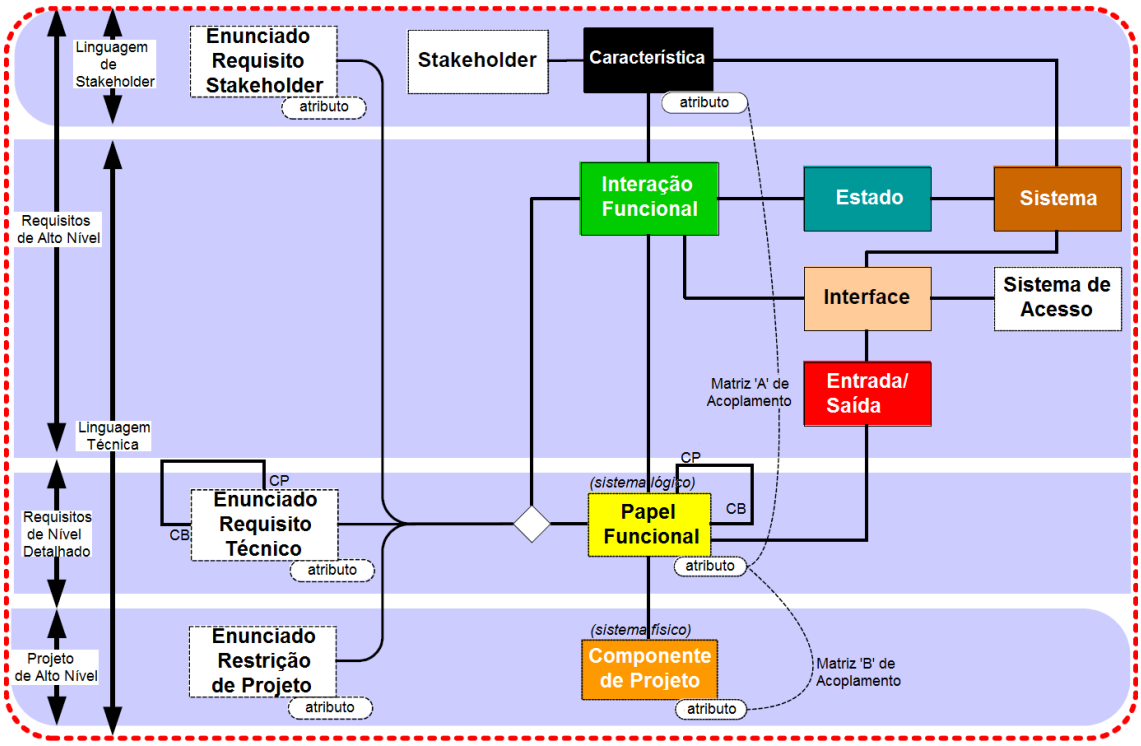
Friedenthal et al. (2014) define um modelo como a representação de um ou mais conceitos realizados no mundo físico. Segundo este autor, a principal característica de um modelo é uma abstração que não contém todos os detalhes da entidade modelada, mas somente aqueles relevantes para o seu propósito e de interesse para os *stakeholders* deste modelo. Por exemplo, um modelo térmico de um satélite tem como propósito o estudo do seu comportamento térmico em resposta às diferentes cargas térmicas às quais ele estará exposto ao longo do seu ciclo de vida (e.g.: lançamento, transição de órbitas, operação, etc). Este modelo, no entanto, não serve para análise de resistência e rigidez mecânica do satélite.

Em um contexto mais sistêmico, Dori (2002) define o modelo de um sistema como uma abstração com o propósito de melhor entender, comunicar, explicar ou projetar os aspectos de interesse deste sistema.

Segundo Holt e Perry (2013), modelos propiciam a minimização das três principais causas de falhas em projetos: complexidade, falta de entendimento e falha de comunicação.

Interligando as discussões sobre modelos de sistemas e gerenciamento da complexidade destes sistemas, Schindel (2011) trabalha com o foco de que o tamanho da menor representação possível de um sistema é uma medida de sua complexidade. De forma geral, o autor responde à pergunta de qual é a menor quantidade de informações necessárias para descrever requisitos e projeto de um sistema, de forma a corresponder à sua complexidade. O autor propõe esta resposta na forma de um metamodelo nomeado de *S*Metamodel*, cujo resumo é apresentado na Figura 2.3 abaixo.

Figura 2.3 – Resumo do metamodelo *S*Metamodel*.



Fonte: Adaptada de Schindel, 2011.

Na figura acima, Schindel (2011) define:

Stakeholder: entidade que tem uma participação de valor no comportamento ou desempenho do sistema.

Característica: aspecto do comportamento ou desempenho do sistema que tem valor a um *stakeholder*, descrito na linguagem e terminologia deste *stakeholder*. São parametrizadas por atributos.

Sistema: sistema de interesse modelado.

Sistema de Acesso: sistema externo que faz interface com o sistema de interesse.

Interação Funcional: troca de material, energia ou informação entre componentes do sistema, que possuem um papel funcional nesta interação.

Papel Funcional: descritos por seus comportamentos e parametrizados por atributos.

Estado: determina qual comportamento o sistema exibirá nas futuras interações, nas quais o sistema pode, inclusive, mudar de estado.

Entrada/Saída: material, energia ou informação trocadas durante uma interação funcional.

Interface: Associação do sistema (que apresenta a interface), uma ou mais Entradas/Saídas (que fluem pela interface), interações (que descrevem o comportamento da interface), e sistemas de acesso (que provêm o meio de transporte entre interfaces).

Componente de Projeto: descrito somente pela sua identidade e não seu comportamento, que é representado pelos papéis funcionais a ele alocados. São parametrizados por atributos.

Enunciados de Requisitos: texto ou outra descrição de comportamento dos papéis funcionais em uma interação. São parametrizados por atributos.

Matriz ‘A’ de Acoplamento: descreve as dependências quantitativas de valores (acoplamento paramétrico) entre os atributos das características de *stakeholder* e os atributos de papel funcional.

Matriz ‘B’ de Acoplamento: descreve as dependências quantitativas de valores (acoplamento paramétrico) entre os atributos de papel funcional e atributos de componente de projeto.

CP e CB: significam “caixa-preta” e “caixa-branca”, respectivamente.

Ainda segundo Schindel (2011), este metamodelo define somente conceitos. Um modelo prático que segue este metamodelo, e chamado de *S*Model* pelo autor, pode ser representado em qualquer linguagem e ferramenta de modelagem, desde que apresente tais conceitos.

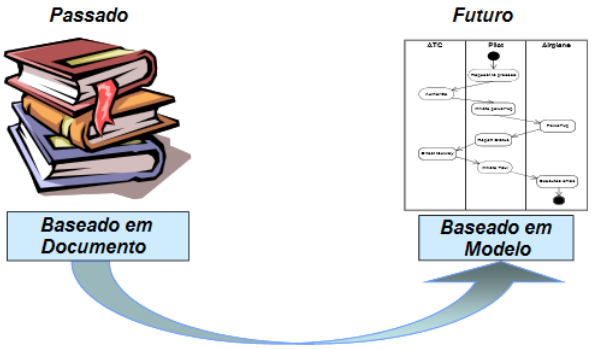
No contexto deste trabalho, tem-se:

Modelo. Representação de uma abstração do sistema real que evidencia somente seus aspectos relevantes ao estudo sendo conduzido.

2.2.4 A Engenharia de Sistemas Baseada em Modelo

As abordagens baseadas em modelo ocorrem cada vez mais na Engenharia de Sistemas, especificamente devido ao constante esforço de padronização e ao aumento de capacidades computacionais. Segundo o INCOSE (2007), a Engenharia de Sistemas baseada em modelo (MBSE) representa uma aplicação formal de modelagem para suportar as atividades de: requisitos de sistema; projeto; análise; verificação e validação. Ela se inicia na fase de projeto conceitual e continua por meio do desenvolvimento, em fases posteriores do ciclo de vida. A MBSE é parte de uma tendência de longo prazo em direção a abordagens centradas em modelo, já adotadas por outras disciplinas de Engenharia. É esperado que a MBSE substitua a abordagem centrada em documentos, atualmente praticada e influencie as futuras práticas da Engenharia de Sistemas, integrando-se à definição de seus processos, como ilustrado na Figura 2.4.

Figura 2.4 – Transição da Engenharia de Sistemas de documentos para modelos.



Fonte: Adaptada de INCOSE (2007).

INCOSE (2014) confirma esta visão para o futuro e vai um pouco mais além. Nesta nova visão do futuro da engenharia de sistemas, a MBSE já é uma prática atual em muitas empresas e o futuro será a utilização de um ambiente computacional, onde modelagem e simulação de todas as disciplinas da engenharia, incluindo a própria engenharia de sistemas, estarão totalmente integradas, possibilitando a extinção generalizada da gestão de documentação a elas associada.

Segundo Friedenthal et al. (2014), a MBSE facilita as atividades de Engenharia de Sistemas, que tradicionalmente se utilizam de abordagens centradas em documentos e resultam em melhorias na precisão de comunicação, especificação, projeto, projeto de integração de sistema e reúso de artefatos de sistema. Ainda de acordo com os autores, ao se adotar a MBSE, o produto gerado pelas atividades da Engenharia de Sistemas são modelos coerentes do sistema, onde a ênfase está na evolução e refinamento destes modelos, usando métodos e ferramentas adequados.

Holt e Perry (2013) definem MBSE como uma abordagem para desenvolver sistemas de sucesso a partir de um modelo compreendendo um conjunto de visões coerentes e consistentes, que refletem múltiplos pontos de vista de um sistema.

Jenney (2010) considera MBSE um substituto do conjunto de documentos que define ou descreve um sistema, incluindo documentos de requisitos, por meio de modelos.

Para Delligatti (2014), ao usar a abordagem MBSE, os engenheiros de sistemas executam as mesmas atividades de ciclo de vida e produzem o mesmo conjunto de entregas, mas o principal artefato destas atividades é um modelo integrado, coerente e consistente do sistema. Todos os outros artefatos são secundários e gerados, automaticamente, a partir do modelo do sistema. Ainda segundo este autor, a MBSE encontra-se baseada nos seguintes pilares de modelagem: linguagens, métodos e ferramentas.

2.3 A Linguagem para Modelagem de Sistemas SysML

Existem diversas linguagens que podem ser usadas na modelagem de, pelo menos, um aspecto de um sistema. Nesta seção, entra-se em detalhes de uma linguagem em especial: a SysML (OMG, 2017). Esta linguagem é uma linguagem visual padrão de modelagem gerenciada pelo *Object Management Group (*OMG) e usada para dar suporte à disciplina de Engenharia de Sistemas.

A SysML é uma linguagem gráfica de modelagem de uso geral para especificar, analisar, projetar e verificar sistemas complexos que podem incluir *hardware*, *software*, informações, pessoas, procedimentos e facilidades. Mais especificamente, esta linguagem provê representações gráficas com fundamento semântico para modelar requisitos de sistema, comportamentos, estruturas e parâmetros, usados para integração com outros modelos de análise de engenharia.

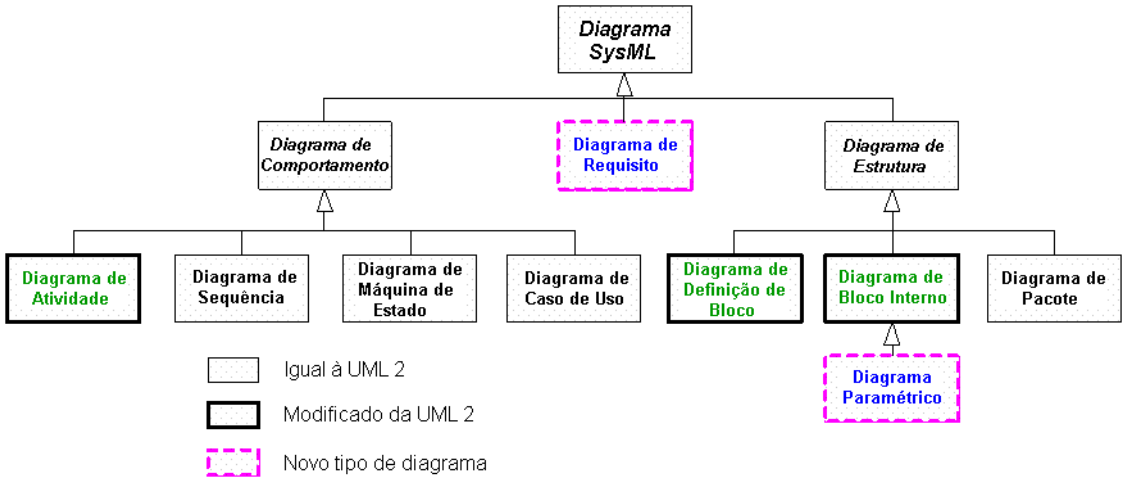
É bom ressaltar que a SysML não prescreve uma metodologia, mas simplesmente provê uma linguagem de modelagem visual. O uso de linguagens visuais padrão de modelagem, como a SysML, permite que membros dos times de desenvolvimento de sistemas, de *software* e de *hardware* se comuniquem de uma maneira padronizada.

Os diferentes tipos de diagramas da SysML estão identificados na Figura 2.5 abaixo. Para uma descrição detalhada dos diagramas da SysML, referencia-se Friedenthal et al. (2014) e Delligatti (2014).

Os diagramas de comportamento incluem: o diagrama de atividade; o diagrama de sequência; o diagrama de máquina de estado; e o diagrama de caso de uso. O diagrama de atividade permite representar os fluxos de material, energia, informação ou controle entre atividades. Um diagrama de sequência propicia a representação da interação entre partes de um sistema que colaboram para atingir um objetivo. O diagrama de máquina de estado fornece mecanismos para se descrever as transições de estado e ações que um sistema ou suas partes realizam em reposta a eventos. Um diagrama de

caso de uso possibilita uma descrição de alto nível de funcionalidades que são alcançadas por meio de interações entre sistemas ou partes de sistemas.

Figura 2.5 – Tipos de Diagramas da SysML.



Fonte: Adaptada de OMG (2017).

A SysML inclui uma construção gráfica para representar requisitos textuais e relacioná-los com outros elementos do modelo. O diagrama de requisitos permite a captura de hierarquias de requisitos e a derivação de requisitos. Os relacionamentos de satisfação e verificação permitem ao engenheiro de sistemas relacionar requisitos a elementos do modelo que, respectivamente, satisfazem ou verificam estes requisitos. Este tipo de diagrama ainda provê uma ligação lógica entre ferramentas típicas de gestão de requisitos e o modelo do sistema.

A estrutura de um sistema é representada por meio de diagramas que incluem: o diagrama de definição de bloco; o diagrama interno de bloco; e o diagrama de pacote. Um diagrama de definição de bloco possibilita a descrição da hierarquia do sistema e a classificação do sistema e/ou seus componentes, tendo o 'bloco' como unidade básica de estrutura e usado para representar *hardware*, *software*, facilidades, pessoal, ou qualquer outro elemento do sistema. O diagrama interno de bloco propicia a descrição da estrutura interna do sistema (e/ou de um de seus componentes) em termos de suas partes, portas (pontos de interação) e conectores (meios pelos quais as partes trocam material, energia ou informação). O diagrama de pacote

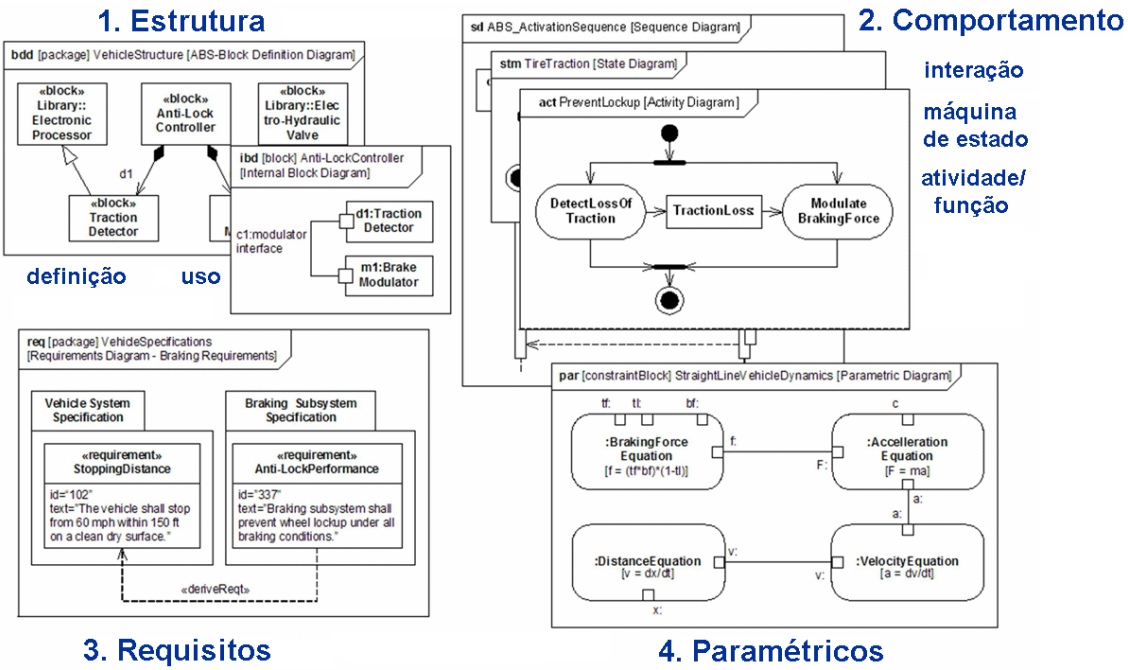
propicia a divisão do sistema em agrupamentos representados por pacotes lógicos, mostrando as dependências entre eles, ou seja, pacotes podem depender de outros pacotes. Ilustram, ainda, a arquitetura de um sistema, as camadas, subsistemas, pacotes, etc. Definem, também, possíveis extensões da linguagem, chamadas de perfis. A própria SysML é um exemplo de perfil, que estende a UML 2 (OMG, 2015). O exemplo mais comum do uso de perfis acontece quando se deseja incorporar à linguagem os jargões utilizados no processo de Engenharia de Sistemas de uma dada organização. Tipos específicos de elementos de modelagem podem ser criados (e.g.: função, cartão eletrônico, etc) e parâmetros dedicados a estes tipos específicos (e.g.: desempenho, grupo de cartão, etc).

Finalmente, o diagrama paramétrico propicia a representação de restrições dos valores das propriedades do sistema, tais como, desempenho, confiabilidade e propriedades de massa, e serve como um meio de integração entre os modelos de especificação e os projetos com modelos de análise de engenharia.

O princípio para modelagem de sistemas com o uso da SysML se baseia em quatro pilares: estrutura, comportamento, requisitos e paramétricos. Na Figura 2.6 abaixo, mostram-se os principais diagramas utilizados em cada um destes pilares. Note que o diagrama de pacote e o diagrama de caso de uso não foram mencionados, mas eles pertencem, respectivamente, aos pilares de estrutura e comportamento. Os termos utilizados nos diagramas que exemplificam cada pilar não foram traduzidos do Inglês ou por se tratarem de palavras reservadas da SysML ou por não terem relevância no entendimento da figura. O primeiro pilar, o de estrutura, contém os diagramas de definição de bloco, interno de bloco e de pacotes para possibilitar a decomposição do sistema em termos de suas partes constituintes e a definição de como são as interfaces entre estas partes de um ponto de vista estático. O segundo pilar, o de comportamento, contém os diagramas de atividade, sequência, máquina de estado e caso de uso para possibilitar a descrição comportamental do sistema e de suas partes constituintes nos

mais diversos cenários do seu ciclo de vida. O terceiro pilar, o de requisitos, contém o diagrama de requisito para propiciar a elaboração da árvore de requisitos que, transversalmente, justifica a existência dos diferentes elementos do modelo. Finalmente, o quarto pilar, o de paramétricos, contém o diagrama paramétrico que propicia uma definição mensurável das restrições das propriedades do sistema.

Figura 2.6 – Os quatro pilares da SysML.



Fonte: Adaptada de OMG (2017).

2.4 Metodologias para MBSE

Nesta seção, apresentam-se várias metodologias de MBSE que se beneficiam da SysML para executar suas atividades.

Não é possível iniciar uma discussão sobre metodologias para MBSE sem mencionar o trabalho de Wynmore (1993), que provê uma rigorosa infraestrutura matemática de modelos de sistemas de tempo discreto e aspectos de modelos de tempo discreto, tais como casualidade e invariância no tempo, acoplamento de sistemas, homomorfismo de sistemas e modos de sistemas. O autor desenvolve uma teoria matemática específica para facilitar o processo de projeto de sistemas chamada *tricotyledon theory*. Esta teoria

sugere que um projeto de sistema requer a definição de três conjuntos ou espaços de: funcionalidade, construção e exequibilidade. O primeiro propicia a definição de requisitos funcionais do sistema. O segundo propicia a cobertura de um conjunto de sistemas possíveis de serem construídos, levando-se em conta as restrições tecnológicas. Finalmente, o terceiro propicia a combinação dos dois conjuntos anteriores e a geração de um conjunto de sistemas implementáveis que satisfaçam os requisitos funcionais.

Várias metodologias para MBSE surgiram desde então. A lista apresentada a seguir está baseada no trabalho de Estefan (2008), com a inclusão de algumas outras metodologias para MBSE, também consideradas como importantes contribuições para melhor compreensão do assunto.

Exclusivamente para as fases do ciclo de vida de especificação de requisitos de sistema e descrição de arquitetura, Hatley et al. (2000) revisitou o trabalho feito por Hatley e Pirbhai (1988) e criou o método chamado Hatley-Hruschka-Pirbhai (HHP). Este método pode ser visto como mais um passo na evolução da série de métodos de análise estruturada, usados na especificação de sistemas, e desenvolvidos por Yourdon e Constantine (1975), DeMarco (1979) e Yourdon (1989). Este método provê passos detalhados para criar modelos funcionais e implementacionais consistentes, focando na especificação de requisitos de sistema e na descrição de arquitetura com um nível de abstração suficiente para garantir bons requisitos.

Loureiro (1999) propõe um *framework* para o desenvolvimento integrado de produtos complexos. Como parte desta estrutura, o autor propõe um método de engenharia de sistemas, cobrindo todo o ciclo de vida do sistema (i.e. desde sua concepção até seu descarte), onde diversos tipos de modelos são empregados para representar os mais diversos aspectos do sistema de interesse. Dentre estes diversos tipos de modelo, podem ser destacados os modelos criados a partir de métodos de análise estruturada, diagramas de blocos e diagramas de transição de estados.

A metodologia OOSEM (LYKINS et al., 2000) integra uma abordagem *top-down* baseada em modelo para suportar a especificação, análise, projeto e verificação de sistemas, e possui os seguintes objetivos: captura e análise de requisitos e informações de projeto para especificar sistemas complexos, integração com *software* orientado a objeto, hardware e outros métodos de engenharia e suporte para reúso no nível de sistema e evolução de projeto. Melhorando técnicas de orientação a objeto com uma abordagem *top-down* mais tradicional de Engenharia de Sistemas, a metodologia cobre as seguintes atividades: analisar as necessidades dos *stakeholders*, definir requisitos de sistema, definir a arquitetura lógica, sintetizar candidatas para a arquitetura alocada, otimizar e avaliar alternativas e validar e verificar o sistema. A linguagem predominante é a SysML, onde casos de uso são usados para capturar as necessidades (casos de uso de missão) e requisitos de sistema (casos de uso de sistema). Seus cenários são descritos usando diagramas de atividades, os quais são também utilizados para a elaboração da decomposição funcional. Blocos representam os componentes lógicos do sistema que alocam as funções identificadas, bem como os componentes físicos do sistema que alocam os componentes lógicos. Diagramas paramétricos são usados para avaliar as diferentes alternativas em termos de projetos, ao comparar valores das propriedades emergentes do sistema sendo modelado em cada uma das alternativas escolhidas (e.g.: massa, tempo médio entre falhas, desempenho, etc).

A metodologia OPM (DORI, 2002) é um paradigma holístico e formal para o desenvolvimento de sistemas, suporte ao ciclo de vida e evolução, com as seguintes principais características: Ontologia, Notação e processo de Desenvolvimento de Sistema. Ontologia inclui os elementos básicos da OPM, seus atributos e os possíveis relacionamentos entre eles. A Notação representa a Ontologia graficamente ou textualmente. O processo de Desenvolvimento de Sistema é dividido em subprocessos que usam a mesma Ontologia e Notação da OPM. São eles: Especificação de Requisitos, Análise e Projeto, Implementação e Uso, e Manutenção. A OPM suporta um conjunto de semânticas de modelagem, combinando modelos visuais

formais, conhecidos por *Object-Process Diagrams* (OPDs), com sentenças restritas de linguagem natural, conhecidas por Object-Process Language (OPL), para expressar a função, a estrutura e o comportamento de sistemas em um único modelo integrado. Todo OPD pode ser convertido em uma sentença ou parte de uma sentença OPL semanticamente equivalente e vice-versa. A OPL é uma linguagem com dois propósitos, orientada para humanos bem como para máquinas. No nível de modelagem, a OPM é construída baseada em três tipos de entidades: objetos (elementos que existem ou tem potencial de existirem fisicamente ou mentalmente), processos (padrões de transformação pelo qual um objeto passa) e estados (situações nas quais um objeto pode estar).

A metodologia RUP SE (CANTOR, 2003) aplica a disciplina e as melhores práticas do RUP (KRUCHTEN, 2003) para o desenvolvimento de *software* nas fases de especificação, análise, projeto e desenvolvimento de sistemas, e possui os seguintes objetivos: ajudar organizações a economizar tempo, cortar custos, reduzir riscos e melhorar a qualidade dos sistemas por elas construídos. Estendendo o RUP para suportar desenvolvimento de sistemas direcionado por modelo, a metodologia RUP SE considera as seguintes atividades: modelagem de negócio, requisitos, análise e projeto, implementação, teste e implantação, por meio das fases de concepção, elaboração, construção e transição de um projeto. Uma vez que as decisões de projeto tenham sido capturadas em pontos de vista e especificadas em níveis no modelo, a arquitetura do sistema é capturada em um conjunto de diagramas UML e SysML. Casos de uso são extensivamente usados para derivar, a partir de requisitos de negócio, requisitos de sistema e, então, requisitos de subsistemas, e são o ponto de partida para a modelagem de atividades.

A metodologia Harmony-SE (DOUGLAS, 2005) espelha o clássico modelo V de desenvolvimento de ciclo de vida de projeto de sistema, apresentado anteriormente na Figura 1.1, e possui os seguintes objetivos principais: identificar e derivar as funcionalidades requeridas do sistema, identificar

modos e estados associados do sistema e, finalmente, alocar as funcionalidades e modos do sistema à arquitetura física. Usando uma abordagem direcionada a requisições de serviços combinada à SysML como linguagem de modelagem, a metodologia inclui os seguintes elementos de processo de alto nível: análise de requisitos, análise funcional do sistema e projeto arquitetural. Na abordagem direcionada a requisições de serviços, a estrutura do sistema é descrita por meio de diagramas estruturais da SysML usando blocos como elemento estrutural básico. A comunicação entre os blocos é baseada na troca de mensagens, descrita em diagramas de sequência. Serviços providos são descritos usando portas e interfaces. A decomposição funcional é feita por meio da decomposição de atividades em diagramas de atividade.

A metodologia JPL SA (INGHAM et al., 2006) melhora uma arquitetura de controle baseada em estado e modelo, que provê um processo para captura de requisitos de sistema e de *software* na forma de modelos explícitos, ajudando a diminuir a distância entre os requisitos de *software* especificados por engenheiros de sistema e a implementação destes requisitos pelos engenheiros de *software*. A metodologia define três principais atividades: modelagem de comportamento baseada em variáveis de estado do sistema e relacionamentos entre elas, projeto de *software* baseado em estado, descrevendo os métodos pelos quais os objetivos serão alcançados, e engenharia de operações direcionada a objetivos, capturando objetivos de missão em cenários detalhados motivados pelas intenções dos operadores. Usando a metodologia, também é possível aumentar o escopo da análise e decomposição funcional tradicionais com elementos da análise de estado para produzir uma abordagem mais abrangente e rigorosa para a modelagem de comportamento do sistema.

Booch et al. (2007), baseado em trabalhos como os de Coad e Yourdon (1991a) e Coad e Yourdon (1991b) sobre métodos orientados a objetos, formalizaram os passos a serem seguidos para especificar completamente um sistema de *software*, dividindo o processo em macro e micro. O propósito

do processo macro é guiar de maneira ampla o desenvolvimento do sistema e divide-se em cinco disciplinas: requisitos, análise e projeto, implementação, teste, e implantação. Estas disciplinas são executadas em uma dada ordem dentro de uma interação e são suportadas pelas seguintes disciplinas: gerenciamento de projeto, gerenciamento de configuração e mudança, e ambiente. Podem existir tantas interações quanto for necessário. Focando na análise e projeto, os autores definem o processo micro para transformar os requisitos produzidos pelo processo macro em uma arquitetura que será implementada, testada e implantada. Neste processo, as fases de análise e projeto são executadas em diferentes níveis de abstração, dependendo principalmente do tamanho do sistema. A análise foca no comportamento que o sistema deve apresentar para resolver o problema e o projeto foca na criação de elementos que proverão o comportamento requerido pelos elementos de análise. A UML é utilizada como linguagem de modelagem em todas as atividades para garantir a consistência dos modelos de análise e projeto.

A metodologia SYSMOD (WEILKIENS, 2007) é baseada em um padrão largamente utilizado: identificar um elemento, descrever o contexto no qual ele está inserido (visão externa) e, então, imergir (visão interna). Ela está dividida em duas partes principais: análise e projeto. Durante a fase de análise, o engenheiro de sistemas considera o sistema como caixa-preta e determina suas funções, executando as seguintes atividades: descrever o contexto do projeto, determinar os requisitos, criar um glossário, modelar o contexto do sistema, modelar os casos de uso e modelar os dados de domínio. Durante a fase de projeto, o engenheiro de sistemas entra nos detalhes dos aspectos implementacionais do sistema, executando as seguintes atividades: modelar as interações entre o sistema e as entidades externas, derivar as interfaces do sistema, modelar as estruturas do sistema e derivar o modelo de estados do sistema. Durante todas estas atividades, a SysML é largamente utilizada. Diagramas interno de bloco são usados para descrever o contexto do sistema, bem como aspectos internos da estrutura do sistema (i.e. como as partes constituintes do sistema estão

interconectadas e o que elas trocam entre si). Diagramas de definição de bloco são usados para descrever os dados de domínio, bem como mostrar a estrutura do sistema. Casos de uso implementados com o auxílio de diagramas de atividade são usados para expressar as funções que o sistema deve realizar. Diagramas de requisito coletam os requisitos de sistema. Diagramas de máquina de estado descrevem como os estados do sistema evoluem no decorrer do tempo. Portas são usadas como pontos de interação do sistema e de seus componentes, e são usadas para descrever interfaces internas e externas do sistema.

Buede (2009) propõe uma metodologia completa para projeto de sistemas, composta por cinco passos principais: requisitos e definição do problema de projeto, desenvolvimento da arquitetura funcional, desenvolvimento da arquitetura física, desenvolvimento da arquitetura alocada e projeto de interface. No primeiro passo descobrem-se os requisitos e define-se o escopo do sistema a ser desenvolvido usando uma abordagem de caixa-preta. Nos três passos intermediários, o autor usa uma abordagem de decomposição funcional para determinar as funções que o sistema executará, a estrutura hierárquica do sistema em termos de seus componentes físicos e, finalmente, a alocação das funções aos componentes do sistema. Durante o processo de alocação funcional, a estrutura física do sistema é revisitada para garantir uma correta alocação (i.e. uma dada função alocada a um único componente físico). Finalmente, no último passo dão-se os detalhes de como as interfaces serão implementadas. A linguagem *Integrated Computer Aided Manufacturing Definition for Function Modeling* (i.e. IDEF0) é utilizada para todas as arquiteturas desenvolvidas ao longo da metodologia.

A metodologia Vitech MBSE (LONG e SCOTT, 2011) é baseada em quatro atividades simultâneas de Engenharia de Sistemas, que são relacionadas e mantidas por meio de um repositório comum. São elas: análise de requisitos de entrada, análise de comportamento, análise de arquitetura e verificação e validação de projeto. Requisitos de entrada são rastreados para

comportamentos e componentes físicos, comportamentos são alocados aos componentes físicos e requisitos, comportamentos e componentes físicos são verificados por elementos de verificação e validação. Este conjunto de atividades de Engenharia de Sistemas é executado de uma maneira interativa, para diferentes camadas de abstração do sistema. O engenheiro de sistemas procegue para a próxima camada quando as atividades da camada anterior forem finalizadas. Uma linguagem dedicada para a definição de sistemas, a *System Definition Language* (SDL), gerencia a sintaxe (i.e. estrutura) e semântica (i.e. significado) dos artefatos do modelo. Para suportar a atividade de análise de comportamento, a metodologia tem à sua disposição as seguintes linguagens de modelagem visual: *Enhanced Function Flow Block Diagrams* (EFFBDs), *Function Flow Block Diagrams* (FFBDs), gráficos N2 e diagramas de comportamento.

Holt e Perry (2013) propõem uma metodologia baseada em modelo completa, composta de três elementos principais: uma ontologia (identificando e descrevendo os conceitos e termos usados na abordagem), um *framework* de MBSE (descrevendo um uso específico da ontologia) e um Ponto de Vista (focando em um subconjunto da ontologia e tem um propósito específico, definindo visões usadas na abordagem). Com o auxílio da notação SysML, os autores desenvolvem um modelo cobrindo principalmente três aspectos do sistema: processo (onde definem-se os *stakeholders* e o ciclo de vida do sistema, em conjunto com as atividades a serem executadas), requisitos (onde desenvolvem-se requisitos de *stakeholders* e de sistema) e arquitetura (onde detalha-se a composição do sistema). O primeiro aspecto é modelado com a ajuda de atividades e diagramas de atividade. O segundo aspecto é modelado usando casos de uso e diagramas de caso de uso. Finalmente, o terceiro aspecto é modelado com blocos e diagramas de definição de bloco, combinados a diagramas internos de bloco.

Schindel (2015) propõe uma metodologia de engenharia de sistemas baseada em padrões chamados de *S*Patterns* e modelos chamados de *S*Models*. Este tipo de modelo deve ser construído seguindo as definições

do metamodelo chamado *S*Metamodel* (SCHINDEL, 2011). Nesta abordagem, o autor propõe a criação de modelos *S*Models* configuráveis e reutilizáveis em diferentes configurações ou famílias de sistema. Estes modelos são, na verdade, os padrões que servirão de base para instanciação de modelos de sistemas de interesse. Ainda segundo o autor, os constantes reusos destes padrões possibilitam o aprendizado e realimentação para seu aperfeiçoamento constante. A aplicação dos padrões em um projeto recebe nome de processo de configuração de padrões e a realimentação para aperfeiçoamento recebe o nome de processo de gestão de padrões.

Friedenthal e Oster (2017) propõem uma metodologia de MBSE, que se ocupa desde o planejamento estratégico do esforço total de modelagem do sistema de interesse até o estágio de integração e verificação deste sistema, definindo três grupos principais de atividades: um para a produção do modelo do sistema de interesse, outro para manter a gestão de requisitos e, finalmente, um terceiro para as análises e tomadas de decisão. Os autores propõem o desenvolvimento de um satélite como exemplo para exercitar a metodologia proposta e usam a SysML como linguagem para elaborar o modelo do sistema de interesse. Dentre as etapas da metodologia proposta pelos autores, destacam-se, no escopo desta Tese de Doutorado, duas delas: aquela com o objetivo de se produzir os requisitos de sistema e aquela com o objetivo de se produzir a arquitetura do sistema de interesse.

Apesar da grande quantidade de metodologias para MBSE, nota-se que todas elas se propõem a separar o ‘o que’ do ‘como’, ou seja, separar os requisitos, da solução. Cada uma destas metodologias o fazem de uma maneira específica e dedicada, usando distintas linguagens e abordagens.

2.5 Conversão de Modelos Descritivos em Texto

A conversão de modelos descritivos em textos já está bem difundida na Engenharia de Software, onde modelos UML são convertidos em esqueletos de código-fonte ou até mesmo em documentação do software modelado (OMG, 2008). Esta conversão se torna possível com a utilização de padronizações, principalmente estabelecidas pela OMG, na área de

linguagens utilizadas para a transformação de modelos. A linguagem mais relevante no contexto deste trabalho é a MOFM2T (OMG, 2008), que pode ser usada para expressar transformações de um modelo descritivo em textos, como por exemplo, códigos-fonte ou documentação. Ela é parte da arquitetura dirigida a modelo mantida pela OMG (*Model Driven Architecture* – MDA) e reusa vários conceitos da MOF (OMG, 2014), que é a padronização da OMG (e também da ISO) para o desenvolvimento de arquiteturas de metamodelos e usada na própria definição da UML.

Leopold et al. (2012) investigam a modelagem de processos como forma de entender, documentar e reprojetar as operações das organizações. Os autores evidenciam, também, o problema de que estes modelos são, em geral, somente compreendidos pelos analistas de negócio, que por sua vez não são especialistas do domínio na qual a organização está focada. Como solução para este problema de comunicação, os autores propõem uma abordagem que transforma, de forma confiável, um modelo de processo feito utilizando a BPMN (OMG, 2013) em textos de linguagem natural.

Meziane et al. (2008) evidenciam um dos principais problemas encontrados no desenvolvimento de sistemas intensivos em software: a consistência entre os requisitos de entrada, usualmente expressos em linguagem natural, e o sistema desenvolvido. Para os autores, esta consistência é considerada pelos engenheiros de software como solução para minimizar os erros e permitir a verificação e validação antecipada do sistema. Entretanto, ainda segundo os autores, o problema reside no fato de que a garantia desta consistência fica confinada aos modelos de análise e projeto criados durante o desenvolvimento do sistema, excluindo o usuário final deste processo. Como solução para este problema, os autores propõem um sistema que gera linguagem natural a partir de diagramas de classe da UML. Com isso, os modelos de análise e projeto poderiam ser transformados em linguagem natural e, consequentemente, os usuários finais poderiam ser envolvidos na verificação de consistência destes modelos. Por outro lado, Deeptimahanti e Sanyal (2011), considerando o mesmo problema e a mesma causa, propõem

justamente o contrário: gerar, de forma automática, modelos UML, a partir de requisitos escritos em linguagem natural.

Peleg e Dori (1999) mostraram através de um estudo de caso como os OPDs podem ser transformados em OPL, com o intuito de apresentar a maneira natural, porém formal, de especificar os resultados e decisões da análise e projeto de um sistema. Segundo estes autores, esta linguagem é projetada para ser bem próxima do Inglês como linguagem natural, mas com uma sintaxe muito mais rigorosa e limitada, tornando-a livre de ambiguidades e provendo uma base estável para implementação da geração automática de códigos executáveis e da definição de estruturas de bases de dados. Indo na linha de geração de código fonte a partir de um modelo descritivo, Reinhartz-Berger e Dori (2004) fazem uma comparação entre a UML e a OPM examinando justamente suas respectivas capacidades de geração de código fonte, chegando à conclusão de que modelos elaborados utilizando a OPM possibilitam a geração da lógica completa de aplicação e não somente o esqueleto do código fonte, como acontece com modelos utilizando a UML.

Feiler e Gluch (2013) descrevem como usar a *Architecture Analysis & Design Language (*AADL), linguagem padronizada pela *Society of Automotive Engineers (*SAE) que provê uma infraestrutura para a engenharia de software baseada em modelo, para capturar tanto as características estáticas quanto as características dinâmicas de um software embarcado. Os elementos e diagramas desta linguagem também podem ser transformados em textos de implementação, que serão utilizados para a geração dos aplicativos executáveis correspondentes ao software modelado.

Esta seção mostrou que a geração de informação textual a partir de modelos descritivos é algo comum, mas que o maior foco ainda está na disciplina de engenharia de software, onde o código-fonte (ou, pelo menos, um esqueleto inicial de código-fonte) e uma eventual documentação do software são gerados automaticamente a partir de seu modelo descritivo.

Neste Capítulo 2, foram apresentados os principais conceitos e definições utilizados no trabalho e as principais linhas de pesquisa que permeiam o

tema desta Tese de Doutorado. Com o objetivo de propor uma metodologia para modelagem de sistemas utilizando a SysML e que possibilite a geração automática de requisitos de sistema, pôde-se revisar aqui os conceitos e definições envolvendo requisitos (i.e. características de bons requisitos e como se deve escrever bons requisitos). Além disto, pôde-se revisar neste capítulo o que é a MBSE e a SysML, algumas metodologias existentes na literatura para a abordagem MBSE e técnicas já existentes que transformam modelos descritivos em textos.

3 REVISÃO BIBLIOGRÁFICA

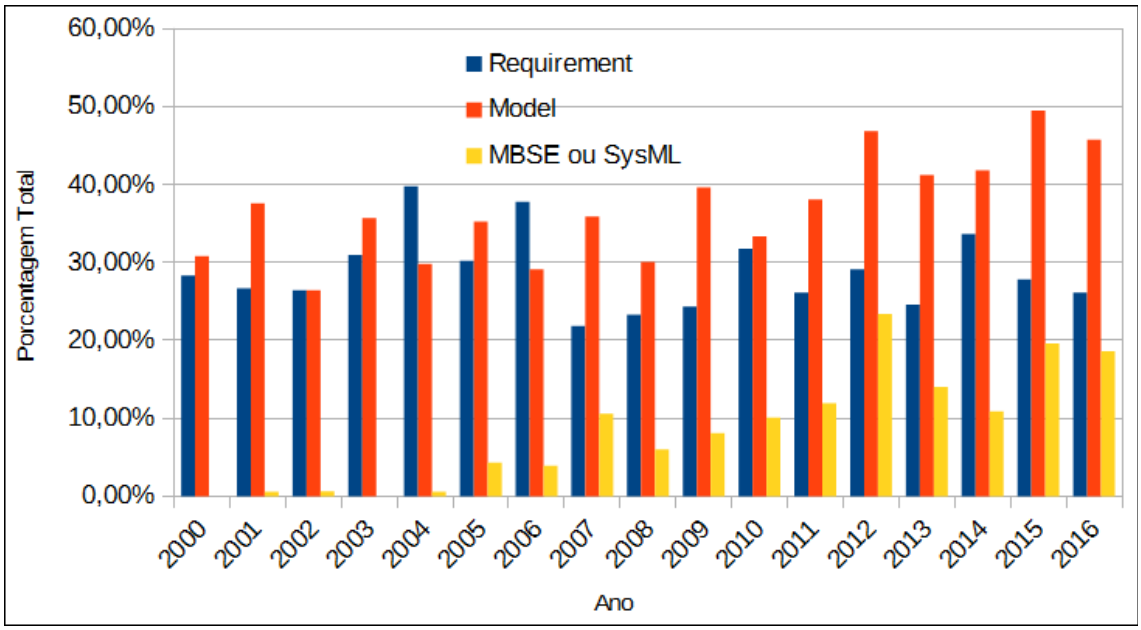
Neste capítulo, apresentam-se um estudo estatístico para se determinar a relevância da pesquisa conduzida e uma revisão da literatura existente sobre temas semelhantes ao tema desta Tese de Doutorado.

3.1 Relevância da Pesquisa Conduzida

Abaixo, apresentam-se os estudos estatísticos realizados de aparições dos termos "*requirement*", "*model*", "MBSE" e "SysML" em artigos publicados em simpósios (ISSN 2334-5837) e periódicos (ISSN 1520-6858) do INCOSE, e estudos estatísticos de correspondências para buscas tanto da frase "*writing good requirements*" quanto dos termos "MBSE" e "SysML" no Google Acadêmico. O conjunto completo dos dados levantados nestes estudos estatísticos encontram-se no Apêndice A.

Na Figura 3.1 abaixo, mostra-se um gráfico com a porcentagem total de artigos publicados nos dezessete últimos simpósios do INCOSE e que apresentaram os termos "*requirement*", "*model*", "MBSE" ou "SysML", nos seus títulos ou nos seus resumos.

Figura 3.1 – Porcentagem Total de Artigos dos Simpósios do INCOSE com Termos Relacionados à Tese.



Fonte: Produção do autor.

No eixo horizontal do gráfico da Figura 3.1 acima têm-se os anos de 2000 a 2016, quando aconteceram os simpósios do INCOSE. Já no eixo vertical, tem-se a porcentagem total de artigos publicados em um dado ano que apresentaram um dado termo.

A partir dos dados usados para gerar o gráfico acima, tem-se que o termo "*requirement*" apareceu na média em 28,75% dos artigos publicados nos últimos dezessete anos, com um desvio padrão de 4,89%. Já o termo "*model*" apareceu na média em 36,87% dos artigos no mesmo período, com um desvio padrão de 6,66%.

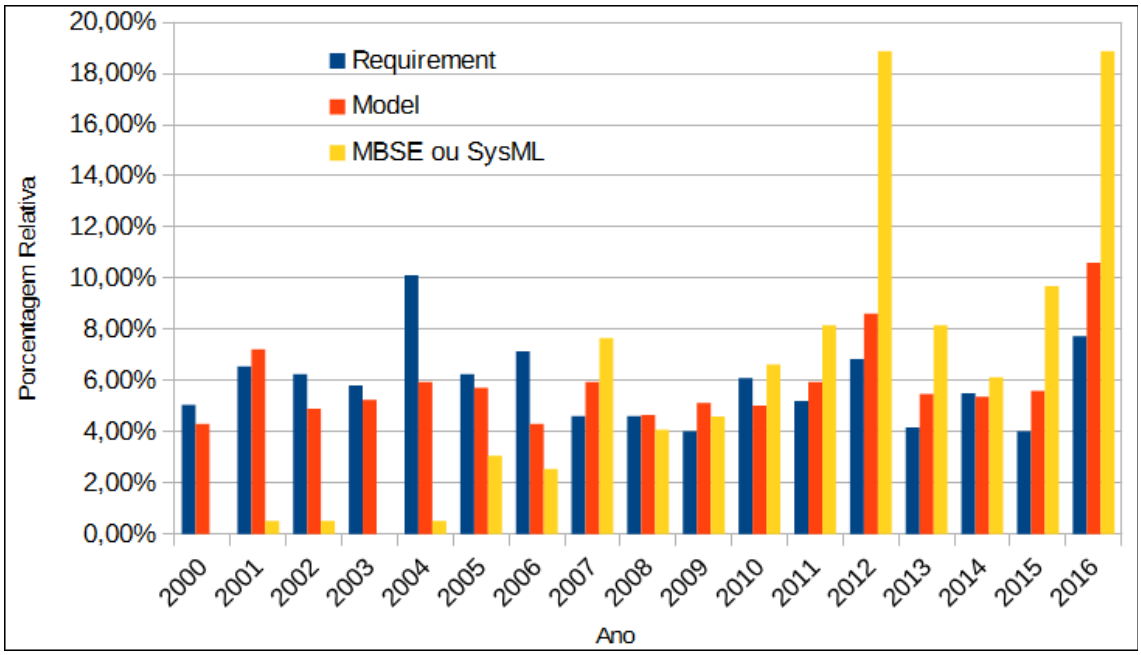
O valor relativamente alto das médias e o valor relativamente baixo dos desvios padrões confirma a importância destes termos para a comunidade de engenharia de sistemas publicando trabalhos nestes simpósios.

Ainda no gráfico da Figura 3.1 acima, nota-se que os termos "MBSE" ou "SysML" começaram a aparecer, de forma mais expressiva, nos artigos publicados nos simpósios do INCOSE a partir dos anos de 2005 e 2006, sendo este último o ano em que a OMG adotou a SysML como linguagem oficial de modelagem para sistemas. A partir dos dados que geraram o gráfico e descartando os cinco primeiros anos, tem-se uma média próxima dos 12% para este grupo de termos. Mais ainda, considerando somente os dois últimos anos, este grupo de termos atinge valores próximos dos 20% de artigos publicados nos correspondentes simpósios do INCOSE. Estes números demonstram que a modelagem de sistemas aplicadas à engenharia de sistemas e o uso da SysML é igualmente um assunto amplamente abordado nestes simpósios e que permanece no foco dos pesquisadores.

Já no gráfico da Figura 3.2 abaixo, mostra-se o mesmo escopo estatístico utilizado na concepção do gráfico da Figura 3.1 acima, mas agora apresentando a porcentagem de artigos para um dado ano contendo um dos termos "*requirement*", "*model*", "MBSE" ou "SysML", no título ou no resumo, relativa ao número total de artigos ao longo dos dezessete anos considerados, contendo o termo em questão. Ou seja, esta porcentagem é calculada como sendo a razão entre o número de artigos de um dado ano

contendo um dado termo pelo número total de artigos nos dezessete anos considerados contendo o mesmo termo.

Figura 3.2 – Porcentagem Relativa de Artigos dos Simpósios do INCOSE com Termos Relacionados à Tese.



Fonte: Produção do autor.

A construção utilizada no gráfico da figura acima normaliza os números para cada um dos termos pesquisados e permite realçar se em algum dado ano algum termo foi mais explorado do que nos demais anos. Ou seja, a soma dos valores das porcentagens para um dado termo dos dezessete anos considerados é igual a 100%.

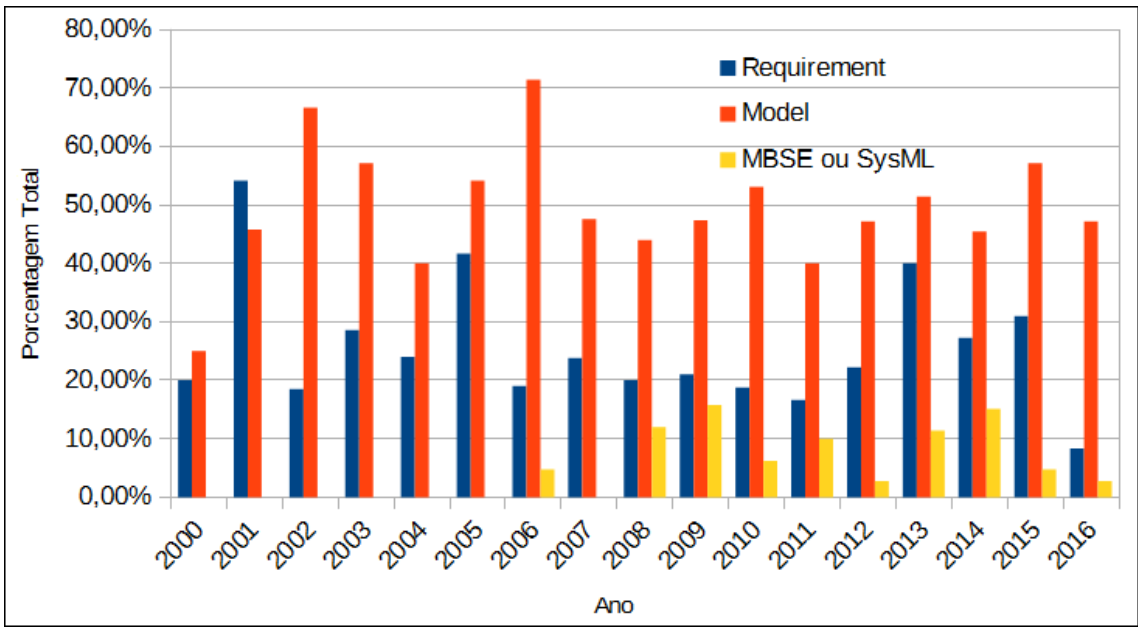
No caso dos termos "*requirement*" e "*model*", observou-se um total de 672 artigos e 858 artigos, respectivamente, contendo estes termos. Ambos os termos apresentaram uma média de 5,88% e desvios padrões de 1,56%, para o termo "*requirement*", e 1,61%, para o termo "*model*". Estes números mostram que estes termos vêm sendo explorados nos últimos dezessete anos em artigos destes simpósios de forma constante e na mesma proporção. Já para os termos "MBSE" ou "SysML", novamente observa-se que só a partir do ano de 2005 começam as publicações de artigos contendo os termos. Observa-se, também, que nos anos específicos de 2012 e 2016 os correspondentes simpósios do INCOSE apresentaram uma maior

concentração de artigos contendo estes termos nos seus títulos ou resumos. Este fato pode ter como uma das causas a oficialização de uma nova versão da especificação da SysML, que no meio de 2012 teve sua revisão 1.3 publicada e no final de 2015, sua versão 1.4. No total, nestes dezessete anos de simpósios do INCOSE, foram publicados 196 artigos contendo um destes dois termos. Mais ainda, desconsiderando os cinco primeiros anos e os anos de 2012 e 2016, chega-se a uma média de 9,75% com desvio padrão de 3,87%, mostrando uma constante presença destes termos nos artigos publicados nos simpósios do INCOSE.

Nas duas próximas figuras, apresentam-se gráficos similares à aqueles já apresentados na Figura 3.1 e na Figura 3.2, entretanto, agora, tendo como fonte dos dados os artigos publicados nos periódicos do INCOSE, desde o ano de 2000 até o ano de 2016.

Na Figura 3.3 abaixo, mostra-se um gráfico da porcentagem total de artigos publicados nos dezessete últimos periódicos do INCOSE e que apresentaram os termos "*requirement*", "*model*", "MBSE" ou "SysML", nos seus títulos ou nos seus resumos.

Figura 3.3 – Porcentagem Total de Artigos dos Periódicos do INCOSE com Termos Relacionados à Tese.



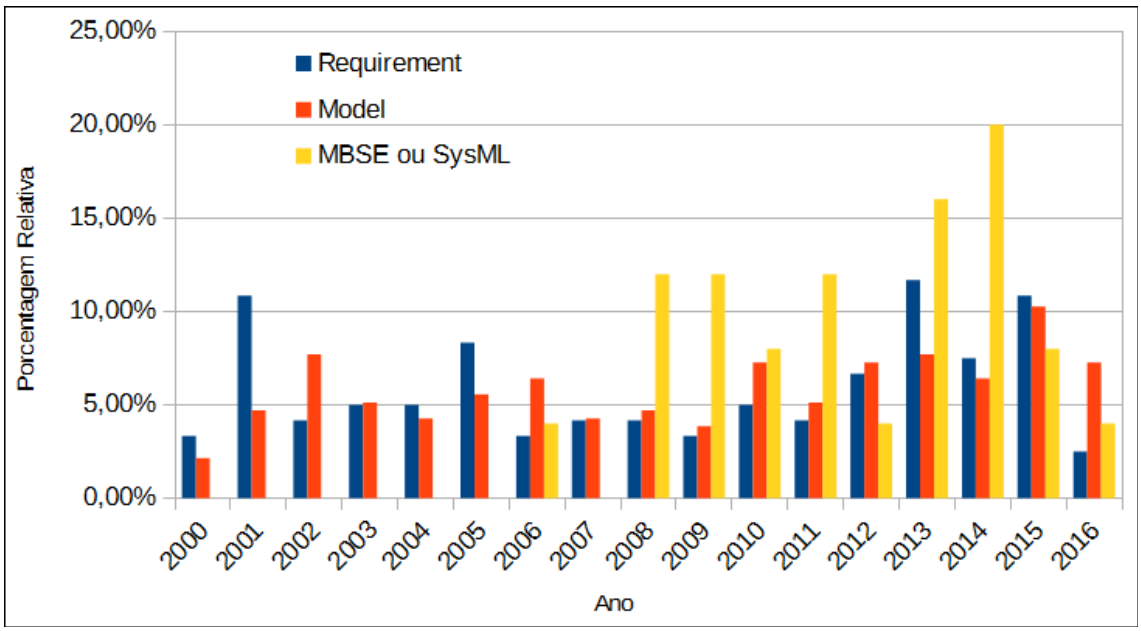
Fonte: Produção do autor.

A partir dos dados usados para gerar o gráfico acima, tem-se que o termo "*requirement*" apareceu, em média, em um quarto dos artigos publicados (25,59%). Já o termo "*model*" apareceu, em média, em metade dos artigos no mesmo período (49,46%). Novamente, constata-se a importância destes termos para a comunidade de engenharia de sistemas que publica trabalhos nestes periódicos.

Já os termos "MBSE" ou "SysML" começaram a aparecer a partir do ano de 2006, ano este em que a OMG adotou a SysML como linguagem padrão para modelagem de sistemas, com uma média de presença em 7,79% dos artigos publicados.

Na Figura 3.4 abaixo, mostra-se o mesmo escopo estatístico utilizado na concepção do gráfico da Figura 3.3 acima, mas agora apresentando a porcentagem de artigos para um dado ano contendo um dos termos "*requirement*", "*model*", "MBSE" ou "SysML", no título ou no resumo, relativa ao número total de artigos ao longo dos dezessete anos considerados, contendo o termo em questão.

Figura 3.4 – Porcentagem Relativa de Artigos dos Periódicos do INCOSE com Termos Relacionados à Tese.



Fonte: Produção do autor.

De forma similar ao gráfico da Figura 3.2, a construção utilizada no gráfico acima normaliza os números para cada um dos termos pesquisados e permite realçar se em algum dado ano algum termo foi mais explorado do que nos demais anos. Ou seja, a soma dos valores das porcentagens para um dado termo dos dezessete anos considerados é igual a 100%.

No caso dos termos "*requirement*" e "*model*", observou-se um total de 120 artigos e 234 artigos contendo estes termos, respectivamente. Ambos os termos apresentaram uma média de 5,88% e desvios padrões de 2,92%, para o termo "*requirement*", e 1,93%, para o termo "*model*". Estes números novamente mostram que estes termos vêm sendo explorados nos últimos dezessete anos em artigos destes periódicos de forma constante e na mesma proporção. Mais ainda, estes valores estão na mesma ordem de grandeza daqueles vistos no caso de artigos publicados nos simpósios do INCOSE para o mesmo período analisado, vistos anteriormente no gráfico da Figura 3.7.

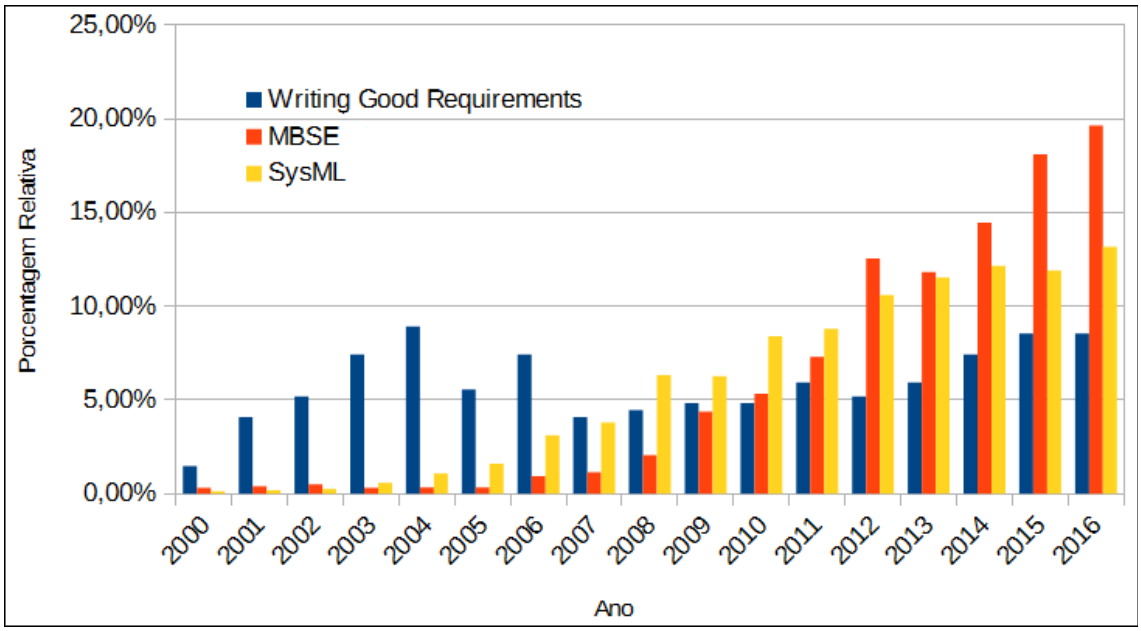
Já para os termos "MBSE" ou "SysML", novamente observa-se que só a partir do ano de 2006 começam as publicações de artigos contendo os termos. No total, nestes dezessete anos de periódicos do INCOSE, foram publicados 25 artigos contendo um destes dois termos. Mais ainda, a média, neste caso, ficou em 9,09% com um desvio padrão de 5,96%. A maior variância quando comparada com o caso de artigos publicados em simpósios do INCOSE, apresentado anteriormente no gráfico da Figura 3.2, pode ser explicada pela diferença na quantidade de artigos publicados. Enquanto nos dezessete anos de simpósios do INCOSE analisados a média de artigos por ano ficou em 137,29 artigos, para o caso de periódicos do INCOSE, no mesmo período, este valor cai para 27,71 artigos. De qualquer forma, esta análise reforça a constante presença dos termos "MBSE" ou "SysML" nos artigos publicados nos periódicos do INCOSE.

Finalizando esta seção, foram feitas algumas buscas no Google Acadêmico para identificar as tendências de pesquisas sendo realizadas em assuntos relacionados a esta tese de doutorado. Nestas buscas, foram pesquisados os

termos “MBSE” e “SysML”, como feito anteriormente para os artigos publicados nos dezessete últimos simpósios e períodos do INCOSE, bem como a frase “*writing good requirements*”, com o objetivo de identificar o esforço despendido em pesquisas sobre como melhorar a escrita de requisitos.

Na Figura 3.5 abaixo, mostra-se um gráfico com a porcentagem de correspondências para buscas no Google Acadêmico pelos termos “MBSE” ou “SysML”, ou pela frase “*writing good requirements*”, relativa ao número total de resultados ao longo dos mesmos dezessete anos considerados anteriormente, contendo o termo ou frase em questão. Ou seja, esta porcentagem é calculada como sendo a razão entre o número de correspondências de um dado ano contendo um dado termo ou frase pelo número total de correspondências nos dezessete anos considerados contendo o mesmo termo ou frase.

Figura 3.5 – Porcentagem Relativa de Resultados de Buscas no Google Acadêmico com Termos Relacionados à Tese.



Fonte: Produção do autor.

A construção utilizada no gráfico acima normaliza os números para cada um dos termos ou para a frase pesquisada e permite realçar se em algum dado ano algum termo ou a frase foi mais explorada do que nos demais anos. Ou

seja, a soma dos valores das porcentagens para um dado termo ou para a frase dos dezessete anos considerados é igual a 100%. A frase "*writing good requirements*" apareceu em 269 resultados de busca, com uma média de 5,88% e desvio padrão 1,96%. De posse destes valores, conclui-se que pesquisas visando melhorar a escrita de requisito vêm sendo constantemente elaboradas nestes dezessete últimos anos. Isto mostra, também, que uma solução sistemática e definitiva para este problema ainda não foi encontrada, o que abre espaço para novas pesquisas visando esta melhoria.

A busca pelo termo "MBSE" resultou em 3.202 correspondências entre os anos de 2000 e 2016, enquanto a busca pelo termo "SysML" resultou em 12.830 correspondências no mesmo período. Pelo gráfico da Figura 3.5 acima, nota-se que as buscas destes termos apresentaram crescentes correspondências ao longo dos anos, o que leva à conclusão de que estes termos vêm sendo cada vez mais explorados em todos os tipos de trabalhos acadêmicos.

Conforme visto no processo de alto nível de engenharia de sistemas apresentado no Capítulo 1, Figura 1.4, e no processo de alto nível proposto no Capítulo 1, Figura 1.6, para iniciar as atividades de definição da solução sistema necessita-se que as entradas do processo estejam disponíveis. Assume-se, portanto, no escopo desta Tese de Doutorado que os requisitos de *stakeholder* além de estarem definidos, já estejam estáveis o suficiente para serem utilizados na definição do sistema que os atenderá. Mais ainda, espera-se que as restrições de tecnologia a serem levadas em conta já estejam igualmente definidas.

Por outro lado, o método que será desenvolvido no decorrer desta Tese de Doutorado deve ser executado dentro do contexto de um sistema de gestão da qualidade. Em outras palavras, de acordo com as definições de 'escopo do produto' e 'escopo do projeto' fornecidas por PMI (2013), bem como suas distinções, considera-se esta Tese de Doutorado dentro do 'escopo de produto'. Ou seja, considera-se fora do escopo desta Tese de Doutorado

atividades de gestão e controle do sistema de interesse, conforme previstas no processo de alto nível de engenharia de sistemas apresentado no Capítulo 1, Figura 1.4. São exemplos destas atividades: a Gestão de Projeto, Gestão de Configuração, Gestão de Mudanças, Planejamento, etc. Mais ainda, exclui-se igualmente do escopo deste trabalho o ciclo de realimentação de verificação previsto no processo de alto nível de engenharia de sistemas apresentado no Capítulo 1, Figura 1.4.

3.2 Revisão da Literatura

Em um primeiro momento, esta revisão bibliográfica se baseou na pesquisa feita por Nicolás e Toval (2009), que elaboraram uma revisão sistemática da literatura relacionada à geração automática de especificações textuais de requisitos a partir de modelos de engenharia de software. Em seguida, utilizando a mesma busca utilizada por estes autores, apresentam-se alguns trabalhos mais recentes encontrados na base de dados da CAPES.

Nicolás e Toval (2009) motivaram a sua pesquisa no fato de que tanto modelos descritivos quanto requisitos textuais são importantes e necessários no processo de desenvolvimento de um sistema intensivo em software. Modelos são, em geral, expressivos, precisos e facilitam a elaboração e entendimento das especificações por parte dos times de desenvolvimento. Por outro lado, requisitos textuais em linguagem natural servem como um contrato entre o cliente e os desenvolvedores, facilitando a validação de requisitos pelo cliente e possibilitando a quantificação do tamanho do projeto.

Segundo Nicolás e Toval (2009), caso os requisitos textuais fossem gerados automaticamente a partir dos modelos descritivos, os seguintes benefícios seriam obtidos: redução do esforço de escrita de requisitos, melhora na completude da especificação de requisitos e automação da rastreabilidade entre os requisitos textuais e os modelos. A pesquisa destes autores abrangeu respostas às seguintes questões: (1) Qual relevância pode ser extraída da literatura com relação à geração de especificações de requisitos a partir de modelos de engenharia de software? (2) Quais técnicas foram

abordadas neste tema? (3) Quais abordagens levam em conta a derivação de requisitos a partir de modelos de linhas de produto de software?

Estes autores focaram sua pesquisa em transformações de modelos em requisitos ou em documentos de requisitos, e utilizaram buscas pelo padrão "*("from" OR "generation" OR "generating" OR "combination" OR "combining" OR "derivation" OR "deriving" OR "integration" OR "integrating") AND ("models" OR "specifications" OR "scenarios" OR "use cases" OR "features" OR "stories") AND ("documentation" OR "documents" OR "requirements")*" nas seguintes fontes de trabalhos:

- Biblioteca Digital IEEE;
- Biblioteca Digital ACM;
- Science@Direct;
- MetaPress (Kluwer + Springer);
- Wiley InterScience;
- Google Acadêmico.

A Tabela 3.1 abaixo contém os resultados obtidos pelos autores para estas buscas, em números de correspondências.

Tabela 3.1 – Correspondências obtidas por Nicolás e Toval (2009).

Fonte	Trabalhos Encontrados	Trabalhos Candidatos	Trabalhos Selecionados
Biblioteca Digital IEEE	50	2	2
Biblioteca Digital ACM	214	8	7
Science@Direct	45	2	2
MetaPress	87	2	2
Wiley InterScience	5	0	0
Google Acadêmico	394	12	10
Total	795	26	23

Fonte: Adaptada de Nicolás e Toval (2009).

Na tabela acima, tem-se o conjunto de trabalhos utilizados pelos autores na sua pesquisa. A coluna 'Trabalhos Encontrados' mostra a quantidade de

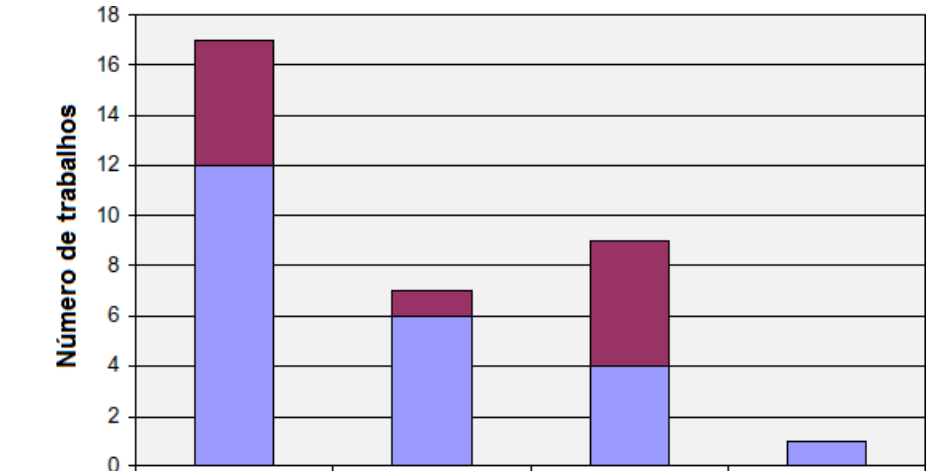
correspondências quando a busca pelo padrão acima exposto foi realizada. A coluna ‘Trabalhos Candidatos’ contém o número de trabalhos que passaram por um filtro de inclusão e exclusão definido pelos autores, de acordo com o objetivo de sua pesquisa. Finalmente, a coluna ‘Trabalhos Selecionados’, contém o número de trabalho que realmente foram escolhidos para realização da pesquisa proposta. As três exclusões que aconteceram foram justificadas pelos autores por serem trabalhos com versões mais recentes (duas delas) e por apresentar em um protótipo de ferramenta para base de dados de gestão de requisitos (uma delas), o que não era o foco da pesquisa. Depois destes filtros, os autores excluíram ainda trabalhos idênticos, encontrados em diferentes fontes, resultando em um total de 15 trabalhos. Apesar de não estar previsto na técnica de pesquisa utilizada pelos autores, estes decidiram acrescentar alguns trabalhos apontados nas bibliografias destes 15 trabalhos finais e que, de alguma maneira, se relacionavam com as perguntas a serem respondidas pela pesquisa, já apresentadas anteriormente. Com isso, os autores pesquisaram um total de 24 trabalhos.

De posse dos trabalhos escolhidos, Nicolás e Toval (2009) os classificaram de acordo com o modo de combinação e o escopo. Como modos de combinação, os autores utilizaram duas classificações: generativa e integrativa. Na primeira classificação, a abordagem propõe algoritmos, regras ou padrões para geração de requisitos textuais. Na segunda classificação, a geração é feita através de guias, sem regras definidas. Em ambos os modos, o requisito textual gerado pode ser expresso em linguagem natural (LN) ou em linguagem formal (LF). Já como escopo, os autores utilizaram duas classificações: requisito e documento. No primeiro, o foco está nos requisitos ou conjunto de requisitos, mas não em sua estruturação em uma documentação específica, o que já é o foco da segunda classificação.

Na Figura 3.6 abaixo, mostra-se um gráfico distribuindo os trabalhos escolhidos nestes dois critérios de classificação, diferenciando, ainda, resultados em linguagem natural de resultados em linguagem formal. Por

este gráfico, conclui-se que mais atenção foi despendida à geração de requisitos (24 trabalhos) em comparação com a geração de documentos de requisitos (10 trabalhos), enquanto abordagens generativas (23 trabalhos) foram mais numerosas do que abordagens integrativas (11 trabalhos). Em relação aos requisitos gerados propriamente ditos, nota-se que a maioria das abordagens (26 trabalhos) utiliza linguagem natural (LN), enquanto uma minoria (8 trabalhos) utiliza linguagem formal (LF). Os autores destacam, ainda, que o mesmo trabalho pode abordar ambos os tipos de linguagem.

Figura 3.6 – Número de Trabalhos por Escopo e Modo de Combinação.

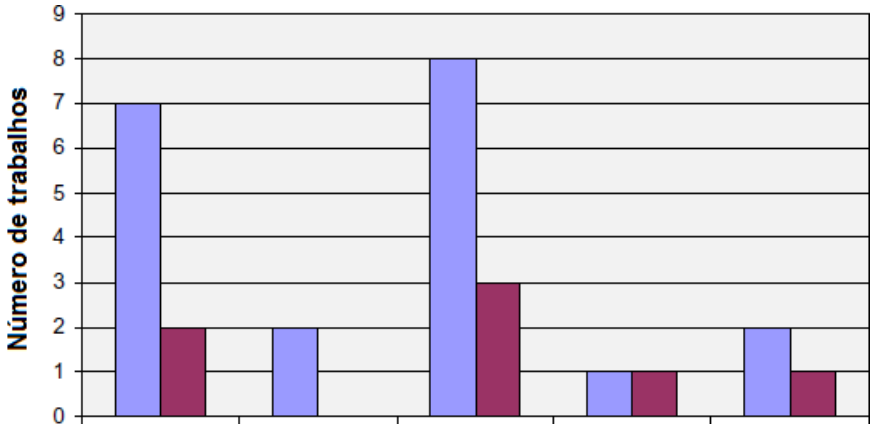


	Requisito (LN)	Requisito (LF)	Documento (LN)	Documento (LF)
Integrativo	5	1	5	0
Generativo	12	6	4	1

Fonte: Adaptada de Nicolás e Toval (2009).

Já na Figura 3.7 abaixo, mostra-se um gráfico distribuindo os trabalhos escolhidos de acordo com o escopo e modelo utilizado. Conclui-se, deste gráfico, que trabalhos relacionados a modelos orientados a objetivos, casos de uso e cenários representam a maioria dos esforços. Mais ainda, os autores destacam somente um trabalho utilizando a UML e que deixa de lado casos de uso e cenários. Segundo eles, um maior esforço deveria ser direcionado para esta linha de pesquisa, mais precisamente, para o uso de técnicas da UML mais voltadas para a engenharia de requisitos.

Figura 3.7 – Número de Trabalhos por Escopo e Modelo Inicial.



	Modelos Orientados a Objetivos	Modelos de Negócio	Casos de Uso & Cenários	UML (exceto casos de uso & cenários)	Interface com Usuário
Requisito	7	2	8	1	2
Documento	2	0	3	1	1

Fonte: Adaptada de Nicolás e Toval (2009).

Agora, utilizando o mesmo padrão de busca proposto por Nicolás e Toval (2009), apresentar-se-ão alguns trabalhos posteriores a 2009 encontrados na base de dados da CAPES e que satisfazem este padrão.

Guo et al. (2010) propõem uma abordagem direcionada por modelo para automatizar o processo de derivação de requisitos funcionais. Esta abordagem consiste na transformação de um modelo de casos de uso de produtos existentes em um modelo de requisitos funcionais de uma linha de produto. Para possibilitar tal transformação, os autores estenderam o padrão clássico de descrição de casos de uso para acomodar as definições de variabilidade de aplicações de uma linha de produto. Com isso, os autores desenvolveram uma transformação que tem como entrada modelos de casos de uso que usam o padrão estendido e tem como saída requisitos funcionais textuais para a linha de produto.

Hartong et al. (2013) propõem uma abordagem de análise verbo-substantivo de casos de uso para gerar requisitos de sistema e seus atributos de segurança associados. Segundo os autores, tradicionalmente os casos de uso não especificam requisitos de segurança nem seus atributos associados que um sistema deve apresentar para operar de forma segura. O algoritmo

proposto pelos autores interpreta os casos de uso e suas descrições para derivação automática destes requisitos e atributos associados.

Mahmood e Khatoon (2013) propõem uma abordagem para derivação de testes de sistema diretamente a partir de uma especificação, sem envolver detalhes funcionais. Nesta abordagem, casos de testes formalizados são gerados para cada fluxo do sistema identificado nos seus casos de uso, cujas pré-condições e pós-condições formam as condições dos testes. Segundo estes autores, as duas grandes contribuições desta abordagem são a antecipação dos testes e o refinamento das especificações. Apesar deste trabalho não estar diretamente relacionado com a geração automática de requisitos textuais a partir de modelo descritivo de um sistema, ele segue na linha da geração de informação textual, a partir de um modelo.

Rago et al. (2013) se concentram nos requisitos de qualidade (ou atributos de qualidade) de um sistema, pois, segundo estes autores, a entrega de um sistema que satisfaça estes requisitos é chave para o sucesso. Ainda segundo os autores, a detecção e a análise destes requisitos deveriam acontecer nas fases iniciais do projeto para reduzir possíveis riscos. Entretanto, estes requisitos geralmente se encontram espalhados em vários documentos e sua captura se torna difícil e custosa. Para este fim, os autores propõem uma abordagem para extrair e gerar requisitos de qualidade a partir de casos de uso do sistema. Esta abordagem possui duas etapas: extração de aspectos iniciais de cenários de casos de uso e derivação de candidatos a requisitos de qualidade a partir destes aspectos iniciais.

Couto et al. (2014) abordam especificações de software por meio de casos de uso que vêm sendo utilizadas com grande sucesso por engenheiros de software e analistas, principalmente por agrupar no mesmo modelo as expectativas dos diferentes *stakeholders* e as necessidades da equipe de desenvolvimento. Por este motivo, os autores propõem uma transformação das descrições de casos de uso escritas em uma linguagem natural controlada em uma ontologia expressa na Linguagem de Ontologia Web (ou

Web Ontology Language), que por sua vez pode ser vasculhada em busca de padrões para requisitos.

Os trabalhos apresentados até aqui corroboram a conclusão de Nicolás e Toval (2009), onde identifica-se o uso extensivo de casos de uso para derivação de requisitos. Entretanto, conforme os autores destacaram, pouco se fez para geração automática de requisitos a partir de um modelo descritivo do sistema de interesse utilizando outras construções disponíveis na UML e, por consequência, na SysML.

Já Lempia et al. (2016) estudaram o uso de diagramas visuais e padrões para a geração de requisitos completos e consistentes. Para tal, os autores se basearam em modelos do tipo *S*Model*, ou seja, modelos que são construídos a partir do metamodelo *S*Metamodel* (SCHINDEL, 2011). De acordo com os autores, para cada interação funcional, um conjunto de requisitos técnicos pode ser descoberto. Mais ainda, segundo os autores, cada comportamento identificado em uma dada interação funcional permite a escrita de um requisito textual. Para auxiliar esta escrita, os autores propõem o seguinte padrão:

Se [**GATILHO**],

o **OBJETO** deve **COMPORTAMENTO**

[(**DADO_SAÍDA** com o **ATRIBUTO_DADO_SAÍDA**(s))(s)]

[**RESTRIÇÃO_DESEMPENHO**]

[usando (**DADO_ENTRADA** com o **ATRIBUTO_DADO_ENTRADA**(s))(s)]

[enquanto no estado **ESTADO**].

No padrão acima, as palavras em maiúsculo e em negrito representam as lacunas do enunciado do requisito que devem ser preenchidas com informações extraídas do modelo descritivo do sistema de interesse. Entre colchetes, estão partes opcionais do enunciado do requisito.

O campo **GATILHO** identifica o evento de entrada que condiciona a função exposta pelo elemento identificado pelo campo **OBJETO**. Já o campo

COMPORTAMENTO identifica a função propriamente dita, exposta pelo elemento nas condições já identificadas. O campo **DADO_SAÍDA** identifica o que é produzido pelo elemento quando a função descrita é executada e o campo **ATRIBUTO_DADO_SAÍDA** identifica um atributo que qualifica este dado. Já o campo **RESTRIÇÃO_DESEMPENHO** qualifica a função em questão. O campo **DADO_ENTRADA** identifica o que a função necessita como entrada para gerar as suas saídas e o campo **ATRIBUTO_DADO_ENTRADA** identifica um atributo que qualifica este dado. Finalmente, o campo **ESTADO** identifica o estado em que o elemento se encontra quando expondo a referida função. Nota-se, ainda pelo padrão, que é prevista a existência de mais de um atributo qualificando um dado de saída e mais de um atributo qualificando um dado de entrada. Mais ainda, é prevista, também, a existência de mais de um dado de saída e de mais de um dado de entrada para a função em questão.

4 METODOLOGIA PROPOSTA

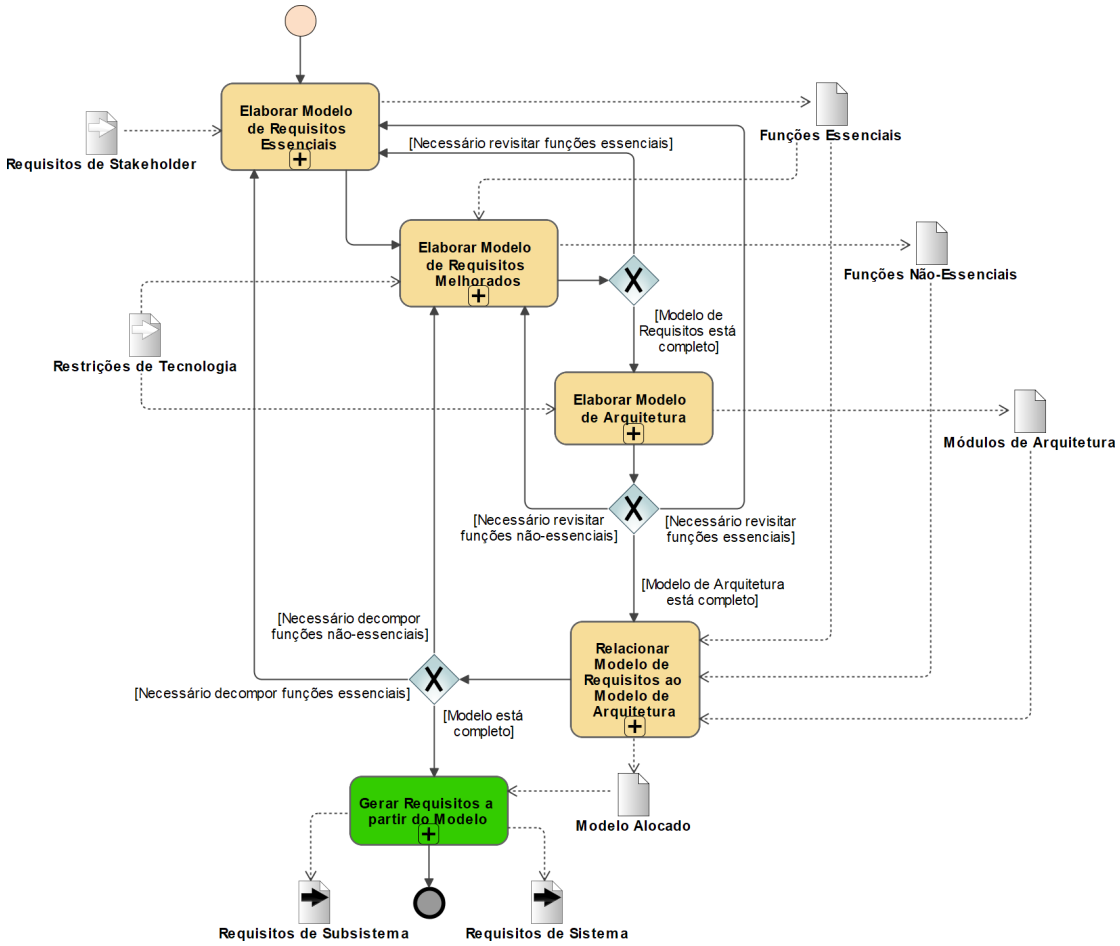
Conforme introduzido no Capítulo 1, esta Tese de Doutorado deve prover uma metodologia para a modelagem descritiva de um sistema de interesse utilizando a SysML e que, ao final, possibilite a extração automática dos enunciados dos requisitos de sistema.

Este capítulo mostra, portanto, o desenvolvimento desta metodologia, que está baseada no método HHP (HATLEY et al., 2000), que implementa o processo elaborado pelos mesmos autores e chamado de PSARE (*Process for System Architecture and Requirements Engineering*). Uma discussão sobre os motivos pelas escolhas deste método base e da SysML como linguagem de modelagem pode ser encontrada no Capítulo 6. Mais ainda, no Apêndice B encontra-se uma descrição detalhada do método HHP.

Na Figura 4.1 abaixo, expande-se o processo de alto nível proposto no Capítulo 1, Seção 1.3, Figura 1.6, e apresentam-se as atividades do processo implementado pela metodologia aqui proposta, baseado no processo PSARE de Hatley et al. (2000). Na mencionada figura, os retângulos com cantos arredondados representam atividades do processo que, devido ao símbolo ‘+’ na sua parte inferior, podem ser vistos como subprocessos, a serem detalhados posteriormente. Os losangos com um ‘X’ dentro representam pontos de decisão do processo, onde somente um dos fluxos de saída acontecerá, de acordo com a condição a ele atribuída e representada pelo texto entre colchetes na sua proximidade. Os ícones representando uma folha de papel representam dados do processo. Alguns deles possuem uma seta branca no seu interior, representando dados de entrada. Já outros, possuem uma seta preta internamente, indicando que se trata de um dado de saída. Aqueles sem nenhum detalhe extra, representam dados sem nenhuma informação sobre sua orientação. Linhas cheias direcionadas representam fluxos de atividades do processo sendo descrito, enquanto linhas pontilhadas direcionadas representam fluxos de dados deste processo. O círculo preenchido em uma cor rósea no topo da figura representa o ponto de início do processo. Já o círculo com bordas mais

grossas e preenchido de cinza na parte de baixo da figura representa o ponto final do processo.

Figura 4.1 – Processo Implementado pela Metodologia Proposta



Fonte: Produção do autor.

Antes de mais nada, ressalta-se que o processo consiste, basicamente, de separar as funções que o sistema deve realizar, representadas pelo seu modelo de requisitos, de como o sistema deve realizar tais funções, representado pelo seu modelo de arquitetura. Após a elaboração destes modelos, o engenheiro de sistemas deve relacioná-los, basicamente alocando funções aos elementos de arquitetura, chamados de módulos de arquitetura. Feito isso, basta executar a geração automática dos requisitos de sistema a partir do modelo final, atividade esta destacada em verde na Figura 4.1 acima.

Hatley et al. (2000) abordam constantemente o fato do processo PSARE não ser sequencial. Ou seja, a elaboração dos modelos é iterativa e um modelo certamente será revisitado durante a elaboração do outro e vice-versa. Mais ainda, os autores reforçam que estes modelos são elaborados em vários níveis de abstração, onde iterações são realizadas conforme o engenheiro de sistema acrescenta detalhes sobre funções e a arquitetura do sistema. Tal iteratividade é representada na Figura 4.1 acima através dos diferentes nós de decisão, que remetem a uma atividade anterior sempre que necessário. Daqui em diante, o processo será descrito de forma linear e sequencial para facilitar sua compreensão.

O processo se inicia, de acordo com a Figura 4.1, pela elaboração do Modelo de Requisitos Essenciais. A partir dos requisitos de *stakeholder*, o engenheiro de sistemas identifica as funções essenciais que o sistema deve realizar. Tais funções representam o comportamento que o sistema deve apresentar para alcançar a missão a ele atribuída, independentemente da tecnologia disponível tanto para o seu desenvolvimento, como aquela utilizada no ambiente onde ele está inserido. Entretanto, levar em conta restrições de tecnologia é necessário para especificar um sistema factível. É aqui que entra o Modelo de Requisitos Melhorados, com o qual o engenheiro de sistemas identifica funções não-essenciais, que permeiam as funções essenciais para acoplá-las ao mundo real. Estas funções não-essenciais são classificadas em quatro tipos: funções de entrada, funções de saída, funções de interface com o usuário e, finalmente, funções de suporte.

As funções de entrada possuem o objetivo de transformar as entradas reais que chegam na fronteira do sistema sendo modelado nas entradas lógicas esperadas pelas funções essenciais. As funções de saída, por sua vez, transformam as saídas lógicas produzidas pelas funções essenciais nas saídas reais que cruzam a fronteira do sistema. As funções de interface com o usuário tem como objetivo traduzir os dados lógicos produzidos pelas funções essenciais nos dados reais trocados com os usuários do sistema. Por último, as funções de suporte existem para tornar possível o correto

funcionamento das funções essenciais, levando-se em conta as tecnologias existentes para construção do sistema e aquelas do seu ambiente operacional. São exemplos de funções de suporte aquelas relacionadas a redundância, manutenção, etc.

Depois de estabilizar o modelo de requisitos, passa-se, então, para o modelo de arquitetura. Através deste modelo, o engenheiro de sistemas define as partes constituintes do sistema, chamadas de módulos de arquitetura, e como estes módulos estão interligados entre si para trocar material, energia ou informação. Seguindo na mesma linha do modelo de requisitos, os módulos de arquitetura são divididos nos seguintes cinco tipos: módulos essenciais, módulos de entrada, módulos de saída, módulos de interface com usuário e, finalmente, módulos de suporte. Cada um destes tipos representa o conjunto de módulos de arquitetura que alocam as correspondentes funções do mesmo tipo, ou seja, módulos essenciais alocam funções essenciais, módulos de entrada alocam funções de entrada, módulos de saída alocam funções de saída e assim por diante.

Terminados os modelos de requisitos e de arquitetura, o engenheiro de sistema aloca todas as funções identificadas aos módulos de arquitetura que as realizarão, criando assim um modelo alocado. No escopo desta Tese de Doutorado, aplicam-se, então, regras de transformação a este modelo alocado para extrair os requisitos de sistema nele definidos.

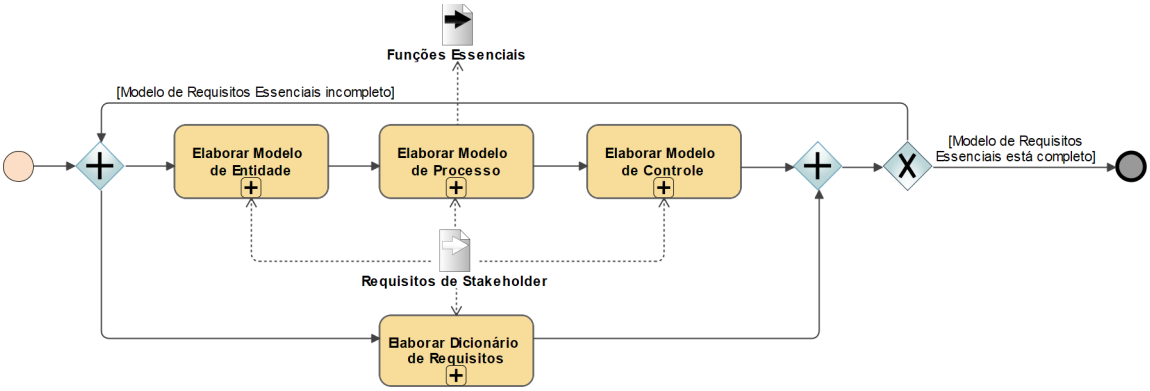
Abaixo, são apresentadas subseções dedicadas a cada uma das atividades do processo apresentado na Figura 4.1.

4.1 Elaborar Modelo de Requisitos Essenciais

Este modelo permite ao engenheiro de sistemas estabelecer o núcleo de requisitos independentes de tecnologia, ou seja, as funções precípuas do sistema, extraídas a partir dos requisitos de *stakeholder*. Ele é composto por três submodelos: modelo de entidade, modelo de processo e modelo de controle. Estes modelos são suplementados por um dicionário de requisitos, sendo este o repositório único das definições de todos os elementos usados

nos modelos. Novamente, não existe prescrição estabelecida para a sequência de elaboração destes modelos, sendo eles criados de maneira interativa. Na Figura 4.2 abaixo, apresentam-se as atividades previstas nesta etapa do processo, bem como suas entradas e saídas esperadas.

Figura 4.2 – Elaborar Modelo de Requisitos Essenciais



Fonte: Produção do autor.

Na figura acima, os losangos com um '+' dentro representam pontos de paralelismo de fluxos de atividades, onde os fluxos de saída acontecem simultaneamente após todos os fluxos de entrada terem alcançado um dado ponto deste tipo. Estes pontos juntamente ao ponto de decisão presente no diagrama acima representam a iteratividade inerente a esta etapa do processo, cujas atividades estão descritas abaixo, em subseções dedicadas.

4.1.1 Elaborar Modelo de Entidade

Este modelo permite ao engenheiro de sistemas descrever os dados (material, energia ou informação) do domínio de negócio do sistema de interesse. O nível de abstração utilizado aqui permite tornar transparante a tecnologia ou implementação envolvida por trás destes dados e mostra, portanto, o que o sistema modelado deverá manipular na sua essência. Mais ainda, o nível semântico deste modelo permite a troca de informações com os *stakeholders*, que serão capazes de compreender o que está presente neste modelo.

4.1.2 Elaborar Modelo de Processo

Este modelo permite ao engenheiro de sistemas capturar as capacidades requeridas de processamento que o sistema deve apresentar em termos de funções e dados trocados entre elas. Os elementos básicos de modelagem são: as funções e seus respectivos parâmetros de entrada e saída, as entidades externas com as quais o sistema troca material, energia e informação, e os repositórios de dados internos ao sistema.

A principal atividade exercida pelo engenheiro de sistemas ao elaborar este modelo é a decomposição funcional. Partindo da missão do sistema, ou seja, da sua função precípua, o engenheiro de sistemas inicia seu desdobramento em subfunções, que por sua vez são igualmente desdobradas até que se atinja o nível desejado de abstração, aqui chamadas de funções terminais, onde a transformação das suas entradas nas suas saídas seja passível de ser expressada de forma livre de ambiguidade.

4.1.3 Elaborar Modelo de Controle

Este modelo permite ao engenheiro de sistemas capturar as capacidades requeridas de comportamento que este sistema deve apresentar em termos dos seus vários estados e modos de operação, determinando quais funções estão ativas em cada um deles, bem como os eventos que fazem o sistema mudar de estado e/ou modo operacional.

Pelo método de engenharia de sistemas proposto por Loureiro (1999), durante a análise funcional do sistema, devem-se identificar os estados do sistema e seus modos operacionais. Segundo o autor, estado define um conjunto de circunstâncias caracterizando o sistema em um dado momento. Já os modos operacionais, ainda segundo o autor, agrupam funcionalidades do sistema, estando este em um determinado estado.

4.1.4 Elaborar Dicionário de Requisitos

O dicionário de requisitos é, basicamente, uma base de dados de todos os elementos usados no modelo de requisitos, cada um com sua descrição

estrutural, ou seja, como o elemento é decomposto em subelementos, seu significado e outros detalhes. Este dicionário define os parâmetros das funções, eventos das transições de estados e modos operacionais, repositório de dados, etc. Elementos primitivos, ou seja, aqueles que não são decompostos adiante, têm suas propriedades definidas, tais como unidade, precisão, resolução, taxa de atualização, intervalo de valores possíveis, etc.

4.2 Elaborar Modelo de Requisitos Melhorados

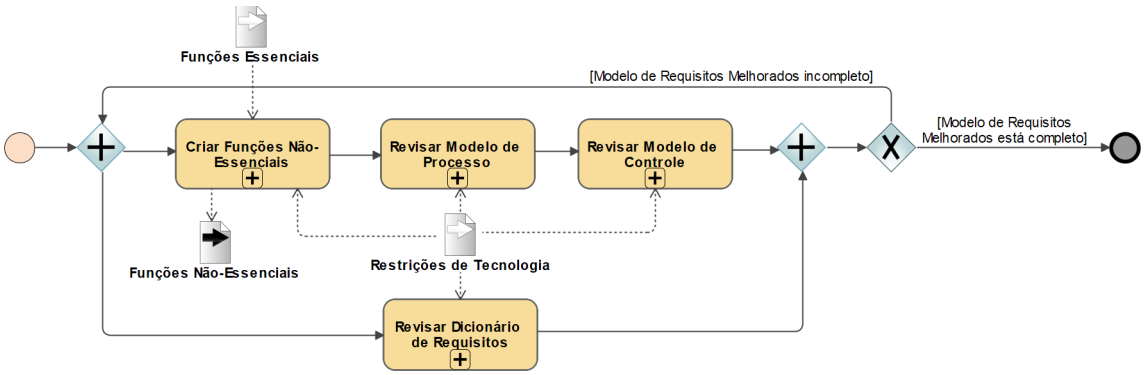
Quando o modelo de requisitos essenciais foi construído, a tecnologia utilizada no sistema e no ambiente ao seu redor foi desconsiderada para garantir requisitos livres de características implementacionais. Parâmetros de entrada e saída das funções do sistema eram, na verdade, parâmetros lógicos, não correspondendo aos parâmetros físicos (i.e. reais) que serão efetivamente tratados pelo sistema. Na sua essência, o sistema não falha e não necessita de nada especial para realizar sua missão.

Ao construir o modelo de requisitos melhorados, o engenheiro de sistemas passa a levar em consideração o mundo real e a tecnologia disponível para permitir o acoplamento entre o sistema e o ambiente ao redor. Melhorar o modelo de requisitos significa, portanto, transformar parâmetros lógicos em parâmetros físicos. No mundo real, sistemas falham e funções de suporte para redundância, manutenção, detecção, correção e proteção de falhas deverão ser incorporadas.

Conforme novas funções não-essenciais são incorporadas ao modelo, novos parâmetros e eventos de controle são determinados, significando que o engenheiro de sistemas deverá revisitar e melhorar o modelo de processo, o modelo de controle e o dicionário de requisitos elaborados até o momento. Especialmente com a incorporação de funções de suporte, novas entidades externas podem ser identificadas, tais como dispositivos de testes ou ferramentas de diagnósticos. A decomposição funcional feita até agora para as funções essenciais deverá ser igualmente feita para as funções não-essenciais identificadas. Na Figura 4.3 abaixo, apresentam-se as atividades

previstas nesta etapa do processo, bem como suas entradas e saídas esperadas.

Figura 4.3 – Elaborar Modelo de Requisitos Melhorados

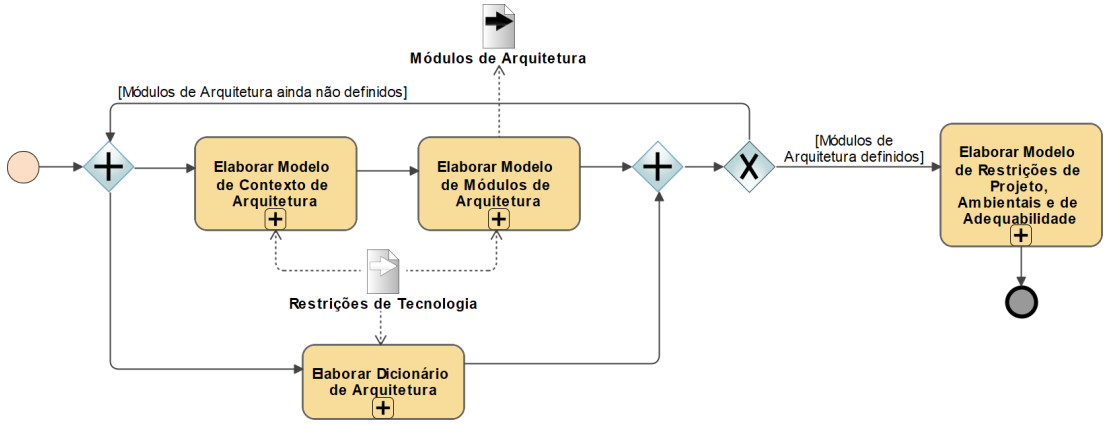


Fonte: Produção do autor.

4.3 Elaborar Modelo de Arquitetura

Este modelo permite ao engenheiro de sistemas mostrar a realidade física do sistema sendo modelado: componentes, interfaces, etc. A ideia central deste modelo é permitir o agrupamento dos requisitos em blocos passíveis de serem construídos. Ele pode ser dividido em dois submodelos: modelo de contexto de arquitetura e modelo de módulos de arquitetura. Novamente, não existe prescrição estabelecida para a sequência de elaboração destes modelos, sendo eles criados de maneira interativa. Na Figura 4.4 abaixo, apresentam-se as atividades previstas nesta etapa do processo, bem como suas entradas e saídas esperadas.

Figura 4.4 – Elaborar Modelo de Arquitetura



Fonte: Produção do autor.

Novamente nota-se aqui a iteratividade inerente a esta etapa do processo, cujas atividades estão descritas abaixo, em subseções dedicadas.

4.3.1 Elaborar Modelo de Contexto de Arquitetura

Este modelo permite ao engenheiro de sistemas definir o ambiente físico do sistema, ou seja, as entidades externas e os fluxos de arquitetura trocados entre elas e o sistema. Mais ainda, é possível mostrar eventuais trocas de mensagens de arquitetura. A diferença entre fluxos e mensagens está, basicamente, na ênfase por eles colocada. Um fluxo representa material, energia ou informação sendo trocado entre dois elementos de arquitetura, sem ênfase em qual deles realmente causou o fluxo propriamente dito. Mensagens, por outro lado, estabelecem um padrão ativo/passivo, onde um elemento de arquitetura (o ativo) envia a mensagem para outro elemento de arquitetura (o passivo). Material, energia ou informação pode, eventualmente, ser transmitido na mesma direção da mensagem (um elemento postando um fluxo para o outro), na direção contrária da mensagem (um elemento requisitando um fluxo para o outro) ou em ambas as direções (um elemento postando e requisitando fluxos do outro).

Neste modelo, o engenheiro de sistemas representa, também, as diversas interconexões físicas entre o sistema e as entidades externas. Uma interconexão representa a interface pela qual os fluxos de material, energia, informação ou mensagens são trocados entre elementos de arquitetura.

4.3.2 Elaborar Modelo de Módulos de Arquitetura

Este modelo permite ao engenheiro de sistemas capturar a estrutura do sistema em termos dos módulos de arquitetura que o compõem, de suas especificações e de seus relacionamentos, estes podendo ser de três tipos distintos: comunicação, meios de comunicação e herança. Os dois primeiros são similares àqueles utilizados no modelo de contexto de arquitetura, ou seja, fluxos e mensagens de arquitetura trocados entre os módulos de arquitetura, e interconexões (i.e. interfaces) pelas quais estas trocas acontecem. Já os relacionamentos de herança tem como objetivo agrupar

em módulos de arquitetura abstratos, características e funcionalidades comuns presentes em diferentes módulos de arquitetura.

Este modelo pode apresentar vários níveis de decomposição de arquitetura, conforme o sistema é desdobrado em módulos menores. Não existe regra para quando se deve parar esta decomposição física. A ideia básica é que, ao final desta atividade, os módulos de arquitetura de mais baixo nível possam ser enviados para a fase de aquisição, ou seja, serem comprados de terceiros como componentes prontos para uso ou serem desenvolvidos, tanto internamente quanto externamente à organização.

4.3.3 Elaborar Dicionário de Arquitetura

Um outro importante artefato que deve ser produzido pelo engenheiro de sistemas é o dicionário de arquitetura. Similar ao dicionário de requisitos, este dicionário é o repositório único dos diferentes tipos utilizados no modelo de arquitetura.

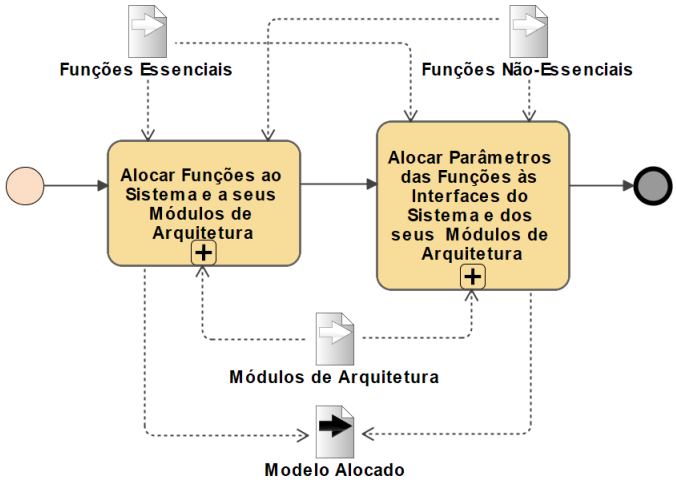
4.3.4 Elaborar Modelo de Restrições de Projeto, Ambientais e de Adequabilidade

Para esta atividade, o engenheiro de sistemas adicionará ao modelo de arquitetura, estruturas para representar requisitos não-funcionais seguindo os três correspondentes padrões definidos por Carson (2015), vistos no Capítulo 2, Seção 2.1.4: requisitos de projeto, requisitos ambientais e requisitos de adequabilidade.

4.4 Relacionar Modelo de Requisitos ao Modelo de Arquitetura

Nesta etapa do processo, o engenheiro de sistemas deve relacionar os dois modelos elaborados até o momento. Este relacionamento se dá através de duas atividades, representadas na Figura 4.5 abaixo.

Figura 4.5 – Relacionar Modelo de Requisitos ao Modelo de Arquitetura



Fonte: Produção do autor.

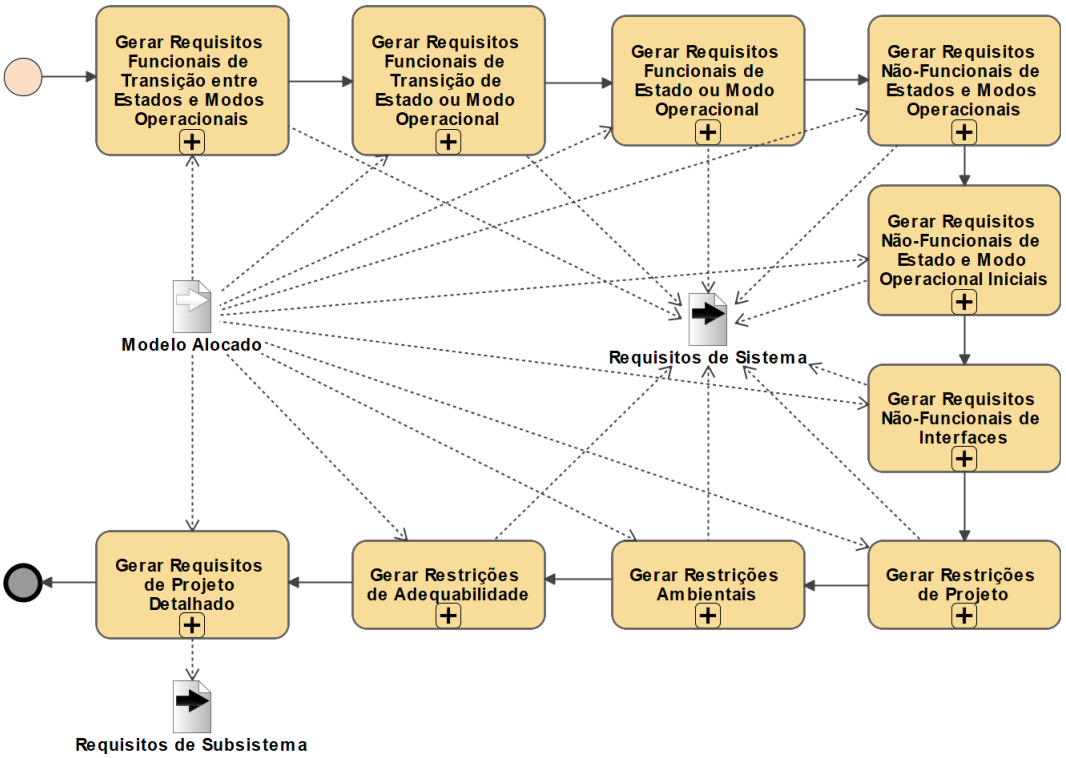
A primeira atividade consiste em alocar cada função identificada no modelo de requisitos (essencial ou não-essencial) ao próprio sistema e a um de seus módulos de arquitetura. Caso uma função não seja passível de uma alocação unívoca a um módulo de arquitetura, ou seja, não seja possível alocar a dada função a um único módulo de arquitetura, deve-se continuar seu desdobramento funcional até que se atinja um nível de abstração onde isto se torna possível. Estas novas subfunções de projeto detalhado não precisam ser alocadas ao sistema, pois a função de nível de abstração mais alto, que deu origem a estas subfunções, já foi devidamente alocada ao sistema.

Já a segunda atividade consiste em dar observabilidade às funções do modelo de requisitos. Para tal, deve-se alocar os parâmetros de cada função identificada (essencial ou não-essencial) a uma interface do próprio sistema e do seu módulo de arquitetura que aloca a correspondente função.

4.5 Gerar Requisitos a partir do Modelo

Finalmente, de posse de um modelo alocado, é possível iniciar a extração automática dos requisitos de sistema. Dentro da divisão tradicional de requisitos em funcionais e não-funcionais, o processo aqui descrito prevê a geração de diversos conjuntos de requisitos, cada um deles identificado por uma atividade na Figura 4.6 abaixo.

Figura 4.6 – Gerar Requisitos a partir do Modelo



Fonte: Produção do autor.

É importante ressaltar que a extração dos requisitos funcionais terá como ponto de partida o modelo de controle criado durante a elaboração do modelo de requisitos, pois é ali que se encontram todas as definições sobre estados, modos operacionais e ativações de funções.

Já a extração dos requisitos não-funcionais terá como principal ponto de partida o modelo de arquitetura. Entretanto, alguns requisitos não-funcionais serão extraídos do modelo de controle, para garantir a presença dos estados e modos operacionais ali identificados.

Cada uma das atividades desta etapa do processo identificada na figura acima gera um conjunto de requisitos de sistema. Mais especificamente, devido à etapa de projeto detalhado, podem-se gerar, também, requisitos de subsistema. Por motivos de facilitar a compreensão, estas atividades serão descritas mais adiante neste capítulo, na Seção 4.7, após o detalhamento de como se criar os modelos aqui identificados com o uso da SysML.

4.6 O Uso da SysML na Construção dos Modelos Propostos

Nesta seção, diagramas, elementos e relacionamentos especificados na SysML serão utilizados para construção dos modelos esperados pelo processo definido na Figura 4.1. O objetivo aqui não é apresentar a SysML por completo, mas somente aquilo que é necessário para o desenvolvimento da metodologia proposta. Para uma apresentação completa e dedicada da linguagem, referencia-se Friedenthal et al. (2014).

Os nomes dos diagramas, elementos e relacionamentos da SysML, quando escritos por extenso, estarão apresentados com suas inciais em letras maiúsculas para facilitar sua identificação. Mais ainda, diagramas da SysML contêm um cabeçalho na parte superior à esquerda, cujo conteúdo está definido na especificação da SysML como sendo: “<TipoDiagrama> [TipoElemento] <NomeElemento> [NomeDiagrama]”, onde o primeiro campo contém um mnemônico para o tipo do diagrama, cujos valores possíveis estão definidos na especificação da SysML, o segundo campo identifica o tipo do elemento do modelo que encapsula o diagrama, cujos valores possíveis estão igualmente definidos na especificação da SysML, o terceiro campo contém o nome do elemento do modelo que encapsula o diagrama e, finalmente, o quarto campo contém o nome do diagrama em si. Somente os dois últimos campos possuem valores livres e que são definidos pelo engenheiro de sistemas no momento da criação do diagrama.

Por se tratar de uma metodologia baseada no método HHP, no Apêndice C encontra-se a ontologia da metodologia proposta, onde mostram-se os principais termos do método HHP e como estes termos estão relacionados entre si. Mais além, esta ontologia define quais elementos da SysML serão utilizados para representar estes termos e relacionamentos.

4.6.1 Perfil SysML para a Metodologia Proposta

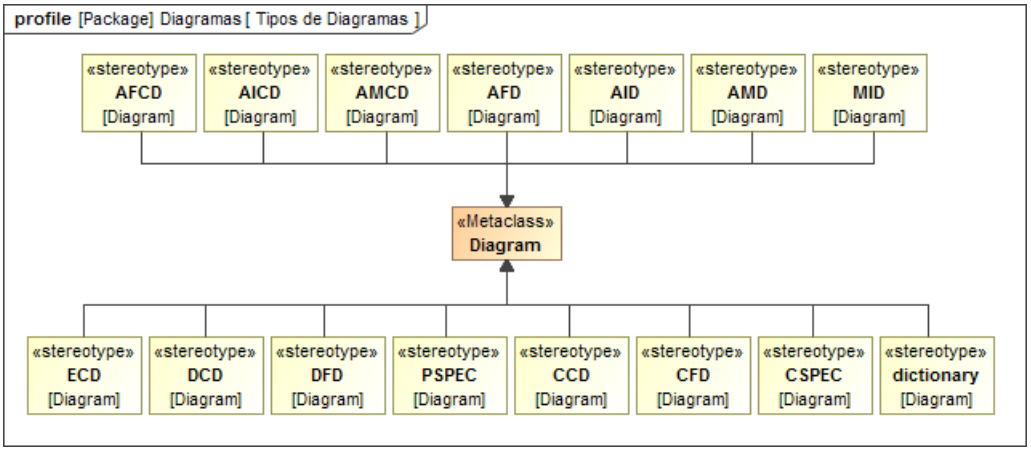
Conforme mencionado no Capítulo 2, Seção 2.3, o engenheiro de sistemas pode estender a SysML, através da criação e uso de perfis. Desta maneira, tem-se a capacidade de adicionar à linguagem novos tipos de elementos ou

diagramas específicos à metodologia utilizada para modelar um sistema. Estes novos tipos são representados por estereótipos dentro de um perfil, criados pelo próprio engenheiro de sistemas, e que devem estender tipos básicos de elementos ou diagramas da SysML (e.g.: Classe, Atividade, Diagrama, Estado, etc).

Com o objetivo de facilitar a transição do método HHP para SysML e de adicionar os novos conceitos, propõe-se, aqui, um perfil composto de um conjunto de estereótipos que tipificarão diagramas e elementos da SysML, de acordo com a nomenclatura utilizada no método HHP e o que é esperado pelo processo definido na Figura 4.1.

Na Figura 4.7 abaixo, mostra-se o conjunto de estereótipos criados para tipificar os diagramas da SysML de acordo com o método HHP. Para cada tipo de diagrama do método HHP foi criado um estereótipo com o mesmo nome e que estende o tipo básico da SysML para diagramas, representado no centro da figura. As caixas com o texto "<<*stereotype*>>" acima do nome representam estes estereótipos. As linhas direcionadas representam o relacionamento de extensão, onde um estereótipo estende o tipo básico da SysML para diagramas.

Figura 4.7 – Diagramas do Método HHP em SysML

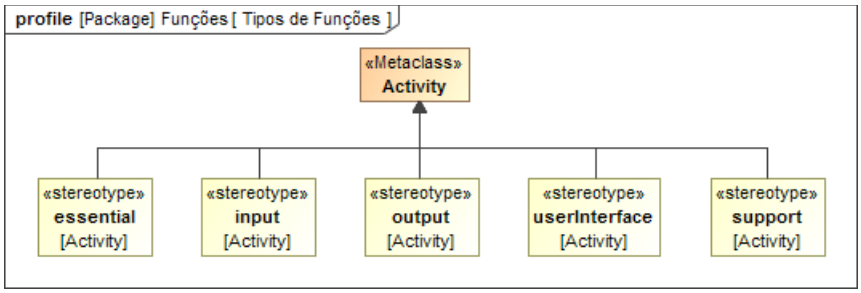


Fonte: Produção do Autor.

De posse deste conjunto de estereótipos, ao criar um diagrama da SysML, o engenheiro de sistemas poderá aplicá-los para identificar o tipo de diagrama em termos do método HHP.

De acordo como método HHP, as funções identificadas no modelo de requisitos podem ser classificadas de acordo com cinco tipos: funções essenciais, funções de entrada, funções de saída, funções de interface com usuário e funções de suporte. Por outro lado, de acordo com o uso da SysML proposto nesta seção, as funções serão representadas pelo elemento Atividade da SysML. Desta maneira, na Figura 4.8 abaixo, mostra-se o conjunto de estereótipos criados para tipificar este tipo de elemento da SysML de acordo com a divisão de funções do método HHP.

Figura 4.8 – Funções do Método HHP em SysML



Fonte: Produção do Autor.

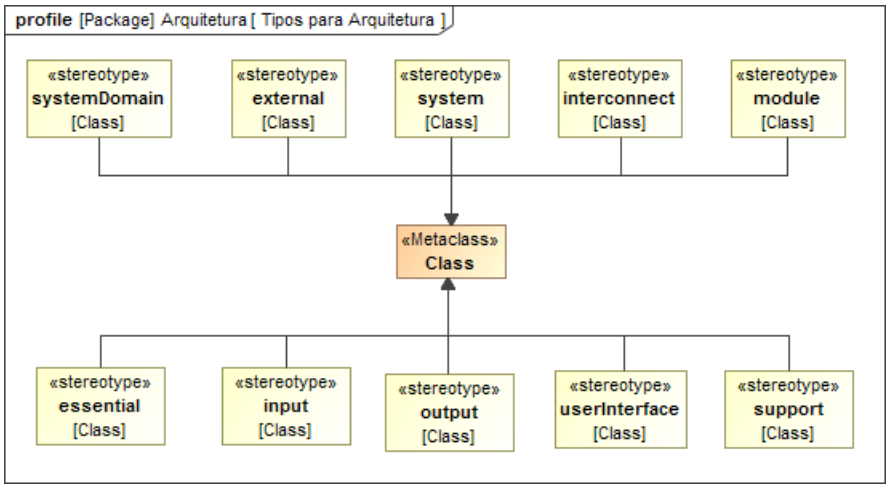
Para cada tipo de função do método HHP criou-se um estereótipo com o mesmo nome e que estende o tipo básico da SysML para Atividades, representado no centro superior da figura acima. Os nomes dos estereótipos estão em Inglês para facilitar a referência cruzada com Hatley et al. (2000). Ou seja, o estereótipo <<*essential*>> tipificará Atividades representando funções essenciais do sistema. O estereótipo <<*input*>> tipificará Atividades representando funções de entrada do sistema. Já o estereótipo <<*output*>> tipificará Atividades representando funções de saída do sistema. O estereótipo <<*userInterface*>> tipificará Atividades representando funções de interface com usuário do sistema. Finalmente, o estereótipo <<*support*>> tipificará Atividades representando funções de suporte do sistema.

De posse deste conjunto de estereótipos, ao criar um elemento Atividade da SysML, o engenheiro de sistemas deverá aplicar um destes para identificar o tipo da função que ela representa em termos do método HHP.

Para a parte da arquitetura do sistema, o engenheiro de sistemas utilizará de forma intensiva o elemento Bloco da SysML. Com o objetivo de diferenciar

seu uso dentro do modelo de arquitetura dos demais usos encontrados em todos os modelos do sistema previstos no processo apresentado na Figura 4.1, e que serão abordados nas sessões subsequentes, na Figura 4.9 abaixo, mostra-se o conjunto de estereótipos, criados de acordo com a arquitetura do sistema proposta no método HHP.

Figura 4.9 – Tipos para Arquitetura do Método HHP em SysML



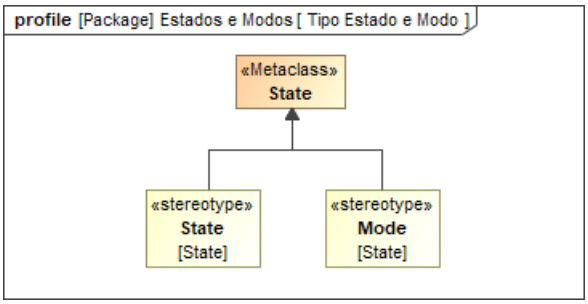
Fonte: Produção do Autor.

Para cada elemento de arquitetura do método HHP criou-se um estereótipo que estende o tipo básico da SysML para Classes, representado no centro da figura. Os nomes dos estereótipos estão em Inglês para facilitar a referência cruzada com Hatley et al. (2000). Ou seja, o estereótipo <<*systemDomain*>> tipifica o Bloco representando o domínio do sistema sendo modelado. O estereótipo <<*external*>> tipifica um Bloco representando uma entidade externa ao sistema de interesse. O estereótipo <<*system*>> tipifica o Bloco representando o sistema sendo modelado propriamente dito. O estereótipo <<*interconnect*>> tipifica eventuais Blocos ou Blocos de Interface representando interconexões de arquitetura. O estereótipo <<*module*>> tipifica Blocos representando módulos de arquitetura do sistema. O estereótipo <<*essential*>> tipifica Blocos representando módulos essenciais do sistema. O estereótipo <<*input*>> tipifica Blocos representando módulos de entrada do sistema. O estereótipo <<*output*>> tipifica Blocos representando módulos de saída do sistema. O estereótipo <<*userInterface*>> tipifica Blocos representando módulos de interface com

usuário do sistema. Finalmente, o estereótipo <<*support*>> tipifica Blocos representando módulos de suporte do sistema.

Para se diferenciar os estados do sistema dos seus modos operacionais, foram acrescentados estereótipos dedicados ao perfil SysML sendo utilizado. Na Figura 4.10 abaixo, mostram-se estes novos estereótipos.

Figura 4.10 – Estados e Modos Operacionais em SysML



Fonte: Produção do Autor.

Na figura acima, notem-se dois estereótipos: um para representar um estado do sistema e outro para representar um modo operacional deste sistema. Os nomes dos estereótipos estão em Inglês para manter o padrão utilizado até agora no perfil SysML, ou seja, o estereótipo <<State>> representará estados do sistema e o estereótipo <<Mode>> representará seus modos operacionais.

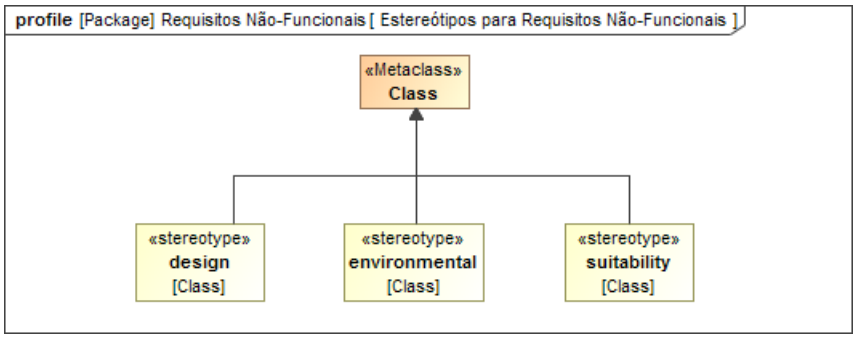
Como visto na Seção 4.3.4, gerar-se-ão três tipos de requisitos não-funcionais, seguindo os correspondentes padrões de enunciado definidos por Carson (2015): restrições de projeto, restrições ambientais e restrições de adequabilidade.

Estes tipos de requisitos não-funcionais são, na verdade, características que o sistema deve apresentar. Tais características serão modeladas utilizando elementos do tipo Propriedade de Restrição da SysML, que podem ser criados para os Blocos representando o sistema, módulos de arquitetura e interconexões (i.e. interfaces). Estas Propriedades de Restrição da SysML devem ser tipificadas por Blocos de Restrição da SysML.

Na Figura 4.11 abaixo, mostram-se, portanto, três estereótipos adicionais utilizados para identificar os diferentes Blocos de Restrição da SysML que

representarão os três tipos distintos de requisitos não-funcionais definidos por Carson (2015).

Figura 4.11 – Estereótipos para Requisitos Não-Funcionais

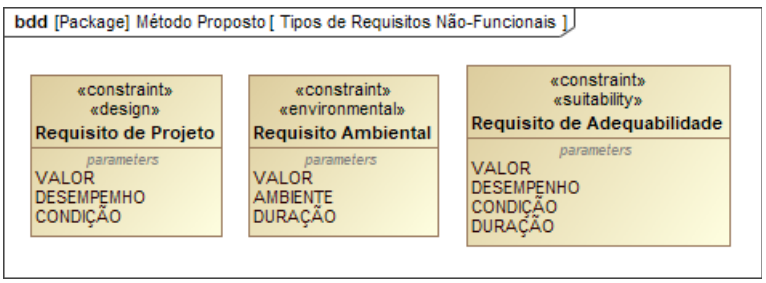


Fonte: Produção do Autor.

Para cada tipo de requisitos não-funcionais definidos por Carson (2015) criou-se um estereótipo que estende o tipo básico da SysML para Classes, representado no centro da figura acima. Os nomes dos estereótipos estão em Inglês para facilitar a referência cruzada com Carson (2015). Ou seja, o estereótipo <<*design*>> tipifica o Bloco de Restrição da SysML representando o tipo de requisito não-funcional para restrições de projeto. O estereótipo <<*environmental*>> tipifica o Bloco de Restrição da SysML representando o tipo de requisito não-funcional para restrições ambientais. Finalmente, o estereótipo <<*suitability*>> tipifica o Bloco de Restrição da SysML representando o tipo de requisito não-funcional para restrições de adequabilidade.

Na Figura 4.12 abaixo, mostra-se os Blocos de Restrição criados para representar os três tipos de requisitos não-funcionais de Carson (2015).

Figura 4.12 – Tipos de Requisitos Não-Funcionais de acordo com Carson (2015).



Fonte: Produção do Autor.

Na figura acima, nota-se que a cada Bloco estão aplicados dois estereótipos. Um deles é o estereótipo <<*constraint*>>, que identifica que o Bloco é, na verdade, um Bloco de Restrição da SysML. O segundo estereótipo aplicado é um dos estereótipos apresentados anteriormente na Figura 4.11 e que identifica o tipo de requisito não-funcional, de acordo com Carson (2015), que o correspondente Bloco de Restrição representa.

De acordo com a especificação da SysML, linhas horizontais em Blocos delimitam seus compartimentos, que organizam como certas informações são apresentadas em diagramas. O compartimento visto no diagrama acima está nomeado como "*parameters*". Este nome, definido na especificação da SysML, identifica o compartimento contendo elementos do tipo Parâmetro de Restrição do correspondente Bloco de Restrição. No caso da metodologia aqui proposta, estes parâmetros corresponderão às lacunas do texto do correspondente requisito não-funcional proposto por Carson (2015). A Seção 4.7 trará as regras de como gerar os correspondentes enunciados, utilizando estes parâmetros.

A partir deste ponto, esta seção será subdividida de acordo com os modelos previstos no processo apresentado na Figura 4.1: modelo de requisitos essenciais, modelo de requisitos melhorados e modelo de arquitetura. Em seguida, uma subseção dedicada ao relacionamento entre o modelo de requisitos e o modelo de arquitetura é acrescentada.

4.6.2 Modelo de Requisitos Essenciais

Nesta seção, o modelo de requisitos essenciais, anteriormente descrito na Seção 4.1, será exposto com o uso da SysML.

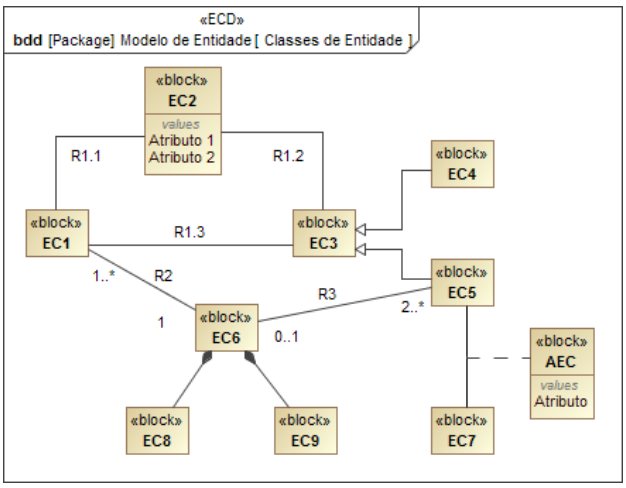
Esta seção está dividida de acordo com as quatro partes constituintes do modelo de requisitos essenciais: modelo de entidade, modelo de processo, modelo de controle e dicionário de requisitos.

4.6.2.1 Modelo de Entidade

Nesta seção, apresenta-se o modelo de entidade, anteriormente descrito na Seção 4.1.1, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Utilizam-se os Diagramas de Definição de Bloco (ou BDD) para apresentar as partes desejadas do modelo de entidade. Nestes diagramas, tem-se Blocos e seus respectivos relacionamentos para criar estruturas de dados e Propriedades de Valor atribuídas aos Blocos para detalhar seus atributos. Na Figura 4.13 abaixo, mostra-se um exemplo de BDD para sua construção.

Figura 4.13 – Exemplo de Modelo de Entidade em SysML



Fonte: Adaptada de Hatley et al. (2000).

Na figura acima, nota-se o texto “<<ECD>>” no centro superior do cabeçalho do diagrama. Este texto é, na verdade, o nome do estereótipo aplicado ao diagrama. Ou seja, ao criar o diagrama, o estereótipo <<ECD>> foi aplicado para tipificar este diagrama dentro do método HHP.

Os Blocos da SysML na figura acima (i.e. caixas com o estereótipo <<block>> acima de seus nomes) representam algumas classes de entidade, nomeadas de “EC1” a “EC9”. Os diferentes relacionamentos da SysML entre os Blocos representam relacionamento entre as respectivas classes de entidade.

Linhas cheias não direcionadas são Associações da SysML, onde os Blocos nas extremidades possuem algum tipo de relacionamento semântico entre eles. Linhas cheias não direcionadas com um diamante preto em uma das extremidades são Composições da SysML, onde o Bloco na extremidade do diamante é composto pelo Bloco na extremidade oposta. Linhas cheias direcionadas com uma seta cheia em uma das extremidades são Generalizações (ou Especializações) da SysML, onde o Bloco na extremidade da seta cheia é uma generalização do Bloco na extremidade oposta.

Textos dispostos próximos das Associações representam os nomes dados pelo engenheiro de sistemas a estes relacionamentos. No diagrama da figura acima, “R1.1” ou “R2” são alguns exemplos destes nomes.

Cardinalidades dispostas nas extremidades de uma dada Associação representam as multiplicidades do relacionamento em relação aos Blocos das extremidades em questão. No diagrama da figura acima, um exemplo destas multiplicidades são os textos “0..1” e “2..*” para a Associação de nome “R3”.

Blocos relacionados por uma linha tracejada a uma Associação representam Blocos de Associação da SysML. Neste caso, o relacionamento entre os Blocos da correspondente Associação acontece por intermédio do Bloco de Associação relacionado. No diagrama da figura acima, o Bloco nomeado “AEC” é um exemplo deste tipo de elemento.

Ao utilizar a SysML, não é possível representar uma Associação envolvendo mais de dois elementos. De acordo com a especificação da SysML (OMG, 2017), existem duas alternativas para se contornar esta situação: criar Associações entre os Blocos em uma combinação dois a dois (alternativa usada no exemplo da figura acima entre “EC1”, “EC2” e “EC3”) ou criar um Bloco concentrador das Associações entre os demais Blocos.

Nos compartimentos chamados “*values*” dos Blocos estão presentes Propriedades de Valor da SysML. Estas Propriedades de Valor devem ser tipificados com Tipos de Valor da SysML. Estes elementos representam tipos

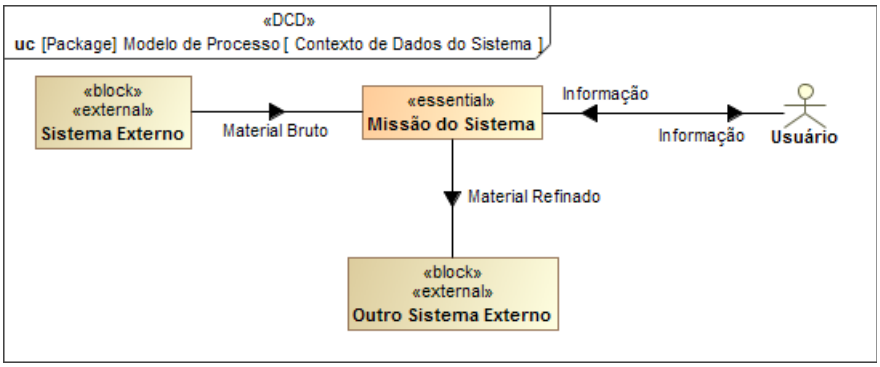
básicos dentro do modelo e estarão presentes no dicionário de requisitos onde serão devidamente detalhados.

4.6.2.2 Modelo de Processo

Nesta seção, apresenta-se o modelo de processo, anteriormente descrito na Seção 4.1.2, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Para apresentar o contexto funcional do sistema, o engenheiro de sistemas utilizará um Diagrama de Caso de Uso com um elemento Atividade da SysML representando a missão do sistema e Blocos ou Atores da SysML, representando as entidades externas. Atores representarão usuários do sistema, enquanto Blocos representarão sistemas externos interfaceando com o sistema de interesse. Na Figura 4.14 abaixo, mostra-se um exemplo de contexto funcional construído com o uso da SysML.

Figura 4.14 – Exemplo de Contexto Funcional em SysML



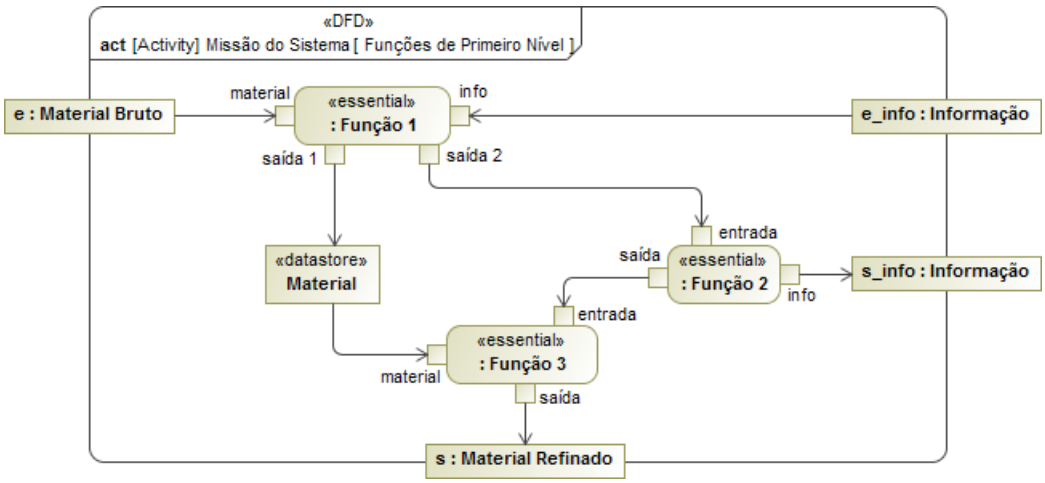
Fonte: Produção do Autor.

Na figura acima, os Blocos “Sistema Externo” e “Outro Sistema Externo” possuem o estereótipo <<*external*>> para diferenciá-los de outros Blocos que existam no modelo e para identificar que estes Blocos representam entidades que estão fora das fronteiras do sistema sendo modelado. O Ator “Usuário” representa um possível usuário do sistema. No centro da figura, tem-se a Atividade “Missão do Sistema” representando a função precípua do sistema, à qual o estereótipo <<*essential*>> foi aplicado para identificar que essa é uma função essencial do sistema. As linhas cheias são relacionamentos de Associação da SysML que foram criados entre os elementos para possibilitar

a realização de Fluxos de Informação, que representam os fluxos de dados trocados neste contexto. Estes Fluxos de Informação, quando realizados por uma Associação, são representados por setas desenhadas em cima deste relacionamento. Tais fluxos carregam informações que são tipificadas por elementos estruturais da SysML (e.g.: Bloco, Tipo de Valor, etc). Estes elementos estarão presentes ou no modelo de entidade ou no dicionário de requisitos, onde o engenheiro de sistemas terá a capacidade de especificar seus detalhes.

Como utilizam-se Atividades para representar funções, utilizar-se-á um Diagrama de Atividade para realizar o desdobramento funcional de uma dada função em suas subfunções, mostrando os fluxos de dados trocados entre elas, garantindo o balanceamento dos dados (i.e. dados de entrada da função devem chegar para suas subfunções e dados de saída, devem ser oriundos de suas subfunções). Na Figura 4.15 abaixo, mostra-se um exemplo de desdobramento funcional construído com o uso da SysML e criado com o intuito de decompor a função representando a missão do sistema.

Figura 4.15 – Exemplo de Desdobramento Funcional em SysML



Fonte: Produção do Autor.

Na figura acima, pelo cabeçalho do diagrama, nota-se de que se trata de um Diagrama de Atividade para a função (ou Atividade) “Missão do Sistema”, anteriormente apresentada na Figura 4.14.

Os retângulos nas bordas do diagrama representam os parâmetros da função, que foram criados de tal forma a espelhar os fluxos de dados representados na Figura 4.14. Dentro destes retângulos, conforme as regras da SysML, tem-se “Nome:Tipo”, onde o nome é arbitrário e tipo é o mesmo utilizado nos fluxos de dados da Figura 4.14, para garantir a consistência do modelo sendo criado.

Os retângulos de cantos arrendondados representam chamadas para as subfunções da função considerada (nota-se o uso do “:” antes do nome da subfunção referenciada), subfunções estas igualmente representadas por Atividades da SysML. Os quadrados nas bordas das subfunções representam seus Parâmetros, que seguem a mesma regra já apresentada. Nota-se, ainda, o uso do estereótipo <<*essential*>> nas subfunções, pois se tratam de subfunções essenciais.

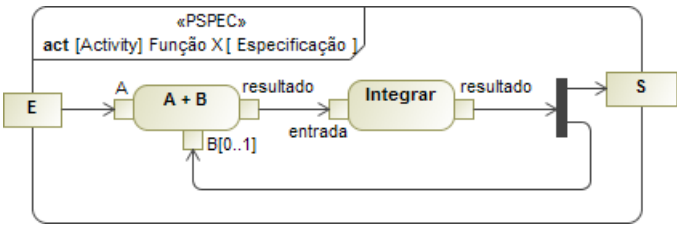
As linhas cheias direcionadas são Fluxos de Objeto da SysML que representam os fluxos de dados entre funções. É através destes Fluxos de Objeto que o sentido de um dado Parâmetro é inferido, ou seja, se ele é um parâmetro de entrada ou de saída.

Já o retângulo de cantos retos com a identificação <<*datastore*>> representa um repositório de dados.

O desdobramento funcional continua até que funções terminais sejam identificadas. O engenheiro de sistemas necessita, agora, especificar de maneira precisa e livre de ambiguidades as transformações exercidas por estas funções terminais. Como o elemento Atividade da SysML está sendo utilizado para representar as funções, faz-se estas especificações utilizando Diagramas de Atividades para as funções terminais. Nestes diagramas, não poderão existir chamadas para subfunções e o engenheiro de sistemas terá à sua disposição toda a variedade de construções possíveis para a modelagem de Atividades previstas na SysML.

Nas duas figuras a seguir, mostram-se Diagramas de Atividade para duas funções terminais hipotéticas “Função X” e “Função Y”, como forma de exemplificar a especificação de suas respectivas transformações.

Figura 4.16 – Exemplo de Especificação para a ‘Função X’ em SysML

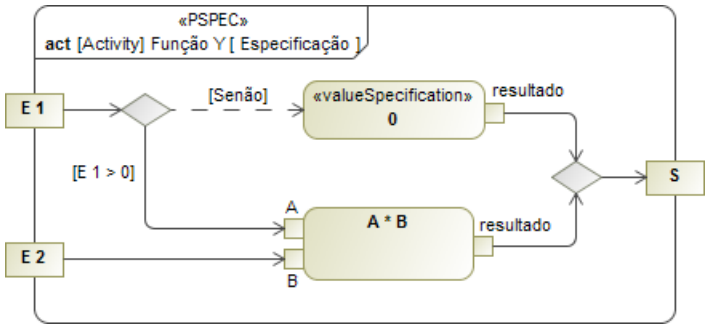


Fonte: Produção do Autor.

Na figura acima, tem-se a especificação da função terminal hipotética “Função X”, materializada através do uso de um Diagrama de Atividade contendo Ações básicas, representadas pelos retângulos de cantos arredondados. A barra vertical representa uma bifurcação, pela qual é possível representar a cópia de dados de uma única fonte para diversos destinatários.

Assumindo-se que o diagrama construído é válido, ou seja, as regras da SysML foram observadas, por construção garante-se que a especificação por ele representada é livre de ambiguidades.

Figura 4.17 – Exemplo de Especificação para a ‘Função Y’ em SysML



Fonte: Produção do Autor.

Na figura acima, o losango com uma entrada e múltiplas saídas representa um nó de decisão, utilizado para representar fluxos alternativos, cujas condições de execução são especificadas textualmente próximas aos correspondentes fluxos. Já o losango com múltiplas entradas e uma única saída representa um nó de mesclagem, que junta fluxos alternativos para dar continuidade à execução da Atividade.

A linha tracejada direcionada representa aquilo que é chamado de Fluxo de Controle em Diagramas de Atividade e que não pode ser confundido com o

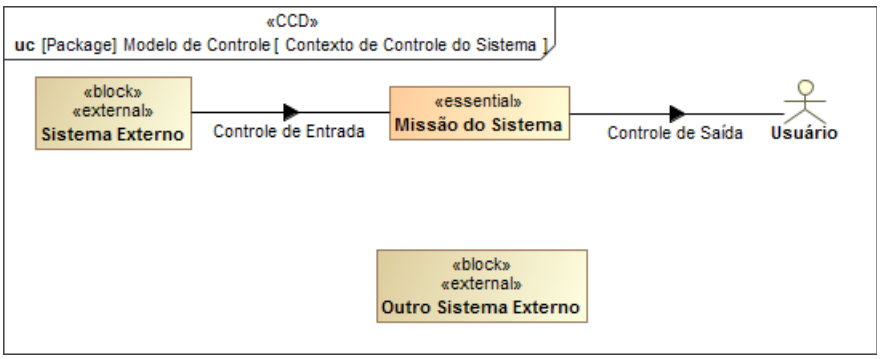
fluxo de controle do método HHP. Fluxos de Controle em Diagramas de Atividade tem por objetivo ativar e sequenciar as ações que ocorrem dentro da correspondente Atividade. Nem sempre é necessário utilizar este tipo de fluxo, mas existem situações onde seu uso é obrigatório. Caso os Fluxos de Objeto existentes no diagrama não determinem de forma única a ativação e sequenciamento das Ações ali presentes, Fluxos de Controle deverão ser adicionados para resolver estas ambiguidades. Outra situação de uso de Fluxos de Controle é com Ações que não possuam parâmetros de entrada. No exemplo da Figura 4.17 acima, a Ação do tipo <<*valueSpecification*>> quando ativada coloca no seu único parâmetro de saída o valor nela especificado. Por não possuir parâmetro de entrada, sua ativação se dá por meio de um Fluxo de Controle.

4.6.2.3 Modelo de Controle

Nesta seção, apresenta-se o modelo de controle, anteriormente descrito na Seção 4.1.3, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Para apresentar o contexto de controle do sistema, o engenheiro de sistemas utilizará novamente um Diagrama de Caso, contendo os mesmos elementos do diagrama criado para o contexto funcional do sistema, ou seja, a Atividade representando a missão do sistema e os Blocos e Atores representando as entidades externas. Na Figura 4.18 abaixo, mostra-se um exemplo de contexto de controle construído com o uso da SysML.

Figura 4.18 – Exemplo de Contexto de Controle em SysML



Fonte: Produção do Autor.

Na figura acima, os elementos da SysML utilizados são praticamente os mesmos daqueles utilizados na Figura 4.14, onde foi apresentado o contexto funcional do sistema. A única diferença está nos Fluxos de Informação, que aqui carregam sinais de controle, representados pelo elemento Sinal da SysML. Estes sinais estarão presentes ou no modelo de entidade ou no dicionário de requisitos, onde o engenheiro de sistemas poderá fornecer seus detalhes para complementar a especificação.

O engenheiro de sistemas necessita, agora, definir a ativação das funções identificadas no modelo de processo através de uma especificação de controle. Para tal, utiliza-se uma Máquina de Estado da SysML, com seu respectivo Diagrama de Máquina de Estado. Na Figura 4.19 abaixo, mostra-se uma especificação de controle hipotética, criada com o uso da SysML.

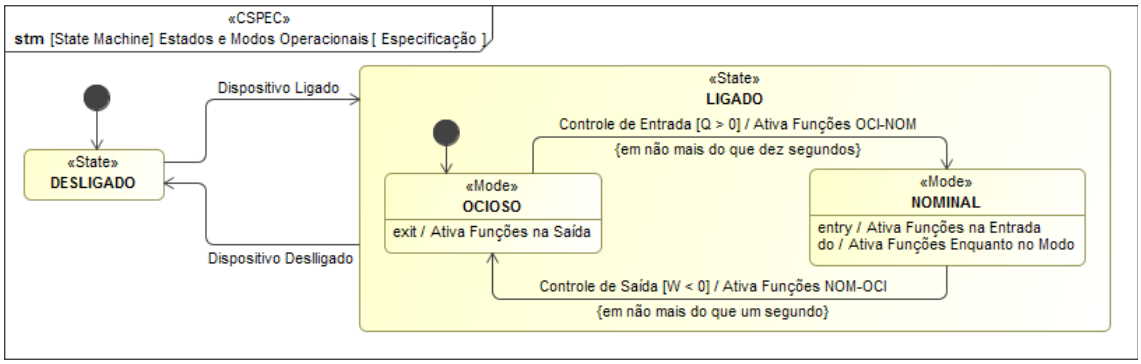
Conforme visto na seção mencionada, dois tipos de artefatos são produzidos pelo engenheiro de sistemas durante o processo de construção do modelo de controle. São eles: diagrama de contexto de controle (ou CCD) e especificação de controle (ou CSPEC). Na abordagem usando a SysML, diferentemente do método HHP, os diagramas de fluxos de controle (ou CFDs) correspondentes aos diferentes DFDs do modelo de processo não serão produzidos. Isso porque, Fluxos de Controle em Diagramas de Atividade da SysML possuem um significado diferente daquele do fluxo de controle utilizado em um CFD do método original. De qualquer forma, a ausência dos CFDs não comprometerá o atingimento do objetivo principal do modelo de controle, que é permitir ao engenheiro de sistemas determinar as ativações e sequenciamentos das funções identificadas no modelo de processo.

O CCD será criado de forma semelhante ao DCD, feito anteriormente na Seção 4.6.2.2, ou seja, com o uso de Atores, Blocos, Atividades e Fluxos de Informação da SysML. Já a CSPEC, que no método HHP original é criada para cada nível da decomposição funcional feita durante elaboração do modelo de processo, será aqui criada uma única vez e representada por uma

Máquina de Estado da SysML, com seu respectivo diagrama de máquina de estado. O propósito da CSPEC na metodologia aqui proposta será permitir ao engenheiro de sistemas definir os estados e modos operacionais do sistema de interesse, assim como sugerido no método de engenharia de sistemas proposto por Loureiro (1999), onde será possível definir, também, quais funções terminais estarão disponíveis em quais modos operacionais de quais estados do sistema.

Para criar o CCD, o engenheiro de sistemas utilizará um Diagrama de Caso de Uso da SysML. Na Figura 4.18 abaixo, mostra-se um exemplo de CCD construído com o uso da SysML.

Figura 4.19 – Exemplo de Especificação de Controle em SysML



Fonte: Produção do Autor.

No diagrama acima, o círculo preto representa o pseudo-estado usado para identificar o estado inicial do sistema, os retângulos com cantos arredondados representam estados ou modos operacionais do sistema, de acordo com o estereótipo a eles aplicados, e as linhas direcionadas representam as transições entre estes elementos. No exemplo utilizado, o sistema apresenta dois estados: desligado e ligado. Enquanto no estado ligado, o sistema apresenta dois modos operacionais: ocioso e nominal. Para um sistema mudar de estado ou de modo operacional, a Transição da correspondente Máquina de Estado responsável por esta mudança deve disparar. Pela especificação da SysML, uma Transição é identificada da seguinte maneira: “Evento[Condição]/Comportamento”, onde ‘Evento’ representa o sinal de controle monitorado para o disparo da transição, ‘Condição’ representa uma expressão que pode ser adicionada para

condicionar o disparo da transição na ocorrência do sinal de controle ‘Evento’, e, finalmente, ‘Comportamento’ representa um comportamento da SysML (e.g.: uma Atividade) que pode ser executado quando a Transição for disparada.

Na Figura 4.19, o sistema inicia no estado desligado e muda deste estado para o estado ligado quando o evento “Dispositivo Ligado” acontecer. De forma análoga, o sistema volta do estado ligado para o estado desligado quando o evento “Dispositivo Desligado” acontecer. Ao entrar no estado ligado, o sistema inicia no modo operacional ocioso. Quando acontecer o evento “Controle de Entrada”, já identificado no contexto de controle da Figura 4.18, e a condição “Q > 0” for verdadeira, o sistema muda para o modo operacional nominal. Nesta transição, o comportamento “Ativa Funções OCI-NOM” é executado com uma Restrição da SysML relacionada, cujo o texto é “{em não mais do que dez segundos}”, para se fornecer o atributo de desempenho desejado para o referido comportamento. O sistema, então, volta do modo operacional nominal ao modo operacional ocioso quando o evento “Controle de Saída”, já identificado no contexto de controle da Figura 4.18, e a condição “W < 0” for verdadeira. Nesta transição, o comportamento “Ativa Funções NOM-OCI” é executado com uma restrição de desempenho “{em não mais do que um segundo}”.

De acordo com a especificação da SysML, o elemento Estado pode, opcionalmente, conter alguns compartimentos para indicar a execução de outros comportamentos da SysML. O compartimento chamado *“entry”* pode ser usado para identificar um comportamento que será executado quando o sistema entrar naquele Estado, antes da execução de qualquer outro comportamento. O compartimento *“do”* pode ser utilizado para identificar um comportamento que será executado enquanto o sistema permanecer naquele Estado e até que ele termine por si só ou o sistema deixe o correspondente Estado, o que acontecer primeiro. Por último, o compartimento *“exit”* pode ser utilizado para identificar um comportamento que será executado quando o

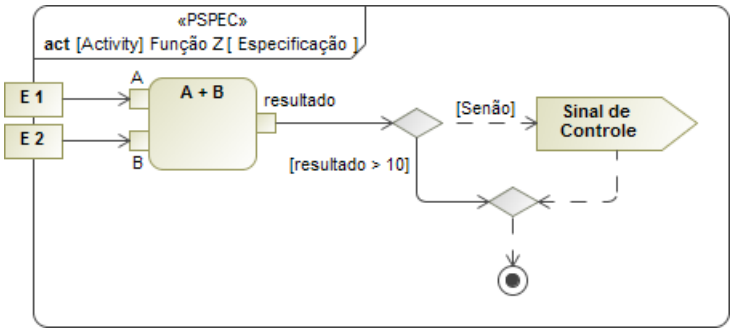
sistema sair daquele Estado, após qualquer outro comportamento ter sido encerrado.

Voltando à Máquina de Estado descrita na Figura 4.19, quando o sistema deixa o modo operacional ocioso, o comportamento “Ativa Funções na Saída” é executado. Quando o sistema entra no modo operacional nominal, o comportamento “Ativa Funções na Entrada” é executado. Finalmente, enquanto permanecer neste modo, o comportamento “Ativa Funções Enquanto no Modo” é executado.

De acordo com a especificação da SysML, os compartimentos ‘*entry’*, ‘*do’* e ‘*exit’* são opcionais, mas se utilizados, podem identificar apenas um único comportamento. Por este motivo, no escopo da metodologia proposta, quando utilizados, estes compartimentos devem referenciar um elemento Atividade da SysML. No correspondente Diagrama de Atividade deve-se, portanto, identificar todas as funções terminais do sistema que estarão ativas ao entrar (i.e. compartimento ‘*entry*’), enquanto permanecer (i.e. compartimento ‘*do*’) ou quando sair (i.e. compartimento ‘*exit*’) do correspondente estado ou modo operacional.

Um fluxo de controle gerado internamente ao sistema e utilizado na sua especificação de controle, chamado aqui de ‘Condição de Dados’, necessita ter a condição na qual ele é gerado precisamente especificada. Para tal, o engenheiro de sistemas utilizará uma Ação chamada de *SendSignalAction* da SysML, que indica o disparo de um elemento Sinal da SysML, representando o fluxo de controle desejado. Na Figura 4.20 abaixo, mostra-se uma especificação de uma função terminal hipotética, que gera um fluxo de controle do tipo ‘Condição de Dados’. Neste diagrama, o retângulo mesclado com um triângulo no canto direito da figura representa uma Ação que envia o sinal de controle nele especificado. Este tipo de elemento é utilizado para gerar fluxos de controle internamente ao sistema, em uma dada condição determinada pelo próprio engenheiro de sistemas. O círculo vazado com um círculo preto dentro representa o nó de Fim de Atividade da SysML, ou seja, quando a execução ali chega, a Atividade é imediatamente encerrada.

Figura 4.20 – Exemplo de ‘Condição de Dados’ em SysML



Fonte: Produção do Autor.

Com o apresentado até este ponto, o engenheiro de sistemas tem à sua disposição as ferramentas necessárias para criar o modelo de requisitos essenciais do sistema de interesse.

4.6.2.4 Dicionário de Requisitos

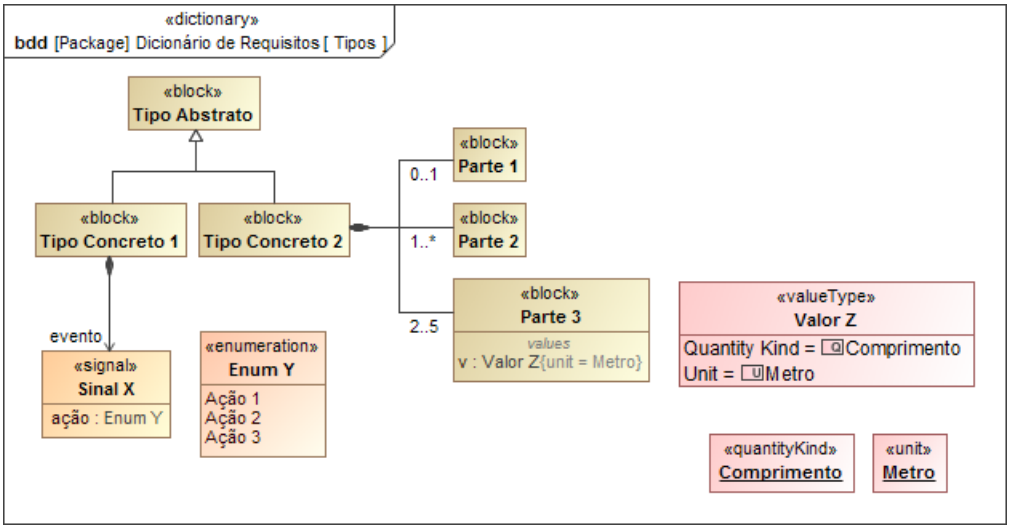
Nesta seção, apresenta-se o dicionário de requisitos, anteriormente descrito na Seção 4.1.4, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Este dicionário deverá especificar a tipificação utilizada nos três modelos apresentados anteriormente: modelo de entidade, modelo de processo e modelo de controle. Nestes modelos, conforme visto nas seções anteriores, foram utilizados diferentes elementos da SysML para representar atributos, fluxos de dados, sinais de controle, etc. Estes elementos possuem, de acordo com a especificação da SysML, um nome escolhido arbitrariamente pelo engenheiro de sistemas e um tipo. Este tipo deve, necessariamente, ser representado por um elemento estrutural da SysML, que pode ser um Bloco, um Tipo de Valor, um Sinal ou uma Enumeração de valores previamente conhecidos. Estes elementos, representando os tipos utilizados nos três modelos mencionados, formam o dicionário de requisitos.

Na Figura 4.21 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML para representar uma parte de um dicionário de requisitos hipotético. Um dicionário de requisitos não precisa ser, necessariamente, representado por completo em um único Diagrama de Definição de Bloco. O engenheiro de

sistemas pode, de acordo com seu julgamento, utilizar vários Diagramas de Definição de Bloco para representar diferentes aspectos do dicionário de requisitos. Além do mais, um único diagrama mostrando todos os tipos envolvidos na definição do modelo de requisitos de um sistema será, certamente, ilegível quando publicado em um documento, ou mesmo quando acessado diretamente na ferramenta de modelagem utilizada.

Figura 4.21 – Exemplo de Dicionário de Requisitos em SysML



Fonte: Produção do Autor.

Na figura acima, os Blocos representam tipos complexos de dados, que podem se desdobrar, criando estruturas de dados complexas. O elemento identificado com <<*signal*>> representa um Sinal da SysML e é utilizado tanto em Máquinas de Estado, no disparo de suas Transições, quanto em Diagramas de Atividade, indicado sinais de controle internos ao sistema. No exemplo, o Sinal contém um atributo chamado “ação” tipificado por uma Enumeração de valores, esta representada pelo elemento com o identificador <<*enumeration*>> sobre seu nome. Especificamente o Bloco “Parte 3” possui um atributo chamado “v”, tipificado pelo Tipo de Valor da SysML nomeado “Valor Z”, elemento este identificado com o identificador <<*valueType*>> sobre seu nome. Em SysML, um Tipo de Valor representa um tipo primitivo (i.e. um tipo que não pode ser decomposto) e pode conter duas especificações adicionais: Tipo de Quantidade e Unidade. A primeira especificação é representada pelo elemento da SysML com o identificador

<<*quantityKind*>>, enquanto a segunda, pelo elemento com o identificador <<*unit*>>.

De posse destes elementos e relacionamentos, o engenheiro de sistemas cria a estrutura de dados que representa as tipificações utilizadas no modelo de requisitos, mais especificamente, nos fluxos de dados, atributos, sinais de controle, etc.

O método HHP originalmente estabelece uma notação própria para definir o dicionário de requisitos. A notação '=' do método HHP corresponde ao relacionamento de Composição da SysML. Multiplicidades nos relacionamentos da SysML correspondem à notação '{ }' e '( )' do método HHP. Enumerações da SysML correspondem à notação '[| |]' do método original. Finalmente, a notação '+' do método HHP corresponde às Propriedades de Blocos da SysML.

Um outro tipo de especificação feita no dicionário de requisitos do método HHP está relacionado às restrições de desempenho que o sistema deve apresentar da recepção de uma entrada até a produção das saídas correspondentes. No método original, isso era feito com o auxílio de uma tabela, listando as entradas e suas correspondentes saídas, e qual desempenho esperado neste processamento. Até a versão 1.5 da SysML (versão corrente da linguagem na época em que esta tese foi elaborada), não existe construções disponíveis para representar este tipo de restrição. O Diagrama de Tempo, disponível na UML (OMG, 2015) e cuja finalidade é exatamente a representação de restrições de desempenho entre dois momentos específicos da execução do *software* sendo modelado, foi retirado da especificação da SysML.

Serão, portanto, utilizados elementos do tipo Restrição da SysML (ou *Constraint*, do Inglês) para representar as restrições de desempenho das funções do sistema. Estes elementos possuem uma expressão, pela qual avalia-se um critério de verificação e cujo resultado deve ser verdadeiro ou falso. Para, então, especificar uma restrição de desempenho no sistema, deve-se criar um elemento do tipo Restrição da SysML e relacioná-lo ao

elemento Atividade da SysML que representa a função restringida. A expressão desta Restrição deve determinar com precisão o critério de aceitação de desempenho da função relacionada. Um exemplo deste tipo de construção já foi utilizado na criação da especificação de controle, apresentada anteriormente na Figura 4.19. Elementos do tipo Restrição da SysML foram relacionados às Atividades representando as funções executadas nas transições entre os modos operacionais ocioso e nominal, tendo suas expressões apresentadas no referido diagrama, abaixo das respectivas transições, em um texto entre chaves.

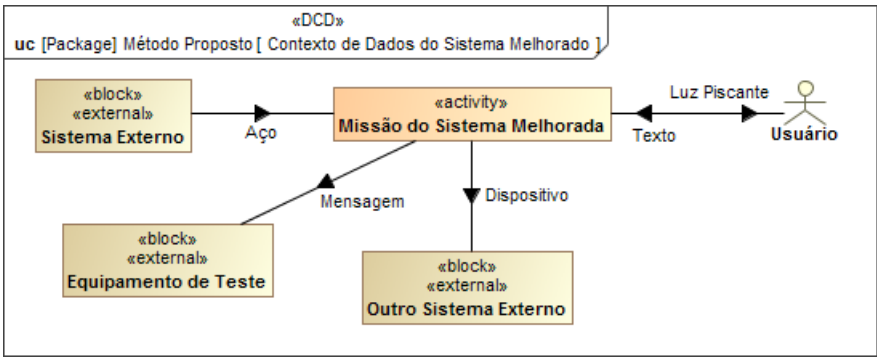
4.6.3 Modelo de Requisitos Melhorados

Nesta seção, apresenta-se o modelo de requisitos melhorados, anteriormente descrito na Seção 4.2, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Conforme visto anteriormente, o objetivo principal deste modelo é acoplar o modelo de requisitos essenciais ao mundo real, através de funções de entrada, de saída, de interface com o usuário e de suporte.

Este acoplamento começa do ponto de vista do contexto funcional do sistema, onde um novo Diagrama de Caso de Use é criado, mostrando a nova realidade do sistema. Na Figura 4.22 abaixo, mostra-se o novo contexto funcional, baseado naquele apresentado anteriormente na Figura 4.14.

Figura 4.22 – Exemplo de Contexto Funcional Melhorado em SysML



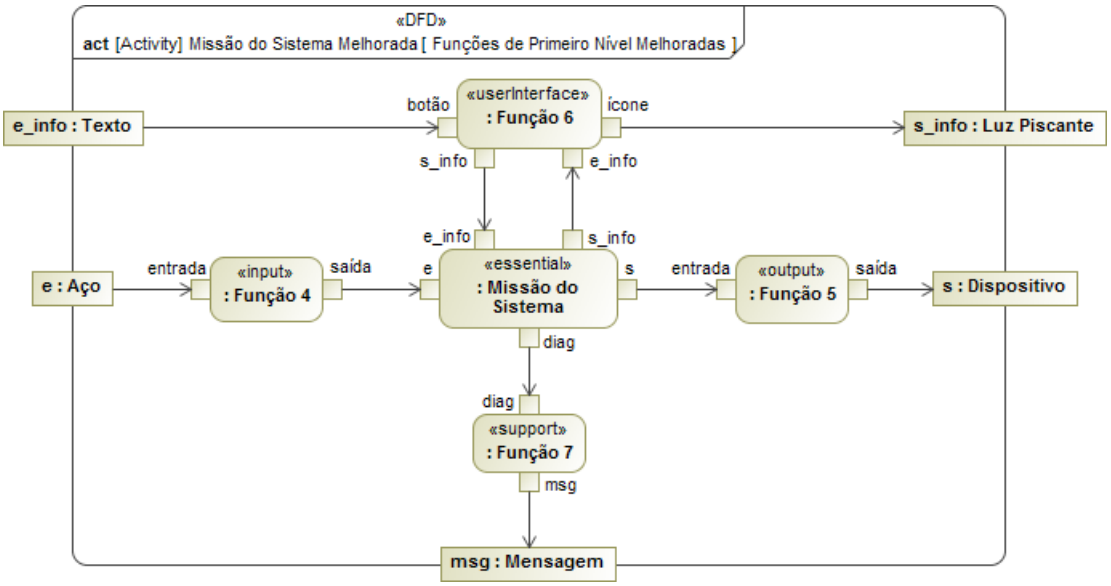
Fonte: Produção do Autor.

Na figura acima, temos as mesmas entidades externas apresentadas anteriormente na Figura 4.14. Entretanto, agora, estas entidades trocam com

o sistema fluxos reais de material, energia ou informação. O sistema, por sua vez, é aqui representado pela sua função precípua melhorada, ou seja, uma função que recebe e envia estes fluxos reais. Novamente, esta função é materializada através de um elemento Atividade da SysML.

Da mesma maneira como feito anteriormente, a função precípua melhorada do sistema será desdobrada até que as novas funções terminais sejam identificadas e suas transformações, precisamente especificadas. Na Figura 4.23 abaixo, apresenta-se um Diagrama de Atividade com o desdobramento funcional para a missão do sistema melhorada.

Figura 4.23 – Exemplo de Desdobramento Funcional Melhorado em SysML



Fonte: Produção do Autor.

Na figura acima, temos no centro da figura uma chamada para a missão do sistema essencial, que foi anteriormente utilizada no contexto funcional da Figura 4.14 e desdobrada no diagrama da Figura 4.15. Ao seu redor, temos as funções não-essenciais identificadas para acoplar a missão do sistema essencial ao mundo real, ou seja, transformar os fluxos reais trocados com entidades externas nos fluxos essenciais esperados por esta função. Novamente, funções não-essenciais estão representadas por Atividades da SysML devidamente identificadas com estereótipos próprios para cada tipo de função. O estereótipo <<*input*>> é usado para identificar atividades representando funções de entrada, o estereótipo <<*output*>>, funções de

saída, o estereótipo <<*userInterface*>>, funções de interface com o usuário e, finalmente, o estereótipo <<*support*>>, funções de suporte.

A partir deste ponto, o engenheiro de sistemas desdobrará as funções melhoradas identificadas até identificar novas funções terminais, cujas transformações serão precisamente especificadas. Novamente, o modelo de controle será revisitado para garantir a ativação e desativação destas novas funções. Como consequência, o dicionário de requisitos será atualizado para incluir as especificações dos tipos de dados utilizados no modelo de requisitos melhorado.

Com isso, o engenheiro de sistemas conclui o modelo de requisitos e segue para a definição da arquitetura do sistema.

4.6.4 Modelo de Arquitetura

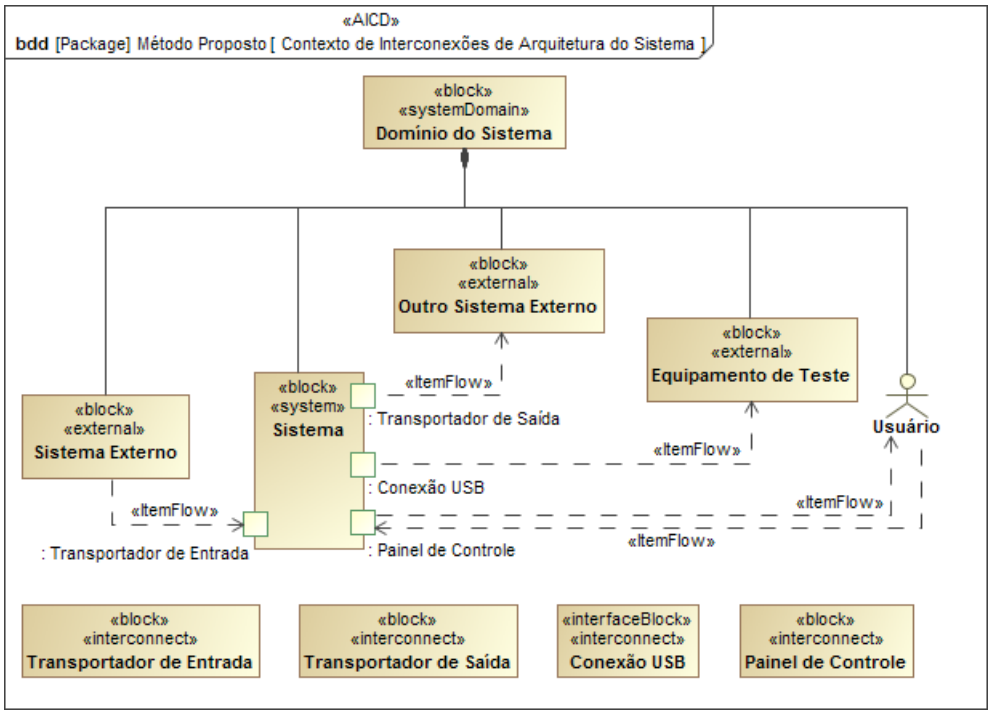
Nesta seção, apresenta-se o modelo de arquitetura, anteriormente descrito na Seção 4.3. Por se tratar de um modelo que representa os aspectos físicos e estruturais, o engenheiro de sistemas utilizará, de maneira abundante, o elemento Bloco da SysML e seus respectivos diagramas e relacionamentos.

4.6.4.1 Modelo de Contexto de Arquitetura

Nesta seção, apresenta-se o modelo de contexto de arquitetura, anteriormente descrito na Seção 4.3.1, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Na Figura 4.24 abaixo, apresenta-se um Diagrama de Definição de Bloco da SysML, onde o engenheiro de sistemas especifica as interconexões do sistema com as entidades externas. Neste diagrama, tem-se o Bloco nomeado como “Domínio do Sistema” com o estereótipo <<*systemDomain*>> para representar o domínio em que o sistema sendo modelo está inserido. Este Bloco possui relacionamentos de composição com os Blocos representando as entidades externas, já vistos anteriormente em outros diagramas, e o Bloco representando o sistema propriamente dito, identificado com o estereótipo <<*system*>>.

Figura 4.24 – Exemplo de Contexto de Interconexões de Arquitetura em SysML



Fonte: Produção do Autor.

Os pontos de interação entre o sistema sendo modelado e as entidades externas presentes no domínio do sistema estão aqui definidos por Portas da SysML, criadas para o Bloco representando o sistema, e são apresentadas pelos quadrados localizados nas bordas deste Bloco. Estas Portas serão tipificadas por Blocos ou Blocos de Interface da SysML, possuindo o estereótipo <<*interconnect*>>, para representar o meio físico pelo qual os fluxos de arquitetura são trocados. Na Figura 4.24 acima, tais elementos estão presentes na parte inferior do diagrama.

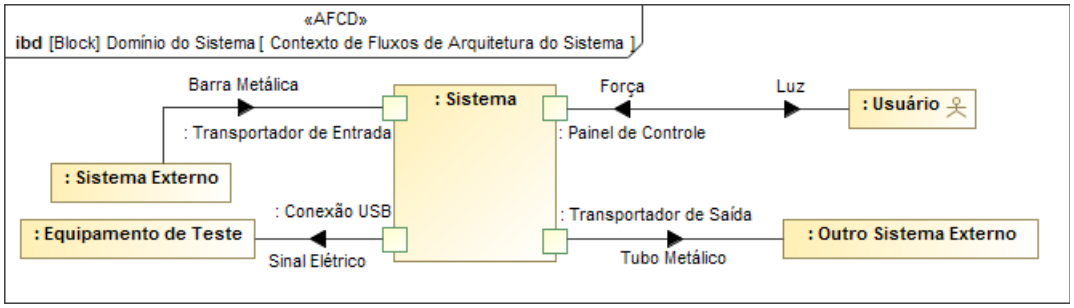
A estes elementos representando as interconexões do sistema, pode-se adicionar Propriedades da SysML com o objetivo de fornecer suas mais diversas características. Estas Propriedades serão tipificadas por elementos definido no dicionário de arquitetura, discutido mais adiante neste capítulo, na Seção 4.6.4.3.

No caso do diagrama da Figura 4.24 acima, optou-se por utilizar o elemento Fluxo de Item da SysML entre as Portas (ou interfaces) do sistema e as entidades externas para identificar quais delas estão envolvidas com qual

interface. Estes elementos estão representados através de uma linha direcionada tracejada, com o estereótipo <<*ItemFlow*>>. Por possuir sentido, caso uma dada interconexão possibilite a comunicação bidirecional entre o sistema e a entidade externa envolvida, dois elementos Fluxo de Item da SysML deverão ser utilizados, um para cada sentido de fluxo de arquitetura.

Após ter definido as interconexões entre o sistema e as entidades externas, o engenheiro de sistemas passa para a definição dos fluxos de arquitetura trocados entre eles. Para isso, utiliza-se um Diagrama Interno de Bloco da SysML para o Bloco representando o domínio onde o sistema está inserido. Na Figura 4.25 abaixo, apresenta-se um exemplo de contexto de fluxo de arquitetura para o sistema.

Figura 4.25 – Exemplo de Contexto de Fluxo de Arquitetura em SysML



Fonte: Produção do Autor.

Na figura acima, pelo cabeçalho do diagrama, nota-se que se trata de um diagrama do tipo ‘ibd’ (do Inglês, *Internal Block Diagram*) para o Bloco “Domínio do Sistema” (apresentado na Figura 4.24) e cujo nome é “Contexto de Fluxos de Arquitetura do Sistema”.

Neste diagrama, notam-se as presenças tanto do sistema sendo modelado, como das entidades externas, todos eles presentes no contexto de interconexões apresentado anteriormente na Figura 4.24. Estes elementos são, na verdade, o que se é chamado de Parte na SysML e que foram criados implicitamente no diagrama, quando se criou o relacionamento de Composição entre o Bloco “Domínio do Sistema” tanto com o Bloco representando o sistema propriamente dito, como com os Blocos e Ator representando as entidades externas. Notam-se, também, os pontos de

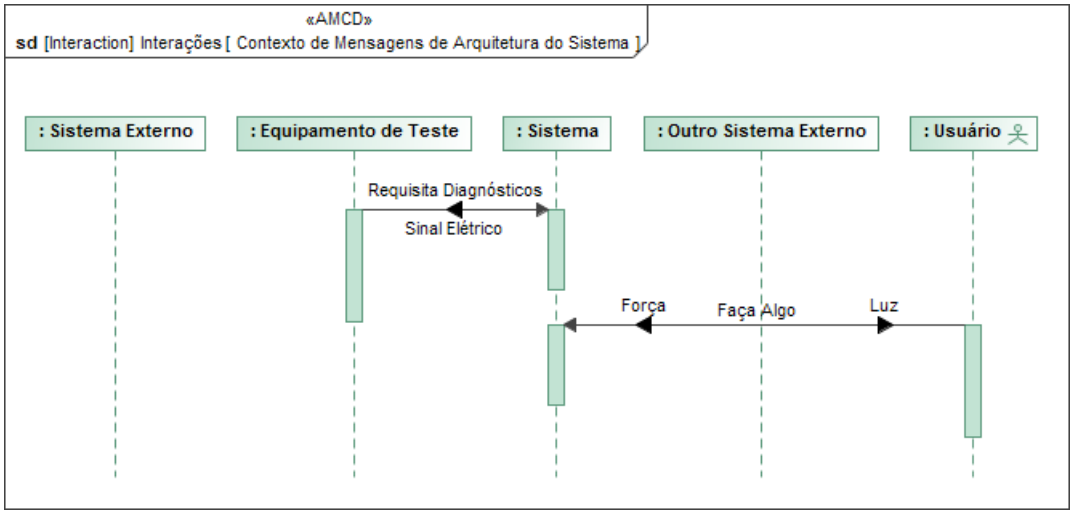
interação do sistema com as entidades externas, representados pelas Portas da SysML criadas durante a confecção do contexto de interconexões da Figura 4.24.

As conexões entre o sistema e as entidades externas dão-se através do elemento Conector da SysML, visto no diagrama da Figura 4.25 como sendo uma linha cheia entre uma dada entidade externa e a Porta do sistema tipificada pela correspondente interconexão.

Os fluxos de arquitetura trocados entre o sistema e as entidades externas são representados por elementos Fluxo de Item da SysML. No diagrama da Figura 4.25 acima, estes fluxos são desenhados nos conectores interligando as partes envolvidas como uma seta cheia, no sentido em que o fluxo flui.

Caso seja necessário acrescentar a informação referente ao sequenciamento esperado entre os fluxos de arquitetura, o engenheiro de sistemas utiliza mensagens de arquitetura entre as partes envolvidas. Para esta etapa, utiliza-se o Diagrama de Sequência da SysML. Na Figura 4.26 abaixo, apresenta-se um exemplo de contexto de mensagens de arquitetura.

Figura 4.26 – Exemplo de Contexto de Mensagens de Arquitetura em SysML



Fonte: Produção do Autor.

Na figura acima, pelo cabeçalho do diagrama, nota-se que se trata de um diagrama do tipo ‘sd’ (do Inglês, *Sequence Diagram*), que de acordo com a especificação da SysML, deve ser criado dentro do escopo de um elemento

Interação da SysML. Por isso, tem-se do cabeçalho que o elemento encapsulando o diagrama é do tipo '*Interaction*', cujo nome é "Interações". Por fim, o nome do diagrama é "Contexto de Mensagens de Arquitetura do Sistema".

Neste diagrama, notam-se as presenças do sistema sendo modelado (ao centro da figura) e das entidades externas (em ambos os cantos da figura), todos eles presentes no diagrama da Figura 4.24. Estes elementos são, na verdade, Linhas de Vida da SysML, que são utilizadas para representar os correspondentes Blocos e Ator (i.e. sistema e entidades externas) em um certo contexto dinâmico.

As mensagens de um elemento para o outro são representadas pelo elemento Mensagem da SysML e são desenhadas como linhas direcionadas indo do elemento de origem da mensagem para o elemento destinatário desta mensagem. Fluxos de arquitetura sendo transportados pelas mensagens são representados pela realização de Fluxos de Item da SysML nas Mensagens correspondentes. Esta construção é representada por uma seta cheia desenhada em cima da linha da Mensagem e com o sentido do Fluxo de Item. No diagrama da Figura 4.26 têm-se alguns dos Fluxos de Item apresentados anteriormente no diagrama da Figura 4.25 sendo realizados pelas Mensagens aqui representadas.

Agora, após modelar as características estruturais e físicas do sistema de um ponto de vista externo, o engenheiro de sistemas passa para o modelo da arquitetura interna do sistema de interesse.

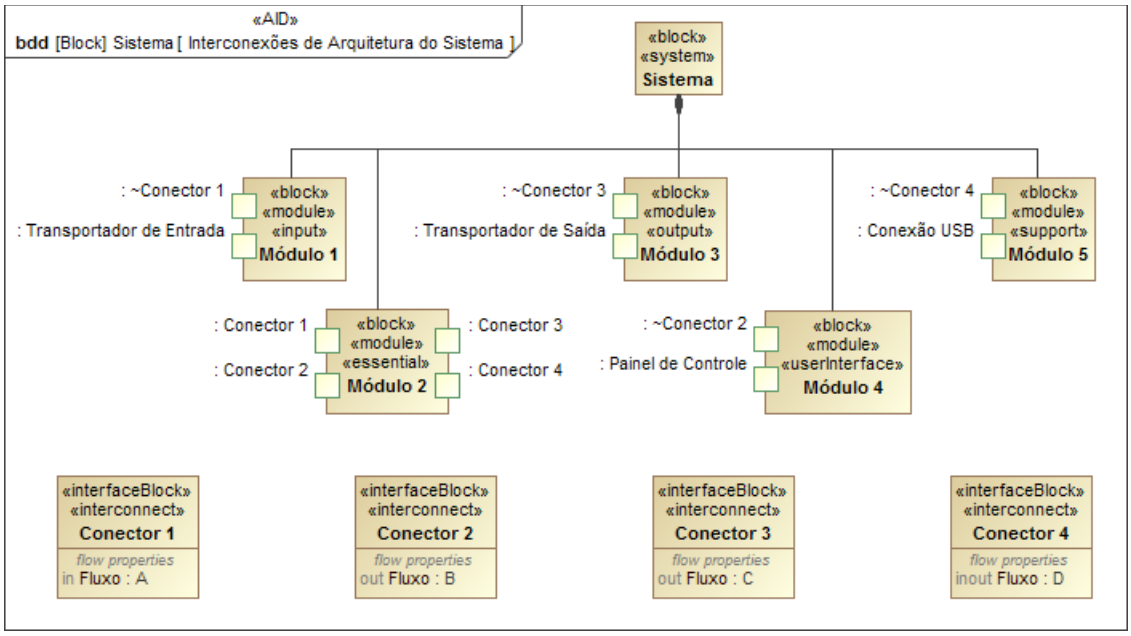
4.6.4.2 Modelo de Módulos de Arquitetura

Nesta seção, apresenta-se o modelo de módulos de arquitetura, anteriormente descrito na Seção 4.3.2, utilizando as construções da SysML mais adequadas para se representar os correspondentes conceitos.

Na Figura 4.27 abaixo, apresenta-se um exemplo de diagrama de interconexão de arquitetura através de um Diagrama de Definição de Bloco da SysML, onde o engenheiro de sistemas especificará as interconexões

entre os módulos de arquitetura. Pela própria similaridade com o diagrama de contexto de interconexões, apresentado anteriormente na Figura 4.24, algumas construções aqui presentes não serão discutidas.

Figura 4.27 – Exemplo de Interconexões de Arquitetura em SysML



Fonte: Produção do Autor.

Na parte superior central da figura, tem-se o Bloco representando o sistema sendo modelado. Compondo o sistema, estão seus módulos de arquitetura, representados por Blocos da SysML possuindo o estereótipo <<*module*>>. Além deste estereótipo, estes Blocos também possuem um estereótipo que os tipificam de acordo com sua finalidade, assim como feito para as funções do sistema. O estereótipo <<*essential*>> identifica módulos essenciais do sistema, o estereótipo <<*input*>>, módulos de entrada, o estereótipo <<*output*>>, módulos de saída, o estereótipo <<*userInterface*>>, módulos de interface com os usuários do sistema e, finalmente, o estereótipo <<*support*>>, módulos de suporte.

Os pontos de interação de um módulo de arquitetura estão representados, na Figura 4.27 acima, por Portas da SysML. Tais Portas estão tipificadas, neste exemplo, por elementos do tipo Bloco de Interface da SysML com o estereótipo <<interconnect>> aplicado, representando as interconexões entre os módulos de arquitetura, aqui chamadas de “Conector 1”, “Conector 2”,

“Conector 3” e “Conector 4”, vistos na parte inferior da Figura 4.27. Para não poluir desnecessariamente o diagrama, neste caso optou-se por não desenhar Fluxos de Item entre os módulos de arquitetura, assim como foi feito no diagrama da Figura 4.24. Entretanto, verificando-se as Portas tipificadas pelo mesmo elemento, pode-se identificar quais são os módulos de arquitetura envolvidos para uma dada interconexão.

Um conceito da SysML que aparece na Figura 4.27 acima é o de Porta conjugada. Estas Portas são identificadas pelo símbolo ´~´ antecedendo o nome dos elementos que as tipificam. Ao estabelecer que uma Porta é conjugada, o engenheiro de sistemas força uma inversão nos sentidos dos possíveis Fluxos de Propriedade da SysML presentes no elemento tipificando a correspondente Porta.

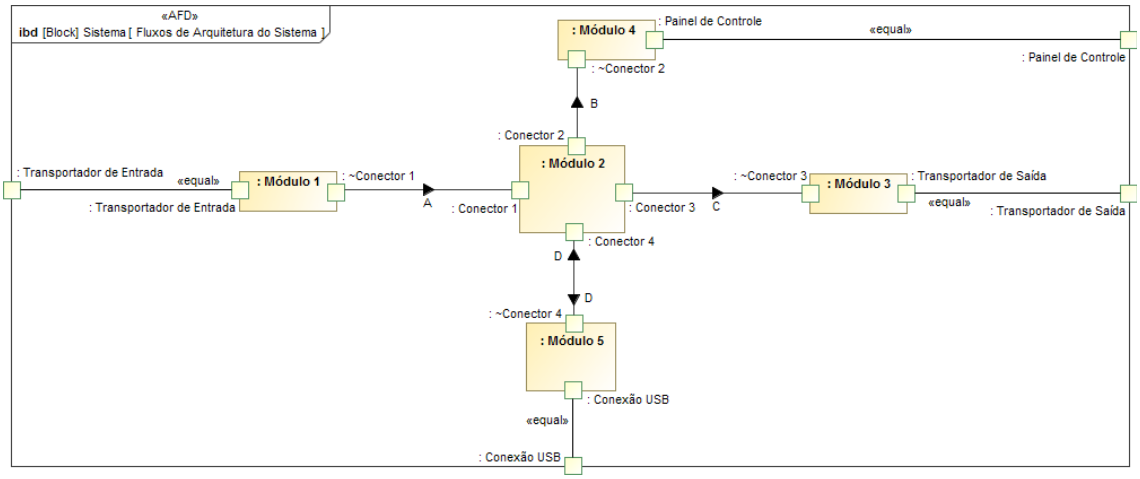
No exemplo do diagrama da Figura 4.27 acima, o “Módulo 1”, o “Módulo 3”, o “Módulo 4” e o “Módulo 5” apresentam Portas conjugadas. No caso do “Módulo 1”, a Porta conjugada está tipificada pela interconexão “Conector 1”, que possui um Fluxo de Propriedade chamado “Fluxo” tipificado por um Bloco chamado “A”. Este é um fluxo de entrada, assim definido pelo classificador ‘*in’* antecedendo seu nome. Isto significa que uma Porta tipificada por esta interconexão recebe fluxos do tipo ‘A’, ao passo que uma Porta conjugada igualmente tipificada por esta mesma interconexão fornece fluxos do tipo ‘A’. Já as interconexões “Conector 2” e “Conector 3” possuem cada uma um Fluxo de Propriedade chamado “Fluxo” do tipo ‘B’ e ‘C’, respectivamente. Nestes casos, estes são fluxos de saída devido ao classificador ‘*out’* antecedendo seus nomes. Logo, Portas tipificadas por estas interconexões fornecem fluxos do tipo ‘B’ ou ‘C’, dependendo da interconexão utilizada na sua tipificação, e Portas conjugadas igualmente tipificadas por estas mesmas interconexões recebem fluxos do tipo ‘B’ ou ‘C’, dependendo da interconexão utilizada na sua tipificação. Finalmente, a interconexão “Conector 4” possuem um Fluxo de Propriedade chamado “Fluxo” e tipificado por um Bloco chamado “D”. Este é um fluxo tanto de entrada como de saída, verificado pelo classificador ‘*inout’* antecedendo seu

nome. Neste caso Portas e Portas conjugadas tipificadas por esta interconexão recebem e fornecem fluxos do tipo ‘D’. A diferença fica a cargo do dinamismo na troca de fluxos entre dois módulos através destas Portas. Em um dado momento, enquanto a Porta fornece o fluxo, a Porta conjugada a ela conectada o recebe e vice-versa.

Outra informação que se pode extrair do diagrama apresentado na Figura 4.27 acima é o relacionamento entre as interconexões definidas para o sistema e os módulos de arquitetura por elas responsáveis. Para isso, basta verificar as Portas dos módulos de arquitetura tipificadas pelas interconexões definidas para o sistema, no diagrama da Figura 4.24.

A partir deste ponto, o engenheiro de sistemas começará a decomposição do sistema em termos de sua arquitetura interna. Esta decomposição mostrará os módulos de arquitetura existentes e os fluxos de arquitetura e mensagens por eles trocados. O engenheiro de sistemas utilizará, agora, Diagramas Internos de Bloco para o Bloco representando o sistema com o objetivo de mostrar a troca de fluxos de arquitetura entre seus diferentes módulos de arquitetura. Não existe limite para a quantidade de diagramas deste tipo, ficando a cargo do engenheiro de sistemas escolher a melhor forma de representação do modelo da arquitetura interna do sistema. Na Figura 4.28 abaixo, apresenta-se um exemplo de diagrama de fluxos de arquitetura através de um Diagrama Interno de Bloco para o Bloco representando o sistema, conforme nota-se pelo seu cabeçalho. Por se tratar de um Diagrama Interno de Bloco da SysML, as bordas deste tipo de diagrama representam, na verdade, as fronteiras do Bloco sendo abordado, no caso, o Bloco representando o sistema propriamente dito. Portanto, nas bordas deste diagrama, têm-se as Portas da SysML tipificadas pelas interconexões definidas para o sistema, vistas no diagrama de contexto de interconexões da Figura 4.24. Dentro das bordas do diagrama, encontram-se os módulos de arquitetura do sistema, vistos no diagrama de interconexões de arquitetura da Figura 4.27.

Figura 4.28 – Exemplo de Fluxo de Arquitetura em SysML



Fonte: Produção do Autor.

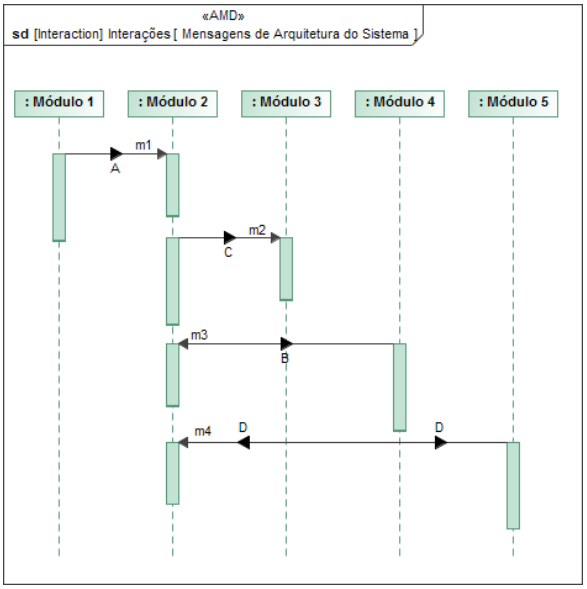
Os módulos de arquitetura responsáveis pelas interfaces com as entidades externas ao sistema possuem Portas igualmente tipificadas pelas interconexões do sistema e relacionadas com as Portas nas bordas do diagrama através de um Conector da SysML, cujo estereótipo <<*equal*>> indica ser, na verdade, um conector de delegação. Ou seja, a Porta na borda do diagrama foi delegada a um dado módulo de arquitetura do sistema.

Já entre os módulos de arquitetura do sistema, Conectores da SysML relacionam as Portas destes módulos de acordo com as interconexões definidas no diagrama da Figura 4.27.

Os fluxos de arquitetura são, novamente, representados por Fluxos de Item da SysML sendo realizados pelos correspondentes Conectores e tendo o desenho da seta definido pelo sentido em que o fluxo efetivamente flui. Mais ainda, na Figura 4.28 acima, têm-se novamente as Portas conjugadas dos módulos de arquitetura, conforme discutido na Figura 4.27, e os fluxos de arquitetura tipificados pelos mesmos tipos utilizados nos Fluxos de Propriedade presentes nas correspondentes interconexões. Aqui, no entanto, se enfatiza o sentido destes fluxos, que devem estar condizentes com os classificadores ‘*in*’, ‘*out*’ ou ‘*inout*’ dos correspondentes Fluxos de Propriedade das interconexões.

Novamente, o engenheiro de sistemas tem à sua disposição uma outra possibilidade para representar trocas internas de informações de arquitetura. Esta possibilidade envolve o uso de mensagens entre os módulos de arquitetura identificados. Para isso, será novamente utilizado o Diagrama de Sequência da SysML. Na Figura 4.29 abaixo, apresenta-se um exemplo de diagrama de mensagens de arquitetura.

Figura 4.29 – Exemplo de Mensagens de Arquitetura em SysML

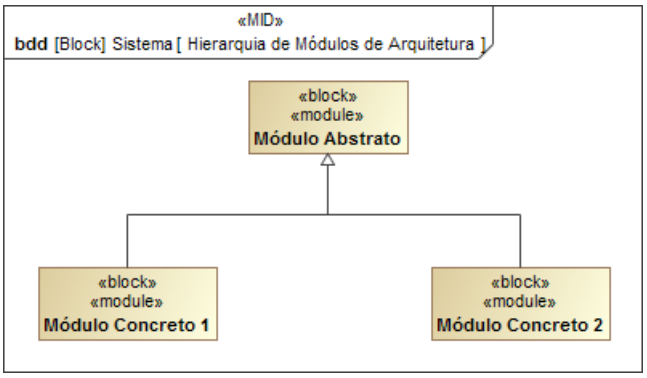


Fonte: Produção do Autor.

Na figura acima, de forma análoga como feito no Diagrama de Sequência da Figura 4.26, notam-se as Linhas de Vida representando os diferentes módulos de arquitetura do sistema e Mensagens da SysML sendo trocadas entre eles, transportando os fluxos de arquitetura identificados anteriormente no diagrama da Figura 4.28.

Por último, o engenheiro de sistemas pode escolher por uma representação de hierarquia entre os módulos de arquitetura do sistema. Isso é feito principalmente para agrupar características comuns aos módulos de arquitetura de um dado nível de abstração. Na Figura 4.30 abaixo, apresenta-se um exemplo de diagrama de herança de módulos através de um Diagrama de Definição de Bloco.

Figura 4.30 – Exemplo de Herança de Módulos em SysML



Fonte: Produção do Autor.

No diagrama acima, mostram-se dois módulos de arquitetura, aqui chamados de “Módulo Concreto 1” e “Módulo Concreto 2”, herdando características de um módulo de arquitetura abstrato chamado de “Módulo Abstrato”.

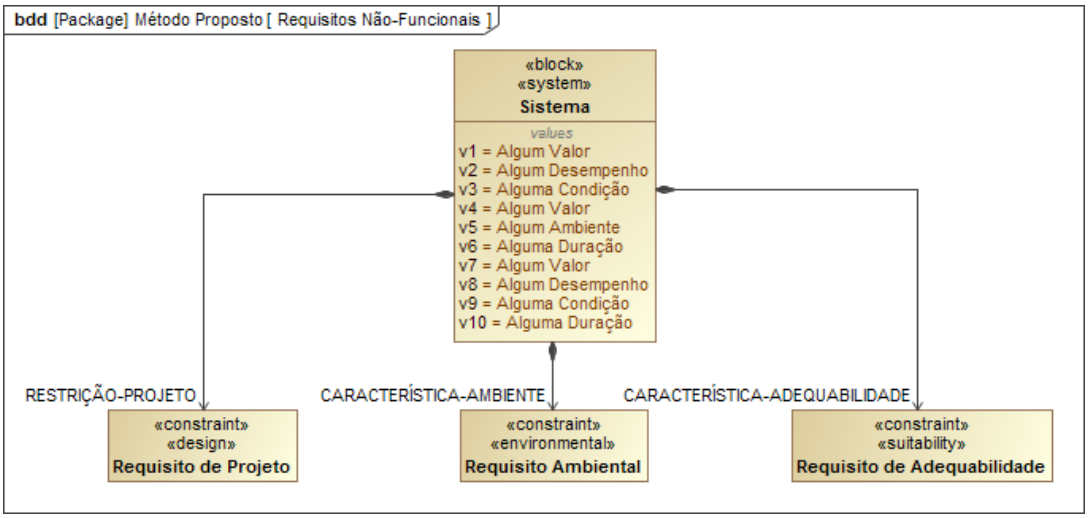
4.6.4.3 Dicionário de Arquitetura

Conforme visto anteriormente na Seção 4.3.3, o dicionário de arquitetura é construído seguindo a mesma linha utilizada para o dicionário de requisitos. Ou seja, todas as construções apresentas na Seção 4.6.2.4, para construção do dicionário de requisitos utilizando a SysML serão igualmente utilizadas para construção do dicionário de arquitetura em SysML. Nota-se, entretanto, que os tipos a serem especificados neste último dicionário representarão as tipificações utilizadas para os fluxos de arquitetura (i.e. Fluxos de Item da SysML), interconexões (i.e. Portas da SysML), etc. Em outras palavras, serão os tipos para as definições da arquitetura do sistema e não mais para as definições de funcionalidades do sistema.

4.6.4.4 Modelo de Restrições de Projeto, Ambientais e de Adequabilidade

Na Figura 4.31 abaixo, mostra-se um exemplo didático, onde o engenheiro de sistemas criou, no Bloco representando o sistema, uma característica para cada tipo de requisitos não-funcional proposto por Carson (2015).

Figura 4.31 – Exemplo de BDD com Requisitos Não-Funcionais



Fonte: Produção do Autor.

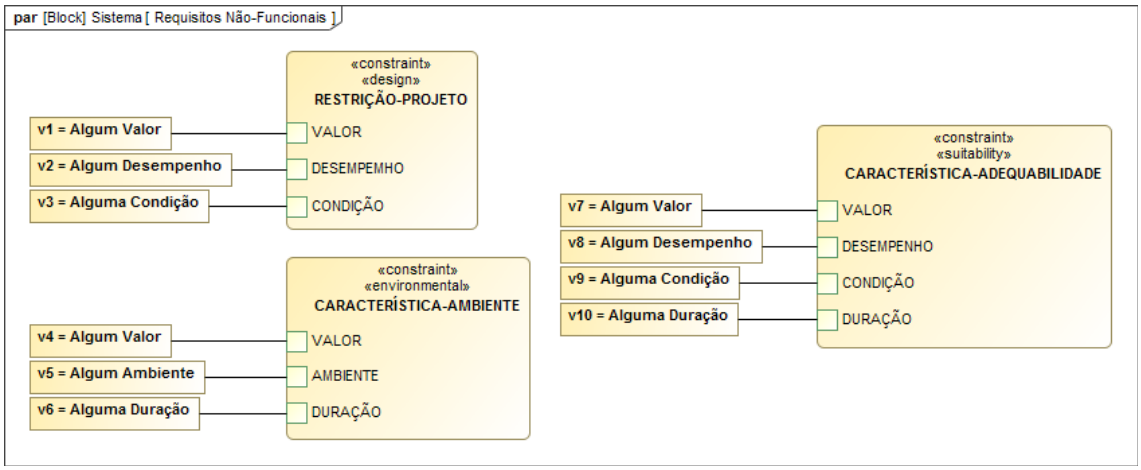
Na figura acima, tem-se ao centro o Bloco representando o sistema sendo modelado. Nele, existe novamente uma linha horizontal delimitando um compartimento nomeado como "*values*". Este nome é definido na especificação da SysML e define o compartimento contendo os elementos do tipo Propriedade de Valor do correspondente Bloco. Foram criadas dez Propriedades de Valor, nomeadas de "v1" a "v10", e que possuem um valor padrão atribuído, apresentado logo após o sinal de igual. Estes elementos serão utilizados mais adiante para preenchimento das lacunas dos textos dos requisitos não-funcionais.

Além destas Propriedades de Valor, existe um relacionamento de composição entre o Bloco representando o sistema e os Blocos de Restrição representando os tipos de requisitos não-funcionais propostos por Carson (2015). Estes relacionamentos implicam na criação de Propriedades de Restrição, tipificadas pelos correspondentes Blocos de Restrição. Os nomes destas Propriedades de Restrição são escolhidos pelo engenheiro de sistemas e estão localizados, no diagrama da Figura 4.31 acima, ao lado das setas dos relacionamentos de composição.

Agora que as características dos requisitos não-funcionais do sistema foram criadas, o engenheiro de sistemas deve mostrar quais Propriedades de Valor preenchem quais lacunas dos textos, estas representadas pelos Parâmetros

de Restrição dos Blocos de Restrição correspondentes aos requisitos não-funcionais em questão. Para isso, o engenheiro de sistemas utilizará um Diagrama Paramétrico da SysML para o Bloco que representa o sistema sendo modelado, cujo exemplo é mostrado na Figura 4.32 abaixo.

Figura 4.32 – Exemplo de Paramétrico com Campos dos Requisitos Não-Funcionais



Fonte: Produção do Autor.

Neste exemplo, têm-se as Propriedades de Valor anteriormente criadas para o Bloco do sistema, representadas por pequenos retângulos de cantos retos. Dentro destes retângulos, tem-se o nome da correspondente Propriedade de Valor e seu valor padrão, atribuído anteriormente pelo engenheiro de sistemas. Representadas por retângulos maiores e de cantos arredondados, têm-se as Propriedades de Restrição anteriormente criadas para o Bloco do sistema. Dentro destes retângulos, têm-se os Parâmetros de Restrição, representados por pequenos quadrados na borda das correspondentes Propriedades de Restrição. O engenheiro de sistemas, então, cria um relacionamento entre uma dada Propriedade de Valor e um dado Parâmetro de Restrição para preencher a correspondente lacuna do texto do requisito não-funcional em questão.

Como isso, o modelo de arquitetura do sistema está completo.

4.6.5 Relacionando os Modelos de Requisitos e de Arquitetura

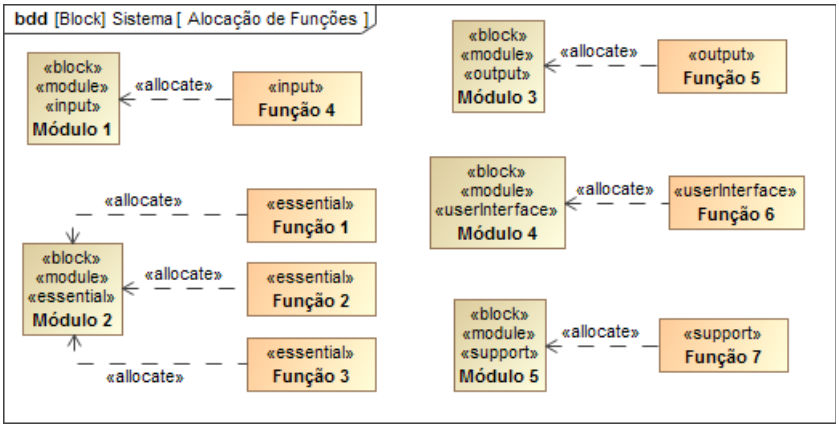
A especificação da SysML prevê um relacionamento dedicado à alocação de elementos em um dado nível de abstração a elementos em outro nível de

abstração, cabendo ao engenheiro de sistemas responsável pelo modelo definir a semântica por trás destes relacionamentos.

No caso da metodologia de modelagem de um sistema utilizando a SysML proposta neste capítulo, este relacionamento de alocação será utilizado para mapear o modelo de requisitos no modelo de arquitetura.

As Atividades representando as funções terminais (i.e. de mais baixo nível da decomposição funcional) criadas no modelo de requisitos devem ser alocadas tanto ao Bloco representando o sistema de interesse, como aos Blocos representando os módulos de mais baixo nível da arquitetura do sistema. Na Figura 4.33 abaixo, mostra-se um Diagrama de Definição de Bloco utilizado para criação das alocações das funções identificadas no modelo de requisitos descrito na Seção 4.6.2 aos módulos de arquitetura identificados no modelo de arquitetura descrito na Seção 4.6.4.

Figura 4.33 – Exemplo de BDD com Alocações de Funções



Fonte: Produção do Autor.

Na figura acima, tem-se um BDD utilizado para a única finalidade de criação dos relacionamentos de alocação das funções aos módulos de arquitetura. As linhas tracejadas direcionadas e com o estereótipo <<*allocate*>> aplicado representam o relacionamento de Alocação da SysML. Este relacionamento impõe uma semântica onde o elemento na extremidade de origem é alocado ao elemento na extremidade de destino, considerando o sentido estabelecido pela seta. Como visto ao longo da Seção 4.6, as funções são representadas por Atividades da SysML com estereótipos identificando o tipo de função, ou

seja, se ela é de entrada, saída, interface com usuário ou suporte. Já os módulos de arquitetura são representados por elementos do tipo Bloco da SysML com estereótipos identificando o tipo de módulo de arquitetura, que segue a mesma classificação utilizada para funções. Por motivos de coerência, Blocos de um dado tipo devem alocar funções do mesmo tipo, assim como se vê na figura acima.

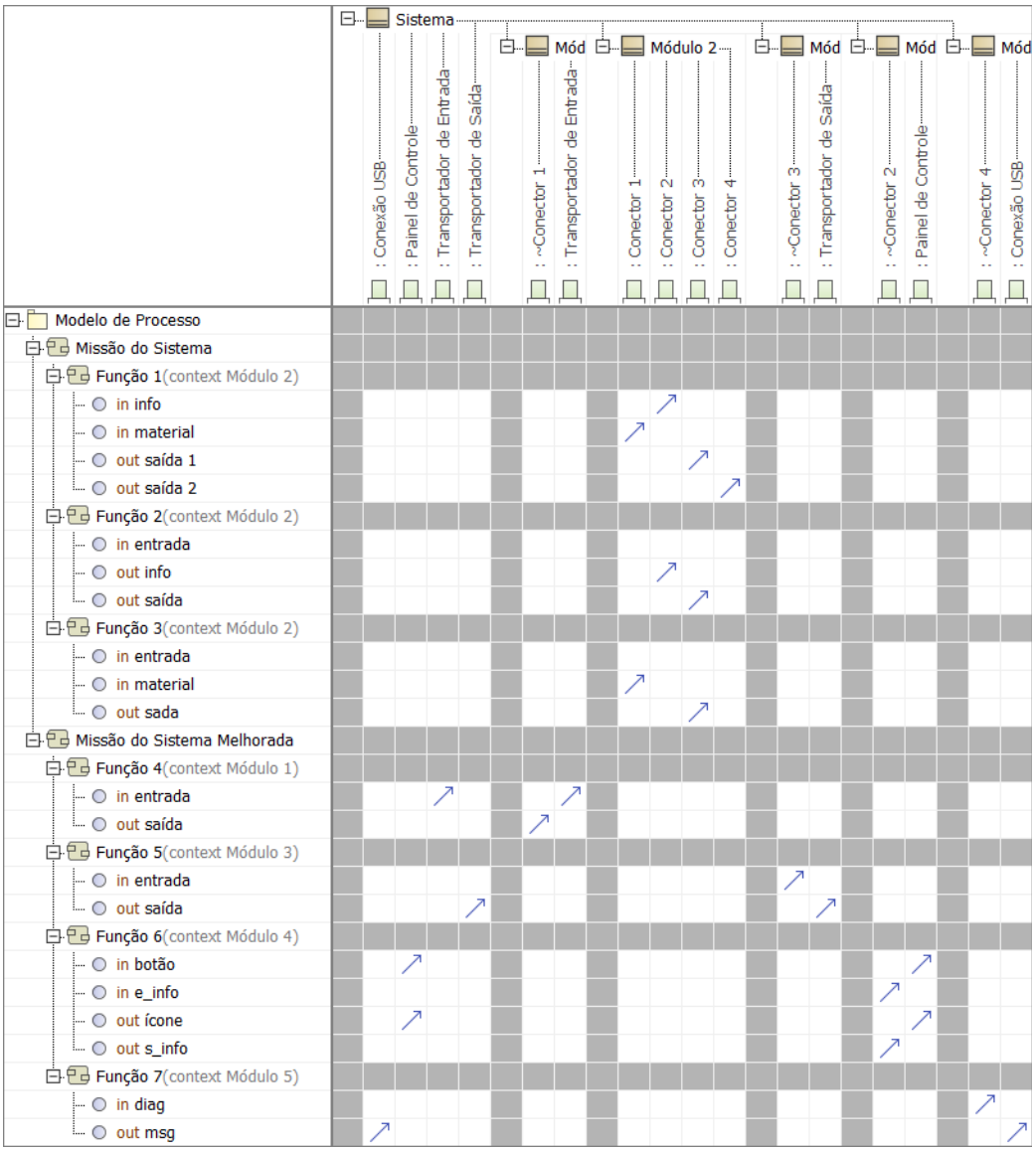
No exemplo utilizado na Figura 4.33 acima, cada função foi alocada para um único módulo de arquitetura. Isto sempre deve ser observado. Caso uma dada função tenha sido alocada para mais de um módulo de arquitetura, o engenheiro de sistemas deverá continuar o processo de desdobramento até que se chegue em subfunções passíveis de serem alocadas para um único módulo de arquitetura.

Para adicionar a informação de observabilidade de uma dada função, conforme orientado por Carson (2015), o engenheiro de sistemas deve alocar os diferentes parâmetros da respectiva Atividade tanto para as Portas do Bloco representando o sistema, como para as Portas do Bloco representando o módulo de arquitetura que aloca a função em questão.

Na Figura 4.34 abaixo, mostra-se uma Matriz de Alocação da SysML utilizada para criação e visualização das alocações dos parâmetros das Atividades representando as funções terminais criadas no modelo para as Portas tanto do Bloco representando o sistema quanto para as Portas dos Blocos representando os módulos de arquitetura do sistema. Na figura, tem-se uma outra opção disponibilizada por ferramentas de modelagem em SysML para criação de relacionamentos de alocação. O engenheiro de sistemas cria um elemento deste tipo e fornece os parâmetros necessários para possibilitar a ferramenta montar a matriz (e.g.: tipos dos elementos das linhas e colunas, tipo de relacionamento entre os elementos das linhas e colunas, sentido deste relacionamento, etc). Após sua criação, o engenheiro de sistemas pode selecionar uma célula da matriz e solicitar a criação ou remoção do relacionamento. No caso da figura acima, foi criada uma matriz com as linhas contendo os parâmetros das Atividades criadas no modelo de requisitos

essenciais e no modelo de requisitos melhorados (Seção 4.6.2 e Seção 4.6.3, respectivamente) e com as colunas contendo as Portas criadas no modelo de arquitetura (Seção 4.6.4). As células contendo um desenho de uma seta na diagonal representam relacionamentos de Alocação, onde o elemento de uma linha está alocado ao correspondente elemento da coluna.

Figura 4.34 – Exemplo de Matriz de Alocação de Parâmetros para Portas



Fonte: Produção do Autor.

4.7 Gerando Requisitos a partir do Modelo SysML

Esta seção está organizada da seguinte maneira. Para cada tipo de requisito sendo considerado na metodologia aqui proposta, identificado pela sua

correspondente atividade apresentada anteriormente na Figura 4.6, existe uma subseção abaixo.

Nestas subseções, os enunciados dos correspondentes requisitos terão suas sintaxes definidas utilizando uma notação baseada na notação EBNF (*Extended Backus-Naur Form* – ISO, 1996), com as seguintes distinções:

- Os símbolos que serão subsequentemente definidos estarão destacados em maiúsculo e negrito para facilitar a leitura;
- Textos livres serão utilizados para definir os símbolos terminais. Para um dado símbolo terminal, este texto livre conterá uma regra para encontrar um elemento do modelo do sistema, cuja alguma propriedade será utilizada no enunciado do correspondente requisito (e.g. nome do elemento). A propriedade do elemento a ser utilizada no enunciado do requisito será igualmente referenciada no texto livre que define o correspondente símbolo terminal;
- O símbolo de definição será ":=" e não "=" como definido por ISO (1996) para facilitar a leitura quando um enunciado de um requisito já contiver o símbolo "=".

Os enunciados dos requisitos funcionais relativos a comportamentos referenciados por Transições da SysML ou por Estados da SysML, ambos presentes na especificação de controle, são adaptações do padrão para requisitos funcionais definido por Lempia et al. (2016).

Já os enunciados dos requisitos dos tipos 'Restrições de Projeto', 'Restrições Ambientais' e 'Restrições de Adequabilidade' são adaptações dos respectivos padrões definidos por Carson (2015).

Os demais enunciados de requisitos aqui propostos são de produção do próprio autor desta Tese de Doutorado.

4.7.1. Gerar Requisitos Funcionais de Transição entre Estados e Modos Operacionais

Para cada Transição da SysML definida na especificação de controle, deve-se gerar um requisito funcional com o seguinte enunciado:

Enunciado := “Se ”, **GATILHO**, “ e o ”, **OBJETO**, “ estiver no ”, [**ESTADO_ORIGEM** | **MODO_ORIGEM**], “, o ” **OBJETO**, “ deve mudar para o ”, [**ESTADO_DESTINO** | **MODO_DESTINO**];
GATILHO := **EVENTO**, [**CONDIÇÃO**];
EVENTO := Evento associado à Transição da SysML em questão;
CONDIÇÃO := Condição associada à Transição da SysML em questão;
OBJETO := Bloco com estereótipo <<*system*>>;
ESTADO_ORIGEM := “estado ”, **NOME_ESTADO_ORIGEM**;
NOME_ESTADO_ORIGEM := Nome do elemento Estado da SysML com estereótipo <<State>> que é origem da Transição em questão;
MODO_ORIGEM := “modo operacional ”, **NOME_MODO_ORIGEM**, “ do estado ”, **NOME_ESTADO_MODO_ORIGEM**;
NOME_MODO_ORIGEM := Nome do elemento Estado da SysML com estereótipo <<Mode>> que é origem da Transição em questão;
NOME_ESTADO_MODO_ORIGEM := Nome do elemento Estado da SysML com estereótipo <<State>> contendo o elemento Estado da SysML com o estereótipo <<Mode>> que é origem da Transição em questão;
ESTADO_DESTINO := “estado ”, **NOME_ESTADO_DESTINO**;
NOME_ESTADO_DESTINO := Nome do elemento Estado da SysML com estereótipo <<State>> que é destino da Transição em questão;
MODO_DESTINO := “modo operacional ”, **NOME_MODO_DESTINO**, “ do estado ”, **NOME_ESTADO_MODO_DESTINO**;
NOME_MODO_DESTINO := Nome do elemento Estado da SysML com estereótipo <<Mode>> que é destino da Transição em questão;
NOME_ESTADO_MODO_DESTINO := Nome do elemento Estado da SysML com estereótipo <<State>> contendo o elemento Estado da SysML com o estereótipo <<Mode>> que é destino da Transição em questão;

A presença ou não do símbolo **CONDIÇÃO** está baseada no preenchimento ou não da respectiva propriedade da Transição da SysML sendo analisada.

Já escolha entre as opções dos símbolos **ESTADO_ORIGEM** e **ESTADO_DESTINO** ou **MODO_ORIGEM** e **MODO_DESTINO** no enunciado do requisito está baseada nos elementos Estado da SysML, respectivamente, de origem e de destino da Transição sendo analisada. Caso estes elementos tenham o estereótipo <<State>>, os símbolos **ESTADO_ORIGEM** e **ESTADO_DESTINO** serão utilizados. Por outro lado, caso os elementos em questão tenham o estereótipo <<Mode>>, os símbolos **MODO_ORIGEM** e **MODO_DESTINO** serão utilizados.

4.7.2. Gerar Requisitos Funcionais de Transição de Estado ou Modo Operacional

Para cada Transição da SysML definida na especificação de controle e que referencie um elemento do tipo Atividade da SysML, deve-se gerar um correspondente requisito funcional com o seguinte enunciado:

Enunciado := “Se ”, **GATILHO**, “ e o ”, **OBJETO**, “ estiver no ”, [**ESTADO** | **MODO_OPERACIONAL**], “, o ” **OBJETO**, “ deve ”, **COMPORTAMENTO**, [**DADO_INTERFACE_SAÍDA**], [**DESEMPENHO**], [**DADO_INTERFACE_ENTRADA**];
GATILHO := **EVENTO**, [**CONDIÇÃO**];
EVENTO := Nome do elemento do modelo referenciado pelo campo Evento da Transição da SysML em questão;
CONDIÇÃO := “ e ”, **TEXTO_CONDIÇÃO**;
TEXTO_CONDIÇÃO := Texto do campo Condição da Transição da SysML em questão;
OBJETO := Bloco com estereótipo <<*system*>> ao qual foi alocada a Atividade referenciada pela Transição em questão;
ESTADO := “estado ”, **NOME_ESTADO**;
NOME_ESTADO := Nome do elemento Estado da SysML com estereótipo <<State>> que é origem da Transição em questão;

MODO_OPERACIONAL := “modo operacional ”, **NOME_MODO_OPERACIONAL**, “ do estado ”, **NOME_ESTADO_MODO**;
NOME_MODO_OPERACIONAL := Nome do elemento Estado da SysML com estereótipo <<Mode>> que é origem da Transição em questão;
NOME_ESTADO_MODO := Nome do elemento Estado da SysML com estereótipo <<State>> contendo o elemento Estado da SysML com o estereótipo <<Mode>> que é origem da Transição em questão;
COMPORTAMENTO := Nome do elemento Atividade da SysML referenciada pela Transição em questão;
DADO_INTERFACE_SAÍDA := {**NOME_DADO_SAÍDA**, [“ de acordo com ”, **NOME_INTERFACE_SAÍDA**]};
NOME_DADO_SAÍDA := Nome do elemento Parâmetro pertencente à Atividade da SysML referenciada pela Transição em questão e que tenha sido classificado como um parâmetro de saída;
NOME_INTERFACE_SAÍDA := Nome do elemento do modelo que tipifica a Porta da SysML para a qual foi alocado o Parâmetro pertencente à Atividade da SysML referenciada pela Transição em questão e cujo nome é igual a **NOME_DADO_SAÍDA**;
DESEMPENHO := Expressão do elemento Restrição da SysML relacionado à Atividade da SysML referenciada pela Transição em questão;
DADO_INTERFACE_ENTRADA := {“usando ”, **NOME_DADO_ENTRADA**, [“ de acordo com ”, **NOME_INTERFACE_ENTRADA**]};
NOME_DADO_ENTRADA := Nome do elemento Parâmetro pertencente à Atividade da SysML referenciada pela Transição em questão e que tenha sido classificado como um parâmetro de entrada;
NOME_INTERFACE_ENTRADA := Nome do elemento do modelo que tipifica a Porta da SysML para a qual foi alocado o Parâmetro pertencente à Atividade da SysML referenciada pela Transição em questão e cujo nome é igual a **NOME_DADO_ENTRADA**;

A presença ou não do símbolo **CONDIÇÃO** está baseada no preenchimento ou não da respectiva propriedade da Transição da SysML sendo analisada.

A escolha entre as opções dos símbolos **ESTADO** ou **MODO_OPERACIONAL** no enunciado do requisito está baseada no elemento Estado da SysML que é origem da Transição sendo analisada. Caso este elemento tenha o estereótipo <<State>>, o símbolo **ESTADO** será utilizado. Por outro lado, caso o elemento em questão tenha o estereótipo <<Mode>>, o símbolo **MODO_OPERACIONAL** será utilizado.

A repetição do símbolo **DADO_INTERFACE_SAÍDA** está baseada na quantidade de Parâmetros de saída da Atividade referenciada pela Transição em questão, que pode ser zero. Mais ainda, a presença ou não do símbolo **NOME_INTERFACE_SAÍDA** está condicionada à alocação dos Parâmetros de saída da Atividade referenciada pela Transição em questão a Portas do Bloco representando o sistema. O mesmo pode ser considerado para o símbolo **DADO_INTERFACE_ENTRADA**.

A presença ou não do símbolo **DESEMPENHO** está baseada no relacionamento de um elemento Restrição à Atividade referenciada pela Transição da SysML sendo analisada.

4.7.3. Gerar Requisitos Funcionais de Estado ou Modo Operacional

Para cada Estado da SysML definido na especificação de controle e que referencie um elemento do tipo Atividade da SysML, deve-se gerar um correspondente requisito funcional com o seguinte enunciado:

Enunciado := “Se ”, **GATILHO**, “, o ” **OBJETO**, “ deve ”, **COMPORTAMENTO**, [**DADO_INTERFACE_SAÍDA**], [**DESEMPENHO**], [**DADO_INTERFACE_ENTRADA**], [**ESTADO** | **MODO_OPERACIONAL**];
GATILHO := **CAMINHO_TRANSIÇÕES**, [{“ ou ”, **CAMINHO_TRANSIÇÕES**}];
CAMINHO_TRANSIÇÕES := **TRANSIÇÃO**, [{“ e ”, **TRANSIÇÃO**}]
TRANSIÇÃO := **EVENTO**, [**CONDIÇÃO**];
EVENTO := Nome do elemento do modelo referenciado pelo campo Evento da Transição da SysML que pertence a um caminho de transições do

pseudo-estado inicial global da especificação de controle até o Estado da SysML em questão;

CONDIÇÃO := “ e ”, **TEXTO_CONDIÇÃO**;

TEXTO_CONDIÇÃO := Texto do campo Condição da Transição da SysML que pertence a um caminho de transições do pseudo-estado inicial global da CSPEC até o Estado da SysML em questão;

OBJETO := Bloco com estereótipo <<*system*>> ao qual foi alocada a Atividade referenciada pelo Estado em questão;

COMPORTAMENTO := Nome do elemento Atividade da SysML referenciada pelo Estado em questão;

DADO_INTERFACE_SAÍDA := {**NOME_DADO_SAÍDA**, [“ de acordo com ”, **NOME_INTERFACE_SAÍDA**]};

NOME_DADO_SAÍDA := Nome do elemento Parâmetro pertencente à Atividade da SysML referenciada pelo Estado em questão e que tenha sido classificado como um parâmetro de saída;

NOME_INTERFACE_SAÍDA := Nome do elemento do modelo que tipifica a Porta da SysML para a qual foi alocado o Parâmetro pertencente à Atividade da SysML referenciada pelo Estado em questão e cujo nome é igual a **NOME_DADO_SAÍDA**;

DESEMPENHO := Expressão do elemento Restrição da SysML relacionado à Atividade da SysML referenciada pelo Estado em questão;

DADO_INTERFACE_ENTRADA := {“usando ”, **NOME_DADO_ENTRADA**, [“ de acordo com ”, **NOME_INTERFACE_ENTRADA**]};

NOME_DADO_ENTRADA := Nome do elemento Parâmetro pertencente à Atividade da SysML referenciada pelo Estado em questão e que tenha sido classificado como um parâmetro de entrada;

NOME_INTERFACE_ENTRADA := Nome do elemento do modelo que tipifica a Porta da SysML para a foi alocado qual o Parâmetro pertencente à Atividade da SysML referenciada pelo Estado em questão e cujo nome é igual a **NOME_DADO_ENTRADA**;

ESTADO := [“ ao entrar em ” | “ enquanto em ” | “ ao sair de ”], “qualquer modo operacional do estado ”, **NOME_ESTADO**;

NOME_ESTADO := Nome do elemento Estado da SysML em questão;
MODO_OPERACIONAL := [“ ao entrar no ” | “ enquanto no ” | “ ao sair do ”], “modo operacional ”, **NOME_MODO_OPERACIONAL**, “ do estado ”, **NOME_ESTADO_MODO**;
NOME_MODO_OPERACIONAL := Nome do elemento Estado da SysML em questão;
NOME_ESTADO_MODO := Nome do elemento Estado da SysML com estereótipo <<State>> contendo o elemento Estado da SysML em questão;

O símbolo **GATILHO** representa uma lógica ‘OU’ entre os diferentes caminhos de transições que levam do pseudo-estado inicial global da especificação de controle até um dado Estado. Mais ainda, um dado caminho de transições, que leva do pseudo-estado inicial global da especificação de controle até este mesmo Estado, representa uma lógica ‘E’ entre os Eventos e Condições das Transições que o compõe.

A presença ou não do símbolo **CONDIÇÃO** está baseada no preenchimento ou não da respectiva propriedade da Transição da SysML sendo analisada.

A repetição do símbolo **DADO_INTERFACE_SAÍDA** está baseada na quantidade de Parâmetros de saída da Atividade referenciada pelo Estado em questão, que pode ser zero. Mais ainda, a presença ou não do símbolo **NOME_INTERFACE_SAÍDA** está condicionada à alocação dos Parâmetros de saída da Atividade referenciada pelo Estado em questão a Portas do Bloco representando o sistema. O mesmo pode ser considerado para o símbolo **DADO_INTERFACE_ENTRADA**.

A presença ou não do símbolo **DESEMPENHO** está baseada no relacionamento de um elemento Restrição à Atividade referenciada pelo Estado da SysML sendo analisada.

A escolha entre as opções dos símbolos **ESTADO** ou **MODO_OPERACIONAL** no enunciado do requisito está baseada no elemento Estado da SysML sendo analisado. Caso este elemento tenha o estereótipo <<State>>, o símbolo **ESTADO** será utilizado. Por outro lado,

caso o elemento em questão tenha o estereótipo <<Mode>>, o símbolo **MODO_OPERACIONAL** será utilizado.

A escolha entre as opções de advérbio de tempo “ao entrar em(no)”, “enquanto em(no)” e “ao sair de(do)” no enunciado é feita de acordo com o compartimento do Estado em questão, utilizado para referenciar o elemento Atividade. No caso da utilização do compartimento “*entry*”, deve-se utilizar o advérbio “ao entrar em(no)”. Já no caso de utilizar o compartimento “*do*”, deve-se, então, utilizar o advérbio “enquanto em(no)”. Finalmente, caso utilize-se o compartimento “*exit*”, deve-se, consequentemente, utilizar o advérbio “ao sair de(do)”.

4.7.4. Gerar Requisitos Não-Funcionais de Estados e Modos Operacionais

A partir da especificação de controle, para cada elemento Estado da SysML utilizado, deve-se gerar requisitos não-funcionais para que o sistema apresente os estados e modos operacionais especificados, utilizando o seguinte enunciado:

Enunciado := “O ”, **OBJETO**, “ deve apresentar o ”, [**ESTADO** | **MODO_OPERACIONAL**];
OBJETO := Bloco com estereótipo <<*system*>>;
ESTADO := “estado ”, **NOME_ESTADO;**
NOME_ESTADO := Nome do elemento Estado da SysML em questão;
MODO_OPERACIONAL := “modo operacional ”, **NOME_MODO_OPERACIONAL**, **NO_ESTADO_DO_MODO**;
NOME_MODO_OPERACIONAL := Nome do elemento Estado da SysML em questão;
NO_ESTADO_DO_MODO := “ no estado ”, **ESTADO_DO_MODO**;
ESTADO_DO_MODO := Nome do elemento Estado da SysML com estereótipo <<State>> que contém o elemento Estado da SysML em questão;

A escolha entre as opções dos símbolos **ESTADO** ou **MODO_OPERACIONAL** no enunciado do requisito está baseada no

elemento Estado da SysML sendo analisado. Caso este elemento tenha o estereótipo <<State>>, o símbolo **ESTADO** será utilizado. Por outro lado, caso o elemento em questão tenha o estereótipo <<Mode>>, o símbolo **MODO_OPERACIONAL** será utilizado.

4.7.5. Gerar Requisitos Não-Funcionais de Estado e Modo Operacional Iniciais

A partir da especificação de controle, gera-se requisitos não-funcionais para os estados e modos operacionais inciais do sistema. Portanto, para cada pseudo-estado inicial definido na especificação de controle, gera-se um requisito não-funcional utilizando um dos dois enunciados abaixo:

Enunciado := “O ”, **OBJETO**, “ deve iniciar no ”, [**ESTADO_INICIAL** | **MODO_INICIAL**];
OBJETO := Bloco com estereótipo <<*system*>>;
ESTADO_INICIAL := “estado ”, **NOME_ESTADO_INICIAL**;
NOME_ESTADO_INICIAL := Nome do elemento Estado da SysML com estereótipo <<State>> relacionado ao pseudo-estado inicial;
MODO_INICIAL := “modo operacional ”, **NOME_MODO_INICIAL**, “ quando entrar no estado ”, **ESTADO_DO_MODO_INICIAL**;
NOME_MODO_INICIAL := Nome do elemento Estado da SysML com estereótipo <<Mode>> relacionado ao pseudo-estado inicial;
ESTADO_DO_MODO_INICIAL := Nome do elemento Estado da SysML com estereótipo <<State>> que contém o elemento Estado da SysML com estereótipo <<Mode>> relacionado ao pseudo-estado inicial;

A escolha entre as opções dos símbolos **ESTADO_INICIAL** ou **MODO_INICIAL** no enunciado do requisito está baseada no elemento Estado da SysML relacionado ao pseudo-estado inicial sendo analisado. Caso este elemento tenha o estereótipo <<State>>, o símbolo **ESTADO_INICIAL** será utilizado. Por outro lado, caso o elemento em questão tenha o estereótipo <<Mode>>, o símbolo **MODO_INICIAL** será utilizado.

4.7.6. Gerar Requisitos Não-Funcionais de Interfaces

Para cada Porta da SysML criada para o Bloco com o estereótipo <<*system*>>, gera-se um requisito com o seguinte enunciado:

Enunciado := “O ”, **OBJETO**, “ deve ser compatível com a ”, **INTERFACE**;
OBJETO := Bloco com estereótipo <<*system*>>;
INTERFACE := “interface ”, **NOME_INTERFACE;**
NOME_INTERFACE := Nome do elemento do modelo que tipifica a Porta da SysML em questão;

4.7.7. Gerar Restrições de Projeto

Para cada Propriedade de Restrição da SysML criada para o Bloco com o estereótipo <<*system*>> e tipificada por um Bloco de Restrição contendo o estereótipo <<*design*>>, gera-se um requisito com o seguinte enunciado:

Enunciado := “O ”, **OBJETO**, “ deve ”, [“exibir ” | “apresentar ”], **RESTRIÇÃO_PROJETO**, **DESEMPENHO_CONDIÇÃO**;
OBJETO := Bloco com estereótipo <<*system*>>;
RESTRIÇÃO_PROJETO := **NOME_RESTRIÇÃO**, [**IGUAL_VALOR_RESTRIÇÃO**];
IGUAL_VALOR_RESTRIÇÃO := “ = ”, **VALOR_RESTRIÇÃO**;
DESEMPENHO_CONDIÇÃO := [**DESEMPENHO**], [**CONDIÇÃO**];
DESEMPENHO := “ de acordo com ”, **VALOR_DESEMPENHO**;
CONDIÇÃO := “ enquanto na ”, **VALOR_CONDIÇÃO**;
NOME_RESTRIÇÃO := Nome da Propriedade de Restrição da SysML em questão;
VALOR_RESTRIÇÃO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “VALOR” da Propriedade de Restrição em questão;
VALOR_DESEMPENHO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “DESEMPENHO” da Propriedade de Restrição em questão;

VALOR_CONDIÇÃO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “CONDIÇÃO” da Propriedade de Restrição em questão;

A escolha do verbo principal do enunciado pode ser feita para tornar a frase mais correta de um ponto de vista semântico. Dois verbos foram colocados como opção (“exibir” e “apresentar”), entretanto, outros poderiam ser utilizados para uma melhor apresentação do enunciado do requisito em questão.

Já a presença ou não dos símbolos **IGUAL_VALOR_RESTRIÇÃO**, **DESEMPENHO** e **CONDIÇÃO** no enunciado está relacionada ao preenchimento dos respectivos Parâmetros de Restrição da Propriedade de Restrição em questão.

4.7.8. Gerar Restrições Ambientais

Para cada Propriedade de Restrição da SysML criada para o Bloco com o estereótipo <<*system*>> e tipificada por um Bloco de Restrição contendo o estereótipo <<*environmental*>>, gera-se um requisito com o seguinte enunciado:

Enunciado := “O ”, **OBJETO**, “ deve ”, [“exibir ” | “apresentar ”], **CARACTERÍSTICA**, [“ durante” | “ depois”], **AMBIENTE_DURAÇÃO**;
OBJETO := Bloco com estereótipo <<*system*>>;
CARACTERÍSTICA := **NOME_CARACTERÍSTICA**, [**IGUAL_VALOR_CARACTERÍSTICA**];
IGUAL_VALOR_CARACTERÍSTICA := “ = ”, **VALOR_CARACTERÍSTICA**;
AMBIENTE_DURAÇÃO := [**AMBIENTE**], [**DURAÇÃO**];
AMBIENTE := [“ a ” | “ da ”], “exposição ao ”, **VALOR_AMBIENTE**;
DURAÇÃO := “ por ”, **VALOR_DURAÇÃO**;
NOME_CARACTERÍSTICA := Nome da Propriedade de Restrição da SysML em questão;

VALOR_CARACTERÍSTICA := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “VALOR” da Propriedade de Restrição em questão;
VALOR_AMBIENTE := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “AMBIENTE” da Propriedade de Restrição em questão;
VALOR_DURAÇÃO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “DURAÇÃO” da Propriedade de Restrição em questão;

A escolha do verbo principal do enunciado pode ser feita para tornar a frase mais correta de um ponto de vista semântico. Dois verbos foram colocados como opção (“exibir” e “apresentar”), entretanto, outros poderiam ser utilizados para uma melhor apresentação do enunciado do requisito em questão.

A escolha entre as opçoes de advérbio de tempo “durante” e “depois” do enunciado deve ser feita para tornar o correspondente requisito correto de um ponto de vista da característica ambiental sendo restringida. O mesmo vale para a preposição que acompanha este verbo.

Já a presença ou não dos símbolos **IGUAL_VALOR_CARACTERÍSTICA**, **AMBIENTE** e **DURAÇÃO** no enunciado está relacionada ao preenchimento dos respectivos Parâmetros de Restrição da Propriedade de Restrição em questão.

4.7.9. Gerar Restrições de Adequabilidade

Para cada Propriedade de Restrição da SysML criada para o Bloco com o estereótipo <<*system*>> e tipificada por um Bloco de Restrição contendo o estereótipo <<*suitability*>>, gera-se um requisito com o seguinte enunciado:

Enunciado := “O ”, **OBJETO**, “ deve ”, [“exibir ” | “apresentar ”], **CARACTERÍSTICA**, **DESEMPENHO_CONDIÇÃO_DURAÇÃO**;
OBJETO := Bloco com estereótipo <<*system*>>;

CARACTERÍSTICA := **NOME_CARACTERÍSTICA**,
[**IGUAL_VALOR_CARACTERÍSTICA**];
IGUAL_VALOR_CARACTERÍSTICA := “ = ”, **VALOR_CARACTERÍSTICA**;
DESEMPENHO_CONDIÇÃO_DURAÇÃO := [**DESEMPENHO**], [**CONDIÇÃO**], [**DURAÇÃO**];
DESEMPENHO := “ de acordo com ”, **VALOR_DESEMPENHO**;
CONDIÇÃO := “ enquanto na ”, **VALOR_CONDIÇÃO**;
DURAÇÃO := “ por ”, **VALOR_DURAÇÃO**;
NOME_CARACTERÍSTICA := Nome da Propriedade de Restrição da SysML em questão;
VALOR_CARACTERÍSTICA := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “VALOR” da Propriedade de Restrição em questão;
VALOR_DESEMPENHO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “DESEMPENHO” da Propriedade de Restrição em questão;
VALOR_CONDIÇÃO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “CONDIÇÃO” da Propriedade de Restrição em questão;
VALOR_DURAÇÃO := Valor padrão da Propriedade de Valor da SysML ligada ao Parâmetro de Restrição de nome “DURAÇÃO” da Propriedade de Restrição em questão;

A escolha do verbo principal do enunciado pode ser feita para tornar a frase mais correta de um ponto de vista semântico. Dois verbos foram colocados como opção (“exibir” e “apresentar”), entretanto, outros poderiam ser utilizados para uma melhor apresentação do enunciado do requisito em questão.

Já a presença ou não dos símbolos **IGUAL_VALOR_CARACTERÍSTICA**, **DESEMPENHO**, **CONDIÇÃO** e **DURAÇÃO** no enunciado está relacionada ao preenchimento dos respectivos Parâmetros de Restrição da Propriedade de Restrição em questão.

4.7.10. Gerar Requisitos de Projeto Detalhado

A metodologia para criação do modelo em SysML para o sistema de interesse proposta nesta Tese de Doutorado possibilita ao engenheiro de sistemas definir requisitos para os subsistemas constituintes do sistema sendo modelado.

Conforme abordado na Seção 4.4, o engenheiro de sistemas relaciona o modelo de requisitos ao modelo de arquitetura, alocando as funções identificadas aos módulos de arquitetura definidos. Com isso, pode-se utilizar as mesmas técnicas descritas na Seção 4.7.1 e na Seção 4.7.2 para gerar requisitos funcionais de subsistema, ou seja, requisitos onde o objeto que apresenta o comportamento em questão é o módulo de arquitetura para o qual a função foi alocada. Mais ainda, a observabilidade deste comportamento (i.e. função) será feita do ponto de vista das interfaces disponibilizadas pelo correspondente módulo de arquitetura.

Utilizando a mesma técnica descrita na Seção 4.7.6, pode-se gerar requisitos não-funcionais de interface para os subsistemas, ou seja, para os diferentes módulos de arquitetura identificados.

Caso necessário, o engenheiro de sistema pode, ainda, revisitar o modelo de arquitetura e criar restrições de projeto, restrições ambientais e restrições de adequabilidade para os diferentes módulos de arquitetura, da mesma maneira como elas foram criadas para o sistema. Feito isso e utilizando as mesmas técnicas descritas na Seção 4.7.7, na Seção 4.7.8 e na Seção 4.7.9, o engenheiro de sistema pode gerar os correspondentes requisitos não-funcionais para os subsistemas.

Com isso, além dos requisitos de sistema, o engenheiro de sistemas tem à sua disposição mecanismos para gerar requisitos para os subsistemas, estes representados pelos diferentes módulos de arquitetura identificados.

5 ESTUDO DE CASO E AVALIAÇÃO QUALITATIVA DOS REQUISITOS

Este capítulo tem como objetivo apresentar os resultados da aplicação, a um sistema espacial, da metodologia de modelagem descrita no Capítulo 4, Seção 4.6, e realizar a extração automática de seus requisitos de sistema de acordo com as técnicas propostas no Capítulo 4, Seção 4.7.

O sistema espacial escolhido foi o Nanossatélite AESP14 (AESP14, 2014) por ser um projeto que já passou por todo o ciclo de desenvolvimento e, portanto, já possui seus requisitos de sistema identificados.

A modelagem do sistema espacial escolhido, utilizando a metodologia proposta nesta Tese de Doutorado e desenvolvida aqui neste capítulo, não tem como objetivo ser exaustiva, mas sim prover uma maneira para comparar seus resultados com o que foi produzido no programa original, identificando, assim, as verdadeiras contribuições das técnicas utilizadas.

Este capítulo está organizado da seguinte maneira. Na Seção 5.1, apresenta-se uma breve introdução ao sistema espacial AESP14, com o objetivo de facilitar o entendimento do seu modelo descritivo, modelo este desenvolvido ao longo da Seção 5.2. Na Seção 5.3, mostram-se alguns dos requisitos de sistema do AESP14, que foram gerados a partir de seu modelo anteriormente elaborado. Finalmente, na Seção 5.4, tem-se uma comparação dos requisitos obtidos na Seção 5.3 com os requisitos presentes em AESP14 (2014).

5.1 Introdução ao Sistema Espacial AESP14

O sistema espacial AESP14 consiste de um nanossatélite do tipo CubeSat 1U desenvolvido por uma cooperação científica e tecnológica entre o Instituto Tecnológico de Aeronáutica (ITA) e o Instituto Nacional de Pesquisas Espaciais (INPE).

Inicialmente, sua missão era investigar o mecanismo de geração das bolhas de plasma equatorial, através da realização de medidas de suas ocorrências e características de distribuição em uma escala global e em função da hora local, estação do ano e condições do ambiente ionosférico. A carga útil do

AESP14 consistiria de uma sonda Langmuir para medições da densidade de elétrons e temperatura do plasma ionosférico.

Entretanto, ao longo do programa, e por motivos irrelevantes ao escopo desta Tese de Doutorado, foi decidido que a missão do AESP14 seria, na verdade, validar uma plataforma nacional de CubeSat. Esta validação aconteceria através de um experimento do Clube de Radioamadores de Americana (CRAM), na forma de uma competição entre radioamadores, onde o AESP14 transmitiria de forma aleatória 100 sequências de caracteres previamente armazenadas no satélite e codificadas pelo algoritmo MD5 (MD5, 1992). As dez primeiras estações de radioamador que conseguissem receber as 100 diferentes mensagens enviadas pelo AESP14 receberiam uma recompensa comemorativa.

Para efeitos de construção do modelo descritivo do sistema escolhido, levou-se em conta a missão original do AESP14, ou seja, a investigação das bolhas de plasma equatorial.

5.2 Modelo Descritivo do AESP14

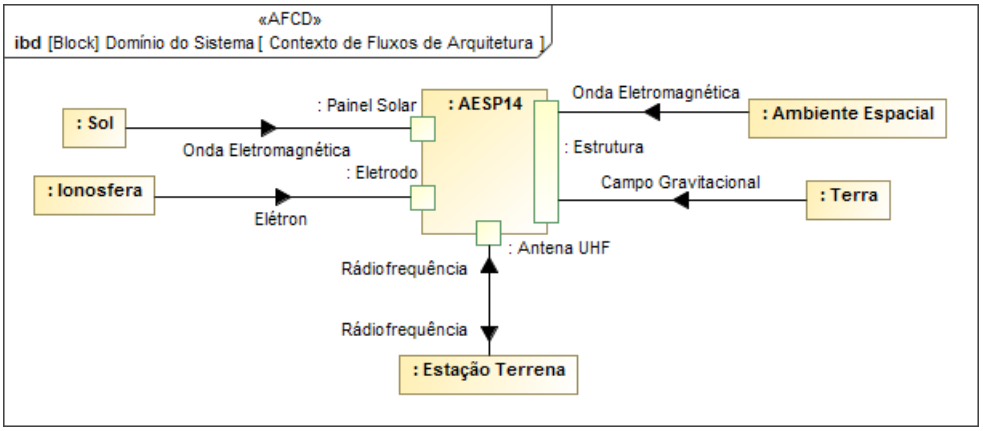
Nesta seção, apresenta-se o modelo descritivo do AESP14, construído utilizando a metodologia elaborada no Capítulo 4, Seção 4.6, e tendo como base o resultado da aplicação da abordagem de engenharia de sistemas, materializado em AESP14 (2014).

Não seguiu-se a sequência elaborada naquela seção, mas, conforme já discutido, a metodologia proposta nesta Tese de Doutorado não prescreve uma sequência a ser seguida. O engenheiro de sistemas tem liberdade para elaborar os correspondentes modelos de forma interativa e sem um sequenciamento definido. Entretanto, no Apêndice D, encontra-se o desenvolvimento completo do modelo do AESP14, seguindo a mesma sequência utilizada no Capítulo 4, Seção 4.6.

5.2.1 Modelo de Arquitetura

Em muitos casos, o engenheiro de sistemas já conhece previamente quais subsistemas existirão, pois está modelando um sistema similar a outros já realizados ou o sistema de interesse utiliza produtos já conhecidos. Nestes casos, começar pelo modelo de arquitetura é mais intuitivo e, mais ainda, este modelo representa um nível de abstração mais próximo da realidade do que aquele do modelo de requisitos. Na Figura 5.1 abaixo, tem-se o diagrama de contexto de fluxo de arquitetura do AESP14.

Figura 5.1 – Contexto de Fluxos de Arquitetura do AESP14.

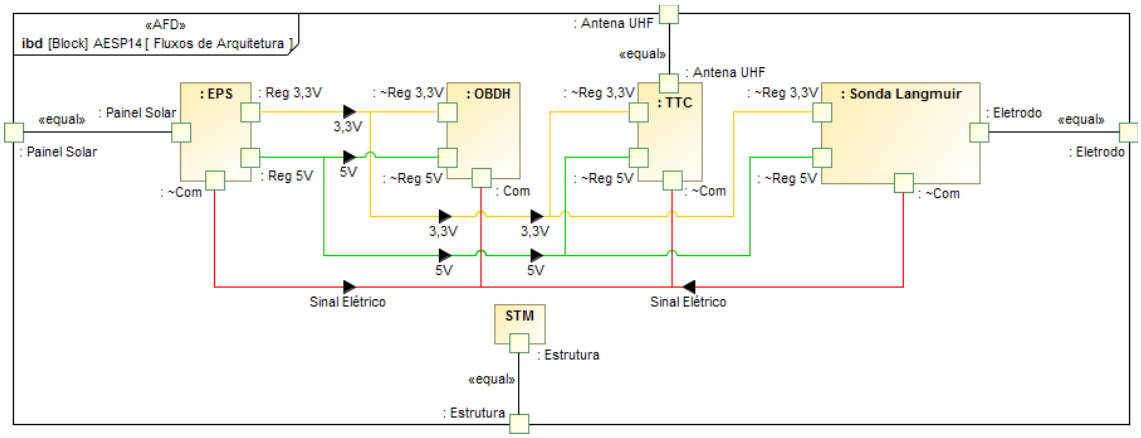


Fonte: Produção do Autor.

No diagrama da figura acima, tem-se o sistema AESP14 ao centro, com suas respectivas interfaces externas. A radiação cósmica imposta pelo ambiente espacial ao redor do satélite chega à sua estrutura através de ondas eletromagnéticas. Também através desta mesma interface, o campo gravitacional do planeta Terra exerce sua atração. As ondas eletromagnéticas emitidas pelo Sol chegam ao painel solar do satélite para que este consiga gerar a energia elétrica necessária. As diferenças de potenciais aplicadas ao eletrodo inserido no plasma ionosférico fazem com que elétrons se desloquem e gerem uma corrente elétrica que é capturada pelo sistema e medida para se determinar as características deste plasma. Sinais de radiofrequência são trocados com a estação terrena através de uma antena de UHF a fim de se transmitir telemetrias e receber telecomandos.

Na Figura 5.2 abaixo, mostra-se o diagrama de fluxos de arquitetura, fluxos estes trocados entre os módulos de arquitetura identificados para o sistema AESP14.

Figura 5.2 – Fluxos de Arquitetura do AESP14.



Fonte: Produção do Autor.

No diagrama da figura acima, vê-se a composição do sistema AESP14 nos seguintes módulos de arquitetura: EPS, OBDH, TTC, Sonda Langmuir e STM. O módulo de arquitetura EPS fornece a energia elétrica necessária dentro do satélite, gerada a partir da energia solar. Já o módulo de arquitetura OBDH é responsável pela capacidade de processamento computacional dentro de satélite. O módulo de arquitetura TTC é responsável pelas telecomunicações dentro do satélite, ou seja, receber telecomandos e enviar telemetrias. O módulo de arquitetura Sonda Langmuir representa a carga útil do satélite, responsável pelas medições das características do plasma ionosférico. Finalmente, o módulo de arquitetura STM representa a estrutura mecânica do satélite, que deve passivamente tanto dissipar o calor excedente quanto controlar a atitude do satélite.

Ainda pelo diagrama da Figura 5.2 acima, têm-se as interfaces que cada módulo de arquitetura expõe. Todos os módulos de arquitetura, com exceção do módulo STM, apresentam uma interface tipificada pela interconexão chamada de “Com”. Esta interconexão representa um barramento de comunicação interna ao satélite, por onde os subsistemas trocam informações. Estes mesmos módulos de arquitetura apresentam duas

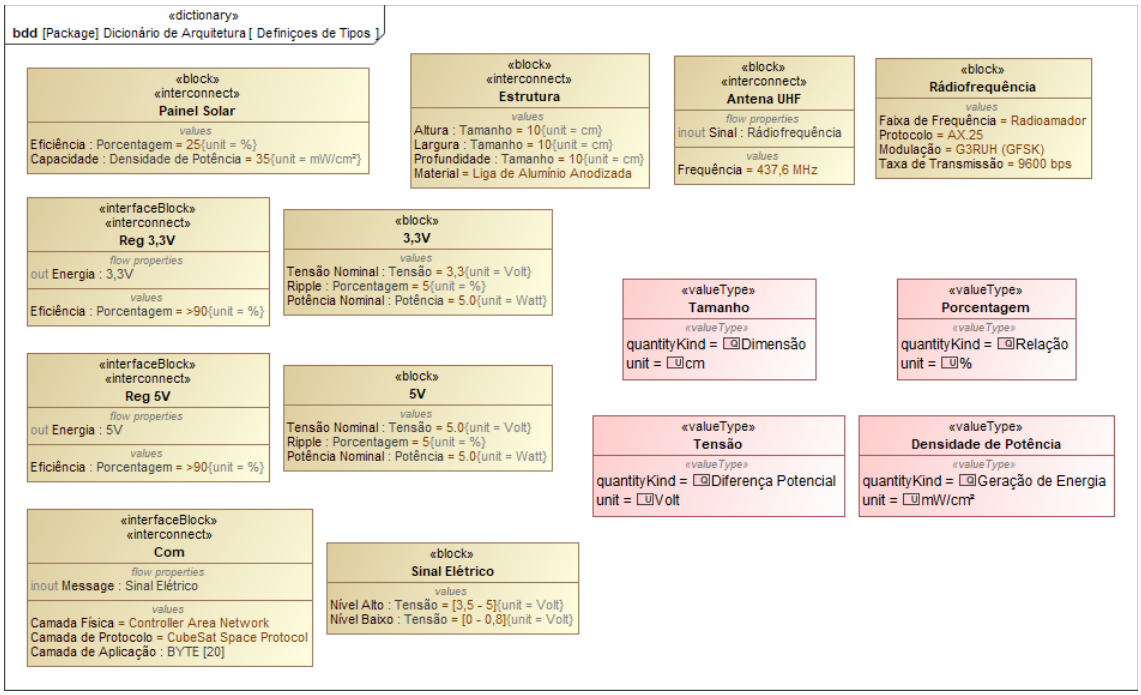
interfaces cada, tipificadas pelas interconexões chamadas de “Reg 3,3V” e “Reg 5V”. Estas interconexões representam os barramentos de tensão regulada de 3,3 Volts e 5 Volts, respectivamente, por onde o subsistema EPS fornece energia elétrica para os demais subsistemas. O módulo de arquitetura EPS apresenta, ainda, uma interface tipificada pela interconexão chamada de “Painel Solar”, por onde este receberá a radiação solar a ser transformada em energia elétrica. O módulo de arquitetura TTC apresenta, também, uma interface tipificada pela interconexão chamada de “Antena UHF”, por onde as ondas de radiofrequência de telemetrias e telecomandos serão trocadas. Já o módulo de arquitetura Sonda Langmuir expõe uma outra interface tipificada pela interconexão chamada de “Eletrodo”, que representa o dispositivo usado para efetuar as medições das características do plasma ionosférico. Finalmente, o módulo de arquitetura STM tem sua única interface tipificada pela interconexão chamada de “Estrutura”, que representa a estrutura mecânica do satélite.

Conforme especificação da linguagem SysML, as bordas do diagrama acima representam, na verdade, as fronteiras do sistema AESP14 com o ambiente externo. Por isso, têm-se nestas bordas as Portas da SysML representando as interfaces do satélite, já apresentadas na Figura 5.1, conectadas às correspondentes interfaces dos módulos de arquitetura que por elas são responsáveis.

No diagrama da Figura 5.2 acima, foram utilizadas cores para os conectores interligando os módulos de arquitetura simplesmente como um facilitador, pois como as linhas se cruzam, deixando-as da mesma cor poderia causar algum tipo de confusão para o leitor do diagrama. Em vermelho, tem-se o barramento de comunicação, por onde os módulos trocam pacotes de dados com as informações necessárias, através de um sinal elétrico específico ao protocolo de comunicação utilizado. Em amarelo, destaca-se o barramento de tensão regulada de 3,3 Volts e, em vermelho, nota-se o barramento de tensão regulada de 5 Volts.

Na Figura 5.3 abaixo, tem-se um diagrama especificando alguns dos tipos utilizados no modelo de arquitetura do sistema AESP14, representando, portanto, uma visão do dicionário de arquitetura.

Figura 5.3 – Dicionário de Arquitetura do AESP14.



Fonte: Produção do Autor.

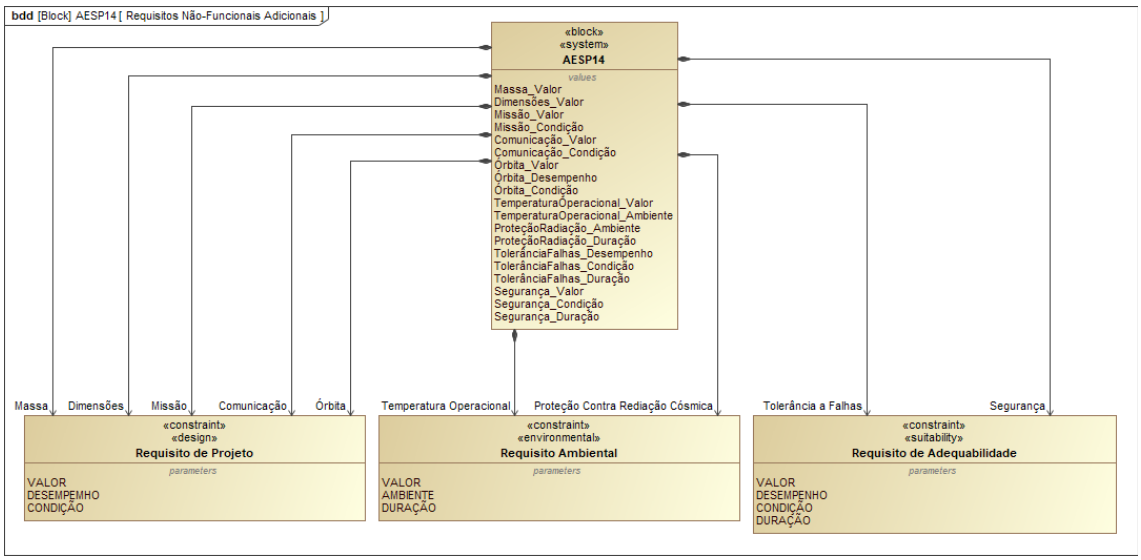
No diagrama da figura acima, veem-se alguns dos tipos utilizados no modelo de arquitetura para tipificar interconexões ou fluxos de arquitetura. A interconexão chamada “Painel Solar” possui valores para definir sua eficiência e sua capacidade de geração de energia. A interconexão chamada “Estrutura” possui valores para especificar suas dimensões e o tipo de material que deve ser utilizado na sua construção. Já a interconexão chamada “Antena UHF” possui um valor para definir qual será a frequência utilizada para trocar sinais de radiofrequência definidos pelo tipo chamado “Radiofrequência”, que contém valores especificando sua faixa de frequência, protocolo, modulação e taxa de transmissão de dados. A interconexão chamada “Reg 3,3V” possui um valor para definir sua eficiência e uma Propriedade de Fluxo da SysML para identificar qual tipo de fluxo pode ser trocado através desta interconexão e em qual sentido esta troca acontece. Neste caso, corresponde ao tipo chamado “3,3V”, que possui

valores definindo sua tensão nominal, *ripple* e potência nominal. Mais ainda, a classificação '*out*' ao lado de seu nome indica que fluxos deste tipo saem da correspondente Porta da SysML tipificada por esta interconexão, no módulo de arquitetura EPS, e chegam nas Portas igualmente tipificada por esta interconexão e conjugadas, nos demais módulos de arquitetura. De forma análoga, a interconexão chamada "Reg 5V" possui um valor para especificar sua eficiência e fluxos tipificados pelo Bloco chamado "5V", que possui valores definindo sua tensão nominal, *ripple* e potência nominal. Finalmente, a interconexão chamada "Com" possui valores definindo a camada física, a camada de protocolo e a camada de aplicação, utilizadas na comunicação entre os módulos de arquitetura. Mais ainda, esta interconexão possui um Fluxo de Propriedade para identificar que sinais elétricos podem entrar e sair por este ponto de interação, sinais estes tipificados pelo Bloco chamado "Sinal Elétrico", que possui valores para definir as tensões de níveis lógicos alto e baixo.

Conforme abordado no Capítulo 4, Seção 4.3.4, alguns requisitos não-funcionais do sistema seguirão o padrão definido por Carson (2015). Para tal, faz-se necessário incorporá-los ao modelo do sistema, conforme visto no Capítulo 4, Seção 4.6.4.4.

Na Figura 5.4 abaixo, tem-se esta incorporação no modelo SysML do sistema AESP14 de algumas restrições de projeto, restrições ambientais e restrições de adequabilidade. Veem-se, no diagrama, o Bloco do sistema e diversos relacionamentos de Composição com os Blocos de Restrição representando os diferentes tipos de requisitos não-funcionais propostos por Carson (2015). Cada relacionamento deste tipo representa uma Propriedade de Restrição no escopo do Bloco do sistema e os nomes destes elementos representam as características do sistema, cujos correspondentes requisitos não-funcionais restringirão. Estes nomes aparecem próximos às setas dos correspondentes relacionamentos de Composição.

Figura 5.4 – Requisitos Não-Funcionais Adicionais do AESP14.

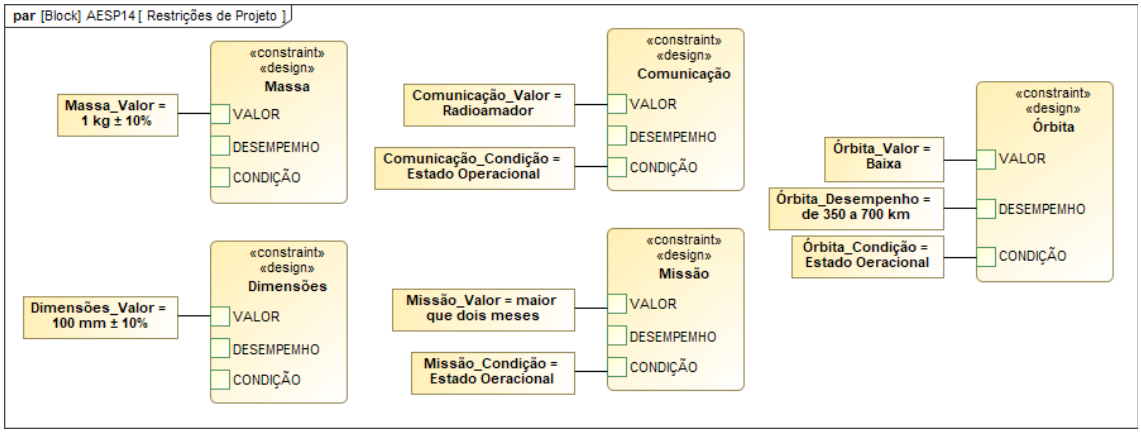


Fonte: Produção do Autor.

Ainda no diagrama da Figura 5.4 acima, dentro do compartimento intitulado "*values*" no Bloco do sistema, notam-se as Propriedades de Valor, cujos valores preencherão as lacunas dos correspondentes padrões de requisitos não-funcionais definidos por Carson (2015), e que aparecerão nos diferentes Diagramas Paramétricos criados para o Bloco do sistema.

Na Figura 5.5 abaixo, tem-se o Diagrama Paramétrico criado para o Bloco do sistema, onde os requisitos não-funcionais do tipo 'Requisitos de Projeto' são parametrizados.

Figura 5.5 – Restrições de Projeto do AESP14.

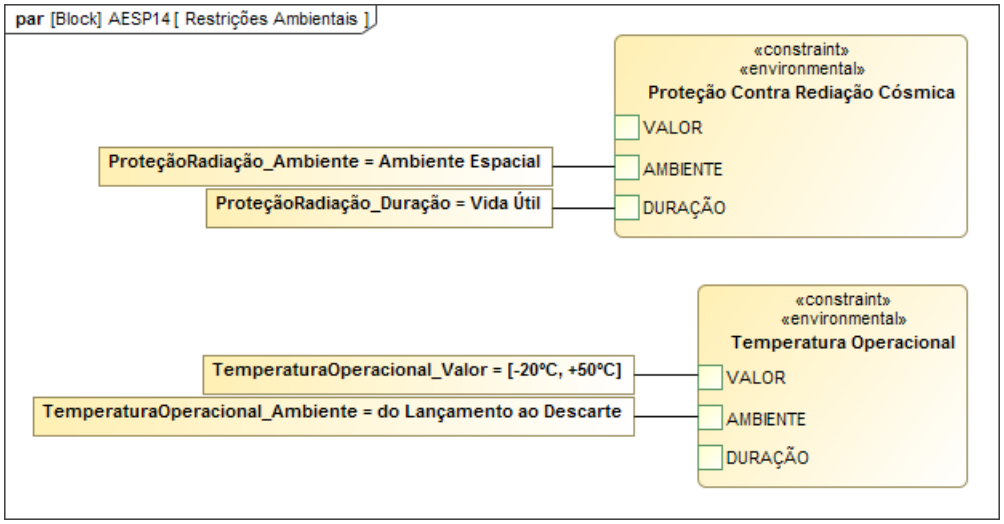


Fonte: Produção do Autor.

No diagrama da figura acima, veem-se as quatro Propriedades de Restrição do AESP14, representando as restrições de projeto criadas. Para cada uma destas restrições, têm-se as lacunas do padrão definido por Carson (2015) e as Propriedades de Valor usadas para seus devidos preenchimentos.

Na Figura 5.6 abaixo, tem-se o Diagrama Paramétrico criado para o Bloco do sistema, onde os requisitos não-funcionais do tipo ‘Requisito Ambiental’ são parametrizados.

Figura 5.6 – Restrições Ambientais do AESP14.

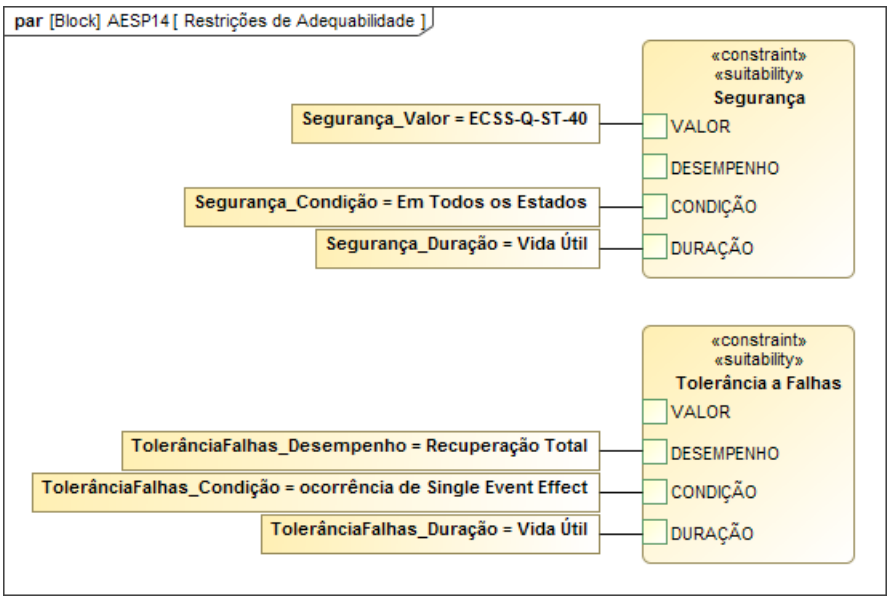


Fonte: Produção do Autor.

No diagrama da figura acima, veem-se as duas Propriedades de Restrição do AESP14, representando as restrições ambientais criadas. Para cada uma destas restrições, têm-se as lacunas do padrão definido por Carson (2015) e as Propriedades de Valor usadas para seus devidos preenchimentos.

Finalmente, na Figura 5.7 abaixo, tem-se o Diagrama Paramétrico criado para o Bloco do sistema, onde os requisitos não-funcionais do tipo ‘Requisito de Adequabilidade’ são parametrizados. Neste diagrama, têm-se as duas Propriedades de Restrição do AESP14, representando as restrições de adequabilidade criadas. Para cada uma destas restrições, têm-se as lacunas do padrão definido por Carson (2015) e as Propriedades de Valor usadas para seus devidos preenchimentos.

Figura 5.7 – Restrições de Adequabilidade do AESP14.

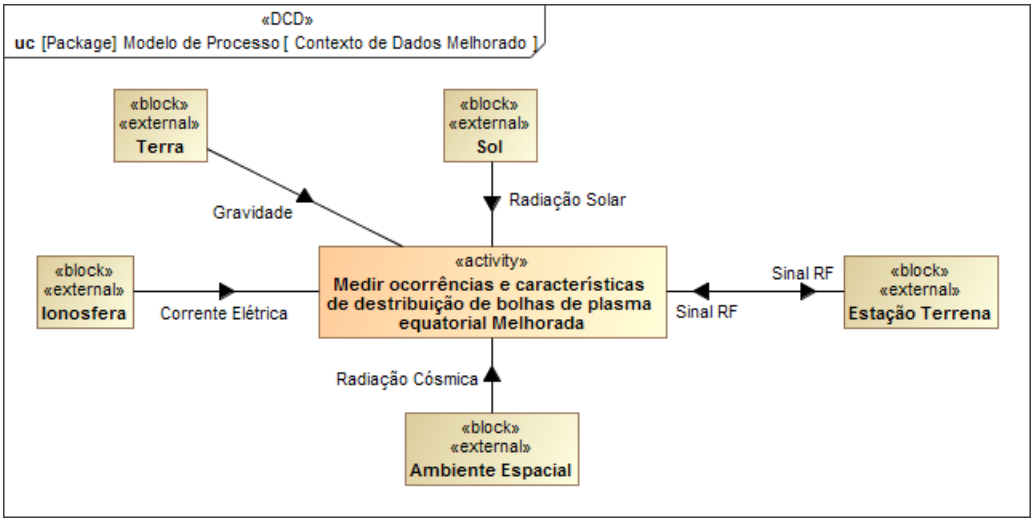


Fonte: Produção do Autor.

5.2.2 Modelo de Requisitos

Na Figura 5.8 abaixo, tem-se o diagrama de contexto funcional do sistema AESP14.

Figura 5.8 – Contexto Funcional do AESP14.



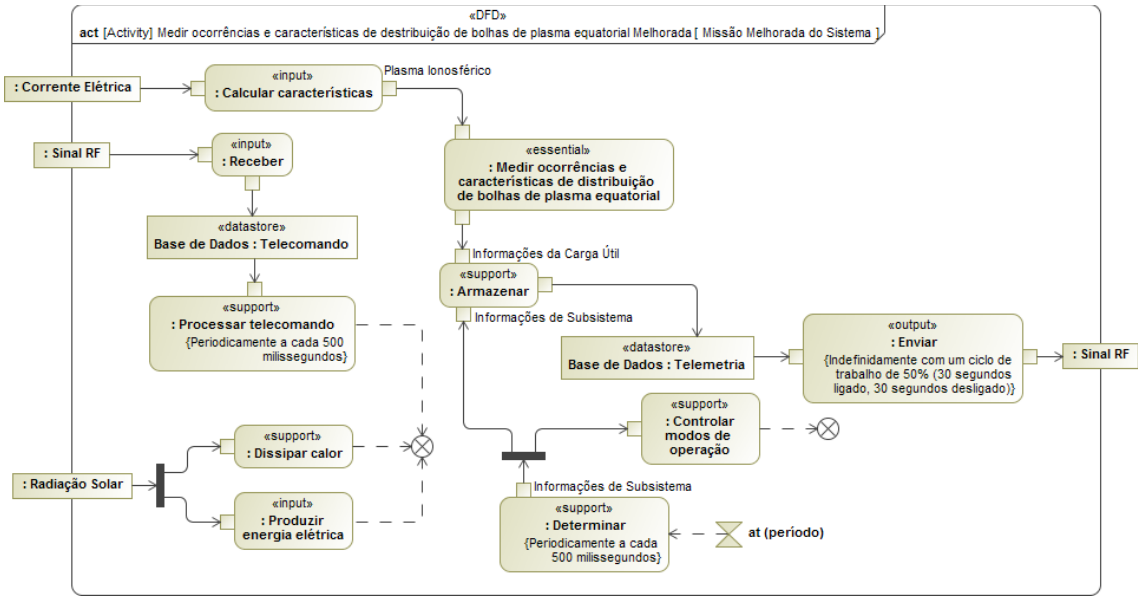
Fonte: Produção do Autor.

No diagrama da figura acima, observa-se que a ionosfera envia uma corrente elétrica, que é medida para se extrair as características do plasma. Sinais de radiofrequência são trocados com a estação terrena para enviar e receber

informações. A Terra, com seu campo gravitacional, influencia no decaimento da órbita do nanossatélite, podendo comprometer sua missão. O Sol fornece a energia necessária para o funcionamento do dispositivo através de sua radiação solar. O ambiente espacial, no qual o AESP14 está inserido, emite radiações cósmicas, que podem comprometer o correto funcionamento de seus circuitos eletrônicos.

Na Figura 5.9 abaixo, tem-se o diagrama de desdobramento funcional para a missão do sistema, apresentada na Figura 5.8 acima.

Figura 5.9 – Diagrama de Desdobramento Funcional do AESP14.



Fonte: Produção do Autor.

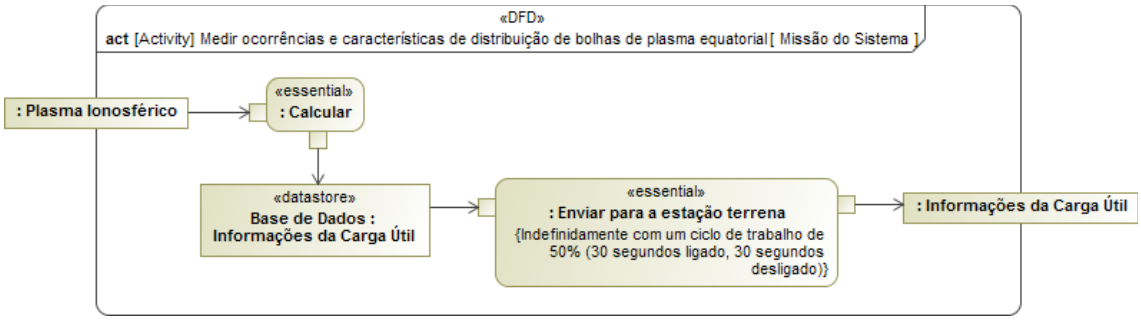
Do diagrama da figura acima, observa-se que existe uma função de entrada para calcular as características do plasma ionosférico, usando a corrente elétrica recebida, para fornecer a entrada esperada pela função essencial do sistema. Outra função de entrada recebe os telecomandos, usando o sinal RF que chega, para, então, serem processados por uma função de suporte, função esta relacionada a uma restrição de desempenho. Uma terceira função de entrada existe para produzir energia elétrica a partir da radiação solar recebida. Existem, também, funções de suporte para dissipar calor, determinar informações dos subsistemas de forma periódica e armazená-las junto às informações da carga útil, fornecidas pela correspondente função

essencial do sistema. Estas telemetrias são enviadas para a estação terrena, transmissão esta, feita por uma função de saída dedicada, que envia um sinal RF contendo as telemetrias anteriormente armazenadas, em ciclos de 30 segundos. Por último, existe uma função de suporte para controlar os modos de operação do AESP14.

Alguns dos fluxos identificados na Figura 5.8 não foram correspondidos por Parâmetros na Atividade representando a missão do sistema, vista na Figura 5.9 acima. Estes fluxos são os correspondentes da gravidade imposta pelo planeta Terra e da radiação cósmica vinda do ambiente espacial onde o satélite está inserido. Para o caso sendo desenvolvido, não existe controle ativo de órbita e atitude. Portanto, não foram necessárias funções para tratamento destes fluxos. Eles estão relacionados com requisitos não-funcionais a respeito de tempo de duração da missão, no caso da gravidade, e qualidade dos componentes eletrônicos utilizados, no caso da radiação cósmica. Entretanto, no caso, por exemplo, de um controle ativo de órbita e atitude, funções seriam criadas para tratar estes fluxos e gerar controles para garantir a missão do satélite.

Continuando a decomposição funcional, tem-se, na Figura 5.10 abaixo, o diagrama de fluxo de dados decompondo a função essencial do sistema, vista no diagrama da Figura 5.9 acima.

Figura 5.10 – Desdobramento da Função Essencial do AESP14.



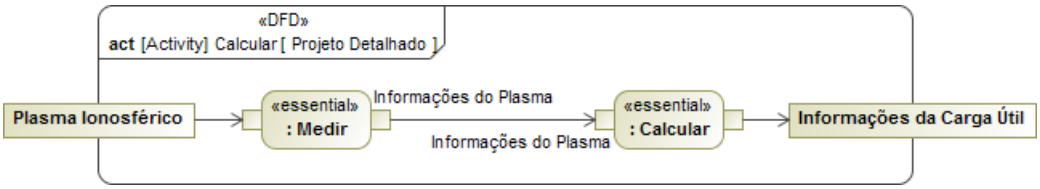
Fonte: Produção do Autor.

No diagrama da figura acima, a função essencial do sistema foi desdobrada em duas subfunções essenciais: Calcular as informações da carga útil usando o plasma ionosférico e enviar para a estação terrena as informações

de carga útil. Estas duas funções trocam os dados através de um armazenador chamado “Banco de Dados”, que armazena dados do tipo ‘Informações da Carga Útil’. Nota-se, ainda, que a função “Enviar para a estação terrena” está relacionada a um elemento Restrição da SysML. A expressão deste elemento aparece entre chaves logo abaixo seu nome.

Da maneira como foi feito o desdobramento funcional da função essencial do sistema AESP14, visto na Figura 5.10 acima, não se tem uma alocação unívoca das subfunções identificadas aos módulos de arquitetura vistos anteriormente na Figura 5.2. Portanto, faz-se necessário continuar a decomposição funcional até atingir o nível de abstração necessário. Na Figura 5.11 e na Figura 5.12 abaixo, têm-se as decomposições funcionais das funções identificadas anteriormente na Figura 5.10.

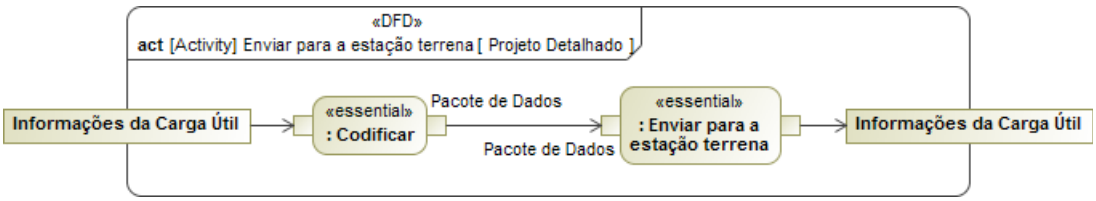
Figura 5.11 – Projeto Detalhado da Função Essencial “Calcular”.



Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função essencial “Calcular” apresentada no diagrama da Figura 5.10 foi decomposta em duas subfunções essenciais: uma para medir informações do plasma usando o próprio plasma ionosférico como entrada e outra para calcular as informações da carga útil usando as informações do plasma previamente medidas. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura Sonda Langmuir e a segunda, ao módulo de arquitetura OBDH.

Figura 5.12 – Projeto Detalhado da Função “Enviar para a estação terrena”.

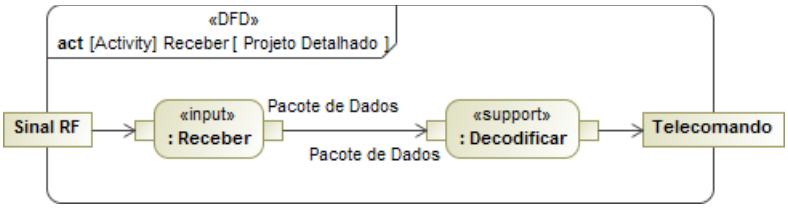


Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função essencial “Enviar para a estação terrena” apresentada no diagrama da Figura 5.10 foi decomposta em duas subfunções essenciais: uma para codificar um pacote de dados usando as informações da carga útil calculadas anteriormente e outra para enviar para a estação terrena as informações da carga útil usando o pacote de dados anteriormente codificado. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura OBDH e a segunda, ao módulo de arquitetura TTC.

Voltando ao diagrama da Figura 5.9, nota-se que as funções de entrada “Receber” e de saída “Enviar” não são passíveis de alocação unívoca a módulos de arquitetura do sistema AESP14. Novamente, portanto, faz-se necessário continuar o desdobramento funcional para estas funções. Na Figura 5.13 e na Figura 5.14 abaixo, têm-se as decomposições funcionais destas funções.

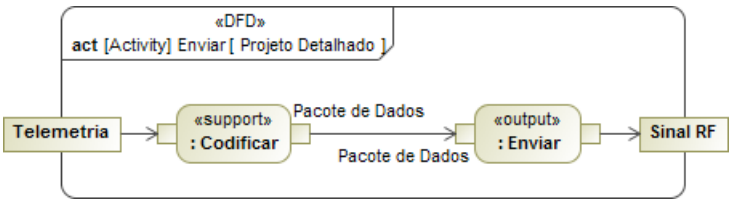
Figura 5.13 – Projeto Detalhado da Função de Entrada “Receber”.



Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função de saída “Enviar” apresentada no diagrama da Figura 5.9 foi decomposta em duas subfunções: uma subfunção de suporte para codificar um pacote de dados usando a telemetria armazenada e outra subfunção de saída para enviar o sinal RF à estação terrena usando o pacote de dados previamente codificado. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura OBDH e a segunda ao módulo de arquitetura TTC.

Figura 5.14 – Projeto Detalhado da Função de Saída “Enviar”.



Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função de entrada “Receber” apresentada no diagrama da Figura 5.9 foi decomposta em duas subfunções: uma subfunção de entrada para receber um pacote de dados usando o sinal RF contendo o telecomando e outra subfunção de suporte para decodificar o telecomando usando o pacote de dados previamente recebido. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura TTC e a segunda ao módulo de arquitetura OBDH.

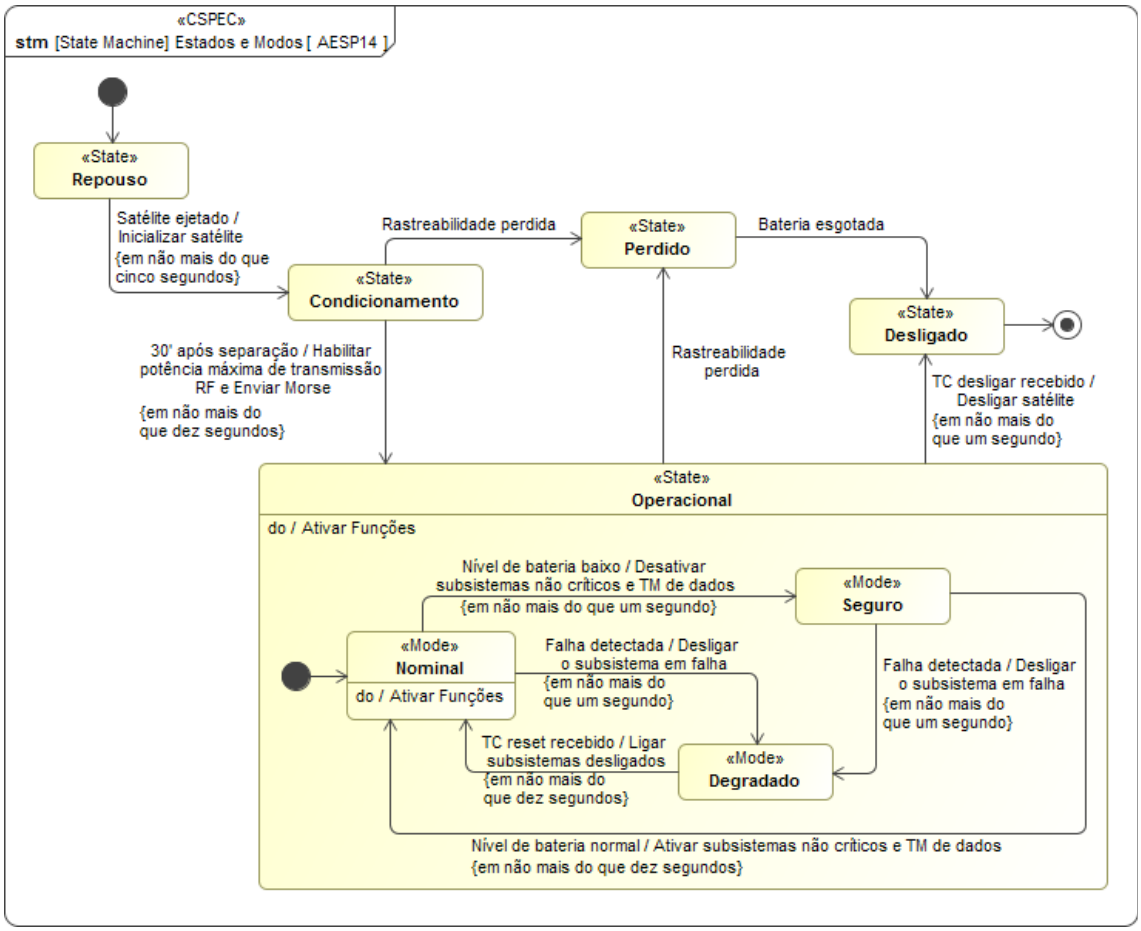
A análise de estados e modos do sistema AESP14, feita em AESP14 (2014), foi reproduzida aqui através da especificação de controle apresentada na Figura 5.15 abaixo.

Como resultado desta análise, foram identificados os seguintes estados: repouso, condicionamento, operacional, perdido e desligado. Enquanto no estado operacional, o sistema apresenta os seguintes modos operacionais: nominal, seguro e degradado. As transições no diagrama identificam as condições em que o sistema passa de um estado para outro ou de um modo operacional para outro. Algumas destas transições identificam, ainda, comportamentos do sistema que são executados na mudança de estado ou modo operacional, com suas respectivas restrições de desempenho.

O estado de repouso é o estado inicial do sistema. Quando o satélite é ejetado, ele é inicializado em não mais do que cinco segundos e passa ao estado de condicionamento. Trinta minutos após a separação, o satélite tem sua potência máxima de transmissão RF habilitada e passa a enviar código morse, em não mais do que dez segundos após este evento. Mais ainda, o satélite muda para seu estado operacional. Ainda no estado de condicionamento ou estando no estado operacional, caso haja perda de

rastreabilidade, o sistema muda para o estado perdido. Do estado perdido, o sistema passa ao estado desligado quando a energia de sua bateria se esgota. Outra possibilidade para chegar ao estado desligado, em não mais do um segundo, é receber um telecomando para desligar o satélite, enquanto este estiver no estado operacional. Este é, portanto, o estado final do satélite.

Figura 5.15 – Estados e Modos do AESP14.



Fonte: Produção do Autor.

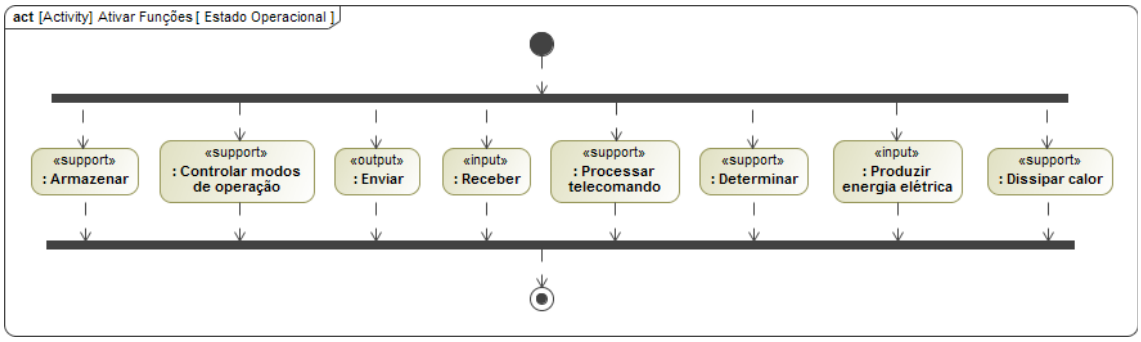
Estando no estado operacional, o modo nominal é o modo inicial do sistema. Caso detecte-se um nível baixo de tensão da bateria, desativam-se os subsistemas não críticos e a transmissão de telemetria, em não mais do que um segundo após o evento, levando o sistema ao modo seguro. Ainda no modo nominal, ao detectar-se uma falha em algum subsistema, desliga-se este subsistema em não mais do que um segundo após esta detecção e o satélite entra no modo degrado. No modo seguro, caso o nível de tensão da bateria volte ao normal, religam-se os subsistemas não críticos e a

transmissão de telemetria em não mais do que dez segundos após o evento, levando o sistema de volta ao modo nominal. Ainda no modo seguro, caso uma falha em algum subsistema seja detectada, desliga-se este subsistema em não mais do que um segundo após a falha e o satélite entra no modo degradado. Neste modo, ao receber um telecomando para reinicializar o satélite, ligam-se os subsistemas desligados em não mais do que dez segundos após o telecomando e o sistema volta ao modo nominal.

O modelo de controle tem como objetivo principal, conforme visto no Capítulo 4, prover um mecanismo de ativação de funções do sistema. Para isto, utilizam-se alguns compartimentos específicos do elemento Estado da SysML para identificar comportamentos que serão executados na entrada, saída ou enquanto o sistema permanecer naquele estado ou modo operacional. Na Figura 5.15 acima, nota-se o uso do compartimento '*do*' tanto no estado "Operacional", como no modo operacional "Nominal". Estes compartimentos referenciam duas Atividades igualmente nomeadas "Ativar funções". Estas Atividades têm por finalidade identificar, respectivamente, quais funções estão ativas no estado "Operacional", independente do modo operacional atual do sistema, e quais funções estão ativas somente quando o sistema estivar no modo operacional "Nominal".

Na Figura 5.16 abaixo, mostra-se um Diagrama de Atividade da SysML para a Atividade "Ativar funções" referenciada no compartimento '*do*' do estado "Operacional".

Figura 5.16 – Funções do AESP14 Ativas no Estado Operacional.

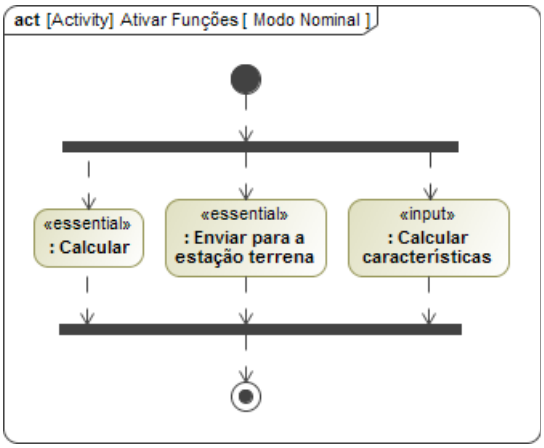


Fonte: Produção do Autor.

Pelo diagrama da figura acima, têm-se as funções ativas enquanto o sistema permanecer no estado “Operacional” são: função de suporte para armazenar as telemetrias, função de suporte para controlar os modos operacionais do satélite, função de saída para enviar telemetrias à estação terrena, função de entrada para receber telecomandos da estação terrena, função de suporte para processar os telecomandos recebidos, função de suporte para determinar as informações dos subsistemas do satélite, função de entrada para produzir energia elétrica e, finalmente, função de suporte para dissipar o calor acumulado no satélite.

Na Figura 5.17 abaixo, mostra-se um Diagrama de Atividade da SysML para a Atividade “Ativar funções” referenciada no compartimento ‘*do*’ do modo operacional “Nominal”.

Figura 5.17 – Funções do AESP14 Ativas no Modo Nominal.



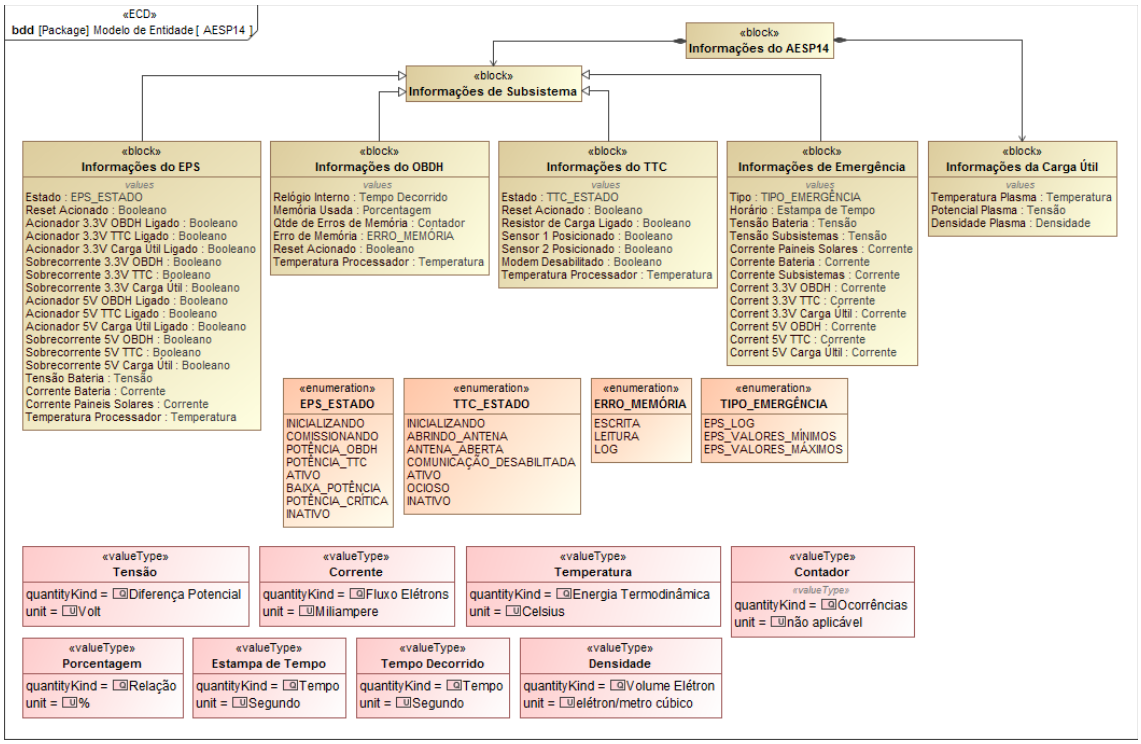
Fonte: Produção do Autor.

Do diagrama da figura acima, têm-se as funções do sistema AESP14 que estão ativas enquanto o satélite estiver no modo operacional “Nominal”. São elas: função essencial para calcular as informações da carga útil usando o plasma ionosférico, função essencial para enviar para a estação terrena as informações da carga útil e função de entrada para calcular as características do plasma ionosférico usando a corrente elétrica medida.

Na Figura 5.18 abaixo, tem-se um diagrama de classes de entidade para o AESP14, cujo propósito restringi-se a mostrar os dados (material, energia ou informação) manipulados pelo sistema, do ponto de vista de seu domínio de

negócio, onde o nível de abstração permite distanciar-se das possíveis tecnologias e implementações que serão ou poderão ser utilizadas no seu desenvolvimento.

Figura 5.18 – Modelo de Entidade para o AESP14.



Fonte: Produção do Autor.

Na parte mais acima do diagrama da figura acima, têm-se Blocos da SysML representando as informações que serão manipuladas pelo sistema AESP14. Criaram-se Blocos para representar as informações que cada subsistema fornecerá, informações de emergência e informações de carga útil. Estas informações estão baseadas em AESP-14 Telemetry (2015). Notam-se dois grandes grupos de informações: “Informações de Subsistema” e “Informações da Carga Útil”. O primeiro é, ainda, subdividido em quatro subtipos distintos: “Informações do EPS”, “Informações do OBDH”, “Informações do TTC” e “Informações de Emergência”. Os três primeiros subtipos referem-se a informações dos subsistemas do sistema AESP14 e o quarto subtipo, as informações de emergência do sistema AESP14. O segundo grupo de informações refere-se aos dados de carga útil.

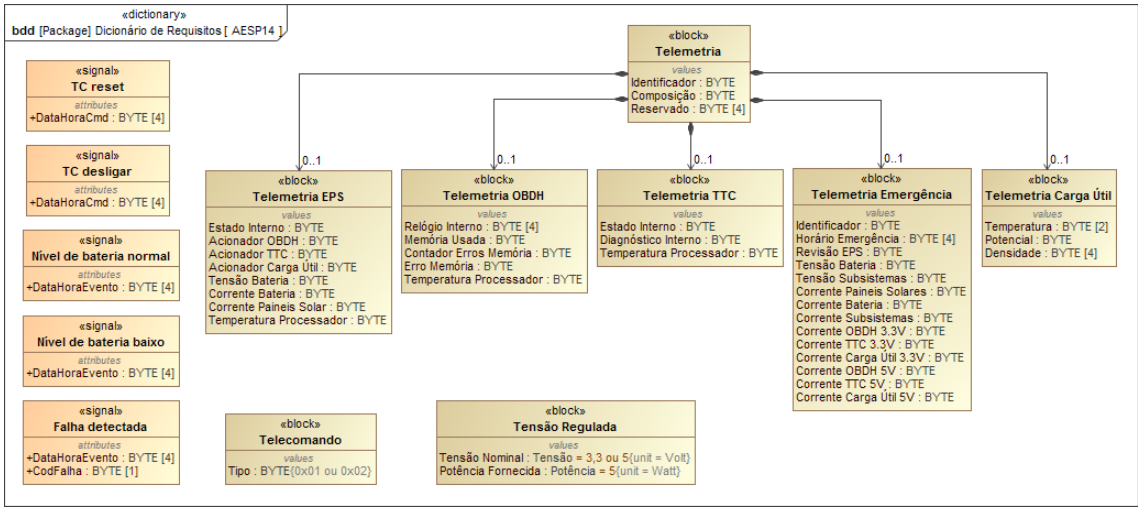
Propriedades de Valor da SysML, criadas para os Blocos acima descritos, compõem o conjunto de dados que cada grupo de informações carrega. Tais elementos devem ser devidamente tipificados com elementos Tipo de Valor da SysML para dar consistência ao modelo de entidade.

No Capítulo 4, Seção 4.6.2.1, mencionou-se que os tipos de dados utilizados no modelo de entidade estariam definidos no dicionário de requisito. Decidiu-se, aqui, mostrá-los no próprio diagrama de classes de entidade para facilitar a compreensão. Além do mais, os diagramas são simplesmente visualizações do modelo. Os elementos propriamente ditos podem estar organizados no modelo da forma que mais convém para o engenheiro de sistemas e podem ser mostrados nos diagramas que mais facilitarem sua comunicação para os leitores do modelo. Mais ao centro do diagrama da Figura 5.18 acima, com nomes em letras maiúsculas, estão Enumerações da SysML (extensões do elemento Tipo de Valor da SysML), representando tipos de dados, cujos valores estão delimitados por um conjunto finito de possibilidades e que não representam uma quantidade mensurável através de uma unidade de medida. Logo abaixo, encontram-se alguns elementos Tipo de Valor da SysML, representando tipos de dados mensuráveis do domínio de negócio do sistema AESP14.

Finalmente, no diagrama da Figura 5.19 abaixo, mostra-se um diagrama do dicionário de requisitos, contendo alguns dos elementos usados para tipificações no modelo de requisitos do sistema AESP14. No canto esquerdo do diagrama da figura abaixo, têm-se os sinais (ou condições de dados) utilizados nos Eventos das Transições da especificação de controle, apresentada anteriormente na Figura 5.15. Mais ao centro, na parte superior, tem-se a definição do tipo utilizado para representar as telemetrias gerenciadas pelo satélite, extraídas de AESP-14 Telemetry (2015), e presente no diagrama da Figura 5.9. Nota-se uma semelhança com os correspondentes tipos definidos no diagrama do modelo de entidade da Figura 5.18, mas aqui o nível de abstração já está mais próximo do implementacional. Na parte de baixo do diagrama, têm-se os tipos para

telecomandos e tensão regulada, utilizados para tipificar parâmetros das funções de processar telecomandos e produzir energia elétrica dentro do satélite.

Figura 5.19 – Dicionário de Requisitos do AESP14.



Fonte: Produção do Autor.

5.3 Requisitos do AESP14 Extraídos do seu Modelo Descritivo

Após a construção do modelo do sistema de interesse utilizando a SysML, conforme apresentado na Seção 5.2 para o AESP14, o engenheiro de sistemas deve relacionar o modelo de requisitos com o modelo de arquitetura, conforme descrito no Capítulo 4, Seção 4.6.5. Após esta etapa, torna-se possível a extração automática dos requisitos do sistema modelado.

Apresentam-se, agora, portanto, alguns exemplos de requisitos do AESP14 extraídos de seu modelo SysML, de acordo com os tipos de requisitos definidos pela metodologia proposta no Capítulo 4, Seção 4.5, Figura 4.6. No Apêndice E, apresenta-se uma tabela contendo todos os requisitos extraídos do modelo do AESP14 elaborado na Seção 5.2.

Nos enunciados dos requisitos abaixo, bem como naqueles apresentados no Apêndice E, Tabela E.1, os termos entre colchetes são aqueles que identificam um elemento do modelo SysML, como por exemplo, Blocos, Atividades, Parâmetros, etc, conforme definido no Capítulo 4, Seção 4.7.

Abaixo, têm-se dois exemplos de requisitos funcionais de transição entre estados e modos operacionais do AESP14:

REQ-AESP14. Se [30’ após separação] e o [AESP14] estiver no estado [Condicionamento], o [AESP14] deve mudar para o estado [Operacional].

REQ-AESP14. Se [Falha detectada] e o [AESP14] estiver no modo operacional [Nominal] do estado [Operacional], o [AESP14] deve mudar para o modo operacional [Degradado] do estado [Operacional].

Abaixo, têm-se dois exemplos de requisitos funcionais de transição de estado ou modo operacional do AESP14:

REQ-AESP14. Se [30’ após separação] e o [AESP14] estiver no estado [Condicionamento], o [AESP14] deve [Habilitar potência máxima de transmissão RF e Enviar Morse] [em não mais do que dez segundos].

REQ-AESP14. Se [Falha detectada] e o [AESP14] estiver no modo operacional [Nominal] do estado [Operacional], o [AESP14] deve [Desligar o subsistema em falha] [em não mais do que um segundo].

Abaixo, têm-se dois exemplos de requisitos funcionais de estado ou modo operacional do AESP14:

REQ-AESP14. Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Armazenar] [Telemetria] usando [Informações da Carga Útil] e [Informações de Subsistema] enquanto em qualquer modo operacional do estado [Operacional].

REQ-AESP14. Se [Satélite ejetado] e [30’ após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [AESP14] deve [Enviar para a estação terrena] [Informações da Carga Útil] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Informações da Carga Útil] enquanto no modo operacional [Nominal] do estado [Operacional].

Abaixo, têm-se dois exemplos de requisitos não-funcionais de estados e modos operacionais do AESP14:

REQ-AESP14. O [AESP14] deve apresentar o estado [Operacional].

REQ-AESP14. O [AESP14] deve apresentar o modo operacional [Nominal] no estado [Operacional].

Abaixo, têm-se dois exemplos de requisitos não-funcionais de estado e modo operacional iniciais do AESP14:

REQ-AESP14. O [AESP14] deve iniciar no estado [Repouso].

REQ-AESP14. O [AESP14] deve iniciar no modo operacional [Nominal] quando entrar no estado [Operacional].

Abaixo, têm-se dois exemplos de requisitos não-funcionais de interfaces do AESP14:

REQ-AESP14. O [AESP14] deve ser compatível com a interface [Antena UHF].

REQ-AESP14. O [AESP14] deve ser compatível com a interface [Painel Solar].

Abaixo, têm-se três exemplos de requisitos, um para uma restrição de projeto, outro para uma restrição ambiental e, finalmente, um para uma restrição de adequabilidade do AESP14:

REQ-AESP14. O [AESP14] deve exibir [Dimensões = 100 mm ± 10%].

REQ-AESP14. O [AESP14] deve exibir [Proteção Contra Radiação Cósmica] durante exposição ao [Ambiente Espacial] por [Vida Útil].

REQ-AESP14. O [AESP14] deve exibir [Tolerância a Falhas] com [Recuperação Total] enquanto na [ocorrência de Single Event Effect] por [Vida Útil].

Abaixo, têm-se quatro exemplos de requisitos de projeto detalhado do AESP14:

REQ-AESP14. Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Armazenar] [Telemetria] usando [Informações da Carga Útil] e [Informações

de Subsistema] enquanto em qualquer modo operacional do estado [Operacional].

REQ-AESP14. Se [Satélite ejetado] e [30' após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [TTC] deve [Enviar para a estação terrena] [Informações da Carga Útil] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Pacote de Dados] de acordo com [Com] enquanto no modo operacional [Nominal] do estado [Operacional].

REQ-AESP14. O [TTC] deve ser compatível com a interface [Antena UHF].

REQ-AESP14. O [EPS] deve ser compatível com a interface [Painel Solar].

5.4 Comparação de Requisitos do AESP14

Nesta seção, comparam-se alguns requisitos extraídos do modelo SysML do AESP14 com seus equivalentes requisitos apresentados em AESP14 (2014). Como base para a comparação, utilizou-se o trabalho de Halligan (1993) que, conforme visto no Capítulo 2, Seção 2.1.4, de acordo com a presença ou não de campos determinados pela análise sintática do enunciado de um requisito é possível calcular um indicador da sua qualidade, sendo este indicador um valor de zero a um, podendo ser transformado em porcentagem para melhorar sua inteligibilidade.

Halligan (1993) estipula que a estrutura sintática de um requisito pode conter até sete campos, a saber: ator, condição, ação, objeto da ação, restrições da ação, refinamento ou origem do objeto e, finalmente, refinamento ou destino da ação. O autor atribui, ainda, para cada campo uma pontuação entre zero e um, sendo que o valor zero indica que o campo é irrelevante e o valor um indica máxima relevância. O indicador de qualidade de um requisito definido pelo autor, e aqui chamado de IQR (Indicador de Qualidade de Requisito), é, portanto, calculado através da Equação 5.1 abaixo:

$$IQR(\%) = \frac{\sum Pontuação\ de\ Campos\ Presentes}{\sum Pontuação\ de\ Campos} \times 100 \qquad (5.1)$$

A relevância de um dado campo, ainda segundo o autor, dá-se por uma análise subjetiva de quem está realizando as correspondentes medidas, baseada no tipo de requisito sendo analisado.

Segundo Carson (2015), definir as interfaces de entrada e saída no seu padrão para escrita de requisitos funcionais promove observabilidade para a referida função e, consequentemente, faz com que o correspondente requisito seja verificável. Levando isto em consideração, no contexto da comparação realizada nesta seção, os campos ‘refinamento ou origem do objeto’ e ‘refinamento ou destino da ação’ de Halligan (1993) serão divididos em dois campos distintos cada um, ou seja, ter-se-ão os seguintes campos: ‘refinamento do objeto’, ‘origem do objeto’, ‘refinamento da ação’ e ‘destino da ação’. Isto foi feito, pois os campos ‘origem do objeto’ e ‘destino da ação’ representam a observabilidade destacada por Carson (2015), tornando o correspondente requisito verificável.

Na Tabela 5.1 abaixo, mostram-se as pontuações de cada campo, consideradas para realizar as comparações entre requisitos extraídos do modelo SysML do AESP14 e seus equivalentes requisitos extraídos de AESP14 (2014).

Tabela 5.1 – Pontuações dos campos sintáticos de um requisito.

Campos Sintáticos	Pontuação Considerada
Ator	1
Condição	1
Ação	1
Objeto da Ação	1
Restrições da Ação	1
Origem do Objeto	1
Refinamento do Objeto	0,5
Destino da Ação	1
Refinamento da Ação	0,5

Fonte: Produção do Autor.

Na tabela acima, nota-se que a maioria dos campos foi considerada como tendo a mesma relevância entre si e somente os campos ‘refinamento do objeto’ e ‘refinamento da ação’ foram considerados com metade da relevância dos demais por trataram, justamente, de informações de refinamento. Estas pontuações foram escolhidas desta maneira, pois o objetivo é simplesmente comparar dois requisitos equivalentes e não ter uma medida precisa da qualidade de um dado grupo de requisitos.

Baseando-se na Equação 5.1 acima e na Tabela 5.1 acima, tem-se a Equação 5.2 abaixo para cálculo do indicador de qualidade de requisito adotado nesta Tese de Doutorado:

$$IQR(\%) = \frac{\sum Pontuação\,de\,Campos\,Presentes}{8} \times 100 \qquad (5.2)$$

AESP14 (2014) traz o requisito identificado como ‘2.05.003’ com o seguinte enunciado:

“O Nanossatélite deve enviar informação relativa à saúde de seus subsistemas para o centro de operação durante cada conexão com a estação terrena.”

Do modelo SysML do AESP14 elaborado na Seção 5.3, tem-se o seguinte enunciado para o equivalente requisito:

“Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Enviar] [Sinal RF] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Telemetria] enquanto em qualquer modo operacional do estado [Operacional].”

Na Tabela 5.2 abaixo, tem-se a análise sintática destes dois requisitos, de acordo com os campos definidos para a comparação. Logo abaixo, calcula-se o IQR para ambos os requisitos.

Tabela 5.2 – Análise sintática dos requisitos.

Campos Sintáticos	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
Ator	O Nanossatélite	o [AESP14]
Condição	durante cada conexão com a estação terrena	1) Se [Satélite ejetado] e [30’ após separação] 2) enquanto em qualquer modo operacional do estado [Operacional]
Ação	enviar	[Enviar]
Objeto da Ação	informação	[Sinal RF]
Restrições da Ação	---	[Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)]
Origem do Objeto	---	---
Refinamento do Objeto	relativa à saúde de seus subsistemas	usando [Telemetria]
Destino da Ação	para o centro de operação	de acordo com [Antena UHF]
Refinamento da Ação	---	---

Fonte: Produção do Autor.

Da tabela acima, calcula-se o IQR para ambos os requisitos:

$$IQR\ AESP\,14(2014)=\frac{5{,}5}{8}\times 100=68{,}75\,\% \tag{5.3}$$

$$IQR\ AESP\,14\,SysML=\frac{6{,}5}{8}\times 100=81{,}25\,\% \tag{5.4}$$

Das Equações 5.3 e 5.4 acima, nota-se que o requisito extraído do modelo SysML do AES14 elaborado na Seção 5.3 possui um indicador de qualidade com valor maior do que o equivalente requisito extraído de AESP14 (2014).

No Apêndice F, mostram-se mais comparações feitas entre requisitos extraídos de AESP14 (2014) e seus equivalentes requisitos extraídos do modelo SysML do AESP14 elaborado na Seção 5.3. Entretanto, na Tabela 5.3 abaixo, mostram-se as comparações do IQR entre outros requisitos extraídos do AESP14 (2014) e seus correspondentes requisitos extraídos do modelo SysML do AESP14 elaborado neste capítulo.

Tabela 5.3 – Indicador de Qualidade dos Requisitos.

Identificador dos Requisitos	**IQR do AESP14 (2014)**	**IQR do Modelo SysML do AESP14**
2.04.004 / REQ_AESP14_081	$\frac{4}{8}\times 100 = 50\,\%$	$\frac{4}{8}\times 100 = 50\,\%$
2.04.005 / REQ_AESP14_082	$\frac{3{,}5}{8}\times 100 = 43{,}75\,\%$	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$
2.05.001 / REQ_AESP14_025	$\frac{3{,}5}{8}\times 100 = 43{,}75\,\%$	$\frac{5}{8}\times 100 = 62{,}50\,\%$
2.05.002 / REQ_AESP14_020	$\frac{3{,}5}{8}\times 100 = 43{,}75\,\%$	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$
2.05.004 / REQ_AESP14_026	$\frac{3{,}5}{8}\times 100 = 43{,}75\,\%$	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$
2.07.002 / REQ_AESP14_079	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$
2.07.003 / REQ_AESP14_078	$\frac{4}{8}\times 100 = 50\,\%$	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$
2.07.014 / REQ_AESP14_048	$\frac{3}{8}\times 100 = 37{,}50\,\%$	$\frac{3}{8}\times 100 = 37{,}50\,\%$
2.07.015 / REQ_AESP14_004	$\frac{5}{8}\times 100 = 62{,}50\,\%$	$\frac{5}{8}\times 100 = 62{,}50\,\%$
2.07.024 / REQ_AESP14_007	$\frac{5}{8}\times 100 = 62{,}50\,\%$	$\frac{5}{8}\times 100 = 62{,}50\,\%$
2.07.025 / REQ_AESP14_018	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$	$\frac{5{,}5}{8}\times 100 = 68{,}75\,\%$

Fonte: Produção do Autor.

6 DISCUSSÕES

Neste capítulo, tem-se como objetivo apresentar discussões e justificativas das escolhas e contribuições desta Tese de Doutorado, estruturadas da seguinte maneira. Na Seção 6.1, justifica-se a escolha da SysML como linguagem de modelagem e do método HHP de Hatley et al. (2000) como base para a metodologia proposta. Já na Seção 6.2, faz-se uma comparação entre os padrões para escrita de requisitos encontrados na literatura com os padrões propostos no Capítulo 4. Na Seção 6.3, apresentam-se as contribuições deste trabalho. Finalmente, na Seção 6.4, destacam-se os aprendizados decorrentes da aplicação da metodologia proposta no estudo de caso apresentado no Capítulo 5.

6.1 Linguagem de Modelagem e Metodologia Base

A SysML foi escolhida por ser um padrão industrial (ISO, 2017), padrão este que consiste de uma linguagem visual de modelagem mantida pela OMG e usada para suportar a disciplina de engenharia de sistemas. Esta linguagem estende a UML 2.0 (OMG, 2015) e provê um conjunto adicional de elementos de modelagem e diagramas para representar sistemas complexos, multidisciplinares e altamente integrados. É importante ressaltar que a SysML não prescreve uma metodologia, mas simplesmente provê uma linguagem visual de modelagem. Conforme visto no Capítulo 2, Seção 2.4, algumas metodologias para a abordagem de engenharia de sistemas baseada em modelos usam a SysML para produzir seus artefatos. Mais ainda, conforme visto no Capítulo 3, Seção 3.1, Figura 3.5, resultados da busca por “SysML” no Google Acadêmico mostram um crescente interesse por esta linguagem desde quando sua versão 1.0 foi oficializada pela OMG.

Ainda no Capítulo 2, Seção 2.4, apresentaram-se metodologias para a abordagem de engenharia de sistemas baseada em modelos que utilizam uma variedade de linguagens de modelagem para elaboração dos modelos do sistema de interesse.

Dentre aquelas apresentadas, a metodologia HHP de Hatley et al. (2000) foi a escolhida como base para a metodologia de modelagem proposta nesta Tese de Doutorado, no Capítulo 4, assim como feito por Loureiro (1999). Diferentemente das outras metodologias abordadas, nesta metodologia os autores mostram um esforço na direção de prescrever atividades, cujo objetivo é entregar para equipes de desenvolvimento um conjunto de especificações completas, concisas e livres de ambiguidades. Estas especificações servirão, então, para o desenvolvimento das partes constituintes do sistema de interesse.

Com relação à metodologia HHP, ela prescreve o desdobramento (ou decomposição) dos comportamentos do sistema até que estes possam ser expressos através de textos estruturados, chamados de PSPECs (*Process Specifications*). Os autores propõem o uso de um subconjunto da linguagem natural e uma certa estruturação na elaboração destes textos, que serão os requisitos funcionais do sistema modelado. Com isso, nota-se um esforço dos autores na direção de se obter, ao final das atividades, um modelo consistente do sistema de interesse e requisitos funcionais de alta qualidade.

Esta metodologia HHP cobre, ainda, a arquitetura do sistema, especificando as restrições físicas (i.e. requisitos não-funcionais) que o sistema deve satisfazer, bem como a etapa de projeto detalhado, onde os requisitos funcionais devem ser alocados aos elementos da arquitetura de forma unívoca. Ou seja, caso um requisito funcional no nível do sistema não esteja suficientemente decomposto para ser alocado a um único elemento arquitetural, novas etapas de desdobramento funcional são feitas, até que se chegue em um nível de abstração passível de se ter a alocação de uma dada função (ou requisito funcional) para um único elemento da arquitetura do sistema. Com isso, nota-se um esforço dos autores na direção de eliminar ambiguidades na alocação funcional.

Tanto para as funções quanto para os módulos de arquitetura, Hatley et al. (2000) utilizam uma classificação de acordo com seu propósito. Funções e módulos de arquitetura podem ser classificados, conforme visto no Capítulo

4, como essencial, entrada, saída, interface com usuário e suporte. Nota-se, portanto, um esforço dos autores em se modelar um sistema resiliente, onde alterações em suas interfaces poderiam ser absorvidas sem impactar funções ou módulos de arquitetura essenciais.

Uma outra característica da metodologia HHP é que o modelo criado ao se utilizá-la pode ser considerado um modelo *S*Model*, por ser condizente com o metamodelo *S*Metamodel* definido por Schindel (2011) e apresentado no Capítulo 2, Seção 2.2.3, Figura 2.3, excluindo-se a parte referente aos *stakeholders* e seus requisitos.

O conceito de 'Interação Funcional' no metamodelo *S*Metamodel* definido por Schindel (2011) é representado pelos diversos diagramas que mostram troca de fluxos de dados, de controle ou de arquitetura, enquanto as funções identificadas representam o conceito de 'Papel Funcional'. Já o conceito de 'Estado' está materializado pelas especificações de controle, enquanto o modelo de entidade, o dicionário de requisitos e o dicionário de arquitetura representam o conceito de 'Entrada/Saída'. Os conceitos de 'Sistema', 'Interface' e 'Componente de Projeto' estão presentes no modelo de arquitetura, onde o próprio sistema e suas interfaces, bem como seus módulos de arquitetura e suas respectivas interfaces, são definidos. Finalmente, o conceito de 'Sistema de Acesso' se encontra definido por alguns dos terminadores, representando as entidades externas ao sistema modelado.

Conforme abordado por Schindel (2011), ao elaborar um modelo condizente com seu metamodelo *S*Metamodel*, tem-se uma representação mínima do sistema, o que facilita o gerenciamento de sua complexidade.

Entretanto, Hatley et al. (2000) valem-se constantemente de especificações em linguagem natural para suportar os diferentes modelos elaborados pela metodologia HHP. Isto insere, inevitavelmente, ambiguidades e problemas de interpretação.

Por possibilitar uma tipificação forte dos dados utilizados no modelo, a releitura da metodologia HHP com o uso da SysML proposta no Capítulo 4,

Seção 4.6, adiciona um nível de formalismo antes não existente. O uso de elementos da SysML para tipificar fluxos de dados, fluxos de controle, fluxos de arquitetura, interfaces e outras propriedades do sistema, diminuem as possíveis ambiguidades, pois quando um elemento do modelo aparece referenciado em algum texto, ele sempre representará o mesmo conceito dentro do escopo do sistema modelado.

6.2 Padrões para Enunciados de Requisitos

Iniciando-se por padrões de enunciados de requisitos funcionais, esta Tese de Doutorado abordou dois trabalhos: o de Carson (2015) e o de Lempia et al. (2016).

Carson (2015) propôs como padrão para enunciado de requisitos funcionais o seguinte: “O **AGENTE** deve **FUNÇÃO** de acordo com **INTERFACE-SAÍDA** com **DESEMPENHO** e **MOMENTO** dado **EVENTO-GATILHO** de acordo com **INTERFACE-ENTRADA** enquanto na **CONDIÇÃO**.”, onde as lacunas a serem preenchidas a cada instanciação do padrão para formar um novo requisito funcional estão em letras maiúsculas e em negrito. Aqui foram desconsideradas as possibilidades de omissão de partes opcionais.

Já Lempia et al. (2016) propôs como padrão para enunciado de requisitos funcionais o seguinte: “Se **GATILHO**, o **OBJETO** deve **COMPORTAMENTO DADO_SAÍDA** com o **ATRIBUTO_DADO_SAÍDA RESTRIÇÃO_DESEMPENHO** usando **DADO_ENTRADA** com o **ATRIBUTO_DADO_ENTRADA** enquanto no estado **ESTADO**.”, onde as lacunas a serem preenchidas a cada instanciação do padrão para formar um novo requisito funcional estão em letras maiúsculas e em negrito. Aqui foram desconsideradas as possibilidades de omissão de partes opcionais.

Observando estes padrões e utilizando a análise sintática de Halligan (1993) para compará-los, tem-se a Tabela 6.1 abaixo.

Nesta tabela, nota-se que o padrão definido por Lempia et al. (2016) não possui as definições para a origem do objeto e para o destino da ação, definições estas presentes no padrão definido por Carson (2015), que as

justifica afirmando que estes campos tornam o respectivo requisito funcional verificável, pois explicitam os pontos de observação do evento gatilho da função (i.e. entrada) e da função propriamente dita (i.e. saída). Por outro lado, o padrão definido por Carson (2015) não possui as definições para o objeto da ação e para o refinamento deste objeto (i.e. saída da função). Ausência, esta, que compromete a completude do requisito associado, uma vez que requisitos funcionais devem especificar os relacionamentos entre as entradas e as saídas do sistema (SCHINDEL, 2005).

Tabela 6.1 – Comparação entre padrões para requisito funcional.

Campos Sintáticos	Carson (2015)	Lempia et al. (2016)
Ator	AGENTE	OBJETO
Condições da Ação	CONDIÇÃO, EVENTO-GATILHO	GATILHO, ESTADO
Ação	FUNÇÃO	COMPORTAMENTO
Restrições da Ação	DESEMPENHO	RESTRIÇÃO_DESEMPENHO
Objeto da Ação	---	DADO_SAÍDA
Refinamento do Objeto	---	ATRIBUTO_DADO_SAÍDA
Origem do Objeto	INTERFACE-ENTRADA	---
Refinamento da Ação	MOMENTO	DADO_ENTRADA, ATRIBUTO_DADO_ENTRADA
Destino da Ação	INTERFACE-SAÍDA	---

Fonte: Produção do Autor.

Ainda analisando o padrão para enunciados de requisitos definido por Lempia et al. (2016), nota-se que é considerado somente o estado em que o sistema se encontra quando o comportamento descrito no respectivo requisito é observado. Pelo método de engenharia de sistemas proposto por Loureiro (1999), durante a análise funcional do sistema, devem-se encontrar os estados do sistema e os modos de operação em cada estado identificado para o sistema. Segundo o autor, um estado define um conjunto de circunstâncias caracterizando o sistema em um dado momento. Já os modos de operação agrupam funcionalidades do sistema em um dado estado. Portanto, identificar somente o estado em que o comportamento descrito em um requisito funcional é observado não é suficiente. Deve-se, ainda,

identificar em qual modo de operação deste estado o comportamento está presente.

Esta comparação mostra que ainda há espaço para aprimoramentos nestes padrões, com o objetivo de melhorar a qualidade dos respectivos requisitos funcionais.

O padrão para enunciados de requisitos funcionais proposto nesta Tese de Doutorado que corresponde aos respectivos padrões de Carson (2015) e Lempia et al. (2106) vistos acima é aquele proposto no Capítulo 4, Seção 4.7.3, e pode ser estruturado da seguinte forma: “Se **GATILHO**, o **OBJETO** deve **COMPORTAMENTO DADO_SAÍDA** de acordo com **INTERFACE_SAÍDA**, **DESEMPENHO**, usando **DADO_ENTRADA** de acordo com **INTERFACE_ENTRADA** enquanto no modo operacional **MODO_OPERACIONAL** do estado **ESTADO**”, onde as lacunas a serem preenchidas a cada instanciação do padrão para formar um novo requisito funcional estão em letras maiúsculas e em negrito. Aqui foram desconsideradas as possibilidades de omissão de partes opcionais.

Na Tabela 6.2 abaixo, tem-se, agora, a análise sintática de Halligan (1993) para o enunciado apresentado acima.

Tabela 6.2 – Análise sintática do padrão proposto para requisito funcional.

Campos Sintáticos	Padrão Proposto
Ator	OBJETO
Condições da Ação	GATILHO, MODO_OPERACIONAL do ESTADO
Ação	COMPORTAMENTO
Restrições da Ação	RESTRIÇÃO_DESEMPENHO
Objeto da Ação	DADO_SAÍDA
Refinamento do Objeto	Implícito em DADO_SAÍDA
Origem do Objeto	INTERFACE_ENTRADA
Refinamento da Ação	DADO_ENTRADA
Destino da Ação	INTERFACE_SAÍDA

Fonte: Produção do Autor.

Pela tabela acima, nota-se que o padrão para requisitos funcionais proposto no Capítulo 4, Seção 4.7.3, acaba com as deficiências anteriormente apontadas para os correspondentes padrões propostos por Carson (2015) e por Lempia et al. (2016). Primeiramente, o conceito de modo operacional foi introduzido no enunciado para melhorar a precisão da condição na qual a respectiva ação do requisito funcional ocorre. Em segundo lugar, garante-se a observabilidade da ação através da identificação das interfaces de saída e de entrada do correspondente requisito funcional. Finalmente, possibilita-se o preenchimento de todos os campos sintáticos definidos por Halligan (1993), o que garantiria um Indicador de Qualidade de Requisito (IQR) igual a 100%.

O refinamento do objeto da ação do respectivo requisito funcional está implícito no próprio objeto, pois ao se utilizar a SysML como linguagem de modelagem do sistema de interesse, o elemento representando este objeto poderá conter atributos e, quando necessário, seus respectivos valores. Portanto, estas informações estarão implícitas no campo **DADO_SAÍDA**, que, na verdade, referencia um elemento do modelo. O mesmo raciocínio aplica-se para o campo **DADO_ENTRADA**.

Ainda analisando padrões para escrita de requisitos funcionais, os trabalhos de Carson (2015) e Lempia at al. (2016) não levaram em conta requisitos funcionais descrevendo possíveis transições entre estados e modos operacionais do sistema, tão pouco requisitos funcionais descrevendo possíveis comportamentos ativados nestas transições. No Capítulo 4, Seções 4.7.1 e 4.7.2, foram propostos enunciados para estes dois tipos de requisitos funcionais.

No caso dos requisitos não-funcionais, esta Tese de Doutorado permite ao engenheiro de sistemas gerar enunciados seguindo os três correspondentes padrões definidos por Carson (2015) e, mais ainda, possibilita a geração de outros tipos de requisitos não-funcionais, não abordados pelos trabalhos referenciados nesta Tese de Doutorado. Estes tipos possuem seus enunciados propostos no Capítulo 4, Seções 4.7.4, 4.7.5 e 4.7.6.

6.3 Contribuições em Relação à Literatura no Tema

Conforme visto no Capítulo 1, Seção 1.3, o estudo estatístico realizado, que pesquisou a ocorrência de alguns termos específicos a esta Tese de Doutorado em trabalhos publicados nos periódicos e simpósios do INCOSE e em resultados de buscas no Google Acadêmico, mostrou que temas relacionados a requisitos, modelagem e SysML vêm sendo constantemente e crescentemente abordados. Mais ainda, a escrita de bons requisitos continua sendo foco de vários trabalhos, o que indica a ausência de uma solução sistemática para o problema.

De posse deste estudo, conclui-se que abordagens e metodologias para melhorar a qualidade de requisitos ainda possuem espaço para discussões e aprofundamentos, sendo justamente nesta linha de pesquisa que se encaixam as contribuições desta Tese de Doutorado.

Conforme visto no Capítulo 3, Figura 3.2, Nicolás e Toval (2009) concluíram que trabalhos relacionados a modelos orientados a objetivos, casos de uso e cenários representam a maioria dos esforços pesquisados pelos autores no sentido da geração automática de requisitos a partir de um modelo descritivo do sistema de interesse. Mais ainda, os autores destacam somente um trabalho utilizando a UML e que deixa de lado casos de uso e cenários. Segundo eles, um maior esforço deveria ser direcionado para esta linha de pesquisa, mais precisamente, para o uso de técnicas da UML mais voltadas para a engenharia de requisitos.

Os trabalhos apresentados no Capítulo 3 posteriores a 2009, com exceção de Lempia et al. (2016), representam esforços de transformação de modelos descritos em textos. Corroborando com a conclusão de Nicolás e Toval (2009), estes trabalhos utilizam casos de uso para derivação de requisitos ou casos de teste.

O trabalho desenvolvido nesta Tese de Doutorado preenche a lacuna apontada por Nicolás e Toval (2009), onde técnicas da SysML mais voltadas para a engenharia de requisitos foram utilizadas para elaboração do modelo

descritivo do sistema de interesse. A partir deste modelo são, então, gerados os requisitos de sistema.

Já em comparação com o trabalho de Lempia et al. (2016) e Carson (2015), existe a contribuição em relação à qualidade do enunciado para requisitos funcionais. Conforme visto na Seção 6.2 acima, utilizando os campos sintáticos propostos por Halligan (1993), viu-se que os enunciados para requisitos funcionais propostos em ambos os trabalhos possuem deficiências em termos de sua completude, que foram sanadas no correspondente enunciado para requisitos funcionais proposto nesta Tese de Doutorado.

Recapitulando a metodologia proposta no Capítulo 4, mais especificamente, a atividade de geração automática de requisitos apresentada na Figura 4.6, nota-se que o trabalho desenvolvido nesta Tese de Doutorado propõe a geração de nove tipos distintos de requisitos, abrangendo tanto requisitos funcionais quanto requisitos não-funcionais. Já Lempia et al. (2016) propõem somente um tipo de requisito, sendo este para representar requisitos funcionais. Carson (2015) propõe quatro tipos distintos de requisitos, sendo um para representar requisitos funcionais e três para representar requisitos não-funcionais. Nota-se, nestes dois últimos trabalhos, a ausência da representação de alguns tipos de requisitos, como por exemplo, requisitos para compatibilidades com interfaces, requisitos para apresentar estados e modos operacionais, bem como requisitos para identificar estados e modos operacionais iniciais.

Do ponto de vista da elaboração do modelo descritivo do sistema de interesse, esta Tese de Doutorado propôs uma metodologia detalhada e prescritiva, ou seja, esta metodologia mostra como e quando usar os diagramas, elementos e relacionamentos da SysML para elaboração dos modelos de requisitos e arquitetura do sistema. Mais ainda, é feita uma associação deste modelo com os conceitos do metamodelo *S*Metamodel* de Schindel (2011) para demonstrar que o modelo elaborado pode ser considerado do tipo *S*Model* (SCHINDEL, 2011), conforme discussão sobre gerenciamento da complexidade feita no Capítulo 2, Seção 2.2.3. Já Lempia

et al. (2016) não mostram como elaborar o modelo e já partem do pressuposto da existência de um modelo do tipo *S*Model* para gerar os requisitos seguindo seu único enunciado proposto.

Já de um ponto de vista da metodologia de elaboração do modelo descritivo do sistema de interesse propriamente dita, esta Tese de Doutorado baseou-se no método HHP de Hatley et al. (2000). Como primeira contribuição, destaca-se a atualização da linguagem utilizada para representar o modelo. Enquanto no método original utilizou-se DFDs (e suas variações), bem como algumas representações proprietárias, esta Tese de Doutorado propôs o uso restrito da SysML, uma linguagem mais moderna e um padrão industrial. Já como uma segunda contribuição, tem-se a redução do número de CSPECs elaboradas. Enquanto no método original tem-se uma CSPEC para cada nível de abstração funcional, nesta Tese de Doutorado propõe-se uma única especificação de controle para o sistema, onde mostram-se os estados e modos operacionais do sistema, bem como a ativação ou inibição das suas funções terminais. Com esta redução, evita-se incorporar no modelo definições desnecessárias e que não serão utilizadas futuramente.

6.4 Aprendizados com o Uso da Metodologia Proposta

Ao aplicar a metodologia proposta no Capítulo 4 ao sistema AESP14 tomado como estudo de caso e apresentado de forma compacta no Capítulo 5 e de forma extensa no Apêndice D, notou-se que a especificação de controle elaborada como parte do modelo de controle consiste na principal base para geração automática da maioria dos tipos de requisitos propostos nesta Tese de Doutorado. Portanto, a qualidade das informações contidas na máquina de estados que representa a especificação de controle torna-se vital para garantir, ao final de tudo, a qualidade dos requisitos gerados automaticamente a partir do modelo descritivo do sistema de interesse.

Ambrosio (2005) propõe uma abordagem para validação de *software* em aplicações espaciais baseada tanto em testes de conformidade (i.e. casos de teste) quanto em injeção de falhas (i.e. casos de falha). Ainda segundo a autora, a abordagem proposta consiste em gerar sequências abstratas de

testes para, então, derivar os casos de teste e casos de falha. Tais sequências abstratas derivam-se, por sua vez, de uma descrição formal do comportamento esperado para o *software* de interesse, comportamento este descrito através de uma máquina de estados finita.

Da mesma forma, o sucesso da abordagem proposta por Ambrosio (2005) depende da qualidade das informações contidas na máquina de estados representando o comportamento do *software* aplicativo. Para assegurar tal qualidade, a autora propõe uma metodologia prescritiva de como conceber a máquina de estados, a fim de garantir que todos os elementos tenham sido levados em conta. Esta metodologia pode, portanto, ser utilizada para auxiliar a elaboração da especificação de controle e, consequentemente, assegurar a qualidade dos requisitos gerados ao final do processo de modelagem do sistema de interesse.

Outro ponto observado durante a aplicação da metodologia proposta nesta Tese de Doutorado relaciona-se com a elaboração de alguns diagramas da SysML e sua utilidade na geração de requisitos.

Durante a elaboração do modelo de arquitetura, por estar baseado no método HHP de Hatley et al. (2000), criam-se Diagramas de Sequência da SysML para mostrar a troca de mensagens entre o sistema e as entidades externas presentes em seu ambiente, ou para mostrar a troca de mensagens entre os módulos de arquitetura constituintes do sistema de interesse. De acordo com a metodologia de geração automática dos requisitos de sistema apresentada no Capítulo 4, Seção 4.7, não se utilizam as informações contidas nestes Diagramas de Sequência nos enunciados dos requisitos gerados. Entretanto, notou-se que o modelo descritivo do sistema elaborado não contempla unicamente os requisitos do sistema modelado. Ele representa, também, a solução do sistema concebida para o problema, este sim representado pelos requisitos. Portanto, apesar de não ser utilizado na geração dos requisitos do sistema, as equipes de montagem e/ou verificação podem utilizar os Diagramas de Sequência criados no modelo para identificar

testes que garantam que o sistema foi corretamente montado e que suas interfaces estão funcionando de acordo com o esperado.

Outro diagrama elaborado na aplicação da metodologia proposta no Capítulo 4 é o Diagrama de Atividade da SysML para representar a especificação de uma função terminal, que originalmente no método HHP de Hatley et al. (2000) era feita através de um texto estruturado. Novamente, de acordo com a metodologia de geração automática dos requisitos de sistema apresentada no Capítulo 4, Seção 4.7, não se utilizam as informações contidas nestes diagramas nos enunciados dos requisitos gerados. Para o caso de uma especificação de uma dada função, seu Diagrama de Atividade representa, na realidade, a transformação das entradas nas saídas da função correspondente. Este diagrama completa, portanto, o requisito, cujo enunciado referencia o nome da função em questão.

Na linha da análise dos enunciados dos requisitos gerados a partir do modelo descritivo do sistema de interesse, observou-se, ainda, um critério subjetivo de avaliação da qualidade de um dado requisito. No Capítulo 5, Seção 5.4, utilizou-se um indicador para medir, de forma objetiva, a qualidade de um requisito de acordo com a análise sintática proposta por Halligan (1993). Entretanto, aproximando-se de uma análise semântica mais subjetiva, notou-se que mesmo os requisitos apresentando, quando comparados, valores idênticos para o IQR (Indicador de Qualidade de Requisito), os requisitos gerados pela metodologia proposta no Capítulo 4 possuem uma maior oportunidade de apresentarem uma qualidade melhor do que os requisitos produzidos de forma puramente textual. Isso porque, conforme INCOSE (2017), os termos-chave utilizados nos enunciados dos requisitos devem sempre estar definidos de forma precisa em um glossário.

Por exemplo, considerando a comparação feita no Apêndice F, o requisito ‘2.04.004’ extraído de AESP14 (2014) e o requisito ‘REQ_AESP14_080’ gerado a partir do modelo SysML do AESP14 apresentaram valores de IQR iguais a 50%. Ambos os requisitos utilizam em seus enunciados termos como “ambiente espacial” e “radiação cósmica”. Entretanto, no requisito ‘2.04.004’

não existe garantia de que estes termos sejam interpretados de maneira livre de ambiguidades, mesmo estando presentes em um glossário, pois eles continuariam ser definidos em linguagem natural, onde ambiguidades são inerentes. Já no requisito ‘REQ_AESP14_080’, estes termos são, na realidade, referências para elementos do modelo SysML que, por construção, apresentam uma interpretação única.

7 CONCLUSÃO

Neste capítulo, apresentam-se as conclusões desta Tese de Doutorado, estruturadas da seguinte maneira. Nas Seções 7.1 e 7.2, comenta-se o atingimento dos objetivos geral e específicos, ambos destacados no Capítulo 1, Seção 1.3. Já na Seção 7.3, faz-se um resumo das contribuições deste trabalho. Na Seção 7.4, mostram-se eventuais limitações da metodologia proposta. Finalmente, na Seção 7.5, destacam-se os possíveis trabalhos futuros, decorrentes desta Tese de Doutorado.

7.1 Atingimento dos Objetivos Geral e Específicos

Conforme visto no Capítulo, Seção 1.3, esta Tese de Doutorado teve como objetivo geral a tradução em requisitos textuais de um modelo descritivo de um sistema de engenharia, elaborado utilizando a linguagem de modelagem SysML. Para isso, definiram-se os seguintes objetivos específicos: apresentar uma metodologia de modelagem descritiva de sistemas de engenharia utilizando a SysML, aplicar esta metodologia em um sistema espacial já desenvolvido, comprar os requisitos textuais originais com aqueles traduzidos do modelo elaborado e, finalmente, propor uma discussão em torno da metodologia proposta e os resultados obtidos.

No Capítulo 4, Figura 4.1, apresentaram-se as atividades envolvidas na metodologia proposta para modelagem descritiva de sistemas de engenharia utilizando a SysML e a atividade de geração de requisitos textuais a partir do modelo criado. Os principais objetivos de cada atividade foram abordados nas cinco primeiras seções do referido capítulo.

No Capítulo 4, Seção 4.6, apresentaram-se, em detalhes, os passos para elaborar o modelo SysML do sistema de interesse. Já no Capítulo 4, Seção 4.7, mostrou-se como realizar a tradução do modelo criado em requisitos textuais.

No Capítulo 5, Seção 5.2, aplicou-se a metodologia proposta no Capítulo 4 a um sistema espacial que já possuía a etapa de engenharia de sistemas realizada, o AESP14. Na referida seção, apresentou-se o modelo SysML

elaborado para este sistema e, na Seção 5.3, apresentaram-se alguns requisitos que foram gerados a partir do modelo criado. Ainda no Capítulo 5, Seção 5.4, apresentou-se uma comparação qualitativa entre os requisitos originais do AESP14 e aqueles traduzidos do modelo SysML para ele criado.

Finalmente, no Capítulo 6, discutiram-se as escolhas feitas para propor a metodologia do Capítulo 4, os resultados obtidos pela sua aplicação e as contribuições observadas ao aplicá-la.

7.2 Resumo das Contribuições

Considerando que a má qualidade de requisitos é uma das causas mais frequentes de fracassos no desenvolvimento de sistemas e que este tema vem sendo constantemente abordado na literatura, indicando que uma solução sistemática para este problema ainda não é visível no horizonte da engenharia de requisitos, abordagens que melhorem a qualidade de requisitos ainda possuem espaço para discussões e aprofundamentos. Mais ainda, o uso de modelos descritivos de sistemas como o artefato gerenciado pelas atividades da engenharia de sistemas vem crescendo ao longo do tempo, entretanto, o seu uso para a extração automática de requisitos ainda é muito pouco explorado.

A contribuição desta pesquisa visa exatamente preencher esta lacuna e aproveitar o já elevado esforço despendido na elaboração de um modelo completo e consistente de um sistema para, então, poupar esforço na escrita de seus requisitos e, ainda, melhorar sua qualidade. Com isso, colabora-se diretamente para redução do custo da não-qualidade ao longo do ciclo de vida do sistema de interesse.

Conforme discutido no Capítulo 6, Seção 6.2, os padrões vistos para a escrita de requisitos possuem oportunidades para melhorar sua qualidade, que foram exploradas no principal padrão para escrita de requisitos proposto nesta Tesa de Doutorado, discutido ainda na mencionada seção. Além disto, a linha de pesquisa aqui desenvolvida vai ao encontro da conclusão de Nicolás e Toval (2009) apresentada no Capítulo 3, Figura 3.2, onde mais

esforços deveriam ser aplicados à utilização de técnicas da UML voltadas para a engenharia de requisitos, que não utilizem apenas casos de uso e cenários para a geração de requisitos.

Ao dispor de uma metodologia prescritiva de como criar um modelo descritivo, o engenheiro de sistema tem à sua disposição uma ferramenta que lhe auxilia a não se esquecer dos principais pontos que devem ser observados ao se elaborar uma especificação e arquitetura do sistema de interesse, de forma completa e concisa. Mais ainda, ao se utilizar uma linguagem de modelagem padronizada, facilita-se a comunicação entre as diferentes equipes e pessoas, bem como possibilita um aprendizado e implantação da metodologia de forma mais ágil, pois certamente existirão à disposição, variados cursos de formação e ferramentas de modelagem com suporte a esta linguagem.

O uso das informações contidas em um modelo descritivo de um sistema para preencher lacunas em padrões de enunciados de requisitos faz com que os requisitos de um mesmo conjunto estejam sempre seguindo um mesmo padrão sintático, campos importantes não sejam negligenciados, ou pelo menos, suas ausências sejam facilmente detectáveis e, finalmente, termos utilizados nos enunciados estejam sempre precisamente definidos no modelo (e.g.: dicionário de requisitos, dicionário de arquitetura, especificações de funções, etc), assim como orientado em INCOSE (2017), onde termos-chave contidos nos enunciados dos requisitos devem sempre estar definidos em um glossário.

7.3 Limitações

Por se tratar de uma extensão da UML, onde estruturas de modelagem para comportamentos de um *software* apresentam-se de forma mais abundante do que estruturas para modelar suas eventuais restrições físicas, a versão 1.5 da especificação da SysML, última versão disponível quando esta Tese de Doutorado foi elaborada, carece, igualmente, de estruturas para modelagem de restrições físicas, comumente encontradas no âmbito de um sistema. Mais ainda, com a remoção dos Diagramas de Tempo da UML, a

SysML perdeu ainda mais construções para a representação de restrições dentro de um modelo.

Pela metodologia proposta nesta Tese de Doutorado, no Capítulo 4, elementos do tipo Restrição da SysML foram utilizados para representar as restrições dentro do modelo de requisitos (e.g.: restrições de desempenho de uma função, etc). Estes elementos basicamente baseiam-se em expressões textuais, ou seja, não existe um formalismo imposto pela linguagem. Mais ainda, os principais requisitos não-funcionais propostos, aqueles baseados nos enunciados estabelecidos por Carson (2015) para restrições de projeto, ambientais e de adequabilidade, incorporam-se ao modelo SysML através de Diagramas Paramétricos. Entretanto, a maneira proposta para elaboração de tais diagramas ainda baseia-se fortemente em textos e pouco se faz uso dos elementos e relacionamentos da SysML.

Este uso extensivo de linguagem natural nas restrições do sistema aumenta a possibilidade de ambiguidades e deixa o engenheiro de sistemas sem uma regra objetiva para estas definições. Entretanto, esta característica é inerente à escolha da SysML como linguagem de modelagem.

7.4 Trabalhos Futuros

Como possíveis desdobramentos da pesquisa elaborada nesta Tese de Doutorado, identificam-se possíveis linhas de pesquisa para a verificação automática de um modelo descritivo de um sistema em SysML, concebido utilizando a metodologia proposta no Capítulo 4, Seção 4.6. Da mesma forma que Halligan (1993) definiu uma métrica para determinar a qualidade de requisitos, seria possível elaborar uma métrica para avaliar a qualidade do modelo criado e que será posteriormente utilizado para geração automática dos requisitos de sistema. Mais ainda, poderia se determinar, através desta mesma métrica, qual nível de maturidade necessário para o modelo de um dado sistema em cada fase de seu projeto de desenvolvimento.

No mecanismo proposto no Capítulo 4, Seção 4.7, para a geração automática dos requisitos de sistema a partir de um modelo em SysML,

criado seguindo a metodologia proposta no Capítulo 4, Seção 4.6, dois pontos poderiam ser explorados de formas diferentes em possíveis trabalhos futuros: a geração de requisitos não-funcionais e a geração de requisitos funcionais a partir das especificações de funções.

Pelo mecanismo aqui proposto, os principais requisitos não-funcionais se baseiam nos enunciados estabelecidos por Carson (2015) para restrições de projeto, ambientais e de adequabilidade. A maneira proposta nesta Tese de Doutorado para incorporá-las no modelo SysML foi utilizando Diagramas Paramétricos, entretanto, ainda fortemente baseados em textos e pouco em elementos e relacionamentos da SysML. Esta característica se deve ao fato da SysML ser extremamente potente e repleta de recursos para a modelagem comportamental, pois se trata de uma extensão da UML, onde comportamentos de um *software* recebem, naturalmente, uma maior atenção, mas, por outro lado, carece de estruturas para a modelagem de restrições. Uma futura linha de pesquisa poderia, por exemplo, se aprofundar em como representar estes mesmos requisitos não-funcionais de uma maneira mais formal do que aquela proposta aqui, com o uso de Diagrama Paramétricos da SysML.

No método HHP de Hatley et al. (2000), uma função terminal (i.e. uma função de último nível na decomposição funcional) é definida através de uma PSPEC, onde se estabelece de maneira precisa como as saídas da função são obtidas a partir de suas entradas. Pode-se dizer que uma PSPEC define o processamento da sua correspondente função terminal. No mecanismo proposto no Capítulo 4, Seção 4.7, para cada função terminal gera-se um correspondente requisito funcional, onde o verbo (ou ação) do enunciado deste requisito é o próprio nome da função. A correspondente especificação da função fica sendo, portanto, um complemento do requisito funcional, assim como acontece para as saídas e entradas da função, onde no enunciado do requisito utilizam-se os nomes dos elementos do modelo que os representam, mas suas definições precisas se encontram nos dicionários de dados do modelo (i.e. dicionário de requisitos e dicionário de arquitetura).

Esta abordagem foi escolhida, pois uma especificação de função é criada utilizando-se um Diagrama de Atividade da SysML para a correspondente função, repleto de possibilidades de construções e estruturas. Para evitar restringir o engenheiro de sistemas no uso de construções para a elaboração destes diagramas, optou-se por não gerar os requisitos funcionais a partir diretamente das especificações de funções. Entretanto, linhas de pesquisa futuras poderíamos explorar diferentes abordagens e achar um subconjunto suficiente de construções para um Diagrama de Atividade, tornando possível a extração automática de requisito diretamente destas especificações.

Uma outra possibilidade para abordar as duas problemáticas descritas acima seria a utilização de diagramas nos enunciados dos requisitos. Ou seja, em vez de realizar uma tradução do modelo em requisitos 100% textuais, como o que foi feito nesta Tese de Doutorado, poderia se pesquisar uma maneira de se utilizar os próprios diagramas produzidos ao elaborar o modelo do sistema de interesse nos enunciados dos requisitos dele extraídos.

Finalmente, a metodologia proposta no Capítulo 4, Seção 4.6, se apoia em dois principais pilares: o método base HHP de Hatley et al. (2000) e a linguagem de modelagem SysML. Futuras linhas de pesquisa poderiam explorar variações destes dois pilares para construções de outras metodologias de modelagem que permitam a extração automática de requisitos de sistema, apontando, ao final, as vantagens e desvantagens de cada uma delas. Por exemplo, com algumas adaptações seria possível aplicar o mecanismo de geração automática de requisitos proposto nesta Tese de Doutorado ao método de projeto de sistema proposto por Friedenthal e Oster (2017).

Com isso, conclui-se esta Tese de Doutorado, onde desenvolveu-se um método de modelagem descritiva de sistemas de engenharia para possibilitar a geração automática de requisitos textuais.

REFERÊNCIAS BIBLIOGRÁFICAS

AEROSPACE RECOMMENDED PRACTICE (ARP). **ARP4754**: certification considerations for highly-integrated or complex aircraft systems. Warrendale, PA, USA, ago. 2000. 88 p.

AESP14: requisitos do sistema. Instituto nacional de pesquisas espaciais (INPE), 2014. 20p. Programa Cubesat. (LIT21-AESP14-ES-004)

AESP-14 AX.25 telemetry decoder. 2015. Version 1.0. Disponível em <http://www.aer.ita.br/~aesp14/AESP-14Telemetry.pdf>. Acesso em: 01 jun. 2017.

ALEXANDER, I.; BEUS-DUKIC, L. **Discovering requirements**: how to specify products and services. West Sussex, England: John Wiley & Sons Ltd, 2009. 478 p. ISBN(978-0-470-71240-5).

ALEXANDER, I.; STEVENS, R. **Writing better requirements**. London, UK: Pearson Education Limited, 2002, 98 p. ISBN(0-321-13163-0).

AMBROSIO, A. M. **COFI**: uma abordagem combinando teste de conformidade e injeção de falhas para validação de software em aplicações espaciais. 2005. 209 p. Tese – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, São Paulo, Brasil, 2005. (INPE-13264-TDI/1031).

BLOOMBERG, J.; SCHMELZER, R. **Service orient or be doomed**: how service orientation will change your business. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2006. 274 p. ISBN(978-0471-76858-6).

BOOCH, G.; MAKSIMCHUK, R. A.; ENGLE, M. W.; YOUNG, B. J.; CONALLEN, J.; HOUSTON, K. A. **Object-oriented analysis and design with applications**. 3ª ed. Boston, MA, USA: Pearson Education, Inc., 2007. 717 p. ISBN(0-201-89551-X).

BUEDE, D. M. **The engineering design of systems**: models and methods. 2ª ed. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2009. 524 p. ISBN(978 0 470 16402 0).

CANTOR, M. **Rational unified process for systems engineering**. RUP SE Version 2.0. IBM Rational Software. 2003. IBM Rational Software White Paper.

CARSON, R. S. Implementing structured requirements to improve requirements quality. In: INCOSE INTERNATIONAL SYMPOSIUM, 25., 2015, Seatle, WA, USA. **Proceedings...** INCOSE, 2014, p. 54-67. DOI: 10.1002/j.2334-5837.2015.00048.x.

COAD, P.; YOURDON, E. **Object-oriented analysis**. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc., 1991a. 233 p. ISBN(0-13-629981-4).

COAD, P.; YOURDON, E. **Object-oriented design**. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc., 1991b. 233 p. ISBN(0-13-630070-7).

COUTO, R.; RIBEIRO, A. N.; CAMPOS, J. C. Application of ontologies in identifying requirements patterns in use cases. In: INTERNATIONAL WORKSHOP ON FORMAL ENGINEERING APPROACHES TO SOFTWARE COMPONENTS AND ARCHITECTURES, 11., 2014, Grenoble, France. **Proceedings...** Prague, 2014, p. 62-76. ISSN 2075-2180.

CROSBY, P. B. **Quality is free**: the art of making quality certain. New York, NY, USA: McGraw-Hill Companies, 1979. 309 p. ISBN(978-0070145122).

DEEPTIMAHANTI, D. K.; SANYAL, R. Semi-automatic generation of UML models from natural language requirements. In: INDIA SOFTWARE ENGINEERING CONFERENCE, 4., 2011, Thiruvananthapuram, Kerala, India. **Proceedings...** New York, NY, USA: ACM, 2011. p. 23-27. ISBN 978-1-4503-0559-4.

DELLIGATTI, L. **SysML distilled:** a brief guide to the systems modeling language. Upper Saddle River, New Jersey, USA: Pearson Education, Inc., 2014. 301 p. ISBN(978-0-321-92786-6).

DEMARCO, T. **Structured analysis and system specification**. New York, NY, USA: Yourdon Inc., 1979. 352 p. ISBN(0-917072-07-3).

DEMING, W. E. **Out of the crisis**. Cambridge, MA, USA: Massachusetts Institute of Technology, 1982. 520 p. ISBN(978-0911379013).

DEPARTMENT OF DEFENSE (DoD). **Systems engineering fundamentals**. Fort Belvoir, Virginia, USA: Defense Acquisition University Press, 2001. 222 p.

DORI, D. **Object-process methodology**: a holistic system paradigm. New York, NY, USA: Springer, 2002. 455 p. ISBN(978-3-540-65471-1).

DOUGLASS, B. P. **The harmony process**. Andover, MA, USA: I-Logix Inc., 2005. 21 p. I-Logix White Paper.

ESTEFAN, J. A. **Survey of model-based systems engineering (MBSE) methodologies**. Revision B. Seattle, WA, USA: INCOSE MBSE Initiative, 2008. 80 p. (INCOSE-TD-2007-003-01)

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **Glossary of terms**. Noordwijk, Netherlands: ECSS, 2012. 58 p. (ECSS-S-ST-00-01C).

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS). **System engineering general requirements**. Noordwijk, Netherlands: ECSS, 2009. 100 p. (ECSS-E-ST-10C).

FEILER, P.; GLUCH, D. **Model-based engineering with AADL**: an introduction to the SAE architecture analysis and design language. Upper Saddle River, New Jersey, USA: Pearson Education, Inc., 2013. 468 p. ISBN(978-0-321-88894-5).

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML**: the systems modeling language. 3ª ed. Waltham, MA, USA: Elsevier Inc., 2014. 630 p. ISBN(978-0-128-00202-5).

FRIEDENTHAL, S.; OSTER, C. **Architecting spacecraft with SysML**. USA: Createspace Independent Publishing Platform, 2017. 130 p. ISBN(978-1544288062).

GENESERETH, M. R.; NILSSON, N. **Logical foundations of artificial intelligence**. San Mateo, CA: Morgan Kaufmann Publishers, 1987.

GUO, J.; ZHANG, Z.; WANG, Y. Model-driven derivation of domain functional requirements from use cases. **Journal of Software Engineering and Applications**, v. 3, n. 9, p. 875-881, 2010.

HALLIGAN, R. **Requirements analysis that works**. 2012. Disponível em <http://www.ppi-int.com/systems-engineering/downloads.php>. Acesso em: 01 jun. 2017.

HALLIGAN, R. Requirements quality metrics: the basis of informed requirements engineering management. In: COMPLEX SYSTEMS ENGINEERING SYNTHESIS AND ASSESSMENT TECHNOLOGY WORKSHOP (CSESAW '93), 1993. **Proceedings...** Calvados, MD, USA: NSWC, 1993.

HAREL, D. Statecharts: A visual approach to complex systems. **Science of Computer Programming**, v. 8 n. 7 p. 231-274, 1987.

HARTONG, M.; GOEL, R.; WIJESEKERA, D. Security requirement derivation by noun–verb analysis of use–misuse case relationships: a case study using positive train control. **Innovations in Systems and Software Engineering Journal**, v. 10, n. 2, p. 103-122, 2013.

HATLEY, D.; PIRBHAI, I. **Strategies for real-time system specification**. New York, NY, USA: Dorset House Publishing Co., Inc., 1988. 386 p. ISBN(0-932633-11-0).

HATLEY, D.; HRUSCHKA, P.; PIRBHAI, I. **Process for system architecture and requirements engineering**. New York, NY, USA: Dorset House Publishing Co., Inc., 2000. 434 p. ISBN(0-932633-41-2).

HOLT, J.; PERRY, S. **SysML for systems engineering**: a model-based approach. 2ª ed. London, UK: The Institution of Engineering and Technology, 2013. 930 p. ISBN(978-1-84919-651-2).

HOOKS, I. Writing good requirements. In: THIRD INTERNATIONAL SYMPOSIUM OF THE NCOSE – Volume 2, 1993. **Proceedings...** San Diego, CA, USA: INCOSE, 1993.

HOOKS, I. F.; FARRY, K. A. **Customer-centered products**: creating successful products through smart requirements management. New York, NY, USA: AMACOM, 2001. 271 p. ISBN(0-8144-0568-1).

HULL, E.; JACKSON, K.; DICK, J. **Requirements engineering**. 3ª ed. London, UK: Springer, 2011. 227 p. ISBN(978-1-84996-404-3).

INGHAM, M. D.; RASMUSSEN, R. D.; BENNETT, M. B.; MONCADA, A. C. Generating requirements for complex embedded systems using state analysis. **Acta Astronautica**, v. 58, n. 12, p. 648–661, Jun 2006.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE). **IEEE STD 1220–2005**: standard for application and management of the systems engineering process. New York, NY, USA, set. 2005. 87 p.

INTERNATIONAL CONCIL ON SYSTEMS ENGINEERING (INCOSE). **Guide for writing requirements**. San Diego, CA, USA: INCOSE, 2017, 102 p. (INCOSE-TP-2010-006-02.1)

INTERNATIONAL CONCIL ON SYSTEMS ENGINEERING (INCOSE). **INCOSE systems engineering handbook**. 4ª ed. San Diego, CA, USA: INCOSE, 2015, 305 p. (INCOSE-TP-2003-002-04)

INTERNATIONAL CONCIL ON SYSTEMS ENGINEERING (INCOSE). **Systems engineering vision 2020**. Version 2.03. San Diego, CA, USA: INCOSE, 2007. 32 p. (INCOSE-TP-2004-004-02).

INTERNATIONAL CONCIL ON SYSTEMS ENGINEERING (INCOSE). **Systems engineering vision 2025**. San Diego, CA, USA: INCOSE, 2014. 30 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). **ISO/IEC 14977**. Extended Backus-Naur form (EBNF). Information technology – Syntactic metalanguage – Extended BNF. Geneve, Switzerland, 1996. 24 p.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO). **ISO/IEC 19514**. Information technology – object management group systems modeling language (OMG SysML). Geneve, Switzerland, 2017. 327 p.

JENNEY, J. **Modern methods of systems engineering**: with an introduction to pattern and model based methods. 2010. ISBN(978-1463777357).

JURAN, J. M; GRYNA, F. M. **Juran's quality control handbook**. 4ª ed. New York, NY, USA: McGraw-Hill, 1988. 1774 p. ISBN(978-0070331761).

KRUCHTEN, P. **The rational unified process**: an introduction. 3ª ed. Reading, MA, USA: Addison-Wesley Professional, 2003. ISBN(978-0321197702).

LEMPIA, D.; SCHINDEL, W. D.; HRABIK, T.; MCGILL, S.; GRABER, M. Using visual diagrams and patterns for consistent and complete requirements. In: INCOSE 2016 INTERNATIONAL SYMPOSIUM, 26., 2016, Edinburgh, Scotland, UK. **Proceedings...** Wiley Online Library, 2016, p. 415-429. Online ISSN(2334-5837).

LEOPOLD, H.; MENDLING J.; POLYVYANYY, A. Generating natural language texts from business process models. In: CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, 24., 2012, Gdansk, Poland. **Proceedings...** Berlin: Springer Berlin Heidelberg, 2012. p. 64-79. ISBN 978-3-642-31094-2.

LONG, D.; SCOTT, Z. **A primer for model-based systems engineering**. Blacksburg, Virginia, USA: Vitech Corporation, 2011. 122 p. ISBN(978-1-105-58810-5).

LOUREIRO, G. **A systems engineering and concurrent engineering framework for the integrated development of complex products**. 1999. 607 p. Tese – Loughborough University, Leicestershire, UK, 1999.

LYKINS, H.; FRIEDENTHAL S.; MEILICH A. Adapting UML for an object-oriented systems engineering method (OOSEM). In: INCOSE 2000 INTERNATIONAL SYMPOSIUM. **Proceedings...** Minneapolis, MN, USA: INCOSE, jul. 2000.

MAHMOOD, A.; KHATOON, S. An approach to generate test goals from use case scenarios. **Information Technology Journal**, v. 12, n. 8, p. 1600-1606, 2013.

Message-Digest algorithm 5 (MD5): request for comments 1321. MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992. Disponível em <https://tools.ietf.org/html/rfc1321>. Acesso em: 01 jun. 2017.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA). **NASA systems engineering handbook**. Washington, D.C, USA: NASA, 2007. 360 p. (NASA/SP-2007-6105).

NICOLÁS, J.; TOVAL, A. On the generation of requirements specifications from software engineering models: a systematic literature review. **Information and Software Technology**, v. 51, n. 9, p. 1291–1307, September 2009.

OBJECT MANAGEMENT GROUP (OMG). **Business process model and notation (BPMN)**. Needham, MA, USA: OMG, 2013. 532 p. (formal/2013-12-09).
Disponível em <http://www.omg.org/spec/BPMN/2.0.2/PDF>.
Acesso em: 01 jun. 2017.

OBJECT MANAGEMENT GROUP (OMG). **Meta object facility (MOF)**. Needham, MA, USA: OMG, 2014. 82 p. (formal/2014-04-05).
Disponível em <http://www.omg.org/spec/MOF/ISO/19508/PDF/>.
Acesso em: 01 jun. 2017.

OBJECT MANAGEMENT GROUP (OMG). **MOF model to text transformation (MOFM2T)**. Needham, MA, USA: OMG, 2008. 48 p. (formal/2008-01-16).
Disponível em <http://www.omg.org/spec/MOFM2T/1.0/PDF>.
Acesso em: 01 jun. 2017.

OBJECT MANAGEMENT GROUP (OMG). **System modeling language (SysML)**. Needham, MA, USA: OMG, 2017. 362 p. (formal/2017-05-01).

Disponível em <http://www.omg.org/spec/SysML/1.5/PDF>.
Acesso em: 01 jun. 2017.

OBJECT MANAGEMENT GROUP (OMG). **Unified modeling language (UML)**. Needham, MA, USA: OMG, 2015. 794 p. (formal/2015-03-01). Disponível em <http://www.omg.org/spec/UML/2.5/PDF>.
Acesso em: 01 jun. 2017.

PELEG, M.; DORI D. From object-process diagrams to a natural object-process language. In: INTERNATIONAL WORKSHOP, NGITS'99, 4., 1999, Zikhron-Yaakov, Israel. **Proceedings...** Berlin: Springer Berlin Heidelberg, 1999. p. 221-228. ISBN 978-3-540-66225-9.

PROJECT MANAGEMENT INSTITUTE (PMI). **Um guia do conhecimento em gerenciamento de projetos:** guia PMBOK. 5ª ed. Newtown Square, Pennsylvania, USA: Project Management Institute, Inc., 2013. 595 p. (ISBN: 978-1-62825-007-7).

RAGO, A.; MARCOS, C.; DIAZ-PACE, J. A. Uncovering quality-attribute concerns in use-case specifications via early aspect mining. **Requirements Engineering**, v. 18, n. 1, p. 67-84, 2013.

REINHARTZ-BERGER, I.; DORI, D. Object-process methodology (OPM) vs. UML: a code generation perspective. In: CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, 16., 2004, Riga, Latvia. **Proceedings...** Berlin: Springer Berlin Heidelberg, 2004. p. 275-286. ISBN 978-3-540-22151-7.

ROBERTSON, S.; ROBERTSON, J. **Mastering the requirements process**: getting requirements right. 3ª ed. Upper Saddle River, New Jersey, USA: Pearson Education, Inc., 2012. 768 p. ISBN(978-0-321-81574-3).

RYAN, M. J.; WHEATCRAFT, L. S. On a cohesive set of requirements engineering terms. **Systems Engineering**, v. 20, n. 2, p. 118-130, March 2017. Online ISSN(1520-6858).

SCHINDEL, W. D. Requirements statements are transfer functions: an insight from model-based systems engineering. In: INCOSE 2005 INTERNATIONAL

SYMPOSIUM, 15., 2005, Rochester, NY, USA. **Proceedings...** Wiley Online Library, 2005, p. 1604-1618. Online ISSN(2334-5837).

SCHINDEL, W. D. What is the smallest model of a system. In: INCOSE 2011 INTERNATIONAL SYMPOSIUM, 21., 2011, Denver, CO, USA. **Proceedings...** Wiley Online Library, 2011, p. 99-113. Online ISSN(2334-5837).

SCHINDEL, W. D. **MBSE methodology summary**: pattern-based systems engineering (PBSE), based on S*MBSE models. V1.5.5A. INCOSE Patterns Working Group, 2015.
Disponível em <http://www.omgwiki.org/MBSE/doku.php?id=mbse:pbse> Acesso em: 01 jun. 2017.

SHAMIEH, C. **Systems engineering for dummies**. IBM Limited Edition. Hoboken, NJ, USA: Wiley Publishing Inc., 2011. 74 p. ISBN(978-1-118-10001-1).

STEVENS, R.; BROOK, P.; JACKSON, K.; ARNOLD, S. **Systems engineering**: coping with complexity. Harlow, Essex, England: Pearson Education Limited, 1998. 374 p. ISBN(978-0-13-095085-7).

YOUNG, R. R. **The requirements engineering handbook**. Norwood, MA, USA: Artech House, Inc., 2004. 275 p. ISBN(1-58053-266-7).

YOURDON, E. **Modern structured analysis**. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc., 1989. 672 p. ISBN(0-13-598624-9).

YOURDON, E.; CONSTANTINE, L. **Structured design**: fundamentals of a discipline of computer program and systems design. New York, NY, USA: Yourdon Inc., 1975. 446 p. ISBN(0-917072-11-1).

WEILKIENS, T. **Systems engineering with SysML/UML**: modeling, analysis, design. Burlington, MA, USA: Elsevier Inc., 2007. 320 p. ISBN(978-0-12-374274-2).

WYMORE, A. W. **Model-based systems engineering**. Boca Raton, Florida, USA: CRC Press LLC, 1993. 710 p. ISBN(0-8493-8012-X).

GLOSSÁRIO

análise estruturada – técnica para construção de modelos de sistemas, com a finalidade de retratar o fluxo e o conteúdo das informações utilizadas pelo sistema, dividir o sistema em partições funcionais e comportamentais e descrever a essência daquilo que será construído.

arquitetura de sistema – definição de como um sistema é dividido em partes e de como estas partes estão integradas de modo a proporcionar as funcionalidades definidas na especificação de sistema.

boilerplates – gabarito utilizado para escrita de requisitos, onde os elementos de um requisito são lacunas de uma sentença pré-montada.

ciclo de vida – estágios pelo qual um sistema passa desde a sua concepção até o seu descarte.

conceito – descrição do objeto investigado a partir de um entre inúmeros cenários contextuais possíveis (caracterização exógena e válida apenas diante da singularidade do universo pesquisado).

cubesat – nanossatélite escalável com unidade básica 1U (cubo de 10 cm de aresta) pesando até 1,33 kg.

datastore – elemento de modelagem da SysML utilizado em diagramas de atividade para representar o elemento de modelagem ‘armazenador’, comumente utilizado em abordagens baseadas em análise estruturada, nos diagramas de fluxo de dados, ou DFDs.

definição – descrição daquilo que o objeto investigado tem de específico e distinto em relação aos demais (caracterização endógena e universal).

engenharia de requisitos – abordagem para entender e acordar o problema a ser resolvido por um sistema que será desenvolvido.

engenharia de sistemas – abordagem multidisciplinar para derivar, evoluir e verificar uma solução balanceada ao longo do ciclo de vida do sistema que satisfaça as necessidades dos *stakeholders*.

especificação de requisitos – documento de requisitos produzido pela atividade de engenharia de requisitos.

especificação de sistema – documento produzido pela atividade de engenharia de sistema contendo a solução para o problema definido pela especificação de requisitos.

framework – infraestrutura conceitual básica que provê suporte a algo.

função – comportamento apresentando por um sistema ao transformar material, energia ou informação.

função terminal – última função na hierarquia funcional criada pelo processo de decomposição ou desdobramento de uma função de alto nível.

hardware – parte física que compõe um aparelho ou dispositivo.

metamodelo – *framework* para construção de modelos.

método – igual a metodologia.

metodologia – conjunto de regras que, ao serem aplicadas a um problema específico, produzem o resultado desejado.

modelagem – conjunto de atividades, cujo objetivo final é a concepção de um modelo.

modelo – representação de uma abstração do sistema real que evidencia somente seus aspectos relevantes ao estudo sendo conduzido.

modelo descritivo – uso de elementos de uma dada linguagem de modelagem, relacionados entre si através das regras gramaticais nela estipuladas, criando, assim, estruturas semânticas que descrevem um sistema em termos de sua composição e comportamentos nos seus diversos cenários de utilização.

ontologia – conjunto dos termos utilizados em uma metodologia e como estes termos estão relacionados entre si, formando, assim, o conjunto de regras desta metodologia.

processo – conjunto de atividades que quando executadas produzem os artefatos estipulados com a qualidade necessária.

requisito – capacidade, característica ou qualidade que o sistema de interesse deve apresentar, com o objetivo de satisfazer uma ou mais necessidades de um ou mais de seus *stakeholders*.

ripple – componente de corrente alternada que se sobrepõe ao valor médio da tensão de uma fonte de corrente contínua.

sistema – conjunto de partes que interagem entre si para entregar capacidades aos seus *stakeholders* que não são disponibilizadas pelas suas partes individualmente.

software – conjunto de instruções computacionais com o objetivo de executar tarefas específicas quando em funcionamento.

solução sistema – especificação e arquitetura do sistema de interesse, levando em conta todo o seu ciclo de vida, desde sua concepção até o seu descarte, e que satisfaça as necessidades de seus *stakeholders*.

stakeholder – pessoa ou organização que possui interesse, afeta ou é afetada pelo desenvolvimento de um sistema, produto ou processo.

APÊNDICE A – DADOS UTILIZADOS PARA ESTATÍSTICAS

Abaixo, têm-se as tabelas contendo os dados utilizados para gerar os gráficos apresentados no Capítulo 1, Seção 1.1. A Tabela A.1 abaixo contém os dados utilizados para gerar o gráfico da Figura 1.1 e o gráfico da Figura 1.2. Já a Tabela A.2, na sequência, contém os dados utilizados para gerar o gráfico da Figura 1.3 e o gráfico da Figura 1.4. Finalmente, a Tabela A.3, no final deste capítulo, contém os dados utilizados para gerar o gráfico da Figura 1.5.

Tabela A.1 – Dados de Trabalhos Publicados em Simpósios do INCOSE.

Ano	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média	Desvio Padrão
Total	120	165	159	126	171	139	127	142	133	111	129	134	158	114	110	97	199	137,29	26,06
Requirement	34	44	42	39	68	42	48	31	31	27	41	35	46	28	37	27	52	39,53	10,50
Requirement %total	28,33%	26,67%	26,42%	30,95%	39,77%	30,22%	37,80%	21,83%	23,31%	24,32%	31,78%	26,12%	29,11%	24,56%	33,64%	27,84%	26,13%	28,75%	4,89%
Requirement %relativa	5,06%	6,55%	6,25%	5,80%	10,12%	6,25%	7,14%	4,61%	4,61%	4,02%	6,10%	5,21%	6,85%	4,17%	5,51%	4,02%	7,74%	5,88%	1,56%
Model	37	62	42	45	51	49	37	51	40	44	43	51	74	47	46	48	91	50,47	13,79
Model %total	30,83%	37,58%	26,42%	35,71%	29,82%	35,25%	29,13%	35,92%	30,08%	39,64%	33,33%	38,06%	46,84%	41,23%	41,82%	49,48%	45,73%	36,87%	6,66%
Model %relativa	4,31%	7,23%	4,90%	5,24%	5,94%	5,71%	4,31%	5,94%	4,66%	5,13%	5,01%	5,94%	8,62%	5,48%	5,36%	5,59%	10,61%	5,88%	1,61%
MBSE ou SysML	0	1	1	0	1	6	5	15	8	9	13	16	37	16	12	19	37	16,08	10,66
MBSE ou SysML %total	0,00%	0,61%	0,63%	0,00%	0,58%	4,32%	3,94%	10,56%	6,02%	8,11%	10,08%	11,94%	23,42%	14,04%	10,91%	19,59%	18,59%	11,79%	6,16%
MBSE ou SysML %relativa	0,00%	0,51%	0,51%	0,00%	0,51%	3,06%	2,55%	7,65%	4,08%	4,59%	6,63%	8,16%	18,88%	8,16%	6,12%	9,69%	18,88%	8,21%	5,44%

Fonte: Produção do Autor.

Tabela A.2 – Dados de Trabalhos Publicados em Periódicos do INCOSE.

Ano	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média	Desvio Padrão
Total	20	24	27	21	25	24	21	21	25	19	32	30	36	35	33	42	36	27,71	6,89
Requirement	4	13	5	6	6	10	4	5	5	4	6	5	8	14	9	13	3	7,06	3,51
Requirement %total	20,00%	54,17%	18,52%	28,57%	24,00%	41,67%	19,05%	23,81%	20,00%	21,05%	18,75%	16,67%	22,22%	40,00%	27,27%	30,95%	8,33%	25,59%	11,01%
Requirement %relativa	3,33%	10,83%	4,17%	5,00%	5,00%	8,33%	3,33%	4,17%	4,17%	3,33%	5,00%	4,17%	6,67%	11,67%	7,50%	10,83%	2,50%	5,88%	2,92%
Model	5	11	18	12	10	13	15	10	11	9	17	12	17	18	15	24	17	13,76	4,51
Model %total	25,00%	45,83%	66,67%	57,14%	40,00%	54,17%	71,43%	47,62%	44,00%	47,37%	53,13%	40,00%	47,22%	51,43%	45,45%	57,14%	47,22%	49,46%	10,58%
Model %relativa	2,14%	4,70%	7,69%	5,13%	4,27%	5,56%	6,41%	4,27%	4,70%	3,85%	7,26%	5,13%	7,26%	7,69%	6,41%	10,26%	7,26%	5,88%	1,93%
MBSE ou SysML	0	0	0	0	0	0	1	0	3	3	2	3	1	4	5	2	1	2,27	1,49
MBSE ou SysML %total	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	4,76%	0,00%	12,00%	15,79%	6,25%	10,00%	2,78%	11,43%	15,15%	4,76%	2,78%	7,79%	5,34%
MBSE ou SysML %relativa	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	4,00%	0,00%	12,00%	12,00%	8,00%	12,00%	4,00%	16,00%	20,00%	8,00%	4,00%	9,09%	5,96%

Fonte: Produção do Autor.

Tabela A.3 – Dados de Correspondências de Buscas no Google Acadêmico.

Ano	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	Média	Desvio Padrão
Writing Good Requirements	4	11	14	20	24	15	20	11	12	13	13	16	14	16	20	23	23	15,82	5,28
Writing Good Requirements %	1,49%	4,09%	5,20%	7,43%	8,92%	5,58%	7,43%	4,09%	4,46%	4,83%	4,83%	5,95%	5,20%	5,95%	7,43%	8,55%	8,55%	5,88%	1,96%
MBSE	10	13	16	10	11	11	30	37	66	140	171	234	402	379	463	580	629	188,35	217,89
MBSE %	0,31%	0,41%	0,50%	0,31%	0,34%	0,34%	0,94%	1,16%	2,06%	4,37%	5,34%	7,31%	12,55%	11,84%	14,46%	18,11%	19,64%	5,88%	6,80%
SysML	17	25	34	75	138	206	401	487	812	805	1080	1130	1360	1480	1560	1530	1690	754,71	622,77
SysML %	0,13%	0,19%	0,27%	0,58%	1,08%	1,61%	3,13%	3,80%	6,33%	6,27%	8,42%	8,81%	10,60%	11,54%	12,16%	11,93%	13,17%	5,88%	4,85%

Fonte: Produção do Autor.

APÊNDICE B – MÉTODO HHP ORIGINAL

A metodologia HHP, desenvolvida por Hatley et al. (2000), implementa o processo elaborado pelos mesmos autores e chamado PSARE (*Process for System Architecture and Requirements Engineering*). Nele, assume-se que o problema a ser resolvido já é conhecido e que os requisitos dos *stakeholders* já foram elicitados. A partir de então, divide-se o modelo do sistema em duas partes, a saber: modelo de requisitos e modelo de arquitetura. O primeiro modelo permite descrever o que o sistema deve fazer para satisfazer os requisitos dos *stakeholders*. O modelo de arquitetura permite a descrição da realidade física do sistema, ou seja, como ele será construído. O engenheiro de sistemas pode criar estes modelos para quantos níveis de abstração do sistema forem necessários ou desejados.

Através do modelo de requisitos essenciais, o engenheiro de sistemas captura as funções que o sistema deve implementar na sua essência (i.e. funções essenciais) e quão bem o sistema deve desempenhá-las. Em outras palavras, é através deste modelo que o engenheiro de sistemas identifica o comportamento que o sistema deve apresentar para alcançar a missão a ele atribuída, independentemente da tecnologia utilizada para seu desenvolvimento e, também, utilizada no ambiente onde ele será inserido. Entretanto, a tecnologia disponível para o desenvolvimento do sistema, bem como a tecnologia utilizada no ambiente onde o sistema será inserido, devem ser levadas em conta para se garantir a construção de um sistema realizável.

Para tal, o engenheiro de sistemas utiliza o modelo de requisitos melhorados para identificar novas funções que serão utilizadas para acoplar as funções essenciais ao mundo real, onde restrições tecnológicas existem e devem ser levadas em conta. Estas novas funções são classificadas em quatro tipos: funções de entrada, funções de saída, funções de interface com o usuário e, finalmente, funções de suporte.

As funções de entrada possuem o objetivo de transformar as entradas reais que chegam na fronteira do sistema sendo modelado nas entradas lógicas esperadas pelas funções essenciais. As funções de saída, por sua vez,

transformam as saídas lógicas produzidas pelas funções essenciais nas saídas reais que cruzam a fronteira do sistema. As funções de interface com o usuário tem como objetivo traduzir os dados lógicos produzidos pelas funções essenciais nos dados reais trocados com os usuários do sistema. Por último, as funções de suporte existem para tornar possível o correto funcionamento das funções essenciais, levando-se em conta as tecnologias existentes para construção do sistema e no seu ambiente operacional. São exemplos de funções de suporte aquelas relacionadas a redundância, manutenção, etc.

Através do modelo de arquitetura, o engenheiro de sistemas define as partes constituintes do sistema, chamadas de módulos de arquitetura, e como estes módulos estarão interligados entre si para trocar material, energia ou informação. Seguindo na mesma linha do modelo de requisitos, os módulos de arquitetura serão divididos nos seguintes cinco tipos: módulos essenciais, módulos de entrada, módulos de saída, módulos de interface com usuário e, finalmente, módulos de suporte. Cada um destes tipos representa o conjunto de módulos de arquitetura que alocam as correspondentes funções do mesmo tipo, ou seja, módulos essenciais alocam funções essenciais, módulos de entrada alocam funções de entrada, módulos de saída alocam funções de saída e assim por diante.

Este particionamento feito para as funções do sistema e, igualmente feito para os módulos do sistema, é chamado de gabarito para o modelo de requisitos e para o modelo de arquitetura, respectivamente, e está ilustrado na Figura B.1 abaixo.

Figura B.1 – Gabarito para os modelos de requisitos e arquitetura

Fonte: Adaptada de Hatley et al. (2000).

A figura acima apresenta o gabarito utilizado por Hatley et al. (2000) para classificar as funções e módulos de arquitetura de um sistema. A ideia a ser

transmitida nesta figura é de que as funções e os módulos de arquitetura essenciais, ou seja, aqueles que definem a essência do sistema, independentemente das tecnologias existentes e utilizadas, estão rodeados por funções e módulos de arquitetura adicionais que acoplam a parte essencial do sistema ao mundo real, estes, sim, restringidos por tais tecnologias.

B.1 Modelo de Requisitos Essenciais

Este modelo permite ao engenheiro de sistemas estabelecer o núcleo de requisitos independentes de tecnologia, ou seja, as funções precípuas do sistema. Ele é composto por três submodelos: modelo de entidade, modelo de processo e modelo de controle. Estes modelos são suplementados por um dicionário de requisitos, sendo este o repositório único das definições de todos os elementos usados nos modelos. Novamente, não existe prescrição estabelecida para a sequência de elaboração destes modelos, sendo eles criados de maneira interativa.

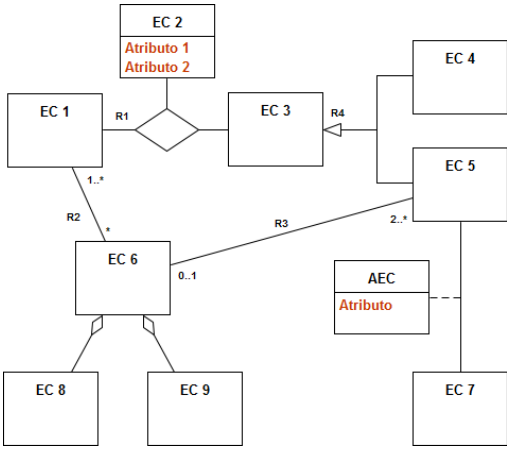
B.1.1 Modelo de Entidade

Este modelo permite ao engenheiro de sistemas descrever os dados (material, energia ou informação) do domínio de negócio do sistema de interesse. O nível de abstração utilizado aqui permite tornar transparante a tecnologia ou implementação envolvida por trás destes dados e mostra, portanto, o que o sistema sendo modelo deverá manipular na sua essência.

Os elementos deste modelo são: classe de entidade (ou simplesmente classe), atributos e relacionamentos. Uma classe de entidade é um item (ou tipo de itens) que o sistema deve utilizar ou armazenar. Um atributo é uma propriedade de uma classe de entidade que pode assumir diferentes valores. Um relacionamento, junto com seu nome, papel e multiplicidade, associa duas ou mais classes de entidade, formando uma estrutura de dados que deve ser tratada. São exemplos de relacionamentos: generalização e especialização, agregação e decomposição, etc.

Para criar o modelo de entidade, o engenheiro de sistemas pode utilizar um ou mais diagramas de classe de entidade, exemplificado na Figura B.2 abaixo e cuja sigla é ECD.

Figura B.2 – Diagrama de Classe de Entidade (ECD)



Fonte: Adaptada de Hatley et al. (2000).

Na figura acima, apresenta-se um ECD através de um exemplo com os seus elementos básicos. As caixas representam as classes de entidade. Os atributos de uma dada classe de entidade são colocados dentro da caixa que a representa. Os relacionamentos são representados por linhas sólidas com seus respectivos nomes e multiplicidades, interligando as classes de entidade neles envolvidas. Quando um relacionamento interligar mais de duas classes de entidade, faz-se necessário o uso de um losango, assim como exemplificado no relacionamento “R1” entre as classes de entidade “EC1”, “EC2” e “EC3”. Multiplicidades definem a quantidade de itens das classes de entidade que estão envolvidos no relacionamento, por exemplo, as multiplicidades do relacionamento “R2” entre as classes de entidade “EC1” e “EC6” definem que um ou mais itens da classe “EC1” estão relacionados com zero ou mais itens da classe “EC6”. Relacionamentos binários (i.e. aqueles que envolvem somente duas classes de entidade) e que possuem um diamante em uma das extremidades representam agregações (ou composições), onde a classe de entidade na extremidade do diamante agrega (ou é composta por) itens da classe de entidade na extremidade oposta ao diamante. Relacionamentos binários que possuem

uma seta cheia em uma das extremidades, como o relacionamento “R4” entre as classes de entidade “EC3”, “EC4” e “EC5”, representam generalizações (ou especializações), onde a classe de entidade na extremidade da seta representa uma generalização da classe de entidade na extremidade oposta à da seta, ou em outras palavras, a classe de entidade na extremidade oposta à da seta representa uma especialização da classe de entidade na extremidade da seta. Um outro tipo possível de relacionamento é o de classe de entidade de associação, representado na figura pelo relacionamento entre as classes de entidade “EC5” e “EC7”, ligada por uma linha tracejada à classe de entidade de associação “AEC”. O objetivo principal deste tipo de relacionamento é permitir ao engenheiro de sistemas especificar em detalhes um relacionamento, onde seus atributos serão especificados na sua correspondente classe de entidade de associação.

B.1.2 Modelo de Processo

Este modelo permite ao engenheiro de sistemas capturar as capacidades requeridas de processamento que o sistema deve apresentar em termos de funções e dados trocados entre elas. Os elementos básicos de modelagem são: os fluxos de dados, as funções, os terminadores (i.e. entidades externas) e armazenadores (i.e. repositórios de dados).

Para criar este modelo, o engenheiro de sistemas tem à sua disposição os diagramas de fluxo de dados, cuja sigla é DFD, e as especificações de processos, cuja sigla é PSPEC.

A criação deste modelo inicia-se através de um tipo especial de DFD chamado diagrama de contexto de dados, exemplificado na Figura B.3 abaixo e cuja sigla é DCD. Neste diagrama, mostra-se a função principal do sistema (ou missão do sistema) trocando os dados necessários com as entidades externas presentes no ambiente onde o sistema está inserido. Em seguida, o engenheiro de sistemas inicia a decomposição funcional da sua missão utilizando DFDs para cada nível de abstração desta decomposição, chamados de DFDs nivelados. Finalmente, quando a decomposição

funcional termina, o engenheiro de sistemas cria as especificações para cada função terminal (i.e. uma PSPEC para cada uma das funções terminais na árvore de decomposição funcional).

Figura B.3 – Diagrama de Contexto de Dados (DCD)

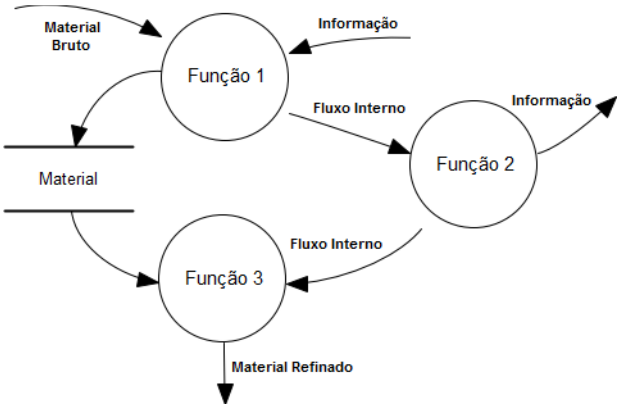
Fonte: Produção do Autor.

Na figura acima, exemplifica-se um DCD. Os retângulos representam os terminadores, ou seja, as entidades externas que trocam dados com o sistema. O círculo (ou bolha, como é comumente chamado por aqueles que utilizam DFDs nas suas especificações) no centro da figura representa a função precípua do sistema (ou missão do sistema). As linhas sólidas direcionadas representam os fluxos de dados trocados neste contexto.

Após a definição do contexto de dados, o engenheiro de sistemas inicia o desdobramento da função principal em subfunções, utilizando, para isso, DFDs nivelados, ou seja, diagramas para os diferentes níveis da decomposição funcional, sendo que um DFD representa o desdobramento de uma dada função. Um aspecto importante que deve ser observado durante o processo de decomposição funcional é o balanceamento dos DFDs, ou seja, as entradas e saídas de uma função sendo apresentada em um dado DFD devem coincidir com as entradas e saídas do DFD que a desdobra. Este detalhamento de funções continua até que o engenheiro de sistemas se julgue capaz de descrever tais funções do último nível da decomposição funcional através de uma especificação chamada PSPEC, podendo ser um texto estruturado, um gráfico, uma expressão matemática, etc. Esta especificação deve definir de forma concisa, completa e livre de ambiguidades, a transformação que a função deverá executar para produzir

as saídas esperadas a partir das entradas recebidas. Abaixo, na Figura B.4, tem-se um exemplo de um DFD desdobrando a função principal do sistema apresentada anteriormente na Figura B.3.

Figura B.4 – Diagrama de Fluxo de Dados (DFD)



Fonte: Produção do Autor.

Na figura acima, exemplifica-se a decomposição da missão do sistema, apresentada na Figura B.3, em três subfunções aqui representadas pelas bolhas assim identificadas. As duas barras paralelas na parte esquerda do diagrama representam um armazenador e as linhas direcionadas representam os fluxos de dados que acontecem neste contexto.

Quanto aos fluxos de dados, algumas observações devem ser feitas. Os dados que chegam em uma função e que não possuem um elemento de origem são os dados de entrada da função sendo decomposta e que foram identificados no DFD do nível de abstração superior a este sendo considerado. No exemplo da figura, os dados nomeados “Material Bruto” e “Informação” e que chegam na função “Função 1” são os dados de entrada da função principal do sistema, identificados no DCD da Figura B.3. Já os dados que saem das funções sem possuir um elemento de destino são os dados de saída da função sendo decomposta e que foram identificados no DFD do nível de abstração superior a este sendo considerado. No exemplo da figura acima, os dados “Material Refinado” e “Informação” que saem da função “Função 3” e da “Função 2”, respectivamente, são os dados de saída da função principal do sistema, identificados no DCD da Figura B.3. Fluxos de dados que possuem uma origem e um destino definidos no diagrama são

dados internos da função sendo desdobrada. No caso de armazenadores, os fluxos que ali chegam ou dali saem não precisam de identificação, pois possuem o mesmo tipo identificado para os próprios armazenadores.

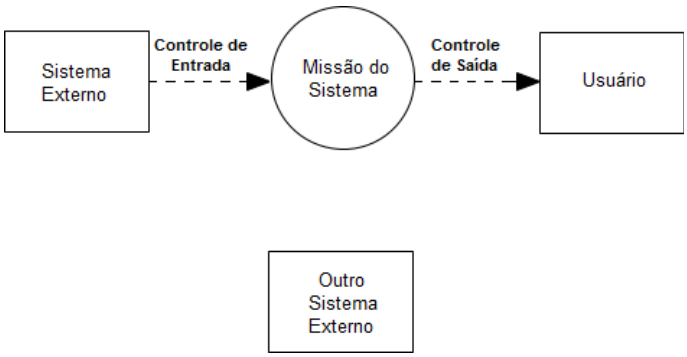
B.1.3 Modelo de Controle

Este modelo permite ao engenheiro de sistemas capturar as capacidades requeridas de comportamento que este sistema deve apresentar em termos dos seus vários modos de operação, determinando quais capacidades de processamento estão ativas nos diferentes modos de operação do sistema.

Para criar este modelo, o engenheiro de sistemas tem à sua disposição os diagramas de fluxo de controle, cuja sigla é CFD, e as especificações de controle, cuja sigla é CSPEC. Esta última, permite a especificação das ativações e desativações, em determinadas condições, das diferentes funções identificadas no modelo de processo. Já o primeiro, permite o roteamento de fluxos de controle de e para as CSPECs (assim como um DFD permite o roteamento de fluxos de dados de e para as PSPECs).

Para cada DFD criado no modelo de processo, deverá ser criado um correspondente CFD no modelo de controle que controlará as ativações e desativações das suas funções sob determinadas condições. Existirão, portanto, um diagrama de contexto de controle, exemplificado na Figura B.5 abaixo e cuja sigla é CCD, correspondendo ao DCD da Figura B.3, e CFDs nivelados para corresponder com os DFDs nivelados, todos eles criados no modelo de processo.

Figura B.5 – Diagrama de Contexto de Controle (CCD)

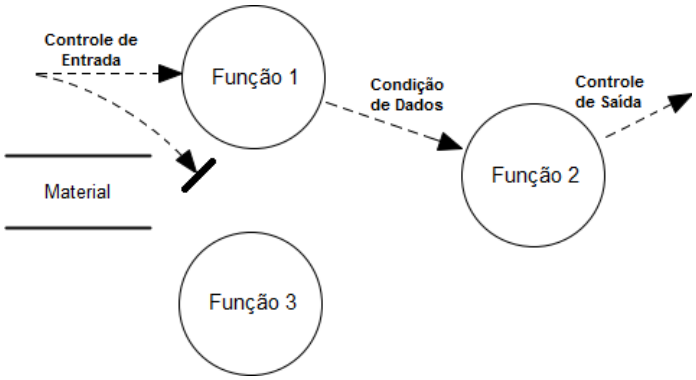


Fonte: Produção do Autor.

Na figura acima, tem-se um exemplo de um CCD para o DCD apresentado na Figura B.3. Novamente, os retângulos representam os terminadores, ou seja, as entidades externas ao sistema e o círculo no centro da figura representa a função precípua do sistema. As linhas tracejadas direcionadas representam os fluxos de controle trocados neste contexto. Mesmo que um terminador não seja responsável por trocas de fluxos de controle, ele é mantido na figura para compatibilidade com o correspondente DCD.

Depois de determinar o contexto de controle, o engenheiro de sistemas passa para os níveis de abstração inferiores da decomposição funcional, onde ele basicamente modela a ativação e desativação das funções identificadas no modelo de processo. Na Figura B.6 abaixo, exemplifica-se um CFD para o DFD anteriormente apresentado na Figura B.4.

Figura B.6 – Diagrama de Fluxo de Controle (CFD)



Fonte: Produção do Autor.

Na figura acima, representa-se a modelagem de controle feita para o nível de abstração da decomposição funcional anteriormente apresentado na Figura B.4. São mantidos as mesmas funções e armazenadores para facilitar o relacionamento entre os dois modelos (i.e. modelo de processo e modelo de controle), mas em vez de fluxos de dados têm-se fluxos de controle, aqui representados pelas linhas tracejadas direcionadas, e que devem obedecer à mesma regra de balanceamentos utilizada no modelo de processo. Pela regra, então, os fluxos de controle de entrada e saída da função identificada no nível de abstração superior, e que aqui está sendo decomposta, devem estar presentes e serem tratados no diagrama para garantir que nada é

perdido ao longo dos diversos níveis de desdobramentos funcionais existentes. A barra presente no diagrama próximo à função “Função 1” representa a especificação de controle (CSPEC) relacionada a este CFD. Pode-se utilizar quantas barras forem necessárias para evitar desordem no diagrama, entretanto todas elas referenciarão à única CSPEC relacionada ao CFD.

Quanto aos fluxos de controle, somente aqueles que chegam na barra representando a correspondente CSPEC deste nível é que efetivamente serão utilizados para se determinar a ativação e desativação das funções apresentadas no correspondente CFD. Fluxos de controle que chegam em uma dada função estão simplesmente sendo direcionados para o seu CFD onde poderão, eventualmente, controlar ativações e desativações das suas subfunções. Fluxos de controle que saem de uma dada função representam controles sendo gerados nos níveis de abstração inferiores da decomposição funcional. Existe, ainda, a possibilidade de fluxos de controle serem gerados internamente ao sistema através do teste de fluxos de dados contra uma condição específica. Estes fluxos são chamados de ‘Condição de Dados’, exemplificado na Figura B.6 entre a função “Função 1” e a função “Função 2”, e devem estar especificados na correspondente PSPEC, onde o teste condicional e o fluxo de controle são claramente identificados.

As especificações de controle (ou CSPECs) permitem capturar os requisitos de controle dos processamento do sistema. Elas transformam fluxos de controle em ativadores de funções, em novos fluxos de controle ou em novos fluxos de dados. Elas podem ser máquinas combinatórias ou sequenciais. O primeiro tipo não possui memória, ou seja, ele transforma entradas em saídas diretamente e independentemente do passado. O segundo tipo possui memória e transforma entradas em saídas dependendo do estado atual e, eventualmente, muda de estado. Máquinas combinatórias podem ser especificadas usando tabelas de decisão ou expressões condicionais, enquanto máquinas sequenciais podem ser especificadas através de

diagramas de estado (HAREL, 1987) ou equivalentes (e.g.: matrizes de transição de estado, etc).

Independentemente da metodologia utilizada para elaboração de uma CSPEC, ao final, uma tabela de ativação de funções é construída para especificar qual função está ativa ou não, em qual sequência as ativações ocorrem, caso necessário, e sob quais condições isso ocorre. O uso de textos na especificação de controle é possível, mas raramente utilizado.

B.1.4 Dicionário de Requisitos

O dicionário de requisitos é, basicamente, uma base de dados de todos os elementos usados nos modelos, cada um com sua descrição estrutural, ou seja, como o elemento é decomposto em subelementos, seu significado e outros detalhes. Este dicionário define os fluxos de dados, fluxos de controle, armazenadores, classes de entidade, relacionamentos, atributos e, também, componentes a partir dos quais estes elementos são formados. Elementos primitivos, ou seja, aqueles que não são decompostos adiante, têm suas propriedades definidas, tais como unidade, precisão, resolução, taxa de atualização, intervalo de valores possíveis, etc.

O método HHP define uma notação específica para permitir a representação deste dicionário de requisitos, que utiliza operadores pré-estabelecidos. Este método determina, também, que os requisitos de desempenho do sistema estejam definidos neste dicionário. Estes requisitos são apresentados em forma tabular, onde para cada fluxo de entrada recebido pelo sistema, os fluxos de saída produzidos a partir deste fluxo de entrada são apresentados e acompanhados do desempenho requerido para tal transformação. O método HHP não determina que sejam levantados requisitos de desempenho para as transformações ocorridas internamente ao sistema.

B.2 Modelo de Requisitos Melhorados

Quando o modelo de requisitos essenciais foi construído, a tecnologia utilizada no sistema e no ambiente ao seu redor foi desconsiderada para garantir requisitos livres de características implementacionais. Fluxos

entrando e saindo do sistema eram, na verdade, fluxos lógicos que não correspondem aos fluxos físicos (i.e. reais) que serão efetivamente tratados pelo sistema. Na sua essência, o sistema não falha e não necessita de nada especial para realizar sua missão.

Ao construir o modelo de requisitos melhorados, o engenheiro de sistemas passa a levar em consideração, se necessário, o mundo real e a tecnologia disponível para permitir o acoplamento entre este sistema e o ambiente ao seu redor. Melhorar o modelo de requisitos significa, portanto, transformar os fluxos lógicos em fluxos físicos. No mundo real, sistemas falham e funções de mitigação para redundância, manutenção e detecção, correção e proteção de falhas, deverão ser incorporadas.

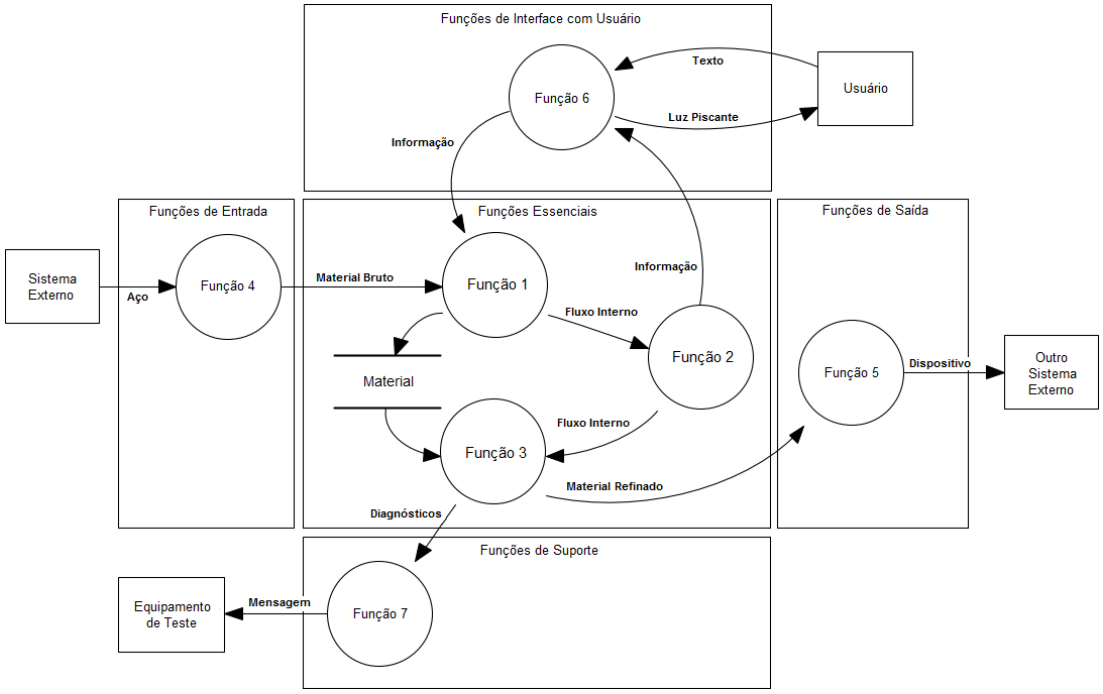
Funções de entrada são aquelas criadas para transformar os fluxos físicos recebidos de entidades externas nos fluxos lógicos esperados pelas funções essenciais. Funções de saída são aquelas ciadas para transformar os fluxos lógicos gerados pelas funções essenciais nos fluxos físicos esperados pelas entidades externas. Funções de interface com o usuário são aquelas criadas para transformar fluxos lógicos produzidos e esperados por funções essenciais nos fluxos físicos recebidos e produzidos pelos usuários do sistema. Finalmente, funções de suporte são aquelas criadas para possibilitar o correto processamento das funções essenciais no ambiente onde o sistema está inserido (e.g.: redundância, manutenção, detecção de erros, etc).

Conforme incorporam-se novas funções não-essenciais ao modelo, determinam-se novos fluxos de dados e de controle. Isso significa que o engenheiro de sistemas deverá revisitar e melhorar tanto o modelo de processo quanto o modelo de controle. Especialmente com a incorporação de funções de suporte, novas entidades externas (i.e. terminadores) podem ser identificadas, tais como dispositivos de testes ou diagnósticos. A decomposição funcional feita até agora para as funções essenciais deverá ser igualmente realizada para as novas funções não-essenciais identificadas. Exatamente como feito para o modelo de requisitos, cada nível de abstração

da decomposição das funções não-essenciais deve apresentar uma CSPEC, determinando quando e em qual sequência essas funções são ativadas e desativadas. Novamente, para cada função não-essencial terminal, uma PSPEC deve ser criada para especificar a transformação de suas entradas nas suas saídas.

Na Figura B.7 abaixo, exemplifica-se um DFD feito para o modelo de requisitos melhorados, levando-se em conta o DFD apresentado anteriormente na Figura B.4. Nela, é possível notar que o DFD da Figura B.4 apresentado anteriormente se encontra bem ao centro da figura, dentro da caixa nomeada “Funções Essenciais”, onde têm-se as mesmas funções essenciais e fluxos lógicos do DFD original, agora rodeadas por novas funções não-essenciais, trocando fluxos físicos com as entidades externas e usuários do sistema.

Figura B.7 – Diagrama de Fluxo de Dados Melhorado



Fonte: Produção do Autor.

A entidade externa “Sistema Externo”, que antes interagia com a função essencial “Função 1”, passa agora a interagir com a função não essencial de entrada “Função 4”, responsável por transformar o fluxo físico “Aço” no fluxo lógico “Material Bruto”.

O mesmo acontece com a entidade externa “Outro Sistema Externo”, que passou a interagir com a função não-essencial de saída “Função 5”, responsável por transformar o fluxo lógico “Material Refinado” no fluxo físico “Dispositivo”.

Já o usuário do sistema “Usuário” passa agora a interagir com a função não-essencial de interface com usuário “Função 6”, que transforma os fluxos lógicos das funções essenciais nos fluxos físicos efetivamente trocados com este.

Uma hipotética necessidade de envio de mensagens para um equipamento de teste tornou necessária a incorporação da função não-essencial de suporte “Função 7”, que transforma um novo fluxo lógico “Diagnóstico” em um novo fluxo físico “Mensagem”, enviado à entidade externa “Equipamento de Teste”.

Toda essa parte não estava presente no modelo de requisitos essenciais, uma vez que não faz parte da essência do sistema diagnosticar falhas, por exemplo.

B.3 Modelo de Arquitetura

Este modelo permite ao engenheiro de sistemas mostrar a realidade física do sistema sendo modelado: componentes, interfaces, funcionalidades, etc. A ideia central deste modelo é permitir o agrupamento dos requisitos em blocos passíveis de serem construídos.

Este modelo é constituído de vários elementos gráficos suportados por especificações. Ele pode ser dividido em dois submodelos: modelo de contexto de arquitetura e modelo de módulos de arquitetura. Novamente, não existe prescrição estabelecida para a sequência de elaboração destes modelos, sendo eles criados de maneira interativa.

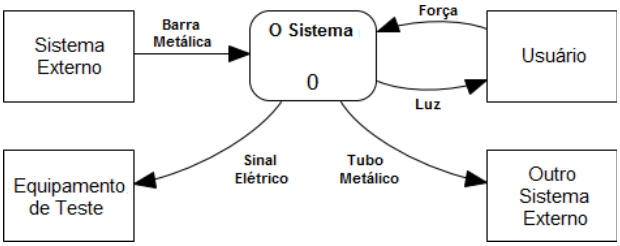
B.3.1 Modelo de Contexto de Arquitetura

Este modelo permite ao engenheiro de sistemas definir o ambiente físico do sistema utilizando três tipos distintos de diagramas: o diagrama de contexto

de fluxo de arquitetura, cuja sigla é AFCD, o diagrama de contexto de mensagem de arquitetura, cuja sigla é AMCD, e, finalmente, o diagrama de contexto de interconexão de arquitetura, cuja sigla é AICD.

O AFCD, exemplificado na Figura B.8 abaixo, permite mostrar o sistema, as entidades externas e os fluxos de arquitetura trocados entre eles.

Figura B.8 – Diagrama de Contexto de Fluxo de Arquitetura (AFCD)



Fonte: Produção do Autor.

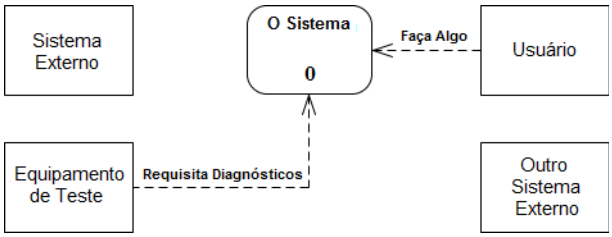
Na figura acima, o retângulo com cantos arredondados representa o sistema e o índice ‘0’ indica que este é o elemento de arquitetura do nível mais alto de abstração. Os retângulos retos representam as entidades externas e usuários do sistema. As linhas cheias direcionadas representam os fluxos de arquitetura.

No modelo de arquitetura, os retângulos com cantos arredondados representam os módulos de arquitetura e, especificamente nos diagramas de contexto, por se tratar do nível de abstração mais alto, existirá somente um módulo de arquitetura, com índice ‘0’ atribuído, representando o sistema propriamente dito.

O AMCD, exemplificado na Figura B.9 abaixo, permite representar as mensagens de arquitetura trocadas entre o sistema e as entidades externas. A diferença entre fluxos e mensagens está, basicamente, na ênfase por eles colocada. Um fluxo representa material, energia ou informação sendo trocado entre dois elementos de arquitetura, sem ênfase em qual deles realmente causou o fluxo propriamente dito. Consiste, basicamente, na mesma abordagem usada em um DFD. Mensagens, por outro lado, estabelecem um padrão ativo/passivo, onde um elemento de arquitetura (o ativo) envia a mensagem para outro elemento de arquitetura (o passivo).

Material, energia ou informação pode, eventualmente, ser transmitido na mesma direção da mensagem (um elemento postando um fluxo para o outro), na direção contrária da mensagem (um elemento requisitando um fluxo para o outro) ou em ambas as direções (um elemento postando e requisitando fluxos do outro).

Figura B.9 – Diagrama de Contexto de Mensagem de Arquitetura (AMCD)

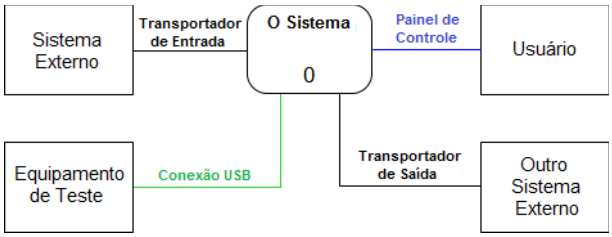


Fonte: Produção do Autor.

Na figura acima, o retângulo com cantos arredondados representa, novamente, o sistema. Os retângulos normais representam as entidades externas e usuários do sistema. As linhas tracejadas direcionadas representam as mensagens de arquitetura que, neste caso, não carregam consigo fluxos de arquitetura.

O AICD, exemplificado na Figura B.10 abaixo, permite representar o sistema, as entidades externas e as interconexões físicas entre eles. Uma interconexão é o meio físico ou canal pelo qual os fluxos de material, energia, informação ou mensagens são trocados entre elementos de arquitetura.

Figura B.10 – Diagrama de Contexto de Interconexão de Arquitetura (AICD)



Fonte: Produção do Autor.

Na figura acima, o retângulo com cantos arredondados representa, novamente, o sistema. Os retângulos com cantos retos representam as entidades externas e usuários do sistema. As linhas não direcionadas representam as interconexões entre os respectivos elementos de arquitetura,

que para melhorar a leitura do diagrama podem ser representadas em diferentes cores e estilos.

B.3.2 Modelo de Módulos de Arquitetura

Este modelo permite ao engenheiro de sistemas capturar a estrutura do sistema em termos dos módulos de arquitetura que o compõem, de suas especificações e de seus relacionamentos, estes podendo ser de três tipos distintos: comunicação, meio de comunicação e herança.

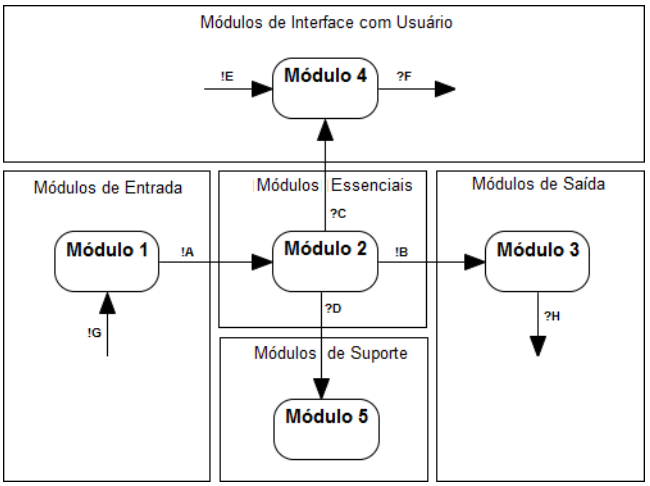
Para compor este modelo, o engenheiro de sistemas tem ao seu dispor quatro diagramas de módulos de arquitetura: o diagrama de fluxo de arquitetura, cuja sigla é AFD, o diagrama de mensagem de arquitetura, cuja sigla é AMD, o diagrama de interconexão de arquitetura, cuja sigla é AID e, finalmente, o diagrama de herança de módulo, cuja sigla é MID.

Estes diagramas podem ser feitos em vários níveis de decomposição de arquitetura, conforme o sistema é desdobrado em módulos menores. Não existe regra para quando se deve parar esta decomposição física. A ideia básica é que, ao final desta atividade, os módulos de arquitetura de mais baixo nível possam ser enviados para a fase de aquisição, ou seja, serem comprados de terceiros como componentes prontos para uso ou serem desenvolvidos, tanto internamente quanto externamente à organização.

Os diagramas AFD, AMD e AID são similares às suas contrapartes no nível do contexto do sistema, apresentados na seção anterior. A diferença aqui é que eles tratarão de módulos de arquitetura internos ao sistema e, portanto, as entidades externas não mais aparecerão. Igualmente aos diagramas no nível do contexto do sistema, o AFD permitirá mostrar os fluxos de arquitetura entre os diferentes módulos de arquitetura, o AMD permitirá representar as mensagens trocadas entre estes módulos e, finalmente, o AID permitirá especificar as interconexões existentes entre eles.

O AFD, exemplificado na Figura B.11 abaixo, permite mostrar os módulos de arquitetura no primeiro nível da decomposição física do sistema e os fluxos de arquitetura trocados entre eles.

Figura B.11 – Diagrama de Fluxo de Arquitetura (AFD)



Fonte: Produção do Autor.

Na figura acima, os retângulos com cantos arredondados representam módulos de arquitetura e as linhas cheias e direcionadas representam os fluxos de arquitetura trocados entre estes módulos. Fluxos que não possuem origem ou destino representam fluxos de arquitetura que estão vindo ou indo, respectivamente, para o nível logo acima da decomposição física do sistema.

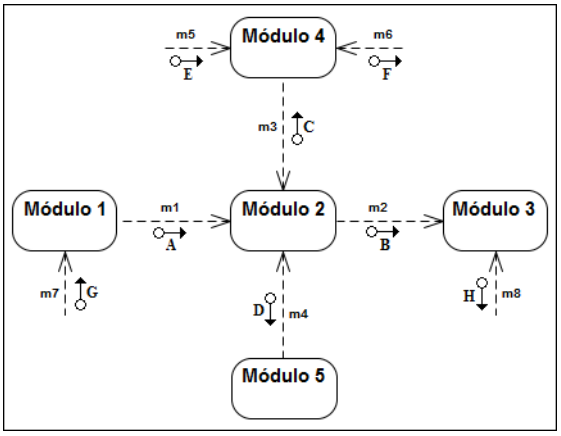
Nota-se, na figura acima, o uso do gabarito para arquitetura de sistema apresentado anteriormente na Figura B.1 para identificar que o módulo de arquitetura “Módulo 1” representa um módulo de entrada, o módulo de arquitetura “Módulo 2” representa um módulo essencial, o módulo de arquitetura “Módulo 3” representa um módulo de saída, o módulo de arquitetura “Módulo 4” representa um módulo de interface com o usuário e, finalmente, o módulo de arquitetura “Módulo 5” representa um módulo de suporte. Essa não é uma prática exigida pelo método HHP, mas pode ser empregada sempre que necessária.

Para fins didáticos, os fluxos aqui foram nomeados de “A” a “H” e os prefixos ‘?’ e ‘!’ presentes nestes fluxos de arquitetura representam uma notação especial utilizada por Hatley et al. (2000) para adicionar uma informação aos fluxos que só seria possível utilizando mensagens no lugar de fluxos (e, consequentemente, um AMD no lugar do AFD). O símbolo ‘?’ significa que o módulo de arquitetura no destino do fluxo está requisitando dados ao módulo

de arquitetura na origem do fluxo. Já o símbolo ‘!’ significa que o módulo de arquitetura na origem do fluxo está postando dados para o módulo de arquitetura no destino do fluxo.

O AMD, exemplificado na Figura B.12 abaixo, permite mostrar os módulos de arquitetura no primeiro nível da decomposição física do sistema e as mensagens de arquitetura trocadas entre eles.

Figura B.12 – Diagrama de Mensagem de Arquitetura (AMD)



Fonte: Produção do Autor.

Na figura acima, os retângulos com cantos arredondados representam, novamente, módulos de arquitetura e as linhas tracejadas e direcionadas representam as mensagens de arquitetura trocadas entre estes módulos. Mensagens que não possuem origem ou destino representam mensagens de arquitetura que estão vindo ou indo, respectivamente, para o nível logo acima da decomposição física do sistema.

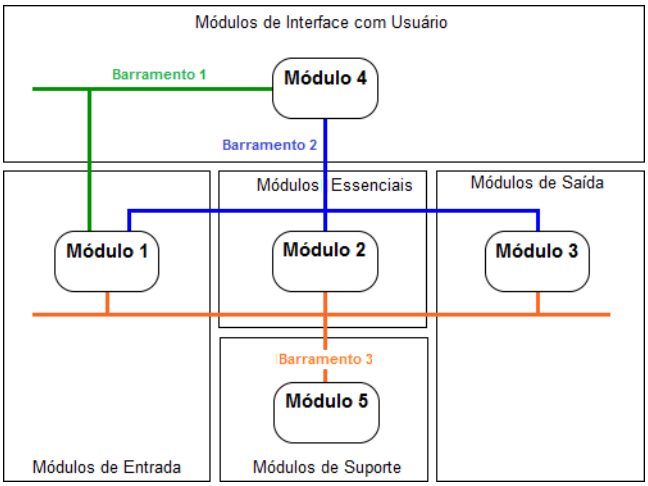
Para fins didáticos, as mensagens aqui foram nomeadas de “m1” a “m8”. Fluxos de dados de arquitetura trafegando junto com as mensagens são representados pelo símbolo ‘o→’.

Nota-se que fluxos de dados no mesmo sentido da mensagem representam dados sendo postado pelo módulo de arquitetura na origem da mensagem para o módulo de arquitetura no destino da mensagem. Fluxos de dados no sentido contrário da mensagem representam dados sendo requisitados pelo módulo de arquitetura no destino da mensagem ao módulo de arquitetura na origem da mensagem.

Para evitar sobrecarregar o AMD, dificultando, assim, sua leitura, pode-se dar detalhes sobre as mensagens em especificações de mensagens, separadas dos diagramas. Estas especificações definiriam, entre outras características importantes, os parâmetros das mensagens e a composição das mensagens.

O AID, exemplificado na Figura B.13 abaixo, permite mostrar os módulos de arquitetura no primeiro nível da decomposição física do sistema e as interconexões existentes entre eles.

Figura B.13 – Diagrama de Interconexão de Arquitetura (AID)



Fonte: Produção do Autor.

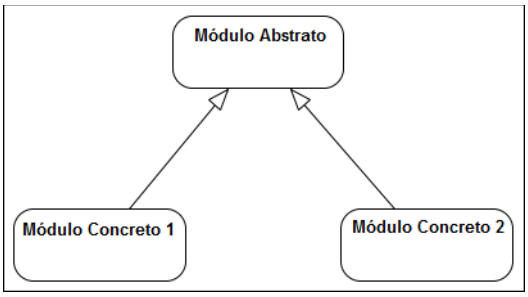
Na figura acima, os retângulos com cantos arredondados representam, novamente, módulos de arquitetura e as linhas coloridas não direcionadas representam as interconexões entre estes módulos, que podem assumir diferentes cores e estilos para facilitar a leitura do diagrama. No exemplo acima, imaginaram-se três barramentos que materializam o meio físico que torna possível a troca de fluxos e mensagens entre os diferentes módulos de arquitetura identificados.

Toda interconexão definida (tanto no AICD quanto nos AIDs) deve ter uma especificação de interconexão de arquitetura, cuja sigla é AIS. Esta especificação contém as definições físicas da interconexão, descrevendo as características do seu meio de transmissão de fluxos e mensagens.

Por fim, o MID, exemplificado na Figura B.14 abaixo, permite detalhar os relacionamentos de herança existentes entre os diferentes módulos de

arquitetura. A ideia principal por trás deste tipo de relacionamento é agrupar características e funcionalidades comuns presentes em diferentes módulos de arquitetura em módulos de arquitetura abstratos.

Figura B.14 – Diagrama de Herança de Módulo (MID)



Fonte: Produção do Autor.

Na figura acima, os retângulos com cantos arredondados representam, novamente, módulos de arquitetura e as linhas cheias com setas nas extremidades representam os relacionamentos de herança, onde um módulo de arquitetura na extremidade da seta representa uma abstração do módulo de arquitetura na extremidade oposta à da seta. No exemplo da figura acima, o módulo de arquitetura “Módulo Abstrato” agrupa características e funcionalidades comuns aos módulos de arquitetura “Módulo Concreto 1” e “Módulo Concreto 2”.

Todo módulo de arquitetura identificado deve ter uma especificação de módulo de arquitetura, cuja sigla é AMS. Esta especificação contém as definições físicas do correspondente módulo de arquitetura, com referências cruzadas às funções do modelo de requisitos alocadas pelo módulo. Ela contém, também, justificativas para as escolhas feitas, descrições das interfaces do módulo de arquitetura e restrições com as quais o módulo deva estar em conformidade (e.g.: confiabilidade, manutenibilidade, segurança, projeto físico, custo, prazo, etc).

B.3.3 Dicionário de Arquitetura

Um outro importante artefato que deve ser produzido pelo engenheiro de sistemas é o dicionário de arquitetura. Similar ao dicionário de requisitos, este dicionário é o repositório único das definições de fluxos encontrados no

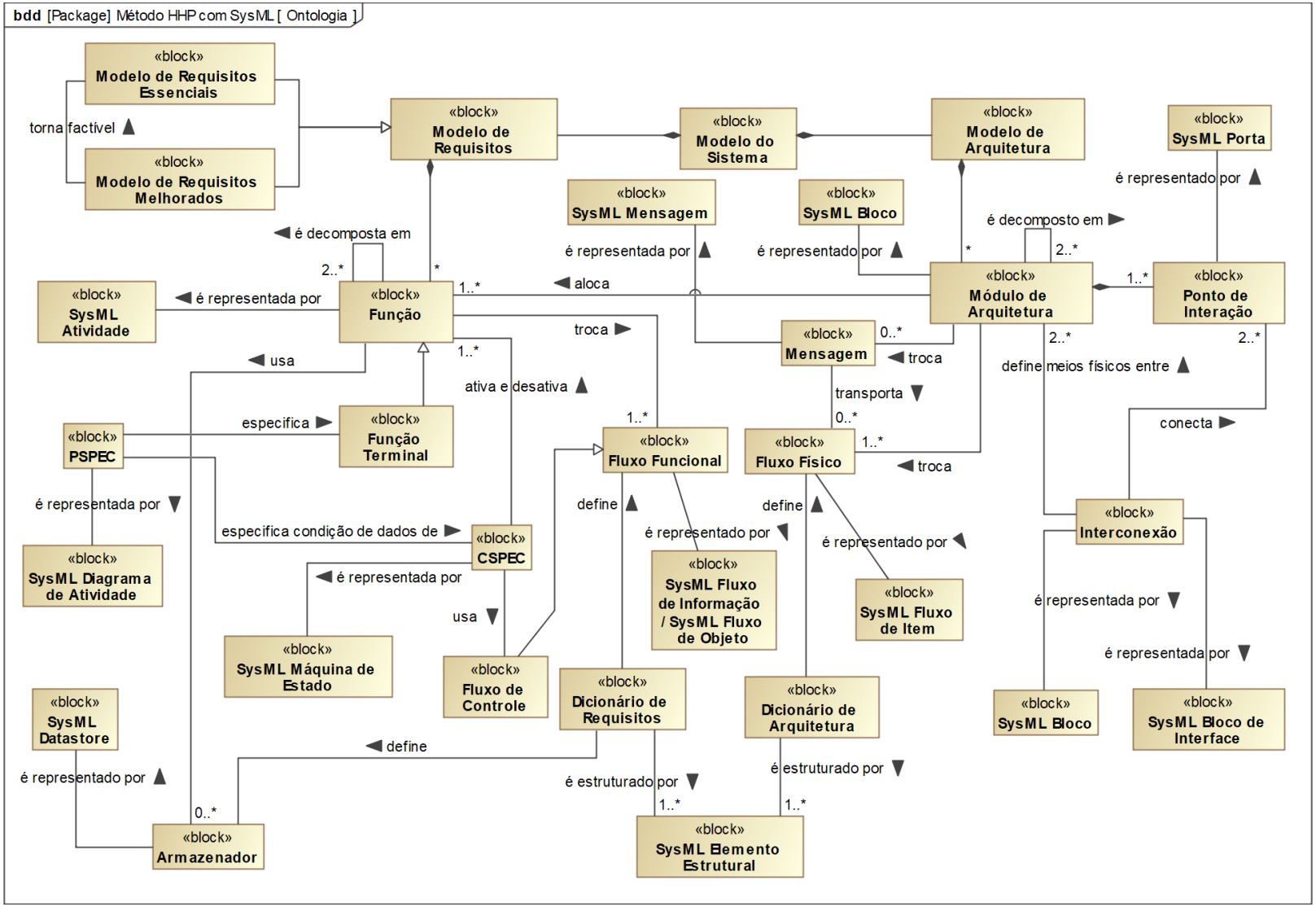
modelo de arquitetura. Para cada fluxo, este dicionário contém, ao menos, o nome, a estrutura (componentes que compõem o fluxo), descrição física, tipo (dado ou controle), módulo de arquitetura de origem, módulo de arquitetura de destino e meio físico utilizado para sua transmissão. Este conjunto de informações pode ser aumentado sempre que necessário.

APÊNDICE C – ONTOLOGIA DA METODOLOGIA PROPOSTA

Na Figura C.1 abaixo, apresenta-se a ontologia da metodologia proposta no Capítulo 4. Conforme Genesereth et al. (1987), nesta ontologia mostram-se os principais termos do método HHP de Hatley et al. (2000) e como estes termos estão relacionados entre si. Mais além, nesta ontologia define-se quais elementos da SysML são utilizados para representar estes termos e relacionamentos. Na Figura C.1 abaixo, tem-se um Diagrama de Definição de Bloco da SysML sendo usado para apresentar a ontologia da metodologia proposta. O quadro ao redor dos elementos é padrão da SysML e delimita o diagrama. Ele contém um cabeçalho na parte superior à esquerda da figura, cujo conteúdo também está definido na especificação da SysML como sendo: “<TipoDiagrama> [TipoElemento] <NomeElemento> [NomeDiagrama]”, onde o primeiro campo contém um mnemônico para o tipo do diagrama, cujos valores possíveis estão definidos na especificação da SysML, o segundo campo identifica o tipo do elemento do modelo que encapsula o diagrama, cujos valores possíveis estão igualmente definidos na especificação da SysML, o terceiro campo contém o nome do elemento do modelo que encapsula o diagrama e, finalmente, o quarto campo contém o nome do diagrama em si. Somente os dois últimos campos possuem valores livres e que são definidos pelo engenheiro de sistemas no momento da criação do diagrama.

No caso da Figura C.1 abaixo, o cabeçalho do diagrama possui “bdd” como tipo do diagrama (significando *Block Definition Diagram*), “*Package*” como tipo do elemento do modelo encapsulando o diagrama, pois o diagrama foi criado dentro de um pacote no modelo, “Método HHP com SysML” como nome do elemento do modelo encapsulando o diagrama, pois este foi o nome escolhido para o pacote dentro do qual o diagrama foi criado, e “Ontologia” como nome do diagrama, pois este foi o nome escolhido para o Diagrama de Definição de Bloco.

Figura C.1 – Ontologia do método HHP em SysML



Fonte: Produção do Autor.

Os retângulos na figura acima são os elementos Bloco da SysML. O texto “<<block>>” acima do nome de cada bloco representa o estereótipo definido na SysML que estende o elemento Classe da UML, criando, portanto, este elemento estrutural da linguagem. As linhas representam os relacionamentos entre Blocos.

Ainda na Figura C.1 acima, linhas com uma seta cheia em uma das extremidades representa um relacionamento de generalização, onde o Bloco na extremidade da seta é uma generalização do Bloco na extremidade oposta. Linhas com um losango preto em uma das extremidades representa um relacionamento de composição, onde o Bloco na extremidade do losango preto é composto pelo Bloco na extremidade oposta (comumente chamado de relacionamento “todo-parte”, onde o Bloco na extremidade do losango representa o todo e o Bloco na extremidade oposta representa uma parte do todo). Linhas sem nenhum adorno nas extremidades representam associações simples, onde o Bloco em uma das extremidades está relacionado de alguma forma ao Bloco na extremidade oposta. A semântica deste relacionamento é dada pelo sentido, nome da associação e multiplicidades da associação, indicando como o relacionamento deve ser lido por uma pessoa: “<multiplicidade na origem da associação> <Bloco na extremidade de origem da associação> <nome da associação> <multiplicidade no destino da associação> <Bloco na extremidade de destino da associação>”. Quando em uma dada associação a multiplicidade não é explicitada, assume-se o valor '1' (um).

Para representar a ontologia da metodologia HHP em SysML, Blocos foram utilizados para representar tanto os termos do método HHP, quanto os elementos da SysML que serão utilizadas para representar tais termos. Estes últimos estão representados pelos Blocos cujo nome contém o prefixo “SysML”. Com exceção do elemento *Datastore* da SysML, os demais elementos tiveram seus nomes traduzidos para o Português. Decidiu-se manter o termo ‘*datastore*’ em sua forma original, ou seja, no Inglês, por questões de não existir um termo traduzido em Português que possuísse a

mesma forma compacta de escrita. Relacionamentos da SysML foram utilizados para representar como cada termo ou elemento da SysML está relacionado com os demais elementos da ontologia.

No centro superior da Figura C.1, tem-se o modelo do sistema, que é composto pelo modelo de requisitos (mais à esquerda na figura) e pelo modelo de arquitetura (mais à direita na figura). O modelo de requisitos, por sua vez, pode ser de dois tipos especializados: modelo de requisitos essenciais e de requisitos melhorados, que torna factível o modelo de requisitos essenciais (extrema esquerda na figura).

Independentemente do tipo do modelo de requisitos, este modelo é composto por funções que são decompostas em subfunções, até que se chegue no nível de funções terminais, estas especificadas através de PSPECs. As funções são representadas pelo elemento Atividade da SysML, enquanto as PSPECs são representadas por Diagramas de Atividade da SysML.

Funções trocam fluxos funcionais (ou fluxos lógicos), representados ou pelo elemento Fluxo de Informação ou pelo elemento Fluxo de Objeto da SysML, e usam armazenadores, representados pelo elemento *Datastore* da SysML. Estes elementos estarão definidos no dicionário de requisitos, que por sua vez será estruturado usando elementos estruturais da SysML (e.g.: Blocos, etc).

Funções são ativadas ou desativadas por CSPECs, que podem ter condições de dados especificadas por PSPECs, e usam fluxos de controle, sendo estes um tipo especializado de fluxo funcional. Estas CSPECs são representadas pelo elemento Máquina de Estado da SysML.

Voltando ao canto superior direito da Figura C.1, o modelo de arquitetura é composto por módulos de arquitetura, que podem ser decompostos em submódulos, todos estes representados pelo elemento Bloco da SysML.

Módulos de arquitetura possuem pontos de interação representados pelo elemento Porta da SysML. Assim como sugerido por Delligatti (2014), não

serão utilizados, para estes elementos, os estereótipos de refinamento <<*FullPort*>> e <<*ProxyPort*>>, existentes a partir da versão 1.3 da SysML.

Estes pontos de interação são relacionados por interconexões de arquitetura, especificando os meios físicos entre os respectivos módulos de arquitetura. Estas interconexões são representadas pelo elemento Bloco ou pelo elemento Bloco de Interface, ambos da SysML. Estes elementos serão utilizados para tipificar as respectivas Portas envolvidas nas correspondentes interconexões.

Através destas interconexões de arquitetura, os módulos de arquitetura trocam fluxos físicos (ou fluxos de arquitetura) e mensagens, estas podendo também transportar fluxos de arquitetura. Uma mensagem é representada pelo elemento Mensagem da SysML, enquanto um fluxo físico é representado pelo elemento Fluxo de Item da SysML e é definido no dicionário de arquitetura, este estruturado usando elementos estruturais da SysML (e.g.: Blocos, Tipos de Valor, etc).

Finalmente, interligando o modelo de arquitetura ao modelo de requisitos, têm-se módulos de arquitetura alocando funções.

APÊNDICE D – MODELO SYSML DO AESP14

Abaixo, apresenta-se o modelo descritivo do AESP14, construído utilizando a metodologia elaborada no Capítulo 4, Seção 4.6. A divisão em subseções se baseia na mesma divisão utilizada naquela seção, com a exceção de que a Seção D.1, contento o modelo de requisitos, agrupará tanto o modelo de requisitos essenciais como o modelo de requisitos melhorados. Decidiu-se por juntar os dois modelos em uma única seção, pois o modelo de requisitos melhorado nada mais é do que uma revisão do modelo de requisitos feito até aquele momento. Na Seção D.2, apresenta-se o modelo de arquitetura. Finalmente, não Seção D.3, faz-se o relacionamento dos dois modelos criados anteriormente.

É importante salientar que este modelo foi desenvolvido tendo como base o trabalho de engenharia de sistemas materializado em AESP14 (2014).

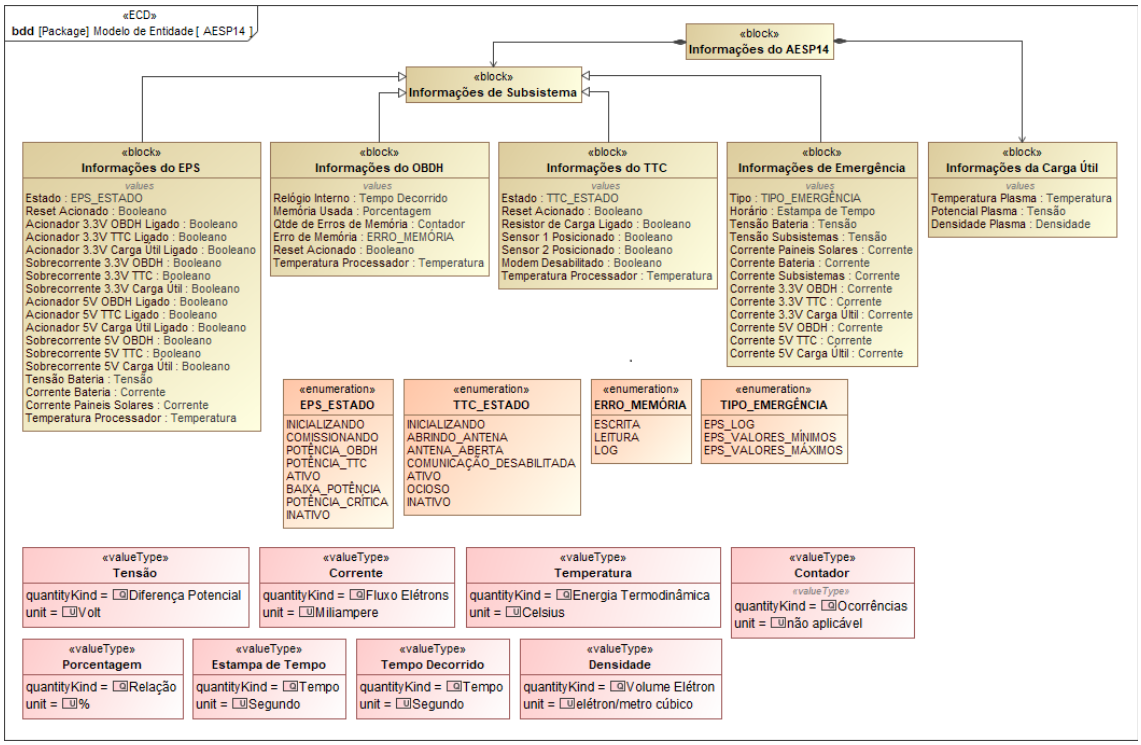
D.1 Modelo de Requisitos

Esta seção apresenta a descrição funcional do AESP14, modelada utilizando a metodologia desenvolvida no Capítulo 4, Seções 4.6.2 e 4.6.3. Ela está organizada da seguinte maneira. Na Subseção D.1.1, apresenta-se o modelo de entidade do AESP14. O modelo de processos para o sistema está descrito na Subseção D.1.2. A Subseção D.1.3 trás o modelo de controle feito para o AESP14. Finalmente, na Subseção D.1.4, apresenta-se o dicionário de requisitos que suplementa os modelos feitos até então.

D.1.1 Modelo de Entidade

Na Figura D.1 abaixo, mostra-se um diagrama BDD da SysML com o modelo de entidade desenvolvido para o AESP14. Lembrando o propósito do modelo de entidade, têm-se os dados (material, energia ou informação) manipulados pelo sistema AESP14, do ponto de vista de seu domínio de negócio, onde o nível de abstração permite distanciar-se das possíveis tecnologias e implementações que serão utilizadas no desenvolvimento deste sistema.

Figura D.1 – Modelo de Entidade para o AESP14.



Fonte: Produção do Autor.

Na parte mais acima do diagrama da figura acima, têm-se Blocos da SysML representando as informações que serão manipuladas pelo sistema AESP14. Criaram-se Blocos para representar as informações que cada subsistema fornecerá, informações de emergência e informações de carga útil. Estas informações estão baseadas em AESP-14 Telemetry (2015). Notam-se dois grandes grupos de informações: “Informações de Subsistema” e “Informações da Carga Útil”. O primeiro é, ainda, subdividido em quatro subtipos distintos: “Informações do EPS”, “Informações do OBDH”, “Informações do TTC” e “Informações de Emergência”. Os três primeiros subtipos referem-se a informações dos subsistemas do sistema AESP14 e o quarto subtipo, as informações de emergência do sistema AESP14. O segundo grupo de informações refere-se aos dados de carga útil.

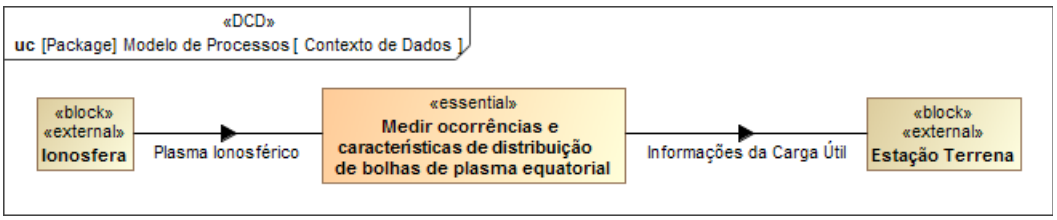
Propriedades de Valor da SysML, criadas para os Blocos acima descritos, compõem o conjunto de dados que cada grupo de informações carrega. Tais elementos devem ser devidamente tipificados com elementos Tipo de Valor da SysML para dar consistência ao modelo de entidade.

Mais ao centro do diagrama da Figura D.1 acima, com nomes em letras maiúsculas, estão Enumerações da SysML (extensões do elemento Tipo de Valor da SysML), representando tipos de dados, cujos valores estão delimitados por um conjunto finito de possibilidades e que não representam uma quantidade mensurável através de uma unidade de medida. Logo abaixo, encontram-se alguns elementos Tipo de Valor da SysML, representando tipos de dados mensuráveis do domínio de negócio do sistema AESP14.

D.1.2 Modelo de Processos

Começando pelo diagrama de contexto funcional, na Figura D.2 abaixo, mostra-se um Diagrama de Caso de Uso da SysML representando tal contexto para o sistema AESP14.

Figura D.2 – Contexto Funcional para o AESP14.

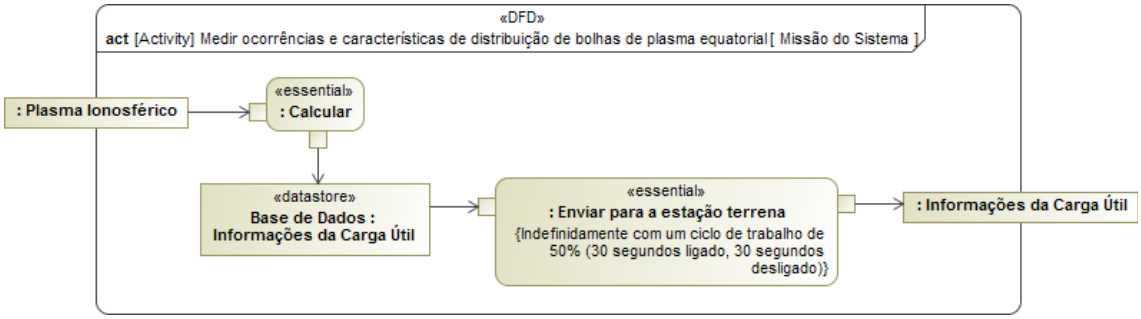


Fonte: Produção do Autor.

Analisando o diagrama da figura acima, notam-se duas entidades externas localizadas nas extremidades do diagrama: a ionosfera e a estação terrena. Mais ainda, tem-se ao centro da figura a Atividade da SysML com o estereótipo <<*essential*>> representando a missão do sistema "Medir ocorrências e características de distribuição de bolhas de plasma equatorial." Neste diagrama, mostra-se a parte essencial do modelo de processo, onde a ionosfera fornece o plasma ionosférico para o sistema (aqui representado pela sua função precípua) e este, por sua vez, envia as informações do plasma ionosférico para a estação terrena. Nota-se, ainda, o uso do tipo 'Informações da Carga Útil' no fluxo para a estação terrena, tipo este definido no modelo de entidade da Seção D.1.1. Este tipo de consistência garante a qualidade dos requisitos que serão gerados a partir deste modelo.

Depois de se estabelecer o contexto do sistema, o próximo passo consiste em desdobrar interativamente a missão do sistema para se descobrir suas diferentes funções essenciais terminais. Na Figura D.3 abaixo, mostra-se um Diagrama de Atividade da SysML para a Atividade representando a missão do sistema AESP14, vista no centro da Figura D.2 acima, com o objetivo de desdobrá-la em funções essenciais terminais.

Figura D.3 – Funções Essenciais do AESP14.



Fonte: Produção do Autor.

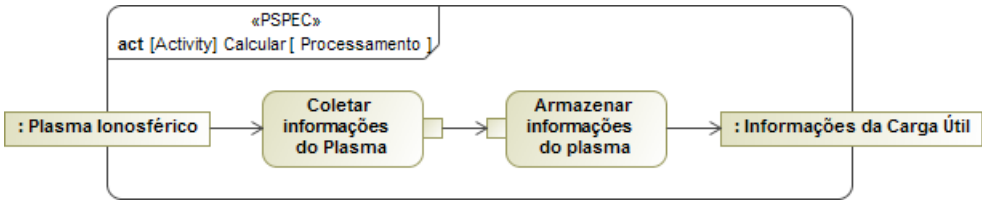
No caso do diagrama da figura acima, notam-se nas bordas do diagrama os parâmetros da missão do sistema devidamente tipificados de acordo com os fluxos identificados no contexto funcional do sistema apresentado na Figura D.2.

Pelo diagrama da figura acima, a missão do sistema foi desdobrada em duas funções essenciais terminais: Calcular as informações da carga útil usando o plasma ionosférico e enviar para a estação terrena as informações de carga útil. Estas duas funções trocam os dados através de um armazenador chamado “Banco de Dados”, que armazena dados do tipo ‘Informações da Carga Útil’, tipo este definido no modelo de entidade da Seção D.1.1.

Nota-se, ainda na Figura D.3 acima, que a função “Enviar para a estação terrena” está relacionada a um elemento Restrição da SysML. A expressão deste elemento aparece entre chaves logo abaixo seu nome.

Agora, deve-se desenvolver as especificações das funções essenciais terminais identificadas para o sistema AESP14. Na Figura D.4 abaixo, mostra-se a PSPEC da primeira função essencial terminal mencionada acima, através do Diagrama de Atividade da correspondente Atividade.

Figura D.4 – Especificação da Primeira Função Essencial do AESP14.

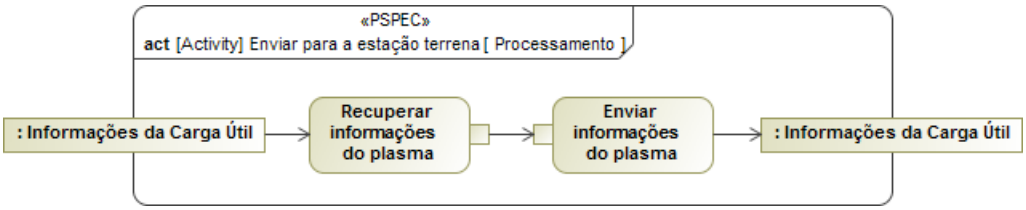


Fonte: Produção do Autor.

Pelo diagrama da figura acima, tem-se que a função essencial terminal do sistema AESP14 que calcula as informações da carga útil usando o plasma ionosférico consiste em coletar as informações do plasma e armazená-las para envio posterior. Nota-se, ainda, a tipificação consistente dos parâmetros da função essencial terminal em questão, o que garantirá a qualidade do correspondente requisito funcional, quando gerado a partir deste modelo de requisitos.

Já na Figura D.5 abaixo, mostra-se a especificação da segunda função essencial terminal identificada na Figura D.3 acima.

Figura D.5 – Especificação da Segunda Função Essencial do AESP14.



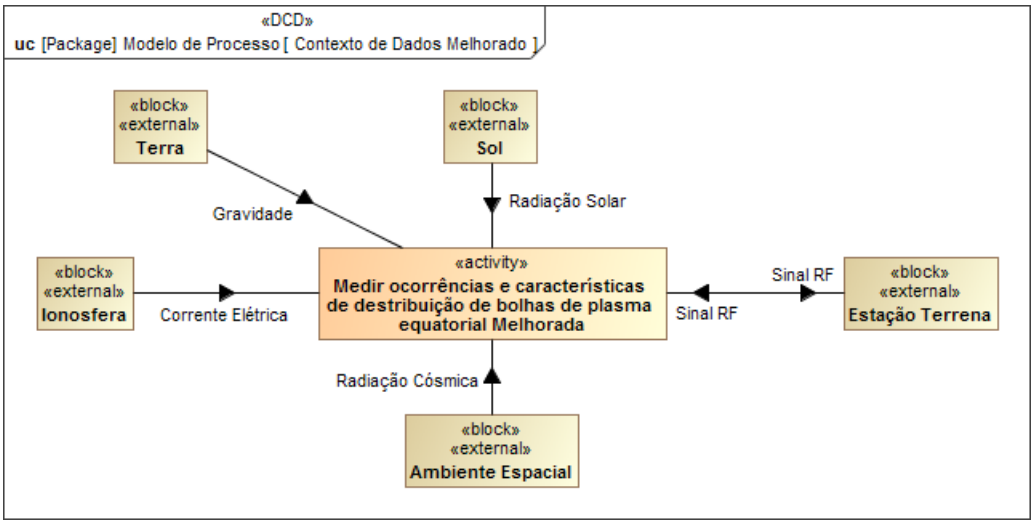
Fonte: Produção do Autor.

Já pelo diagrama da figura acima, tem-se a função essencial terminal do sistema AESP14 que envia para a estação terrena as informações da carga útil, consistindo em recuperar as informações do plasma, anteriormente armazenadas, e, efetivamente, enviá-las para a estação terrena. Novamente, a tipificação consistente dos parâmetros desta função essencial terminal garante a qualidade do correspondente requisito funcional, quando gerado a partir deste modelo de requisitos.

Revisitando o diagrama de contexto funcional da Figura D.2 e acrescentando entidades externas e fluxos de dados que aparecem devido à realidade em que o sistema se encontra, cria-se o contexto funcional melhorado. Na Figura

D.6 abaixo, mostra-se um Diagrama de Caso de Uso da SysML representando o contexto funcional melhorado do sistema AESP14.

Figura D.6 – Contexto Funcional Melhorado do AESP14.

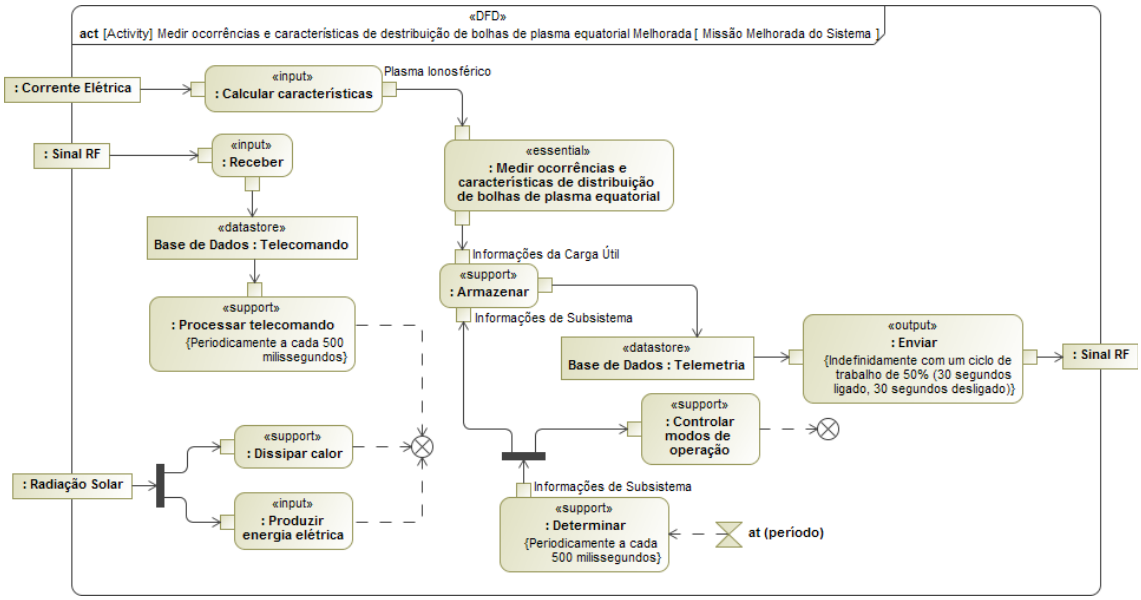


Fonte: Produção do Autor.

Saindo do domínio do ideal e entrando no domínio da realidade, observa-se que a ionosfera envia, na verdade, uma corrente elétrica, que é medida para se extrair as características do plasma. Sinais de radiofrequência são trocados com a estação terrena para enviar e receber informações. A Terra, com seu campo gravitacional, influencia no decaimento da órbita do nanossatélite, podendo comprometer sua missão. O Sol fornece a energia necessária para o funcionamento do dispositivo através de sua radiação solar. O ambiente espacial, no qual o AESP14 está inserido, emite radiações cósmicas, que podem comprometer o correto funcionamento de seus circuitos eletrônicos.

Desta forma, depois de revisitar e melhorar o contexto funcional do sistema, trazendo-o para a realidade, deve-se igualmente revisitar e melhorar o desdobramento funcional feito para a missão do sistema, acrescentando as funções auxiliares conforme apresentado no Capítulo 4, Seção 4.6, Subseção 4.6.3. Na Figura D.7 abaixo, mostra-se um Diagrama de Atividade da SysML representando a missão melhorada do sistema, onde funções terminais auxiliares foram adicionadas.

Figura D.7 – Funções Adicionais do AESP14.



Fonte: Produção do Autor.

No diagrama da figura acima, observa-se que foi acrescentada uma função de entrada para calcular as características do plasma ionosférico, usando a corrente elétrica recebida para fornecer a entrada necessária e esperada pela correspondente função essencial do sistema. Outra função de entrada recebe os telecomandos, usando o sinal RF que chega, para, então, serem processados por uma função de suporte, função esta relacionada a uma restrição de desempenho. Uma terceira função de entrada foi adicionada para produzir energia elétrica a partir da radiação solar recebida. Foram adicionadas, também, funções de suporte para dissipar calor, determinar informações dos subsistemas de forma periódica e armazená-las junto às informações da carga útil, fornecidas pela correspondente função essencial do sistema como telemetrias a serem enviadas para a estação terrena. Esta transmissão é feita por uma função de saída dedicada, que envia um sinal RF, usando as telemetrias anteriormente armazenadas, em ciclos de 30 segundos. Por último, uma função de suporte foi acrescentada para controlar os modos de operação do AESP14.

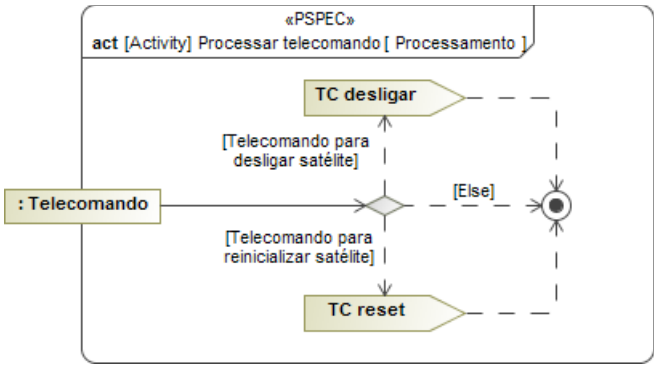
Alguns dos fluxos identificados na Figura D.6 não foram correspondidos por Parâmetros na Atividade representando a missão do sistema, vista na Figura D.7 acima. Estes fluxos são o da gravidade imposta pelo planeta Terra e da

radiação cósmica vinda do ambiente espacial onde o satélite está inserido. Para o caso sendo desenvolvido, não existe controle ativo de órbita e atitude. Portanto, não foram necessárias funções para tratamento destes fluxos. Eles estão relacionados com requisitos não-funcionais a respeito de tempo de duração da missão, no caso da gravidade, e qualidade dos componentes eletrônicos utilizados, no caso da radiação cósmica. Entretanto, no caso, por exemplo, de um controle ativo de órbita e atitude, funções seriam criadas para tratar estes fluxos e gerar controles para garantir a missão do satélite.

Seguindo a metodologia proposta, deve-se, agora, criar as especificações para estas funções terminais adicionais, assim como foi feito para as funções essenciais. Conforme as técnicas para geração automática de requisitos a partir do modelo descritivo do sistema apresentada no Capítulo 4, Seção 4.7, as especificações não são utilizadas diretamente para preencher lacunas dos padrões para enunciados de requisitos propostos naquela seção. Elas representam, na verdade, o processamento interno das funções estipuladas nos requisitos funcionais gerados a partir do modelo do sistema. Portanto, somente algumas das funções terminais adicionais apresentadas na Figura D.7 acima terão suas especificações apresentadas, a título de exemplo.

Na Figura D.8 abaixo, mostra-se a especificação para a função que processa os telecomandos recebidos da estação terrena.

Figura D.8 – Especificação da Função do AESP14 para Processar Telecomandos.



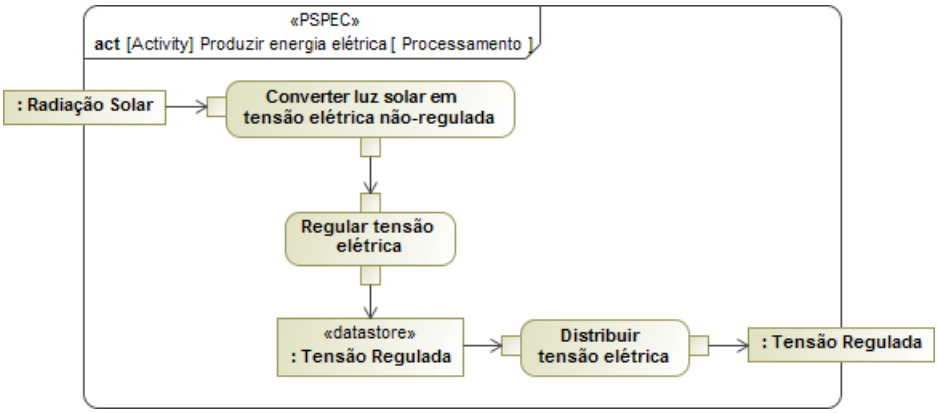
Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se, primeiramente, a tipificação do parâmetro de entrada da função com o tipo ‘Telecomando’ para garantir

consistência do modelo. Este tipo está precisamente definido no dicionário de requisitos. Mais ainda, do diagrama, percebe-se a existência de dois telecomandos esperados: telecomando para desligar o satélite e telecomando para reinicializar o satélite. Em cada um dos casos, a função dispara um sinal de controle correspondente ao telecomando recebido (as chamadas 'Condições de Dados' do método HHP, apresentado no Apêndice B). Estes eventos poderiam ser utilizados por outras funções para disparar algum tipo de processamento específico à ocorrência destas situações.

Na Figura D.9 abaixo, mostra-se a especificação para a função que produz energia elétrica dentro do satélite.

Figura D.9 – Especificação da Função do AESP14 para Produção de Energia Elétrica.



Fonte: Produção do Autor.

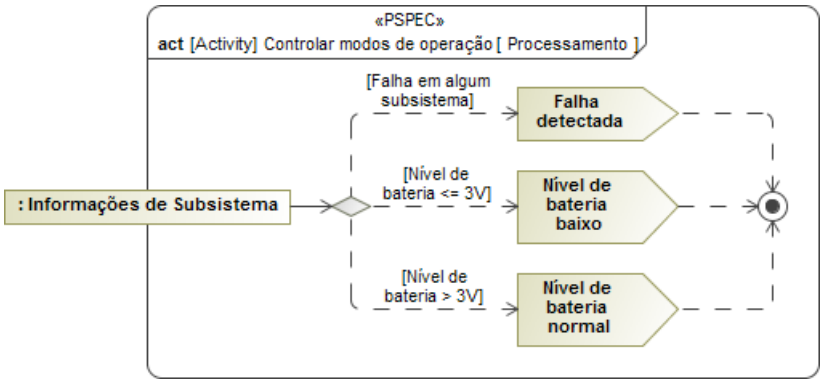
Novamente, no diagrama da figura acima, nota-se a tipificação dos parâmetros da correspondente função com tipos a serem definidos no dicionário de requisitos. Usando radiação solar, o sistema gera uma tensão não-regulada, que, por sua vez, é regulada e armazenada para posterior distribuição.

Na Figura D.10 abaixo, mostra-se a especificação para a função que controla os modos operacionais do sistema AESP14.

Neste caso, a função usa as informações de subsistema (tipo este já definido no modelo de entidade, na Seção D.1.1) para gerar alguns eventos internos ao sistema, indicando determinadas situações, que podem representar tanto falhas como normalidades dentro do satélite. Estes eventos serão utilizados

no modelo de controle, a ser apresentado na Seção D.1.3, como condições de disparo de transições de modo operacional.

Figura D.10 – Especificação da Função do AESP14 para Controle de Modos Operacionais.



Fonte: Produção do Autor.

Com isso, encerra-se o modelo de processo para o sistema AESP14, onde suas funções essenciais e auxiliares foram identificadas e descritas, de forma consistente, com o uso de tipos explicitamente definidos no modelo. Esta tipificação forte é o que garantirá a qualidade dos requisitos funcionais gerados a partir do modelo descritivo criado para o sistema.

D.1.3 Modelo de Controle

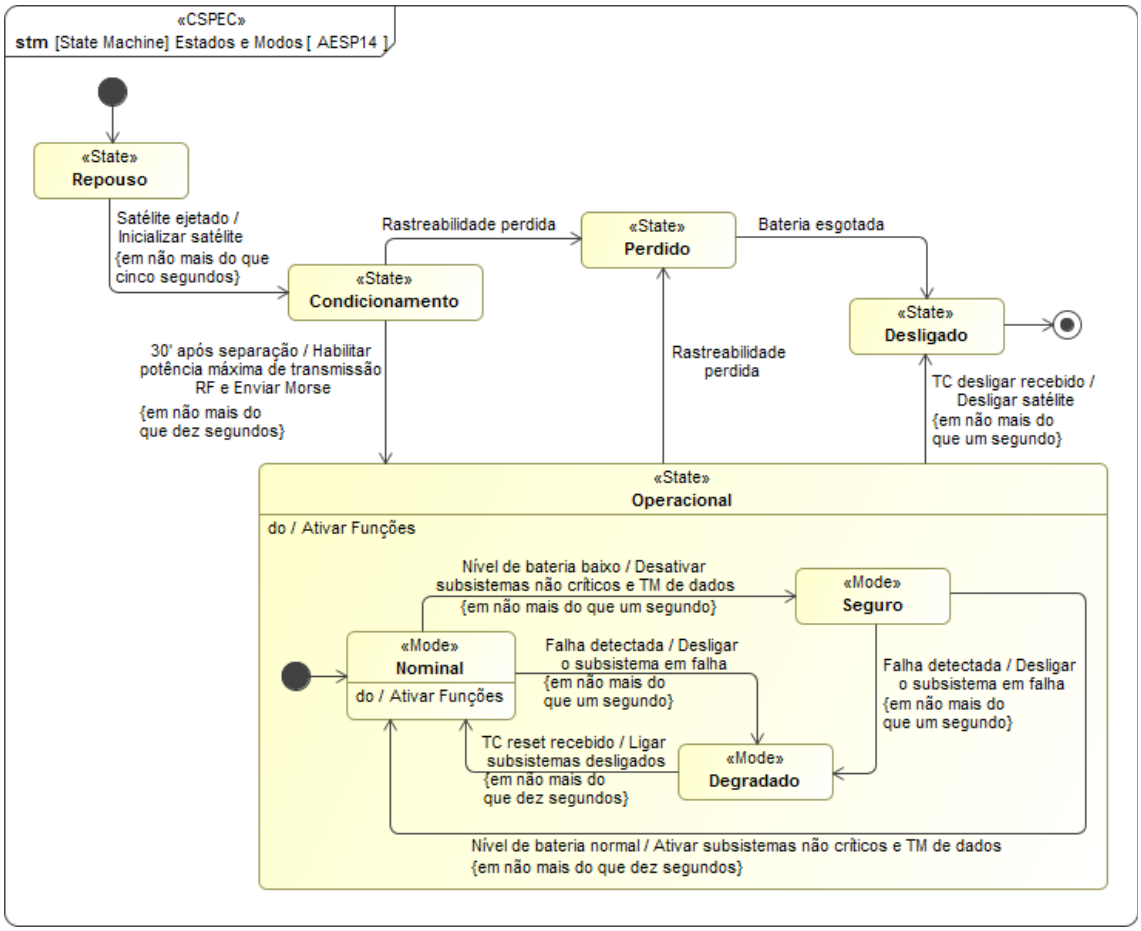
Conforme exposto no Capítulo 4, Seção 4.6, Subseção 4.6.2.3, o modelo de controle consiste, na verdade, na análise de estados e modos do sistema de interesse. Esta análise foi feita para o sistema AESP14 em AESP14 (2014) e aqui, portanto, será reproduzida utilizando a metodologia proposta na correspondente subseção do Capítulo 4.

Na Figura D.11 abaixo, mostra-se um Diagrama de Máquina de Estado da SysML com o resultado da análise de estados e modos do sistema AESP14.

Como resultado desta análise, foram identificados os seguintes estados: repouso, condicionamento, operacional, perdido e desligado. Enquanto no estado operacional, o sistema apresenta os seguintes modos operacionais: nominal, seguro e degradado. As transições no diagrama identificam as condições em que o sistema passa de um estado para outro ou de um modo operacional para outro. Algumas destas transições identificam, ainda,

comportamentos do sistema que são executados na mudança de estado ou modo operacional, com suas respectivas restrições de desempenho.

Figura D.11 – Estados e Modos do AESP14.



Fonte: Produção do Autor.

O estado de repouso é o estado inicial do sistema. Quando o satélite é ejetado, ele é inicializado em não mais do que cinco segundos e passa ao estado de condicionamento. Trinta minutos após a separação, o satélite tem sua potência máxima de transmissão RF habilitada e passa a enviar código morse, em não mais do que dez segundos após este evento. Mais ainda, o satélite muda para seu estado operacional. Ainda no estado de condicionamento ou estando no estado operacional, caso haja perda de rastreabilidade, o sistema muda para o estado perdido. Do estado perdido, o sistema passa ao estado desligado quando a energia de sua bateria se esgota. Outra possibilidade para chegar ao estado desligado, em não mais

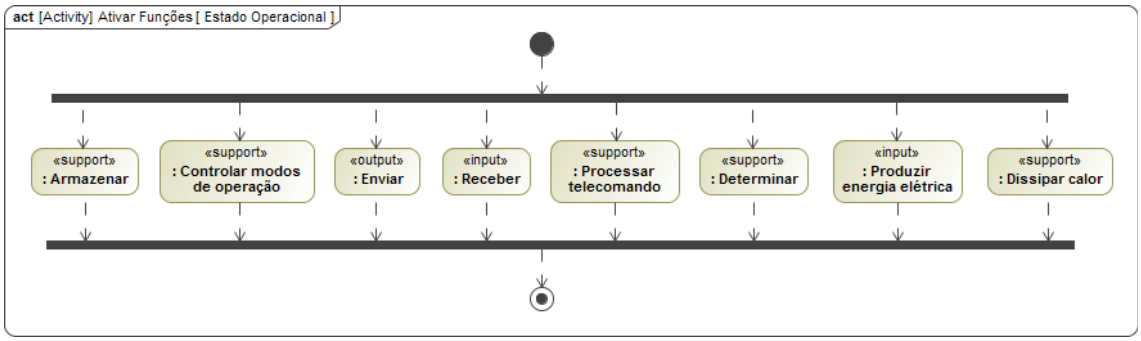
do um segundo, é receber um telecomando para desligar o satélite, enquanto este estiver no estado operacional. Este é, portanto, o estado final do satélite.

Estando no estado operacional, o modo nominal é o modo inicial do sistema. Caso detecte-se um nível baixo de tensão da bateria, desativam-se os subsistemas não críticos e a transmissão de telemetria, em não mais do que um segundo após o evento, levando o sistema ao modo seguro. Ainda no modo nominal, ao detectar-se uma falha em algum subsistema, desliga-se este subsistema em não mais do que um segundo após esta detecção e o satélite entra no modo degrado. No modo seguro, caso o nível de tensão da bateria volte ao normal, religam-se os subsistemas não críticos e a transmissão de telemetria em não mais do que dez segundos após o evento, levando o sistema de volta ao modo nominal. Ainda no modo seguro, caso uma falha em algum subsistema seja detectada, desliga-se este subsistema em não mais do que um segundo após a falha e o satélite entra no modo degradado. Neste modo, ao receber um telecomando para reinicializar o satélite, ligam-se os subsistemas desligados em não mais do que dez segundos após o telecomando e o sistema volta ao modo nominal.

O modelo de controle tem como objetivo principal prover um mecanismo de ativação de funções do sistema. Para tal, utilizam-se alguns compartimentos específicos do elemento Estado da SysML para identificar comportamentos que serão executados na entrada, saída ou enquanto o sistema permanecer naquele estado ou modo operacional. Na Figura D.11 acima, nota-se o uso do compartimento '*do*' no estado operacional e no modo nominal. Estes compartimentos referenciam duas Atividades com o nome "Ativar funções". Estas atividades têm por finalidade identificar, respectivamente, quais funções estão ativas no estado operacional, independente do modo operacional atual do sistema, e quais funções estão ativas somente quando o sistema estivar no modo nominal.

Na Figura D.12 abaixo, mostra-se um Diagrama de Atividade da SysML para a Atividade "Ativar funções" referenciada no compartimento '*do*' do estado operacional.

Figura D.12 – Funções do AESP14 Ativas no Estado Operacional.

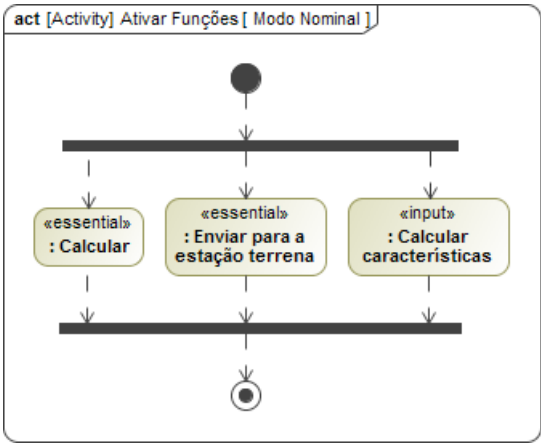


Fonte: Produção do Autor.

Pelo diagrama da figura acima, têm-se as funções ativas enquanto o sistema permanecer no estado operacional são: função de suporte para armazenar as telemetrias, função de suporte para controlar os modos operacionais do satélite, função de saída para enviar telemetrias à estação terrena, função de entrada para receber telecomandos da estação terrena, função de suporte para processar os telecomandos recebidos, função de suporte para determinar as informações dos subsistemas do satélite, função de entrada para produzir energia elétrica e, finalmente, função de suporte para dissipar o calor acumulado no satélite.

Na Figura D.13 abaixo, mostra-se um Diagrama de Atividade da SysML para a Atividade “Ativar funções” referenciada no compartimento ‘*do*’ do modo nominal.

Figura D.13 – Funções do AESP14 Ativas no Modo Nominal.



Fonte: Produção do Autor.

Do diagrama da figura acima, têm-se as funções do sistema AESP14 que estão ativas enquanto o satélite estiver no modo nominal de operação. São elas: função essencial para calcular as informações da carga útil usando o plasma ionosférico, função essencial para enviar para a estação terrena as informações da carga útil e função de entrada para calcular as características do plasma ionosférico usando a corrente elétrica medida.

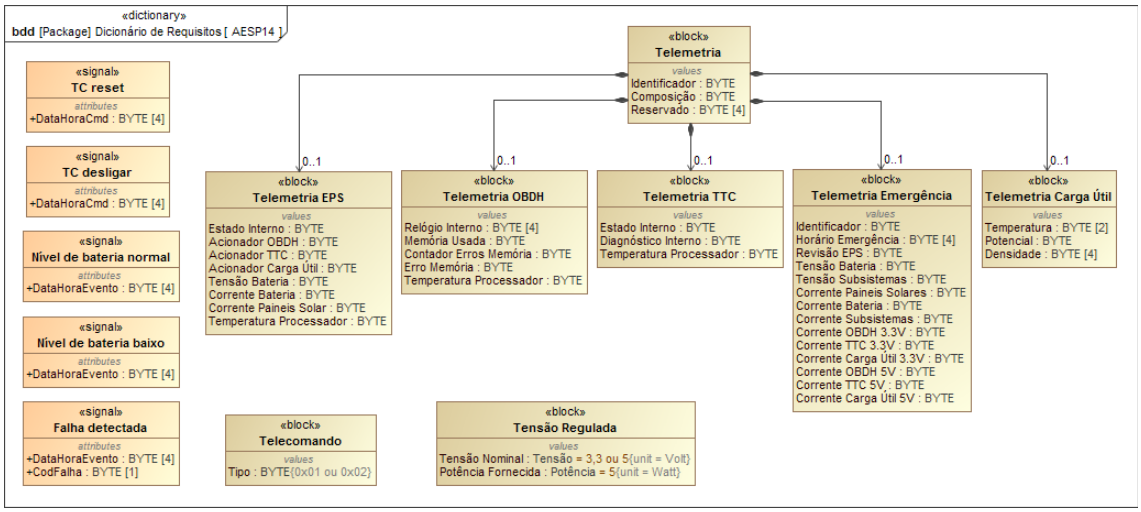
Com isso, encerra-se o modelo de controle para o sistema AESP14, onde seus estados e modos operacionais foram identificados, bem como as condições para as transições entre estados e modos operacionais. Mais ainda, foram determinadas as ativações das funções do sistema nos diferentes estados, modos operacionais e transições entre estes.

D.1.4 Dicionário de Requisitos

Conforme apresentado no Capítulo 4, Seção 4.6, Subseção 4.6.2.4, aqui estarão expostos os elementos da SysML usados para tipificar os parâmetros das funções identificadas anteriormente.

Na Figura D.14 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML contendo alguns dos elementos usados para tipificações no modelo de processos do sistema AESP14, desenvolvido na Seção D.1.2.

Figura D.14 – Dicionário de Requisitos do AESP14.



Fonte: Produção do Autor.

No canto esquerdo do diagrama da figura acima, têm-se os sinais (ou condições de dados) utilizados nas especificações das funções de processamento de telecomandos e controle de modos operacionais do satélite, apresentadas, respectivamente, na Figura D.8 e na Figura D.10. Mais ao centro, na parte superior, tem-se a definição do tipo utilizado para representar as telemetrias gerenciadas pelo satélite, extraídas de AESP-14 Telemetry (2015), e presente no diagrama da Figura D.7. Nota-se uma semelhança com os correspondentes tipos definidos no modelo de entidade, na Seção D.1.1, mas aqui o nível de abstração já está mais próximo do implementacional. Na parte de baixo do diagrama, têm-se os tipos para telecomandos e tensão regulada, utilizados para tipificar parâmetros das funções de processar telecomandos e produzir energia elétrica dentro do satélite, apresentadas, respectivamente, na Figura D.8 e na Figura D.9.

Ainda na Figura D.14 acima, nota-se o uso de Propriedades de Valor da SysML para os Blocos e Atributos da SysML para os Sinais. Estas informações fornecem os detalhes necessários para que os requisitos funcionais, que referenciarem estes Blocos ou Sinais, sejam completos.

Com isso, encerra-se o dicionário de requisitos para o sistema AESP14, onde toda a tipificação utilizada é precisamente definida e detalhada. Passa-se, agora, para o modelo de arquitetura do AESP14.

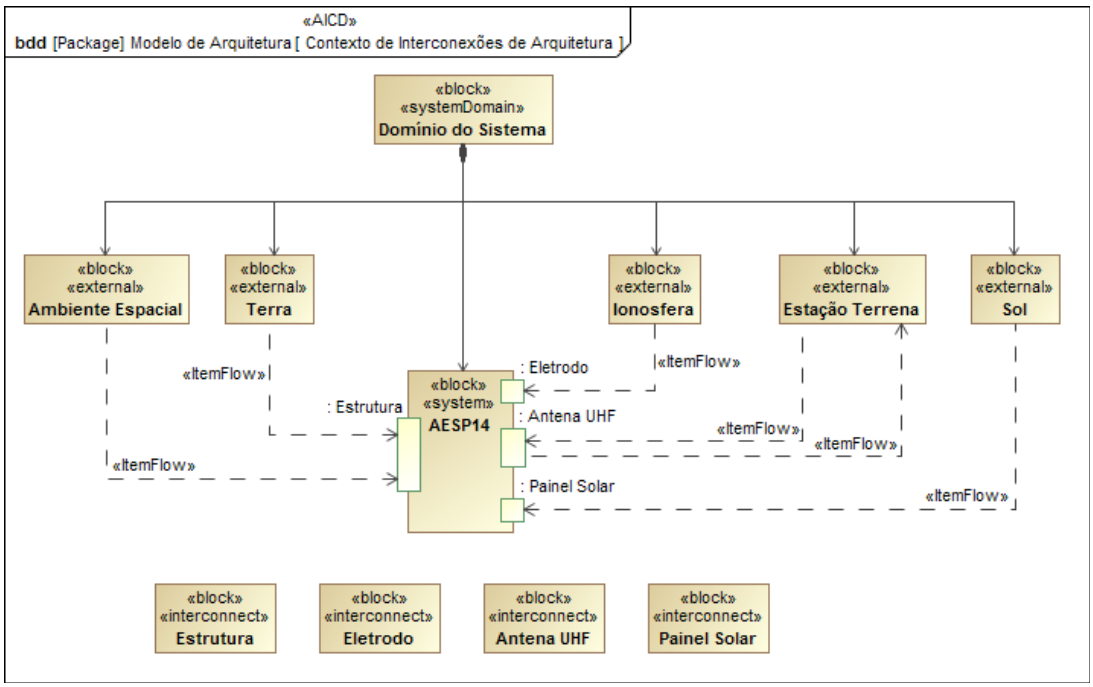
D.2 Modelo de Arquitetura

Esta seção apresenta a descrição da arquitetura de sistema AESP14, modelada utilizando a metodologia desenvolvida no Capítulo 4, Seção 4.6, Subseção 4.6.4. Ela está organizada da seguinte maneira. Na Subseção D.2.1, apresenta-se o modelo de contexto de arquitetura do AESP14. Os módulos de arquitetura e seus relacionamentos estão descritos na Subseção D.2.2. Na Subseção D.2.3, apresenta-se o dicionário de arquitetura que suplementa a descrição feita até então. Finalmente, na Subseção D.2.4, modelam-se algumas restrições de projeto, ambientais e de adequabilidade.

D.2.1 Modelo de Contexto de Arquitetura

Começando pelo diagrama de contexto de interconexões de arquitetura, na Figura D.15 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML representando as interfaces externas do sistema AESP14.

Figura D.15 – Contexto de Interconexões de Arquitetura do AESP14.



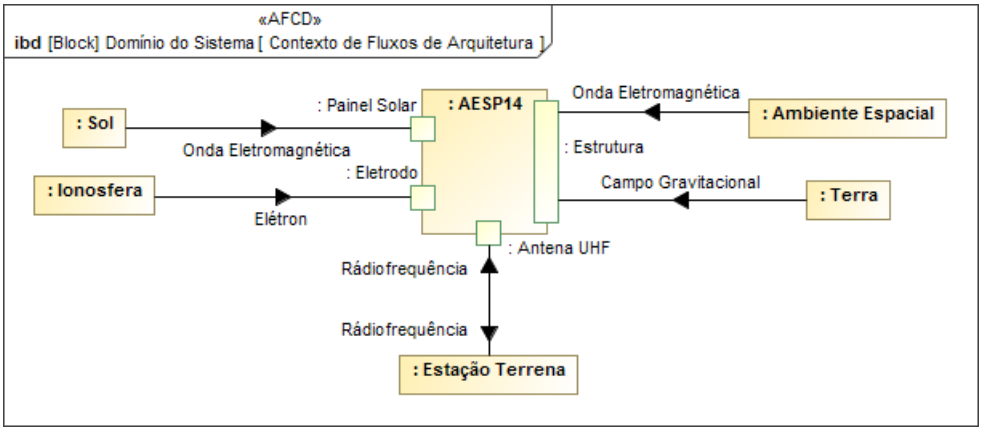
Fonte: Produção do Autor.

No diagrama da figura acima, vemos o Bloco representando o sistema AESP14 com as Portas da SysML que foram criadas para representar suas interfaces com as entidades externas. Veem-se, também, as linhas pontilhadas direcionadas representando, simplesmente, que fluxos de arquitetura são trocados entre o sistema e as entidades externas. A interface com o ambiente espacial no entorno do satélite e a interface com a Terra se dá pela própria estrutura mecânica do sistema, que sofrerá a ação de diversas radiações cósmicas e do campo gravitacional do planeta. A interface com a ionosfera se dá por um eletrodo capaz de realizar as medidas necessárias do plasma. Já a interface com a estação terrena se dá por uma antena UHF, por onde ondas de radiofrequência são recebidas e emanadas. Finalmente, a interface com o Sol se dá através de painéis solares capazes de captar a radiação solar para transformá-la em energia elétrica.

Na parte mais abaixo da Figura D.15 acima, têm-se os Blocos representando estas interfaces propriamente ditas. Estes estão devidamente identificados com o estereótipo <<interconnect>>. O objetivo de colocá-los no digrama é simplesmente mostrar que estes elementos são aqueles que tipificam as interfaces do sistema. De uma maneira geral, eles aparecerão no dicionário de arquitetura para uma definição precisa de suas propriedades.

Depois de se estabelecer o contexto de interconexões de arquitetura, deve-se modelar o contexto de fluxos de arquitetura. Na Figura D.16 abaixo, mostra-se um Diagrama Interno de Bloco da SysML, criado para o Bloco “Domínio do Sistema” visto na Figura D.15 acima, onde mostram-se os fluxos trocados entre o sistema AESP14 e as entidades externas em seu entorno.

Figura D.16 – Contexto de Fluxos de Arquitetura do AESP14.



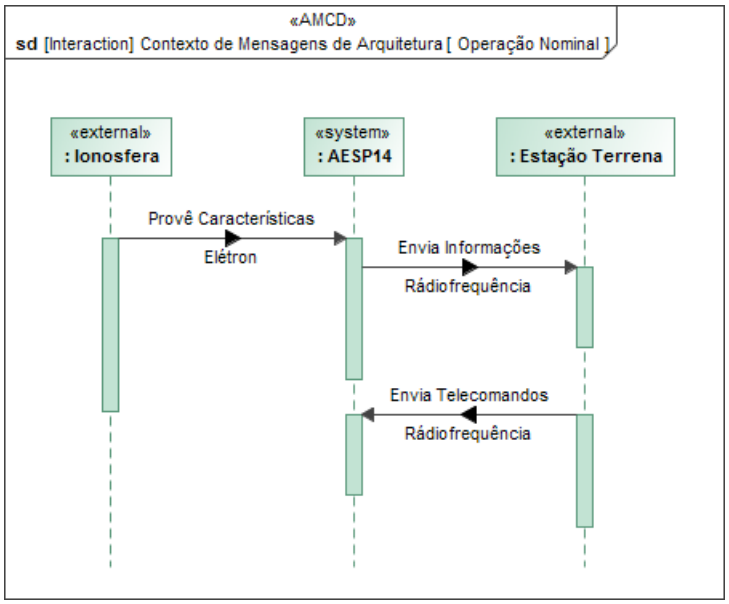
Fonte: Produção do Autor.

No diagrama da figura acima, tem-se o sistema AESP14 ao centro, com suas respectivas interfaces externas. Conforme já mencionado, e agora de forma explícita no diagrama, a radiação cósmica imposta pelo ambiente espacial ao redor do satélite chega à sua estrutura através de ondas eletromagnéticas. Também através desta mesma interface, o campo gravitacional do planeta Terra exerce sua atração. As ondas eletromagnéticas emitidas pelo Sol chegam ao painel solar do satélite para que este consiga gerar a energia elétrica necessária. As diferenças de potenciais aplicadas ao eletrodo inserido no plasma ionosférico fazem com que elétrons se desloquem e gerem uma corrente elétrica que é capturada pelo sistema e medida para se

determinar as características deste plasma. Sinais de radiofrequência são trocados com a estação terrena através de uma antena de UHF a fim de se transmitir telemetrias e receber telecomandos.

Com o objetivo de mostrar um dado cenário operacional, o engenheiro de sistemas pode elaborar Diagramas de Sequência da SysML com as partes envolvidas. Na Figura D.17 abaixo, mostra-se a troca de mensagens de arquitetura no cenário nominal de operação do sistema AESP14.

Figura D.17 – Contexto de Mensagens de Arquitetura do AESP14.



Fonte: Produção do Autor.

No diagrama da figura acima, tem-se um cenário simples quando o sistema determina as características do plasma ionosférico pela corrente elétrica gerada na movimentação dos elétrons através do eletrodo nele inserido e envia as telemetrias correspondentes à estação terrena através de sinais de radiofrequência. Mostra-se ainda, o recebimento de telecomandos, também através de sinais de radiofrequência.

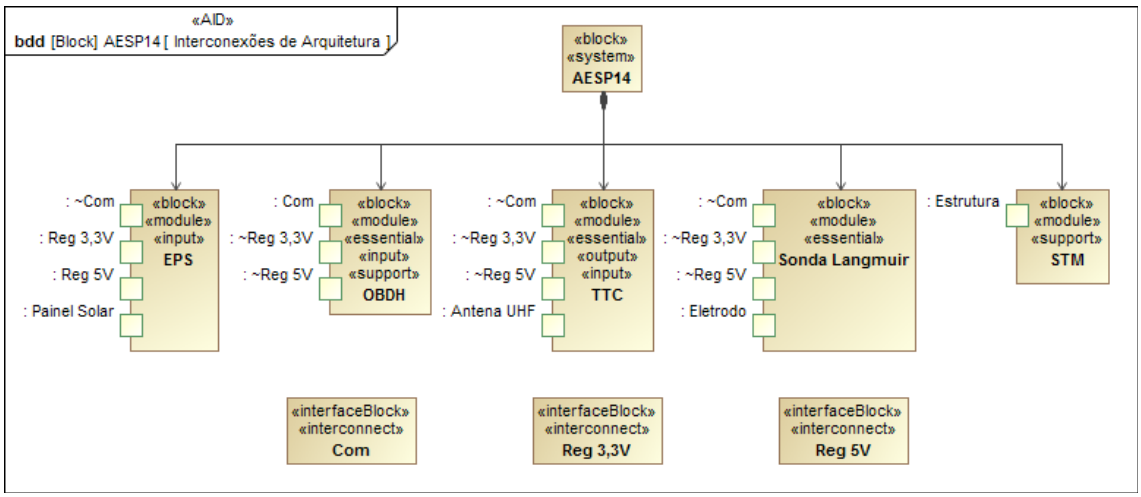
Termina aqui o modelo de contexto de arquitetura, onde foi definido como o sistema AESP14 trocará material, energia e informação com as entidades externas identificadas. Passa-se agora para a etapa onde define-se as partes constituintes deste sistema e como elas interagem para promover as

funcionalidades especificadas no modelo de requisitos, elaborado na Seção D.1 deste apêndice.

D.2.2 Modelo de Módulos de Arquitetura

Depois de definir o contexto de arquitetura do sistema, passa-se para a definição da sua arquitetura interna. Na Figura D.18 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML com as interconexões de arquitetura para o sistema AESP14.

Figura D.18 – Interconexões de Arquitetura do AESP14.



Fonte: Produção do Autor.

No diagrama da figura acima, vê-se a composição do sistema AESP14 nos seguintes módulos de arquitetura: EPS, OBDH, TTC, Sonda Langmuir e STM. O módulo de arquitetura EPS fornece a energia elétrica necessária dentro do satélite, gerada a partir da energia solar, e, por isso, foi classificado como um módulo de entrada. Já o módulo de arquitetura OBDH, considerado tanto como um módulo essencial como módulo de entrada e de suporte, é responsável pela capacidade de processamento computacional dentro de satélite. O módulo de arquitetura TTC é responsável pelas telecomunicações dentro do satélite, ou seja, receber telecomandos e enviar telemetrias, e, por isso, foi classificado tanto como um módulo essencial como um módulo de entrada e de saída. O módulo de arquitetura Sonda Langmuir representa a carga útil do satélite, responsável pelas medições das características do plasma ionosférico, e que, por isso, foi classificado como sendo um módulo

essencial do sistema. Finalmente, o módulo de arquitetura STM, considerado como um módulo de suporte, representa a estrutura mecânica do satélite, que deve passivamente tanto dissipar o calor excedente quanto controlar a atitude do satélite.

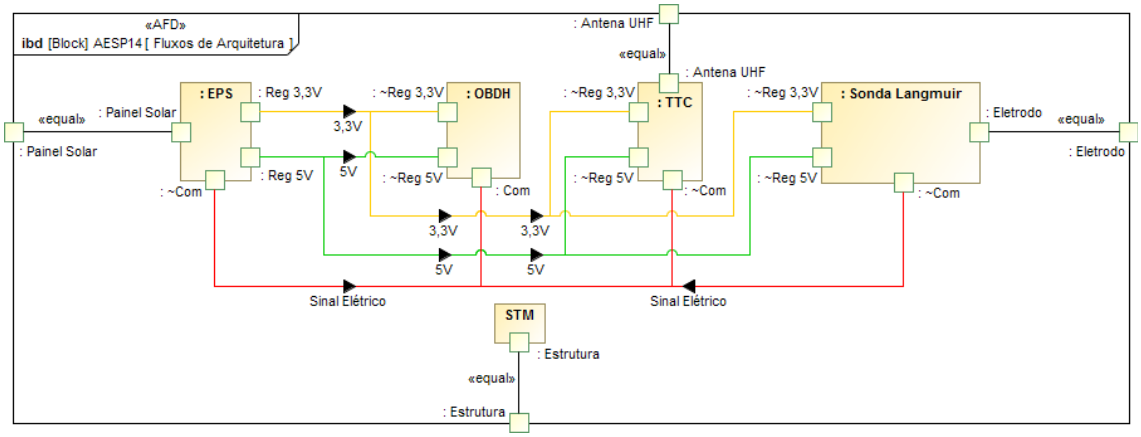
Mais ainda, pelo diagrama da Figura D.18 acima, têm-se as interfaces que cada módulo de arquitetura expõe. Todos os módulos de arquitetura, com exceção do módulo STM, apresentam uma interface tipificada pela interconexão chamada de “Com”. Esta interconexão representam um barramento de comunicação interna ao satélite, por onde os subsistemas trocam informações. Estes mesmos módulos de arquitetura apresentam duas interfaces cada, tipificadas pelas interconexões chamadas de “Reg 3,3V” e “Reg 5V”. Estas interconexões representam os barramentos de tensão regulada de 3,3 Volts e 5 Volts, respectivamente, por onde o módulo EPS fornece energia elétrica e os demais subsistemas a consomem. O módulo de arquitetura EPS apresenta, ainda, uma interface tipificada pela interconexão chamada de “Painel Solar”, por onde este receberá a radiação solar a ser transformada em energia elétrica. O módulo de arquitetura TTC apresenta, também, uma interface tipificada pela interconexão chamada de “Antena UHF”, por onde as ondas de radiofrequência de telemetrias e telecomandos serão trocadas. Já o módulo de arquitetura Sonda Langmuir expõe uma outra interface tipificada pela interconexão chamada de “Eletrodo”, que representa o dispositivo usado para efetuar as medições das características do plasma ionosférico. Finalmente, o módulo de arquitetura STM tem sua única interface tipificada pela interconexão chamada de “Estrutura”, que representa a estrutura mecânica do satélite.

Assim como visto no diagrama de contexto de interconexões de arquitetura do sistema AESP14, apresentado anteriormente na Figura D.15, neste diagrama de interconexões dos módulos de arquitetura da Figura D.18 acima, mostram-se na parte inferior os Blocos de Interface utilizados como interconexões e que foram utilizados para tipificar as Portas da SysML representando as diferentes interfaces dos módulos de arquitetura. Estes

elementos aparecerão novamente no dicionário de arquitetura, onde se fornecerão maiores detalhes.

Após identificar as interfaces que cada módulo de arquitetura expõe, detalham-se os fluxos de arquitetura entre eles trocados. Na Figura D.19 abaixo, mostra-se um Diagrama Interno de Bloco da SysML, criado para o Bloco representando o sistema AESP14 visto na Figura D.18 acima, com os fluxos de arquitetura trocados entre seus módulos, igualmente encontrados na Figura D.18 acima.

Figura D.19 – Fluxos de Arquitetura do AESP14.



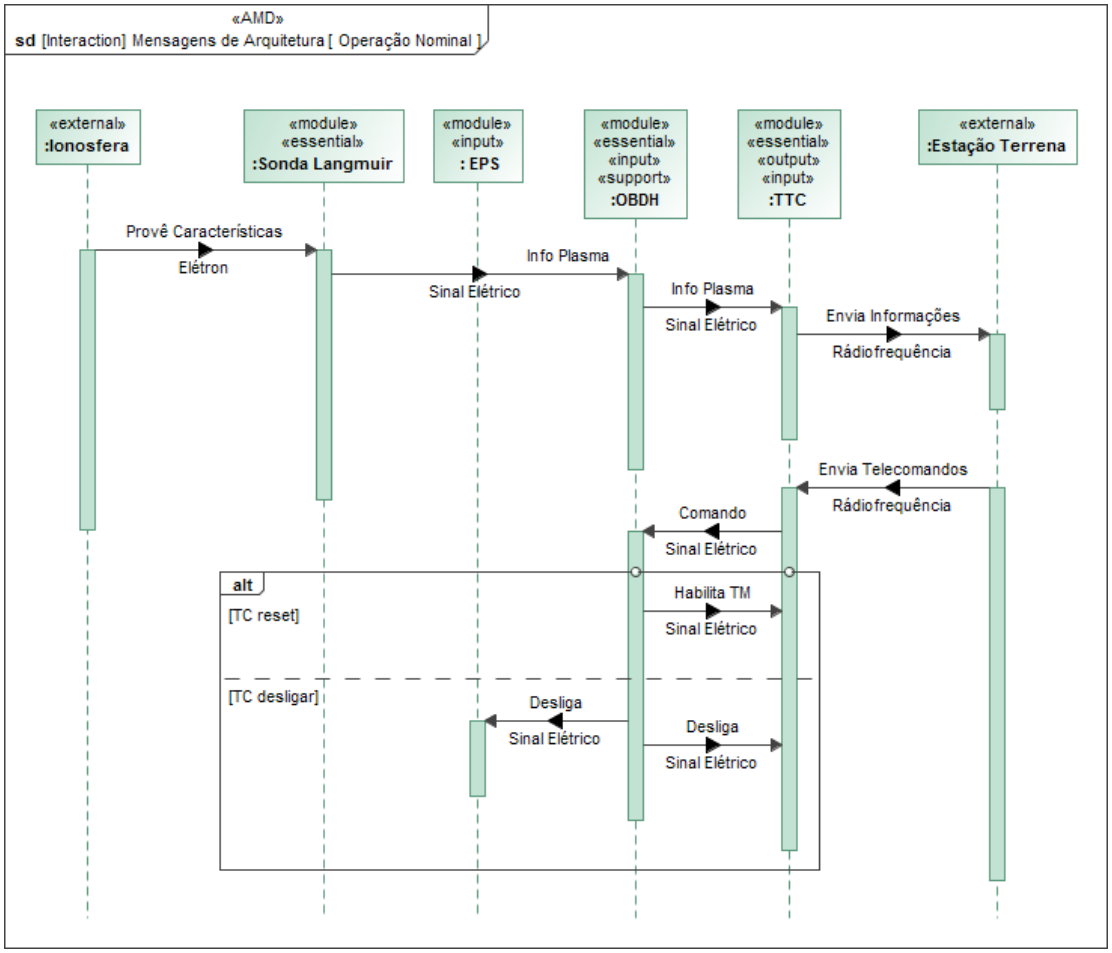
Fonte: Produção do Autor.

Conforme definido na SysML, as bordas do diagrama da figura acima representam, na verdade, as fronteiras do sistema AESP14 com o ambiente externo. Por isso, têm-se, nestas bordas, as Portas da SysML representando as interfaces do satélite, já apresentadas na Figura D.15 e na Figura D.16, conectadas às correspondentes interfaces dos módulos de arquitetura. No diagrama da Figura D.19 acima, foram utilizadas cores para os conectores interligando os módulos de arquitetura simplesmente como um facilitador, pois como as linhas se cruzam, deixando-as da mesma cor poderia causar algum tipo de confusão para o leitor do diagrama. Em vermelho, tem-se o barramento de comunicação, por onde os módulos trocam pacotes de dados com as informações necessárias, através de um sinal elétrico específico ao protocolo de comunicação utilizado. Em amarelo, destaca-se o barramento de tensão regulada de 3,3 Volts e, em vermelho, nota-se o barramento de

tensão regulada de 5 Volts. Ambos, utilizados pelo módulo EPS para fornecimento de energia elétrica aos demais módulos de arquitetura.

Assim como feito para o sistema AESP14 e as entidades externas com as quais interage, pode-se optar por mostrar mensagens de arquitetura trocadas entre os módulos de arquitetura do sistema em um dado cenário operacional. Na Figura D.20 abaixo, mostra-se um Diagrama de Sequência da SysML representando a troca de mensagens de arquitetura no cenário nominal de operação do satélite.

Figura D.20 – Mensagens de Arquitetura do AESP14.



Fonte: Produção do Autor.

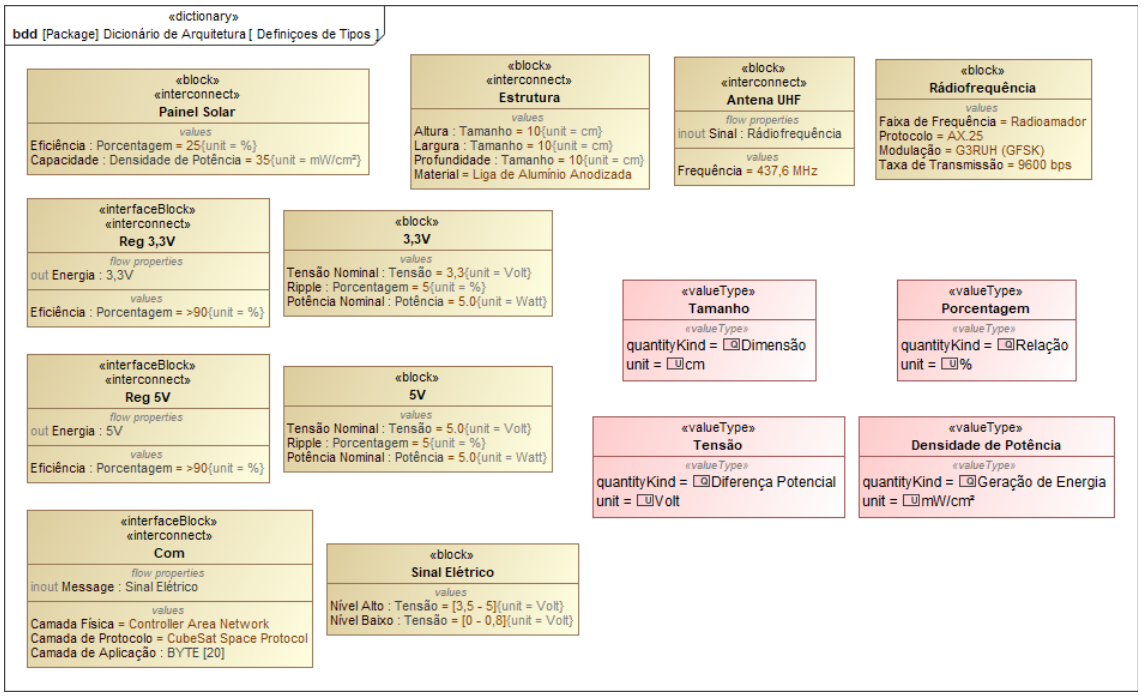
No diagrama da figura acima, mostra-se o mesmo cenário operacional nominal apresentado anteriormente na Figura D.17, agora com os módulos de arquitetura ao invés do sistema AESP14. A Sonda Langmuir é a responsável por captar os elétrons do plasma ionosférico e realizar medidas

para determinar suas características. Depois, este módulo envia as informações ao computador de bordo, representado pelo módulo de arquitetura OBDH, que as envia ao módulo TTC para que sejam transmitidas à estação terrena via telemetrias. Já os telecomandos são recebidos pelo módulo TTC, que os encaminha ao computador de bordo. Este os processa e implementa os comandos solicitados pela estação terrena: reinicializar ou desligar o satélite. O primeiro causa o envio de um comando de reinício ao módulo TTC e o segundo se traduz em um comando de desligamento enviado ao módulo EPS.

D.2.3 Dicionário de Arquitetura

Na Figura D.21 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML especificando alguns dos tipos utilizados no modelo de arquitetura do sistema AESP14.

Figura D.21 – Dicionário de Arquitetura do AESP14.



Fonte: Produção do Autor.

No diagrama da figura acima, veem-se alguns dos tipos utilizados no modelo de arquitetura para tipificar interconexões ou fluxos de arquitetura. Assim como feito no dicionário de requisitos, deve-se especificar com precisão

todos os tipos utilizados no modelo de arquitetura. No modelo de requisitos, o seu correspondente dicionário de dados faz com que os requisitos gerados a partir do modelo descritivo do sistema sejam completos e livres de ambiguidades. Já aqui no modelo de arquitetura, seu correspondente dicionário de dados, além de contribuir para a completude e clareza dos requisitos gerados a partir do modelo, faz, também, com que estes sejam observáveis e, portanto, verificáveis.

Do diagrama da Figura D.21 acima, tem-se que a interconexão chamada “Painel Solar” possui valores para definir sua eficiência e sua capacidade de geração de energia. A interconexão chamada “Estrutura” possui valores para especificar suas dimensões e o tipo de material que deve ser utilizado na sua construção. Já a interconexão chamada “Antena UHF” possui um valor para definir qual será a frequência utilizada para trocar sinais de radiofrequência definidos pelo tipo chamado “Radiofrequência”, que contém valores especificando sua faixa de frequência, protocolo, modulação e taxa de transmissão de dados. A interconexão chamada “Reg 3,3V” possui um valor para definir sua eficiência e uma Propriedade de Fluxo da SysML para identificar qual tipo de fluxo pode ser trocado através desta interconexão e em qual sentido esta troca acontece. Neste caso, corresponde ao tipo chamado “3,3V”, que possui valores definindo sua tensão nominal, *ripple* e potência nominal. Mais ainda, a classificação ‘*out*’ ao lado de seu nome indica que fluxos deste tipo saem da correspondente Porta da SysML tipificada por esta interconexão, no módulo de arquitetura EPS, e chegam nas Portas igualmente tipificada por esta interconexão e conjugadas, nos demais módulos de arquitetura. De forma análoga, a interconexão chamada “Reg 5V” possui um valor para especificar sua eficiência e fluxos tipificados pelo Bloco chamado “5V”, que possui valores definindo sua tensão nominal, *ripple* e potência nominal. Finalmente, a interconexão chamada “Com” possui valores definindo a camada física, a camada de protocolo e a camada de aplicação utilizadas na comunicação entre os módulos de arquitetura. Mais ainda, esta interconexão possui um Fluxo de Propriedade para identificar que sinais elétricos podem entrar e sair por este ponto de interação, sinais estes

tipificados pelo Bloco chamado “Sinal Elétrico”, que possui valores para definir as tensões de níveis lógicos alto e baixo.

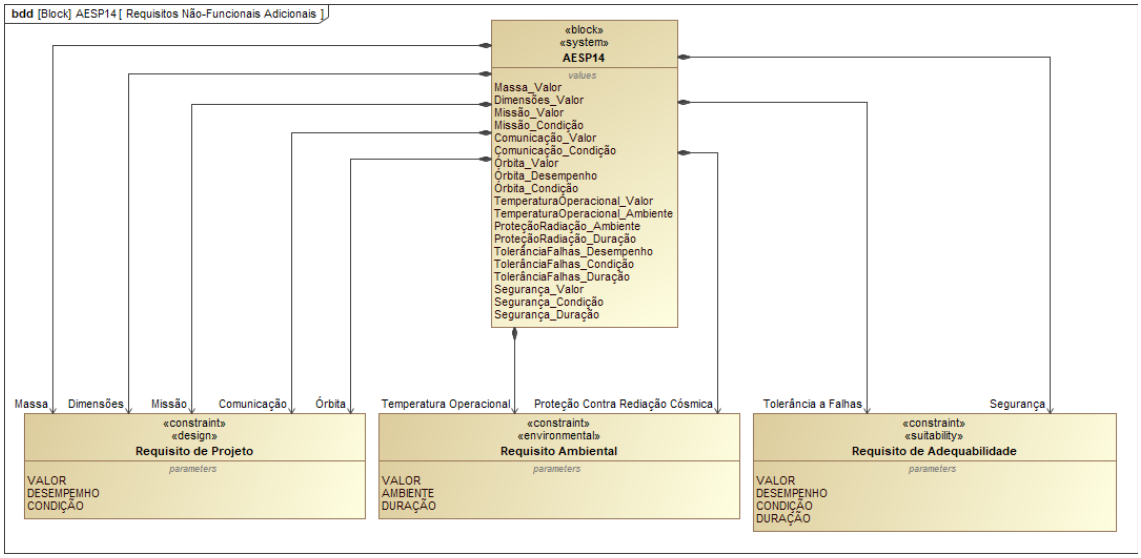
Ao completar o dicionário de arquitetura, o modelo do sistema AESP14 está concluído. Basta, agora, relacionar o modelo de requisitos ao modelo de arquitetura para possibilitar a geração automática dos seus requisitos de sistema.

D.2.4 Restrições de Projeto, Ambientais e de Adequabilidade do AESP14

Conforme abordado no Capítulo 4, Seção 4.6, Subseção 4.6.4.4, alguns requisitos não-funcionais do sistema seguirão o padrão definido por Carson (2015). Para tal, faz-se necessário incorporá-los ao modelo do sistema.

Na Figura D.22 abaixo, mostra-se um diagrama com a incorporação no modelo SysML do sistema AESP14 de algumas restrições de projeto, ambientais e de adequabilidade.

Figura D.22 – Requisitos Não-Funcionais Adicionais do AESP14.



Fonte: Produção do Autor.

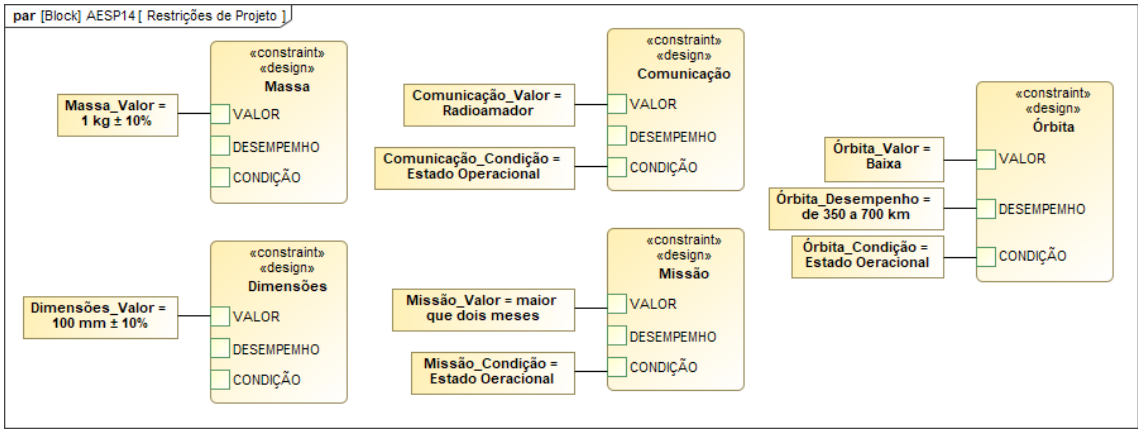
No diagrama da figura acima, vê-se o Bloco do sistema e diversos relacionamentos de Composição com os Blocos de Restrição representando os diferentes tipos de requisitos não-funcionais propostos por Carson (2015). Cada relacionamento deste tipo representa uma Propriedade de Restrição no escopo do Bloco do sistema e os nomes destes elementos representam as

características do sistema, cujos correspondentes requisitos não-funcionais restringirão. Estes nomes aparecem próximos às setas dos correspondentes relacionamentos de Composição.

Notam-se, dentro do compartimento intitulado "*values*" no Bloco do sistema, as Propriedades de Valor, cujos valores preencherão as lacunas dos correspondentes enunciados dos requisitos não-funcionais, e que aparecerão nos diferentes Diagramas Paramétricos criados para o Bloco do sistema.

Na Figura D.23 abaixo, mostra-se o Diagrama Paramétrico criado para o Bloco do sistema, onde os requisitos não-funcionais do tipo 'Requisitos de Projeto' são parametrizados.

Figura D.23 – Restrições de Projeto do AESP14.

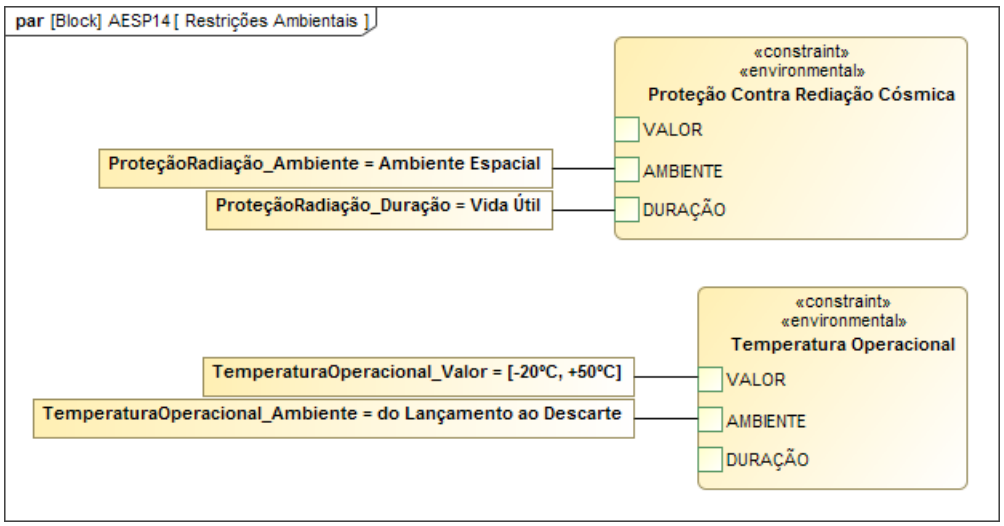


Fonte: Produção do Autor.

No diagrama da figura acima, veem-se as quatro Propriedades de Restrição do AESP14, representando as restrições de projeto criadas. Para cada uma destas restrições, têm-se as lacunas do padrão definido por Carson (2015) e as Propriedades de Valor usadas para seus devidos preenchimentos.

Na Figura D.24 abaixo, mostra-se o Diagrama Paramétrico criado para o Bloco do sistema, onde os requisitos não-funcionais do tipo 'Requisito Ambiental' são parametrizados.

Figura D.24 – Restrições de Projeto do AESP14.

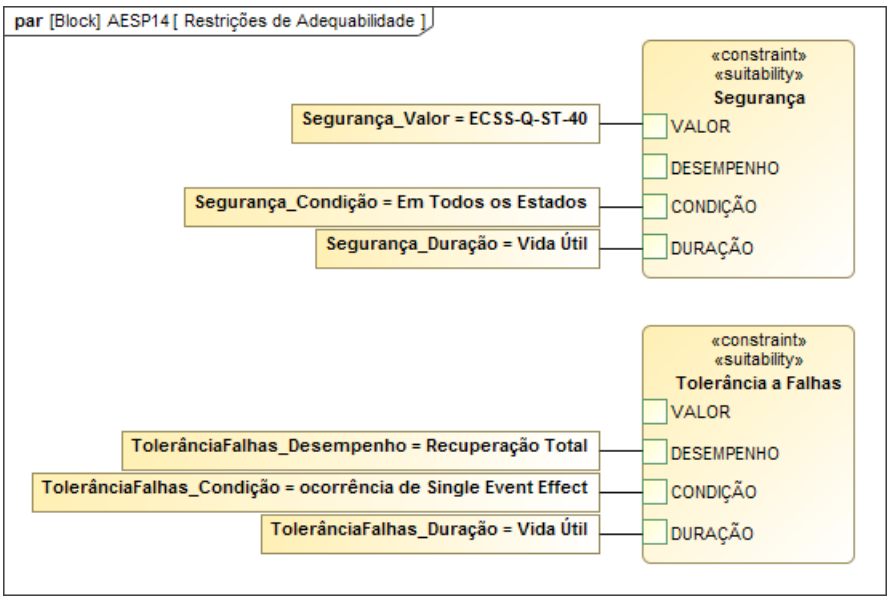


Fonte: Produção do Autor.

No diagrama da figura acima, veem-se as duas Propriedades de Restrição do AESP14, representando as restrições ambientais criadas. Para cada uma destas restrições, têm-se as lacunas do padrão definido por Carson (2015) e as Propriedades de Valor usadas para seus devidos preenchimentos.

Finalmente, na Figura D.25 abaixo, mostra-se o Diagrama Paramétrico criado para o Bloco do sistema, onde os requisitos não-funcionais do tipo ‘Requisito de Adequabilidade’ são parametrizados.

Figura D.25 – Restrições de Adequabilidade do AESP14.



Fonte: Produção do Autor.

No diagrama da figura acima, têm-se as duas Propriedades de Restrição do AESP14, representando as restrições de adequabilidade criadas. Para cada uma destas restrições, têm-se as lacunas do padrão definido por Carson (2015) e as Propriedades de Valor usadas para seus devidos preenchimentos.

Como isso, encerra-se a atividade de criação de restrições para o sistema AESP14. Passa-se, então, para a geração automática de requisitos a partir do modelo SysML criado para o satélite.

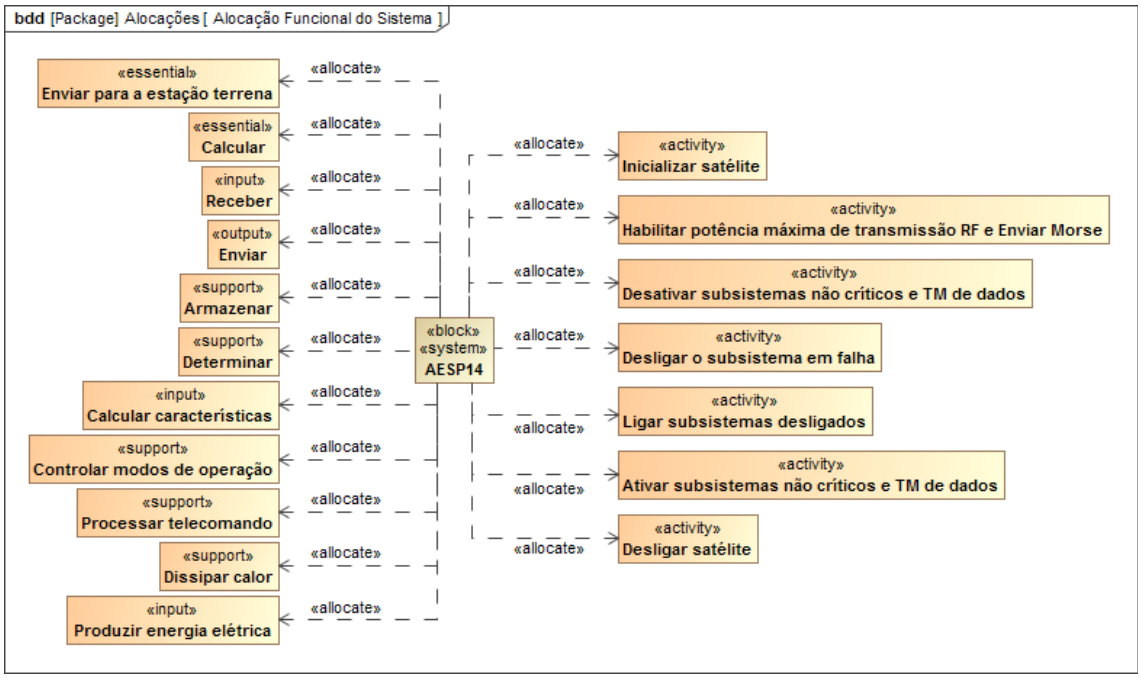
D.3 Relacionando os Modelos de Requisitos e de Arquitetura

D.3.1 Alocação de Funções e seus Parâmetros

De acordo com a metodologia descrita no Capítulo 4, Seção 4.6, Subseção 4.6.5, deve-se alocar as funções terminais identificadas no modelo de requisitos tanto ao sistema quanto aos módulos de arquitetura definidos no modelo de arquitetura. Este último tipo de alocação deve ser unívoco, ou seja, uma função deve estar alocada exclusivamente a um único módulo de arquitetura. Caso isso não seja possível, deve-se continuar o processo de desdobramento funcional para as funções nestas condições, até que se obtenha subfunções terminais passíveis de serem alocados a um único módulo de arquitetura. Este é o chamado projeto detalhado. Deve-se, também, alocar os parâmetros das funções terminais existentes no modelo de requisitos às Portas da SysML tanto do sistema como dos módulos de arquitetura definidos no modelo de arquitetura.

Na Figura D.26 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML, utilizado para realizar a alocação funcional ao sistema AESP14. Nele, notam-se, à esquerda do Bloco representando o sistema, as funções terminais do modelo de requisitos apresentadas na Figura D.3 e na Figura D.7 alocadas ao sistema AESP14. Já à direita deste Bloco, notam-se as funções referenciadas pelas Transições da CSPEC, vista na Figura D.11, alocadas ao sistema AESP14.

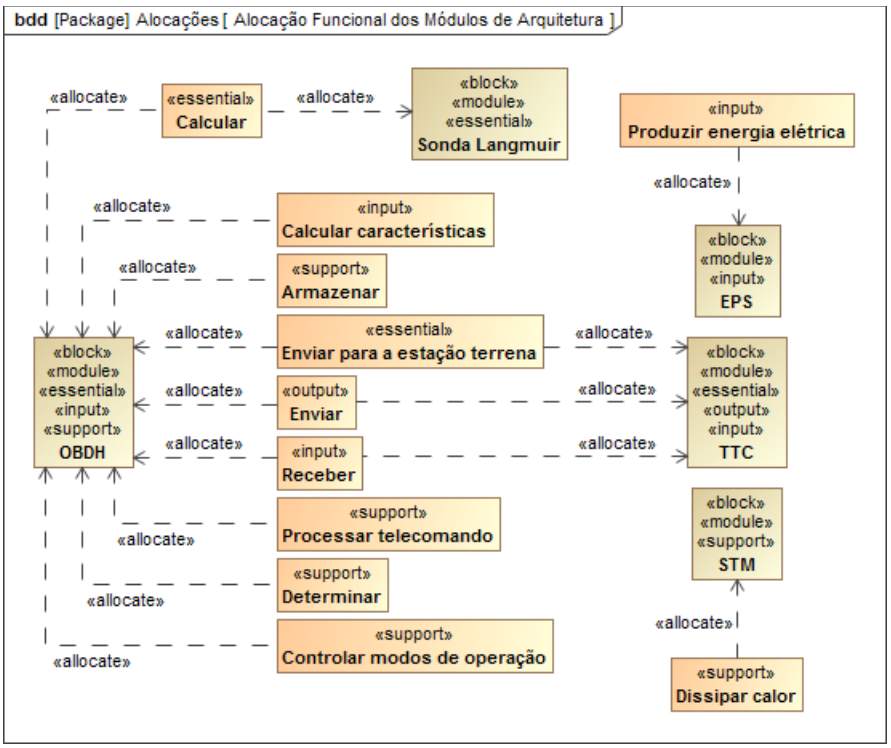
Figura D.26 – Alocação Funcional do Sistema AESP14.



Fonte: Produção do Autor.

Já na Figura D.27 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML, utilizado para alocar funções aos módulos de arquitetura do AESP14.

Figura D.27 – Alocação Funcional dos Módulos de Arquitetura do AESP14.



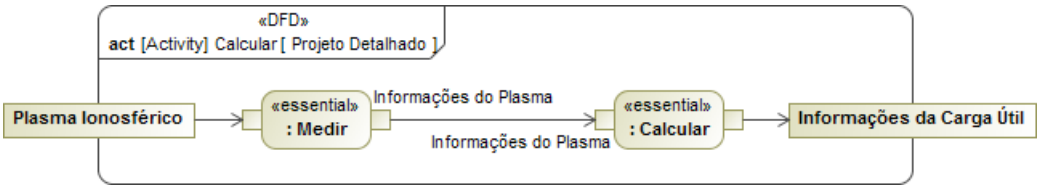
Fonte: Produção do Autor.

No diagrama da figura acima, notam-se as mesmas funções terminais vistas anteriormente na Figura D.26, mas agora alocadas aos diferentes módulos de arquitetura do sistema AESP14. Observa-se que algumas destas funções foram alocadas a mais de um módulo de arquitetura. A função essencial “Calcular” foi alocada tanto ao módulo Sonda Langmuir como ao módulo OBDH. A função essencial “Enviar para a estação terrena”, a função de saída “Enviar” e a função de entrada “Receber” foram alocadas tanto ao módulo OBDH como ao módulo TTC. Portanto, estas funções deverão ser desdobra das em subfunções que sejam passíveis de alocação a um único módulo de arquitetura do sistema AESP14.

D.3.2 O Projeto Detalhado

Na Figura D.28 abaixo, mostra-se o desdobramento da função essencial “Calcular” através de um Diagrama de Atividade da SysML.

Figura D.28 – Projeto Detalhado da Função Essencial “Calcular”.

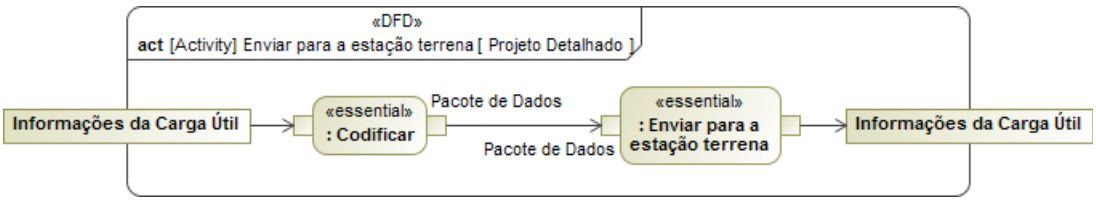


Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função essencial “Calcular” apresentada no diagrama da Figura D.3 foi decomposta em duas subfunções essenciais: uma para medir informações do plasma usando o próprio plasma ionosférico como entrada e outra para calcular as informações da carga útil usando as informações do plasma previamente medidas. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura Sonda Langmuir e a segunda ao módulo de arquitetura OBDH.

Na Figura D.29 abaixo, mostra-se o desdobramento da função essencial “Enviar para a estação terrena” através de um Diagrama de Atividade da SysML.

Figura D.29 – Projeto Detalhado da Função “Enviar para a estação terrena”.

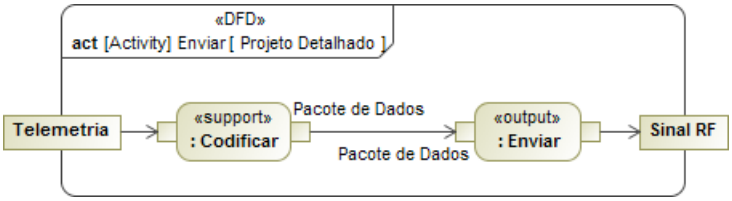


Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função essencial “Enviar para a estação terrena” apresentada no diagrama da Figura D.3 foi decomposta em duas subfunções essenciais: uma para codificar um pacote de dados usando as informações da carga útil calculadas anteriormente e outra para enviar para a estação terrena as informações da carga útil usando o pacote de dados anteriormente codificado. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura OBDH e a segunda ao módulo de arquitetura TTC.

Na Figura D.30 abaixo, mostra-se o desdobramento da função de saída “Enviar” através de um Diagrama de Atividade da SysML.

Figura D.30 – Projeto Detalhado da Função de Saída “Enviar”.

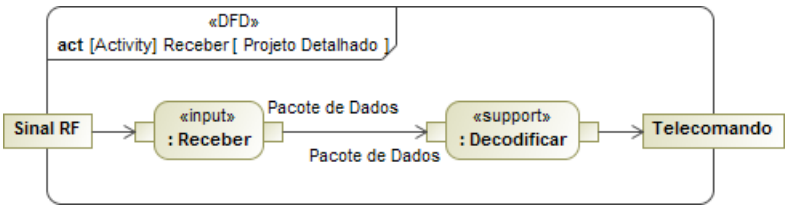


Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função de saída “Enviar” apresentada no diagrama da Figura D.7 foi decomposta em duas subfunções: uma subfunção de suporte para codificar um pacote de dados usando a telemetria armazenada e outra subfunção de saída para enviar o sinal RF à estação terrena usando o pacote de dados previamente codificado. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura OBDH e a segunda ao módulo de arquitetura TTC.

Na Figura D.31 abaixo, mostra-se o desdobramento da função de saída “Receber” através de um Diagrama de Atividade da SysML.

Figura D.31 – Projeto Detalhado da Função de Entrada “Receber”.

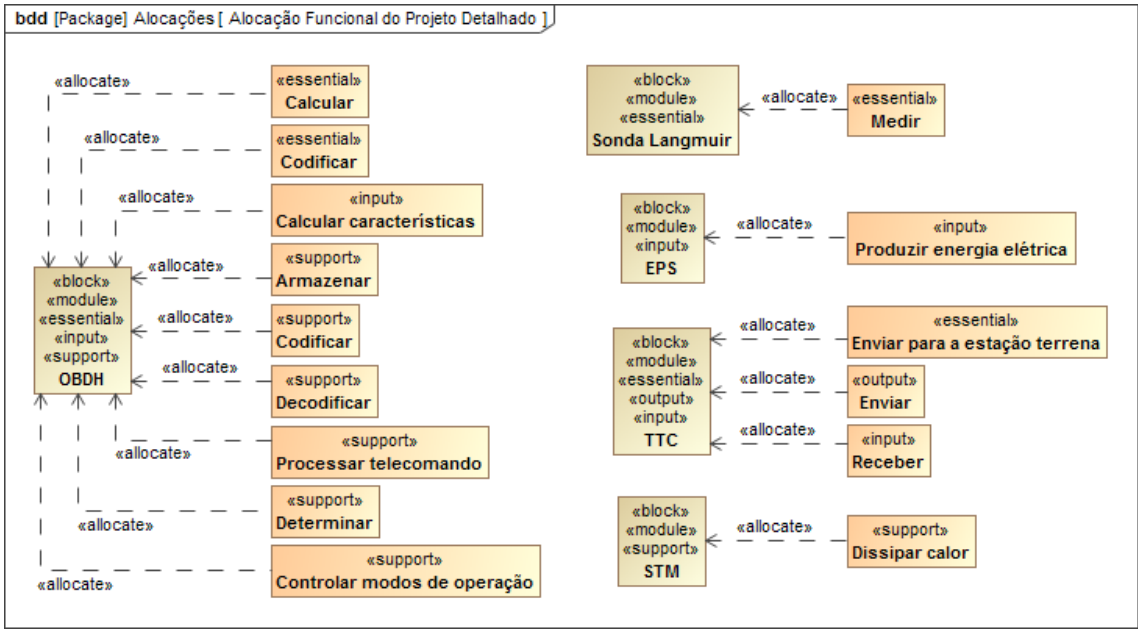


Fonte: Produção do Autor.

Pelo diagrama da figura acima, nota-se que a função de entrada “Receber” apresentada no diagrama da Figura D.7 foi decomposta em duas subfunções: uma subfunção de entrada para receber um pacote de dados usando o sinal RF contendo o telecomando e outra subfunção de suporte para decodificar o telecomando usando o pacote de dados previamente recebido. Com essa decomposição, torna-se possível alocar a primeira ao módulo de arquitetura TTC e a segunda ao módulo de arquitetura OBDH.

De posse destas novas funções, é possível refazer a alocação funcional aos módulos de arquitetura para termos o projeto detalhado completo. Na Figura D.32 abaixo, mostra-se um Diagrama de Definição de Bloco da SysML com esta alocação.

Figura D.32 – Alocação Funcional para o Projeto Detalhado do AESP14.



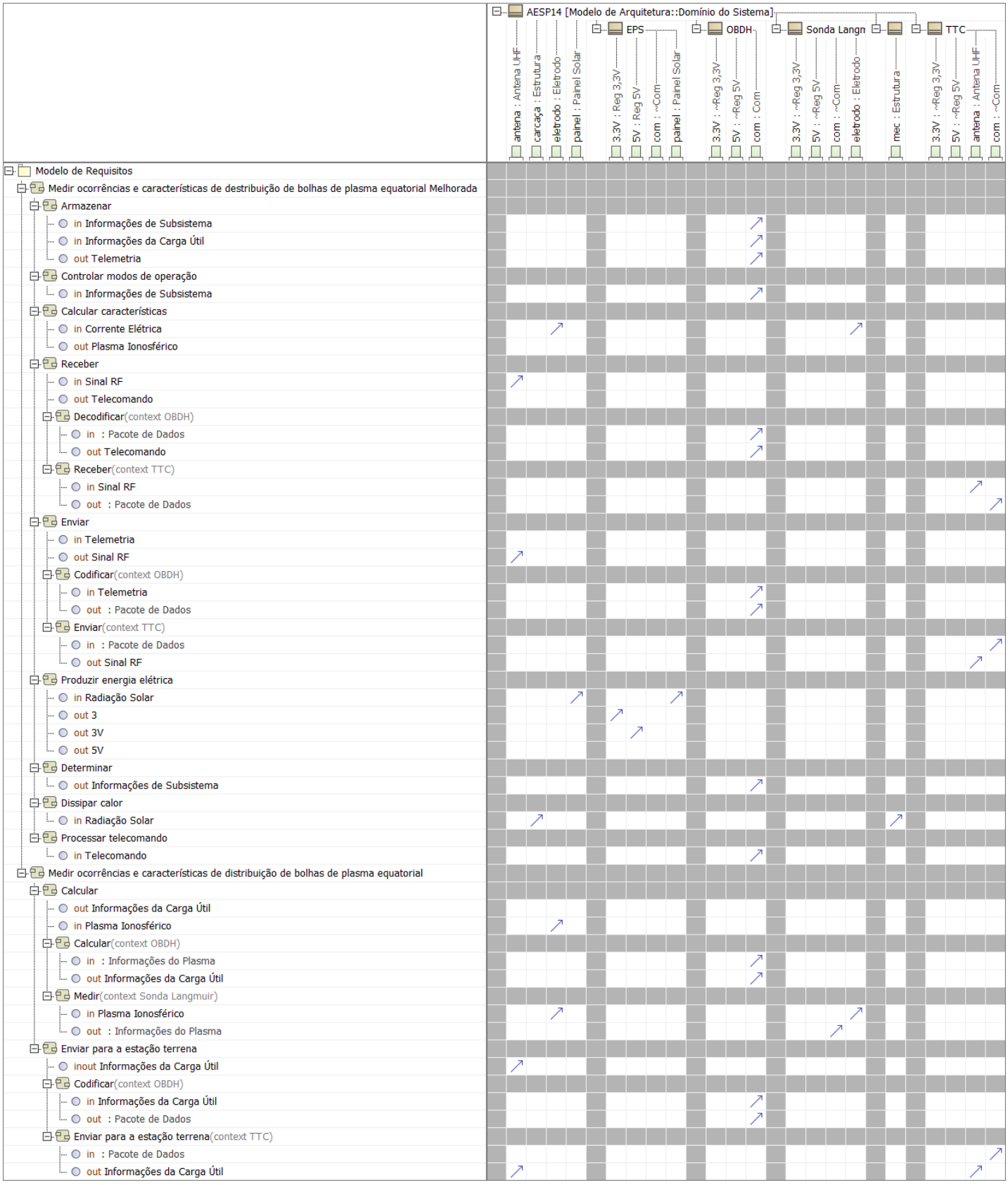
Fonte: Produção do Autor.

No diagrama da figura acima, têm-se novamente as funções do sistema, mas agora com as novas funções identificadas durante projeto detalhado, substituindo aquelas que as originaram. Nota-se que, agora, a alocação funcional é unívoca para os módulos de arquitetura do sistema AESP14.

Finalmente, deve-se fazer a alocação dos parâmetros das funções terminais (essenciais e auxiliares) para as Portas tanto do sistema AESP14 como dos seus módulos de arquitetura. Na Figura D.33 abaixo, mostra-se uma matriz de alocação com esta distribuição, onde nota-se que todos os parâmetros das funções terminais do sistema AESP14 e dos seus módulos de arquitetura tiveram seus parâmetros alocados às correspondentes Portas para garantir observabilidade das respectivas funções. Nota-se, também, que não existe Porta do sistema ou de um de seus módulos de arquitetura que não tenha parâmetro alocado. Isso pode servir como uma verificação antecipada de que nada está faltando ou de que haja excesso de especificação.

Como isso, encerra-se a atividade de relacionamento do modelo de requisito com o modelo de arquitetura do sistema AESP14. Tem-se, portanto, o modelo alocado pronto para extração dos requisitos de sistema, de acordo com as técnicas expostas no Capítulo 4, Seção 4.7.

Figura D.33 – Alocação para Observabilidade das Funções do AESP14.



Fonte: Produção do Autor.

APÊNDICE E – TABELA DE REQUISITOS DO AESP14

Abaixo, tem-se a Tabela E.1 com o conjunto completo de requisitos do sistema AESP14, gerados a partir do modelo SysML criado para evidenciar os resultados apresentados no Capítulo 5 e que foi exposto no Apêndice D.

Tabela E.1 – Requisitos do Sistema AESP14.

Identificação	Enunciado
REQ_AESP14_001	Se [Satélite ejetado] e o [AESP14] estiver no estado [Repouso], o [AESP14] deve mudar para o estado [Condicionamento].
REQ_AESP14_002	Se [30’ após separação] e o [AESP14] estiver no estado [Condicionamento], o [AESP14] deve mudar para o estado [Operacional].
REQ_AESP14_003	Se [Rastreabilidade perdida] e o [AESP14] estiver no estado [Condicionamento], o [AESP14] deve mudar para o estado [Perdido].
REQ_AESP14_004	Se [Rastreabilidade perdida] e o [AESP14] estiver no estado [Operacional], o [AESP14] deve mudar para o estado [Perdido].
REQ_AESP14_005	Se [TC desligar recebido] e o [AESP14] estiver no estado [Operacional], o [AESP14] deve mudar para o estado [Desligado].
REQ_AESP14_006	Se [Bateria esgotada] e o [AESP14] estiver no estado [Perdido], o [AESP14] deve mudar para o estado [Desligado].
REQ_AESP14_007	Se [Nível de bateria baixo] e o [AESP14] estiver no modo operacional [Nominal] do estado [Operacional], o [AESP14] deve mudar para o modo operacional [Seguro] do estado [Operacional].
REQ_AESP14_008	Se [Falha detectada] e o [AESP14] estiver no modo operacional [Seguro] do estado [Operacional], o [AESP14] deve mudar para o modo operacional [Degradado] do estado [Operacional].
REQ_AESP14_009	Se [Nível de bateria normal] e o [AESP14] estiver no modo operacional [Seguro] do estado [Operacional], o [AESP14] deve mudar para o modo operacional [Nominal] do estado [Operacional].

(continua)

Tabela E.1 – Continuação.

Identificação	Enunciado
REQ_AESP14_010	Se [TC reset recebido] e o [AESP14] estiver no modo operacional [Seguro] do estado [Operacional], o [AESP14] deve mudar para o modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_011	Se [Falha detectada] e o [AESP14] estiver no modo operacional [Nominal] do estado [Operacional], o [AESP14] deve mudar para o modo operacional [Degradado] do estado [Operacional].
REQ_AESP14_012	Se [Satélite ejetado] e o [AESP14] estiver no estado [Repouso], o [AESP14] deve [Inicializar satélite] [em não mais do que cinco segundos].
REQ_AESP14_013	Se [30’ após separação] e o [AESP14] estiver no estado [Condicionamento], o [AESP14] deve [Habilitar potência máxima de transmissão RF e Enviar Morse] [em não mais do que dez segundos].
REQ_AESP14_014	Se [TC Desligar recebido] e o [AESP14] estiver no estado [Operacional], o [AESP14] deve [Desligar satélite] [em não mais do que um segundo].
REQ_AESP14_015	Se [Nível de bateria baixo] e o [AESP14] estiver no modo operacional [Nominal] do estado [Operacional], o [AESP14] deve [Desativar subsistemas não críticos e TM de dados] [em não mais do que um segundo].
REQ_AESP14_016	Se [Falha detectada] e o [AESP14] estiver no modo operacional [Seguro] do estado [Operacional], o [AESP14] deve [Desligar o subsistema em falha] [em não mais do que um segundo].
REQ_AESP14_017	Se [Nível de bateria normal] e o [AESP14] estiver no modo operacional [Seguro] do estado [Operacional], o [AESP14] deve [Ativar subsistemas não críticos e TM de dados] [em não mais do que dez segundos].
REQ_AESP14_018	Se [Falha detectada] e o [AESP14] estiver no modo operacional [Nominal] do estado [Operacional], o [AESP14] deve [Desligar o subsistema em falha] [em não mais do que um segundo].
REQ_AESP14_019	Se [TC *reset* recebido] e o [AESP14] estiver no modo operacional [Degradado] do estado [Operacional], o [AESP14] deve [Ligar subsistemas desligados] [em não mais do que dez segundos].

(continua)

Tabela E.1 – Continuação.

Identificação	Enunciado
REQ_AESP14_020	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Armazenar] [Telemetria] usando [Informações da Carga Útil] e [Informações de Subsistema] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_021	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Controlar modos de operação] usando [Informações de Subsistema] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_022	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Enviar] [Sinal RF] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Telemetria] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_023	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Receber] [Telecomando] usando [Sinal RF] de acordo com [Antena UHF] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_024	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Processar telecomando] [periodicamente a cada 500 milissegundos] usando [Telecomando] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_025	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Determinar] [Informações de Subsistema] [periodicamente a cada 500 milissegundos] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_026	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Produzir energia elétrica] [Tensão Regulada] usando [Radiação Solar] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_027	Se [Satélite ejetado] e [30’ após separação], o [AESP14] deve [Dissipar calor] usando [Radiação Solar] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_028	Se [Satélite ejetado] e [30’ após separação] ou [TC *reset* recebido] ou [Nível de bateria normal], o [AESP14] deve [Calcular] [Informações da Carga Útil] usando [Plasma Ionosférico] de acordo com [Eletrodo] enquanto no modo operacional [Nominal] do estado [Operacional].

(continua)

Tabela E.1 – Continuação.

Identificação	Enunciado
REQ_AESP14_029	Se [Satélite ejetado] e [30’ após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [AESP14] deve [Enviar para a estação terrena] [Informações da Carga Útil] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Informações da Carga Útil] enquanto no modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_030	Se [Satélite ejetado] e [30’ após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [AESP14] deve [Calcular características] [Plasma Ionosférico] usando [Corrente Elétrica] de acordo com [Eletrodo] enquanto no modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_031	Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Armazenar] [Telemetria] usando [Informações da Carga Útil] e [Informações de Subsistema] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_032	Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Controlar modos de operação] usando [Informações de Subsistema] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_033	Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Codificar] [Pacote de Dados] de acordo com [Com] usando [Telemetria] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_034	Se [Satélite ejetado] e [30’ após separação], o [TTC] deve [Enviar] [Sinal RF] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Pacote de Dados] de acordo com [Com] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_035	Se [Satélite ejetado] e [30’ após separação], o [TTC] deve [Receber] [Pacote de Dados] de acordo com [Com] usando [Sinal RF] de acordo com [Antena UHF] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_036	Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Decodificar] [Telecomando] de acordo com [Com] usando [Pacote de Dados] de acordo com [Com] enquanto em qualquer modo operacional do estado [Operacional].

(continua)

Tabela E.1 – Continuação.

Identificação	Enunciado
REQ_AESP14_037	Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Processar telecomando] [periodicamente a cada 500 milissegundos] usando [Telecomando] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_038	Se [Satélite ejetado] e [30’ após separação], o [OBDH] deve [Determinar] [Informações de Subsistema] [periodicamente a cada 500 milissegundos] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_039	Se [Satélite ejetado] e [30’ após separação], o [EPS] deve [Produzir energia elétrica] [Tensão Regulada] usando [Radiação Solar] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_040	Se [Satélite ejetado] e [30’ após separação], o [STM] deve [Dissipar calor] usando [Radiação Solar] enquanto em qualquer modo operacional do estado [Operacional].
REQ_AESP14_041	Se [Satélite ejetado] e [30’ após separação] ou [TC *reset* recebido] ou [Nível de bateria normal], a [Sonda Langmuir] deve [Medir] [Informações do Plasma] de acordo com [Com] usando [Plasma Ionosférico] de acordo com [Eletrodo] enquanto no modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_042	Se [Satélite ejetado] e [30’ após separação] ou [TC *reset* recebido] ou [Nível de bateria normal], o [OBDH] deve [Calcular] [Informações da Carga Útil] de acordo com [Com] usando [Informações do Plasma] de acordo com [Com] enquanto no modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_043	Se [Satélite ejetado] e [30’ após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [OBDH] deve [Codificar] [Pacote de Dados] de acordo com [Com] usando [Informações da Carga Útil] de acordo com [Com] enquanto no modo operacional [Nominal] do estado [Operacional].

(continua)

Tabela E.1 – Continuação.

Identificação	Enunciado
REQ_AESP14_044	Se [Satélite ejetado] e [30’ após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [TTC] deve [Enviar para a estação terrena] [Informações da Carga Útil] de acordo com [Antena UHF] [Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)] usando [Pacote de Dados] de acordo com [Com] enquanto no modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_045	Se [Satélite ejetado] e [30’ após separação] ou [TC reset recebido] ou [Nível de bateria normal], o [OBDH] deve [Calcular características] [Plasma Ionosférico] usando [Corrente Elétrica] de acordo com [Eletrodo] enquanto no modo operacional [Nominal] do estado [Operacional].
REQ_AESP14_046	O [AESP14] deve apresentar o estado [Repouso].
REQ_AESP14_047	O [AESP14] deve apresentar o estado [Condicionamento].
REQ_AESP14_048	O [AESP14] deve apresentar o estado [Operacional].
REQ_AESP14_049	O [AESP14] deve apresentar o estado [Perdido].
REQ_AESP14_050	O [AESP14] deve apresentar o estado [Desligado].
REQ_AESP14_051	O [AESP14] deve apresentar o modo operacional [Nominal] no estado [Operacional].
REQ_AESP14_052	O [AESP14] deve apresentar o modo operacional [Seguro] no estado [Operacional].
REQ_AESP14_053	O [AESP14] deve apresentar o modo operacional [Degradado] no estado [Operacional].
REQ_AESP14_054	O [AESP14] deve iniciar no estado [Repouso].
REQ_AESP14_055	O [AESP14] deve iniciar no modo operacional [Nominal] quando entrar no estado [Operacional].
REQ_AESP14_056	O [AESP14] deve ser compatível com a interface [Antena UHF].
REQ_AESP14_057	O [AESP14] deve ser compatível com a interface [Estrutura].
REQ_AESP14_058	O [AESP14] deve ser compatível com a interface [Eletrodo].
REQ_AESP14_059	O [AESP14] deve ser compatível com a interface [Painel Solar].

(continua)

Tabela E.1 – Continuação.

Identificação	Enunciado
REQ_AESP14_060	O [EPS] deve ser compatível com a interface [Reg 3,3V].
REQ_AESP14_061	O [EPS] deve ser compatível com a interface [Reg 5V].
REQ_AESP14_062	O [EPS] deve ser compatível com a interface [Com].
REQ_AESP14_063	O [EPS] deve ser compatível com a interface [Painel Solar].
REQ_AESP14_064	O [OBDH] deve ser compatível com a interface [Reg 3,3V].
REQ_AESP14_065	O [OBDH] deve ser compatível com a interface [Reg 5V].
REQ_AESP14_066	O [OBDH] deve ser compatível com a interface [Com].
REQ_AESP14_067	A [Sonda Langmuir] deve ser compatível com a interface [Reg 3,3V].
REQ_AESP14_068	A [Sonda Langmuir] deve ser compatível com a interface [Reg 5V].
REQ_AESP14_069	A [Sonda Langmuir] deve ser compatível com a interface [Com].
REQ_AESP14_070	A [Sonda Langmuir] deve ser compatível com a interface [Eletrodo].
REQ_AESP14_071	O [STM] deve ser compatível com a interface [Estrutura].
REQ_AESP14_072	O [TTC] deve ser compatível com a interface [Reg 3,3V].
REQ_AESP14_073	O [TTC] deve ser compatível com a interface [Reg 5V].
REQ_AESP14_074	O [TTC] deve ser compatível com a interface [Com].
REQ_AESP14_075	O [TTC] deve ser compatível com a interface [Antena UHF].
REQ_AESP14_076	O [AESP14] deve exibir [Massa = 1 kg ± 10%].
REQ_AESP14_077	O [AESP14] deve exibir [Dimensões = 100 mm ± 10%].
REQ_AESP14_078	O [AESP14] deve exibir [Missão = maior que dois meses] enquanto no [Estado Operacional].
REQ_AESP14_079	O [AESP14] deve exibir [Comunicação = Radiamador] enquanto no [Estado Operacional].
REQ_AESP14_080	O [AESP14] deve exibir [Órbita = Baixa] de acordo com [de 350 a 700 km] enquanto no [Estado Operacional].

(continua)

Tabela E.1 – Conclusão.

Identificação	Enunciado
REQ_AESP14_081	O [AESP14] deve exibir [Proteção Contra Radiação Cósmica] durante exposição ao [Ambiente Espacial] por [Vida Útil].
REQ_AESP14_082	O [AESP14] deve exibir [Temperatura Operacional = [-20ºC, +50ºC]] durante exposição [do Lançamento ao Descarte].
REQ_AESP14_083	O [AESP14] deve exibir [Segurança = ECSS-Q-ST-40] enquanto em [Todos os Estados] por [Vida Útil].
REQ_AESP14_084	O [AESP14] deve exibir [Tolerância a Falhas] com [Recuperação Total] enquanto na [ocorrência de *Single Event Effect*] pela [Vida Útil].

Fonte: Produção do Autor.

APÊNDICE F – COMPARAÇÃO DE REQUISITOS DO AESP14

Na Tabela F.1 abaixo, tem-se a análise sintática de alguns requisitos extraídos de AESP14 (2014) e seus equivalentes requisitos extraídos do modelo SysML do AESP14 elaborado no Capítulo 5. Esta análise sintática foi feita da mesma maneira apresentada no Capítulo 5, Seção 5.4.

Tabela F.1 – Análise sintática dos requisitos.

Halligan (1993)	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
	2.04.004	**REQ_AESP14_081**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	durante exposição ao Ambiente Espacial
Ação	estar protegido contra	Exibir Proteção Contra
Restrições da Ação	---	---
Objeto da Ação	radiação cósmica	Radiação Cósmica
Refinamento do Objeto	---	---
Origem do Objeto	do ambiente espacial de sua órbita	---
Refinamento da Ação	---	---
Destino da Ação	---	---
	2.04.005	**REQ_AESP14_082**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	do Lançamento ao Descarte
Ação	suportar	Exibir
Restrições da Ação	---	---
Objeto da Ação	temperaturas	Temperatura Operacional
Refinamento do Objeto	externas do ambiente espacial de sua órbita	-20ºC, +50ºC
Origem do Objeto	---	---
Refinamento da Ação	---	---
Destino da Ação	---	---

(continua)

Tabela F.1 – Continuação.

Halligan (1993)	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
	2.05.001	**REQ_AESP14_025**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	1) Satélite ejetado e 30' após separação 2) enquanto em qualquer modo operacional do estado Operacional
Ação	realizar	Determinar
Restrições da Ação	---	periodicamente a cada 500 milissegundos
Objeto da Ação	medidas	Informações de Subsistema
Refinamento do Objeto	relativas à saúde de seus subsistemas	---
Origem do Objeto	---	---
Refinamento da Ação	---	---
Destino da Ação	---	---
	2.05.002	**REQ_AESP14_020**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	1) Satélite ejetado e 30' após separação 2) enquanto em qualquer modo operacional do estado Operacional
Ação	armazenar	Armazenar
Restrições da Ação	---	---
Objeto da Ação	medidas	Telemetria
Refinamento do Objeto	relativas à saúde de seus subsistemas	---
Origem do Objeto	---	---
Refinamento da Ação	---	usando Informações da Carga Útil e Informações de Subsistema
Destino da Ação	---	---

(continua)

Tabela F.1 – Continuação.

Halligan (1993)	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
	2.05.003	**REQ_AESP14_022**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	1) Se Satélite ejetado e 30’ após separação 2) enquanto em qualquer modo operacional do estado Operacional
Ação	enviar	Enviar
Restrições da Ação	durante cada conexão com a estação terrena	Indefinidamente com um ciclo de trabalho de 50% (30 segundos ligado, 30 segundos desligado)
Objeto da Ação	informação	Sinal RF
Refinamento do Objeto	relativa à saúde de seus subsistemas	---
Origem do Objeto	---	---
Refinamento da Ação	---	usando Telemetria
Destino da Ação	para o centro de operação	Antena UHF
	2.05.004	**REQ_AESP14_026**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	1) Se Satélite ejetado e 30’ após separação 2) enquanto em qualquer modo operacional do estado Operacional
Ação	adquirir	Produzir energia elétrica
Restrições da Ação	---	---
Objeto da Ação	energia elétrica	Tensão Regulada
Refinamento do Objeto	---	---
Origem do Objeto	---	---
Refinamento da Ação	através de radiação solar	usando Radiação Solar
Destino da Ação	---	---

(continua)

Tabela F.1 – Continuação.

Halligan (1993)	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
	2.07.002	**REQ_AESP14_079**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	enquanto no Estado Operacional
Ação	permanecer	exibir
Restrições da Ação	durante um período não menor de 60 dias	---
Objeto da Ação	operacional	Missão
Refinamento do Objeto	---	maior que dois meses
Origem do Objeto	---	---
Refinamento da Ação	em órbita	---
Destino da Ação	---	---
	2.07.003	**REQ_AESP14_078**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	enquanto no Estado Operacional
Ação	comunicar-se, enviar ou receber	exibir
Restrições da Ação	utilizando uma radiofrequência destinada ao radioamadorismo	---
Objeto da Ação	informação	Comunicação
Refinamento do Objeto	---	Radiamador
Origem do Objeto	---	---
Refinamento da Ação	---	---
Destino da Ação	---	---
	2.07.014	**REQ_AESP14_048**
Ator	o Nanossatélite	o AESP14
Condições da Ação	---	---
Ação	ter	apresentar
Restrições da Ação	---	---

(continua)

Tabela F.1 – Continuação.

Halligan (1993)	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
Objeto da Ação	Estado Operacional	o estado Operacional
Refinamento do Objeto	---	---
Origem do Objeto	---	---
Refinamento da Ação	---	---
Destino da Ação	---	---
	2.07.015	**REQ_AESP14_004**
Ator	o Nanossatélite	o AESP14
Condições da Ação	quando ocorrer a perda de rastreabilidade no Estado Operacional	se Rastreabilidade perdida e o AESP14 estiver no estado Operacional
Ação	mudar	mudar
Restrições da Ação	---	---
Objeto da Ação	Estado	estado
Refinamento do Objeto	---	---
Origem do Objeto	---	---
Refinamento da Ação	---	---
Destino da Ação	Perdido	Perdido
	2.07.024	**REQ_AESP14_007**
Ator	o Nanossatélite	o AESP14
Condições da Ação	quando ocorrer o nível de bateria baixo no Modo Nominal	se Nível de bateria baixo e o AESP14 estiver no modo operacional Nominal do estado Operacional
Ação	mudar	mudar
Restrições da Ação	---	---
Objeto da Ação	Modo	modo operacional
Refinamento do Objeto	---	---
Origem do Objeto	---	---
Refinamento da Ação	---	---

(continua)

Tabela F.1 – Conclusão.

Halligan (1993)	Requisito de AESP14 (2014)	Requisito do Modelo SysML do AESP14
Destino da Ação	Seguro	Seguro do estado Operacional
	2.07.025	**REQ_AESP14_018**
Ator	o Nanossatélite	o AESP14
Condições da Ação	ao mudar do Modo Nominal para o Modo Degradado	se Falha detectada e o AESP14 estiver no modo operacional Nominal do estado Operacional
Ação	Desligar	Desligar
Restrições da Ação	---	em não mais do que um segundo
Objeto da Ação	subsistema	subsistema
Refinamento do Objeto	em falha	em falha
Origem do Objeto	---	---
Refinamento da Ação	---	---
Destino da Ação	---	---

Fonte: Produção do Autor.

Já na Tabela F.2 abaixo, têm-se os valores do indicador de qualidade de requisito (IQR) calculados para os requisitos apresentados na Tabela F.1 acima. A ordem das linhas na Tabela F.2 abaixo segue a ordem na qual os requisitos foram apresentados na Tabela F.1 acima.

Tabela F.2 – Indicador de Qualidade dos Requisitos.

Identificador dos Requisitos	IQR do AESP14 (2014)	IQR do Modelo SysML do AESP14
2.04.004 / REQ_AESP14_081	$\frac{4}{8}\times 100 = 50\,\%$	$\frac{4}{8}\times 100 = 50\,\%$
2.04.005 / REQ_AESP14_082	$\frac{3{,}5}{8}\times 100 = 43{,}75\,\%$	$\frac{4{,}5}{8}\times 100 = 56{,}25\,\%$
2.05.001 / REQ_AESP14_025	$\frac{3{,}5}{8}\times 100 = 43{,}75\,\%$	$\frac{5}{8}\times 100 = 62{,}50\,\%$

(continua)

Tabela F.2 – Conclusão.

Identificador dos Requisitos	IQR do AESP14 (2014)	IQR do Modelo SysML do AESP14
2.05.002 / REQ_AESP14_020	$\frac{3,5}{8}\times 100 = 43,75\%$	$\frac{4,5}{8}\times 100 = 56,25\%$
2.05.004 / REQ_AESP14_026	$\frac{3,5}{8}\times 100 = 43,75\%$	$\frac{4,5}{8}\times 100 = 56,25\%$
2.07.002 / REQ_AESP14_079	$\frac{4,5}{8}\times 100 = 56,25\%$	$\frac{4,5}{8}\times 100 = 56,25\%$
2.07.003 / REQ_AESP14_078	$\frac{4}{8}\times 100 = 50\%$	$\frac{4,5}{8}\times 100 = 56,25\%$
2.07.014 / REQ_AESP14_048	$\frac{3}{8}\times 100 = 37,50\%$	$\frac{3}{8}\times 100 = 37,50\%$
2.07.015 / REQ_AESP14_004	$\frac{5}{8}\times 100 = 62,50\%$	$\frac{5}{8}\times 100 = 62,50\%$
2.07.024 / REQ_AESP14_007	$\frac{5}{8}\times 100 = 62,50\%$	$\frac{5}{8}\times 100 = 62,50\%$
2.07.025 / REQ_AESP14_018	$\frac{4,5}{8}\times 100 = 56,25\%$	$\frac{5,5}{8}\times 100 = 68,75\%$

Fonte: Produção do Autor.