UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO - CTC
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Diego Câmara Sales

Abordagem para Evolução da Arquitetura de Sistemas Embarcados com uso Intenso de Sensores e Atuadores

Florianópolis
2022

Diego Câmara Sales

Abordagem para Evolução da Arquitetura de Sistemas Embarcados com uso Intenso de Sensores e Atuadores

Tese submetida ao programa de pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de doutor em Automação e Sistemas.
Orientador: Prof. Leandro Buss Becker, Dr.
Coorientador: Prof. Cristian Koliver, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Sales, Diego Câmara
Abordagem para Evolução da Arquitetura de Sistemas Embarcados com uso Intenso de Sensores e Atuadores / Diego Câmara Sales ; orientador, Leandro Buss Becker, coorientador, Cristian Koliver, 2021.
112 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Projeto de arquitetura de sistemas. 3. evolução de arquiteturas. 4. Troca de sensores a atuadores. 5. Ontologia de arquitetura de sistemas. I. Buss Becker, Leandro . II. Koliver, Cristian. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Diego Câmara Sales

Abordagem para Evolução da Arquitetura de Sistemas Embarcados com uso Intenso de Sensores e Atuadores

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Raimundo Barreto, Dr.
Instituição UFAM - IComp

Prof. Eduardo Augusto Bezerra, Dr.
Instituição UFSC - EEL

Prof. Rodrigo Castelan Carlson, Dr.
Instituição UFSC - DAS

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de doutor em Automação e Sistemas.

Documento assinado digitalmente
Werner Kraus Junior
Data: 18/01/2022 18:06:18-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Programa de
Pós-Graduação

Documento assinado digitalmente
Leandro Buss Becker
Data: 18/01/2022 22:12:00-0300
CPF: 688.357.870-68
Verifique as assinaturas em https://v.ufsc.br

Prof. Leandro Buss Becker, Dr.
Orientador

Florianópolis, 2022.

Uma pequena gota no oceano da ciência. Dedico este trabalho à comunidade científica, em especial aos engenheiros de sistemas, que, com amor e dedicação ao próximo, compartilham suas ideias e pesquisas aplicadas, auxiliando no lapidar constante da evolução tecnológica em favor de um mundo melhor.

AGRADECIMENTOS

Agradeço a Deus por conceder saúde, resiliência e conhecimento em minha caminhada. Aos meus familiares, mãe, Maria da Glória Câmara, e pai, José Maria Sales Pinheiro, pelo amor incondicional dado a mim, na busca de ver seu filho formado. A minha esposa, Kelly Vinente dos Santos, por sua paciência e amor ao me auxiliar nos momentos difíceis do meu trabalho, realizando sonhos e momentos felizes. Agradeço ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PPGEAS) por fornecer as condições necessárias ao meu desenvolvimento enquanto pesquisador, ao meu supervisor Leandro Buss Becker por acreditar e me apoiar durante as etapas de desenvolvimento deste trabalho. Ao grupo de pesquisadores do projeto de veículos aéreos não tripulados ProVant que de alguma forma enriqueceram meu limitado ponto de vista quanto ao domínio aeroespacial. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. A Fundação de amparo a Pesquisa do Estado do Amazonas (FAPEAM) e Instituto Federal do Amazonas (IFAM) pela oportunidade de crescimento profissional em busca de disseminar o que aprendi aos meus conterrâneos, instituição e futuros alunos.

“É preciso que o discípulo da sabedoria tenha o coração grande e corajoso.
O fardo é pesado e a viagem longa”
(Confúcio em Os analectos, 551 e 479 a.C)

RESUMO

A constante evolução dos dispositivos de sensoriamento e atuação (S&A) faz com que os projetistas avaliem potenciais modificações de projeto da Arquitetura de Sistemas (AS) de Cyber Physical System (CPS) durante o seu ciclo de vida. Estes dispositivos desempenham um papel crítico, pois são os mecanismos pelos quais o software (ciber) interage com o mundo físico. A troca ou inclusão destes dispositivos na arquitetura é uma atividade complexa composta de um conjunto de etapas dedicadas à modelagem das características, propriedades e requisitos do sistema. Desta forma, detalhar as etapas e atividades de desenvolvimento do projeto pode auxiliar a equipe de projetistas durante a representação e gerenciamento dos recursos da arquitetura. Entretanto, algumas etapas são mais discutidas na comunidade científica, tais como a modelagem e análise das características da arquitetura, e outras carecem de mais estudos, como por exemplo a exploração de cenários de troca dos dispositivos de S&A. Neste contexto, é necessário que os projetistas tenham experiência para realizar as atividades que contemplem a seleção, exploração e análise de compatibilidade desses dispositivos, uma vez que, a ausência de informações dificulta o desenvolvimento das etapas de projeto. Existem diferentes abordagens na literatura que buscam fornecer um conjunto de etapas e atividades de suporte ao desenvolvimento de CPS. Entretanto, elas não descrevem em detalhes as atividades e ferramentas de suporte ao fluxo de etapas de projeto. Por conta disso, esta tese apresenta uma abordagem que busca contribuir com o processo de troca de dispositivos de S&A da arquitetura, e consequentemente guiar os projetistas na realização das atividades de avaliação e e análise de características. A Engenharia Dirigida por Modelos (*Model Driven Engineering* - MDE) é utilizada como base da abordagem proposta e fornece suporte à modelagem arquitetural através de Linguagens de Descrição de Arquiteturas (LDA). Permitindo, desta forma, a representação dos componentes através de ontologia e da exploração de dispositivos candidatos, incluindo a análise inicial e a posterior seleção do cenário evoluído. Além disso, optou-se por utilizar duas abordagens apoiadas por ferramentas que foram desenvolvidas ao longo desta tese: *OWL2AADL* e *DevCompatibility*. A abordagem *OWL2AADL* permite a transformação de modelos de ontologia do domínio de arquitetura de sistemas, descritos em Ontology Web Language (OWL), para modelos arquiteturais Architecture Analysis and Design Language (AADL). Já a abordagem *DevCompatibility* fornece suporte automatizado às atividades de exploração, análise e ranqueamento de cenários da abordagem de evolução de arquiteturas. Para demonstrar a aplicabilidade da proposta, a abordagem e ferramentas desenvolvidas são aplicadas ao projeto de um Veículo Aéreo Não Tripulado (VANT) do tipo tilt-rotor. É importante ressaltar que os detalhes da abordagem proposta são demonstrados no processo de evolução da arquitetura de um VANT em questão.

Palavras-chave: Projeto de arquitetura de sistemas. Evolução de arquiteturas. Troca de sensores a atuadores. Ontologia de Arquitetura de Sistemas.

ABSTRACT

The constant evolution of sensing and actuation devices (S&A) causes designers to evaluate potential modifications in the operational architecture of CPS during the project lifecycle. These devices are the mechanisms by which (cyber) software interacts with the physical world. The exchange or inclusion of these devices in the architecture is a complex activity, composed of a set of phases dedicated to the modeling of characteristics, and properties and requirements of the system. This way, detailing the phases and the development of the project activities can be of help to the team of designers, during the representation and management of architectural resources. Meanwhile, some phases are more discussed in the scientific community, such as modeling and analysis of architectural characteristics. Others demand more studies, as for example, the exploitation of S&A device exchange scenarios. In this context, it is necessary that designers have experience in performing these activities of selection and, exploitation and analysis of these devices, where the absence of information makes it difficult to the development of the project phases. There are different approaches in literature, aiming to supply a set of phases and support activities to the development of CPS. However, they do not describe in detail the activities and support tools to the step steam of the project. Thereby, this thesis presents an approach that aims to contribute with the S&A device exchange process, and consequently guide the designers through the accomplishment of these activities. MDE - Model Driven Engineering is used as a basis to the proposed approach and provides support to architectural modeling, through Architectural Description Languages (ADL), and components representation through ontology and exploitation of candidate devices, including the analysis and selection of evolved scenario. Two approaches and tools were developed - *OWL2AADL* and DevCompatibility to support the proposed approach activities. *OWL2AADL* provides the transformation of ontology models of the domain of systems architecture described in OWLto the AADL architectural models. Meanwhile, the *DevCompatibility* provides automatized support to the activities of exploitation, and analysis and ranking of scenarios. The proposed approach and tools are applied to the Unmanned Aerial Vehicle (UAV) tilt-rotor type project. The proposed approach details are demonstrated in the evolving architectural process of a UAV, described in the study case.

Keywords: architecture system design. Architecture evolution. sensor and actuators devices tradeoff. Ontology of architecture system.

LISTA DE FIGURAS

Figura 1 – Conceitos de cyber-physical system. 15
Figura 2 – VANT e seus sensores. 16
Figura 3 – Visão geral do processo de MDE. 26
Figura 4 – Visão geral do processo de transformação de modelos. 28
Figura 5 – As camadas da web semântica. 31
Figura 6 – Componentes AADL 34
Figura 7 – Componentes AADL e relacionamentos. 35
Figura 8 – Estrutura de etapas da abordagem proposta. 57
Figura 9 – modelo CAD do ProVant 1.0. 65
Figura 10 – Exemplo de PEA do ProVant 4.0. 66
Figura 11 – Instância dos componentes de hardware em OWL. 68
Figura 12 – Metodologia proposta no projeto da OAS. 75
Figura 13 – Exemplo de mapeamento do reuso e componentes da OAS. 78
Figura 14 – Ontologia proposta integrando as classes SOSA. 80
Figura 15 – Conjunto de classes da OAS com reuso e módulos integrados. 81
Figura 16 – Classes reutilizadas da SCM. 82
Figura 17 – Relacionamento das seções e tipos de componentes. 83
Figura 18 – Relacionamentos dos componentes da implementados. 84
Figura 19 – Declaração de propriedades e tipo de dados usados na SA. 85
Figura 20 – Visão geral da ferramenta OWL para AADL. 89
Figura 21 – Processo do motor de transformação de modelos. 89
Figura 22 – Fragmento do metamodelo RDFS e OWL. 90
Figura 23 – Metamodelo AADL 90
Figura 24 – Modelagem em OWL da propriedade do dado Mass. 94
Figura 25 – Definições do indivíduo AXI2830. 94
Figura 26 – Estrutura da abordagem de troca de S&A. 100
Figura 27 – Interface gráfica de seleção dos modelos. 101
Figura 28 – Seleção do sistema ProVant 4.0 para troca de S&A. 101
Figura 29 – Tela de apresentação dos cenários. 102
Figura 30 – Seleção do tipo de exploração de projeto de arquitetura. 102
Figura 31 – Estrutura do analisador de modelo ReqSpec. 104
Figura 32 – Estrutura do analisador de modelo AADL. 105
Figura 33 – Fluxo de atividades da exploração de arquiteturas. 106
Figura 34 – Seleção dos dispositivos a serem trocados. 107
Figura 35 – Conjunto de funcionalidades e relações com dispositivos. 108
Figura 36 – Conexões entre dispositivos e sistema. 109
Figura 37 – Lista dos barramentos e CPU disponíveis. 111

Figura 38 – lista dos cenários identificados com interface compatível. 112
Figura 39 – Lista de possíveis cenários compatíveis criados. 113
Figura 40 – Atributos e pesos do ProVant 4.0. 114
Figura 41 – Definição das estratégias de ranqueamento dos cenários. 115
Figura 42 – Definição de pesos e cálculo do ranqueamento de cenários. 117
Figura 43 – Detalhes das modificações realizadas em cada cenário do ranqueamento. 118
Figura 44 – Análise comparativa do cenário atual e selecionado. 118
Figura 45 – Modelo AADL do cenário 6, gerado na *DevCompatibility*. 119

LISTA DE TABELAS

Tabela 1 – Visão geral dos trabalhos relacionados. 44
Tabela 2 – Visão geral das ontologias levantadas 48
Tabela 3 – Visão geral das abordagens pesquisadas. 55
Tabela 4 – Componentes de hardware do ProVANT 4.0. 67
Tabela 5 – Ontologias candidatas ao reuso. 77
Tabela 6 – Mapeamento das classes SOSA e componentes da OAS. 79
Tabela 7 – Tipos de componentes da arquitetura 80
Tabela 8 – Exemplo de propriedades do objeto da OAS. 82
Tabela 9 – Propriedades de dados do domínio aeroespacial. 86
Tabela 10 – Entidades OWL e relacionamentos de componentes AADL. 91
Tabela 11 – Conjunto de propriedades de dados de suporte ao alinhamento. . . 92
Tabela 12 – Alinhamento entre os dados OWL e AADL. 92
Tabela 13 – Lista de atributos extraídos dos cenários e sistema selecionado. . . 113

LISTA DE ABREVIATURAS E SIGLAS

AADL	Architecture Analysis and Design Language
ADL	Architectural Description Languages
AHP	Analytical Hierarchy Process
AS	Arquitetura de Sistemas
CAvA	CPS Architecture Evolution Approach
CO	Communication Overhead
COTS	Commercial Off-The-Shelf
CPS	Cyber Physical System
DSML	Domain-specific Modeling Languages
DTR	Data Transmission Reliability
EMF	Eclipse Modeling Framework
FCS	Flight Control System
GATSE	Guided Architecture Trade Space Explorer
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
LDA	Linguagens de Descrição de Arquiteturas
M2C	Modelo para Código
M2M	Modelo para Modelo
M2T	Modelo para Texto
MDA	Model-Driven Architecture
OAS	Ontologia de Arquitetura de Sistemas
OMG	Object Mana-gement Group
OSATE	Open Source AADL Tool Environment
OWL	Ontology Web Language
PIM	Platform Independent Model
PSM	Platform-Specifi Models
RDF	Resource Description Framework
ReqSpec	Requirement Specification Language
SA	Sensoriamento e Atuação
SAE	Sociedade automotiva de engenheiros
SEI	Software Engineering Institute
SO	Sistemas Operacionais
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
VANT	Veículo Aéreo Não Tripulado
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO 15
1.1 PROBLEMAS IDENTIFICADOS: 17
1.2 MOTIVAÇÃO 18
1.3 RESUMO DA ABORDAGEM PROPOSTA 20
1.4 OBJETIVOS 21
1.5 ORGANIZAÇÃO DA TESE 22
1.6 LISTA DE PUBLICAÇÕES 22
2 REFERENCIAL TEÓRICO 24
2.1 ENGENHARIA DIRIGIDA A MODELOS (MDE) 24
2.1.1 Modelos e Metamodelos 26
2.1.2 Transformação de modelos 27
2.2 ONTOLOGIA 29
2.2.1 Web Semântica e tecnologias aplicadas 30
2.3 FERRAMENTAS E LINGUAGENS 30
2.3.1 Ontology web language (OWL) 31
2.3.2 Linguagem de regras da web semântica 32
2.3.3 Protegé 33
2.3.4 Architectural Analysis Design Language (AADL) 33
2.3.5 OSATE 2 37
2.4 SUMÁRIO 38
3 TRABALHOS CORRELATOS 39
3.1 MÉTODOS DE PROJETO DE CPS 39
3.2 ONTOLOGIA APLICADA À ARQUITETURA DE SISTEMAS 45
3.2.1 Ontologia: Sensores e atuadores 45
3.2.2 Ontologia: Plataformas computacionais 46
3.2.3 Ontologia: Robótica e automação 47
3.3 ABORDAGEM DE TROCA DE S&A EM PROJETO DE CPS 49
4 ABORDAGEM DE EVOLUÇÃO DE ARQUITETURAS DE CPS . . . 56
4.1 ATIVIDADES PARA EVOLUÇÃO DA ARQUITETURA DE SISTEMAS 57
4.2 PROJETO DE EVOLUÇÃO DE UM VANT-CP 64
4.3 SUMÁRIO 72
5 ONTOLOGIA DE ARQUITETURA DE SISTEMAS 74
5.1 METODOLOGIA DE PROJETO DA OAS 75
5.2 TRANSFORMAÇÃO DE MODELOS OWL PARA AADL 88
5.2.1 Motor de transformação de modelos OWL para AADL 89
5.3 FERRAMENTA DE TRANSFORMAÇÃO DE MODELOS OWL2AADL 93
5.4 SUMÁRIO 97

6 EXPLORAÇÃO, ANÁLISE E OTIMIZAÇÃO DA ARQUITETURA . . 98
6.1 CONSIDERAÇÕES INICIAIS 98
6.2 ABORDAGEM DE TROCA S&A EM MODELOS AADL 99
6.2.1 Etapa de requisitos do Sistema 103
6.2.2 Etapa de Exploração de Arquiteturas 104
6.2.3 Etapa de Otimização de cenários 114
6.3 SUMÁRIO 119
7 CONSIDERAÇÕES FINAIS E ATIVIDADE FUTURAS 121
REFERÊNCIAS 126

1 INTRODUÇÃO

Os sistemas embarcados, em sua grande maioria, são caracterizados por realizar a interação com o processo físico. Tais sistemas são chamados de CPS, devido à caracterização da integração entre o processo físico, computacional e monitores de rede. Tipicamente, CPS utilizam dispositivos de Sensoriamento e Atuação (SA) no controle do processo físico (BALABAN *et al.*, 2009; CHANDHOKE *et al.*, 2011).

Resumidamente, a estrutura do CPS é composta de três partes principais, conforme apresentado na Figura 1. A primeira é a planta física, que é a parte ***physical*** do sistema que inclui partes mecânicas, biológicas ou processo químicos, ou operadores humanos. A segunda parte, ***platform*** é composta de uma ou mais plataformas computacionais, armazenamento de dados, dispositivos de S&A e um ou mais Sistemas Operacionais (SO). A terceira parte é a ***rede***, na qual fornece os mecanismos de comunicação dos computadores. A intersecção das três partes forma o CPS e a junção da ***platform*** com ***rede*** forma o ***cyber*** (LEE, E.; SESHIA, 2016).

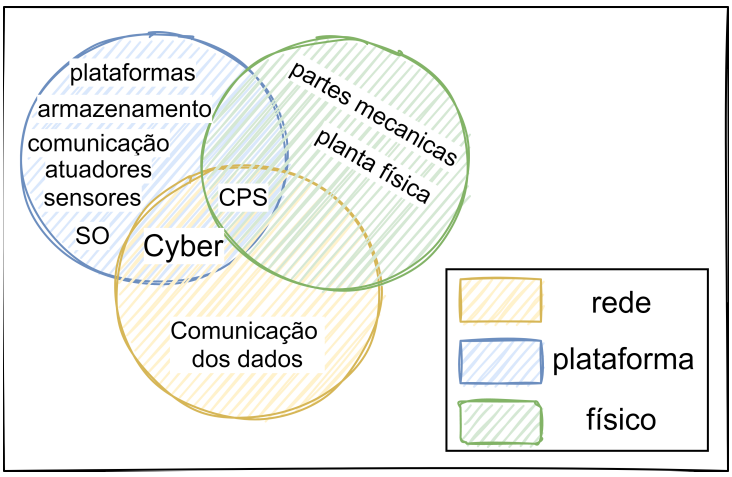


Figura 1 – Conceitos de cyber-physical system.
Fonte: Figura do autor.

Os dispositivos de S&A são componentes importantes na arquitetura de CPS devido a sua capacidade fornecer informações/dados do ambiente/planta (físico) ao software (ciber) que efetua o controle enviando sinais elétricos aos atuadores para realização do movimento mecânico. Com a evolução tecnológica, estes dispositivos expandiram sua aplicação (agricultura, automotivo, militar) (REGTIEN *et al.*, 2018), novas funcionalidades e integração de componentes de software.

A evolução da arquitetura, a partir da troca destes dispositivos, possibilita a melhoria de características de qualidade do software como desempenho, confiabilidade, segurança e adequação funcional, possibilitando a redistribuição de recursos e definição de novos ajustes, funcionalidades e requisitos em busca de estender o ciclo de vida do sistema e cumprimento de novos requisitos (ALVES, C. F.; FILHO *et al.*, 2001).

Em relação às aplicações voltadas para o ambiente aeroespacial, foco desta pesquisa, destaca-se a evolução tecnológica e a necessidade de atualização dos dispositivos de S&A utilizados em VANTs. Os VANTs são exemplos típicos de CPS que utilizam intensivamente os dispositivos de S&A, sendo composto por diversos subsistemas e elementos que operam de forma sincronizada, estruturada e interconectada. Os VANTs são utilizados em aplicações civis e militares, devido a sua flexibilidade para se integrar a diferentes ambientes e possuem a capacidade de serem operados remotamente, transmitindo informações em tempo real para uma estação base. Estes veículos possuem potencial para realizar diversos tipos de missões, de acordo com suas características e configurações, dentre elas: transporte de carga, busca e resgate, agricultura e vigilância (GONCALVES, F., 2018a).

Basicamente, um VANT é automaticamente pilotado por uma plataforma computacional chamado de sistema de controle de voo Flight Control System (FCS), que lê informações de uma ampla variedade de sensores (acelerômetros, giroscópios, GPS) e conduz a missão do VANT ao longo de um plano de voo predeterminado (TSOURDOS, 2000).

De acordo com o tipo de missão, o VANT pode acoplar ao seu FCS um sistema de carga útil, que habilita a realização de aplicações com o uso de dispositivos de S&A. A Figura 2 apresenta um exemplo de VANT com um conjunto de sensores, FCS, controle de missão e carga útil (*mission and payload control*), sistema de comunicação (*communication system*) com a estação base (*UAV base station*) e carga útil (câmeras e estabilizador giroscópico).

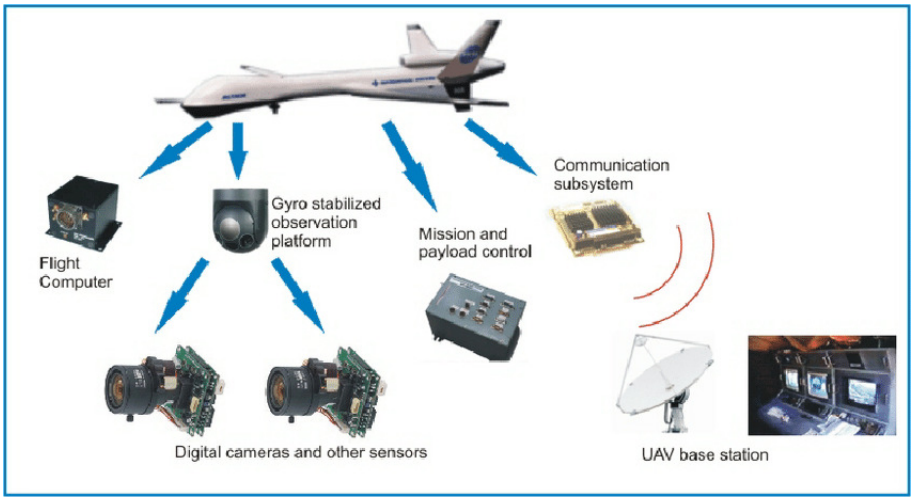


Figura 2 – VANT e seus sensores.
Fonte:(PASTOR; LOPEZ; ROYO *et al.*, 2006).

O uso intenso de S&A na arquitetura do sistema demonstra a complexidade de se realizar modificações ou atualizações, pois estes componentes são considerados críticos e uma falha deles pode desencadear a incidência de eventos catastróficos (BALABAN *et al.*, 2009). Desta forma, devido a fatores como obsolescência, custos de

projeto, inclusão de novas funcionalidades, melhora de desempenho e segurança, a troca dos dispositivos de S&A da arquitetura durante o ciclo de vida do sistema, sem uma análise de impacto, pode acarretar em problemas do tipo estruturais quanto à compatibilidade entre os componentes do sistema, de cumprimento dos requisitos e desempenho.

Além disso, nota-se que a mudança de um componente da arquitetura do sistema sem a utilização de uma abordagem previamente definida e estruturada pode prejudicar alguns aspectos de qualidade da arquitetura. Para decidir como implementar as mudanças, é preciso usar um método que permita identificar e avaliar as características e propriedades arquitetônicas em busca de manter a coerência da arquitetural, maximizando a viabilidade do software e minimizando os custos da plataforma computacional do sistema.

Neste contexto, diferentes abordagens foram propostas com objetivo de fornecer diretrizes às equipes de projetistas. Algumas dessas abordagens são baseadas em MDE, a fim de apoiar a representação das características e comportamento dos CPS (SCHMIDT, 2006).

A MDE possibilita a criação de modelos complementares que representam as diferentes domínios, auxiliando a avaliar diversas características de funcionamento e comportamento do sistema nos mais diferentes níveis de abstração (BECKER, L. *et al.*, 2010; LEE, E.; SESHIA, 2016), facilitando a interação entre diferentes linguagens e modelos usados ao longo do ciclo de projeto (KORDON, Fabrice *et al.*, 2013).

Outra característica da MDE é a capacidade de transformação de modelos para automatização de tarefas. Desse modo, modelos abstratos de diferentes linguagens podem prover dados para geração de modelos complementares até a implementação da aplicação em código fonte enviado a plataforma computacional.

1.1 PROBLEMAS IDENTIFICADOS:

Neste contexto, e considerando o desenvolvimento de VANTs, diferentes métodos foram propostos com o objetivo de fornecer uma orientação para as equipes de design para construção da AS e, ao menos, três dificuldades foram observadas durante as etapas, atividades de desenvolvimento e evolução da arquitetura de CPS.

A primeira demonstrou que as etapas e atividades pouco abordam o conceito de evolução de arquitetura de arquiteturas em operação. O tópico de troca, modificações e desenvolvimento de plataformas computacionais são amplamente difundidos na comunidade científica, onde os estudos de troca de dispositivos de S&A são menos discutidos.

O segundo problema foi quanto à disponibilidade de ferramentas de suporte ao desenvolvimento das atividades de evolução de arquiteturas que contemplem desde a modelagem dos requisitos do projeto, análise e geração da arquitetura evoluída. In-

tegrar ferramentas de diferentes etapas e pontos de vista de desenvolvimento da AS é um problema causado devido às dificuldades de mapeamento de modelos, alinhamento de conceitos e uso de vocabulário comum, o que dificulta a interação entre os projetistas de diferentes áreas.

O terceiro problema está relacionado às dificuldades de representar, de forma semântica e abstrata, os componentes da arquitetura de um sistema, suas características e propriedades, assim como os relacionamentos dos componentes. Ontologias de domínios semelhantes ao de AS , tais como internet das coisas, redes de sensores, robótica e automação não representam os componentes arquiteturais necessários para implementação da AS em projetos de CPS.

1.2 MOTIVAÇÃO

Ao longo dos anos tive paixão por carros antigos, em especial dos anos 70, e senti grandes dificuldades para realizar modificações de projeto em busca de melhorar o entretenimento e a visualização das informações de velocidade, motorização, medição de combustível e bateria do meu Puma GTE do ano de 1974. A troca dos medidores por display sensível ao toque resultou em incompatibilidade de leitura dos sensores de temperatura, tacógrafo e combustível, que usam tecnologias diferentes, e apresentaram incompatibilidade de interface de hardware e software. Quanto a troca da motorização, o motor original possui refrigeração a ar e o novo motor possuía refrigeração a água. Esta mudança impactou diretamente nos sensores e estrutura mecânica do automóvel, apresentando problemas de fidelidade das informações dos sensores e super aquecimento do sistema.

Diante destes problemas, que despertaram o meu interesse na busca de entender como ocorre o processo de evolução de um sistema a partir da troca de seus componentes, busquei estudar e compreender quais etapas e atividades eu deveriam ser executadas, na busca de realizar modificações de projeto antigos para integrar tecnologias atuais, mantendo a compatibilidade e fazendo proveito de novas funcionalidades e aprimoramentos de desempenho do CPS.

O desenvolvimento de um CPS perpassa por um conjunto de etapas e atividades que detalham a representação e o relacionamento dos componentes que compõem a arquitetura em busca de avaliar, por meio de análises, simulações e testes, o cumprimento dos requisitos de projeto (**gonccalves2016model**). Neste contexto, verificou-se que durante o ciclo de vida do projeto é comum a necessidade de atualizar os componentes da arquitetura em virtude de um conjunto de fatores, tais como: obsolescência, avanço tecnológico e mudanças na aplicação que impactam na atualização dos requisitos (WASSON, 2015; IEEE, 2017).

Assim, diferentes metodologias foram propostas para o projeto de arquitetura de CPS com objetivo de guiar os projetistas durante o desenvolvimento do projeto, tal

como o processo de desenvolvimento para sistemas embarcados críticos (BECKER, L. *et al.*, 2010), metodologia de projeto de CPS baseado em modelos (JENSEN; CHANG LEE *et al.*, 2011), e padrões internacionais IEEE que detalham o processo de ciclo de vida de sistemas (SALES, Diego Câmara; BECKER, Leandro Buss, 2018). Entretanto, a evolução ou modificação da arquitetura a partir da troca de componentes foram pouco discutidos, deixando a responsabilidade para os projetistas realizarem modificações baseados em suas experiências em projetos anteriores, sujeito a erros, dificuldades de comunicação, dentre outros (BOEHM; TURNER, 2015).

Neste contexto, as abordagens de desenvolvimento de CPS's, baseadas em MDE, auxiliam na descrição das atividades a serem realizadas no projeto da arquitetura. As ferramentas de engenharia de software auxiliada por computador apoiam a aplicação das abordagens fornecendo suporte a realização das etapas e das atividades de construção do modelo arquitetural. Estas ferramentas possibilitam a realização de análises, simulações, testes e validação das características, tais como Ptolemy (LEE, E. A.; JOHN, 1999) e Papyrus (LANUSSE *et al.*, 2009), que auxiliam os projetistas desde a modelagem dos requisitos até a geração do código fonte. Entretanto, estas ferramentas não apresentam suporte à representação de aspectos da arquitetura importantes na análise de troca dos dispositivos de S&A, tal como a interface de hardware e software, parâmetros comportamentais e funcionais ligados à arquitetura.

Uma das formas de prover a representação destes aspectos é por meio de LDA. Dentre as LDA, a AADL (AS5506, 2004) é amplamente utilizada na comunidade científica devido à capacidade de descrever a arquitetura de software, plataformas de execução, propriedades e interface dos componentes aplicados em sistemas de tempo real. Adicionalmente, a AADL fornece um conjunto de propriedades da linguagem para análise de diferentes características da arquitetura, tais como latência (YU; YANG, 2012), comportamento do sistema (ZHANG *et al.*, 2017) e modelagem de erros (DELANGE; FEILER, P., 2014).

Entretanto, alguns desafios foram encontrados durante a modelagem e análise de troca dos dispositivos de S&A da arquitetura descritos em AADL, demonstrando ser um trabalho exaustivo e sujeito a erros, devido a complexidade de modelar manualmente, em baixo nível de abstração, a extensa biblioteca de candidatos, e cenários (PROCTER; L.WRAGE, 2019).

Atualmente, abordagens baseadas em ontologia são aplicadas na modelagem de dados e na representação semântica de conceitos, relacionamentos e características de um determinado domínio, tal como automação e robótica (OLSZEWSKA *et al.*, 2017), internet das coisas (MARTÍN-LAMMERDING *et al.*, 2020) e rede de sensores (HALLER *et al.*, 2017; JANOWICZ *et al.*, 2019). Dentre os benefícios desta abordagem, temos: o alto nível de abstração, reuso de informações, facilidade de compartilhamento de conhecimentos entre os domínios e vocabulário comum (NOY;

MCGUINNESS *et al.*, 2001). Entretanto, as ontologias citadas não apresentam suporte suficiente para representar a arquitetura de sistemas e seus aspectos, demonstrando a necessidade de pesquisar a respeito do desenvolvimento de uma ontologia aplicada à modelagem da arquitetura.

Com relação ao projeto da arquitetura CPS, as abordagens MDE e LDA são utilizadas durante as atividades ou etapas que visam representar a AS. A LDA ajuda a promover a comunicação mútua e permite a análise antecipada e o teste de viabilidade de decisões de projeto arquitetônico (CLEMENTS, P. C., 1996). Sendo assim, percebeu-se que desenvolver uma ontologia para desenvolvimento de AS baseada em LDA pode ajuda a representar conceitos comuns, alinhando vocabulários e conhecimentos.

Outro desafio observado em diferentes abordagens levantadas é o fato de durante o desenvolvimento das etapas e atividades do projeto, diferentes modelos serem criados para representar pontos de vista complementares da arquitetura com níveis de abstração distintos, porém não possuem integração entre si, e geralmente são desenvolvidos manualmente. Com isso, os projetistas realizam um esforço adicional para representar as características da arquitetura em diferentes modelos, assim como a transição entre modelos ou geração do código fonte da aplicação (GONCALVES, F., 2018b).

A integração manual entre os modelos se mostra como um trabalho exaustivo e sujeito a falhas técnicas que executam processos de transformação de modelos automatizado que podem ser aplicadas para apoiar a geração de modelos, ou código fonte da aplicação, e em diferentes etapas e atividades de desenvolvimento da arquitetura. São exemplos, a técnica de transformar requisitos e análises preliminares de projeto representados em UML-MARTE para modelos AADL (BRUN *et al.*, 2008), a proposta *Assisted Models Transformation* (AST) (PASSARINI *et al.*, 2015) que permite gerar modelos arquiteturais AADL a partir de modelos funcionais Simulink, e, mais recentemente, a técnica guiada de integração de sensores e atuadores provenientes de modelos funcionais Simulink para modelo arquitetural AADL (GONCALVES, F., 2018b).

1.3 RESUMO DA ABORDAGEM PROPOSTA

Diante das problemáticas e desafios encontrados foi proposto nesta tese um método de evolução de AS em CPS desde a da troca de componentes de S&A denominado CAvA. Constituído por um conjunto de etapas e atividades que são descritas a começar do levantamento de requisitos, perpassando pelo levantamento de candidatos e as análises arquitetônicas, até a consolidação da arquitetura evoluída. Sendo assim, a abordagem CAvA busca guiar os projetistas no processo de evolução de AS a partir da troca de componentes da arquitetura, com o objetivo de detalhar a representação de AS em CPS, também foi proposta uma ontologia para representar os relacionamen-

tos entre os componentes da AS e o processo físico de CPS, possibilitando a criação de um modelo ontológico OWL que pode auxiliar na avaliação de inconsistências de projeto e uso de regras semânticas para identificar o impacto gerado por modificações nos componentes da arquitetura.

A base de conhecimento gerada na OAS fornece informações utilizadas na representação de modelos arquitetônicos em AADL, onde, através de um processo de transformação de modelos OWL para AADL foi possível gerar automaticamente um conjunto de arquivos utilizados no desenvolvimento das etapas e atividades da abordagem de evolução arquiteturas de CPS.

Uma abordagem que contempla as etapas de exploração de cenários candidatos, análise e otimização de arquiteturas baseada em MDE com uso dos modelos arquiteturais AADL gerados e modelo de requisitos em linguagem ReqSpec é apresentada, demonstrando a aplicação do método proposto e ferramentas de suporte desenvolvidas. Portanto, foram realizados experimentos aplicando a abordagem a partir do modelo OWL que representa um CPS do domínio aeroespacial, detalhando tanto os componentes da AS e de seus relacionamentos quanto o processo físico da aplicação. A partir do modelo OWL criado foi aplicado um processo de transformação dos modelos OWL para AADL, gerando um conjunto de modelos AADL referentes ao sistema implementado que deseja-se evoluir, biblioteca de componentes candidatos e conjunto de propriedades do sistema utilizados como arquivos de entrada para realizar das etapas mencionadas. O usuário utiliza a ferramenta de suporte, denominada Devcompatibility para leitura e análise de modelos AADL para suporte a aplicação das etapas e atividades da abordagem CAvA.

1.4 OBJETIVOS

Este trabalho tem como objetivo principal propor uma abordagem para evolução da arquitetura de CPS com uso intenso de S&A, fundamentada nos conceitos de MDE e uso de ontologia aplicada à construção de arquitetura de CPS.

Além do objetivo principal, os seguintes objetivos específicos são considerados para garantir a aplicação da abordagem proposta:

1. Gerar uma base de conhecimentos sobre a evolução e troca de componentes da AS.
2. Propor um método de projeto de evolução da arquitetura de CPS com uso intenso de S&A, provendo para o time de projetistas um guia sistematizado de etapas, atividades e ferramentas de suporte.
3. Explorar a representação de AS com uso de ontologia.
4. Prover contribuições na representação de AS com uso da ontologia para detalhar as características, propriedades e relacionamentos dos componentes

do sistema e arquitetura, criando uma base de conhecimento que poderá auxiliar na construção da biblioteca de componentes e exploração de arquiteturas de forma automatizada.

5. Desenvolver uma abordagem de suporte ao processo de transformação de modelos e análise de AS, a partir da troca de S&A, de tal forma que os requisitos e objetivos de projeto sejam atingidos.

1.5 ORGANIZAÇÃO DA TESE

No capítulo 2, são apresentados os conceitos e técnicas relacionadas ao tópico de pesquisa desta tese, tal como MDE, transformação de modelo, linguagem de descrição de arquiteturas, ontologia e outras mais.

Em seguida, o capítulo 3 apresenta os trabalhos relacionados ao tema. Inicialmente, métodos e abordagens de desenvolvimento de AS, uso de ontologia para representar componentes e sistemas em domínios relacionados. E finalmente, trabalhos que abordam a troca, modificação e evolução de componentes da arquitetura.

No que concerne ao capítulo 4, é proposta uma abordagem para evolução de arquiteturas de CPS com uso intensivo de S&A, sendo descrito como um conjunto de etapas e atividades que buscam guiar os projetistas no processo de evolução da AS.

No capítulo 5 é introduzida a Ontologia de Arquitetura de Sistemas (OAS) que fornece conceitos, relacionamentos e representação dos componentes que compõem AS baseados na linguagem AADL, padrão AS5506. Uma ferramenta de transformação de modelos descritos em OWL para AADL é apresentada para suporte a modelagem de arquiteturas a partir da representação de modelos OWL.

Já no capítulo 6 é apresentada uma abordagem de troca S&A, em modelos AADL. A abordagem proposta descreve um conjunto de etapas de atividades para exploração, análise e ranqueamento de cenários fundamentados no conceito MDE, com uso da linguagem AADL. Esta abordagem possui suporte com uso da ferramenta *DevCompatibility*, que guia os projetistas na realização das atividades de forma automatizada e interativa, gerando como resultado, um novo modelo AADL da arquitetura evoluída. No Capítulo 7, apresentam-se experimentos realizados no projeto de evolução do protótipo de um VANT, aplicando as abordagens propostas.

Finalmente, o Capítulo 8 traz contribuições, resultados e sugere futuras linhas de pesquisa mediante os problemas encontrados.

1.6 LISTA DE PUBLICAÇÕES

Ao longo do desenvolvimento desta tese, os seguintes artigos foram produzidos:

- (SALES, Diego Câmara; BECKER, Leandro Buss, 2018) Systematic Literature Review of System EngineeringDesign Methods. SBESC 2018, p. 6,

Salvador/BA. http://dx.doi.org/10.1109/SBESC.2018.00040

- (SALES, Diego Camara; KOLIVER; BECKER, Leandro Buss, 2020) Ontology and Rules for Characterization of Sensors and Actuators Devices in AADL Models. SBESC 2020, p. 8, online. http://dx.doi.org/10.1109/sbesc51047.2020.9277856
- (SALES, Diego C; BECKER, Leandro Buss, 2021) Approach for Evolving Sensing and Actuation Devices in Cyberphysical Systems Architectures. 9th International Conference on Model-Driven Engineering and Software Development, 2021, p. 306. http://dx.doi.org/10.5220/0010310303060313
- The Systems Architecture Ontology (SAO): an Ontology-based Design Method for Cyber-Physical Systems. Journal *Applied Computing and Informatics* em Setembro de 2021.

2 REFERENCIAL TEÓRICO

Desenvolver CPS que utilizam intensivamente sensores e atuadores em sua arquitetura é um desafio, especialmente em projetos de VANT que exigem componentes específicos para diversas aplicações. Esta complexidade é inerente às aplicações e ambiente de operação, bem como às garantias exigidas que podem ser aplicadas a estes sistemas (LEE, E. A., 2008). Sendo assim, utilizar uma abordagem de projeto para representar e descrever suas características, funcionalidades e relacionamento entre os componentes é de fundamental importância na compreensão, manutenção e atualização do sistema. Este processo é composto por um conjunto de passos com objetivo de detalhar o sistema e sua aplicação. Nesse sentido, com objetivo de prover suporte às atividades de desenvolvimento, são utilizadas tecnologias e técnicas (LEE, E. A., 2017; GONÇALVES; BECKER, L. B., 2016).

Desta forma, as tecnologias e técnicas fornecem suporte aos projetistas na modelagem dos CPS's, evitando erros de projeto e detalhando os componentes da arquitetura. Diferentes técnicas podem ser aplicadas no desenvolvimento de CPS's e suas aplicações, como, por exemplo, sistemas de VANT's. Nesse contexto, técnicas, linguagens de modelagem e descrição de arquiteturas, assim como ferramentas utilizadas durante o processo de desenvolvimento do projeto são apresentadas neste capítulo.

2.1 ENGENHARIA DIRIGIDA A MODELOS (MDE)

A MDE é uma abordagem que usa modelos e notações para elevar o nível de abstração e para representar diferentes pontos de vista do sistema, tais como: representação dos componentes mecânicos e elétricos, plataformas computacionais e controle do sistema (SCHMIDT, 2006).

Esta abordagem tem como foco representar as características do sistema por meio da construção de modelos. O objetivo é fatorar a complexidade em diferentes níveis de abstração, desde modelos conceituais de alto nível até os aspectos individuais dos componentes do sistema e seu código-fonte (ALANEN *et al.*, 2004). Para isso, a MDE possui suporte ao processo de geração automatizada de códigos baseados nos modelos desenvolvidos, estrutura do software e plataforma que deseja-se implementar.

Durante as etapas de desenvolvimento de CPS, diferentes modelos são gerados com intuito de fornecer informações de pontos de vista do sistema, evitando erros e esforço durante o projeto. Cada modelo possui uma estrutura e linguagem específica de um determinado domínio, que precisa ser integrado para compor a aplicação. A MDE busca auxiliar nesse processo de integração, assim como expressar de forma efetiva os conceitos da aplicação. A MDE combina dois artefatos tecnológicos para realizar esta integração (SCHMIDT, 2006):

- **Linguagens de modelagem específicas de domínio** (do inglês, *Domain-specific Modeling Languages (DSML)*:Linguagens que formalizam a estrutura da aplicação, comportamento e requisitos de um domínio particular. As DSML são descritas usando metamodelos, que definem os relacionamentos entre os conceitos em um domínio e especificam a semântica e restrições associadas.
- **Mecanismos de transformação e geradores** (do inglês, *Transformation engines and generators*): Realizam a análise de certos aspectos do modelo e sintetizam diferentes tipos de artefatos, tal como um código-fonte, entradas de simuladores, descrição de arquivos XML ou representações de modelo alternativos. A capacidade de sintetizar artefatos de modelos, ajuda a garantir a consistência entre implementações de aplicativos e informações de análise associadas a requisitos funcionais dos modelos.

A MDE propõe o processo de transformação de modelos, onde a partir de um modelo de domínio específico, descrito como modelo-fonte, se torna possível à geração de diferentes modelos, descritos como modelos-alvo, os quais devem estar em conformidade com o modelo-fonte. O processo de transformação pode ocorrer de três formas. Sendo elas: Modelo para Modelo (M2M) para a geração de novos modelos; Modelo para Texto (M2T) com a geração de documentação e representação taxonômica; e Modelo para Código (M2C) com a geração de código-fonte.

A Figura 3 apresenta um processo genérico de MDE, onde o projetista de software inicia criando um modelo n, conforme a linguagem de modelagem n. O modelo n descreve o sistema em alto nível de abstração. Neste caso, a transformação do modelo é realizada de duas maneiras, gerando novas representações de modelos M2M e fornecendo o código da aplicação M2C. Na transformação M2M o modelo n no modelo $n-1$, que, por sua vez, está em conformidade com outra linguagem de modelagem $n-1$, o projetista detalha as características específicas da plataforma e simultaneamente diminui o nível de abstração. Ao final do processo ocorre o M2C, quando o código-fonte pode ser gerado a partir do modelo 1 (por exemplo, código Java) em conformidade com um formalismo gramatical, por exemplo, *extended Backus–Naur form* (EBNF) (STAAB *et al.*, 2010).

Uma das iniciativas mais conhecidas do MDE é a Arquitetura Orientada por Modelos ,do inglês Model-Driven Architecture (MDA) proposta pelo Object Management Group (OMG). Essa abordagem definiu um conjunto de padrões para permitir o desenvolvimento das aplicações e especificação das funcionalidades do sistema ao nível de arquitetura. As metodologias baseadas na MDA permitem o desenvolvimento das aplicações por meio de processos automatizados de mapeamento e referência de modelos com suas respectivas implementações.

A MDA é baseada na modelagem dos requisitos de projeto e conceito de mape-

Figura 3 – Visão geral do processo de MDE.
Fonte: Modificado de (STAAB *et al.*, 2010).

amento, onde modelos independentes de plataforma computacional, Platform Independent Model (PIM), são definidos para descrever o sistema na perspectiva do software e na sequência ocorre a moedelagem para um plataforma específica, Platform-Specifi Models (PSM). A modelagem é baseada nos princípios de MDE com uso de metamodelos que descrevem a sintaxe e semântica operacional, assim como linguagens específicas de modelagem (MELLOR *et al.*, 2004).

Outra abordagem bastante difundida na comunidade cientifica é a Eclipse Modeling Framework (EMF), que fornece uma base sólida para o desenvolvimento de aplicativos por meio do uso de modelagem pragmática e estruturada, com recursos de geração de código automatizado. A EMF unifica três tecnologias importantes: Java, *eXtensible Markup Language (XML)* e *Unified Modeling Language (UML)* onde, independente da linguagem de design, a representação fornecida pode ser considerada um modelo comum com propriedades dessas três linguagens. Esta capacidade possibilita definir um método de transformação EMF aplicado a outras tecnologias (STEINBERG *et al.*, 2008).

2.1.1 Modelos e Metamodelos

Os modelos representam um conjunto de declarações do sistema que deseja-se implementar (SEIDEWITZ, 2003). Um modelo busca descrever uma visão simplificada da realidade (SELIC, 2003). Sendo formado por um conjunto de elementos formais que descrevem algo, e podem ser analisados usando vários métodos (MELLOR *et al.*, 2004).

Além do que é especificado em sua definição básica de um modelo, é recomendável que um modelo de engenharia possua cinco características principais. São elas: *1)* **Abstração**- um modelo é sempre uma representação reduzida do sistema que ele representa e envolve ignorar informações que não são de interesse em um determinado contexto; *2)* **Compreensibilidade**- não é suficiente apenas abstrair os detalhes; é necessário apresentar o que deve ser modelado e quais característica devem ser consideradas, por meio da intuição e da base de conhecimentos; *3)* **Acurácia**- um modelo deve fornecer uma representação realista das características do sistema modelado de

interesse; *4)* **Preditividade**- a partir do modelo devemos ser capazes de prever corretamente as propriedades desejadas, mas não óbvias, através de experimentações e análise formal; *5)* **Inexpressividade**- o modelo deve ser significativamente barato de ser construído e atualizado do que o sistema modelado (SELIC, 2003).

O ponto central do MDA é a criação de diferentes modelos com níveis distintos de abstração e detalhamento de características do sistema para serem vinculados a uma plataforma específica. Alguns desses modelos não possuem relação direta com o software, enquanto outros podem ser complementares, inter-relacionados e reutilizados na geração de novos modelos a partir dos metamodelos e processo de transformação (MELLOR *et al.*, 2004).

Metamodelo é outro conceito usado no MDA para definir a estrutura, semântica e restrições de modelos que compartilham sintaxe e semântica comuns. Metamodelo é um modelo de linguagem que provê suporte à construção de modelos. Estes descrevem as definições a serem seguidas na criação semântica dos modelos e sua estrutura, definindo os relacionamentos entre os elementos do modelo (MELLOR *et al.*, 2004). A linguagem utilizada na construção dos metamodelos é chamada de meta-meta-modelo (MMM).

2.1.2 Transformação de modelos

A transformação de modelos representa a geração automatizada de um modelo alvo com base em um modelo fonte do mesmo sistema (BELAUNDE *et al.*, 2003). O processo de transformação consiste na definição de um conjunto de regras que detalham como os modelos, na linguagem fonte, são transformados em modelos equivalentes nas linguagens alvo (BELAUNDE *et al.*, 2003).

Uma visão geral do processo de transformação de modelos é apresentada na Figura 4. A partir de um modelo M_a, em conformidade com um metamodelo MM_a, deseja-se realizar a transformação para o modelo alvo M_b que está em conformidade com o metamodelo MM_b. A transformação é definida pelo modelo de transformação de modelo M_t, que está em conformidade com um metamodelo de transformação de modelo MM_t. Os metamodelos MM_a,MM_b e MM_t devem estar em conformidade com o meta-meta-modelo *MMM* (ECLIPSE, 2018).

No trabalho de Mens e Van Gorp (2006) foram definidos alguns conceitos quanto ao processo de transformação de modelos. Se os metamodelos de origem e de destino são idênticos, a transformação é chamada de endógena; caso contrário, é chamada de exógena. Se o nível de abstração não mudar, a transformação é chamada de transformação horizontal. Se o nível de abstração mudar, a transformação é chamada de transformação vertical.

O processo de transformação é guiado por um conjunto de regras de mapeamento que descrevem como as instâncias do metamodelo fonte são mapeados para

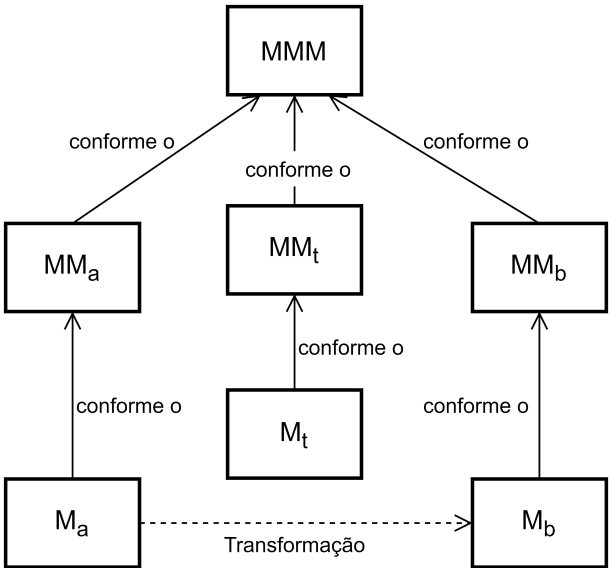


Figura 4 – Visão geral do processo de transformação de modelos.
Fonte: (ECLIPSE, 2018).

geração de instâncias de metamodelos especificados na linguagem alvo (BELAUNDE *et al.*, 2003). As regras de mapeamento especificam os relacionamentos entre os diferentes elementos dos modelos. Identificar e caracterizar estes relacionamentos, estrutura e semântica é fundamental para estabelecer o mapeamento de diferentes tipos de relacionamentos entre os elementos.

As categorias de mapeamento entre os elementos dos modelos pode ser realizado das seguintes formas: de um para um elemento, de um para múltiplos e de múltiplos para um. Esta classificação é baseada no conceito de estrutura e semântica semelhantes entre os elementos dos metamodelos (LOPES *et al.*, 2006).

Um mapeamento de um-para-um ($1 : 1$) é caracterizado por um elemento de um metamodelo alvo que pode representar a estrutura e semântica semelhantes de um elemento de um metamodelo fonte. No mapeamento de um-para-muitos ($1 : n$) um elemento de um metamodelo fonte pode apresentar estrutura e semântica para mais de um metamodelo alvo. O mapeamento de muitos para um elemento ($n : 1$) é caracterizado por um conjunto de elementos não vazio e não unitário de um metamodelo fonte que pode representar a estrutura e semântica semelhante a um elemento de um metamodelo alvo.

Os conceitos de MDE e MDA apresentados fornecem suporte ao processo de desenvolvimento das etapas de projeto de CPS, sendo que diferentes linguagens de transformação e modelagem, bem como ferramentas foram propostas ao longo dos anos por empresas e comunidade científica. Os ambientes de modelagem assistida por computador e ferramentas complementares auxiliam na descrição da arquitetura do sistema e compreensão da interação entre os componentes, análise e avaliação das características declaradas, fornecendo dados importantes que guiam os projetistas

no processo evolucionário dos modelos até a geração do código-fonte.

2.2 ONTOLOGIA

Na ciência da computação, a ontologia é uma forma de destacar os recursos, características, atributos ou parâmetros de um domínio, além de representar como eles se relacionam (GRUBER *et al.*, 1993). Uma vez que as ontologias identificam conceitos, taxonomias, suas relações e hierarquias que existem em algum domínio, elas são teorias de conteúdo por excelência (CHANDRASEKARAN; JOSEPHSON; BENJAMINS *et al.*, 1999).

Dentre os benefícios de utilizar ontologia no projeto de CPS temos a definição de um vocabulário comum e estruturado que auxilia na comunicação contínua entre múltiplos times de projetistas, reduzindo a ambiguidade na transferência de conhecimentos e conceitos entre equipes de projetistas, evitando o compartilhamento de informações incorretas. A interpretação errada de informações - ou pior ainda - o seu compartilhamento entre os participantes, pode gerar consequências negativas durante as etapas e atividades de projeto. O mesmo se aplica a CPS, onde o conhecimento (conceitos e relacionamentos) são fundamentais para o entendimento e desenvolvimento da arquitetura. Além disso, ontologias possibilitam a integração de conhecimentos de outras ontologias, ou parte delas, por meio de um conceito bastante difundido na engenharia de software, a reutilização (SCHLENOFF *et al.*, 2015).

Uma ontologia é uma descrição formal explícita de um domínio, constituído por classes, ou conceitos, propriedades das classes, que descrevem suas características, e atributos das classes que descrevem um conjunto de regras ou propriedades. Deste modo, uma ontologia junto com um conjunto de instâncias individuais de classes constitui uma base de conhecimento (NOY; MCGUINNESS *et al.*, 2001).

O consórcio World Wide Web Consortium (W3C) propõe um conjunto de técnicas, padrões e modelos de dados aplicados a modelagem de vocabulários e ontologias. Isso inclui estrutura de descrição de recursos Resource Description Framework (RDF) e RDF Schema utilizados para descrever as propriedades e classes dos recursos RDF com uma semântica de hierarquias (PAN, 2009). O Sistema de Organização de Conhecimento Simples (*Simple Knowledge Organization System* - SKOS) (MILES; BECHHOFER, 2009), linguagem de Ontologia da Web *(Ontology Web Language* - OWL) (MCGUINNESS; VAN HARMELEN *et al.*, 2004) e o formato de Intercâmbio de Regras (*Rule Interchange Format* - RIF) (KIFER, 2008). É importante destacar que a escolha entre essas diferentes tecnologias depende da complexidade e do rigor exigido por uma aplicação específica.

As ontologias em si expressam um conjunto de declarações em uma linguagem natural. No entanto, as declarações em linguagem natural são difíceis de processar em um computador. Uma das formas de representar as ontologias em um computador é

por meio da web semântica e das linguagens formais (GAŠEVIC; DJURIC; DEVEDŽIC *et al.*, 2009).

2.2.1 Web Semântica e tecnologias aplicadas

O termo "web semântica"refere-se à visão do W3C em relação aos dados vinculados à Web. As tecnologias da web semântica permitem que as pessoas criem armazenamentos de dados na web baseados em ontologias e escrevam regras para manipulação de dados, por meio de linguagens formais. Os dados podem ser potencializados mediante o uso de tecnologias como RDF, XML, OWL e SKOS que são lidos e processados por computadores.

A Web Semântica fornece uma descrição de seus conteúdos em forma legível por máquina, modelando a representação dos conceitos da ontologia. Desta forma, facilita a interoperabilidade e o compartilhamento de base de conhecimento que representam o domínio da arquitetura de CPS. Seu principal objetivo é, portanto, tornar a informação na web acessível e compreensível para humanos e computadores de tal forma que possam ser executadas tarefas úteis de raciocínio e inferência por meio de um conjunto de regras semânticas e axiomas (TAYE, 2010).

Os grupos de trabalho do W3C definiram os princípios da Web Semântica baseados em um conjunto de camadas de tecnologias e padrões conforme apresentado na Figura 5. As camadas *Unicode* e *URI* usam os conjuntos de caracteres internacionais e fornecem meios para identificar os objetos na Web Semântica. A camada XML com definições de NameSpace e esquema integram as definições da web semântica com os outros padrões baseados em XML. Na camada de dados, o RDF com RDFSchema fornecem as declarações de objetos, propriedades e relacionamentos, definição de atributos e dados com base no vocabulário e referenciados por URI's. A camada vocabulário da ontologia suporta a definição dos vocabulários utilizados na representação da web semântica através de tecnologias de linguagem. A camada lógica define um conjunto de axiomas, regras semânticas e inferências baseados na ontologia. A camada de assinatura digital é utilizada na detecção de alterações nas camadas lógica, vocabulário e RDF (BERNERS-LEE; HENDLER; LASSILA *et al.*, 2001).

2.3 FERRAMENTAS E LINGUAGENS

Durante as atividades de desenvolvimento de CPS é utilizado um conjunto de ferramentas e tecnologias de suporte a modelagem de CPS baseados em MDE e MDA. As ferramentas, em sua grande maioria, fornecem um ambiente de desenvolvimento gráfico e funcionalidades que facilitam a organização, estruturação, avaliações e análises de múltiplos pontos de vista relacionados ou não com a arquitetura. As linguagens fornecem os meios de descrever as características e propriedades dos

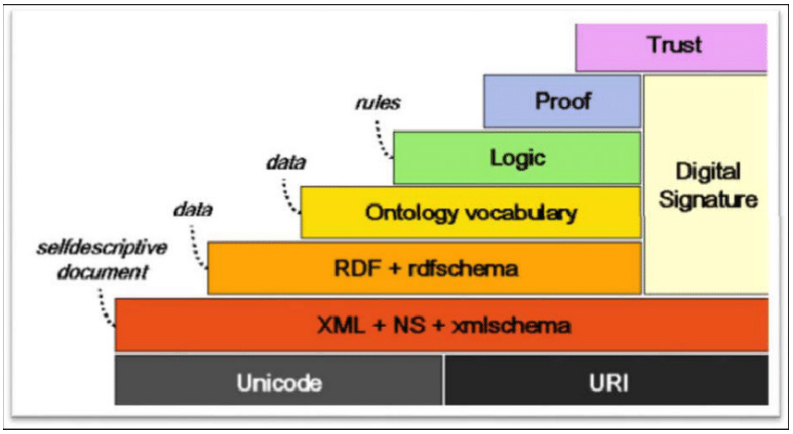


Figura 5 – As camadas da web semântica.
Fonte: (BERNERS-LEE; HENDLER; LASSILA *et al.*, 2001).

componentes que compõem a arquitetura em diferentes níveis de abstração, através de uma estrutura de declarações, semântica e formalismo baseados nos princípios da MDA.

2.3.1 Ontology web language (OWL)

A tecnologia OWL é uma linguagem de ontologia da web, amplamente utilizada na comunidade científica para modelagem de conceitos e base de conhecimentos. Desenvolvida durante as atividades dos grupos de trabalho da W3C, OWL está atualmente na versão 2 e evoluiu a partir da linguagem *DARPA Agent Markup Language* (DAML) e *Ontology inference Layer* (OIL) proposta pela Agência de Projetos de Pesquisa Avançada de Defesa, (*Defense Advanced Research Projects Agency* (DARPA) (MCGUINNESS *et al.*, 2002). Atualmente, a OWL2 é a linguagem de representação de ontologias mais popular.

A OWL2 fornece uma estrutura de declaração das classes, propriedades, relacionamentos, vocabulário, indivíduos e valores de dados utilizados na web semântica, representando a tecnologia utilizada na camada de vocabulário da ontologia mostrado na Figura 5. A OWL2 foi projetada para processar o conteúdo da informação e facilitar a interpretação por máquinas, fornecendo um vocabulário adicional baseado em XML.

O vocabulário OWL2 inclui um conjunto de elementos e atributos XML, com significados bem definidos. Estes são usados para descrever propriedades e classes: entre outros, relações entre classes (por exemplo, disjunção), cardinalidade, igualdade, tipificação mais rica de propriedades, características de propriedades e classes (MCGUINNESS; VAN HARMELEN *et al.*, 2004).

A linguagem OWL2 permite ao projetista descrever com mais detalhes o relacionamento entre os componentes da arquitetura, enriquecendo a modelagem quanto à declaração das características e propriedades, além de representar a hierarquia das classes relacionando com ontologias de reuso. Dessa forma, utilizar OWL2 na

modelagem dos CPSs pode auxiliar na contextualização e compreensão do projeto da arquitetura.

2.3.2 Linguagem de regras da web semântica

A linguagem de regras da semântica, (*Semantic Web Rule Language* (SWRL), fornece uma sintaxe abstrata de alto nível para definir regras baseadas na combinação de modelos OWL, RDF, XML, Unicode e URI. A SWRL estende o conjunto de axiomas OWL para declarar regras baseadas em fórmulas lógicas com propriedades uteis para uso em programação lógica, especificação formal e teoria de modelo, conhecido, no campo da lógica matemática e programação lógica, como cláusula Horn.

As regras permitem a combinação de cláusulas Horn com uma base de conhecimento modelada em OWL. Uma extensão das semânticas do modelo OWL também fornece um significado formal para as ontologias com a inclusão de regras escritas. As regras propostas são da forma de uma implicação entre um antecedente (corpo) e um consequente (cabeça). O significado pretendido pode ser lido como: sempre que as condições especificadas no antecedente forem válidas, as condições especificadas no consequente também devem ser válidas.

A sintaxe semântica de uma regra é apresentada na Equação (1), onde o antecedente e consequente são átomos escritos com base nas entidades (classes, propriedades dos objetos e dados) da ontologia modelada em OWL. As variáveis são indicadas usando a convenção padrão da sintaxe SWRL, onde a entidade OWL é seguida de parênteses contendo variáveis e um ponto de interrogação, $OWLentidade(?x,?y)$. Usando esta sintaxe, uma regra que afirma que a composição da classe $parente(?x,?y)$ e ($\wedge$) $irmao(?y,?z)$ implica que é $tio(?x,?z)$.

$$\underbrace{parente(?x,?y) \wedge irmao(?y,?z)}_{\text{antecedentes}} \Rightarrow \underbrace{tio(?x,?z)}_{\text{Consequente}} \tag{1}$$

A SWRL é uma abordagem modular que permite extensões e comandos adicionais (build-in), dentro de uma taxonomia hierárquica, fornecendo flexibilidade para várias implementações de formalismos. Por exemplo, a Equação (2) apresenta uma regra SWRL que permite avaliar instâncias que possuem a propriedade de dados com valor numérico menor que 5, que quando verdadeiro, implica na classificação do tipo Dispositivo.

$$\underbrace{Sensor(?x)}_{\text{antecedente}} \wedge \underbrace{temMassa(?x,?y) \wedge swrlb:lessThan(?y,5)}_{\text{átomo de propriedade de dados da OWL}} \Rightarrow \underbrace{Dispositivo(?x)}_{\text{Consequente}} \tag{2}$$

Quanto ao projeto de arquitetura de CPS, as regras semânticas SWRL com comandos adicionais, que permite a criação de um conjunto de átomos, conforme a base

de conhecimento, para avaliar os conceitos, restrições de projeto e classificação dos componentes inerentes a arquitetura. Para isso, é necessário utilizar uma ferramenta, ambiente de desenvolvimento, capaz de prover uma interface gráfica de modelagem de ontologias.

2.3.3 Protegé

Atualmente, Protegé é o principal ambiente de modelagem de ontologias. Este ambiente foi desenvolvido na Universidade de Standford com o objetivo de facilitar o desenvolvimento da ontologia na definição de classes, propriedades dos objetos e dados, taxonomia e restrições, bem como instâncias de classes ou indivíduos. Sua interface gráfica auxilia o projetista a organizar a estrutura conceitual da ontologia e criar a base de conhecimento, definindo as instâncias da ontologia (CORCHO; FERNÁNDEZ-LÓPEZ; GÓMEZ-PÉREZ, 2003).

O ambiente integra um conjunto de ferramentas, plug-ins, para realizar a modelagem de regras semânticas com SWRL (SWRLTab), verificação de inconsistências semânticas na definição da ontologia (Ontop), geração de gráficos espaciais (VOWL), métricas da ontologia e oferece suporte a várias linguagens de representação de ontologias, incluindo OWL e RDF+RDFSchema. Dentre os plug-ins disponíveis existe um conjunto de raciocinadores (Pallet, ECK, Hermit, Jcel e Ontop) que, baseados no modelo desenvolvido em OWL e regras semânticas SWRL, realizam testes de inferência e satisfatibilidade automaticamente.

2.3.4 Architectural Analysis Design Language (AADL)

A linguagem de projeto e análise de arquitetura, do inglês *Architecture Analysis & Design Language* (AADL), foi proposta pela Sociedade automotiva de engenheiros (SAE), e definida no padrão AS5506. A linguagem AADL tem sido amplamente utilizada na descrição de sistemas embarcados críticos aplicados no ramo automotivo e aviônico (FEILER, Peter H.; LEWIS; VESTAL *et al.*, 2006). Esta linguagem propõe a descrição da arquitetura de sistemas a fim de facilitar a análise por meio de diferentes técnicas (simulação, método formal, verificação de modelo), incluindo a sua implementação em plataforma específica (KORDON, F.; J. HUGUES A. CANALS, 2013).

O padrão AADL inclui um formato de intercâmbio com arquivos XML, definido em termos de um metamodelo para AADL (XMI), para facilitar o intercâmbio de modelos, integração com outras ferramentas e transformação automatizada de modelos AADL. A linguagem AADL descreve o conjunto de componentes que compõem a arquitetura de um sistema. Deste modo, por meio de ferramentas de modelagem, os projetistas podem criar, avaliar e analisar características do modelo, assim como a geração do código fonte para plataformas específicas. Modelos arquiteturais descritos em AADL buscam representar as características, propriedades e integração entre

os componentes de software, hardware e sistema da arquitetura (FEILER, Peter H.; LEWIS; VESTAL *et al.*, 2006).

Os recursos disponíveis em AADL são representados na Figura 6, sendo organizados em três categorias: componentes de software, componentes de hardware e composição (sistema). Os componentes de (*Software da aplicação*), são representados por processos (códigos e dados compilados), tarefas (caminhos de execução), grupo de tarefas e subprogramas. Estas são as principais abstrações de software em tempo de execução dentro do AADL.

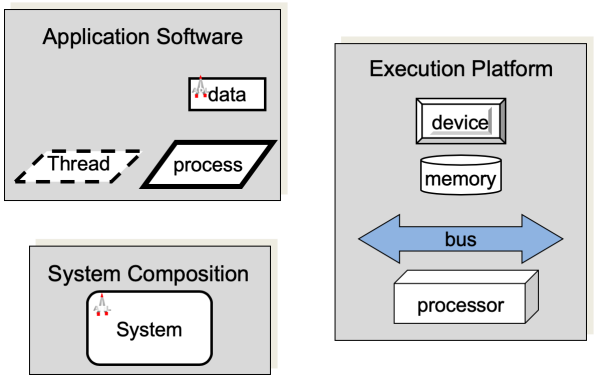


Figura 6 – Componentes AADL
Fonte: (FEILER, Peter H; GLUCH, 2012).

Os componentes de hardware, (*plataforma de execução*), detalham as características dos dispositivos físicos, sejam estes sensores, atuadores, memória, processador e barramentos. Mais recentemente, na versão 2.0, foram incluídos os componentes virtuais para representar processadores e barramentos. Estes componentes fornecem os ambientes de execução, caminhos de comunicação física e interfaces externas para um sistema de computação. Os componentes, (*composição do sistema*), são responsáveis por permitir a construção da arquitetura, composta por elementos de hardware e software. Componentes abstratos e protótipos podem compor a arquitetura de um sistema.

O padrão AADL descreve a representação dos componentes de forma hierárquica e organizada em pacotes, onde cada pacote é detalhado com a inclusão de características (*Features*) e propriedades (*Properties* e *Properties Set*) declaradas aos tipos de componentes (*Component Type*) e componentes implementados (*Component Implementation*). Essa hierarquia estabelece todas as interações entre os componentes e a estrutura arquitetônica necessária para definir um sistema executável. Estes englobam trocas de dados e eventos, conexões físicas entre os componentes (*Connections*), bem como a atribuição de software para hardware (por exemplo, onde os dados e instâncias de execução do software são executados) (FEILER, Peter H; GLUCH, 2012). A Figura 7 apresenta o relacionamento hierárquico das categorias de componentes, tipos e implementação dos pacotes.

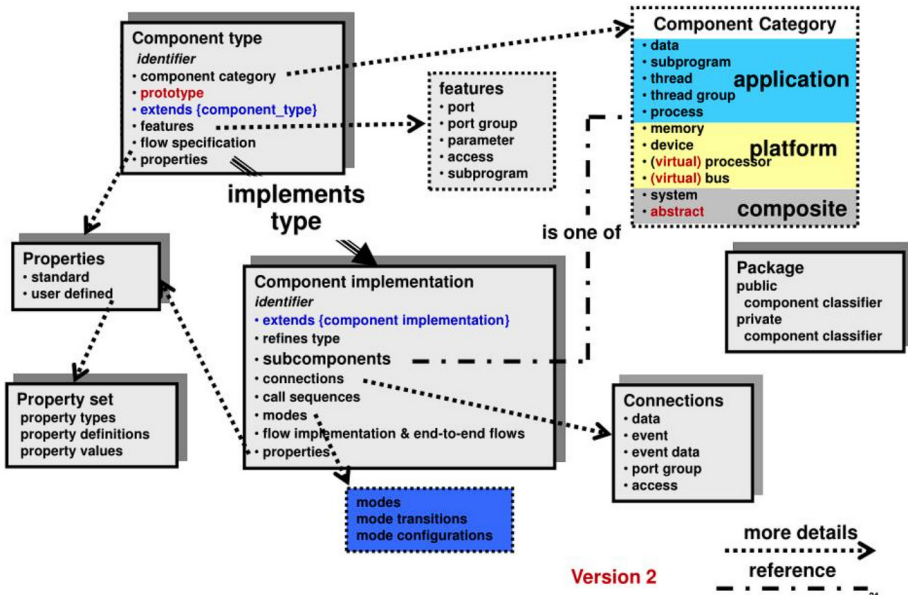


Figura 7 – Componentes AADL e relacionamentos.
Fonte: (FEILER, Peter H; GLUCH, 2012).

A linguagem AADL suporta a extensão de anotações nos modelos com a declaração de propriedades através de sub-linguagens chamadas de anexos. Com isso, diferentes propriedades podem ser adicionadas na representação dos componentes com objetivo de detalhar propriedades voltadas à capacidade do sistema, tais como confiabilidade, segurança e desempenho do sistema.

As propriedades declaradas fornecem suporte à avaliação do sistema instanciado em um determinado cenário modelado. Assim, por meio do uso de instâncias do modelo AADL, os projetistas podem avaliar as características definidas em busca de identificar se a aplicação desenvolvida atendeu às restrições e requisitos de projeto.

A linguagem de especificação de requisitos, Requirement Specification Language (ReqSpec) tem como base o rascunho de um anexo de análise de requisitos, *Requirements Definition* and *Analysis Language* (RDAL) *Annex*. O RDAL é baseado no padrão de prática recomendada para especificações de requisitos de software (COMMITTEE; BOARD, 1998) para especificação de requisitos em forma de anotações em modelos AADL. O objetivo da ReqSpec é apoiar a elicitação, definição e modelagem de requisitos para sistemas embarcados de tempo real em um processo iterativo. A ReqSpec suporta o refinamento de requisitos junto com o projeto do sistema; a análise qualitativa e quantitativa da especificação de requisitos criada; e, finalmente, a verificação dos modelos de arquitetura de sistema associados para que se certifique de que atendem aos requisitos.

A ReqSpec permite que os projetistas definam os objetivos, ou requisitos das partes interessadas, e requisitos, ou requisitos do sistema. Os objetivos são expressos por declarações de objetivos e os requisitos por declarações de requisitos em arqui-

vos do tipo *goaldoc* e *reqdoc* respectivamente. Os objetivos e requisitos podem ser organizados de acordo com a estrutura da arquitetura, a partir da associação com os componentes AADL e suas implementações, ou através de um documento estruturado em seções. Os requisitos das partes interessadas são declarados para representar os objetivos específicos dos componentes da arquitetura e o conjunto de objetivos declarados no arquivo do tipo *goals*. Os requisitos do sistema declaram os requisitos dos componentes específicos da arquitetura, sendo armazenados em arquivos do tipo *reqspec* (FEILER, Peter H *et al.*, 2016).

Diferentes ferramentas de modelagem de arquiteturas em linguagem AADL podem ser utilizados no desenvolvimento do sistema. As de código aberto são TOP-CASED(FARAIL *et al.*, 2006), TASTE (PERROTIN *et al.*, 2012) e OSATE2 (FEILER, Peter, 2004) . Estas ferramentas fornecem suporte à modelagem dos componentes de hardware, software e avaliação de capacidades do sistema.

O projeto TOPCASED é um acrônimo para *Toolkit in Open Source for Critical Applications Systems Development*. Esta ferramenta de código aberto foi desenvolvida para suporte à modelagem de aplicações críticas e desenvolvimento de arquitetura de sistemas. Contém um IDE baseado no framework da plataforma de desenvolvimento Eclipse, ao qual agrega funcionalidades, principalmente relacionadas com a implementação. Seu ambiente de desenvolvimento suporta a modelagem dos componentes de hardware e software, assim como sua implementação (FARAIL *et al.*, 2006).

A ferramenta é baseada em metamodelos, que fornecem uma representação gráfica, customização e suporte à transformação de modelos de tecnologias, tal como UML e AADL. Possui funcionalidades adicionais de modelagem das propriedades e requisitos, simulação de modelos, teste, validação, geração de código, e documentação de projetos (FARAIL *et al.*, 2006).

Outra ferramenta utilizada de arquiteturas é a TASTE, que fornece um ambiente de desenvolvimento dedicado aos sistemas embarcados em tempo real e foi criada por iniciativa da Agência Espacial Europeia em 2008, após a conclusão de um projeto FP6 chamado ASSERT. TASTE pode ser usada para projetar sistemas de pequeno a médio porte; ela se apoia em linguagens formais e tem como premissa construir softwares "corretos por construção". As principais tecnologias usadas em TASTE são AADL, ASN.1 e SDL (PERROTIN *et al.*, 2012).

Ressalta-se que a ferramenta TASTE permite que os projetistas de software integrem facilmente partes heterogêneas de código produzidas manualmente (em C ou Ada) ou automaticamente por ferramentas de modelagem externas, como MATLAB Simulink, SCADE ou *Real-Time Developer Studio* (PERROTIN *et al.*, 2012).

Uma das ferramentas amplamente utilizadas na comunidade científica para modelagem de arquiteturas é o OSATE 2, que fornece um ambiente para projetar, avaliar, analisar e simular sistemas CPS utilizando a tecnologia AADL. Essa ferramenta

foi aplicada no escopo desta tese, sendo apresentada, na sequência, uma seção destinada a detalhar suas funcionalidades e suporte à modelagem de arquiteturas.

2.3.5 OSATE 2

A ferramenta Open Source AADL Tool Environment (OSATE), versão 2, é um ambiente de desenvolvimento de código aberto para a linguagem AADL desenvolvida no Software Engineering Institute (SEI). A ferramenta OSATE 2 consiste em um conjunto de plug-ins para a plataforma Eclipse de software livre. Nesse ambiente, os arquitetos de software podem projetar e analisar modelos e, em seguida, gerar partes do código de implementação. OSATE 2 inclui um compilador para AADL textual, um editor gráfico para AADL, um gerador de modelo de instância e suporte ao formato de intercâmbio XMI baseado em XML para AADL com base em sua especificação de metamodelo.

A ferramenta OSATE 2 fornece suporte à modelagem de arquitetura de sistemas, avaliação, análise, verificações de características e propriedades do modelo arquitetural, assim como a geração de código fonte por meio dos anexos e dos plug-ins, fornecidos por SEI e projetos de parceiros, disponíveis no ambiente de desenvolvimento.

A ferramenta possibilita a importação de modelos da ferramenta Mathworks® MATLAB™ Simulink™ para o OSATE 2, criando modelos AADL a partir das definições e marcações. OSATE 2 inclui a capacidade de realizar uma variedade de análises de recursos, tais como: latência do início ao fim do fluxo de dados; análise funcional de integração dos componentes que possuem propriedades declaradas do protocolo ARINC429; verificação de consistência das conexões de portas (interface) do sistema, assim como a análise da topologia; análise de peso dos componentes e sistema; análise de potência elétrica e análise de recursos e consumo do sistema.

Dentre as análises dos recursos, temos o de análise de consumo da capacidade de processamento em milhões de instruções por segundo (MIPS) com base nos processos e tarefas alocados nos processadores; análise do consumo de memória, auxiliando a fm de garantir que a memória fornecida seja suficiente para atender às demandas de memória dos componentes do aplicativo; análise de largura de banda dos barramentos com base na vinculação inferida nas conexões e barramentos (FEILER, Peter H; GLUCH, 2012). Quanto às características relacionadas à confiabilidade da arquitetura, o projeto TOPCASED propôs anexos e plug-ins para análise de falha e propagação de dados com o anexo Error Model (FEILER, Peter; RUGINA, 2007), avaliação Funcional de Perigos (FHA), análise do tempo médio até a falha (MTTF) e comportamento dos componentes do sistema, demonstrado em aplicações em CPS (REDMAN *et al.*, 2010).

2.4 SUMÁRIO

Neste Capítulo, foram apresentados os principais conceitos, tecnologias e técnicas utilizados no desenvolvimento deste trabalho. Dentre eles, os conceitos de MDE, ontologia, ferramentas e linguagens aplicadas no projeto de CPS.

A abordagem de desenvolvimento de CPS com uso de MDE e seus conceitos provê suporte à implementação de modelos computacionais, que podem representar diferentes aspectos de uma arquitetura, com intuito de realizar um conjunto de avaliações e análises em busca de facilitar o processo de desenvolvimento do sistema e validação dos requisitos de projeto.

A ontologia é uma abordagem poderosa na modelagem de dados e representação de um conjunto de conceitos e relacionamentos dentro de um domínio. Desta forma, seu uso pode trazer benefícios durante o desenvolvimento da arquitetura do sistema, quanto à representação das características, propriedades e relacionamentos dos componentes.

3 TRABALHOS CORRELATOS

O presente capítulo apresenta os principais trabalhos científicos relacionados a esta tese de doutorado. Os mesmos foram organizados por categorias, sendo definido um conjunto de critérios de avaliação para cada uma das categorias abordadas.

Assim, foram definidos três temas de pesquisa relacionados ao conceito de evolução da arquitetura por meio da troca de dispositivos de S&A. O primeiro tema aborda as diferentes metodologias propostas para guiar os projetistas no desenvolvimento de projetos de arquiteturas de CPS. No segundo tema são apresentados trabalhos relacionados à representação de componentes da AS com uso de ontologia. O terceiro tema discute em mais detalhes, as abordagens de troca de componentes da arquitetura. Neste último, buscou-se focar em abordagens que envolvam a troca de dispositivos de S&A.

Com o objetivo de apresentar os trabalhos relacionados e prover o entendimento e avaliação, foi definido um conjunto de critérios. Os critérios foram definidos com base no estudo das abordagens que propõem a modificação, avaliação, exploração de cenários, troca de componentes da arquitetura disponíveis no mercado e denominados de Commercial Off-The-Shelf (COTS). Os COTS fornecem benefício quanto à redução de custos e prazos de desenvolvimento da arquitetura, devido aos fabricantes disponibilizarem no mercado componentes de software e hardware que outrora deveriam ser produzidos sob medida para o sistema.

Deste modo, foram levantados diferentes trabalhos de desenvolvimento de arquiteturas com o uso de COTS foram levantados com o intuito de identificar os métodos e abordagens amplamente divulgados na comunidade científica. Sendo definido um conjunto de critérios para nortear a proposta desta tese.

3.1 MÉTODOS DE PROJETO DE CPS

As abordagens ora apresentadas e discutidas foram selecionadas com base em critérios que incluem:*adequação para seleção de único componente (SUC), adequação para seleção de múltiplos componentes (SMC), capacidade de abordar de maneira sistemática as incompatibilidades durante/após o processo de seleção (CSIC), ferramenta de modelagem disponível (FMD), abordagem com normas/padrões (ANP), análise de requisitos (AR), análise de desempenho (AD), e análise de funcionalidades (AF)*.

Os critérios levantados foram propostos com base no trabalho de revisão sistemática de métodos e abordagens aplicados em AS e desenvolvidos no escopo desta tese (SALES, Diego Câmara; BECKER, Leandro Buss, 2018). Direcionando assim, as contribuições propostas nesta tese.

A partir desses critérios, verificou-se quais autores abordam a seleção de único

componente (SUC) e a seleção de múltiplos componentes (SMC). Nesse cenário, notou-se que o processo de seleção é fundamental para identificar componentes candidatos à troca. Sendo assim, evidenciou-se que a seleção de um único componente a ser trocado envolve uma complexidade menor do que a troca de múltiplos componentes devido à necessidade de avaliar o impacto nas características e propriedades da arquitetura atual.

Observou-se, também, quais abordagens possuem a *capacidade de abordar de maneira sistemática as incompatibilidade durante ou após o processo de seleção (CSIC)* e que dispõem de *ferramenta de modelagem disponível (FMD)* para realização de seleção e adequação dos componentes da arquitetura. Identificar quais os impactos na interface de hardware, software e propriedades da arquitetura é fundamental para guiar os projetistas na escolha do cenário escolhido e mensurar quais modificações devem ser feitas.

Considerou-se relevante os trabalhos que propõem a realização de *análise de requisitos (AR), análise de desempenho (AD), análise de funcionalidades (AF)* da arquitetura do sistema. A análise dos requisitos fornece um conjunto de parâmetros de projeto que auxiliam os projetistas na seleção e definição dos componentes candidatos. A AD ocorre o com objetivo de identificar impactos de desempenho na arquitetura devido à troca dos dispositivos de S&A. À vista disso, realizar a AD é fundamental para mensurar as modificações realizadas. A AF busca identificar os candidatos e suas funcionalidades. Atualmente, com o avanço tecnológico, os dispositivos de S&A integram múltiplas funcionalidades. Com isso, a análise auxilia a identificar e a avaliar a troca de múltiplos componentes da arquitetura por um único que integre múltiplas funcionalidades.

A literatura apresenta diferentes abordagens de projeto de sistemas que contemplam a evolução e avaliação da arquitetura. Essas abordagens, usualmente, são compostas por um conjunto de etapas e atividades que buscam orientar a equipe de projetistas durante o desenvolvimento dos sistemas.

Os trabalhos pesquisados retratam diferentes níveis de cobertura, ou seja, alguns trabalhos cobrem todas as etapas e atividades para desenvolvimento do projeto, enquanto outros são mais focados em etapas específicas que detalham a análise de mudanças na arquitetura, táticas de como realizar as modificações de acordo com critérios de qualidade e troca de componentes da arquitetura.

Diferentes autores propuseram a definição de um conjunto de etapas e atividades que auxiliam os projetistas no desenvolvimento dos sistemas e evolução da arquitetura durante o ciclo de vida do projeto, apresentando guias descritivos de "o que" deve ser feito para avaliar a troca de componentes da arquitetura.

O método de análise de troca de arquitetura, do inglês *Architecture Tradeoff Analysis Method* (ATAM), é usado há mais de uma década para avaliar arquiteturas de

software em domínios que variam de automotivo, financeiro à defesa militar (KAZMAN *et al.*, 1998). O ATAM foi desenvolvido pelo SEI. Este método apresenta um conjunto de etapas e táticas para mitigação de riscos no desenvolvimento de sistemas, sugerindo o uso de atributos de qualidade para análise.

O ATAM é composto por quatro fases que abordam diversos aspectos importantes para realização de mudanças na arquitetura que detalham a tomada de decisão de mudanças do projeto, elaboração de cenários, articulação dos objetivos do negócio, requisitos e atributos de qualidade priorizados, mapeamento de decisões arquiteturais com uso requisitos de qualidade e levantamento dos pontos críticos do software do sistema.

Mario Barbacci *et al.* (1998) apresenta, em detalhes, uma extensão do método ATAM, com objetivo de fornecer princípios para efetuar a análise de troca de componentes da arquitetura com base na modelagem dos atributos de qualidade. Os atributos de qualidade listados foram: segurança, desempenho e disponibilidade. Os autores destacam que estes atributos podem interagir entre si, gerando conflitos caso ocorram mudanças. Não foi apresentado como ocorre a análise dos atributos de qualidade, e nem as ferramentas disponíveis para modelagem. O autor também não aborda o uso de dispositivos COTS na arquitetura do sistema.

Nord *et al.* (2003) propuseram o método *Integrating the Architecture Tradeoff Analysis Method* (ATAM) *with the cost benefit analysis method* (CBAM) para avaliação do custo-beneficio de modificações na arquitetura. Esse método apresenta o conceito de avaliar os riscos de modificações com a avaliação de custo financeiro do projeto. Para melhor modularidade e capacidade de realizar modificações futuras na arquitetura, os autores propõem aos projetistas que realizem a exploração de cenários com uso de componentes COTS integrados à arquitetura do sistema.

Já o método ALMA (*Architecture Level Modifiability Analysis*) foca na modificabilidade e manutenção (BENGTSSON *et al.*, 2004). O ALMA consistem em cinco passos: (1) seleção do objetivo; (2) descrição da arquitetura do software; (3) levantamento de cenários das modificações; (4) avaliação dos cenários das modificações; e (5) interpretação dos resultados. O autor propõe três abordagens para comparar as arquiteturas candidatas: (1) apontar o melhor candidato para cada cenário; (2) ranqueamento de cada cenário candidato; e (3) estimar o peso de cada cenário candidato.

No que se refere à modificações arquitetônicas, Bass, Paul Clements, Kazman *et al.* (2013) apresentam em mais detalhes os conceitos, passos e táticas do método ATAM. Os autores descreveram o ATAM em seis partes: (1) fonte do estimulo; (2) o estimulo; (3) artefato; (4) ambiente; (5) resposta; e (6) medições e resposta. *Fonte de estímulo* especifica quem efetua a mudança: o desenvolvedor, o administrador, ou o usuário final. O *estímulo* especifica a mudança que deve ser feita, como adição, modificação e exclusão de funções. As mudanças podem ser feitas para melhoria

ou inclusão de novos atributos de qualidade e funcionalidade do sistema. O *artefato* especifica quais componentes, módulos, plataformas, interfaces e sistemas devem ser modificados. O *ambiente* especifica quando a mudança deve ser feita, tempo de projeto, tempo de compilação. A *resposta* efetua a mudança, testa e realiza o *deployment*. Por último, a *medição e a resposta* onde ocorrem as medições dos novos defeitos introduzidos devido à mudança e outros efeitos nos atributos de qualidade.

Os métodos ATAM, ATAM estendida, ATAM+CBA, ALMA, ADD e CBAM foram bem documentados e descrevem princípios fundamentais para realização de trocas de componentes na arquitetura, sendo sugeridas etapas e atividades para realizar trocas e avaliação de potenciais cenários. Entretanto, estes métodos não apresentam em detalhes a troca de componentes da arquitetura que utilizam e não possuem ferramentas disponíveis para modelagem ou realização de análises de compatibilidade de hardware e software da arquitetura.

Os autores mencionados detalham os passos do que deve ser feito para realizar modificações, trocas e evolução na arquitetura e sugerem o uso de componentes COTS, destacando a possibilidade de utilizar componentes COTS na arquitetura do sistema e seus subsistemas para habilitar a modularidade e capacidade de modificações futuras. Entretanto, a integração de componentes COTS na arquitetura não é trivial. Para isso, foi realizado um estudo buscando identificar quais os critérios utilizados para seleção e avaliação dos componentes COTS utilizados na arquitetura.

Os autores Mohamed, Guenther Ruhe, Eberlein *et al.* (2007a) e Garg (2017) realizaram estudos sistemáticos das abordagens utilizadas para resolver o desafio de selecionar e avaliar os componentes COTS. Ao todo, foram listadas dezesseis abordagens, das quais foram apresentadas suas principais características,vantagens e desvantagens. Estas abordagens foram avaliadas com intuito de identificar as atividades propostas no processo de seleção, definição, análises e objetivos que culminaram a escolha dos componentes candidatos.

As seguintes abordagens foram pesquisadas: abordagem *Off-The-Shelf Option* (OTSO) (KONTIO, 1995) apresenta dois fatores para seleção, custo da integração do componente e seu valor de aquisição; A abordagem IusWare da empresa (*IUStitia softWARis*) (MORISIO *et al.*, 1997) destaca multi-critérios para tomada de decisão, incluindo peso, preço e disponibilidade; os métodos PRISM (*Portable, Reusable, and Integrated Software Modules*) (LICHOTA; VESPRINI; SWANSON *et al.*, 1997), PORE (Procurement-Oriented Requirement Engineering) (NCUBE; MAIDEN, 1999), STACE (*Social-Technical Approach to COTS Evaluation*) (KUNDA; BROOKS *et al.*, 2003), BAREMO (*Balanced Reuse Model focuses*) (LOZANO-TELLO *et al.*, 2002), CARE (COTS-*Aware Requirements Engineering*) (CHUNG; COOPER, 2004), MiHOS (*Mismatch-Handling aware COTS Selection*) (MOHAMED; RUHE, Guenther; EBERLEIN *et al.*, 2007b), UnHOS (*Uncertainty Handling in Commercial Off-The-Shelf*) (IBRAHIM *et al.*,

2011) destacam a avaliação de requisitos, partes interessadas, reutilização e compatibilidade para integração com o sistema usando pesos ou pontuação.

A abordagem CRE (*COTS-based Requirements Engineering*) (ALVES, C.; CASTRO, 2001) foca no comparativo dos requisitos funcionais e não-funcionais dos componentes para seleção e avaliação; CEP (*Comparative Evaluation process*) (RUHE, Günther, 2002) expande a abordagem OTSO com critérios de desempenho, gerenciamento, funcionalidade e características funcionais que são avaliadas baseada em média ponderada; DEER (*DEcision support for componEnt-based softwaRe*) (CORTELLESSA *et al.*, 2008) auxilia na escolha de múltiplos componentes COTS realizando uma combinação otimizada com lógica fuzzy para minimizar os custos e satisfazer os requisitos de projeto.

As abordagens PECA (*Plan, Establish, Collect and Analyze*) (COMELLA-DORDA *et al.*, 2002), DesCOTS (*Description, evaluation and selection of COTS*) (GRAU *et al.*, 2004) efetuam a seleção e avaliação baseado no padrão ISO/IEC9126 de 2003, que foi substituída pela norma ISO/IEC 25010 em 2011, sendo definidos os critérios com base nas características e sub-características de qualidade do componente. A abordagem DesCOTS apresentou posteriormente uma ferramenta que auxilia a modelagem e avaliação guiada de componentes COTS denominado DesCOTS-EV (QUER; FRANCH; LOPEZ-PELEGRIN *et al.*, 2005);

A abordagem ACT (*Architecture Centric Tradeoff*) (SARKAR, 2012) apresenta três processos, sendo eles: meta-modelo, heurística e processador para prover o suporte a seleção e avaliação de componentes COTS. O meta-modelo possui três camadas chamadas conceitual, logica e física. O processo heurístico é utilizado para computar o custo dos componentes. O processador é usado para realizar a troca optimizada dos componentes COTS candidatos selecionados. ACT é baseado no documento técnico *Evolutionary Process for Integrating COTS* (EPIC) (ALBERT *et al.*, 2002) do SEI para guiar os projetistas nas atividades de integração de componentes COTS.

Farshidi *et al.* (2018) apresentaram uma abordagem conceitual de um sistema de apoio à decisão para seleção de tecnologia de software utilizando tomada de decisão de múltiplos critérios apresentado por Majumder (2015). O sistema levanta potências alternativas e efetua tomadas de decisão definindo pesos aos critérios definidos para reduzir as incertezas de seleção do software. A avaliação do modelo é realizada com base nos critérios de qualidade de software definidos no padrão ISO/IEC 25010, sendo definido os relacionamentos de acordo com o domínio e conhecimento dos projetistas. Entretanto, os autores não abordam como ocorre a avaliação das características e propriedades de dispositivos de S&A que possuem componentes de software integrados, seleção de múltiplos dispositivos, assim como uso de ferramentas de modelagem e arquitetura.

Ademais, os trabalhos aqui apresentados e avaliados de acordo com os critérios identificados demonstraram que alguns aspectos como a seleção e adequação de múltiplos dispositivos, modelagem dos dispositivos de S&A e arquitetura, uso de normas e padrões de qualidade, são assuntos que carecem de estudos. Com isso, nessa pesquisa é apresentada a abordagem de evolução de arquitetura CAvA (**C**PS **A**rchitecture E**v**olution **A**pproach) que busca detalhar um conjunto de etapas e atividades que contemplam os critérios apresentados. Dentre os benefícios da CAvA, destaca-se a seleção de múltiplos componentes, análise de funcionalidades e ferramenta de modelagem alinhada ao uso da LDA AADL. A Tabela 1 apresenta a relação dos critérios e métodos apresentados.

Abordagens Avaliadas	AF [8]	AD [7]	AR [6]	ANP [5]	FMD [4]	CSIC [3]	ASMC [2]	SUC [1]
OTSO (KONTIO, 1995)	●	●	◑	○	○	○	○	●
IusWare (MORISIO *et al.*, 1997)	●	◑	◑	◑	○	○	○	●
PRISM (LICHOTA; VESPRINI; SWANSON *et al.*, 1997)	○	○	◑	◑	○	○	○	●
PORE (NCUBE; MAIDEN, 1999)	●	○	◑	○	●	◑	○	●
CRE (ALVES, C.; CASTRO, 2001)	○	○	●	◑	○	◑	○	●
BAREMO (LOZANO-TELLO *et al.*, 2002)	○	●	○	○	●	○	○	●
PECA (COMELLA-DORDA *et al.*, 2002)	○	◑	○	○	○	◑	○	●
STACE (KUNDA; BROOKS *et al.*, 2003)	○	◑	●	○	○	◑	○	●
CARE (CHUNG; COOPER, 2004)	○	●	●	○	○	●	○	●
MiHOS (MOHAMED; RUHE, Guenther; EBERLEIN *et al.*, 2007b)	○	○	●	○	●	●	○	●
UnHOS (IBRAHIM *et al.*, 2011)	○	●	●	●	●	●	○	●
DEER (CORTELLESSA *et al.*, 2008)	○	○	○	○	○	○	○	●
CEP (RUHE, Günther, 2002)	◑	◑	○	○	○	○	○	●
desCOTS (QUER; FRANCH; LOPEZ-PELEGRIN *et al.*, 2005)	◑	◑	◑	●	●	○	○	●
ACT (SARKAR, 2012)	○	●	○	○	○	○	○	●
DSS (FARSHIDI *et al.*, 2018)	●	●	●	●	●	○	○	●
CAvA (Proposta)	●	◑	◑	◑	●	◑	●	●

Tabela 1 – Visão geral dos trabalhos relacionados.
●Atende completamente, ◑Parcialmente e ○não atende.

[1] SUC: Seleção de único componente.
[2] SMC: Seleção de múltiplos componentes
[3] CSIC: Capacidade de abordar de maneira sistemática as incompatibilidades durante/após o processo de seleção
[4] FMD: Ferramenta de modelagem disponível
[5] ANP: Abordagem com normas/padrões
[6] AR: Análise de requisitos
[7] AD: Análise de desempenho
[8] AF: Análise de funcionalidades

3.2 ONTOLOGIA APLICADA À ARQUITETURA DE SISTEMAS

Os seguintes critérios foram identificados quanto à representação dos componentes da arquitetura e sistema: *componentes de software, componentes da plataforma, componentes do sistema, implementação do sistema, conjunto de propriedades*. Cada um dos critérios busca identificar detalhes quanto à representação dos componentes que compõem uma arquitetura de CPS.

O critério *componentes de software* busca identificar quais ontologias detalham a representação de processos, tarefas, dados e subprogramas relacionados ao software do sistema. Os *componentes da plataforma* englobam a representação de processador, memória, barramentos e dispositivos da arquitetura (hardware do sistema). Enquanto, o critério *componentes do sistema* busca identificar a representação de interfaces de hardware e software do sistema. O *conjunto de propriedades* busca identificar quais propriedades foram elicitadas na representação de componentes da arquitetura e sistema.

Durante os anos, diferentes autores e organizações desenvolveram ontologias aplicadas a domínios relacionados à AS que integram a representação de componentes da arquitetura em suas bases de conhecimento. Desta forma, foi realizado um estudo com objetivo de identificar quais ontologias cobrem os critérios previamente definidos, sendo apresentadas em três tópicos: ontologias aplicadas a sensores e atuadores, ontologias aplicadas às plataformas computacionais e ontologia aplicada à robótica e à automação.

3.2.1 Ontologia: Sensores e atuadores

As primeiras ontologias que abordam em detalhes a representação de sensores foram os padrões internacionais desenvolvidos pelo *Open Geospatial Consortium* (OGC), denominado Sensor Web Enablement (SWE). Os SWE permitem que os projetistas representem um pequeno grupo de sensores, transdutores e dados relacionados aos sensores em aplicações na Web. Dentre os padrões SWE, a ontologia Observations and Measurements (O&M) (COX, 2006) representa a observação e as medições de sensores baseados na linguagem Sensor Model Language (SensorML) (BOTTS; ROBIN, 2007).

O World Wide Web Consortium (W3C) é a principal organização de padronização da World Wide Web que aprofundou a representação semântica de sensores em aplicações de rede de sensores. O grupo de pesquisa da W3C, Semantic Sensor Network Incubator Group (SSN-XG), apresentou a ontologia de especificação semântica de sensores, Semantic Sensor Network (SSN) (COMPTON, M. *et al.*, 2009, 2012) fundamentada na reutilização de duas ontologias, a Dolce UltraLight (DUL) e o Sensor-Stimulus-Observation (SSO) (JANOWICZ; COMPTON *et al.*, 2010). A SSN tem como

foco a representação de sensores, suas características e propriedades, abstraindo os componentes de software e plataforma, assim como a implementação do sistema.

Na sequência a W3C expandiu a SSN com a criação da ontologia SOSA (Sensor, Observation, Sample, and Actuator). A ontologia SOSA é composta por nove módulos que cobrem a representação da implementação do sistema, procedimentos, propriedades e características do sistema, observação, amostragem e atuação dos sensores e atuadores do sistema (JANOWICZ *et al.*, 2019). Apesar da SOSA fornecer uma vasta representação dos sensores e atuadores, foi observado um alto nível de abstração dos componentes de software, hardware e sistema. Onde, a classe de procedimentos engloba a definição de processos, tarefas e subprogramas sem detalhar como é feita a arquitetura do software. Outro problema observado foi quanto à classe sistema que se limita a indicar que os sensores e atuadores são alocados a um determinado sistema, não descrevendo os componentes que compõem a plataforma, interfaces de hardware e software utilizadas e alocação.

Outros trabalhos relacionados a arquitetura de sistemas com foco na representação da plataforma computacional foram identificados e são apresentados em mais detalhes, na próxima seção.

3.2.2 Ontologia: Plataformas computacionais

A plataforma computacional é parte fundamental de qualquer arquitetura de sistema. Portanto, muitas ontologias incluem uma classe de plataforma e relacionamentos com componentes de hardware e software. Esforços na literatura têm sido amplamente utilizados para descrever plataformas de computação em diferentes domínios.

Um dos esforços mais consideráveis é a ontologia voltada a ambientes inteligentes apresentada por Preuveneers et al. Preuveneers *et al.* (2004). Ela inclui uma seção de plataforma que fornece a descrição do software que está disponível no dispositivo (sistema operacional, máquina virtual e middleware) e o hardware que especifica os recursos (energia, memória, CPU, armazenamento, rede) do dispositivo para fornecer um serviço em um domínio de ambiente inteligente. Entretanto, a ontologia carece de representação dos componentes do sistema, tal como interfaces de hardware (portas e barramentos de comunicação), conexões entre os dispositivos, conjunto de propriedades e implementação do sistema (alocação dos recursos).

A reconstrução de arquitetura (AR) é outro domínio que utiliza os benefícios das ontologias de plataforma computacional. A ontologia PLATOnt (MUSAVI; HASHEMI, 2019), inspirada no *framework AR Symphony*, apresenta uma extensa representação em alto nível de abstração dos componentes de um sistema de computação. Ao total, foram definidas seis superclasses e oitenta e quatro entidades que cobrem sistemas operacionais, hipervisores, firmware de CPU, memórias ROM e elementos virtuais. A ontologia PLATOnt é focada em sistemas de computação e apresenta uma exten-

siva base de conhecimentos de implementação de software e sistemas. Entretanto, a arquitetura de sistemas embarcados, foco desta tese, demonstra que a PLATOnt cobre parte da representação dos componentes de software (processos, tarefas e subprogramas), componentes da plataforma (sensores, atuadores e barramentos de comunicação), assim como os componentes e propriedades do sistema (conexões entre os componentes da plataforma e software, e alocação de recursos) dedicados a sistemas embarcados.

Outro domínio que integra o uso de AS é a internet das coisas Internet of Things (IoT). Porém, escolher quais ontologias pesquisar é uma tarefa desafiadora, pois existem mais de 200 ontologias que abordam um extenso conjunto de aplicações divididas em consumidor, comercial, industrial e infraestrutura (MILOSLAVSKAYA *et al.*, 2019). Em geral, a plataforma de computação é representada por um sistema composto por dispositivos (sensores, atuadores e tags). Um trabalho em andamento da associação de padrões do Instituto de Engenheiros Elétricos e Eletrônicos , do inglês Institute of Electrical and Electronics Engineers (IEEE), propõe o padrão P1451-99 para harmonização de dispositivos e sistemas IoT. Ele define um método para compartilhamento de dados, interoperabilidade e segurança de mensagens em uma rede, onde sensores, atuadores e outros dispositivos podem interoperar, independentemente da tecnologia de comunicação (LAPLANTE *et al.*, 2016). No entanto, as entidades para representar o software e os componentes de sistema não são cobertos e não fornecem suporte para implantação de plataforma.

3.2.3 Ontologia: Robótica e automação

A ontologia do domínio da robótica e da automação é solida e atualmente possui um padrão internacional IEEE 1872/2015 ou ontologia ORA (SCHLENOFF *et al.*, 2012), o qual é baseado no padrão internacional ISO 8373/2012 de vocabulário de dispositivos robóticos e robótica. A ORA possui um conjunto de extensões que descrevem conceitos voltados a representação de modelos de sistemas robóticos aplicados em diferentes domínios, tal como CORA e SUMO (NILES; PEASE, 2001; FIORINI, Sandro Rama *et al.*, 2015).

A ontologia SUMO é uma ontologia de nível superior que define categorias ontológicas básicas em todos os domínios relacionados à robótica e automação. O foco da SUMO é representar entidades físicas e abstratas do sistema robótico, tais como peças mecânicas, ambientes físicos e componentes arquiteturais em alto nível de abstração com objetivo de detalhar o comportamento temporal (NILES; PEASE, 2001). A ORA representa somente duas classes relacionadas aos componentes da plataforma e sistema da AS , `platform` e `system`, não cobrindo os demais componentes de software, implementação de sistemas e conjunto de propriedades.

As ontologias citadas demonstram que existe uma lacuna na representação de

arquitetura de sistemas, carecendo de informações sobre os componentes de software, plataforma e sistema, assim como detalhes quanto à implementação dos sistemas, conforme apresentado na Tabela 2. Com objetivo de solucionar as lacunas observadas no critérios avaliados, é proposta a construção de uma ontologia voltada ao domínio de arquitetura de sistemas, denominada OAS, apresentada em detalhes no capítulo5.

Existem muitas metodologias dedicadas à construção de uma ontologia. Dentre os pesquisados destacam-se os trabalhos de Uschold *et al.* (1996) que discutiu o papel das linguagens e técnicas formais na especificação, implementação e avaliação de ontologias; Fernandez *et al.* (1997) que apresentam um conjunto de atividades e técnicas de construção de ontologias; e o trabalho Noy, McGuinness *et al.* (2001) que apresentam as técnicas e os cuidados que os projetistas devem ter na declaração de entidades em busca de evitar sobreposição de conceitos e mal entendidos. Nestas metodologias mencionadas, geralmente, é incentivada a reutilização de módulos e ontologias visando manter um padrão de entidades, vocabulário comum e outros benefícios citados por W3C (JANOWICZ *et al.*, 2019). Deste modo, a ontologia proposta, deve seguir as boas práticas apresentadas nas metodologias pesquisadas e incrementar atividades que auxiliam na avaliação de reuso de ontologias e módulos.

Critérios Avaliados	OAS	SSN (COMPTON, M. *et al.*, 2012)	SOSA (JANOWICZ *et al.*, 2019)	(PREUVENEERS *et al.*, 2004)	PLATOnt (MUSAVI; HASHEMI, 2019)	P1451-99 (LAPLANTE *et al.*, 2016)	ORA (SCHLENOFF *et al.*, 2012)
Componentes de software	●	○	○	◑	◑	○	○
Componentes da plataforma	●	◑	◑	◑	◑	◑	◑
Componentes do sistema	●	◑	◑	○	◑	◑	◑
Implementação do sistema	◑	◑	◑	◑	●	◑	◑
Conjunto de propriedades	◑	◑	◑	◑	◑	○	○

Tabela 2 – Visão geral das ontologias levantadas
●Atende completamente, ◑Parcialmente e ○não atende.

3.3 ABORDAGEM DE TROCA DE S&A EM PROJETO DE CPS

Quanto à troca de componentes da arquitetura, foram identificadas abordagens relacionadas à troca de componentes da arquitetura, que cobrem os seguintes critérios: *modificação de hardware e software, análises das características e propriedades, inclusão de componentes, análise de requisitos e interatividade* necessárias para avaliar as modificações arquitetônicas.

Os critérios definidos foram baseados no trabalho de exploração de arquiteturas apresentado por Procter e L.Wrage (2019), tendo o objetivo de expandir a análise de arquiteturas que utilizam dispositivos de S&A. Os critérios adotados foram: *modificação de hardware, modificação de software, inclusão de componentes, análise de características, análise de propriedades, análise de requisitos.*

A *modificação de hardware* foi avaliada em busca de compreender como ocorrem as mudanças na interface do sistema ao efetuar a troca dos componente de hardware, isto é, dos dispositivos de S&A da arquitetura. A *modificação de software* foi avaliada para identificar quais mudanças afetam processos, tarefas e ligações do sistema.

O critério *inclusão de componentes*, especificadamente dispositivos de S&A, na arquitetura do sistema foi avaliado para verificar como ocorre a compatibilidade e as modificações necessárias na arquitetura. Verificou-se quais parâmetros foram utilizados para realizar as análises da arquitetura após a realização de modificações em seus componentes. Em *análise de características* e *análise de propriedades* foram identificados quais trabalhos utilizam estes parâmetros para avaliação da arquitetura.

O critério *análise de requisitos* tem por objetivo identificar quais trabalhos realizavam o mapeamento dos requisitos que foram impactados na mudança dos componentes da arquitetura. Sendo assim, verificou-se que é possível avaliar quais cenários possuem compatibilidade com o sistema.

Assim, nesta subseção são apresentados os trabalhos que abordam o conceito de exploração de espaço de projeto. Este conceito aborda o levantamento de possíveis cenários de arquitetura mediante a realização de mudanças em seus componentes, avaliando o impacto no sistema.

Os trabalhos que foram levantados realizam a exploração de cenários baseados em múltiplos atributos de qualidade, em sua grande maioria, para realização de análise de impacto no desempenho. As abordagens e ferramentas pesquisadas tiveram como foco a avaliação de troca de dispositivos, modificação, inclusão de componentes para evolução e exploração de arquiteturas candidatas descritas em ADL. Entretanto, é importante salientar que a linguagem AADL, foco de estudo desta tese, apresenta poucos trabalhos na literatura.

Destaca-se neste contexto a ferramenta ArcheOpterix (ALETI; BJORNANDER *et al.*, 2009), a qual se integra ao ambiente de desenvolvimento OSATE e utiliza algorit-

mos evolucionários para guiar os projetistas na exploração de arquiteturas candidatas, sendo capaz de avaliar a alocação de recursos da arquitetura gerando um conjunto de cenários possíveis para análise.

O *framework* do ArcheOpterix é composto por: *AADL Model parser (AMP)* que efetua a interpretação e extração das abstrações do modelo AADL especificado; *Architecture Analysis Module (AAM)* que atua como o banco de dados central contendo todas as informações relevantes especificadas no modelo AADL, restrições e atributos declarados e pode ser utilizado como *hub* comum à interface de aplicações específicas para serem implementadas por outros módulos; *Architecture Optimization (AO)* que estabelece uma estratégia baseada em padrões de projeto reutilizáveis para preparar a implementação de diferentes tipos de algoritmos otimizados. O algoritmo otimizado gera um conjunto de cenários baseado nos objetivos e pesos definidos aos atributos avaliados; *AADL Model Generator (AMG)* gera os modelos AADL dos cenários e possibilita também criar modelos a partir de um arquivo de entrada contendo os atributos declarados.

Foi observado que os autores propõem a avaliação de dois atributos de qualidade, o Data Transmission Reliability (DTR) e o Communication Overhead (CO) que são definidos a partir da formulação de equações baseadas nos parâmetros extraídos dos componentes da arquitetura, como: frequência de interação, tamanho da mensagem, confiabilidade da rede, atraso da rede e largura de banda entre os componentes. A descrição das equações são definidas no AAM que acessam os parâmetros do AMP, extraindo as declarações definidas nas propriedades dos componentes da arquitetura. Os atributos DRT e CO são calculados e fornecem os resultados para o AO que executa o algoritmo otimizado. O algoritmo otimizado processa os dados e gera um conjunto de cenários baseado nos objetivos e nos pesos definidos, alocando diferentes combinações de *deployment* da arquitetura, como por exemplo a alocação de processadores e componentes.

Neste sentido, a ferramenta ArchOpterix demonstra a capacidade de criar a avaliação de atributos de qualidade por meio da formulação de equações baseadas nas propriedades declaradas e geração dos cenários otimizados. Entretanto, a ferramenta demonstra restrições quanto à abrangência das análises dos cenários, como: as conexões de hardware entre os componentes devem ser compatíveis, avaliando somente modificações nos valores declarados nas propriedades dos componentes; as funcionalidades e tipos de componentes não são abordados, sendo que este atributo de qualidade é de grande relevância para a análise de compatibilidade e o uso de dispositivos de S&A na exploração da arquitetura; não são contemplados os componentes de software na ferramenta; as características dos componentes não são abordadas, somente realizando análise das propriedades declaradas no modelo AADL.

Outra ferramenta que aborda a análise e a exploração de arquiteturas candida-

tas, descrita em AADL e integrada ao ambiente OSATE, é a *Adventium's*, da empresa Adventium Labs (ADVENTIUM, 2020). Essa ferramenta assume como entrada uma planilha com opções de componentes com suas propriedades descritas em AADL, incluindo anexos. O projetista seleciona os componentes que o interessam, de forma manual, para então o sistema ser instanciado.

Após a instância da arquitetura, o projetista efetua manualmente as análises no ambiente OSATE, onde os resultados são armazenados em um arquivo. Este arquivo é aberto na ferramenta de exploração de espaço de projeto (ATSV) para visualização dos resultados obtidos nas análises realizadas. Uma vantagem da ferramenta *Adventium's* em relação à ArcheOpterix é a possibilidade de incluir análises relativas ao anexo de erros e anexo de falhas dos componentes.

Uma proposta mais recente para exploração de arquiteturas candidatas descritas em AADL, é apresentada por Procter e L.Wrage (2019): a ferramenta Guided Architecture Trade Space Explorer (GATSE). Essa ferramenta se integra ao ambiente OSATE e à ferramenta ATSV. A ferramenta GATSE propõe a especificação e configuração do sistema modelado, por meio de um arquivo com linguagem específica compatível com AADL, onde o usuário define quais elementos do modelo são candidatos à modificação, escolhendo as propriedades que serão alteradas e analisadas. Na escolha dessas propriedades, foram considerados os cinco graus de liberdade da arquitetura descritos por Anne Koziolek (2014), sendo eles: alterar um elemento; alterar mais de um elemento; selecionar as regras dos elementos que serão alterados (propriedades às quais estão associados); restrições de interação; e regras para definir os valores que os elementos do modelo selecionados podem assumir. O resultado das análises da ferramenta GATSE é apresentado com o auxílio do ATSV de forma interativa, permitindo aos projetistas a exploração dos graus de liberdade da arquitetura do sistema.

Resumindo as características principais da ferramenta GATSE, possibilitam a criação de cenários baseados em seis graus de liberdade e consideram que os componentes avaliados possuem a mesma funcionalidade e características; apresentam interatividade com o usuário habilitando o ajuste manual dos graus de liberdade; consideram que o projetista tenha realizado previamente no OSATE os testes de compatibilidade de conexões, ligações e dados.

Desta forma, esta tese busca expandir as análises de exploração de cenários propostos por Procter e L.Wrage (2019) com a avaliação da troca de dispositivos de S&A, incluindo os graus de liberdade funcionalidade, compatibilidade de hardware e software. Tendo como principal contribuição permitir a análise da qualidade de modificabilidade do sistema e seu impacto nos demais subsistemas e componentes da arquitetura.

Também foram levantados também trabalhos que não usam AADL mas que

abordam a modificação, alteração e exploração de arquiteturas. Kerzhner (2012) desenvolveu uma linguagem para representar problemas de exploração de arquiteturas baseado em SysML. Particularmente, o método descreve em detalhes o processo de busca que identifica potenciais trocas.

A proposta inclui a necessidade de visualização das trocas entre os elementos, analisando o desempenho, acurácia em um determinado espaço de exploração. Esse método utiliza uma coleção de declarações matemáticas para analisar o desempenho do sistema, sendo uma extensão do SysML. No que concerne à ferramenta, considerando as características principais, da ferramenta temos: realiza a seleção e criação de cenários com base nos atributos de qualidade; propõe a modificação de propriedades de desempenho; possui interatividade e auxilia na compra dos dispositivos avaliados. O autor não aborda a análise de características, propriedades, requisitos e funcionalidades dos componentes da arquitetura ou utilização de graus de liberdade.

O autor Iacobucci (2012), apresentou a metodologia Rapid Architecture Alternative Modeling - RAAM, sendo que a mesma destina-se à exploração de espaço de projeto no início do desenvolvimento do sistema. Verifica-se que a metodologia tem enfoque na avaliação do desempenho da arquitetura e redução de modelos complexos. A arquitetura é descrita usando o *framework* do Departamento de Defesa dos Estados Unidos, que utiliza uma linguagem específica para descrever as capacidades e gerar, por meio da ferramenta RAAM, as possíveis arquiteturas do sistema. Resumindo, as características principais da ferramenta RAAM, tem-se: limitado ao framework DoDAF; utiliza uma linguagem padrão de modelagem de alto nível; realiza a análise de desempenho da arquitetura com base nas propriedades declaradas; possui interatividade para visualização dos cenários. O autor não aborda a análise de requisitos e características baseadas nos atributos de qualidade dos componentes da arquitetura.

Outra ferramenta utilizada para exploração do espaço de soluções arquitetônicas é GuideArch (ESFAHANI *et al.*, 2013). Esta ferramenta fornece técnicas baseadas na matemática difusa (fuzzy) para ajudar os projetistas na tomada de decisões. Apresenta uma formalização que permite a comparação de arquiteturas com valores ambíguos de propriedades sob várias ponderações e restrições. Os autores desenvolveram a ferramenta baseada na web e discutiram aspectos positivos e negativos descobertos durante a avaliação. Considerando as características principais da ferramenta tem-se: não realiza a modificação de hardware e software da arquitetura; Não é fundamentada em uma linguagem descritiva de arquitetura de sistemas de baixo nível; propõe a realização de análises da arquitetura com foco em custo, tempo de desenvolvimento e uso da bateria; não cobre a análise de requisitos da arquitetura; Não possui interatividade com o projetista para seleção de objetivos ou critérios para direcionamento da avaliação de arquiteturas.

Anne Koziolek (2014) formulou, em sua tese, um método automatizado para

gerar e avaliar cenários com base nos graus de liberdade definidos no modelo arquitetural descrito em Palladio Component Model (PCM) (BECKER, S.; KOZIOLEK, H.; REUSSNER *et al.*, 2009). O método fornece suporte na fase inicial de desenvolvimento da arquitetura do sistema, avaliando potenciais cenários a partir das propriedades de desempenho da arquitetura. Entretanto, o método proposto restringe a aplicação para modelos descritos em PCM, limitando a realização de análises com foco nas propriedades de desempenho dos componentes e do alto grau de abstração quando comparado com o AADL.

Foi proposta a realização de previsões de qualidade e exploração automatizada de cenários por meio da análise de um conjunto de propriedades, denominado "graus de liberdade". Os graus de liberdade são baseados nos atributos de qualidade e representam as propriedades do tipo temporal, custo, consumo de potência e desempenho da arquitetura. A autora destaca seis graus de liberdade que podem ser utilizados para avaliar a exploração de cenários. Cada grau de liberdade representa um tipo de mudança no modelo arquitetural do sistema, tais como: seleção de componentes, definição de parâmetros de configuração, alocação de software para o hardware, adicionar novos elementos, regras para modificações, regras para seleção dos componentes dentre outros.

Entretanto, não foram abordados na exploração de cenários e troca de componentes da arquitetura, assim como graus de liberdade que habilitem a análise mais detalhada dos atributos de qualidade do sistema, especificadamente a característica de modificabilidade da arquitetura (ISO/IEC, 2011). Graus de liberdade como interface, conexões e barramentos do hardware não foram explorados, assim como as modificações nas ligações e conexões dos componentes de software. Esses graus de liberdade são importantes para realizar a exploração das características e funcionalidades dos dispositivos de S&A.

Ross *et al.* (2019) apresentaram uma ferramenta e linguagem para exploração de arquiteturas automotivas nos mais diversos níveis de abstração e perspectivas. O trabalho utiliza a linguagem Clafer que possibilita a modelagem em múltiplas perspectivas com foco na descrição das características e atributos de qualidade de sistemas automotivos.

Os autores desenvolveram um conjunto de padrões/regras para avaliar a exploração de possíveis cenários e atributos de qualidade e realizaram um estudo comparativo das ferramentas de modelagem de arquitetura desenvolvidas em trabalhos anteriores. Foram listadas as seguinte ferramentas: AcheOpterix, AutoFOCUS-AF3, OSATE e AAOL.

Nota-se que os autores buscaram comparar a capacidade das ferramentas de realizar a exploração de arquiteturas, usando como referência o modelo de arquitetura EAST-ADL para modelagem automotiva (elétrica e eletrônica), encontrando assim

muitas limitações na modelagem devido a forma como EAST-ADL propõe a modelagem das características, funções, conectores, dispositivos e deployment. Não ficou claro como os autores realizaram a busca e a definição do espaço de exploração dos componentes candidatos a troca, sendo apresentado somente um conjunto de quinze possíveis arquiteturas condicionadas às regras definidas e treze possíveis restrições de atributos de qualidade da arquitetura. Resumindo, as características principais da ferramenta Clafer, tem-se: geração de cenários conforme regras criadas previamente; efetua a modificação de hardware e funcionalidades dos dispositivos; permite a análise das características e propriedades com foco nos atributos de qualidade peso, custo e desempenho declarados nas propriedades dos dispositivos; possui interatividade/ suporte à compra dos dispositivos avaliados (mediante definição das regras que limitam do espaço de exploração).

Sendo assim, esta tese propõe o uso da abordagem de evolução de arquiteturas por meio da ferramenta *DevCompatibility* considerando o foco em arquiteturas que utilizam intensivamente dispositivos de S&A e que eventualmente tenham a necessidade de modificar ou incluir novos componentes e funcionalidades, mantendo o atendimento aos requisitos de projeto. A ferramenta é baseada em MDE e foi desenvolvida como um plug-in do OSATE2, tendo como entrada o modelo AADL do sistema em questão, biblioteca de potenciais dispositivos candidatos também descritos em AADL e um modelo de requisitos do sistema atual descrito em linguagem *ReqSpec*.

O *DevCompatibility* implementa parte das etapas do método CAvA, apresentado no próximo capítulo, e fornece suporte à aplicação da abordagem de troca de dispositivos de S&A em modelos AADL. A ferramenta será mostrada em mais detalhes no capítulo 6.

A Tabela 3 apresenta uma visão geral dos trabalhos relacionados de acordo com os critérios avaliados. Vale destacar que o critério de modificação de hardware está diretamente ligado a realização de análises de características da arquitetura, compatibilidade de interface, portas e conexões com a arquitetura. Outro critério importante para realização de evoluções da arquitetura, a *inclusão de componentes* busca salientar a análise de integração de novos componentes e de suas modificações de hardware e software para garantir o cumprimento dos requisitos.

Critérios Avaliados	DevCompatibility	(PROCTER; L.WRAGE, 2019)	(ADVENTIUM, 2020)	(ROSS *et al.*, 2019)	(IACOBUCCI, 2012)	(ESFAHANI *et al.*, 2013)	(KERZHNER, 2012)	(ALETI; BJORNANDER *et al.*, 2009)
Modificação de hardware	●	◑	◑	●	●	◑	◑	◑
Modificação de software	●	○	◑	○	○	◑	○	○
Análises de propriedades	◑	●	●	◑	●	◑	○	●
Análises de características	●	○	●	◑	○	○	○	○
Inclusão de componentes	○	○	◑	◑	○	○	○	○
Análise de requisitos	◑	○	○	○	○	○	○	○
Interatividade	●	●	●	●	●	○	◑	●

Tabela 3 – Visão geral das abordagens pesquisadas.
●Atende completamente, ◑Parcialmente e ○não atende.

4 ABORDAGEM DE EVOLUÇÃO DE ARQUITETURAS DE CPS

Neste capítulo é apresentada uma abordagem de evolução de arquiteturas, denominada CPS Architecture Evolution Approach (CAvA) (**C**PS **A**rchitecture E**v**olution **A**pproach), que busca guiar os projetistas no processo de troca de componentes de S&A da arquitetura para que garanta a integração e a compatibilidade do dispositivo candidato com o CPS. A abordagem CAvA consiste em um conjunto de fases e atividades que orientam os projetistas na exploração de cenários, avaliação e análise dos requisitos de arquitetura, assim como os atributos de qualidade impactados pela alteração de um ou mais dispositivos S&A da arquitetura.

Além disso, a proposição usa conceitos de MDE a fim de possibilitar a exploração de cenários, fornecendo suporte às ferramentas de modelagem, avaliação e análise realizadas nas atividades propostas.Sendo assim foi Integrado a proposta, um conjunto de tecnologias (AADL e ReqSpec) e técnicas para conduzir o desenvolvimento da abordagem com modelos. A AADL oportuniza descrever em detalhes a AS com intuito de implementação, enquanto a ReqSpec foca na documentação dos requisitos de projeto relacionados aos modelos descritos em AADL. A integração entre estas linguagens proporciona ao projetista o rastreamento dos requisitos em nível de componentes da arquitetura, facilitando a avaliação e a análise de modificações estruturais.

A abordagem proposta é composta de cinco etapas: planejamento, exploração de componentes candidatos, análise das arquiteturas candidatas a evolução, otimização de arquitetura e arquitetura evoluída consolidada. Cada etapa possui um conjunto de quatro atividades que guiam os projetistas no desenvolvimento da evolução da arquitetura. O resultado do conjunto de atividades, realizado em cada etapa, alimenta a etapa seguinte conforme apresentado na Figura 8. Ao final da última etapa o projetista pode realizar recursivamente novas evoluções da arquitetura atual.

A etapa 1, *planejamento e requisitos do sistema* é composta por quatro atividades: (1) planejar Evolução e aposentadoria; (2) elicitar requisitos; (3) análise dos requisitos específicos e (4) análise dos resultados. A etapa 2, *Exploração de arquiteturas candidatas*, é composta por quatro atividades, sendo elas: (2.1) definição dos dispositivos de S&A da arquitetura; (2.2) definição dos dispositivos de S&A candidatos; (2.3) elicitar atributos de qualidade; (2.4) exploração de cenários; A etapa 3, *Análise das arquiteturas candidatas*, é composta por quatro atividades: (3.1) análise de hardware e software; (3.2) análise de componentes impactados; (3.3) análise dos requisitos impactados; (3.4) análise dos atributos de qualidade. A etapa 4, *Otimização da arquitetura*, é composta por: (4.1) definição múltiplos objetivos; (4.2) definição de pesos dos objetivos; (4.3) ranqueamento dos cenários; (4.4) seleção da arquitetura candidata; A etapa 5, *Arquitetura consolidada escolhida*, é composta por quatro atividades:

Figura 8 – Estrutura de etapas da abordagem proposta.
Fonte: Figura do autor.

(5.1) testes na arquitetura candidata; (5.2) atualização de requisitos específicos; (5.3) atualização do plano de evolução e aposentadoria; (5.4) arquitetura consolidada.

A aplicação da CAvA requer que o projetista possua os arquivos de modelagem da AS e requisitos com intuito de realizar as atividades proposta de evolução de CPS. Uma das formas de representar o relacionamento entre o processo físico, rede e plataforma, é através de modelos baseados em ontologia, conforme mostrado no Capítulo 2. Desta forma, o projetista pode identificar inconsistências de projeto e avaliar o impacto de modificações em componentes da arquitetura e processo físico.

Com intuito de expandir a capacidade de representar os componentes da AS, no capítulo 5 é apresenta uma proposta de ontologia voltada ao domínio de AS (OAS). A OAS tem por objetivo de prover uma base de conhecimento para representação dos componentes da arquitetura utilizados em projetos de CPS. Desta forma, o projetista pode realizar a modelagem da arquitetura em ontologia para realizar a aplicação da abordagem CAvA. O Capítulo 6 apresenta em detalhes a aplicação das etapas de 2 a 4 da CAvA através da troca de dispositivos de S&A com modelos AADL e ReqSpec,

A seguir são apresentadas em mais detalhes as etapas e atividades propostas.

4.1 ATIVIDADES PARA EVOLUÇÃO DA ARQUITETURA DE SISTEMAS

A abordagem proposta inicia com a fase de planejamento, onde os modelos da arquitetura, requisitos de projeto e dados da AS foram definidos e validados. Posteriormente, são realizadas em quatro atividades descritas a seguir.

Etapa 1 ***Planejamento e requisitos do sistema:*** *Nesta etapa é realizado o levantamento dos requisitos de projeto, requisitos específicos que descrevem o comportamento e funcionamento da arquitetura do sistema, dados de validação e desempenho das simulações e testes em modelos computacionais do sistema atual para*

análise da capacidade de evolução e aposentadoria.

Atividade 1.1 ***Planejar Evolução e Aposentadoria:*** *Nesta atividade são levantados os objetivos das partes interessadas, restrições e limitações definidas no projeto para atender as exigências da aplicação. As informações levantadas devem direcionar os projetistas no levantamento dos requisitos de projeto que devem ser modificados e potenciais dispositivos candidatos.*

O Plano de Evolução e Aposentadoria (PEA) avalia periodicamente as potenciais melhorias e inclusões de novas funcionalidades com base no avanço tecnológico e pesquisa de componentes disponíveis comercialmente, dando início ao registro da necessidade de efetuar modificações e inclusão de novos requisitos no projeto.

O PEA contém um descritivo do ciclo de vida do projeto, quantidade de versões realizadas e quais modificações foram feitas a cada versão. O registro de cada modificação efetuada busca auxiliar os projetistas na rastreabilidade de eventuais problemas que possam ocorrer na arquitetura. Apesar de realizar testes e simulações exaustivas, a inclusão de novas funcionalidades, troca de dispositivos de S&A similares provenientes de diferentes fabricantes, pode apresentar problemas durante seu ciclo de vida. O PEA apresenta um histórico técnico e fundamental que auxilia os projetistas na identificação e rastreabilidade de modificações.

Atividade 1.2 ***Elicitar requisitos:*** *nessa atividade são levantados os requisitos de projeto e modelos computacionais que descrevem as funcionalidades e comportamento do sistema em operação, identificando os candidatos requisitos que serão incluídos ou atualizados.*

Os requisitos de projeto relacionados com a troca/inclusão dos dispositivos de S&A são levantados. Assim, o projetista avalia os requisitos e identifica o conjunto de funcionalidades, requisitos funcionais e não funcionais específicos atrelados à modificação. A inclusão ou atualização dos requisitos pode impactar diretamente na arquitetura, sendo realizada uma análise detalhada das especificações dos componentes e composição do sistema.

Atividade 1.3 ***Análise dos requisitos específicos:*** *nessa atividade as funcionalidades, requisitos funcionais e não funcionais representados em modelos computacionais e relacionados aos dispositivos de S&A que deseja-se modificar são analisados.*

Os requisitos funcionais e não funcionais da arquitetura em operação, relacionados com a troca dos dispositivos de S&A, são analisados com objetivo de identificar a relação com os componentes de software, hardware e sistema, características e propriedades (atributos de qualidade) da arquitetura que podem necessitar de modificações. Para os casos em que novos requisitos e funcionalidades são incluídos, o projetista deve indicar os componentes adicionados à arquitetura.

Atividade 1.4 ***Analisar resultados das simulações e análises da arquitetura:*** *nessa atividade são obtidos os resultados das simulações e análises de desem-*

penho realizadas em simulações de modelos arquitetônicos e testes do sistema em operação. Os dados obtidos nesta atividade servirão para analisar a capacidade de modificações e exploração de arquiteturas candidatas.

Os resultados obtidos em simulações, testes de bancada, validação e análises de arquitetura são usados como referência para avaliar os potenciais cenários de evolução arquitetônica. Esses dados são usados no modelo de arquitetura para uma análise mais aprofundada dos atributos de qualidade, implantação e parâmetros atuais da arquitetura.

Dessa forma, os projetistas podem ter uma visão geral da capacidade de realizar modificações na arquitetura, identificando as restrições de projeto que devem ser respeitadas para atender a troca dos dispositivos de S&A sem comprometer os atributos de qualidade do sistema.

Etapa 2 ***Exploração de componentes candidatos:*** *nessa etapa ocorre o levantamento dos dispositivos de S&A da arquitetura atual e candidatos para analisar a compatibilidade de interface, modificações a serem realizadas na arquitetura do sistema e requisitos que devem ser incluídos ou modificados.*

Atividade 2.1 ***Definição dos dispositivos de S&A da arquitetura:*** *nessa atividade são identificados os dispositivos candidatos de S&A da arquitetura em operação que devem ser substituídos.*

Os projetistas efetuam a seleção e definição dos dispositivos de S&A que devem ser substituídos na arquitetura. É importante destacar que a seleção de dispositivos com múltiplas funcionalidades deve ser observada para que os dispositivos candidatos atendam. Considerando a aplicação baseada em MDE, o modelo arquitetônico atual é utilizado como entrada desta atividade para que sejam selecionados os componentes a serem trocados.

Atividade 2.2 ***Definição dos dispositivos de S&A Candidatos:*** *nesta atividade são definidos os dispositivos de S&A candidatos a substituição ou inclusão na arquitetura do sistema em operação.*

A equipe de projetista efetua uma pesquisa técnica de mercado dos dispositivos de S&A comerciais disponíveis com características físicas, funcionalidade e propriedades compatíveis com a arquitetura do sistema e que atendam os novos requisitos do projeto. Após definir os candidatos, o projetista efetua a modelagem para posterior análise de compatibilidade de interface do hardware, software e atributos de qualidade que se deseja avaliar. Cada componente candidato é modelado para compor a biblioteca que deverá ser utilizada no processo de troca.

Caso sejam definidos novos requisitos no projeto que exigirão a inclusão de dispositivos de S&A na arquitetura, o projetista deverá efetuar a modelagem das características e propriedades para posterior análise da compatibilidade com a plataforma embarcada e arquitetura do sistema.

Atividade 2.3 ***Elicitar os atributos de qualidade:*** *Os atributos de qualidade relacionados aos componentes de S&A candidatos são elicitados. As características, propriedades físicas, elétricas, de interface de hardware e de software são identificadas.*

Os atributos de qualidade do sistema são estruturados com base nas oito características descritas no padrão internacional ISO/IEC 25010, sendo elas: adequação funcional, confiabilidade, eficiência de desempenho, compatibilidade, usabilidade, segurança, manutenção e portabilidade (ISO/IEC, 2011). Cada característica possui um conjunto de sub-características e atributos de qualidade que podem ser declarados no escopo do modelo arquitetônico para exploração de arquiteturas candidatas.

Atividade 2.4 ***Exploração de cenários:*** *nessa atividade os cenários são explorados com base nas funcionalidades e atributos, levantando a compatibilidade da interface de hardware dos dispositivos candidatos com o sistema.*

A exploração de arquiteturas ocorre em três partes. Na primeira é feita a avaliação das funcionalidades dos candidatos compatíveis com os antigos dispositivos, interface disponíveis na arquitetura e plataforma embarcada. Na segunda parte é gerado um conjunto de cenários que abrangem as funcionalidades selecionadas e as conexões de interface com a arquitetura. Os dispositivos de S&A que possuem múltiplas funcionalidades integradas podem ser trocados por um ou mais componentes, portanto que preencham no mínimo as funcionalidades do antigo dispositivo, gerando assim um conjunto de cenários com dispositivos combinados. Na terceira parte ocorre a geração de cenários com dispositivos candidatos que não possuem interface compatível com a arquitetura, sendo sinalizada a necessidade de adicionar dispositivos conversores e conexões novas para realizar a compatibilidade.

É importante destacar que o objetivo dessa atividade é explorar a maior quantidade de cenários possível para posterior avaliação, respeitando a premissa de que os dispositivos candidatos e combinações devem possuir no mínimo a mesma funcionalidade do dispositivo selecionado para troca. Dessa forma, o processo de otimização de arquitetura pode avaliar uma maior quantidade de candidatos, sem que ocorra uma filtragem nas etapas iniciais de geração de cenários.

Etapa 3 ***Análise das arquiteturas candidatas:*** *nessa etapa o projetista analisa os cenários gerados, com ênfase nas características de hardware e software, conformidade com os requisitos, componentes impactados e atributos de qualidade.*

Atividade 3.1 ***Análise do Software e hardware:*** *a análise dos componentes de software e hardware dos cenários gerados em comparação com a arquitetura selecionada é realizada.*

Desse modo, os cenários gerados são analisados e comparados com a arquitetura atual, identificando diferenças e similaridades entre os componentes de hardware e software, tais como: portas, barramento de comunicação e conexão, tarefas, pro-

cessos e tipo de dados. Em casos onde os dispositivos S&A candidatos possuem internamente componentes de software e hardware dedicados a funcionalidades específicas, como o de tratamento de sinais, que até então eram executados na plataforma do sistema embarcado. Com isso, a análise de componentes impactados, redistribuição de recursos, remoção ou ajustes nos componentes, integração de elementos como wrappers são propostos na próxima atividade.

Atividade 3.2 ***Análise dos componentes impactados:*** *nessa atividade são realizadas análises de impacto das modificações nos demais componentes da arquitetura. A troca dos dispositivos de S&A podem ocasionar mudanças que impactam nos subsistemas, dispositivos de atuação e dados, sendo realizado uma análise de quais componentes do sistema são impactados.*

Os cenários gerados são comparados com a arquitetura selecionada para identificar os componentes de hardware e software que foram afetados pela troca. Sendo efetuado o levantamento dos componentes que sofreram alterações quando comparado com a arquitetura selecionada. Inicialmente o levantamento ocorre com os componentes de hardware, tais como interface, conexões, fonte de alimentação e barramento de comunicação; componentes de software, tais como conexões, tipos de dados, configuração da velocidade do barramento de comunicação, processos e tarefas que sofreram modificações para se adequar à integração do novo dispositivo, identificando as diferenças e fornecendo informações extraídas dos cenários para compor um conjunto de pontuações que apresentam o impacto da modificação.

Atividade 3.3 ***Análise de requisitos impactados:*** *nessa atividade são realizados o levantamento e análise dos requisitos do sistema que foram impactados com a troca dos dispositivos de S&A. Cada cenário gerado possui o resultado das análises das características de qualidade e são confrontados com os requisitos funcionais e não funcionais do sistema.*

Nessa atividade, ocorre a análise dos requisitos funcionais e não funcionais que foram impactados com a mudança do dispositivo de S&A. Cada requisito correspondente ao componente modificado fornece uma referência das características e valores dos atributos de qualidade do sistema que serão analisados na próxima atividade.

Atividade 3.4 ***Análise dos atributos de qualidade dos cenários:*** *nessa atividade ocorre a análise dos atributos de qualidade dos cenários gerados em comparação com a arquitetura selecionada.*

A análise dos atributos de qualidade, ou propriedades, vinculados aos dispositivos S&A, tais como: precisão, resolução, relação do sinal, ruído, sensibilidade, período, massa, torque de saída, operação de temperatura são comparados, mensurando as diferenças para compor um conjunto de dados, estruturados de acordo com as definições realizadas na atividade 2.3.

Com isso, após as análises das atividades da fase 3, obtemos um conjunto de

dados dos cenários gerados, em comparação com a arquitetura selecionada, que nos fornece subsídio para realizar a otimização da arquitetura proposta na próxima etapa.

Etapa 4 ***Otimização da arquitetura:*** *nessa atividade ocorre a definição de múltiplos objetivos, pesos para ranqueamento dos cenários com base nos resultados das análises obtidas na fase 3.*

Os atributos de qualidade dos cenários extraídos e os parâmetros dos requisitos de design quantificáveis são apresentados ao projetista para aplicação de técnicas de tomada de decisão, como Processo de Hierarquia Analítica, do inglês Analytical Hierarchy Process (AHP). Os dados quantificáveis dos atributos são ranqueados de acordo com a lógica de projeto desejado. Cada atributo pode ser ranqueado de acordo com um valor máximo ou mínimo. Por exemplo, o projetista pode definir que o ranqueamento dos cenários deve possuir o atributo preço deve ser classificado com base no menor valor, enquanto o atributo de capacidade do processador pode ser classificado no ranqueamento como o maior valor.

Atividade 4.1 ***Definição dos múltiplos objetivos:*** *O projetista define quais objetivos devem ser avaliados com base nos atributos de qualidade obtidos.*

Nessa atividade ocorre a seleção dos múltiplos atributos de qualidade que serão utilizados no ranqueamento dos cenários. Com base nas análises da etapa 3, cada cenário fornece informações sobre a quantidade de componentes, tipo de interface de hardware (portas e barramentos), dispositivos conversores adicionados, wrappers e conexões, componentes impactados e atributos de qualidade quando comparados com a arquitetura selecionada. Desta forma, o projetista seleciona um ou mais objetivos que deseja utilizar pra ranqueamento dos cenários.

Atividade 4.2 ***Definição de pesos dos objetivos:*** *nessa atividade o projetista define os pesos de cada atributo selecionado com base nos objetivos os critérios de projeto que deseja avaliar.*

De acordo com os objetivos definidos para a exploração dos cenários, o projetista define pesos aos atributos para classificar os cenários.

Atividade 4.3 ***Ranqueamento dos cenários:*** *no que concerne à classificação dos cenários é realizada de acordo com os objetivos e pesos definidos e técnica de otimização.*

A ranqueamento dos cenários é realizado a partir do tratamento matemático dos atributos e dos pesos definidos. Como resultado, os cenários são ranqueados de acordo com os múltiplos objetivos e pesos definidos pelo projetista. O tratamento pode ocorrer por meio do uso de técnicas de otimização. Por exemplo, a técnica AHP propõe a normalização dos valores numéricos dos atributos de qualidade. Com a AHP os pesos são definidos em uma escala de 0 a 1, sendo a somatória dos pesos igual a 1.

Atividade 4.4 ***Seleção da arquitetura evoluída:*** *nessa atividade, o projetista*

seleciona a arquitetura evoluída que deseja implementar.

Nessa atividade o projetista seleciona, dentre o ranqueamento de cenários gerado, qual cenário deseja implementar, dando sequência a um conjunto de testes, simulações e avaliações da arquitetura não cobertas na análise dos atributos de qualidade elicitados.

Etapa 5 ***Geração da arquitetura evoluída consolidada:*** *Nessa etapa são definidos os novos parâmetros de desempenho da arquitetura evoluída, atualização dos requisitos específicos bem como dos requisitos de projeto.*

Nessa etapa ocorre a realização de testes e validação da arquitetura evoluída para posterior atualização dos requisitos de projeto e desempenho do sistema.

Atividade 5.1 ***Testes e Validação:*** *nessa atividade são realizados os testes e validação da arquitetura evoluída.*

São sugeridos o uso de técnicas como: *Hardware in-the-loop* (HIL), *Software in-the-loop* (SIL) e *processor in-the-loop* (PIL) para extrair em ambiente real simulado dados de comportamento e desempenho da arquitetura evoluída. Os resultados dos testes são usados para alimentar, recursivamente, as fases 2 e 3, refinando o modelo arquitetônico e realizando novas análises.

Atividade 5.2 ***Atualização de requisitos de projeto:*** *Nessa atividade ocorre a atualização dos requisitos específicos de projeto obtidos a partir dos dados de desempenho da arquitetura e testes realizados.*

Nessa atividade são validados os novos requisitos de projeto obtidos a partir da análise de desempenho da arquitetura evoluída com a definição do novo *deployment*. As modificações e inclusão de novas funcionalidades que culminaram no cumprimento dos novos requisitos de projeto são atualizadas no PEA para posterior avaliação de melhorias no sistema. Os novos requisitos de design obtidos a partir da análise de desempenho da arquitetura evoluída com a definição da nova implementação são validados.

Atividade 5.3 ***Atualização do plano de evolução e aposentadoria:*** *Nesta atividade ocorre a documentação de todas as alterações realizadas na arquitetura evoluída. As modificações realizadas são registradas no PEA, requisitos de projeto e modelo da arquitetura para posterior evolução ou definição de aposentaria do sistema.*

Nessa atividade são levantados todos os registros de troca dos dispositivos de S&A da arquitetura, assim como as definições de múltiplos objetivos e pesos utilizados na escolha do novo cenário. Esse registro provém do ciclo de atualizações e modificações dos componentes da arquitetura e pode ser utilizado para estimar o momento de aposentadoria e início de um novo projeto.

Atividade 5.4 ***Arquitetura consolidada:*** *nessa atividade o projetista efetua a atualização da arquitetura atual para o novo cenário selecionado. Novas evoluções/-modificações na arquitetura podem ocorrer agora tendo como base de comparação a*

nova arquitetura consolidada.

Nesta atividade o projetista efetua a consolidação da arquitetura evoluída após terem sido realizadas as demais atividades propostas. A arquitetura evoluída é definida como a arquitetura atual do sistema, sendo indicada a sua versão de atualização. Novas evoluções e modificações podem ser realizadas tendo como base a arquitetura consolidada. Assim, o projetista pode efetuar novas atualizações durante o ciclo de vida do sistema, controlando a versão da arquitetura e estimando através das funcionalidades e modificações a serem realizadas, quando pode ser realizada a aposentaria do sistema.

A abordagem proposta visa orientar as equipes de projetistas no planejamento e desenvolvimento de modificações arquitetônicas em VANTs e suas aplicações. Uma sequência de etapas e atividades foi apresentada para obter as características e propriedades da arquitetura de uma aeronave, fornecendo meios para analisar, selecionar e definir uma arquitetura candidata de acordo com os objetivos do projeto.

Durante o processo de evolução, vários modelos são criados especificando os dispositivos de S&A candidatos, arquitetura atual, requisitos do sistema e potenciais cenários de evolução. Com isso, os projetistas definem os objetivos da nova arquitetura, gerando um conjunto de documentos que rastreiam as modificações realizadas. Finalmente, com a escolha da arquitetura os projetistas efetuam testes e simulações com objetivo de validar a construção e implementação da nova arquitetura.

Essa abordagem é aplicada no projeto de um VANT do tipo VTOL-CP para ilustrar a troca de dispositivos de S&A da arquitetura com objetivo de aprimorar um conjunto de atributos de qualidades definidos por projetistas. A seguir, é apresentado um estudo de caso com o desenvolvimento das etapas e atividades propostas.

4.2 PROJETO DE EVOLUÇÃO DE UM VANT-CP

Em busca de fornecer uma melhor perspectiva quanto à aplicação da abordagem proposta, um projeto de VANT VTOL-CP foi desenvolvido. A execução da abordagem é descrita em detalhes com a evolução da arquitetura de uma aeronave.

O projeto do VANT VTOL-CP é, na verdade, parte de um esforço de pesquisa denominado ProVant. Este projeto envolve duas instituições de pesquisa brasileiras, a Universidade Federal de Minas Gerais (UFMG) e a Universidade Federal de Santa Catarina (UFSC), em colaboração com a Universidade de Sevilha (Espanha).

No contexto do ProVant, a equipe de projetistas desenvolveu um VANT aplicado a missões de busca e resgate (*Search and Rescue* - SAR). Nesse tipo de missão o uso dessas aeronaves são importantes para situações onde o resgate ocorre em regiões de difícil acesso, fornecendo assistência adequada e ágil. As missões de SAR têm algumas características que precisam ser levadas em consideração no projeto VANT. Essas características incluem dimensões reduzidas da aeronave, versatilidade para

executar missões SAR e facilidade de uso.

O ProVant 1.0, apresentado na Figura 9, foi o primeiro contato com o desenvolvimento de um VANT-VTOL para fins acadêmicos que possui unicamente o modo de voo "hover". Nesse projeto, as instituições envolvidas tiveram como foco avaliar o comportamento da estrutura mecânica, controladora de voo, modos de operação, integração de fontes renováveis e desempenho e segurança da arquitetura do sistema.

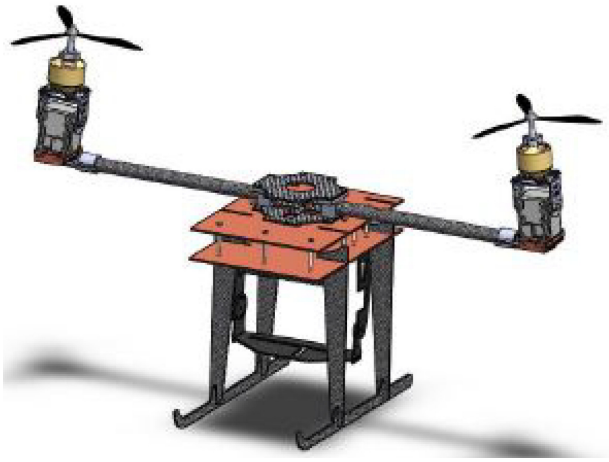


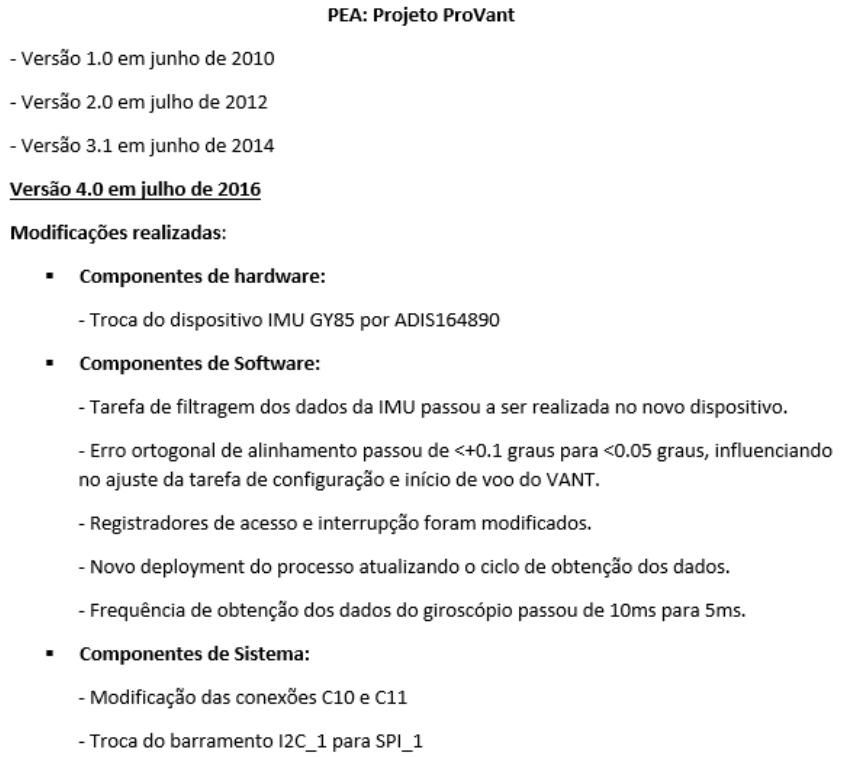
Figura 9 – modelo CAD do ProVant 1.0.
Fonte: Projeto ProVant (UFSC, 2020).

Desde o início do ProVant, quatro protótipos de aeronaves foram projetados. Além de aprimorar as propriedades de voo, cada nova versão visou solucionar os problemas encontrados no projeto predecessor, incrementando funcionalidades e realizando ajustes na AS. A aeronave atualmente está na sua quarta geração, e este estudo colaborou para refinar o projeto tanto da versão 3.0 quanto da versão 4.0 em busca de avaliar cenários e guiar os projetistas na definição de uma nova versão (versão 4.1) a ser construída.

Aplicando a abordagem proposta, dá-se início à etapa de planejamento, **Etapa 1**), com o levantamento dos requisitos de projeto, dos requisitos específicos definidos com base na aplicação e das características do VANT da versão atual. Tem-se como resultado, documentos do PEA, modelos da arquitetura descritos em AADL e de requisitos em ReqSpec, assim como as análises prévias de testes e simulações que validam a atual arquitetura e cumprimento dos requisitos.

Inicialmente, o documento do PEA é obtido para avaliar o histórico de modificações nos componentes de hardware da arquitetura ProVant 4.0 e seu ciclo de vida, conforme descrito na **Atividade 1.1**. Dessa forma, os projetistas podem obter os registros técnicos das modificações realizadas na arquitetura, versões e datas de operação do sistema. A Figura 10, apresenta um exemplo do PEA ProVant indicando as modificações realizadas durante o ciclo de vida do projeto.

Na sequência ocorre a elicitação dos requisitos de projeto, **Atividade 1.2**. Nesse



PEA: Projeto ProVant

- Versão 1.0 em junho de 2010

- Versão 2.0 em julho de 2012

- Versão 3.1 em junho de 2014

Versão 4.0 em julho de 2016

Modificações realizadas:

- **Componentes de hardware:**
  - Troca do dispositivo IMU GY85 por ADIS164890
- **Componentes de Software:**
  - Tarefa de filtragem dos dados da IMU passou a ser realizada no novo dispositivo.
  - Erro ortogonal de alinhamento passou de <+0.1 graus para <0.05 graus, influenciando no ajuste da tarefa de configuração e início de voo do VANT.
  - Registradores de acesso e interrupção foram modificados.
  - Novo deployment do processo atualizando o ciclo de obtenção dos dados.
  - Frequência de obtenção dos dados do giroscópio passou de 10ms para 5ms.
- **Componentes de Sistema:**
  - Modificação das conexões C10 e C11
  - Troca do barramento I2C_1 para SPI_1

Figura 10 – Exemplo de PEA do ProVant 4.0.

contexto, o OSATE fornece o anexo e o plug-in da ferramenta ALISA que integra elementos para especificação e modelagem de requisitos textuais com a linguagem *ReqSpec*. O código 4.1 apresenta um exemplo da modelagem dos requisitos do ProVant 4.0 com a linguagem *ReqSpec*, sendo definidos os requisitos do sistema, valores e unidades de medida (linhas 4, 10, 16 e 25), categorias (linhas 5, 11, 17 e 22) e descrição (linhas 2, 6).

```
1  system requirements scfreqs for SimpleControlSystem::SCS [
2 description "These are requirement for SCS of ProVant4.0"
3 requirement R1 : "SCF weight limit" [
4  val  MaximumWeight = 1.2 kg
5  category Quality.Mass
6  description this " shall be within weight of " MaximumWeight
7  value predicate MaximumWeight == #SEI::WeightLimit
8  see goal SCFgoals.ng1]
9 requirement R2 : "SCF inlet power" for SimpleControlSystem::SCS [
10  val MaximumPowerBudget = 5.0 W
11  category Quality.Performance
12  compute actualvolt: Physical::Voltage_Type
13  value predicate MaximumPowerBudget == #SEI::PowerBudget
14  see goal SCFgoals.g2]
15 requirement R3 : "SCF period" for sensor1 [
16 val period = 50.0
17 category Quality.Performance
18 compute actuaperiod: period
19 value predicate period == actualperiod
20 see goal SCFgoals.g3]
21 requirement R4 : "Failure probability" for sensor1 [
22 category Quality.Reliability
```

```
23 compute actualprob: real
24 compute specprob : real
25 value predicate specprob < 0.00004]
```

Código 4.1 – Requisitos modelados em Resqpec

Após elicitar os requisitos, o time de projetistas realiza uma análise dos requisitos específicos, buscando relacionar com os componentes da arquitetura e com suas propriedades, conforme descrito na **Atividade 1.3**.

É importante destacar que a realização de análises na arquitetura, proposto na **Atividade 1.4**, deve ser feita com base no modelo arquitetural em OWL previamente desenvolvido no projeto. Desse modo, com intuito de facilitar a compreensão do leitor, é apresentada a Tabela 4 do conjunto de componentes de hardware utilizados na arquitetura da aeronave ProVant 4.0 modelados em OWL.

Tabela 4 – Componentes de hardware do ProVANT 4.0.

Qnt	Componentes	Tipo	Descrição/Funcionalidade
2	Placa STM32F767ZI	Placa de desenvolvimento	Placa de desenvolvimento com microcontrolador STM32F767ZI.
1	Placa Jetson TX2	Placa de desenvolvimento	Placa de desenvolvimento com microcontrolador Nvidea.
1	Navio 2	HAT piloto automático	Placa de piloto automático composto por 2 x IMU, 1x GPS, 1 x Barômetro
1	ADIS164890	IMU	Unidade móvel inercial de alta precisão que fornece dados de movimentação do VANT ao sistema.
1	3DR Pixhawk Airspeed	Tubo Pitot	Sensor de velocidade do vento.
1	MB2530 IRXL -MaxSonar-CS3	Sonar	Sensor sonar que fornece dados de distância.
1	UBLOX NEO-M8T	GPS	Sistema de posicionamento global que fornece dados ao sistema.
2	Flat Maxon motor brushless EC 45	Motor sem escovas	Motor usado para fornecer a capacidade VTOL da aeronave.
2	Maxon controller ESCON 36/3	Controlador de velocidade	Controladores de velocidade dos motores da capacidade VTOL da aeronave.
2	Maxon Sensor Encoder MILE, 512	Encoder	Sensor mecânico de movimento dos motores. Fornecem dados de rotação por minuto.
6	Hitec D145SW Digital HV	Servo-motor	Servo-motor usado para controle de movimentos das asas da aeronave.
2	AXI 2830/10 V2 LONG	Motor sem escovas	Motores usados na propulsão da aeronave.
2	Mezon 160 ESC	ESC	Controlador eletrônico de velocidade dos motores de propulsão.
1	Turnigy iA6C PPm/SBUS	Transceptor de rádio	Transceptor de radio usado no envio e recebimento de dados transmitidos entre aeronave e base de controle no chão.
1	Battery Management System (BMS)	Sistema de controle de energia	Sistema de controle de consumo de corrente e recarga da bateria por painéis solares acoplados na aeronave.
1	Max3232 RS232/UART	Conversor UART-RS232	Dispositivo conversor de barramento RS232 usado para compatibilidade de interface de hardware entre os componentes.
2	KNACRO RS422 to TTL UART	Conversor UART-RS422	Dispositivo conversor de barramento RS422 usado para interface de hardware entre GPS e placa de desenvolvimento.
4	LIFE 1500mah 25C 3S	Kit de baterias	Conjunto de baterias que fornece energia ao sistema e subsistemas.

O modelo OWL representa formalmente a arquitetura do sistema, descrevendo seus componentes de hardware, software, características, propriedades e relacionamentos com uso da base de conhecimentos da OAS proposto nesta tese.

A instância do sistema ProVant 4.0 é apresentada na Figura 11. As declarações da instância representam seu tipo, nesse caso classificado como sistema (system), e

um conjunto de afirmações de propriedades do objeto relacionando os componentes de hardware que compõem a arquitetura, conforme descrito na Tabela 4.

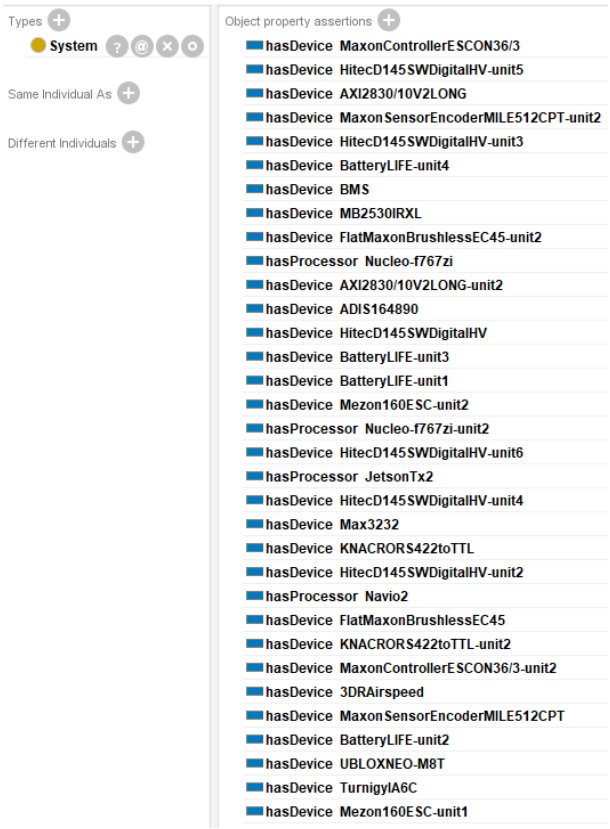


Figura 11 – Instância dos componentes de hardware em OWL.
Fonte: Figura do autor.

Considerando que a abordagem contempla a evolução da arquitetura, um conjunto de análises de resultados de desempenho da arquitetura atual encontra-se disponível e são utilizados com objetivo de identificar melhorias a serem realizadas na arquitetura, conforme descrito na **Atividade 1.4**. A equipe de engenharia aeronáutica identificou, na fase de testes e simulação, a necessidade de troca do motor de propulsão da aeronave para sanar problemas encontrados na decolagem e na inclinação dos rotores. Anteriormente, na versão 3.0, foi identificada a necessidade de trocar o sensor de unidade móvel inercial GY85. Para mais informações acesse o Github [1].

Para resolver os problemas identificados na versão 4.0, é necessário melhorar os requisitos específicos relacionados às propriedades dos motores de propulsão: aumentar a capacidade de levantamento, que atualmente é de 2.000 gramas; melhorar a eficiência dos rotores de propulsão, atualmente em 85%; a potência deve ser superior a 3800 watts. Como resultado dessa atividade, foi identificado a necessidade de troca dos motores de propulsão AXI2830/10V2LONG da arquitetura, dando início à **Etapa 2** de exploração de arquiteturas candidatas, com a definição do dispositivo de S&A selecionado para a troca, conforme descrito na **Atividade 2.1**.

[1] https://github.com/diegosales/DevCompatibility

Na sequência, com base nos requisitos que deseja-se aprimorar, são elicitados os atributos de qualidade, **Atividade 2.3**, que norteiam a seleção e a definição dos dispositivos de S&A candidatos, **Atividade 2.2**. Após uma pesquisa técnica no site de fornecedores de motores, foram identificados 5 dispositivos candidatos que cobrem os requisitos. Esses dispositivos candidatos foram modelados em web semântica usando OWL, sendo eles: AXI 5330/24 3D EXTREME KV197 V2, AXI 2835/12 GOLD LINE V2 LONG, AXI 5330/20 3D EXTREME KV235 V2, AXI 5345/16 HD 3D EXTREME V2 KV195 e AXI 5345/14 HD 3D EXTREME V2 KV225.

De posse do modelo OWL contendo os componentes de hardware da arquitetura ProVant 4.0 e conjunto de dispositivos candidatos, os projetistas criam o modelo AADL. Este novo modelo extrai da representação ontológica descrita em OWL a base de conhecimentos e dados relacionados à construção da arquitetura, diminuindo o nível de abstração e apresentando detalhes da integração dos componentes de hardware, software, sistema e implementação. Para exemplificar, o Código 4.2 e Código 4.3 apresentam os modelos AADL criados a partir da representação OWL da arquitetura ProVant 4.0. Sendo que o Código 4.2 apresenta uma parte das propriedades declaradas no modelo OWL utilizadas na modelagem dos componentes da arquitetura AADL.

```
1 property set SAO is
2 BatteryCapacity_Units: type units( Watt);
3 BatteryCapacity_Min : constant aadlinteger units SAO:BatteryCapacity_Units => 0;
4 BatteryCapacity_Max : constant aadlinteger units SAO:BatteryCapacity_Units =>
    1000;
5 BatteryCapacity_Range : type aadlinteger SAO:BatteryCapacity_Min .. SAO:
    BatteryCapacity_Max units SAO:BatteryCapacity_Units;
6 BatteryCapacity : BatteryCapacity_Range applies to (bus, device, memory,
    processor);
7 Mass_Units: type units( g);
8 Mass_Min : constant aadlreal units SAO:Mass_Units => 0;
9 Mass_Max : constant aadlreal units SAO:Mass_Units => 2000;
10 Mass_Range : type aadlreal SAO:Mass_Min .. SAO:Mass_Max units SAO:Mass_Units;
11 Mass : Mass_Range applies to (device, memory, processor, system);
...
31 Accelerometer_Accuracity_Units: type units( mg/sqrHz);
32 Accelerometer_Accuracity_Min : constant aadlinteger units SAO:
    Accelerometer_Accuracity_Units => 0;
33 Accelerometer_Accuracity_Max : constant aadlinteger units SAO:
    Accelerometer_Accuracity_Units => 100;
34 Accelerometer_Accuracity_Range : type aadlinteger SAO:
    Accelerometer_Accuracity_Min .. SAO:Accelerometer_Accuracity_Max units SAO:
    Accelerometer_Accuracity_Units;
35 Accelerometer_Accuracity : Accelerometer_Accuracity_Range applies to (device,
    system);
```

Código 4.2 – Conjunto de propriedades do modelo AADL

Na sequência, o modelo AADL representando os componentes de hardware da arquitetura é apresentado no Código 4.3. As instâncias criadas no modelo OWL foram utilizadas na geração da representação detalhada dos componentes da arquitetura, candidatos e suas propriedades.

Para prover a geração do modelo arquitetônico, esta tese propõe a Ontologia de Arquitetura de Sistemas (OAS), que fornece a base de conhecimento do domínio de AS com ênfase na aplicação aeroespacial, e um processo de transformação de modelos OWL para AADL. Adicionalmente, uma ferramenta automatizada de suporte ao processo de transformação, chamado *OWL2AADL*, foi desenvolvido no escopo desta tese. Com isso, utilizando a ferramenta e o modelo OWL de entrada, apresentado na Figura 11, geramos automaticamente os Códigos 5.3 e 4.4. Desse modo, optou-se por discorrer de maneira mais detalhada, no capítulo 5, a cerca da abordagem de desenvolvimento com o uso de ontologia e do processo de transformação dos modelos mencionados.

```
package owl2aadl
public

system implementation ProVANT_4_0
  subcomponents
    MaxonControllerESCON363 : device MaxonControllerESCON363;
    MaxonControllerESCON363-unit2 : device MaxonControllerESCON363-unit2;
    MEZON160ESC-unit2 : device MEZON160ESC-unit2;
    MEZON160ESC : device MEZON160ESC;
    Nucleo-f767zi-unit2 : processor Nucleo-f767zi-unit2;
    Nucleo-f767zi : processor Nucleo-f767zi;
    JetsonTx2 : processor JetsonTx2;
    HitecD145SWDigitalHV-unit6 : device HitecD145SWDigitalHV-unit6;
    HitecD145SWDigitalHV-unit5 : device HitecD145SWDigitalHV-unit5;
    HitecD145SWDigitalHV-unit4 : device HitecD145SWDigitalHV-unit4;
    HitecD145SWDigitalHV-unit3 : device HitecD145SWDigitalHV-unit3;
    HitecD145SWDigitalHV-unit2 : device HitecD145SWDigitalHV-unit2;
    HitecD145SWDigitalHV : device HitecD145SWDigitalHV;
    AXI2830/10V2LONG  device AXI2830/10V2LONG;
    AXI2830/10V2LONG-unit2 device AXI2830/10V2LONG-unit2;
    Max232 : device Max232;
    MB2530RXL : device MB2530RXL;
    FlatMaxonBrushlessEC45-unit2 : device FlatMaxonBrushlessEC45-unit2;
    FlatMaxonBrushlessEC45 : device FlatMaxonBrushlessEC45;
    KNACRORS422-unit2 : device KNACRORS422-unit2;
    3DRPixhawkAirSpeed : device 3DRPixhawkAirSpeed;
    UBLOXNEO-M8T : device UBLOXNEO-M8T;
  properties
    SAO:BatteryCapacity => 3000 mah;
    SAO:flightDuration => 40 minutes;
    SAO:MaxFlightRange => 20 km;
    SAO:payloadCapacity => 6 Kg;
end ProVANT_4_0;
end owl2aadl;
```

Código 4.3 – Modelo arquitetural do ProVant 4.0

A partir dos arquivos AADL gerados e do modelo de requisitos do sistema, criado em linguagem ReqSpec, o projetista dá início ao processo de seleção dos componentes que deseja efetuar a troca a partir dos atributos de qualidade, funcionalidade e interface dos componentes candidatos, conforme apresentado na **Atividade 2.3**. Na sequência são criados modelos AADL dos cenários, sendo um para cada candidato,

conforme a **Atividade 2.4** descreve. A quantidade de cenários criados depende da compatibilidade de funcionalidades e interface de hardware da arquitetura. Nesse caso, foram identificados seis motores candidatos que possuem uma única porta de entrada cada, criando assim seis cenários, sendo que um é o próprio sistema atual. Dessa forma, é possível efetuar comparações entre os cenários.

Os cenários gerados passam por um conjunto de análises definidas na **Etapa 3** em busca de identificar modificações nos componentes de software e hardware da arquitetura, **Atividade 3.1**, análise dos componentes **Atividade 3.2**, e requisitos, **Atividade 3.3**, componentes impactados. Os resultados das análises compõem os atributos de qualidade que são analisados e estruturados de acordo com as características e sub-características, descritas na **Atividade 3.4**, para apoiar a otimização da arquitetura e a seleção do cenário.

A escolha do cenário é contemplada na **Etapa 4** de otimização da arquitetura. Os atributos de qualidade e resultados quantitativos das análises dos cenários são usados na definição dos múltiplos objetivos do projeto descrito na **Atividade 4.1**, onde os projetistas definem quais atributos devem ser observados na evolução da arquitetura. Para cada atributo foi definido um peso de acordo com os objetivos do projeto, conforme definido na **Atividade 4.2**. Após as definições, é realizado o ranqueamento dos cenários, **Atividade 4.3**, e seleção do cenário, conforme descrito na **Atividade 4.4**.

Após o ranqueamento dos cenários o projetista escolhe qual cenário será implementado, gerando um modelo arquitetural AADL. O modelo arquitetural do cenário consolidado é apresentado no Código 4.4. Na sequência, o projetista efetua um conjunto de testes e simulação no modelo AADL do cenário selecionado, **Atividade 5.1**. Ao validar os resultados, o projetista efetua a atualização dos requisitos específicos **Atividade 5.2**, e PEA, **Atividade 5.3**, tornando a arquitetura candidata como a nova versão consolidada, **Atividade 5.5**.

```
package owl2aadl
public

system implementation ProVANT_4_0
  subcomponents
    MaxonControllerESCON363 : device MaxonControllerESCON363;
    MaxonControllerESCON363-unit2 : device MaxonControllerESCON363-unit2;
    MEZON160ESC-unit2 : device MEZON160ESC-unit2;
    MEZON160ESC : device MEZON160ESC;
    Nucleo-f767zi-unit2 : processor Nucleo-f767zi-unit2;
    Nucleo-f767zi : processor Nucleo-f767zi;
    JetsonTx2 : processor JetsonTx2;
    HitecD145SWDigitalHV-unit6 : device HitecD145SWDigitalHV-unit6;
    HitecD145SWDigitalHV-unit5 : device HitecD145SWDigitalHV-unit5;
    HitecD145SWDigitalHV-unit4 : device HitecD145SWDigitalHV-unit4;
    HitecD145SWDigitalHV-unit3 : device HitecD145SWDigitalHV-unit3;
    HitecD145SWDigitalHV-unit2 : device HitecD145SWDigitalHV-unit2;
    HitecD145SWDigitalHV : device HitecD145SWDigitalHV;
```

```
    AXI5330/243DV2 :  device AXI5330/243DV2;            %***** TROCA *****
    AXI5330/243DV2-unit2 : device AXI5330/243DV2-unit2; %***** TROCA *****
    Max232 : device Max232;
    MB2530RXL : device MB2530RXL;
    FlatMaxonBrushlessEC45-unit2 : device FlatMaxonBrushlessEC45-unit2;
    FlatMaxonBrushlessEC45 : device FlatMaxonBrushlessEC45;
    KNACRORS422-unit2 : device KNACRORS422-unit2;
    3DRPixhawkAirSpeed : device 3DRPixhawkAirSpeed;
    UBLOXNEO-M8T : device UBLOXNEO-M8T;
  properties
    SAO:BatteryCapacity => 3000 mah;
    SAO:flightDuration => 40 minutes;
    SAO:MaxFlightRange => 20 km;
    SAO:payloadCapacity => 6 Kg;
end ProVANT_4_0;
end owl2aadl;
```

Código 4.4 – Modelo arquitetural do ProVant 4.0

Para apoiar o desenvolvimento das atividades descritas nas etapas **2, 3** e **4** é apresentada no Capítulo 6 uma abordagem de troca de S&A baseado em modelos AADL e *ReqSpec*. Adicionalmente, uma ferramenta baseada em MDE chamada *DevCompatibility* foi desenvolvida para guiar os projetistas na realização das atividades. Dessa forma, os modelos ReqSpec e AADL apresentados nos Códigos 4.1, 5.3 e 4.3 foram usados como entrada do processo, em conjunto com as definições dos projetistas, para gerar o novo modelo arquitetural apresentado no Código 4.4 usando a abordagem CAvA. No Capítulo 6 é possível conhecer mais informações sobre o processo de exploração de arquiteturas com uso de MDE mencionado.

4.3 SUMÁRIO

Este Capítulo apresenta uma abordagem de evolução de arquiteturas aplicada a um VANT. As etapas e atividades apresentadas buscam cobrir as particularidades das modificações arquitetônicas devido à troca de dispositivos de S&A. Os projetistas, por meio da realização das atividades, podem obter um conjunto de dados que auxiliam no processo de exploração, análise e seleção de uma nova versão da arquitetura.

Apesar da abordagem apresentar um estudo de caso voltado à evolução de arquitetura de VANTs, é possível expandir para demais aplicações de CPS. Com isso, o uso de modelos em web semântica e ontologia fornecem uma base de conhecimentos que provê suporte a modelagem de diferentes domínios, o que facilita o processo de modelagem, em especial, da arquitetura do sistema.

Observando as etapas e atividades da abordagem proposta, foi identificado que as etapas de otimização e consolidação da arquitetura não foram amplamente discutidas, e algumas das atividades podem ser melhor detalhadas no futuro. Outro ponto não coberto nas atividades de consolidação foi a geração de código fonte, atividade esta que fica implícita nas etapas de projeto.

Para realizar as atividades propostas, um conjunto de ferramentas foram desenvolvidas neste trabalho. Dentre elas, a ferramenta OWL2AADL automatiza o processo de transformação de modelos OWL para modelos arquiteturais AADL. O modelo OWL fornece suporte aos projetistas na criação de modelos arquiteturais, criação de biblioteca de componentes com alto nível de abstração. Além disso, a partir do modelo OWL, a ferramenta gera automaticamente o modelo arquitetural e conjunto de propriedades em AADL, que detalham as características e propriedades da arquitetura em baixo nível de abstração. Os detalhes do processo de transformação de modelos são descritos no próximo capítulo.

Outra ferramenta proposta que provê suporte ao desenvolvimento das etapas de exploração e análise de arquiteturas candidatas e otimização da arquitetura, onde a partir dos modelos ReqSpec e AADL, o projetista seleciona os componentes de hardware da arquitetura que deseja trocar e também define um conjunto de objetivos e pesos para gerar um ranqueamento de cenários e definir qual arquitetura deseja implementar. O processo de exploração automatizada da arquitetura é descrito no capítulo 6.

5 ONTOLOGIA DE ARQUITETURA DE SISTEMAS

A OAS foi desenvolvida no âmbito desta pesquisa, tem por objetivo fornecer uma base de conhecimento contendo a representação formal dos principais conceitos, propriedades, relacionamentos e axiomas do domínio de arquitetura de sistemas.

O uso da ontologia, na fase inicial de projeto de CPS, pode auxiliar projetistas na representação formal e avaliação de inconsistências envolvendo o processo físico, rede, plataforma computacional e seus relacionamentos.

Uma ontologia dedicada ao AS pode permitir modelar conceitos relacionados à composição da arquitetura, componentes de software e hardware, características e propriedades, fornecendo detalhes sobre a interação com o processo físico. Com isso, os engenheiros podem avaliar a existência de inconsistências no projeto da arquitetura do sistema, como a compatibilidade das características e propriedades dos componentes em diferentes domínios e aplicações que integram componentes da AS.

Dentre os benefícios da ontologia proposta, buscou-se definir as terminologias e vocabulário baseado na linguagem de modelagem de arquiteturas AADL (AS5506C). Dessa forma, a OAS reutiliza a base de conhecimento sólida de uma LDA de padrão internacional, fornecendo suporte à MDE bem como estabelecendo o processo de transformação de modelos. Com isso, ontologias de outros domínios relacionados à AS tal como IOT, automóveis e rede de sensores podem reusar a OAS para representar os componentes de hardware, software e sistema da arquitetura.

A OAS foi projetada utilizando uma metodologia de desenvolvimento de ontologias proposta neste trabalho. A metodologia foi inspirada no trabalho de Noy and McGuinness (NOY; MCGUINNESS *et al.*, 2001), sendo expandidos, em mais detalhes, os passos que auxiliam na seleção e integração de reuso, e na definição de regras semânticas do domínio.

A partir da base de conhecimentos da OAS o projetista realiza a modelagem do CPS detalhando os componentes da arquitetura em linguagem OWL, nesse caso, é possível reutilizar o modelo OWL como entrada de um processo de transformação de modelos para representar o modelo arquitetural em AADL.

As seguintes contribuições podem ser aqui destacadas:

- Proposta de uma metodologia de desenvolvimento de ontologias inspirado no trabalho de Noy and McGuinness (NOY; MCGUINNESS *et al.*, 2001), incluindo os passos de seleção e integração de ontologias de reuso, e definição de regras semânticas de projeto de AS.
- Proposta de uma ontologia voltada ao domínio de AS que contemple a representação e os relacionamentos entre os componentes fundamentada na LDA AADL.
- Propor uma ferramenta de transformação de modelos OWL para AADL.

5.1 METODOLOGIA DE PROJETO DA OAS

Este trabalho foi inspirado no guia de metodologia de projeto de ontologias proposto por Noy e McGuinness (NOY; MCGUINNESS *et al.*, 2001), sendo adicionado dois novos passos em busca de detalhar o processo de seleção de reuso de ontologias, Atividade 3, e declaração de regras semânticas usadas na modelagem de aplicações no domínio de AS, Atividade 9.

Com isso, é proposta uma metodologia contendo um fluxo de nove passos que buscam cobrir diferentes aspectos da construção de uma ontologia. A Figura 12 mostra o fluxo das etapas e uma breve descrição do que foi realizado na criação da ontologia de AS.

Figura 12 – Metodologia proposta no projeto da OAS.
Fonte: Figura do autor, inspirada em (NOY; MCGUINNESS *et al.*, 2001).

Passo 1 - Concentra-se em determinar o domínio e o escopo da ontologia. Considerando a base de conhecimento fornecida na linguagem de descrição de arquitetura AADL, ocorre a declaração provisória dos principais conceitos, terminologias e vocabulário utilizados na OAS.

O padrão AS5506C (AS5506, 2004) da linguagem AADL fornece os conceitos iniciais de componentes de hardware (dispositivo, processador, memória e barramento), software (processo e tarefas), sua interação (conexões e vinculação) e sistema (co-

nexões, alocação de recursos, subcomponentes e implementações). A representação detalhada dos componentes segue o conceito de seções que descrevem as características e propriedades técnicas. Dessa forma, a AADL forneceu uma boa base de vocabulário, terminologias e compreensão de relacionamento dos componentes da AS e suas aplicações. Entretanto, as declarações da OAS são provisórias devido a capacidade de integrar o reuso de ontologias e expandir a representação e relacionamento dos componentes do sistema.

Passo 2 - Propõe aos projetistas o reuso de ontologias existentes. Existem muitas ontologias relacionadas a AS. Elas são geralmente direcionadas a aplicações específicas que usam componentes da AS. O reuso visa construir uma ontologia com base em outras previamente existentes, habilitando o projeto orientado a objetos e fornecendo flexibilidade na estruturação da ontologia desejada.

As ontologias que possuem entidades de componentes da AS podem ser encontradas em diferentes domínios na literatura, conforme apresentado no capítulo 3, bem como podem ser utilizadas para compor outras ontologias, importando módulos ou partes da ontologia (por ex. classes e propriedades). Dessa forma, o projetista evita reinventar a roda e se vale de uma vasta variedade de ontologias aceitas na comunidade cientifica e que podem ser integradas. Entretanto, essa atividade levanta uma questão fundamental observada durante o desenvolvimento dessa pesquisa.

Apesar da metodologia de Noy-McGuinness considerar a reutilização de ontologias preexistentes, existem algumas questões importantes não cobertas por essa metodologia: (1) Como ocorre a seleção das ontologias candidatas para reutilização? (2) Como são definidas as ontologias que realmente serão reutilizadas? (3) O projetista deve integrar as ontologias em seu projeto ou desenvolver uma nova ontologia, apenas aproveitando alguns conceitos utilizados nas ontologias selecionadas?

O reuso de módulos e entidades de outras ontologias podem gerar divergências, sendo necessário identificar quais ontologias podem "de fato"contribuir para o desenvolvimento de uma nova ontologia, quantificando o esforço para realizar a sua integração. Devido aos diferentes domínios que apresentam vocabulários similares, e até mesmo, idênticos, o significado pode não ser o mesmo utilizado no domínio alvo.

A avaliação de quais ontologias e módulos podem ser reutilizados para compor a ontologia de AS depende inicialmente da seleção, em alto nível de abstração, de quais componentes da arquitetura são cobertos no reuso. Com isso, foi proposta no terceiro passo, a realização da seleção de ontologias reutilizáveis.

Passo 3 - ocorre a seleção das ontologias reutilizáveis pesquisadas no passo anterior. A seleção é baseada no conjunto de critérios de classificação: (1) quantidade de classes; (2) quantidade de propriedades de objetos semelhantes; e (3) vocabulário comum. O último critério visa priorizar ontologias candidatas propostas por grupos de trabalho (como W3C), centros de pesquisa (por exemplo, SEI), ou certificados como

padrão internacional. Dessa forma, possibilita gerar uma ontologia alinhada e modular reutilizável.

Ao total, foram levantadas cinco ontologias candidatas à reutilização: SOSA, SSN, PLATont, CORA e QUDT. As ontologias SOSA e SSN (módulo SCM) detalham um conjunto de módulos e relacionamentos, aplicados ao domínio de sensores e atuadores que correspondem aos dispositivos da AS. A ontologia PLATont representa os componentes de software no domínio de plataformas computacionais. A ontologia CORA detalha a plataforma e os sensores no domínio robótico. A ontologia QUDT, desenvolvida para o projeto *NASA Exploration Initiatives Ontology Models* (NexIOM) (BAILIN; HODGSON; KELLER, s.d.), representa Quantidades, Unidades, Dimensões e Tipos de dados que podem ser aplicados às propriedades AS.

Em seguida, foram levantados o número de classes e propriedades de cada ontologia, bem como a semelhança de significado com o domínio de AS. A Tabela 5 mostra os resultados. Considerando os critérios apresentados, a ontologia SOSA apresentou maior compatibilidade (13 classes) com a proposta de AS, na sequência o módulo SCM (10 classes) da ontologia SSN. Ambos têm um vocabulário comum e cobrem parte dos componentes da AS. PLATont e CORA fornecem quatro e duas classes alinhadas com componentes da AS respectivamente. No entanto, PLATont não se baseia em um vocabulário padrão. QUDT possui 12 classes focadas em propriedades de componentes, sendo considerado um vocabulário de uso comum.

Tabela 5 – Ontologias candidatas ao reuso.

Candidatos ao reuso	**Quant. de Classes / similar**	**Quant. de propriedades / similar**	**Vocabulário comum**	**Componentes**
SOSA	16 / 13	21 / 12	padrão W3C	dispositivo, sistema, propriedades e implementação
SSN-SCM	46 / 10	44 / 16	padrão W3C	Atributos de capacidade do sistema
PLATont	84 / 4	33 / 6	Não	sistema, parte do software
IEEE 1827/2015 CORA	56 / 2	26 / 6	Sim	dispositivo, sistema e propriedades
QUDT: Quantidade Unidade, Dimensão e Tipos	366 / 12	302 / 16	Sim	propriedades da AS

Após a seleção das ontologias de reuso e sua classificação, foram definidos critérios de escolha, sendo eles: possuir maior quantidade de classes similares, possuir um padrão de vocabulário comum e cobrir a maior quantidade de componentes e classes relacionadas à AS. Com isso, as duas primeiras ontologias foram selecionadas, devido à abrangência dos componentes da arquitetura e propriedades sem sobreposição de classes. A ontologia QUDT foi selecionada devido à capacidade de representar quantidades, unidades, dimensões e dados utilizados na representação e relacionamento com as propriedades dos componentes.

É importante destacar que as entidades elicitadas devem possuir terminologias

e vocabulário semelhantes, entretanto, pode ocorrer que os significados sejam diferentes dependendo do domínio. Com isso, a atividade de mapeamento e alinhamento de terminologias de reuso, passo 4, é fundamental para evitar erros e sobreposições com o objetivo de alcançar a convergência de terminologias.

Passo 4 - É realizada a integração das ontologias de reuso por meio do mapeamento e alinhamento dos módulos, entidades, terminologias e vocabulário com o esboço da ontologia alvo. Considerando que a ontologia de AS é baseada na linguagem AADL, seus componentes compõem o esboço das classes da ontologia. A Figura 13 apresenta um exemplo de alinhamento das classes entre as ontologias SOSA/SSN e componentes AADL que guiam a descrição da OAS.

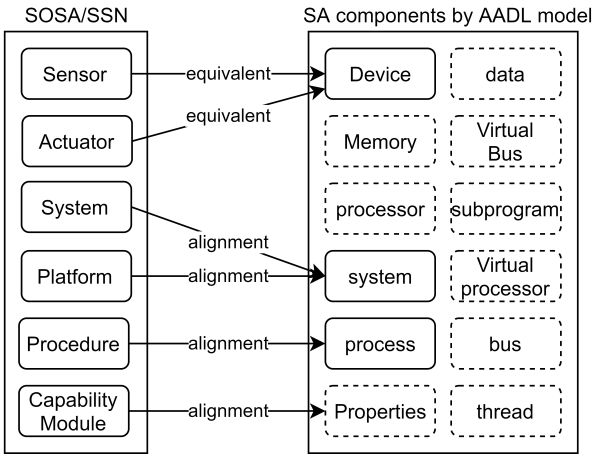


Figura 13 – Exemplo de mapeamento do reuso e componentes da OAS.
Fonte: Figura do autor.

Classes que possuem relacionamento com mais de um componente devem passar por uma análise mais detalhada, observando as propriedades do objeto e definindo regras de mapeamento para classificar classes que têm o mesmo significado. Como, por exemplo, as classes `sosa:Platform` e `sosa:System` poderem ser representadas com os componentes `sao:system` ou `sao:processor`. As classes de SOSA que não cobrem completamente a representação dos componentes são apresentadas com uma caixa rasurada, como acontece com `sosa:Capacity_Module` e `sao:Properties`. Neste caso, as classes de SOSA cobrem parcialmente a representação das propriedades descritas nos componentes da arquitetura.

No total, foram listadas treze classes reutilizadas da ontologia SOSA. Em seguida, foram mapeadas as classes SOSA relacionadas às classes dos componentes AADL, de modo a identificar semelhanças, equivalências e relacionamentos. A Tabela 6 descreve o mapeamento entre as classes da ontologia de reuso SOSA com os componentes AADL da OAS.

Algumas das classes de SOSA mostram alto grau de abstração, descrevendo que uma classe pode ter múltiplas compreensões conforme descrito na sequência.

A classe `sosa:procedures` pode representar fluxo de trabalho, protocolo, plano, algoritmo ou método computacional a ser executado por um sistema `sosa:system`, podendo ser classificada como subclasse de processo `oas:process` da OAS. As classes `sosa:Sampling`, `sosa:Observation` e `sosa:Actuation` executam ações que correspondem a classe `oas:thread` da OAS. As propriedades de observação e atuação são subclasse de `oas:properties`.

Tabela 6 – Mapeamento das classes SOSA e componentes da OAS.

Ontologia SOSA	**Mapeamento**	**componentes da OAS**
Sensor	classe equivalente	device
Actuator	classe equivalente	device
Sampler	classe equivalente	device
System	classe equivalente	system
Platform	classe equivalente	processor
Deployment	classe equivalente	system implementation
Result	classe equivalente	Result
Procedure	subclasse de	process
Sampling	subclasse de	thread
Observation	subclasse de	thread
Actuation	subclasse de	thread
ObservableProperty ActuableProperty	subclasse de	properties

Um modelo OWL da OAS e reuso (SOSA, SSN-SCM), apresentado na Figura 14, representa graficamente o relacionamento entre o componente `oas:device`, propriedade do componente, propriedades dos sensores e atuadores da ontologia SOSA que executam uma observação/atuação e fornecem um resultado. As classes do módulo `SSN-SCM` representam um conjunto de propriedades do componente importadas integralmente sem a necessidade de realizar o mapeamento e alinhamento OAS. As demais classes da OAS são definidas no próximo passo.

Outra ontologia, reusada integralmente foi a QUDT (HODGSON *et al.*, 2014), fornecendo um conjunto de unidades de medida, quantidade, dimensões e tipos de dados usados na declaração de propriedades dos componentes e não requer mapeamento e alinhamento para integração com OAS.

Passo 5 - Ocorre a definição das classes que não são abordadas na prévia identificada no passo 1 e ontologias de reuso. A partir da análise do metamodelo AADL, foi identificado que os componentes da arquitetura podem ser classificados de duas formas: do tipo de componente e da implementação do componente. Ambos diferem na estrutura e no conteúdo das declarações organizadas por meio de seções. O tipo de componente é apresentado na Tabela 7. A partir dessa classificação, foram definidas as classes relacionadas aos componentes, bem como as seções que cobrem as características e propriedades.

A Figura 15 apresenta as classes da OAS. A classe `sao:Sections` representa

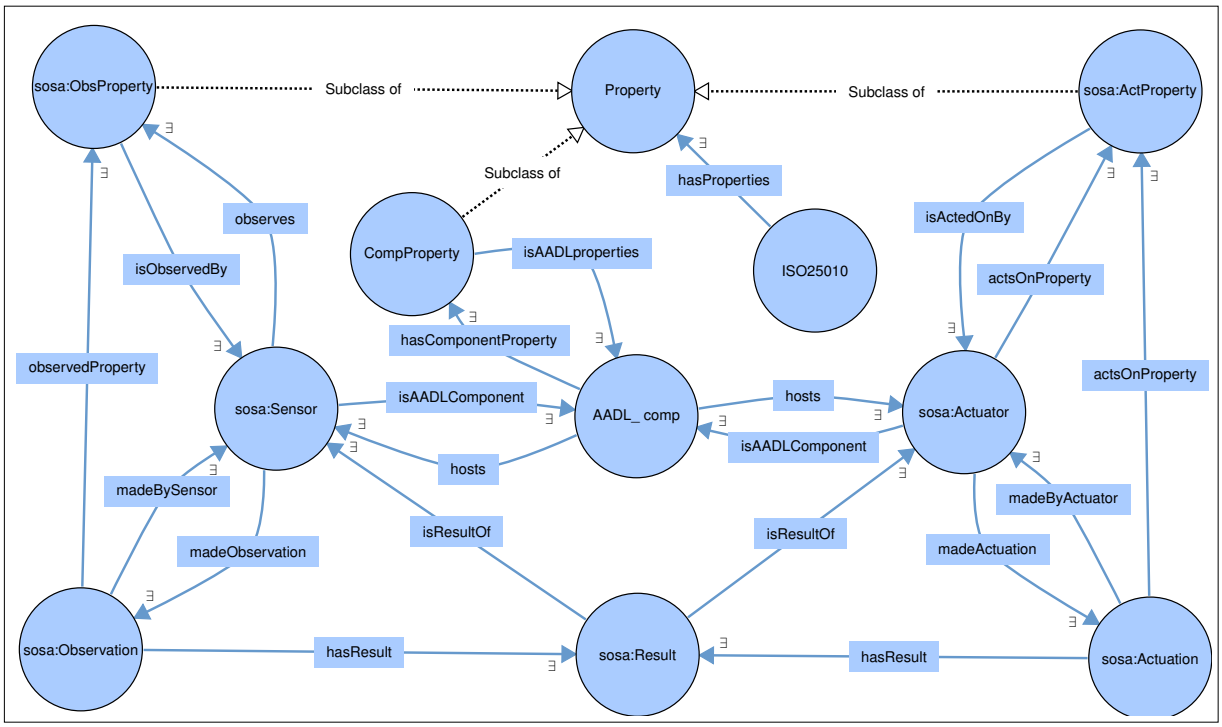


Figura 14 – Ontologia proposta integrando as classes SOSA.
Fonte: Figura do autor.

Tabela 7 – Tipos de componentes da arquitetura

(FEILER, Peter H.; LEWIS; VESTAL *et al.*, 2006).

ApplicationSoftware	Data	Dados no código-fonte e tipos de dados do aplicativo.
	Thread	Uma unidade programável de execução simultânea.
	ThreadGroup	Uma abstração para organizar logicamente threads, grupos de threads e componentes de dados dentro de um processo.
	Process	Composto por threads que são executados pelo processador.
	Subprogram	Código executável sequencialmente que pode ser chamado. Representa conceitos como métodos de retorno e chamada.
	SubprogramGroup	Uma abstração para organizar subprogramas em bibliotecas.
Plataforma de Execução	Processor	Agenda e executa threads e processadores virtuais.
	VirtualProcessor	Logical resource that is capable of scheduling and executing threads that must be bound to or be a subcomponent of one or more physical processors.
	Memory	Stores code and data.
	Bus	Interconnects processors, memory, and devices.
	VirtualBus	Represents a communication abstraction such as a virtual channel or communication protocol.
	Device	Represents sensors, actuators, or other components that interface with the external environment.
Composite	System	Integrates software, hardware, and other system components into a distinct unit within an architecture.
PropertySet	Allows you to enter additional properties and property types.	

o conteúdo dos componentes `sao:ComponentType` e `sao:ComponentImplementation`, compostos por suas características `sao:Features` e propriedades `sao:Properties`. Os recursos `sao:Features` representam a interface dos componentes como portas de conexão e barramentos de comunicação de dados. As propriedades `sao:Propriedades`

representa as características dos componentes e têm relacionamentos com a classe de reuso `ssn:SystemCapabilityModule`. As conexões dos componentes são representadas com a classe `sao:Connections`. Os anexos, modos de operação, fluxo, protótipo e extensões são classes que representam o conteúdo do componente e prover suporte para detalhar mais características voltadas a implementação do componente. A classe `sao:SOSA` representa a reutilização das classes importadas.

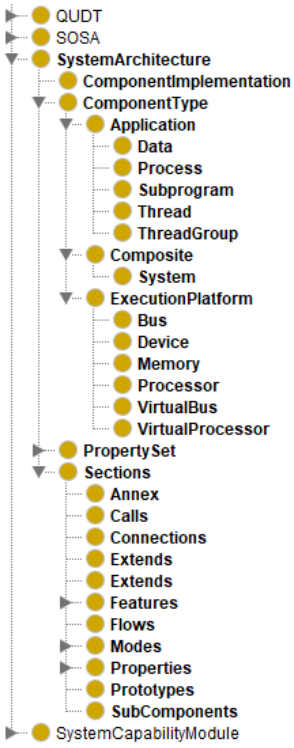


Figura 15 – Conjunto de classes da OAS com reuso e módulos integrados.
Fonte: Figura do autor.

O módulo de capacidade do sistema, importado integralmente da ontologia SSN (HALLER *et al.*, 2019), `ssn:SystemCapabilityModule`, representa um conjunto de propriedades que descrevem a operação, condições de funcionamento e capacidade do sistema que tem relação com a classe *sao:Properties*. A Figura 16 mostra classes importadas de SCM. Estendendo SCM, incluímos a subclasse *ComponentProperty* que descreve características adicionais de S&A que cobrem hardware, software e design de arquitetura.

Passo 6 - São definidas as propriedades das classes e o alinhamento com as propriedades de reuso. O conjunto de propriedades do objeto provenientes da reutilização são mapeados, sendo identificados os relacionamentos com as classes da OAS. Após integrar as propriedades de reuso, o projetista define as demais propriedades do objeto com base nas classes da ontologia.

A partir das classes e da estrutura de relacionamentos apresentadas no metamodelo AADL, são definidas as propriedades do objeto utilizadas na representação

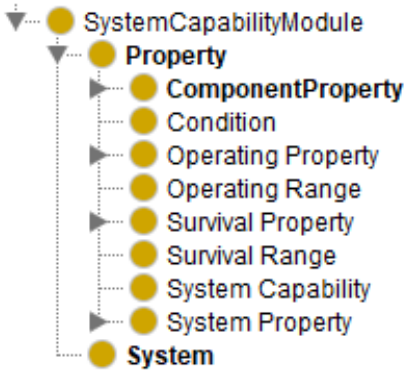


Figura 16 – Classes reutilizadas da SCM.
Fonte: Figura do autor.

dos relacionamentos entre as classes de componentes, implementação e a descrição do conteúdo, conforme mostrado na Tabela 8. Na sequência é mostrado um modelo gráfico OWL para visualização dos relacionamentos das classes `sao:ComponentType` e `sao:ComponentImplementation` com `sao:Sections`.

Tabela 8 – Exemplo de propriedades do objeto da OAS.

Propriedades do objeto		
hasDataType	hasBussAccess	hasDeviceProperties
belongsToCategory	Binding_Component	hasComponentProperty
ActualConnectionBinding	hasApplicationSwProperties	hasDeviceComponent
hasCompositeProperties	hasBuss	isDeviceComponent
hasConnection	hasData	requireBus
hasDevice	hasExecutionPlatform	isSubComponent
hasExecutionPlatformProperties	hasPort	isPartOf
hasProcess	hasProcessImplementation	isImplementedBy
hasProcessor	hasProperty	implements
hasSubComponents	hasThread	hasThreadGroup
isSectionOf	hasPrototypes	hasAnnex
hasProperties	isComponentType	hasFeatures
hasModes	isImplementationOf	hasBusAllocation
hasProcessAllocation	hasExtends	hasFlows
hasCalls	hasConnections	hasBussAccessConnection

Relacionamento dos tipos de componentes: De acordo com o modelo AADL, os componentes da arquitetura podem ser definidos em tipos e modelados em uma estrutura de seções que podem ser agregadas conforme a necessidade do projetista. Cada seção busca detalhar as características e propriedades específicas e reservadas ao componente modelado.

Os tipos de componentes representados nas classes principais `sao:Application Software`, `sao:Composite` e `sao:ExecutionPlatform` podem ser compostos por uma estrutura de sete seções declaradas por meio das propriedades do objeto. Essas seções são declaradas seguindo a representação de relacionamentos do metamodelo AADL. Na classe `Features` são declaradas as interfaces de um componente, como portas e requisição de acesso ao barramento de dados. A seção `sao:Flow` permite representar o fluxo de dados através de um componente. A seção `sao:Modes` permite definir modos de operação do componente. A seção `sao:Prototypes` permite que seus

principais parâmetros sejam definidos como um padrão. A classe `sao:Extends` refina o modelo, `sao:Properties` é onde os valores são declarados para propriedades associadas a um componente. Na Figura 17, é possível verificar a representação do tipo de componente e composição das seções de conteúdo.

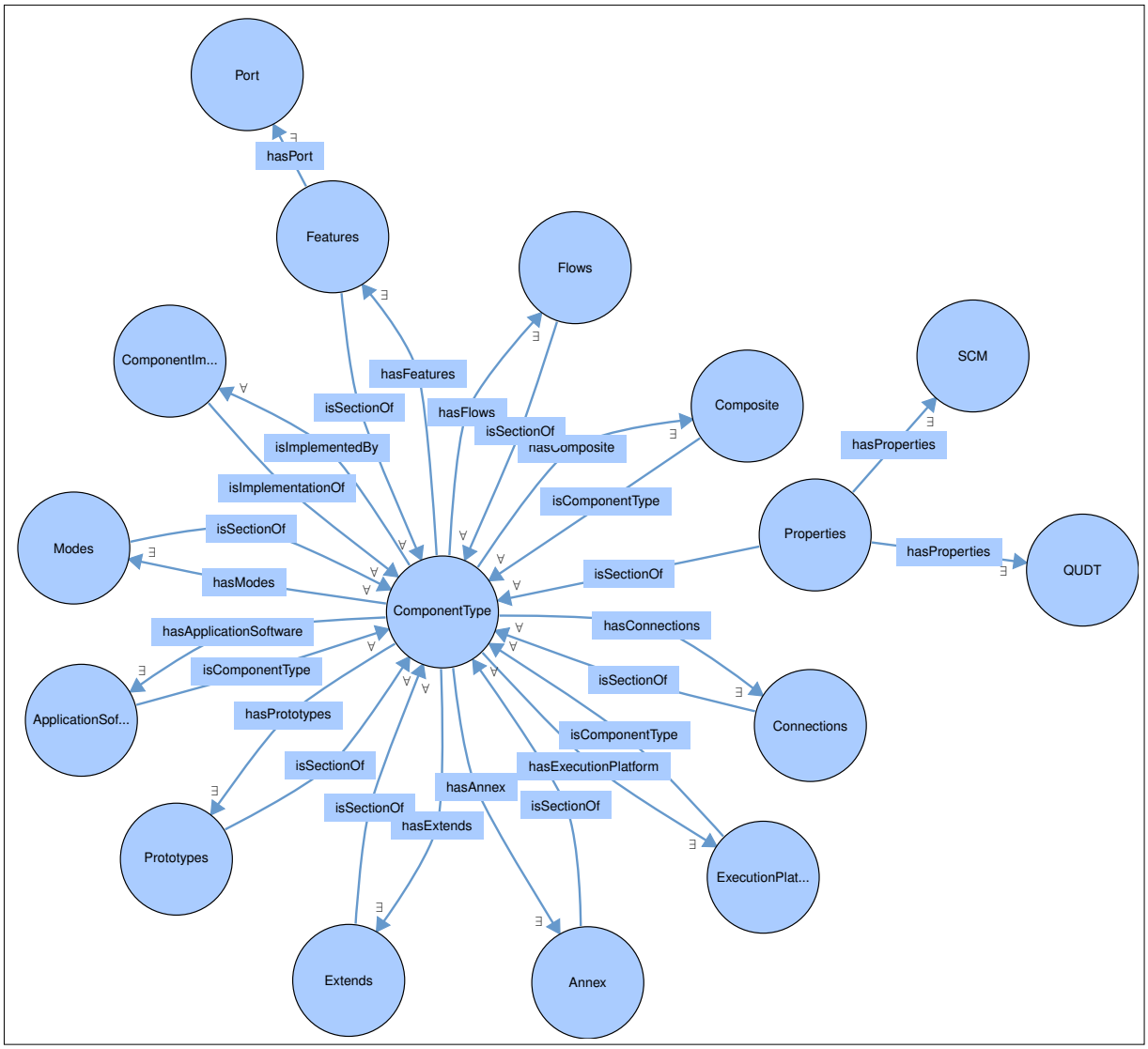


Figura 17 – Relacionamento das seções e tipos de componentes.
Fonte: Figura do autor.

Relacionamento dos componentes implementados: A classe de componentes implementados pode ser representada em até nove relacionamentos com as subclasses de seções, sendo elas: `sao:Extends`, `sao:Prototypes`, `sao:Properties` e `sao:Annex` utilizadas nos relacionamentos com as categorias de componentes `sao:ComponentType` Na seção de subcomponentes `sao:Subcomponents` são definidos os indivíduos que compõem a implementação do componente. A subclasse `sao:Connection` representa como os indivíduos interagem entre si. A subclasse `sao:Flows` representa o fluxo de dados através dos subcomponentes. A subclasse `sao:Calls` representa as sequências

de chamadas dos subprogramas definidos nas implementações de tarefas e processos do sistema. Na Figura 18 são apresentados os relacionamentos entre as classes e propriedades de objeto.

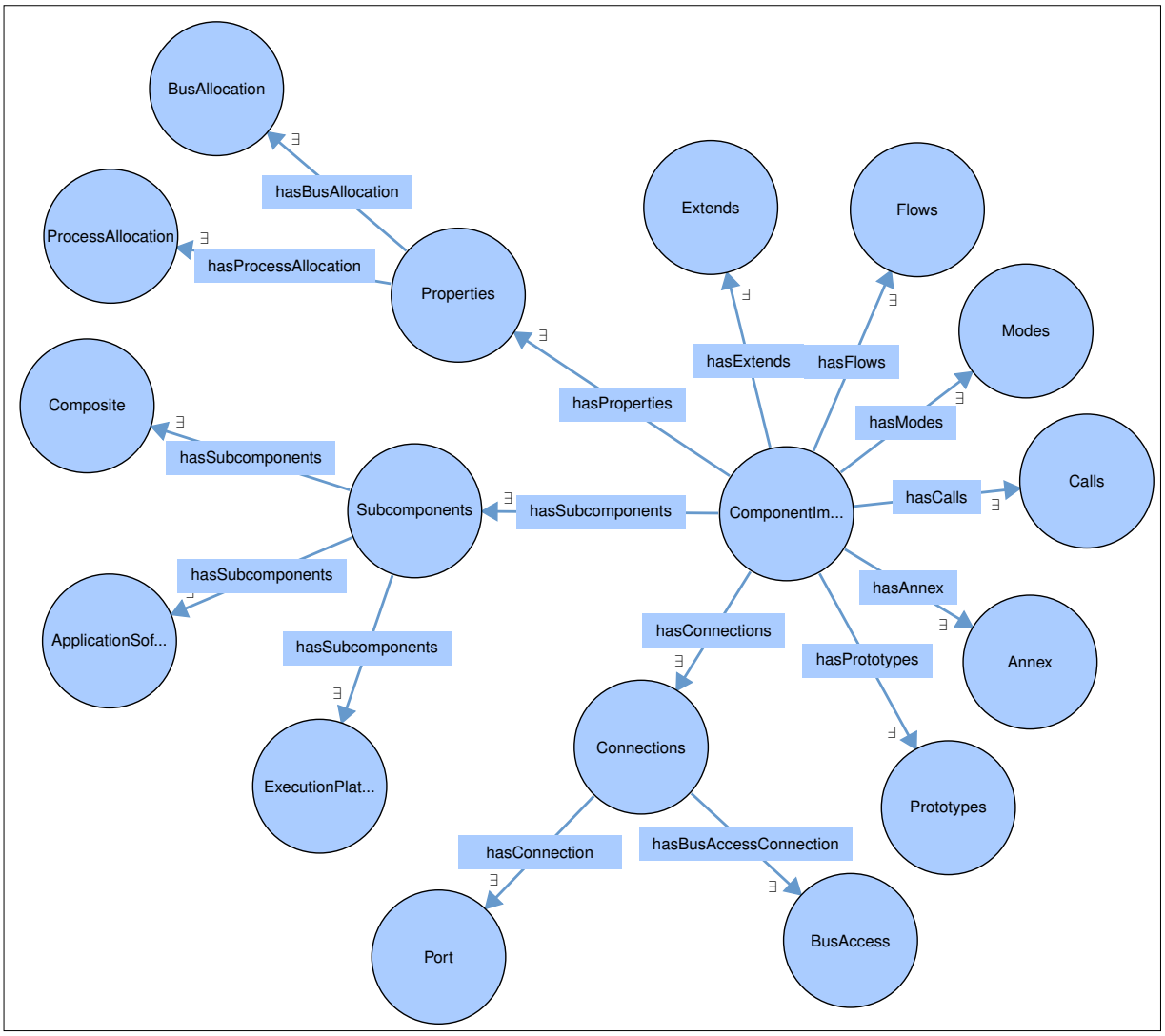


Figura 18 – Relacionamentos dos componentes da implementados.
Fonte: Figura do autor.

As seções dos componentes implementados são destinadas a representar a estrutura, interação e interconexão do componente, descrevendo detalhes em nível de implementação e instância dos subcomponentes que o compõem.

Passo 7 - São definidos os dados e tipos de dados relacionados às classes da ontologia. Nesse caso foi reusado a ontologia "QUDT"que fornece um conjunto de classes de medidas físicas. Na Figura 19 é mostrado um conjunto de dados e suas propriedades referentes à unidade de medição de peso, utilizado nas definições de propriedades dos componentes da arquitetura.

A declaração dos dados e tipos de dados segue a seguinte estrutura: Declaração do nome da propriedade de dado, nesse caso a unidade de medição de massa

`qudt:Mass`; Declaração do tipo de classe ao qual pertence, massa é sublcasse da `qudt:Weight`; Declarar a unidade de medição, reutilizando a ontologia QUDT que fornece a unidade que vamos utilizar nesta medição representada com a terminologia (g); Declaração de valores máximos (ValueMax) e mínimos (ValueMin) permitidos e tipo de dado, considerando a aplicação, os componentes devem possuir massa inferior a 2000 gramas e são representados em dados inteiros `type:xsd:int`.

Os dados podem ser quantitativos, como por exemplo a massa, bem como qualitativos que auxiliam na descrição de funcionalidade e tipo de componente da arquitetura. Por exemplo, a declaração da propriedade de dado `Device_Fucionality` e `Device_Type` é do tipo qualitativo e possui como dado a definição de uma palavra do tipo `type:xsd:string` que indique o tipo de funcionalidade do componente.

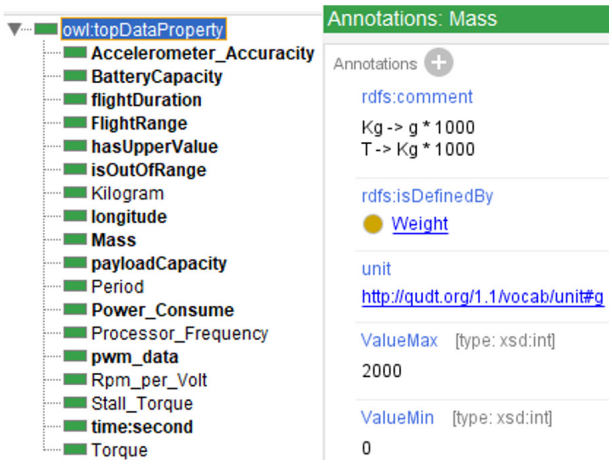


Figura 19 – Declaração de propriedades e tipo de dados usados na SA.
Fonte: Figura do autor.

Os dados declarados devem possuir relação com as capacidades do sistema de uma determinada aplicação e as propriedades dos componentes que deseja-se modelar. Conforme o tipo de aplicação, o conjunto de dados declarados provê suporte a representação dos atributos de qualidade do sistema e seus componentes. Com isso, nesse passo, é importante que o projetista identifique quais os atributos de qualidade deseja modelar na representação das propriedades de dados dos indivíduos (instâncias) que serão definidas.

Ao total, OAS possui vinte e três propriedades de dados declarados para representação da arquitetura de um sistema no domínio aeroespacial conforme apresentado na Tabela 9. Estas propriedades de dados foram declaradas com base nas características técnicas fornecidas por fabricantes de componentes arquiteturais obtidos em manuais técnicos e datasheet.

É importante destacar que até este passo, as entidades da OAS estão consolidadas e podem ser utilizadas na criação dos indivíduos e criação da base de conhecimentos do domínio. A seguir será apresentado o passo que detalha a criação

Tabela 9 – Propriedades de dados do domínio aeroespacial.

Propriedade	**Descrição**
Acurácia do Acelerômetro	Medição da acurácia do sensor de aceleração linear.
Capacidade de Levantamento 3D	Peso máximo que pode ser levantado com segurança ou carregado por um motor.
RPM/V	Rotação por minuto e por tensão.
Largura de banda máxima	Máxima capacidade de transmissão de dados do barramento.
Preço	O preço de compra no mercado.
Resolução	O menor valor de mudança de sinal que o dispositivo pode observar.
Ressonância	A frequência na qual o pico de magnitude de saída ocorre.
Nível de proteção	Classificação do nível de proteção física do dispositivo.
Capacidade da bateria	Medição de carga energética armazenada na bateria, tipicamente em ampere por hora (unidade de medição).
Sensibilidade	Razão da diferença entre unidade de medida de entrada e saída.
Eficiência	Para motores, a eficiência é a relação entre a potência mecânica fornecida pelo motor (saída) e a energia elétrica fornecida ao motor (entrada).
Consumo de potência	É a quantidade de energia de entrada (medida em watts) necessária para que um componente funcione.
Voltagem de operação	Nível de tensão do componente no qual suas características operacionais funcionam.
Máxima potência consumida	A energia elétrica máxima por unidade de tempo, fornecida para operar um componente do sistema.
Velocidade do processador	É o número de ciclos por segundo em que o processador opera e é capaz de processar.
Massa	Unidade de medida de peso.
Período	É o número de ocorrências de um evento repetido por unidade de tempo.
Funcionalidade	O conjunto de recursos associados a um componente de execução da plataforma.
Ruído do sinal	Modificações que um sinal pode sofrer durante a captura, armazenamento, transmissão, processamento ou conversão.
Torque de saída	Unidade de medição para motores lineares e rotacionais.
Faixa de operação	Faixa ou escala de medição do fenômeno.
Sem corrente de carga	É a corrente que está fluindo quando a carga em um motor é desconectada ou não está presente em um momento.

dos indivíduos.

Passo 8 - São criados os indivíduos que representam a materialização das classes. Indivíduos podem representar uma descrição genérica ou precisa das classes. Indivíduos genéricos representam alguma descrição não relacionada aos componentes, mas relacionada à capacidade, comentário ou características do sistema, (por exemplo, cores, unidade de medida e palavras). A descrição precisa representar um tipo de componente e implementação do componente.

A modelagem do indivíduo é composta por três declarações de asserções: tipo de classe (*Types*), asserções de propriedade do objeto (*Object property assertions*) e asserções das propriedades dos dados (*Data property assertions*). No tipo de classe são declaradas as classes a que o indivíduo pertence ou está relacionado. As propriedades do objeto representam as características e composição de um indivíduo. As propriedades dos dados representam as propriedades e dados específicos do indivíduo.

Passo 9 - As regras semânticas permitem expressar um conjunto de axiomas a fim de apoiar a avaliação formal, classificação e transferência de características dos indivíduos modelados. As regras são representadas na forma de uma implicação entre um antecedente, contendo as condições a serem testadas, e as condições do consequente (inferências) aplicadas aos indivíduos avaliados.

Os axiomas são construídos a partir dos tipos de classes, propriedades de obje-

tos e dados da SAO. Cada axioma pode avaliar alguma funcionalidade, característica, classificação (mapeamento) e cumprimento dos requisitos dos componentes da arquitetura modelados nos indivíduos. Dessa forma, a construção das regras depende do entendimento das entidades da ontologia, da forma que os dados foram estruturados, conforme apresentado no passo 7, e dos objetivos desejados na avaliação. A definição das regras é feita através de uma linguagem de semântica da web, neste caso foi usado a *semantic web rule language* (SWRL) que fornece um conjunto de comandos de suporte a avaliação dos indivíduos.

É importante destacar que o projetista pode definir as regras com base nos parâmetros de capacidade e funcionalidade do sistema, avaliando o cumprimento de requisitos do projeto que estejam relacionados aos indivíduos. Com isso, é possível avaliar quais arquiteturas e componentes atendem às condições criadas, facilitando a seleção e filtragem prévia dos indivíduos incompatíveis.

Por exemplo, uma regra semântica usando SWRL foi criada com intuito de avaliar quais indivíduos modelados atendem a um conjunto de requisitos de projeto. A regra proposta avalia quais indivíduos do tipo dispositivo (*Device(?)*) têm uma capacidade de peso inferior a cinco quilogramas (*swrlb:lessThan(?y,5)*) e podem ser compatíveis com o sistema (*(ComponentImplementation(?x)*). Os indivíduos candidatos modelados são avaliados e, caso atendam a estes argumentos, recebe uma inferência da classe do tipo *ComponentImplementation* que indica que o componente é candidato à implementação. A equação (3) mostra um exemplo de regra SWRL com um argumento de axioma de peso usando um comando SWRL de comparação (menor que).

$$\underbrace{Device(?x)}_{\text{class argument}} \wedge \underbrace{hasWeight(?x,?y) \wedge swrlb:lessThan(?y,5)}_{\text{data property argument}} \Rightarrow \underbrace{ComponentImplementation(?x)}_{\text{inference result}} \tag{3}$$

O uso de regras semânticas fornece suporte ao mapeamento e ao alinhamento entre ontologias (como por exemplo, SOSA para SAO) e transferência de propriedades do objeto, auxiliando na transformação de modelos de diferentes domínios que desejase alinhar com a OAS. Por exemplo, o uso de indivíduos da aplicação IoT fornece um conjunto de sensores que podem ser usados na base de conhecimento da OAS. Domínios relacionados à OAS, como no caso a SOSA, podem declarar os dispositivos de sensoriamento da classe *sosa:Sensor* que podem ser relacionados à classe *oas:Device* na OAS através da declaração de regras semânticas. A equação (4) mostra um exemplo de regra semântica com objetivo de alinhamento de indivíduos entre o

tipo de classe SOSA e SAO.

$$\underbrace{Sensor(?x)}_{\text{individuals with SOSA class}} \quad \Rightarrow \quad \underbrace{Device(?x)}_{\text{SAO class inference result}} \tag{4}$$

Ao completar as etapas, a ontologia se consolida e fornece uma base de conhecimento de representação formal da AS. É importante destacar que a OAS pode ser reutilizada em outros domínios para apoiar a representação da arquitetura. Por exemplo, o domínio aeroespacial que possui um vocabulário específico de VANTs e reusar a OAS para representar AS. O modelo OWL, que reusa a OAS, pode ser utilizado como entrada no processo de transformação de modelos e assim gerar um conjunto de arquivos AADL de suporte à análise e avaliação das capacidades do sistema.

Na próxima seção será apresentado o processo de transformação de modelos OWL para AADL com o objetivo de extrair os dados da base de conhecimento da OAS e representar um modelo arquitetônico derivado das definições ontológicas. Dentre os benefícios da transformação de modelos OWL para AADL temos a criação de uma vasta biblioteca de componentes, conjunto de propriedades e dados modelados que podem ser utilizados na criação de um modelo arquitetural menos abstrata, a utilização da base de conhecimento na criação de plug-ins adicionais e bibliotecas de análise das características e propriedades da arquitetura no ambiente OSATE2.

5.2 TRANSFORMAÇÃO DE MODELOS OWL PARA AADL

Baseado na OAS, o projetista desenvolve um modelo OWL contendo um conjunto de indivíduos que representam os componentes da AS e biblioteca de componentes candidatos de acordo com o domínio e aplicação desejada. O modelo OWL fornece os dados de entrada do motor de transformação que gera como resultado um conjunto de arquivos (pacote) de extensão *.aadl*. Os arquivos criados pelo mecanismo de transformação são: um arquivo contendo o conjunto de propriedades utilizados na arquitetura modelada; um arquivo contendo o modelo arquitetural e seus componentes; uma pasta contendo um conjunto de arquivos *.aadl* referente a cada indivíduo relacionado aos componentes da arquitetura (biblioteca de componentes). Essa estrutura de arquivos segue as boas práticas de projeto de arquiteturas apresentado por Peter Feiler (2019).

A Figura 20 apresenta o fluxo da transformação de modelos proposta. Foi desenvolvido no âmbito desta tese uma ferramenta de suporte à transformação de modelos OWL para AADL, denominada OWL2AADL. A ferramenta foi desenvolvida como plug-in do ambiente OSATE2 e habilita aos desenvolvedores de arquitetura a importação e transformação automatizada de modelos. O plug-in está disponível no GitHub [1].

[1] https://github.com/diegosales/OWL2AADL

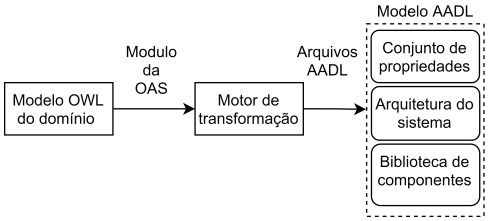


Figura 20 – Visão geral da ferramenta OWL para AADL.
Fonte: Figura do autor.

5.2.1 Motor de transformação de modelos OWL para AADL

O processo de transformação de modelos OWL para AADL proposto segue os conceitos de M2M apresentados no Capítulo 2, onde a partir dos meta-modelos, meta-meta-modelos, conjuntos de regras e especificações declaradas é definida a estrutura do motor de transformação em blocos, conforme apresentado na Figura 21. Inicialmente, o processo requer a entrada de um *Modelo de Domínio OWL* em conformidade com a *estrutura OAS*, seguindo a sintaxe RDF/XML definida pelo W3C.

O bloco de *Especificação de Transformação* contém o mapa de relacionamentos entre a *estrutura OAS* e a *estrutura AADL* usado na criação das *Regras de transformação*. No bloco *Especificação de transformação*, as entidades OWL são definidas com base no metamodelo OWL2, classes de mapeamento, propriedades do objeto e dados relacionados à estrutura do modelo AADL e seus elementos. O bloco de *Regras de transformação* usa os dados do bloco de especificações de mapeamento para criar um conjunto de regras de transformação entre o modelo OWL de entrada e o modelo AADL de saída.

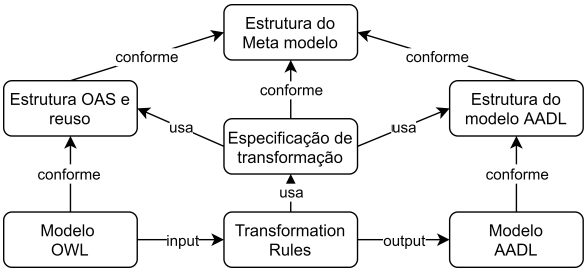


Figura 21 – Processo do motor de transformação de modelos.
Fonte: Figura do autor.

Os metamodelos dos modelos AADL e OWL fornecem a estrutura das linguagens utilizadas no processo de transformação de modelos. A linguagem OWL, por ser capaz de representar uma grande abrangência de domínios, segue as entidades declaradas na OAS e são usadas no processo de especificação da transformação para o meta modelo alvo (AADL). A Figura 22 apresenta um fragmento do meta modelo RDF/OWL de entrada utilizado no motor de transformação.

A Figura 23 apresenta o meta modelo AADL, baseado no padrão AS5506, dos

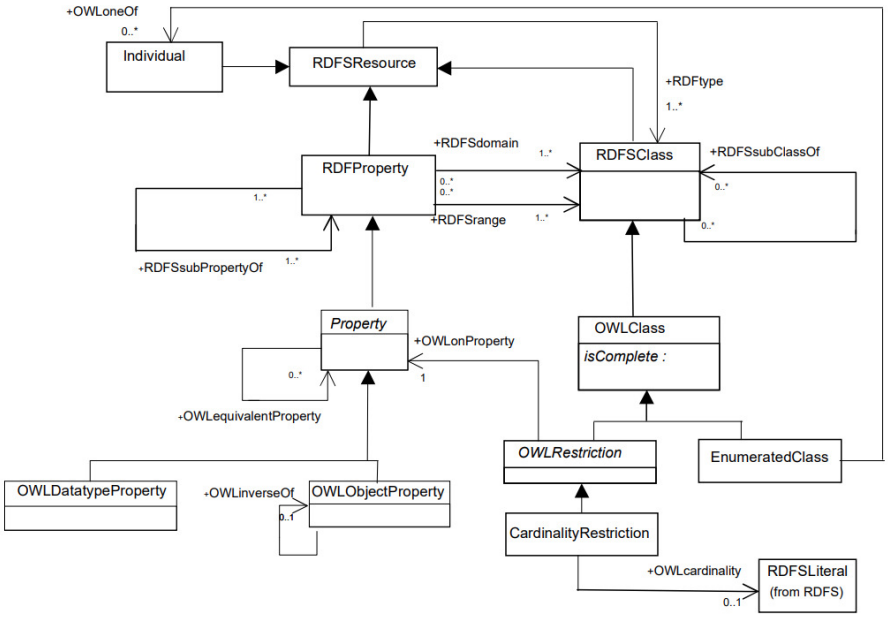


Figura 22 – Fragmento do metamodelo RDFS e OWL.
Fonte: (COLOMB *et al.*, 2006).

principais componentes da arquitetura, secções e conjunto de propriedades que compõem a estrutura do modelo alvo AADL.

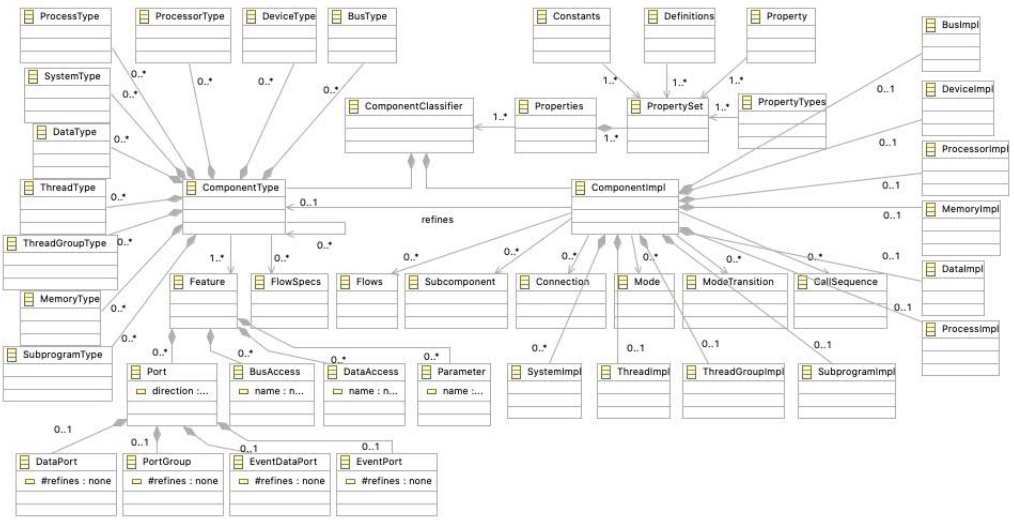


Figura 23 – Metamodelo AADL
Fonte: Figura do autor, baseado no padrão AS5506 AS (AS5506, 2004).

A Tabela 10 apresenta o mapa de relacionamentos entre as entidades OWL e componentes da estrutura AADL utilizados na criação dos blocos de especificação da transformação. A *classe* OWL pode corresponder a um dos elementos AADL, tal como componentes, seção ou capacidade do sistema. Os *indivíduos* OWL correspondem à instância das classes, características e propriedades dos componentes. As *propriedades do objeto* descrevem a composição dos componentes AADL, recursos (porta, barramento, dispositivo) e propriedades (massa, período, torque).

Tabela 10 – Entidades OWL e relacionamentos de componentes AADL.

Entidades OWL	Elementos AADL
Classe (class)	Componentes, seções e capacidades da AS.
Indivíduos (individuals)	Instância dos componentes e características
Propriedades dos objetos (objectproperty)	Composição dos componentes. (características e propriedades)
Propriedades dos dados (dataproperties)	Conjunto de propriedades dos componentes da AS.
Faixa de dados (DataRange)	
Anotações de propriedades (AnnotationProperty)	

As entidades OWL *Propriedades dos dados, Faixa de dados, Anotações de propriedades* compõem a criação do conjunto de propriedades (*property set*) AADL. As *propriedade dos dados* são utilizadas na representação das unidades de medição e propriedades utilizadas na capacidade do sistema e seus componentes. A *faixa de dados* fornece a lista de classes que utilizam o conjunto de propriedades AADL. Por fim, as *anotações de propriedades* fornecem informações adicionais ao conjunto de propriedades AADL.

O modelo de declaração do *conjunto de propriedades* ADDL segue uma estrutura composta por: declaração do tipo da propriedade, declaração da propriedade e declaração de constantes da propriedade, conforme mostrado no Código 5.1. Nas linhas 1 e 2 são declaradas o nome do conjunto de propriedades e pacote ao qual pertence. Na linha 3, são declarados o tipo da propriedade, faixa de valores numéricos aceitáveis para uma propriedade. Na linha 4, ocorre a declaração do nome da propriedade, identificando quais componentes AADL fazem uso da propriedade. Na linha 5, são declaradas as constantes de propriedades tais como unidades de medição.

```
1 property set <property set name> is
2 [with <property set or package name(s)>]
3 <property type declaration>
4 <property declaration>
5 <property constant declaration>
6 end <property set name>;
```

Código 5.1 – Modelo de declaração do conjunto de propriedades AADL

Com objetivo de fornecer suporte ao alinhamento das entidades OWL e o conjunto de propriedades AADL, foi criado um padrão de declarações das propriedades de dados OWL contendo sete propriedades, conforme mostrado na Tabela 11. Estas propriedades fornecem a estrutura básica de dados utilizados na especificação de transformação entre as propriedades de dados OWL e o conjunto de propriedades AADL.

A propriedade `type` indica o nome do conjunto de propriedades, para suportar a criação de diferentes arquivos de conjuntos de propriedades. A propriedade `constant` suporta o valor numérico constante de uma unidade de medida. A propriedade `unit` define as unidades de medida de um dado, podendo ser declarado com uso de ontologias de reuso, tal como a ontologia QUDT. As propriedades `Valuemin` e `Valuemax` são usadas para definir o intervalo de valores numéricos da propriedade. E `Range` define

quais componentes da arquitetura podem usar a propriedade criada.

Tabela 11 – Conjunto de propriedades de dados de suporte ao alinhamento.

Propriedade de dados OWL	**AADL property set**
<owl:DataProperties rdf:about="http...#type"/>	Declaração do tipo de propriedade
<owl:DataProperties rdf:about="http...#constant"/>	Declaração de propriedades constantes
<owl:DataProperties rdf:about="http...#unit"/> <owl:DataProperties rdf:about="http...#Valuemin"/> <owl:DataProperties rdf:about="http...#Valuemax"/> <rdfs:domain rdf:resource="http...#AADLCompoent"/> <rdfs:isDefinedBy rdf:resource="http...#Measurement"/> <owl:DataProperties rdf:about="http...#Range"/>	Declaração de propriedades

A Tabela 12 resume o alinhamento entre os tipos de dados OWL e AADL. Nesse caso, as declarações de tipo de dados OWL são relacionadas à construção do conjunto de propriedades da linguagem AADL.

Tabela 12 – Alinhamento entre os dados OWL e AADL.

Tipo de dados OWL	**Tipo de dados AADL**
xsd:boolean	aadlboolean
xsd:string	aadlstring
sao:enumeration	enumeration
sao:units	units
owl:real	aadlreal
xsd:int	aadlinteger
sao:range	range of
sao:classifier	classifier
sao:comment	reference
sao:record	record

O bloco de *transformação de regras* é composto por duas atividades sequenciais. Na primeira é executada a transformação do arquivo OWL em objetos da linguagem Java seguindo a estrutura do metamodelo OWL. O processo de transformação de arquivos OWL foi desenvolvido utilizando uma biblioteca padrão java (disponível no site org.w3c.dom) por ser baseada em XML. Essa biblioteca retorna um objeto Java contendo todo o conteúdo do arquivo.

O motor de transformação faz uma busca nos nós das classes, propriedades dos dados e objetos, e finalmente os indivíduos. É importante seguir essa ordem por causa das referências definidas pelo modelo de estrutura OWL2. Encontrando um nó definido, o motor inicia o tratamento individualizado: verifica os sub-nós e seus atributos. Essa varredura procura componentes AADL conforme definido na base de conhecimento da OAS. A partir dessa varredura, os dados são separados para serem usados na conversão. Entidades que não seguem as regras definidas são ignoradas e podem ser incluídas no futuro para representar outros domínios e anexos de vocabulário.

Na segunda atividade, os objetos Java são relacionados com a estrutura do metamodelo AADL por meio do bloco de *especificação de transformação*, onde os objetos relacionados com os elementos AADL são usados como base para geração dos arquivos AADL, ocorrendo a conversão de objetos relacionados a OWL em objetos relacionados a AADL. O motor de transformação gera um conjunto de arquivos na

saída do processo, sendo eles: biblioteca de componentes contendo um conjunto de arquivos AADL referente a cada indivíduo relacionado a componentes AADL; um arquivo AADL contendo o conjunto de propriedades AADL e o modelo arquitetural do sistema.

5.3 FERRAMENTA DE TRANSFORMAÇÃO DE MODELOS OWL2AADL

Esta seção é organizada em três partes. A primeira está relacionada com a aplicação da OAS e criação de um modelo OWL da aeronave ProVant 4.0 utilizando a ferramenta Protegé. Na sequência é demonstrado um exemplo de regras semânticas para avaliação de requisitos de projeto da arquitetura. E finalmente, a transformação do modelo OWL para AADL por meio da ferramenta OWL2AADL desenvolvida.

A modelagem em OWL inicia com o levantamento da lista de componentes de hardware da aeronave ProVant 4.0, fornecido nos documentos da equipe de engenharia e apresentados previamente no capítulo anterior, Tabela 4. Cada componente tem um conjunto de propriedades específicas que devem ser identificadas com intuito de avaliar a cobertura e inclusão de propriedades de dados da OAS, seguindo o padrão apresentado na declaração de dados abordado no passo 6 da metodologia de desenvolvimento de ontologias.

A definição das propriedades dos dados da ontologia são preenchidos com os valores numéricos obtidos dos documentos técnicos e datasheet fornecidos por fabricantes. Cada propriedade que se deseja modelar pode ser adicionada seguindo o **Passo 7** da metodologia. No total, foram definidas 20 propriedades de dados para representar a aeronave ProVant 4.0.

Um exemplo de propriedade de dados referente à propriedade de massa é mostrado na Figura 24. Essa propriedade é do domínio dos componentes `sao:dispositivo, sistema, processador, memória`, sendo definidos pela medição do peso `ssn:Weight` com uma unidade de medida `qudt: g` (gramas) reusada da ontologia QUDT. O Range de valores numéricos indica o valor mínimo *ValueMin* 0 e valor máximo *ValueMax* de 2000 expressos com o tipo de dado inteiro. No campo de anotações são definidos comentários para representar a escala de multiplicação das unidades de medição.

Após declarar as propriedades dos dados dos componentes, dá-se início à modelagem individual dos componentes listados, seguindo o **Passo 8** da abordagem. Cada indivíduo contém três campos de definições modelados com base nas propriedades de objetos e dados. Um exemplo de modelagem do componente AXI2830 com base na OAS é mostrado na Figura 25. Este indivíduo possui a funcionalidade "motor brushless", sendo representado pelo tipo de componente `sao:device`. A interface de hardware do componente é definida nas propriedades do objeto. Nesse caso, foi definida através da propriedade do objeto objeto representam as características do indivíduos (portas e barramento de comunicação) como *pwm_bus*, para ajudar na

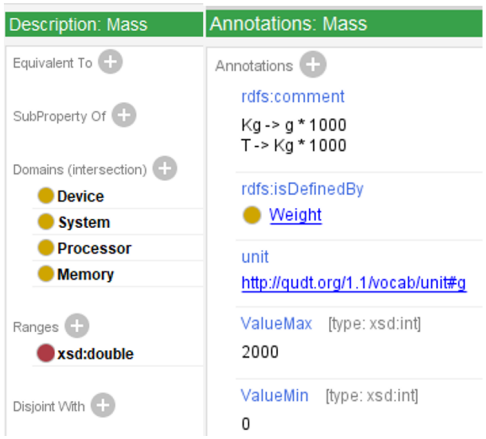


Figura 24 – Modelagem em OWL da propriedade do dado Mass.
Fonte: Figura do autor.

composição da interface. As asserções de um conjunto de propriedade de dados foram definidos para suporte à descrição técnica do componente.

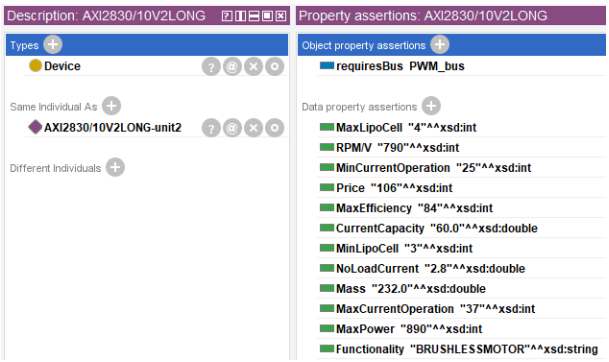


Figura 25 – Definições do indivíduo AXI2830.
Fonte: Figura do autor.

Após modelar em OWL, as propriedades dos dados e indivíduos que representam os sistemas ProVant e seus componentes, o projetista utiliza a ferramenta de transformação OWL2AADL desenvolvida nesta pesquisa. É importante destacar que dos trinta e três indivíduos definidos para representar os componentes do sistema ProVant 4.0, vinte e cinco deles são dispositivos relacionados aos sensores e atuadores da arquitetura, demonstrando o uso intensivo desses componentes no desenvolvimento de um sistema CPS.

A ferramenta OWL2AADL foi desenvolvida como um plugin do ambiente OSATE2 e realiza o processo de transformação de modelos OWL em AADL. O processo é baseado em conceitos de MDE, onde a partir dos metamodelos de entrada (OWL) e saída (AADL), um conjunto de regras de transformação propostas no motor de transformação são executadas. A ferramenta lê o arquivo OWL de entrada e processa o relacionamento entre as entidades OWL e elementos AADL, gerando arquivos do modelo AADL contendo o conjunto de propriedades, modelo arquitetônico e pacote de componentes

(biblioteca). Esses arquivos gerados contêm a estrutura AADL mínima necessária para modelar o sistema e realizar avaliações e análises em OSATE2.

O conjunto de propriedades AADL gerado é baseado nas propriedades de dados OWL declarados nas definições apresentadas na Tabela 9. O arquivo do conjunto de propriedades gerado segue a estrutura de declaração AADL, onde são definidos o tipo de dado, unidade de medição, valores máximos, mínimos e nome da capacidade. Uma parte do arquivo gerado é apresentado no Código 5.2. Ao total, foram gerados automaticamente 42 linhas de código que habilitam a modelagem em AADL das propriedades dos componentes da arquitetura ProVant 4.0.

```
1 property set SAO is
2 BatteryCapacity_Units: type units( Watt);
3 BatteryCapacity_Min : constant aadlinteger units SAO:BatteryCapacity_Units => 0;
4 BatteryCapacity_Max : constant aadlinteger units SAO:BatteryCapacity_Units =>
    1000;
5 BatteryCapacity_Range : type aadlinteger SAO:BatteryCapacity_Min .. SAO:
    BatteryCapacity_Max units SAO:BatteryCapacity_Units;
6 BatteryCapacity : BatteryCapacity_Range applies to (bus, device, memory,
    processor);
7 Mass_Units: type units( g);
8 Mass_Min : constant aadlreal units SAO:Mass_Units => 0;
9 Mass_Max : constant aadlreal units SAO:Mass_Units => 2000;
10 Mass_Range : type aadlreal SAO:Mass_Min .. SAO:Mass_Max units SAO:Mass_Units;
11 Mass : Mass_Range applies to (device, memory, processor, system);
12 Power_Consume_Units: type units( Watt);
13 Power_Consume_Min : constant aadlinteger units SAO:Power_Consume_Units => 0;
14 Power_Consume_Max : constant aadlinteger units SAO:Power_Consume_Units =>
    10000;
15 Power_Consume_Range : type aadlinteger SAO:Power_Consume_Min .. SAO:
    Power_Consume_Max units SAO:Power_Consume_Units;
16 Power_Consume : Power_Consume_Range applies to (device, memory, processor,
    system);
...
31 Accelerometer_Accuracity_Units: type units( mg/sqrHz);
32 Accelerometer_Accuracity_Min : constant aadlinteger units SAO:
    Accelerometer_Accuracity_Units => 0;
33 Accelerometer_Accuracity_Max : constant aadlinteger units SAO:
    Accelerometer_Accuracity_Units => 100;
34 Accelerometer_Accuracity_Range : type aadlinteger SAO:
    Accelerometer_Accuracity_Min .. SAO:Accelerometer_Accuracity_Max units SAO:
    Accelerometer_Accuracity_Units;
35 Accelerometer_Accuracity : Accelerometer_Accuracity_Range applies to (device,
    system);
```

Código 5.2 – Modelo AADL gerado com a ferramenta OWL2AADL

O segundo arquivo AADL gerado pela ferramenta é o pacote OWL2AADL. Este contém os componentes AADL da arquitetura, resultantes do processo de transformação dos indivíduos OWL. A entrada do modelo OWL ProVant 4.0 criado fornece oitenta e três indivíduos, entre eles a arquitetura ProVANT, recursos de componentes e biblioteca de componentes candidatos à troca. O arquivo AADL gerou, através do processo de transformação de modelos proposto, setenta e quatro componentes AADL, representados em quatrocentos e nove linhas de comando geradas automaticamente.

A quantidade de componentes AADL gerados na transformação foi menor do que a quantidade de indivíduos OWL, pois existem indivíduos que não possuem declarados o tipo de classe relacionado aos componentes AADL. Sendo que, dos oitenta e três indivíduos, nove são declarados como tipo *secção* e são usados na representação da interface de hardware dos componentes. Por brevidade, é apresentada no Código 5.3 uma parte dos componentes AADL da arquitetura do sistema ProVant 4.0 que foram gerados usando a ferramenta OWL2AADL.

```
package owl2aadl
public
device ADIS164890
  features
    i2c_bus: requires bus access i2c_bus_400kbps;
  properties
    SAO:Functionality => IMU;
    SAO:Mass => 100.0g;
    SAO:Power_Consume => 0.02W;
end ADIS164890;
...
system implementation ProVANT_4_0
  subcomponents
    MaxonControllerESCON363 : device MaxonControllerESCON363;
    MaxonControllerESCON363-unit2 : device MaxonControllerESCON363-unit2;
    MEZON160ESC-unit2 : device MEZON160ESC-unit2;
    MEZON160ESC : device MEZON160ESC;
    Nucleo-f767zi-unit2 : processor Nucleo-f767zi-unit2;
    Nucleo-f767zi : processor Nucleo-f767zi;
    JetsonTx2 : processor JetsonTx2;
    HitecD145SWDigitalHV-unit6 : device HitecD145SWDigitalHV-unit6;
    HitecD145SWDigitalHV-unit5 : device HitecD145SWDigitalHV-unit5;
    HitecD145SWDigitalHV-unit4 : device HitecD145SWDigitalHV-unit4;
    HitecD145SWDigitalHV-unit3 : device HitecD145SWDigitalHV-unit3;
    HitecD145SWDigitalHV-unit2 : device HitecD145SWDigitalHV-unit2;
    HitecD145SWDigitalHV : device HitecD145SWDigitalHV;
    AXI2830/10V2LONG  device AXI2830/10V2LONG;
    AXI2830/10V2LONG-unit2 device AXI2830/10V2LONG-unit2;
    Max232 : device Max232;
    MB2530RXL : device MB2530RXL;
    FlatMaxonBrushlessEC45-unit2 : device FlatMaxonBrushlessEC45-unit2;
    FlatMaxonBrushlessEC45 : device FlatMaxonBrushlessEC45;
    KNACRORS422-unit2 : device KNACRORS422-unit2;
    3DRPixhawkAirSpeed : device 3DRPixhawkAirSpeed;
    UBLOXNEO-M8T : device UBLOXNEO-M8T;
  properties
    SAO:BatteryCapacity => 3000 mah;
    SAO:flightDuration => 40 minutes;
    SAO:MaxFlightRange => 20 km;
    SAO:payloadCapacity => 6 Kg;
end ProVANT_4_0;
end owl2aadl;
```

Código 5.3 – Parte do código gerado dos componentes AADL.

A ferramenta OWL2AADL gera o conjunto de arquivos AADL que podem ser abertos no ambiente OSATE2 para dar continuidade no projeto da arquitetura, efetuando a instância dos componentes e sistema que deseja-se implementar e realizar

várias análises relacionadas aos atributos, comportamento e características de qualidade do sistema a partir dos demais plug-ins nativos.

5.4 SUMÁRIO

Neste capítulo foi apresentada uma OAS com o objetivo de representar semanticamente arquiteturas de CPS que utilizam intensivamente sensores e atuadores, fornecendo terminologias, vocabulário e conceitos, identificando as relações dos componentes com base na linguagem de descrição de arquiteturas AADL. A proposta busca solucionar problemas identificados nos trabalhos relacionados quanto à representação semântica de arquitetura de sistemas, com a intenção de usar os benefícios da modelagem de alta abstração, por meio de ontologias, no desenvolvimento de CPS.

Foram apresentados detalhes quanto à metodologia de desenvolvimento utilizada na criação da OAS foram apresentados com intuito de mostrar aos leitores os conceitos, entidades, ontologias reutilizadas e regras semânticas utilizadas no suporte à avaliação de requisitos de projeto de AS. Durante o desenvolvimento da OAS foi realizada a modelagem das entidades e relacionamentos utilizando a linguagem OWL e o ambiente de desenvolvimento Protegé. Dessa forma o leitor pode visualizar detalhes quanto à estrutura da ontologia proposta.

Outra contribuição apresentada foi o processo de transformação de modelos OWL para AADL. Uma ferramenta de transformação de modelos OWL para AADL foi desenvolvida a fim de usar os benefícios da representação semântica com ontologias na geração de um modelo de arquitetura do sistema. Além disso, foi apresentado um exemplo de aplicação no domínio aeroespacial com modelagem OWL para demonstrar detalhes quanto à descrição, objetos e propriedades de dados que suportam o processo de transformação para o modelo AADL.

6 EXPLORAÇÃO, ANÁLISE E OTIMIZAÇÃO DA ARQUITETURA

As etapas 2 a 4 da abordagem CAvA apresentam um conjunto de atividades voltadas à exploração e análise de cenários de arquiteturas a partir da seleção dos dispositivos de S&A e candidatos a troca. Entretanto, realizar estas atividades manualmente não é trivial e estão sujeitas a erros devido a grande quantidade de cenários e implementações factíveis. Neste capítulo será apresentada, em detalhes, a aplicação destas etapas com uso de uma abordagem de troca S&A, em modelos AADL, visando guiar e automatizar o desenvolvimento das etapas mencionadas.

Com o intuito de guiar os projetistas no desenvolvimento das atividades da abordagem proposta, foi desenvolvido um plug-in do ambiente OSATE2, denominado *DevCompatibility*. A ferramenta usa como entrada os modelos AADL e biblioteca de componentes, gerados no processo de transformação de modelos apresentado no capítulo 5, e os requisitos da arquitetura descritos no modelo ReqSpec. O código fonte para integrar o plug-in ao OSATE2, manual passo a passo e o executável Java compilado encontra-se disponível no Github [1].

6.1 CONSIDERAÇÕES INICIAIS

As considerações tem por objetivo evitar problemas quanto à aplicação da abordagem e uso da ferramenta *DevCompatibility*, fornecendo o conjunto de modelos AADL compatíveis e que segue a estrutura de definições propostas no capítulo 4, sendo elas:

Modelagem da Ontologia: Caso o projetista deseje efetuar a modelagem ontológica do sistema por meio da OAS, é necessário que as propriedades dos dados devem seguir o padrão de declaração de variáveis apresentado no Capítulo 5 com intuito de habilitar a criação do conjunto de propriedades utilizadas no modelo AADL da arquitetura gerada.

Requisitos e critérios modelados: Admite-se que os projetistas de software, em conjunto com as partes interessadas, já tenham identificado os critérios qualitativos e quantitativos do projeto através de um modelo que descreva os requisitos funcionais e não funcionais. Se um novo grau de liberdade, novo parâmetro, for incluído nos requisitos, o projetista deve incluir uma nova análise de critério em busca de avaliar o cumprimento dos requisitos modelados.

Modelo arquitetural disponível: Considera-se que o projetista de software tenha desenvolvido previamente um modelo arquitetural que deseja avaliar o espaço de exploração para evolução da arquitetura, podendo utilizar os modelos AADL gerados automaticamente na ferramenta *OWL2AADL*, conforme apresentado no capítulo 5. O modelo arquitetural em questão deve estar instanciado e com as características, propriedades e alocações previamente definidas. A declaração de duas propriedades

[1] https://github.com/diegosales/DevCompatibility

qualitativas devem ser realizadas para análise das funcionalidades do sistema, sendo elas *Device_Functionality* e *Device_Type* para declaração de funcionalidade e tipo do dispositivo.

Biblioteca de dispositivos modelados: Considera-se que o projetista tenha realizado previamente a modelagem dos dispositivos de S&A candidatos que deseja avaliar, definindo suas características e propriedades de acordo com os atributos de qualidade utilizados como critério de avaliação.

6.2 ABORDAGEM DE TROCA S&A EM MODELOS AADL

Com intuito de aplicar as etapas 2 a 4 da abordagem CAvA, foi desenvolvida uma abordagem baseada em MDE para auxiliar no processo de troca dos dispositivos de S&A em modelos arquiteturais descritos em AADL e de requisitos com uso da linguagem ReqSpec. A abordagem proposta é composta de três etapas: (1) requisitos do sistema, (2) exploração da arquitetura e (3) otimização da arquitetura apresentadas na Figura 26. A primeira etapa cobre o levantamento e análise de requisitos do sistema, sendo composta de duas atividades: Extração de requisitos do modelo ReqSpec e análise de requisitos. Como resultado, são gerados um conjunto de parâmetros relacionados aos requisitos dos dispositivos de S&A do modelo arquitetural. Estes parâmetros fornecem informações quanto a restrições e limitações de projeto da arquitetura para a exploração de arquiteturas.

A segunda etapa tem por objetivo realizar a exploração da arquitetura, sendo composta de três atividades: seleção interativa, analisador de modelos AADL e exploração de arquitetura. Como resultado, são gerados automaticamente um conjunto de cenários em AADL a partir das seleções e definições do projetista, sendo enviados para a terceira etapa.

Na terceira etapa é realizada a otimização da arquitetura, sendo composta de três atividades: seleção interativa de objetivos, programação de meta ponderada e gerador de modelos AADL. Tendo como entrada os cenários gerados, o projetista define um conjunto de objetivos da nova arquitetura através da definição de pesos que são utilizados no ranqueamento dos cenários. Como resultado, são gerados relatórios contendo o ranqueamento dos cenários e o arquivo AADL do cenário que deseja implementar (arquitetura evoluída).

A ferramenta *DevCompatibility* apresenta uma interface gráfica contendo um fluxo de telas que auxiliam os projetistas na realização das três etapas da abordagem de forma semi-automatizada, habilitando a seleção e definição de um conjunto de parâmetros que norteiam a execução automatizada das atividades relacionadas às análises, exploração de arquiteturas, otimização com a geração do ranqueamento de cenários e geração do modelo AADL. A seguir, serão apresentadas em mais detalhes as atividades das etapas com uso da ferramenta com o estudo de caso do ProVant

Figura 26 – Estrutura da abordagem de troca de S&A.
Fonte: Figura do autor.

4.0.

Seleção dos modelos AADL e ReqSpec

Nessa atividade, o projetista seleciona a pasta do projeto que contém o Arquivo AADL da AS, biblioteca de componentes candidatos e modelo ReqSpec que serão utilizados no processo de troca de dispositivos de S&A. Dessa forma, os parâmetros iniciais de exploração da arquitetura são definidos de acordo com os objetivos do projeto. Para isso, na ferramenta *DevCompatibility*, foi desenvolvida uma interface gráfica de acesso aos arquivos e seleção do sistema e dispositivos de S&A, no ambiente OSATE2. A Figura 34 apresenta a interface gráfica inicial de seleção da pasta de projeto `src` que contém quatro pastas: Library, Provant_3_0 e ProVant_4_0. Sendo a pasta *Library* contendo arquivos AADL dos componentes e Provant_3_0 e ProVant_4_0 os arquivos AADL da AS.

Ao clicar no botão "Next", a ferramenta apresenta ao projetista os sistemas disponíveis para realização da troca dos S&A. Nesse caso, foi selecionado o sistema ProVant_4_0 apresentado no estudo de caso desta tese. É possível obter mais detalhes quanto aos subcomponentes e conexões do sistema selecionado com intuito de apresentar previamente ao projetista os componentes da arquitetura. A Figura 28 apresenta a interface de seleção de um sistema do projeto.

A ferramenta *DevCompatibility* é organizada em duas áreas de visualização. Na área da esquerda, são apresentados os cenários de acordo com as definições

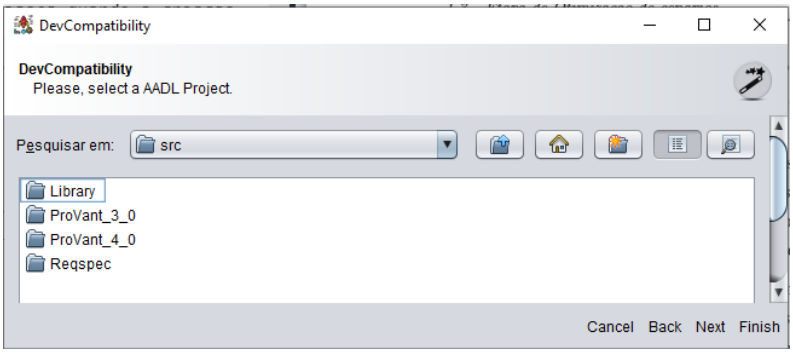


Figura 27 – Interface gráfica de seleção dos modelos.
Fonte: Figura do autor.

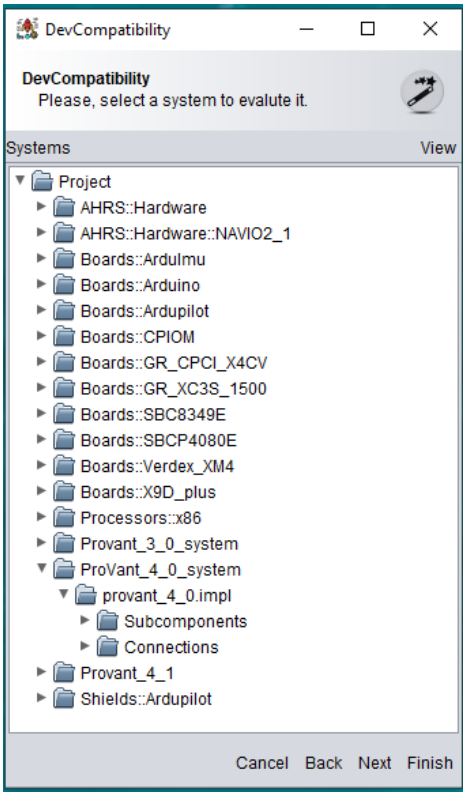


Figura 28 – Seleção do sistema ProVant 4.0 para troca de S&A.
Fonte: Figura do autor.

escolhidas. Na área da direita, são apresentadas as informações quanto às mudanças realizadas em cada cenário analisado, A Figura 29 apresenta as áreas de visualização da ferramenta.

Ao clicar no botão **“Add”** são apresentados os três modos de operação disponíveis na ferramenta: 1) Modificações manuais no sistema (modo manual); 2) Troca de um componente por outro (modo semi-automático); 3) Geração automática de cenários (modo automático). A Figura 30 apresenta a tela com os modos de operação. Cada modo tem por função prover suporte a diferentes formas de ajuste e definição de parâmetros de exploração de cenários, possibilitando que o usuário tome decisões quando a criação dos cenários e modificações na arquitetura selecionada.

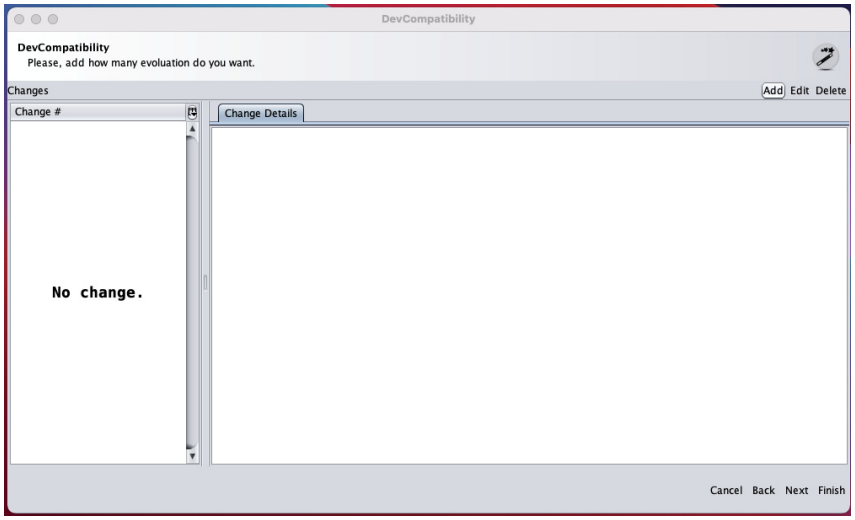


Figura 29 – Tela de apresentação dos cenários.
Fonte: Figura do autor.

O modo de modificações manuais possibilita realizar modificações nos componentes e parâmetros da arquitetura. Esse modo deixa livre a modificação da arquitetura e possui o mínimo de suporte, guiado pela abordagem de troca de S&A, para que o projetista efetue ajustes e refinamento do sistema que deseja avaliar. O modo semi-automático habilita o usuário a selecionar os dispositivos da arquitetura e candidatos que deseja efetuar a troca. Dessa forma, a criação do cenário é direcionada a seleção dos dispositivos candidatos definidos pelo projetista. No modo automático, o usuário efetua a seleção do dispositivo que deseja realizar a troca e a ferramenta realiza um conjunto de análises automatizadas com objetivo de avaliar a compatibilidade dos dispositivos disponíveis na biblioteca.

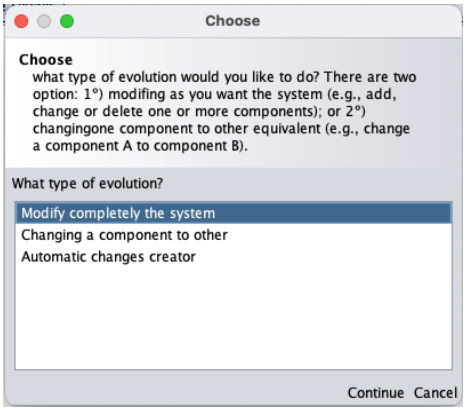


Figura 30 – Seleção do tipo de exploração de projeto de arquitetura.
Fonte: Figura do autor.

A atividade de seleção interativa possui um papel importante na abordagem de exploração automatizada de cenários ao propor que o usuário realize a definição dos parâmetros iniciais de seleção dos modelos, sistema implementado, dispositivos que devem ser trocados e biblioteca de dispositivos candidatos, restringindo assim o espaço

de exploração de cenários arquiteturais. As definições realizadas pelo projetista são armazenadas na memória da ferramenta e registradas no relatório final das análises da exploração de cenários. Assim, o projetista tem um histórico de todos os dispositivos que foram selecionados para a troca na arquitetura e pode avaliar quais dos cenários atende melhor os objetivos do projeto.

6.2.1 Etapa de requisitos do Sistema

Os requisitos funcionais e não funcionais descrevem as funcionalidades, comportamento, especificações e restrições definidas no âmbito do projeto e são utilizados para direcionar os projetistas na seleção e definição dos dispositivos de S&A candidatos. A linguagem de especificação de requisitos textuais, *ReqSpec*, permite que os usuários realizem a modelagem de metas, requisitos, requisitos das partes interessadas e requisitos do sistema de modelos AADL com uso de plug-ins que possuem suporte à ferramenta OSATE2.

A troca dos S&A da arquitetura impacta diretamente no cumprimento do conjunto de requisitos do sistema modelado. Com isso, identificar quais requisitos e especificações possuem ligação com os componentes e parâmetros do modelo AADL relacionado é fundamental para avaliar a exploração de cenários e identificar quais cenários observados atendem ao cumprimento dos requisitos do sistema, assim como o impacto causado na troca do componente. Para isso, são propostas a extração dos requisitos funcionais e não funcionais do modelo *ReqSpec* relacionado ao modelo AADL selecionado para realizar a exploração de cenários. Os requisitos e parâmetros de projeto extraídos são analisados para identificar a relação com os dispositivos de S&A e suas especificações que serão armazenados em memória da ferramenta para posterior avaliação de cumprimento dos requisitos realizados na etapa de análise da arquitetura.

Analisador de modelo de requisito (Parser)

Inicialmente, o modelo *ReqSpec* é utilizado na entrada dessa atividade para extração dos requisitos e parâmetros de projeto da arquitetura com objetivo de converter os requisitos do modelo *ReqSpec* em objetos da linguagem Java (*ReqSpecObject*).

A estrutura do analisador é mostrada na Figura 31, tendo como entrada o modelo ReqSpec da arquitetura selecionada. O analisador de modelo *ReqSpec* efetua a leitura do modelo ReqSpec e o lexer efetua a conversão em fluxo de tokens, enquanto o analisador (*Parser*) processa esse fluxo de token e cria objetos (*ReqSpecObject*) com base no metamodelo do subconjunto do ReqSpec, intitulado *ReqSpecModel*.

Os objetos e atributos de *ReqSpecObject* são armazenados na memória interna e fornecem os atributos dos requisitos funcionais e não funcionais extraídos. Com base nos objetos e atributos extraídos do modelo ReqSpec, a ferramenta estrutura um

Figura 31 – Estrutura do analisador de modelo ReqSpec.
Fonte: Figura do autor.

conjunto de dados que são utilizados na avaliação de cumprimento dos requisitos de projeto e análise dos cenários.

Análise de requisitos

A análise de requisitos tem por objetivo realizar o levantamento dos objetos, atributos e parâmetros de *ReqSpecObject* que possuem referência aos sensores e atuadores da arquitetura do sistema, suas especificações técnicas e requisitos não funcionais. Os sensores e atuadores declarados no modelo ReqSpec e relacionados com os dispositivos do modelo AADL são identificados e filtrados para serem extraídos os requisitos e especificações armazenados nos objetos e atributos.

6.2.2 Etapa de Exploração de Arquiteturas

A exploração de arquiteturas é a parte central da abordagem de troca de S&A da arquitetura, sendo composta por três atividades: (1) Seleção interativa, (2) Análise do modelo AADL e (3) Exploração de arquitetura. A atividade de seleção interativa define quais modelos AADL devem ser utilizados no processo de exploração de cenários, sendo realizada a seleção e definição da arquitetura do sistema implementado e a escolha dos dispositivos de S&A da arquitetura que devem ser trocados. A atividade de análise do modelo AADL tem por objetivo efetuar a extração dos dados da arquitetura selecionada e da biblioteca de componentes para obter as características e propriedades que serão utilizadas na exploração. A atividade de exploração de arquitetura realiza a avaliação das características e propriedades extraídas na atividade anterior, identificando a compatibilidade de interface entre o sistema selecionado e os dispositivos candidatos disponíveis na biblioteca.

Ao final da realização da etapa de exploração de arquiteturas, a ferramenta *DevCompatilibity* armazena os registros das modificações realizadas, cenários criados, assim como os parâmetros de projeto definidos durante o desenvolvimento das atividades.

Analisador de modelos AADL

A linguagem AADL é baseada na linguagem de programação orientada a objetos Java. Com isso, a leitura de modelos AADL segue a estrutura de objetos e atributos. Cada objeto armazena os atributos contendo as propriedades dos componentes declarados no modelo. A principal função do analisador de modelo AADL é converter o modelo de arquitetura AADL textual e não estruturado em objeto de modelo de arquitetura AADL para extrair os dados do modelo AADL que serão utilizados na análise e exploração de cenários. A estrutura do analisador do modelo AADL é mostrada na Figura 32, sendo a entrada desta o modelo arquitetural do sistema implementado e a biblioteca de dispositivos previamente selecionados no módulo anterior.

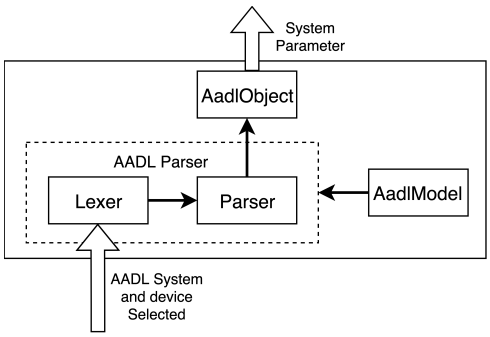


Figura 32 – Estrutura do analisador de modelo AADL.
Fonte: Figura do autor.

O analisador de modelos AADL recebe os modelos AADL que são convertidos seguindo o fluxo de tokens definidos pelo *lexer*. Enquanto o analisador (*Parser*) processa esse fluxo de tokens e cria objetos (*AadlObject*) com base na estrutura do modelo AADL *AadlModel*. O *AadlModel* é baseado no metamodelo AADL apresentado no capítulo anterior. Os objetos e atributos dos componentes correspondentes às classes do *AadlModel* são armazenados em *AadlObject*. Os objetos *AadlObject* são armazenados na memória interna da ferramenta e fornecem atributos do sistema instanciado, dispositivos selecionados e candidatos. Com base nos atributos extraídos dos modelos, a ferramenta estrutura um conjunto de dados que são utilizados na avaliação e análise dos dispositivos e arquitetura selecionados.

Exploração de arquiteturas

A partir do objeto *AADLObject* e parâmetros definidos na seleção interativa, a ferramenta realiza a exploração de cenários em busca de identificar os dispositivos e arquiteturas compatíveis. A exploração de cenários é composta de duas etapas. A primeira etapa busca realizar a análise de compatibilidade dos dispositivos de S&A selecionados e candidatos.

Nessa etapa, ocorre o levantamento dos atributos dos dispositivos selecionados e candidatos com objetivo de identificar similaridades, evitando que a troca cause riscos quanto ao cumprimento dos requisitos funcionais e não funcionais. Inicialmente, são identificadas as funcionalidades, hardware (portas e barramentos), software(processos e tarefas) e propriedades específicas dos dispositivos selecionados e candidatos, avaliando e identificando similaridades e diferenças. Os dispositivos candidatos e combinações que cobrem as funcionalidades dos dispositivos selecionados são listados para dar início à geração dos cenários de arquiteturas.

A segunda etapa tem por objetivo analisar as modificações a serem realizadas na arquitetura para integrar os componentes candidatos e gerar os cenários. As modificações identificadas são armazenadas em um relatório técnico que descreve os tipos de alterações que precisam ser realizadas para implementar cada cenário. Nessa etapa são avaliados os cenários gerados na etapa anterior, identificando o impacto da troca dos dispositivos selecionados. Sendo levantado os componentes de hardware e software da arquitetura que tiveram modificações, realocação dos recursos e parâmetros de configuração. A Figura 33 apresenta o fluxo de atividades que contemplam as duas etapas.

Figura 33 – Fluxo de atividades da exploração de arquiteturas.
Fonte: Figura do autor.

***Atividade* 1**- *Extração de atributos* - Os dispositivos e sistemas selecionados armazenados no objeto *AADLObject* são levantados para extração dos atributos que serão utilizados para avaliação de troca e análise da exploração de cenários. Os atributos fornecem as características e propriedades dos componentes e sistemas implementados para avaliação da troca e levantamento de modificações na arquitetura do sistema. O primeiro atributo avaliado é a funcionalidade, devido a importância de cumprimento dos requisitos funcionais do sistema ao realizar modificações e troca de dispositivos.

***Atividade* 2**- *Análise de compatibilidade funcional* - Os dispositivos AADL possuem declarados no escopo do modelo uma propriedade de funcionalidade, conforme apresentado no Capítulo 5. As funcionalidades dos dispositivos selecionados são levantadas e comparadas com as funcionalidades dos dispositivos candidatos disponíveis

na biblioteca. A análise dessa propriedade possibilita identificar quais dispositivos candidatos possuem compatibilidade de funcionalidade e estão aptos a serem utilizados na troca.

Após a análise das funcionalidades a ferramenta *DevCompatibility* apresenta o quantitativo de candidatos que possuem compatibilidade de funcionalidade e permite ao projetista selecionar qual componente deseja efetuar a troca. Dando continuidade ao estudo de caso da arquitetura ProVant 4.0, foi selecionada a troca do dispositivo AXI2830V2LONG. A ferramenta apresentou ao total 5 candidatos que serão analisados quanto ao cumprimento de funcionalidades.

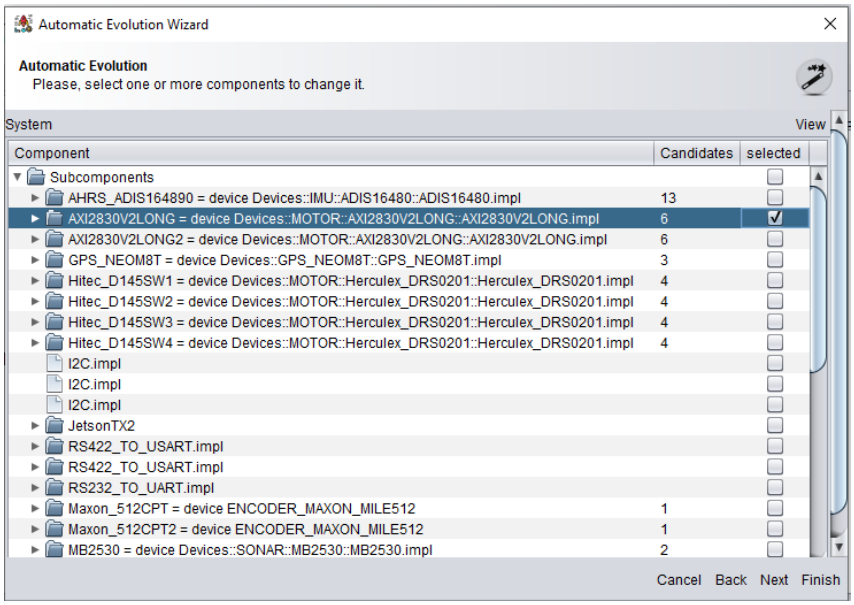


Figura 34 – Seleção dos dispositivos a serem trocados.
Fonte: Figura do autor.

***Atividade* 3**- *Geração de combinações funcionais* - As funcionalidades dos dispositivos levantados são utilizadas como referência para a criação dos cenários. As funcionalidades dos dispositivos candidatos e selecionados são comparadas, filtrando os dispositivos candidatos que não atendem as funcionalidades e gerando uma lista de dispositivos que atendem ao menos uma funcionalidade selecionada. A geração de cenários é feita com base na somatória de combinação de funcionalidades.

Cada dispositivo selecionado para troca possui uma ou mais funcionalidades integradas que devem ser cobertas pelo novo dispositivo ou combinação de candidatos. O levantamento e identificação de funcionalidades apresenta quais dispositivos e combinações podem ser utilizadas na exploração de cenários, delimitando inicialmente, o espaço de projeto para geração de cenários compatíveis em funcionalidade.

A Figura 35 apresenta a relação entre os atributos de funcionalidade, dispositivos selecionados e candidatos. Por exemplo, o atributo de funcionalidades é representado pelo conjunto F que contem as funcionalidades do sistema. O conjunto D_s que contém os dispositivos selecionados e possui relação com o conjunto F representando

quais funcionalidades cada dispositivo possui. O conjunto D_c representa os dispositivos candidatos à troca, onde são indicados a relação dos dispositivos selecionados e candidatos. As funcionalidades f_1 e f_2 são cobertas por Dev_1, f_2 e f_n são cobertas por Dev_2. O dispositivo candidato Dev_1 possui as funcionalidades f_1, f_2 e f_3 e pode trocar os dispositivos Dev_1 e Dev_2, gerando uma nova combinação de cenários.

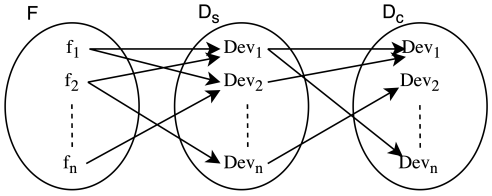


Figura 35 – Conjunto de funcionalidades e relações com dispositivos.
Fonte: Figura do autor.

***Atividade* 4**- *Análise de portas e barramentos* - Ocorre a varredura e identificação dos dispositivos de S&A candidatos e combinações que possuem interfaces compatíveis com os dispositivos selecionados. São avaliadas as características de interface como fonte de alimentação, consumo de corrente, portas de entrada e saída, barramentos de comunicação, e tipo de conector físico. O resultado desta avaliação fornece dados quanto às diferenças de hardware entre os dispositivos, utilizados posteriormente na análise de exploração dos cenários, abordado no próximo Capítulo.

***Atividade* 5**- *Análise de processos, tarefas e dados* - O comparativo entre os dispositivos selecionados e candidatos é realizado, identificando diferenças de compatibilidade de software. São levantados os tipos de dados, processos e tarefas alocados nas plataformas embarcadas. Os tipos de dados são avaliados identificando a modelagem realizada para representar a leitura e escrita dos sinais. O tipo de protocolo de comunicação e formatação do pacote de dados são identificados e comparados.

O tipo de dado utilizado pelo dispositivo de S&A possui relação com a definição das portas e ligações dos processos e tarefas da arquitetura. Com isso, avaliar os dados dos dispositivos selecionados e candidatos, identificando as diferenças na modelagem dos dados, fornece uma visão de quais processos e tarefas devem ser modificados para atender a troca. Os dispositivos que possuem novos processos e tarefas integrados são identificados e listados para, posteriormente, realizar a avaliação de realocação dos processos e tarefas da arquitetura selecionada.

***Atividade* 6**- *Análise de propriedades* - É realizada a avaliação dos atributos definidos no campo de propriedades do modelo AADL referente aos parâmetros físicos, mecânicos e específicos dos dispositivos de S&A provenientes da OAS.

A OAS apresentada no capítulo 5 fornece um conjunto de vinte e três propriedades relacionadas aos dispositivos de S&A e foram utilizadas na definição das propriedades do modelo AADL. Essas propriedades são analisadas e comparadas com os dispositivos candidatos disponíveis na biblioteca com intuito de identificar as

diferenças de valores e cumprimento dos requisitos de projeto.

***Atividade 7**- Geração das conexões de hardware* - Esta atividade realiza a criação das conexões dos dispositivos de S&A candidatos com a arquitetura implementada para identificar a compatibilidade de portas de entrada e saída, barramentos de comunicação e fontes de alimentação. A avaliação de compatibilidade de interface, realizada na atividade 4, listou as portas e barramentos dos dispositivos que fornecem para esta atividade as informações para avaliar a interface com arquitetura do sistema. A criação das conexões ocorre com a varredura das portas e barramentos dos dispositivos candidatos, identificando a compatibilidade por meio do comparativo de tipo de porta e barramento fornecido pelos processadores e subsistemas da arquitetura. Os dispositivos que possuem compatibilidade de interface com o sistema, ou seja, que as portas e barramentos de comunicação requisitados pelos dispositivos sejam fornecidos pelo sistema, são utilizados para geração dos cenários.

Caso existam portas e barramentos do sistema que não sejam compatíveis, os dispositivos e combinações são sinalizados com avisos de incompatibilidade de hardware. Entretanto, para ampliar o espaço de cenários, os dispositivos e combinações incompatíveis realizam a conexão com a arquitetura com a inclusão de um componente integrador do tipo dispositivo que realiza a conversão de portas de entrada e saída e barramentos de comunicação, habilitando a compatibilidade de interface. Ocorre também a criação de um conjunto de conexões adicionais para fazer a ligação entre os dispositivos de S&A e o dispositivo conversor, assim como do conversor para o sistema implementado. A Figura 36 apresenta as duas situações de conexões entre os dispositivos e o sistema. Dessa forma, amplia-se a capacidade de exploração de cenários com o registro de avisos e adição de dispositivos na arquitetura que podem ser avaliados posteriormente.

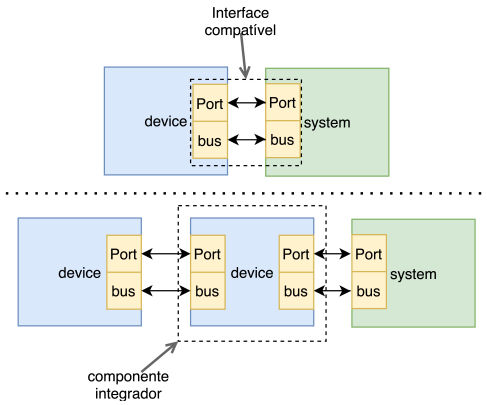


Figura 36 – Conexões entre dispositivos e sistema.
Fonte: Figura do autor.

A avaliação de hardware realiza o comparativo entre as interfaces dos dispositivos e combinações candidatos com as interfaces disponíveis na arquitetura do sistema efetuando uma varredura das portas de entrada e saída, fontes de alimentação e bar-

ramentos de comunicação. Caso os dispositivos de S&A candidatos e combinações não possuam interface com a arquitetura selecionada, ocorre o registro de um aviso na memória indicando que não ocorreu a compatibilidade de interface com o candidato. Para isso, um novo componente é sugerido com intuito de efetuar a integração com a arquitetura do sistema. O componente integrador tem por função realizar a compatibilidade de interface entre o hardware do novo dispositivo e as interfaces disponíveis na arquitetura. Dessa forma, a avaliação do hardware efetua a varredura e comparação de interfaces compatíveis e sugere o uso de componentes integradores para então efetuar a exploração de um espaço maior de cenários, incluindo os dispositivos que possuem incompatibilidade de interface.

***Atividade* 8**- *Análise do tipo de dados* - Nessa atividade é realizado o levantamento dos componentes adicionais necessários para realizar a compatibilidade dos dados dos dispositivos candidatos com os processos e tarefas da arquitetura. Anteriomente, na atividade 5, detalhou-se a avaliação dos dados entre os dispositivos selecionados e candidatos, levantando as diferenças existentes, fornecendo assim uma lista de quais dispositivos possuem diferenças quanto ao tipo de dado.

Com isso, para solucionar os dados incompatíveis, são definidos a implementação de empacotadores (wrappers) que possuem a função de realizar a compatibilidade dos dados fornecidos com o sistema. Com a definição dos empacotadores é possível avaliar quantos processos, tarefas e dados necessitam utilizar empacotadores, sendo realizado o levantamento quantitativo que provê suporte à análise dos cenários.

***Atividades* 9**- *Análise de tarefas* - Nessa atividade é realizada a análise das tarefas que foram impactadas com a troca dos dispositivos de S&A candidatos. Cada cenário criado é avaliado identificando os componentes de software, tarefas, que foram modificados com as trocas.

As tarefas possuem em suas propriedades um conjunto de parâmetros temporais e de desempenho que são definidos, em sua maioria, de acordo com as especificações técnicas dos dispositivos de S&A como período e de latência dos dados. Com isso, após a identificação de quais tarefas foram impactadas é realizada uma análise comparativa entre os dispositivos trocados e diferenças encontradas indicando um aviso de ocorrência do aumento ou diminuição dos valores numéricos declarados.

***Atividades* 10**- *Análise de processos* - Nessa atividade ocorre a análise dos processos de cada cenário gerado que foram impactados na troca dos dispositivos candidatos. As portas de entrada e saída dos processos e composição de tarefas ligados aos dispositivos de S&A trocados são levantados, identificando assim o impacto das mudanças realizadas.

***Atividades* 11**- *Análise de ligações de software* - As tarefas e processos que tiveram impacto com a troca dos dispositivos de S&A possuem um conjunto de ligações que estruturam o fluxo dos dados desde a entrada no processo, processamento das

tarefas e saída dos dados do processo. Com isso, após o levantamento das tarefas e processos impactados são identificadas também as ligações com intuito de avaliar o quantitativo de mudanças necessárias no software para realizar a compatibilidade com o sistema.

***Atividade* 12**- *Análise de realocação de recursos* - A realocação de recursos tem por objetivo identificar potenciais modificações da arquitetura atual em busca de integrar os novos dispositivos de S&A. Considerando a evolução tecnológica, dos dispositivos que integram componentes de software e aprimoram interfaces de hardware, a realocação dos recursos da arquitetura atual pode ser realizada.

A realocação dos processos e tarefas impactam diretamente no comportamento e desempenho do sistema, sendo sugerida posteriormente a realização de análises dos atributos de desempenho. Caso os dispositivos de S&A e combinações não possuam embarcado componentes de software compatíveis com o sistema, os cenários permanecem inalterados com foco na realocação de barramentos.

A ferramenta *DevCompatibility* disponibiliza ao projetista a seleção do processador e barramento ao qual o novo dispositivo pode ser realocado. Os sistemas que possuem mais de um processador e barramentos disponíveis na arquitetura podem ser selecionados com objetivo de efetuar o remanejamento do novo dispositivo trocado. Para isso, o projetista deve clicar no botão “Bus & CPU” para ter acesso aos componentes disponíveis e selecionar a nova configuração da arquitetura. A Figura 37 apresenta um exemplo dos processadores e barramentos disponíveis na arquitetura do projeto ProVant 4.0 no qual o projetista pode definir o novo deployment indicando o barramento e CPU que deseja utilizar.

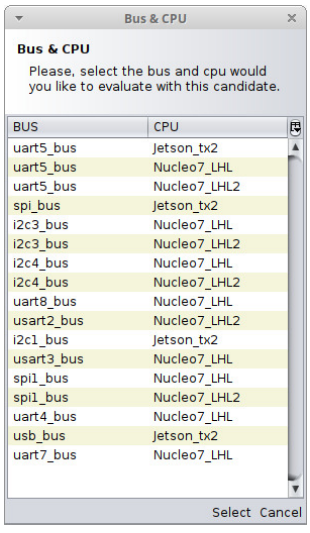


Figura 37 – Lista dos barramentos e CPU disponíveis.
Fonte: Figura do autor.

Vale ressaltar que caso não sejam realizadas modificações quanto aos barramentos e processador, a ferramenta considera a utilização dos componentes previa-

mente utilizados pelo componente selecionado para a troca.

***Atividade* 13**- *Geração de cenários* - Nessa atividade ocorre a geração dos cenários mediante os parâmetros de implementação definidos e análises automatizadas realizadas durante as atividades anteriores.

Tendo como base os dados objetivos das avaliações realizadas nas atividades anteriores, armazenados na memória interna da ferramenta, e parâmetros de projeto definidos, a ferramenta *DevCompatibility* apresenta um conjunto de alternativas de combinações de cenários indicando quais dispositivos e interfaces de hardware e software são compatíveis. Caso sejam realizadas modificações na interface para habilitar a compatibilidade do dispositivo com a arquitetura, uma mensagem sugerindo a inclusão de um dispositivo de hardware do tipo conversor. Quanto ao software, no caso de candidatos que apresentam incompatibilidade, a ferramenta sugere a realização de ***wrapper*** para compatibilidade da arquitetura. A Figura 38 apresenta as alternativas de conexão de interface e software para cada componente candidato, indicando por meio de mensagens de aviso as modificações necessárias para realizar a compatibilidade com o sistema. Dessa forma, o projetista pode visualizar alternativas de interface antes de realizar a geração dos cenários.

No caso do projeto ProVant 4.0, a troca dos motores não necessita realizar modificações de interface de hardware e software devido a porta de comunicação e dados serem os mesmos utilizados na arquitetura atual. Entretanto, avaliar os dados técnicos dos motores podem fornecer detalhes quanto à troca do controle da velocidade (ESC), dispositivo no qual enviamos dados da controladora de voo para acionamento dos motores, impactando diretamente na modificação do software.

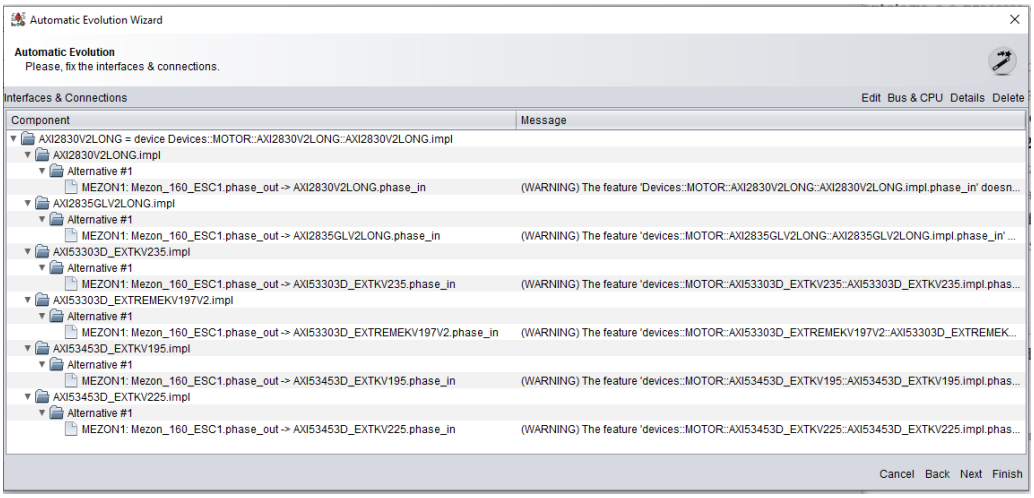


Figura 38 – lista dos cenários identificados com interface compatível.
Fonte: Figura do autor.

Clicando no botão “Next”, a ferramenta gera automaticamente os cenários conforme apresentado na Figura 39. As possíveis combinações de cenários são apresentadas na esquerda da janela indicando o número de cenários (*evolutions*) que foram

gerados em conjunto com uma mensagem contendo os tipos de modificações realizadas. O projetista pode selecionar um dos cenários criados e obter mais detalhes da integração e refino da arquitetura, clicando na aba "*Change Details*".

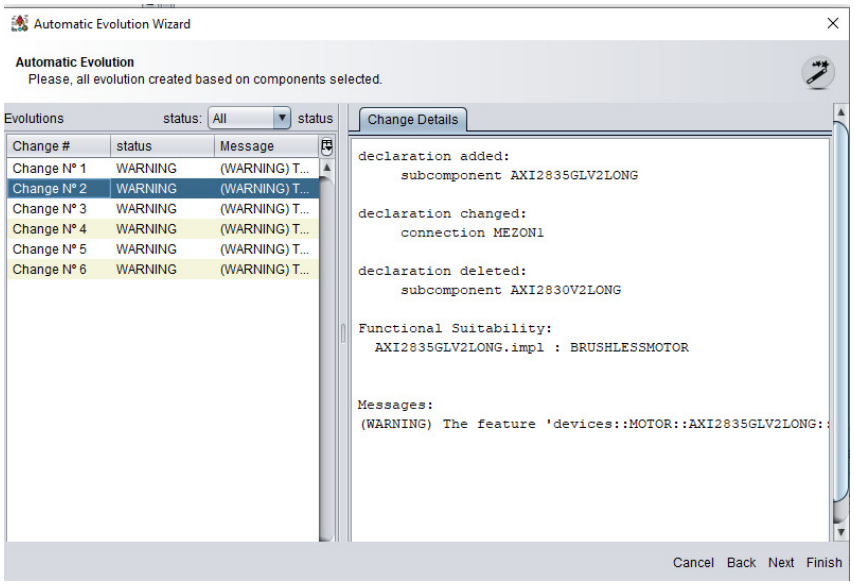


Figura 39 – Lista de possíveis cenários compatíveis criados.
Fonte: Figura do autor.

Um conjunto de atributos foram extraídos dos cenários para quantificar e identificar as diferenças arquitetônicas que demonstram a incompatibilidade de portas, tipos de dados e barramentos, assim como a inclusão de dispositivos de integração e empacotadores de componentes de software utilizados para integração dos dispositivos candidatos. A Tabela 13 apresenta o conjunto de atributos elicitados na exploração de cenários e sistema implementado. Esses atributos são utilizados para realizar o ranqueamento das cenários.

Atributo Elicitado	Detalhes	Atributo Elicitado	Detalhes
Conexões do sistema	Número de conexões declaradas no sistema implementado.	Sistemas implementados	Número de sistemas declarados como subcomponentes do sistema implementado.
Conexões de hardware	Número de conexões declaradas entre os dispositivos, plataforma embarcada, memórias e barramentos.	Funcionalidades declaradas	Número de funcionalidades declaradas nos dispositivos implementados nos subcomponentes do sistema.
Conexões de software	Número de conexões realizadas entre os processos e tarefas implementadas no sistema.	Funcionalidades descobertas	Número de funcionalidades descobertas com a troca dos dispositivos selecionados no sistema implementado.
Dispositivos implementados	Número de dispositivos declarados como subcomponentes da arquitetura implementada.	Empacotadores de software	Número de empacotadores adicionados para realizar a compatibilidade de software dos dispositivos trocados.
Barramentos implementados	Número de barramentos fornecidos e requisitados que são definidos no sistema implementado.	Dispositivos de integração	Número de dispositivos adicionados para realizar a conversão de interface entre o dispositivo e o sistema implementado.
Processos implementados	Número de processos declarados como subcomponentes da arquitetura implementada.	Portas incompatíveis	Número de portas de entrada e saída dos dispositivos trocados que são incompatíveis com o sistema implementado.
Tarefas implementadas	Número de tarefas implementados nos processos, suas conexões externas e ligações internas.	Barramentos incompatíveis	Número de barramentos de comunicação dos dispositivos trocados que são incompatíveis com o sistema implementado.

Tabela 13 – Lista de atributos extraídos dos cenários e sistema selecionado.

Adicionalmente à lista de atributos gerada e atributos específicos definidos na atividade 6, foram realizadas análises de consumo da largura de banda dos barramentos e uso dos processadores dos cenários a partir das definições de alocação dos recursos definidos nas atividades 1 e 2. O resultado das análises e conjunto de atributos da exploração de cenários são utilizados para realizar a comparação com o

sistema selecionado. O resultado das análises dos atributos são utilizados para avaliar a otimização da arquitetura apresentada a seguir.

6.2.3 Etapa de Otimização de cenários

Nesta etapa ocorre o ranqueamento dos cenários gerados a partir da seleção de um conjunto de múltiplos objetivos e definição de pesos. Como resultado desta etapa, o projetista seleciona qual dos cenários deseja gerar o modelo AADL.

Seleção de objetivos

Com base nos cenários gerados previamente, o projetista seleciona os múltiplos-objetivos de tomada de decisão. As propriedades e atributos de qualidade quantitativos obtidos dos resultados das análises das atividades anteriores são apresentados ao projetista.

Na sequência, o projetista seleciona quais atributos de qualidade deseja utilizar para compor o ranqueamento dos cenários a partir da definição de pesos. O valor do peso deve ser entre 0 e 1, onde o valor maior representa o atributo de maior importância, sendo que o somatório dos pesos deve ser igual a 1.

A Figura 40 apresenta um exemplo dos atributos do ProVant 4.0 e definição de pesos para os seguintes atributos: Capacidade de levantamento *LIFT3D* no valor de 0.6; Consumo de potência *Power Consume* com o valor 0.2 e máxima eficiência *Max Efficiency* no valor de 0.2. A janela à esquerda da tela, apresenta as colunas: características ao qual os atributos pertencem; atributos ou propriedades dos componentes e faixa de valores obtidos nas análises de cada cenário; e referência (*reference*) que apresenta o valor máximo que o atributo pode ter, sendo extraído do modelo ReqSpec; A coluna *score* indica os valores entre 0 e 1 ao qual o projetista define os pesos.

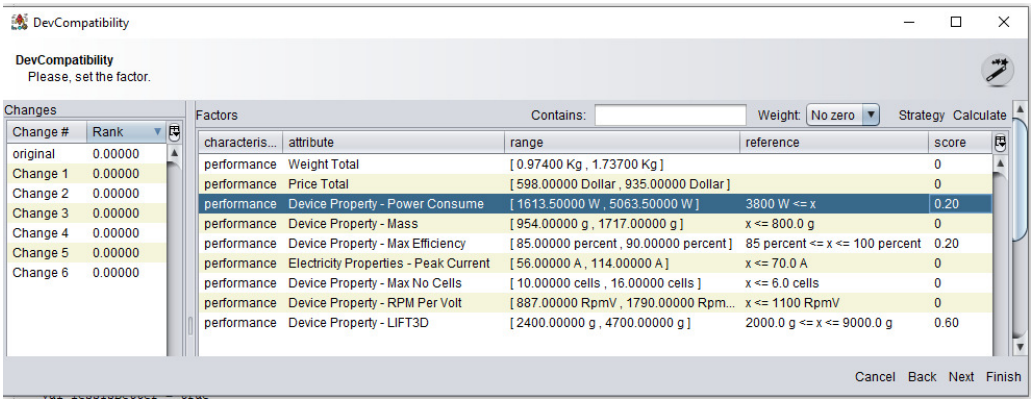


Figura 40 – Atributos e pesos do ProVant 4.0.
Fonte: Figura do autor.

A ferramenta possibilita ao usuário adicionar estratégias de definição de pesos para efetuar o ranqueamento dos cenários no código fonte. A Figura 41 apresenta

duas estratégias implementadas, sendo a primeira a definição de pesos comuns a todos os atributos e a segunda uma proporção dependendo da quantidade de atributos selecionados.

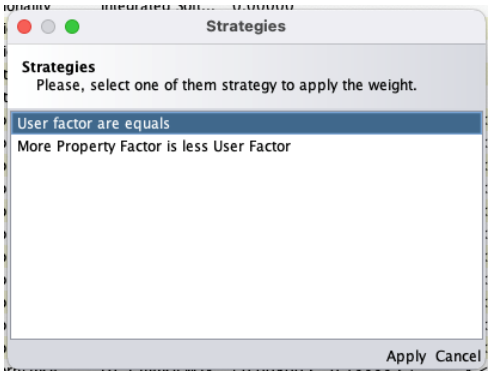


Figura 41 – Definição das estratégias de ranqueamento dos cenários.
Fonte: Figura do autor.

Algoritmo de programação de meta ponderada

Após a definição dos pesos é executado um algoritmo de programação de meta ponderada, do inglês *Weighted Goal Programming Algorithm*, onde são realizados cálculos matemáticos de ponderação, classificação e pesos dos atributos quantitativos escolhidos. O cálculo do ranqueamento de cenários é obtido com base no cálculo de três pesos, sendo eles: peso calculado, peso definido e peso da propriedade, sendo apresentados em mais detalhes na sequência.

Peso Calculado

Inicialmente, é realizado o tratamento dos valores numéricos de cada atributo selecionado com intuito de normalizar e identificar o peso calculado. Para cada cenário é realizada a varredura dos atributos selecionados em busca de identificar o maior e menor valor numérico com intuito de avaliar o peso calculado, conforme mostrado na Equação (5).

$$Pesocalculado = 1/(Valor_{max} - Valor_{min}) \tag{5}$$

Considerando que o $Valor_{max}$ e $Valor_{min}$ são os valores máximo e mínimo encontrados da mesma propriedade entre as evoluções. Para entender melhor, vejamos o seguinte exemplo: um dos atributos selecionados é a “quantidade de componentes”. Supondo que, entre os cenários, o maior número desse atributo foi 32 e o menor foi 22; dessa forma, teremos um peso calculado de 0,10 (resultado de $1/(32-22)$).

Outro exemplo: a “quantidade de conexões” é outra propriedade avaliada. Considerando que, entre os cenários,o maior número de conexões foi de 25 e o menor 23;

obteremos um peso calculado de 0,5 (resultado de $1/(25-23)$). Como pode ser visto nesses dois exemplos, quanto maior o distanciamento entre o valor máximo e o valor mínimo, menor o peso calculado. Em outras palavras, o peso calculado próximo a um significa menor distanciamento entre o valor máximo e mínimo encontrados no mesmo atributo entre os cenários; enquanto o valor próximo de zero, maior o distanciamento entre o valor máximo e mínimo. Esta métrica auxilia o projetista a identificar potenciais cenários considerando a análise de cada propriedade.

Peso Definido

Após obter os pesos calculados de todas as propriedades, a ferramenta mostra para o projetista todas as propriedades sumarizadas com o respectivo peso calculado. Observando essas informações, o projetista definirá o seu peso para cada propriedade (chamado de peso definido). O peso definido será utilizado para ranquear as evoluções na próxima parte e o seu somatório deve ser igual a um. Por exemplo, se duas propriedades tiverem peso definido igual a 0,5, as demais devem ter o peso definido igual a zero.

Existem duas formas de atribuir o peso definido: de forma manual ou forma calculada. Como o nome sugere, o projetista vai definir manualmente o valor entre 0 e 1 para cada propriedade. A segunda forma é utilizar cálculos já pré-definidos em conjunto de propriedades selecionadas pelo projetista.

Existem dois cálculos preferidos denominados equals e less is better. O cálculo denominado equals irá atribuir o mesmo peso para todas as propriedades (Peso Definido =1 quantidade). Por exemplo, se o projetista selecionar duas propriedades, o cálculo equals irá definir 0.5 para cada uma. Desta maneira, as propriedades com menor peso calculado terão uma maior influência no ranqueamento das evoluções.

O cálculo denominado less is better é baseado no peso calculado: quanto maior o valor do peso calculado, menor o valor do peso definido.

$$PesoDefinido = 1 - (PesoCalculado/(\sum_{i=0}^{n} PesoCalculado_n) \quad (6)$$

Peso da propriedade

Existem duas equações para calcular o peso da propriedade. O cálculo utilizado vai depender do tipo da propriedade, as quais podem ser separadas em dois tipos: tipo 1 e tipo 2. O tipo 1 considera que quanto maior o valor da propriedade, melhor. o tipo 2 considera o inverso: quanto menor o valor, melhor. Por exemplo, a “propriedade largura de banda” é tipo 1; enquanto a propriedade “quantidade de componentes” e

“propriedade de custo” são tipo 2.

$$PesoTipo1 = PesoDefinido(valor - Valor_{min}) \quad (7)$$

$$PesoTipo2 = 1 - (PesoDefinido(valor - Valor_{min})) \quad (8)$$

ranqueamento dos cenários

O peso de uma evolução para elaborar o ranqueamento. Esse peso está entre 0 e 1, sendo os valores 0 menos importantes, e os valores próximos de 1, os mais importantes. Além das evoluções, o peso do sistema sem nenhuma modificação (chamado de sistema original) também é calculado. Assim, o projetista pode observar as evoluções que possuem um peso melhor do que o sistema original. O ranqueamento do cenário é o produto dos pesos das propriedades da evolução.

$$RanqueCenario = PesoPropriedade_0 x PesoPropriedade_1 x ... PesoPropriedade_n \quad (9)$$

Ao definir os pesos (score), o projetista realiza o ranqueamento dos cenários clicando no botão “Calculate”, que executa um algoritmo dos pesos e objetivos e selecionados, gerando o ranqueamento de cada cenário gerado. Ao clicar na coluna “Rank” o projetista pode organizar os cenários por peso e identificar quais cenários possuem o melhor rank quando comparado com a arquitetura inicial (original) conforme apresentado na Figura 42.

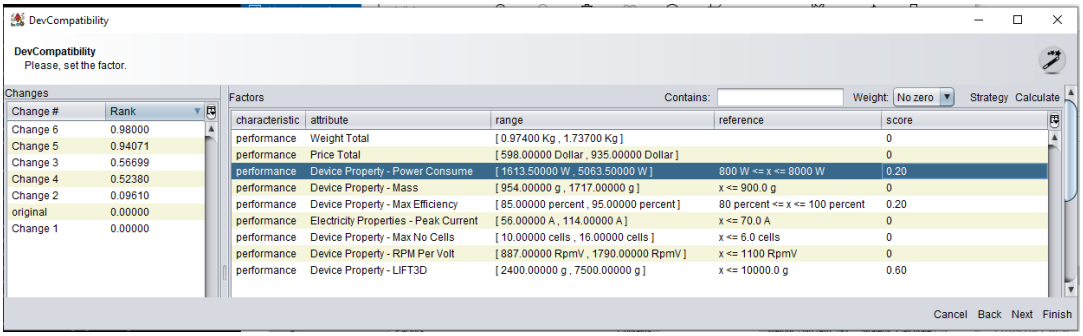


Figura 42 – Definição de pesos e cálculo do ranqueamento de cenários.
Fonte: Figura do autor.

Após a classificação dos cenários por ranqueamento de pesos o projetista pode visualizar mais detalhes quanto aos cenários que obtiveram melhor resultado clicando no botão **“Next”** e assim efetuar a seleção de um dos cenários que deseja utilizar como evolução da arquitetura, descrito na *Etapa 04: Atividade 4*. A Figura 43 apresenta o cenário que obteve o maior valor de ranqueamento, cenário 6 com a escolha do motor AXI53453D_EXTKV225. As análises realizadas nas atividades anteriores são

apresentadas em detalhes ao projetista com intuito de registrar as modificações necessárias para integração do novo componente do sistema, indicando as conexões que foram modificadas, funcionalidades que foram alteradas e modificações de interface de software e hardware.

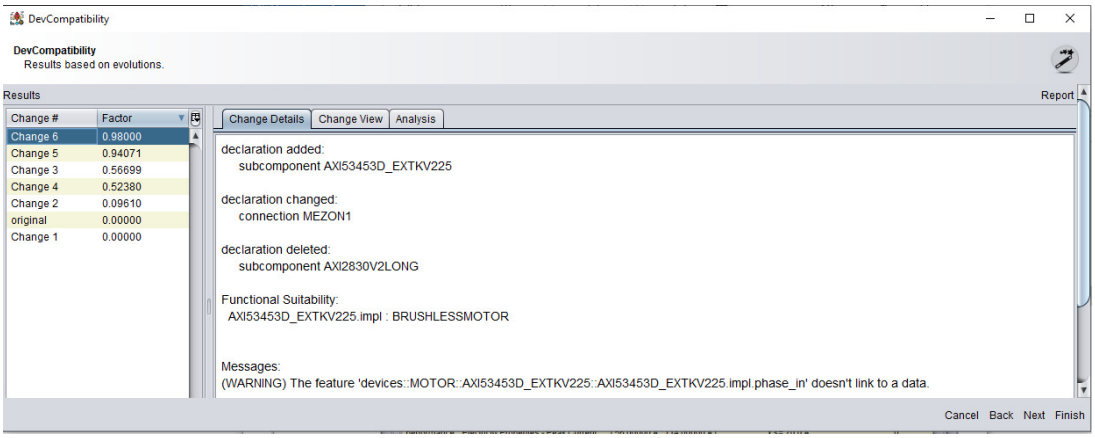


Figura 43 – Detalhes das modificações realizadas em cada cenário do ranqueamento.
Fonte: Figura do autor.

Na aba **"analysis"**, a ferramenta apresenta o comparativo entre o cenário selecionado e a arquitetura inicial (original). O resultado das análises dos atributos dos componentes e sistema são mostrados ao projetista com intuito de avaliar o impacto das modificações, apresentando em porcentagem as diferenças encontradas entre o cenário selecionado e arquitetura inicial. A Figura 44 apresenta o resultado das análises dos atributos do sistema. Por exemplo, o levantamento de peso, LIFT3D, relacionado ao dispositivo de atuação, demonstra que a escolha do cenário 6 para implementação pode melhorar em 212,5% a capacidade deste atributo.

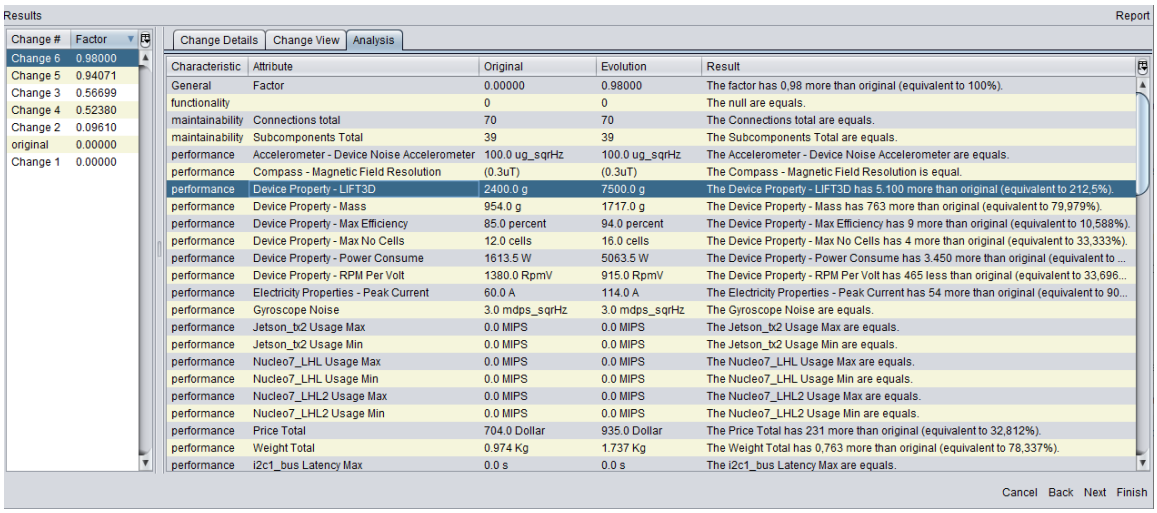


Figura 44 – Análise comparativa do cenário atual e selecionado.
Fonte: Figura do autor.

É importante destacar que a seleção do cenário com base nos resultados das

análises serve como guia ao projetista para escolha da nova arquitetura. Entretanto, é necessário observar as restrições de projeto da arquitetura atual quanto a capacidade de modificação e custos relacionados (por exemplo: tempo de implementação e recursos financeiros). Com isso, a partir do relatório de modificações, o usuário pode quantificar e selecionar qual cenário atende aos novos requisitos de projeto.

Por exemplo, arquiteturas fundamentadas no conceito modular possibilitam ao projetista realizar a troca de componentes do sistema de forma prática e simples devido à disponibilidade de barramentos, portas de dados, e conectores de formato padrão que facilitam efetuar alterações nos componentes arquiteturais.

Gerador de modelos AADL

A aba **"Change View"** apresenta uma prévia do modelo arquitetural de cada cenário gerado. A Figura 45 apresenta a visualização do modelo AADL do cenário 6, que obteve o maior valor no ranqueamento para os três objetivos e pesos definidos. Até este momento os dados das análises e modelos gerados encontram-se armazenados na memória interna da ferramenta. Ao clicar no botão **Report** a ferramenta gera um conjunto de pastas contendo a biblioteca de componentes em AADL, o modelo arquitetural AADL e um arquivo contendo os resultados das análises realizadas.

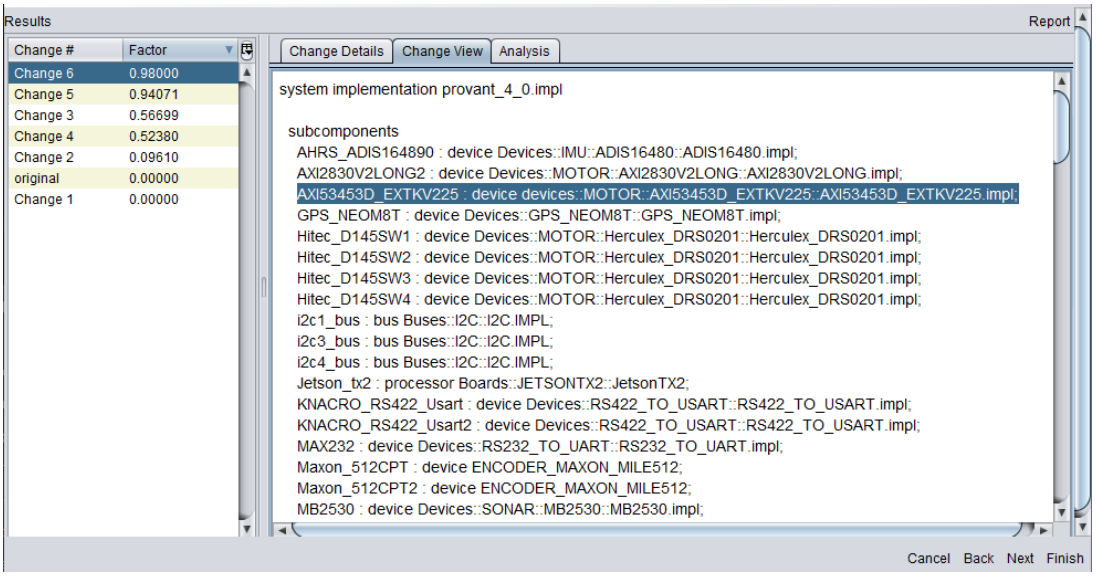


Figura 45 – Modelo AADL do cenário 6, gerado na *DevCompatibility*.
Fonte: Figura do autor.

6.3 SUMÁRIO

A abordagem proposta busca guiar os projetistas no planejamento, desenvolvimento e implementação da evolução de uma arquitetura de sistemas. Essa abordagem

define uma sequência de atividades para representar adequadamente as características e propriedades da arquitetura, fornecendo meios para garantir a evolução do sistema.

Em resumo, a abordagem proposta inicia com a fase de planejamento, definição do plano de evolução e aposentadoria composto por um conjunto de atividades que auxiliam no gerenciamento dos modelos que foram criados para especificar os requisitos e características da AS selecionado. Durante o desenvolvimento da arquitetura evoluída, os dispositivos de S&A selecionados e candidatos são avaliados em busca de identificar a compatibilidade de interface de hardware e componentes de software por meio da análise comparativa entre os componentes e integração com o sistema. Finalmente, o projetista seleciona o cenário que deseja implementar avaliando os resultados das modificações, análises e requisitos do projeto.

Com isso, a avaliação e a análise de cada arquitetura modificada que anteriormente era realizada na ferramenta OSATE2 de forma manual com a modelagem de cada uma das arquiteturas dos cenários utilizando o refino ou extensão do dispositivo de S&A selecionado, gerando resultados para cada cenário, foi automatizada com a geração dos cenários e análise com o plug-in proposto nesta tese.

7 CONSIDERAÇÕES FINAIS E ATIVIDADE FUTURAS

Nesta tese, foram apresentadas contribuições focadas na melhoria do desenvolvimento de arquitetura que usam intensivamente dispositivos de S&A, apresentando um estudo de caso aplicado ao projeto de um VANT. As contribuições propostas foram definidas após uma extensiva análise de diferentes métodos e abordagens de desenvolvimento de AS. Sendo realizados estudos avaliando a modelagem e representação em ontologia dos componentes que constituem a AS. Assim como, estudos quanto à troca dos componentes de hardware em operação em busca de aprimorar os atributos e prolongar o ciclo de vida do projeto. Baseado nesses estudos, foi observada a existência de algumas lacunas na representação e na cobertura dos componentes na abordagem baseada em ontologia. Da mesma forma, abordagens de desenvolvimento de AS com uso de ADM (Arquitetura Dirigida por Modelos) que não cobrem em detalhes como ocorre a análise e troca de componentes da arquitetura utilizando a AADL.

Nesse contexto, quatro contribuições principais foram propostas:

1) Uma abordagem de evolução de AS em operação (CAvA), com ênfase na avaliação da troca de um ou mais componentes da arquitetura por candidatos, identificando a compatibilidade de funcionalidade dos componentes, interface de hardware e software da arquitetura e requisitos do sistema. Esta abordagem contempla o rastreamento de requisitos modificados e definição de múltiplos objetivos de projeto na escolha da arquitetura evoluída.

2) Uma OAS para representar os componentes que compõem a AS e seus relacionamentos com a rede e planta física de um CPS. Adicionalmente, foi demonstrado o uso de regras semânticas na realização de avaliações da AS e criação de arquiteturas candidatas, assim como o mapeamento de entidades OWL de diferentes ontologias que deseja-se realizar o alinhamento.

3) Uma abordagem de transformação de modelos OWL para AADL, *OWL2AADL*, foi desenvolvida no escopo desta pesquisa com objetivo de aproveitar os benefícios da modelagem baseada em ontologia na geração automatizada de modelos arquiteturais AADL, contendo uma biblioteca de componentes e conjunto de propriedades. Dessa forma, o projetista pode, por meio da base de conhecimento da OAS, avaliar conceitos formais de representação e relacionamento das partes do CPS e gerar automaticamente o modelo arquitetural.

4) Uma abordagem de troca de dispositivos de S&A foi apresentada para suporte a aplicação das etapas 2, 3 e 4 da CAvA com uso da MDE. Adicionalmente, uma ferramenta plug-in do ambiente OSATE2, chamada *DevCompatiliby*, foi desenvolvida para suporte a abordagem em modelos AADL. A ferramenta executa um conjunto de análises de compatibilidade de hardware, software e atributos com os componentes

candidatos modelados na biblioteca AADL, apresentando um conjunto de cenários compatíveis, onde o projetista define os pesos e objetivos que norteiam a escolha da nova arquitetura evoluída. Na sequência, a ferramenta gera um arquivo AADL da arquitetura selecionada e um relatório detalhando as análises realizadas.

Os experimentos utilizando a abordagem CAvA e ferramentas desenvolvidas foram aplicadas no contexto do projeto ProVANT versão 4.0. No capítulo x foi apresentado a troca do dispositivo AXI2830V2LONG com objetivo de aprimorar as propriedades de desempenho relacionadas a função “tilt-rotor” do sistema. Nesse caso, três propriedades foram avaliadas com seus respectivos pesos: consumo de potência com peso de 0.2, máxima eficiência com peso 0.2 e empuxo vertical “LIFT3D” com peso 0.6. Como resultado do algoritmo WGP, o cenário 6 apresentou o maior valor numérico de ranqueamento, 0.98, na sequência o cenário 5 com 0.94. A escolha das propriedades e definição dos pesos demonstrou que, mesmo atendendo aos novos requisitos de projeto, um conjunto de cenários demonstraram ser aptos à implementação. Com isso, a escolha do cenário pode ser refinada com o ajuste dos pesos e/ou inclusão de novas propriedades que estejam relacionadas a outros requisitos, tais como o peso e preço do dispositivo.

A troca de dispositivos que integram mais de uma funcionalidade, tal como a IMU GY 85, que é composta dos sensores acelerômetro, giroscópio e magnetômetro, também foi alvo de experimentos de aplicação da abordagem CAvA, conforme apresentado no artigo (SALES, Diego C; BECKER, Leandro Buss, 2021). No experimento de troca do dispositivo IMU GY85 da arquitetura do ProVANT versão 3.0, foram modelados oito dispositivos candidatos que cobrem parcialmente as funcionalidades e quatro que cobrem integralmente. Ao total, 12 dispositivos foram levantados e modelados em AADL para compor a biblioteca de exploração de cenários.

No total, foram gerados 126 cenários contendo a análise comparativa com a arquitetura selecionada. A abordagem CAvA amplia a exploração de cenários baseados na avaliação de funcionalidades, interfaces de hardware compatíveis e não compatíveis a serem analisados, de acordo com as propriedades selecionadas e pesos definidos. Nesse caso, foram selecionadas as propriedades de desempenho do acelerômetro, com peso de 0.8, e preço do componente, com peso de 0.2, para avaliação de evolução da arquitetura. No total, dez cenários obtiveram ranque entre 0.95 e 0.98, sendo que, três deles não eram soluções triviais do projetista identificar manualmente e podem ser implementados atendendo aos requisitos, demonstrando que a abordagem e ferramenta auxiliaram no processo de seleção e evolução do sistema.

Espera-se que as contribuições apresentadas nesta tese possam proporcionar benefícios aos projetistas que necessitam realizar a troca de dispositivos de S&A de arquiteturas em operação, pois a abordagem CAvA fornece aos projetistas uma visão das etapas e atividades importantes no processo de planejar, analisar, selecionar e definir

qual cenário atende melhor aos objetivos, pesos e requisitos de evolução do projeto. A ferramenta *DevCompatiliby* fornece suporte semi-automatizado a realização destas atividades, gerando um conjunto de informações quanto à análise de impacto das modificações de cada cenário gerado, que podem ser utilizadas para guiar a escolha da arquitetura evoluída, evitando erros, diminuindo o tempo e esforço de projeto.

Nesse sentido, é importante salientar que o uso da OAS trouxe como benefício a representação detalhada de componentes da arquitetura, suas características, conjunto de propriedades, relacionamentos e dados utilizados. Dessa forma, nota-se que o projetista pode efetuar a modelagem semântica das instâncias, com base na OAS, e utilizá-las no desenvolvimento da arquitetura em modelos AADL, por meio da ferramenta *OWL2AAL*, gerando automaticamente o conjunto de propriedades (property set), biblioteca de componentes e arquitetura do sistema. O reuso de ontologias consolidadas, tal como SOSA, amplia a base de conhecimentos e possibilita integrar as instâncias dos dispositivos de S&A aplicados em diferentes domínios, tal como rede de sensores, ampliando a biblioteca de componentes da AS.

No entanto, no decorrer da presente pesquisa também evidenciaram-se limitações nas abordagens, as quais serão apresentadas a seguir:

Resultado da transformação de modelos OWL para AADL: O processo de transformação de modelos OWL para AADL gera dois arquivos de extensão AADL, sendo um dedicado à propriedade dos componentes e outro de modelagem dos componentes da arquitetura.

Modelagem dos requisitos: A notação ReqSpec possui vários recursos que possibilita a modelagem e especificação da arquitetura em diferentes níveis. Neste trabalho, foi implementada a leitura dos arquivos de modelagem dos requisitos específicos e globais descritos no arquivo *reqspec*. A análise dos requisitos que foram afetados nas mudanças realizadas e baseadas nos requisitos específicos, arquivo *reqspec*, que possui a descrição dos componentes de hardware e suas propriedades. Os demais níveis da notação Reqspec não foram implementados devido à complexidade de integração com o modelo AADL. Os arquivos de objetivos e metas da linguagem ReqSpec podem ser integrados futuramente na abordagem e ferramenta com intuito de cobrir a rastreabilidade dos modelos e objetivos utilizados na definição da nova arquitetura.

Ferramenta DevCompatibility: A abordagem e as ferramentas propostas não fornecem cobertura, no modo semi-automatizado, à inclusão de novos dispositivos de S&A na arquitetura, tendo como objetivo principal avaliar somente a troca por outro dispositivo candidato com no mínimo a mesma funcionalidade. Quanto à inclusão de novos componentes na arquitetura, o modo manual (modo 01) da ferramenta possibilita esta funcionalidade, entretanto, sem a supervisão das atividades por meio das telas de acompanhamento da abordagem.

Atualmente, a ferramenta possibilita avaliar uma única arquitetura implementada

por vez, não contemplando a avaliação do impacto em subsistemas.

Na **CAvA**, as etapas e atividades da abordagem seguem um fluxo linear onde os resultados de cada atividade alimentam a atividade seguinte. Entretanto, dependendo das ferramentas e técnicas aplicadas, pode ser feita a realimentação dos resultados em diferentes etapas a critério dos projetistas. Por exemplo, a atividade 5.1 de testes nas arquiteturas candidatas, fornece resultados importantes que podem impactar nas atividade de análise dos atributos de qualidade, atividade 3.4, e com isso podem ser utilizados para ajustar novos valores.

ATIVIDADES FUTURAS

A seguir, é apresentada uma lista de possíveis trabalhos futuros relacionados a esta tese:

1. Revisão da proposta de projeto da metodologia, melhorando as características de suporte ao reuso de entidades. Uma avaliação automatizada do processo de avaliação do reuso pode ser desenvolvida com intuito de melhorar a integração no projeto de uma nova ontologia, evitando a importação completa de entidades que podem possuir significados similares e expandindo a representação dos demais componentes da arquitetura.
2. A ontologia pode fornecer dados que podem ser utilizados por diferentes equipes de desenvolvimento da arquitetura. O ambiente OSATE2 possui um conjunto de plug-ins que, a partir do pacote de propriedades, habilitam a realização de diferentes tipos de avaliações e análises de acordo com o ponto de vista de projeto que deseja-se modelar. Por exemplo, a ferramenta OSATE2 fornece os seguintes plug-ins: avaliações de perigos funcionais (*safety analysis for Functional Hazard Assessments* - FHAs), análise de árvore de falhas (*Fault Tree Analysis* - FTA), análise de efeitos de modos de falha (*Failure Modes Effects Analysis* - FMEA), análise de modo comum e diagramas de blocos de confiabilidade (*Common Mode Analysis, and Reliability Block Diagrams* - RBD). Com isso, pode ser feito um levantamento de ontologias que abordem estas análises com intuito de reuso e integração com a OAS e expandir a capacidade de avaliar os cenários.
3. Estender a ferramenta OWL2AADL com o uso de uma interface gráfica guiada para integrar o processo de reuso de ontologias com o mapeamento e alinhamento com as entidades do domínio de AS. Dessa forma, o projetista pode realizar ajustes e correções antes de efetuar a transformação de modelos e expandir a base de conhecimentos.
4. Continuidade nos estudos de projetos de AS baseado em ontologia com objetivo de utilizar as regras SWRL no processo de seleção dos componentes

candidatos e arquiteturas. Estudar a implementação de um novo conceito intitulado "engenheiro virtual", onde uma ferramenta com inteligência artificial baseada nas regras SWRL e rodando na nuvem pode analisar cenários de AS e criar uma base de conhecimento de componentes a partir dos dados semânticos da web. Desta forma, a inteligência artificial pode analisar a compatibilidade de interface, funcionalidade e criar instâncias de arquiteturas candidatas capazes de atender aos requisitos de projeto.

5. Melhorar a ferramenta *DevCompatibility* com a inclusão da nova funcionalidade de análise de subsistemas da arquitetura. A abordagem de troca dos dispositivos de S&A é aplicada para um único sistema implementado selecionado. Dessa forma, estudos futuros buscaram expandir a capacidade de identificar o impacto e ajustes nos componentes de software dos subsistemas da arquitetura.

REFERÊNCIAS

ADVENTIUM, Labs. **Demonstration: Combined Use of DSE, RBD, and TSE for Trade Space Analysis**. 2020.

ALANEN, Marcus; LILIUS, Johan; PORRES, Ivan; TRUSCAN, Dragos. Model driven engineering: A position paper. *In*: CITESEER. PROCEEDINGS of the 1 st International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES). Turku Centre for Computer Science. [*S.l.*: *s.n.*], 2004. v. 29, p. 25–29.

ALBERT, Cecilia; BROWNSWORD, Lisa; BENTLEY, David; BONO, Thomas; MORRIS, Edwin; PRUITT, Deborah. **Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview**. Pittsburgh, PA, 2002. Disponível em: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6093`.

ALETI, A.; BJORNANDER, S. *et al.* ArcheOpterix: An extendable tool for architecture optimization of AADL models. *In*: 2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software. [*S.l.*: *s.n.*], 2009. P. 61–71.

ALVES, Carina; CASTRO, Jaelson. CRE: A systematic method for COTS components selection. *In*: RIO DE JANEIRO, BRAZIL. XV Brazilian Symposium on Software Engineering (SBES). [*S.l.*: *s.n.*], 2001.

ALVES, Carina Frota; FILHO, Pinto *et al.* Analysing the Tradeoffs Among Requirements, Architectures and COTS Components. *In*: WER. [*S.l.*: *s.n.*], 2001. P. 20–31.

AS5506, SAE. Architecture analysis and design language (aadl). **Embedded Computing Systems Committee, SAE**, 2004.

BAILIN, Sidney C; HODGSON, Ralph; KELLER, Paul J. Large-Scale Knowledge Sharing for NASA Exploration Systems.

BALABAN, Edward; BANSAL, Prasun; STOELTING, Paul; SAXENA, Abhinav; GOEBEL, Kai F; CURRAN, Simon. A diagnostic approach for electro-mechanical actuators in aerospace systems. *In*: IEEE. 2009 IEEE Aerospace conference. [*S.l.*: *s.n.*], 2009. P. 1–13.

BARBACCI, Mario; CARRIERE, S; LONGSTAFF, T; WEINSTOCK, C; FEILER, P. **Steps in an architecture tradeoff analysis method: Quality attribute models and analysis**. [*S.l.*], 1998.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick *et al.* **Software Architecture in Practice**. [*S.l.*]: Pearson, 2013.

BECKER, LB; FARINES, J; BODEVEIX, J; FILALI, M; VERNADAT, F. Development process for critical embedded systems. **Architecture**, p. 95–108, 2010.

BECKER, Steffen; KOZIOLEK, Heiko; REUSSNER, Ralf *et al.* The Palladio component model for model-driven performance prediction. **Journal of Systems and Software**, Elsevier, v. 82, n. 1, p. 3–22, 2009.

BELAUNDE, Mariano *et al.* **MDA Guide Version 1.0. 1**. [*S.l.*: *s.n.*], 2003.

BENGTSSON, PerOlof; LASSING, Nico; BOSCH, Jan; VLIET, Hans van. Architecture-level modifiability analysis (ALMA). **Journal of Systems and Software**, Elsevier, v. 69, n. 1-2, p. 129–147, 2004.

BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora *et al.* The semantic web. **Scientific american**, JSTOR, v. 284, n. 5, p. 34–43, 2001.

BOEHM, Barry; TURNER, Richard. The incremental commitment spiral model (ICSM): principles and practices for successful systems and software. *In*: ACM. PROCEEDINGS of the 2015 International Conference on Software and System Process. [*S.l.*: *s.n.*], 2015. P. 175–176.

BOTTS, Mike; ROBIN, Alexandre. OpenGIS® Sensor Model Language (SensorML) Implementation Specification. Version 1.0. 0. Open Geospatial Consortium, 2007.

BRUN, Matthias; VERGNAUD, Thomas; FAUGERE, Madeleine; DELATOUR, Jerome. From UML to AADL: an Explicit Execution Semantics Modelling with MARTE, jan. 2008.

CHANDHOKE, Sundeep; HAYLES, Tim; KODOSKY, Jeff; WANG, Guoqiang. A model-based methodology of programming cyber-physical systems. *In*: IEEE. 2011 7th International Wireless Communications and Mobile Computing Conference. [*S.l.*: *s.n.*], 2011. P. 1654–1659.

CHANDRASEKARAN, Balakrishnan; JOSEPHSON, John R; BENJAMINS, V Richard *et al.* What are ontologies, and why do we need them? **IEEE Intelligent Systems and their applications**, IEEE, v. 14, n. 1, p. 20–26, 1999.

CHUNG, Lawrence; COOPER, Kendra. Matching, ranking, and selecting components: A cots-aware requirements engineering and software architecting approach. *In*: INTERNATIONAL Workshop on Models and Processes for the Evaluation of COTS Components at ICSE. [*S.l.*: *s.n.*], 2004. P. 41–44.

CLEMENTS, Paul C. A survey of architecture description languages. *In*: IEEE. PROCEEDINGS of the 8th international workshop on software specification and design. [*S.l.*: *s.n.*], 1996. P. 16–25.

COLOMB, Robert; RAYMOND, Kerry; HART, Lewis; EMERY, Patrick; WELTY, Chris; XIE, Guo Tong; KENDALL, Elisa. The object management group ontology definition metamodel. *In*: ONTOLOGIES for software engineering and software technology. [*S.l.*]: Springer, 2006. P. 217–247.

COMELLA-DORDA, Santiago; DEAN, John C; MORRIS, Edwin; OBERNDORF, Patricia. A process for COTS software product evaluation. *In*: SPRINGER. INTERNATIONAL Conference on COTS-Based Software Systems. [*S.l.*: *s.n.*], 2002. P. 86–96.

COMMITTEE, IEEE Computer Society. Software Engineering Standards; BOARD, IEEE-SA Standards. **IEEE Recommended Practice for Software Requirements Specifications**. [*S.l.*]: IEEE, 1998. v. 830.

COMPTON, Michael *et al.* The SSN ontology of the W3C semantic sensor network incubator group. **Journal of Web Semantics**, Elsevier, v. 17, p. 25–32, 2012.

COMPTON, Michael; HENSON, Cory Andrew; LEFORT, Laurent; NEUHAUS, Holger; SHETH, Amit P *et al.* A survey of the semantic specification of sensors, 2009.

CORCHO, Oscar; FERNÁNDEZ-LÓPEZ, Mariano; GÓMEZ-PÉREZ, Asunción. Methodologies, tools and languages for building ontologies. Where is their meeting point? **Data & knowledge engineering**, Elsevier, v. 46, n. 1, p. 41–64, 2003.

CORTELLESSA, Vittorio; CRNKOVIC, Ivica; MARINELLI, Fabrizio; POTENA, Pasqualina. Experimenting the Automated Selection of COTS Components Based on Cost and System Requirements. **J. UCS**, v. 14, n. 8, p. 1228–1255, 2008.

COX, Simon. Observations and measurements. **Open Geospatial Consortium Best Practices Document. Open Geospatial Consortium**, p. 21, 2006.

DELANGE, J.; FEILER, P. Architecture Fault Modeling with the AADL Error-Model Annex. *In*: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications. [*S.l.*: *s.n.*], 2014. P. 361–368.

ECLIPSE. [*S.l.*: *s.n.*], out. 2018.

ESFAHANI, N. *et al.* GuideArch: Guiding the exploration of architectural solution space under uncertainty. *In*: 2013 35th International Conference on Software Engineering (ICSE). [*S.l.*: *s.n.*], 2013. P. 43–52.

FARAIL, Patrick; GAUFILLET, Pierre; CANALS, Agusti; LE CAMUS, Christophe; SCIAMMA, David; MICHEL, Pierre; CRÉGUT, Xavier; PANTEL, Marc. The TOPCASED project: a toolkit in open source for critical aeronautic systems design. *In*: CONFERENCE ERTS'06. [*S.l.*: *s.n.*], 2006.

FARSHIDI, Siamak; JANSEN, Slinger; JONG, Rolf de; BRINKKEMPER, Sjaak *et al.* A decision support system for software technology selection. **Journal of Decision systems**, Taylor & Francis, v. 27, sup1, p. 98–110, 2018.

FEILER, Peter. Open source aadl tool environment (osate). *In*: AADL Workshop, paris. [*S.l.*: *s.n.*], 2004. P. 1–40.

FEILER, Peter. **The Open Source AADL Tool Environment (OSATE)**. [*S.l.*], 2019.

FEILER, Peter; RUGINA, Ana. **Dependability modeling with the architecture analysis & design language (AADL)**. [*S.l.*], 2007.

FEILER, Peter H *et al.* **A requirement specification language for AADL**. [*S.l.*], 2016.

FEILER, Peter H; GLUCH, David P. **Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language**. [*S.l.*]: Addison-Wesley, 2012.

FEILER, Peter H.; LEWIS, Bruce A; VESTAL, Steve *et al.* The SAE Architecture Analysis & Design Language (AADL) a standard for engineering performance critical systems. *In*: IEEE. COMPUTER Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE. [*S.l.*: *s.n.*], 2006. P. 1206–1211.

FERNANDEZ, Mariano *et al.* Methontology: from ontological art towards ontological engineering. *In*: STANFORD, USA. PROCEEDINGS of the AAAI97 spring symposium series on ontological engineering. [*S.l.*: *s.n.*], 1997. P. 33–40.

FIORINI, Sandro Rama *et al.* Extensions to the core ontology for robotics and automation. **Robotics and Computer-Integrated Manufacturing**, Elsevier, v. 33, p. 3–11, 2015.

GARG, R. A systematic review of COTS evaluation and selection approaches. **Accounting**, v. 3, n. 4, p. 227–236, 2017.

GAŠEVIC, Dragan; DJURIC, Dragan; DEVEDŽIC, Vladan *et al.* **Model driven engineering and ontology development**. [*S.l.*]: Springer Science & Business Media, 2009.

GONCALVES, Fernando. **Integrated Method For Designing Complex Cyber-Physical Systems**. 2018. Tese (Doutorado) – Universidade Federal de Santa Catarina.

GONCALVES, Fernando. **Integrated Method For Designing Complex Cyber-Physical Systems**. 2018. Tese (Doutorado) – Universidade Federal de Santa Catarina.

GONÇALVES, F. S.; BECKER, L. B. Model driven engineering approach to design sensing and actuation subsystems. *In*: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). [*S.l.*: *s.n.*], 2016. P. 1–8.

GRAU, Gemma; CARVALLO, Juan Pablo; FRANCH, Xavier; QUER, Carme *et al.* DesCOTS: a software system for selecting COTS components. *In*: IEEE. PROCEEDINGS. 30th Euromicro Conference, 2004. [*S.l.*: *s.n.*], 2004. P. 118–126.

GRUBER, Thomas R *et al.* A translation approach to portable ontology specifications. **Knowledge acquisition**, Academic Press, v. 5, n. 2, p. 199–220, 1993.

HALLER, Armin; JANOWICZ, Krzysztof; COX, Simon; PHUOC, Danh; LEFRANÇOIS, Maxime. **Semantic Sensor Network Ontology**. [*S.l.*: *s.n.*], out. 2017.

HALLER, Armin *et al.* The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. **Semantic Web**, IOS Press, v. 10, n. 1, p. 9–32, 2019.

HODGSON, Ralph; KELLER, Paul J; HODGES, Jack; SPIVAK, Jack. QUDT-quantities, units, dimensions and data types ontologies. **USA Available http://qudt. org March**, v. 156, 2014.

IACOBUCCI, Joseph. Rapid Architecture Alternative Modeling (RAAM): a framework for capability-based analysis of system of systems architectures. *In*:

IBRAHIM, Hamdy; ELAMY, Abdel-Halim H; FAR, Behrouz H; EBERLEIN, Armin. UnHOS: A method for uncertainty handling in Commercial Off-The-Shelf (COTS) selection. **International Journal of Energy, Information & Communications**, v. 2, n. 3, 2011.

IEEE. ISO/IEC/IEEE International Standard - Systems and software engineering – Software life cycle processes. **ISO/IEC/IEEE 12207:2017(E) First edition 2017-11**, p. 1–157, 2017.

ISO/IEC. **ISO/IEC 25010: 2011 Systems and software engineering–Systems and software Quality Requirements and Evaluation (SQuaRE)–System and software quality models**. [*S.l.*]: CH: ISO Geneva, 2011.

JANOWICZ, Krzysztof *et al.* SOSA: A lightweight ontology for sensors, observations, samples, and actuators. **Journal of Web Semantics**, Elsevier, v. 56, p. 1–10, 2019.

JANOWICZ, Krzysztof; COMPTON *et al.* The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. **SSN**, v. 668, 2010.

JENSEN, Jeff C; CHANG LEE, Edward *et al.* A model-based design methodology for cyber-physical systems. *In*: IEEE. WIRELESS Communications and Mobile Computing Conference (IWCMC), 2011 7th International. [*S.l.*: *s.n.*], 2011. P. 1666–1671.

KAZMAN, Rick; KLEIN, Mark; BARBACCI, Mario; LONGSTAFF, Tom; LIPSON, Howard; CARRIERE, Jeromy. The architecture tradeoff analysis method. *In*: IEEE. PROCEEDINGS. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No. 98EX193). [*S.l.*: *s.n.*], 1998. P. 68–78.

KERZHNER, Aleksandr A. **Using logic-based approaches to explore system architectures for systems engineering**. 2012. Tese (Doutorado) – Georgia Institute of Technology.

KIFER, Michael. Rule interchange format: The framework. *In*: SPRINGER. INTERNATIONAL Conference on Web Reasoning and Rule Systems. [*S.l.*: *s.n.*], 2008. P. 1–11.

KONTIO, Jyrki. OTSO: a systematic process for reusable software component selection. Citeseer, 1995.

KORDON, F.; J. HUGUES A. CANALS, A. Dohet. **Embedded Systems: Analysis and Modeling with SysML, UML and AADL**. [*S.l.*]: Wiley-ISTE, 2013. (ISTE). ISBN 1848215002.

KORDON, Fabrice; HUGUES, Jérôme; CANALS, Agusti; DOHET, Alain. **Embedded systems: analysis and modeling with SysML, UML and AADL**. [*S.l.*]: John Wiley & Sons, 2013.

KOZIOLEK, Anne. **Automated improvement of software architecture models for performance and other quality attributes**. 2014. Tese (Doutorado).

KUNDA, Douglas; BROOKS, Laurence *et al.* STACE: social technical approach to COTS software evaluation. *In*: PROCEEDINGS of the 4th UKAIS Conference. [*S.l.*: *s.n.*], 2003. P. 552–565.

LANUSSE, Agnes; TANGUY, Yann; ESPINOZA, Huascar; MRAIDHA, Chokri; GERARD, Sebastien; TESSIER, Patrick; SCHNEKENBURGER, Remi; DUBOIS, Hubert; TERRIER, François. Papyrus UML: an open source toolset for MDA. *In*: CITESEER. PROC. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009). [*S.l.*: *s.n.*], 2009. P. 1–4.

LAPLANTE, Phillip A *et al.* Standards for the Internet of Things: A case study in disaster response. **Computer**, IEEE, v. 49, n. 5, p. 87–90, 2016.

LEE, Edward; SESHIA, Sanjit. **Introduction to embedded systems: A cyber-physical systems approach**. [*S.l.*]: MIT Press, 2016.

LEE, Edward A. Cyber physical systems: Design challenges. *In*: IEEE. OBJECT oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on. [*S.l.*: *s.n.*], 2008. P. 363–369.

LEE, Edward A. Fundamental limits of cyber-physical systems modeling. **ACM Transactions on Cyber-Physical Systems**, ACM, v. 1, n. 1, p. 3, 2017.

LEE, Edward A; JOHN, II. **Overview of the ptolemy project**. [*S.l.*]: Electronics Research Laboratory, College of Engineering, University of Berkeley, 1999.

LICHOTA, Randall W; VESPRINI, Robert L; SWANSON, Bruce *et al.* PRISM Product Examination Process for component based development. *In*: IEEE. PROCEEDINGS Fifth International Symposium on Assessment of Software Tools and Technologies. [*S.l.*: *s.n.*], 1997. P. 61–69.

LOPES, Denivaldo; HAMMOUDI, Slimane; BÉZIVIN, Jean; JOUAULT, Frédéric. Mapping specification in MDA: From theory to practice. *In*: INTEROPERABILITY of enterprise software and applications. [*S.l.*]: Springer, 2006. P. 253–264.

LOZANO-TELLO, Adolfo *et al.* BAREMO: how to choose the appropriate software component using the analytic hierarchy process. *In*: PROCEEDINGS of the 14th international conference on Software engineering and knowledge engineering. [*S.l.*: *s.n.*], 2002. P. 781–788.

MAJUMDER, Mrinmoy. Multi criteria decision making. *In*: IMPACT of urbanization on water shortage in face of climatic aberrations. [*S.l.*]: Springer, 2015. P. 35–47.

MARTÍN-LAMMERDING, David; CÓRDOBA, Alberto; ASTRAIN, José Javier; MEDRANO, Pablo; VILLADANGOS, Jesús. An ontology based system to collect WSN-UAS data effectively. **IEEE Internet of Things Journal**, IEEE, 2020.

MCGUINNESS, D.L.; FIKES, R.; HENDLER, J.; STEIN, L.A. DAML+OIL: an ontology language for the Semantic Web. **IEEE Intelligent Systems**, v. 17, n. 5, p. 72–80, 2002.

MCGUINNESS, Deborah L; VAN HARMELEN, Frank *et al.* OWL web ontology language overview. **W3C recommendation**, v. 10, n. 10, p. 2004, 2004.

MELLOR, Stephen J; SCOTT, Kendall; UHL, Axel; WEISE, Dirk. **MDA distilled: principles of model-driven architecture**. [*S.l.*]: Addison-Wesley Professional, 2004.

MENS, Tom; VAN GORP, Pieter. A taxonomy of model transformation. **Electronic notes in theoretical computer science**, Elsevier, v. 152, p. 125–142, 2006.

MILES, Alistair; BECHHOFER, Sean. SKOS simple knowledge organization system reference. **W3C recommendation**, World Wide Web Consortium, 2009.

MILOSLAVSKAYA, Natalia; NIKIFOROV, Andrey; PLAKSIY, Kirill; TOLSTOY, Alexander. Standardization issues for the internet of things. *In*: SPRINGER. WORLD Conference on Information Systems and Technologies. [*S.l.*: *s.n.*], 2019. P. 328–338.

MOHAMED, Abdallah; RUHE, Guenther; EBERLEIN, Armin *et al.* COTS selection: past, present, and future. *In*: IEEE. 14TH Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07). [*S.l.*: *s.n.*], 2007. P. 103–114.

MOHAMED, Abdallah; RUHE, Guenther; EBERLEIN, Armin *et al.* MiHOS: an approach to support handling the mismatches between system requirements and COTS products. **Requirements Engineering**, Springer, v. 12, n. 3, p. 127–143, 2007.

MORISIO *et al.* IusWare: A methodology for the evaluation and selection of software products. **IEE Proceedings-Software**, IET, v. 144, n. 3, p. 162–174, 1997.

MUSAVI, Seyyedeh Atefeh; HASHEMI, Mahmoud Reza. An Ontology-Based Method for HW/SW Architecture Reconstruction. **IEEE Transactions on Computers**, IEEE, v. 68, n. 7, p. 1007–1018, 2019.

NCUBE, Cornelius; MAIDEN, Neil AM. PORE: Procurement-oriented requirements engineering method for the component-based systems engineering development paradigm. *In*: INTERNATIONAL workshop on component-based software engineering. [*S.l.*: *s.n.*], 1999. P. 130–140.

NILES, Ian; PEASE, Adam. Towards a standard upper ontology. *In*: PROCEEDINGS of the international conference on Formal Ontology in Information Systems-Volume 2001. [*S.l.*: *s.n.*], 2001. P. 2–9.

NORD, Robert L; BARBACCI, Mario R; CLEMENTS, Paul; KAZMAN, Rick; KLEIN, Mark *et al.* **Integrating the Architecture Tradeoff Analysis Method (ATAM) with the cost benefit analysis method (CBAM)**. [*S.l.*], 2003.

NOY, Natalya F; MCGUINNESS, Deborah L *et al.* **Ontology development 101: A guide to creating your first ontology**. [*S.l.*]: Stanford knowledge systems laboratory technical report KSL-01-05 and . . ., 2001.

OLSZEWSKA, Joanna Isabelle *et al.* Ontology for autonomous robotics. *In*: IEEE. 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). [*S.l.*: *s.n.*], 2017. P. 189–194.

PAN, Jeff Z. Resource description framework. *In*: HANDBOOK on ontologies. [*S.l.*]: Springer, 2009. P. 71–90.

PASSARINI, Rosane Fatima; FARINES, Jean Marie; FERNANDES, Joao M.; BECKER. Cyber-physical systems design: transition from functional to architectural models. **Design Automation for Embedded Systems**, Springer US, v. 19, n. 4, p. 345–366, 2015. ISSN 15728080.

PASTOR, Enric; LOPEZ, Juan; ROYO, Pablo *et al.* A hardware/software architecture for UAV payload and mission control. *In*: IEEE. 2006 ieee/aiaa 25TH Digital Avionics Systems Conference. [*S.l.*: *s.n.*], 2006. P. 1–8.

PERROTIN, Maxime; CONQUET, Eric; DELANGE, Julien; TSIODRAS, Thanassis. TASTE: An open-source tool-chain for embedded system and software development. *In*: EMBEDDED Real Time Software and Systems (ERTS2012). [*S.l.*: *s.n.*], 2012.

PREUVENEERS, Davy *et al.* Towards an extensible context ontology for ambient intelligence. *In*: SPRINGER. EUROPEAN Symposium on Ambient Intelligence. [*S.l.*: *s.n.*], 2004. P. 148–159.

PROCTER, S.; L.WRAGE. Guided Architecture Trade Space Exploration: Fusing Model Based Engineering Design by Shopping. *In*: 2019 ACM/IEEE 22nd International

Conference on Model Driven Engineering Languages and Systems (MODELS). [*S.l.*: *s.n.*], 2019. P. 117–127.

QUER, Carme; FRANCH, Xavier; LOPEZ-PELEGRIN, Xavier *et al.* DesCOTS-EV: a tool for the evaluation of COTS components. *In*: IEEE. 13TH IEEE International Conference on Requirements Engineering (RE'05). [*S.l.*: *s.n.*], 2005. P. 457–458.

REDMAN, David; WARD, Donald; CHILENSKI, John; POLLARI, Greg. Virtual integration for improved system design. *In*: CITESEER. FIRST Analytic Virtual Integration of Cyber-Physical Systems Workshop. [*S.l.*: *s.n.*], 2010. P. 57–64.

REGTIEN, Paul *et al.* **Sensors for mechatronics**. [*S.l.*]: Elsevier, 2018.

ROSS, Jordan A; MURASHKIN, Alexandr; LIANG, Jia Hui; ANTKIEWICZ, Michał; CZARNECKI, Krzysztof. Synthesis and exploration of multi-level, multi-perspective architectures of automotive embedded systems. **Software & Systems Modeling**, Springer, v. 18, n. 1, p. 739–767, 2019.

RUHE, Günther. Intelligent support for selection of COTS products. *In*: SPRINGER. NET. ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World. [*S.l.*: *s.n.*], 2002. P. 34–45.

SALES, Diego C; BECKER, Leandro Buss. Approach for Evolving Sensing and Actuation Devices in Cyberphysical Systems Architectures. *In*: MODELSWARD. [*S.l.*: *s.n.*], 2021. P. 306–313.

SALES, Diego Camara; KOLIVER, Cristian; BECKER, Leandro Buss. Ontology and Rules for Characterization of Sensors and Actuators Devices in AADL Models. *In*: IEEE. 2020 X Brazilian Symposium on Computing Systems Engineering (SBESC). [*S.l.*: *s.n.*], 2020. P. 1–8.

SALES, Diego Câmara; BECKER, Leandro Buss. Systematic Literature Review of System Engineering Design Methods. **SBESC**, p. 6, 2018.

SARKAR, Subhankar Kumar. Architecture Centric Tradeoff - A Decision Support Method for COTS Selection and Life Cycle Management. *In*: ICSEA 2012. [*S.l.*: *s.n.*], 2012.

SCHLENOFF, Craig *et al.* An IEEE standard ontology for robotics and automation. *In*: IEEE. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. [*S.l.*: *s.n.*], 2012. P. 1337–1342.

SCHLENOFF, Craig *et al.* Ieee standard ontologies for robotics and automation, 2015.

SCHMIDT, Douglas C. Model-driven engineering. **COMPUTER-IEEE COMPUTER SOCIETY-**, Citeseer, v. 39, n. 2, p. 25, 2006.

SEIDEWITZ, Edwin. What models mean. **IEEE software**, IEEE, v. 20, n. 5, p. 26–32, 2003.

SELIC, Bran. The pragmatics of model-driven development. **IEEE software**, IEEE, v. 20, n. 5, p. 19–25, 2003.

STAAB, Steffen; WALTER, Tobias; GRÖNER, Gerd; PARREIRAS, Fernando Silva. Model driven engineering with ontology technologies. *In*: SPRINGER. REASONING Web International Summer School. [*S.l.*: *s.n.*], 2010. P. 62–98.

STEINBERG, Dave; BUDINSKY, Frank; MERKS, Ed; PATERNOSTRO, Marcelo. **EMF: eclipse modeling framework**. [*S.l.*]: Pearson Education, 2008.

TAYE, Mohammad Mustafa. Understanding semantic web and ontologies: Theory and applications. **arXiv preprint arXiv:1006.4567**, 2010.

TSOURDOS, Antonios. Flight control systems: Practical issues in design and implementation. **Proceedings of the Institution of Mechanical Engineers**, SAGE PUBLICATIONS, INC., v. 214, n. 5, p. 323, 2000.

UFSC, UFMG. **Provant project**. [*S.l.*: *s.n.*], mai. 2020. `https://provant.paginas.ufsc.br`.

USCHOLD, Michael *et al.* Ontologies: Principles, methods and applications. **The Knowledge Engineering Review**, v. 11, jan. 1996.

WASSON, Charles S. **System engineering analysis, design, and development: Concepts, principles, and practices**. [*S.l.*]: John Wiley & Sons, 2015.

YU, H.; YANG, Y. Latency Analysis of Automobile ABS Based on AADL. *In*: 2012 International Conference on Industrial Control and Electronics Engineering. [*S.l.*: *s.n.*], 2012. P. 1835–1838.

ZHANG, F.; ZHAO, Y.; MA, D.; NIU, W. Formal Verification of Behavioral AADL Models by Stateful Timed CSP. **IEEE Access**, v. 5, p. 27421–27438, 2017. ISSN 2169-3536.