Subsídios para a representação de arquiteturas de referência de sistemas embarcados

Milena Guessi

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 26 de Abril de 2013

Assinatura:_____

Subsídios para a representação de arquiteturas de referência de sistemas embarcados

Milena Guessi

Orientadora: **Profa. Dra. Elisa Yumi Nakagawa**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Abril de 2013

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

G936s

Guessi, Milena
Subsídios para a Representação de Arquiteturas de Referência de Sistemas Embarcados / Milena Guessi; orientadora Elisa Yumi Nakagawa. -- São Carlos, 2013.
176 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2013.

1. Arquitetura de Referência. 2. Representação Arquitetural. 3. Sistemas Embarcados. I. Nakagawa, Elisa Yumi, orient. II. Título.

Dedico este trabalho a Valdir e Marilda.

Agradecimentos

A Deus pelo dom da Vida.

Aos meus pais Valdir e Marilda pelos seus ensinamentos, amor, carinho e confiança que me permitiram chegar até aqui.

À minha irmã Vanessa por seu amor e paciência durante todos esses anos.

Ao Paulo por seu amor, dedicação, compreensão e apoio que me motivaram a seguir esse caminho.

À minha nova família Lélia, Edson, Maria Rita, Gabriel, Maidy, Fábio e Tatiana, por terem me acolhido.

À minha orientadora Profa. Dra. Elisa Yumi Nakagawa, pelos seus ensinamentos, pela confiança e também incrível dedicação.

Ao Prof. Dr. Flavio Oquendo, por sua importante contribuição a este projeto de mestrado.

A todos os meus amigos e, em especial, ao Felipe, Vitor e Thiago José pela companhia e pelo apoio.

A todos os meus amigos e colegas do Labes e, em especial, ao Lucas, Kátia, Bruno e Vinicíus, pela companhia, conselhos e ajuda que foram tão importantes nesse caminho.

Ao CNPq, por ter possibilitado e financiado esta pesquisa.

A todos os professores e funcionários do ICMC.

Foi o tempo que
dedicaste à tua rosa
que a fez tão importante

Antoine de Saint-Exupéry

Resumo

Arquiteturas de referência são construídas por combinarem as melhores práticas, padrões, plataformas e componentes para a construção e padronização de sistemas de software de um determinado domínio. De fato, diversas arquiteturas de referência podem ser encontradas para o domínio de sistemas embarcados, motivadas principalmente pela importância e complexidade crescentes que esses sistemas de software vêm apresentando. Dentre as atividades para elaboração de uma arquitetura de referência, a descrição apropriada dessa arquitetura é essencial para permitir que ela seja de fato utilizada. Contudo, não há na literatura um consenso sobre qual a melhor maneira de descrever arquiteturas de referência do domínio de sistemas embarcados, os tipos de informação que devem ser capturados ou ainda o conjunto de pontos de vista que pode ser construído. Visando sistematizar e padronizar a representação de arquiteturas de referência de sistemas embarcados, este trabalho propõe o método ProSA-Re. O método baseia-se nos resultados de uma revisão sistemática conduzida sobre o assunto e estabelece um conjunto de atividades e diretrizes para a construção da representação de arquiteturas de referência de sistemas embarcados. O método também esclarece o ciclo de vida de arquiteturas de referência, de modo a auxiliar na manutenção e na evolução das representações construídas com o seu apoio. Para ilustrar o ProSA-Re, uma representação da arquitetura de referência de sistemas multiagentes locais foi elaborada. Em seguida, a realização de uma avaliação com especialistas da área de arquitetura de software e um estudo de caso com usuários dessa representação permitiram a identificação de vantagens e limitações desse método. Espera-se que os resultados alcançados nesta dissertação possam contribuir para o reúso do conhecimento arquitetural e o desenvolvimento mais eficiente de sistemas de software do domínio de sistemas embarcados.

Abstract

Reference architectures combine the best practices, standards, platforms, and components to standardize software systems of a given domain. In this sense, reference architectures can be found for embedded systems, motivated by the increasing importance and complexity that these systems must cope with. In particular, the representation of the reference architecture is an essential activity for it to be used in practice. However, there is no consensus in the literature on what is the best way to describe reference architectures of embedded systems, including what types of information should be captured in those descriptions and also the set of viewpoints that could be adopted. Thus, this work establishes ProSA-Re, a method for systematizing and standardizing the representation of reference architectures of embedded systems. ProSA-Re considers the results of a systematic review on this subject and specifies a set of viewpoints, concerns, tasks, and guidelines to describe reference architectures of embedded systems. ProSA-Re also supports the reference architectures' life cycle by clarifying the evolution of architectural descriptions built with it. To illustrate the method, a representation for the reference architecture of situated multiagent systems was built. Then, a case study was conducted to evaluate the resulting representation and specialists were consulted to evaluate the method description. We hope with this method to further improve the reuse of architectural knowledge, thus contributing for the development of software systems in this domain.

Sumário

1 Introdução **1**
1.1 Contextualização 1
1.2 Motivação 4
1.3 Objetivos 5
1.4 Organização 5

2 Arquitetura de Software e Arquitetura de Referência **7**
2.1 Considerações Iniciais 7
2.2 Arquiteturas de Software 8
2.2.1 Conceitos e Terminologia Básicos 8
2.3 Arquiteturas de Referência 10
2.3.1 Tipos de Arquiteturas de Referência 11
2.3.2 ProSA-RA 14
2.3.3 Exemplos de Arquiteturas de Referência 15
2.4 Representação Arquitetural 18
2.4.1 Decisões Arquiteturais 22
2.4.2 Linguagens de Descrição Arquitetural 23
2.5 Considerações Finais 28

3 Arquitetura de Sistemas Embarcados **29**
3.1 Considerações Iniciais 29
3.2 Sistemas Embarcados 30
3.3 Arquiteturas de Referência de Sistemas Embarcados 32
3.3.1 Exemplos 33
3.4 Representação de Arquiteturas de Referência de Sistemas Embarcados . . . 39
3.5 Considerações Finais 49

4 ProSA-Re: Um Método para a Representação de Arquiteturas de Referência de Sistemas Embarcados **51**
4.1 Considerações Iniciais 51
4.2 Descrição do ProSA-Re 52
4.2.1 Produtos de Trabalho Utilizados 54
4.2.2 Papéis Considerados 58

4.2.3 Tarefas Planejadas 59
4.2.4 Diretrizes para Utilização 66
4.3 Considerações Finais 74

5 Avaliação do ProSA-Re 75
5.1 Considerações Iniciais 75
5.2 Aplicação do Método na Representação de uma Arquitetura de Referência 76
5.3 Avaliação da Representação Arquitetural Construída com o Método 78
5.3.1 Planejamento do Estudo de Caso 78
5.3.2 Resultados do Estudo de Caso 79
5.3.3 Discussão dos Resultados da Avaliação da Representação Arquitetural 86
5.4 Avaliação da Descrição do Método 87
5.4.1 Resultados da Avaliação com Especialistas 87
5.4.2 Discussão dos Resultados da Avaliação com Especialistas 90
5.5 Considerações Finais 90

6 Conclusão 93
6.1 Caracterização da Pesquisa Realizada 93
6.2 Contribuições 94
6.3 Dificuldades e Limitações 95
6.4 Trabalhos Futuros 96

Referências 106

A SysML 107

B SPEM 111
B.1 Overview 111
B.2 Elements 112

C Reference Architecture for Situated Multiagents Systems 115
C.1 Introduction 115
C.2 Objectives 116
C.3 Concerns 117
C.4 Architectural Views 117
C.4.1 Conceptual View 117
C.4.2 Structural View 120
C.4.3 Shared Data View 123
C.4.4 Collaborative Components View 125
C.4.5 Process View 135
C.4.6 Architectural Decisions Detail View 140
C.4.7 Variability View 144
C.5 Open Decisions 149
C.6 Change History 150

D Relatório da Avaliação do Método **151**
D.1 Contextualização e Motivação 151
D.2 Planejamento do Estudo de Caso 152
D.2.1 Modelo de Questionário para Avaliação da Representação da Arquitetura de Referência 155
D.2.2 Respostas Obtidas na Avaliação da Representação Arquitetural . . 163
D.3 Planejamento da Avaliação com Especialistas 169
D.3.1 Modelo de Questionário para Avaliação do ProSA-Re 169

Lista de Figuras

2.1 Relação entre modelo de referência, padrão arquitetural, arquitetura de referência e arquiteturas concretas 11
2.2 Estrutura geral do ProSA-RA 14
2.3 Arquitetura de referência para navegadores web 16
2.4 Arquitetura de referência orientada a aspectos para ambientes de engenharia de software 17
2.5 Arquitetura de referência para sistemas de contrato eletrônico 18
2.6 Modelo conceitual para a representação de uma arquitetura 20
2.7 Modelo de visões "4+1" 21
2.8 Tipos de diagramas da UML 26
2.9 Tipos de diagramas da SysML 27

3.1 Visão simplificada de componentes e interfaces da AUTOSAR 35
3.2 Abordagens utilizadas para representação de arquiteturas de sistemas embarcados 44
3.3 Linguagens de descrição arquitetural utilizadas na representação de arquiteturas de sistemas embarcados 47

4.1 Fluxo de trabalho do ProSA-Re 53
4.2 Participação do arquiteto funcional 58
4.3 Participação do arquiteto técnico 59
4.4 Participação do desenvolvedor 59
4.5 Fluxo de trabalho para a atividade de Definição do Escopo 60
4.6 Detalhes da atividade de Definição do Escopo 60
4.7 Fluxo de trabalho para a atividade de Elaboração das Visões 61
4.8 Detalhe da atividade de Elaboração das Visões 63
4.9 Fluxo de trabalho para a atividade de Representação das Decisões Arquiteturais 63
4.10 Detalhe da atividade de Representação das Decisões Arquiteturais 65
4.11 Fluxo de trabalho para a atividade de Avaliação da Representação 65
4.12 Detalhe da atividade de Avaliação da Representação 66
4.13 Cone Arquitetural para o Ciclo de Vida de Arquiteturas de Referência . . . 67
4.14 Corte horizontal do Cone Arquitetural 68

5.1 Avaliação do conjunto de visões arquiteturais da representação original . . 81
5.2 Avaliação de aspectos da representação original 82
5.3 Nível de conhecimento dos participantes do grupo A 82
5.4 Avaliação do conjunto de visões arquiteturais da representação elaborada com auxílio do ProSA-Re 84
5.5 Avaliação de aspectos da representação elaborada com auxílio do ProSA-Re 84
5.6 Nível de conhecimento dos participantes do grupo B 85
5.7 Nível de conhecimento dos especialistas consultados 87
5.8 Avaliação da descrição do método por especialistas 88

C.1 Conceptual view for the situated multiagent reference architecture 118
C.2 Structural view of the reference architecture for multiagent systems 121
C.3 Shared data view for the multiagent system 124
C.4 Collaborative components view for the multiagent system 126
C.5 Distribution of interface realizations in the situated multiagent system . . . 127
C.6 Collaborating components for the perception functionality 129
C.7 Collaborating components for the action functionality 130
C.8 Collaborating components for the communication functionality 134
C.9 Processes view of the situated multiagent system 137
C.10 Processes additional view of the situated multiagent system 139
C.11 Variability view of the situated multiagent system 145
C.12 Concepts variability view 147

Lista de Tabelas

2.1 *Framework* para classificação e comparação de ADLs 25

3.1 Fontes de pontos de vista para representação de arquiteturas de software e arquiteturas de referência 39
3.2 Pontos de vista para a representação de arquiteturas de software e arquiteturas de referência 40
3.3 Interesses e tipos de informação associados 45

4.1 Descrição dos pontos de vista 56
4.2 Técnicas para representação de pontos de vista usando a SysML 70
4.3 Modelo para descrição de decisões arquiteturais 72

5.1 Tabela comparativa de informações apresentadas em cada representação da arquitetura de referência 77
5.2 Projeto do estudo de caso 79
5.3 Sumário das respostas dos participantes do grupo A 80
5.4 Sumário das respostas dos participantes do grupo B 83

B.1 Description of SPEM elements 113

D.1 Respostas gerais obtidas no questionário 164
D.2 Respostas obtidas para percepção da representação original 168
D.3 Respostas obtidas para percepção da representação elaborada com auxílio do ProSA-Re 168

Lista de Abreviaturas e Siglas

AADL	*Architecture Analysis and Design Language*
ADL	*Architecture Description Language*
ASDL	*Architecture Structure Description Language*
ATAM	*Architecture Trade-off Analysis Method*
CCM	*CORBA Component Model*
CORBA	*Common Object Request Broker Architecture*
DRTSADL	*Distributed/Embedded Real-Time System ADL*
EAST-ADL	*Electronic Architecture and Software Tools ADL*
ECU	*Electronic Control Unit*
FSM	*Finite State Machines*
IEEE	*Institute of Electrical and Electronic Engineers*
MARTE	*UML Profile for Modeling and Analysis of Real-time and Embedded Systems*
MetaH	*Avionics Architecture Description Language*
NCES	*Net Condition/Event Systems*
OEM	*Original Equipment Manufacturer*
OMG	*Object Management Group*
PADL	*Process Algebraic Architectural Description Language*
SAAM	*Software Architecture Analysis Method*
SAE	*Society of Automotive Engineers*
SAG	*Software Architecture Graph*
SDL	*Specification and Description Language*
SysML	*Systems Modeling Language*
UML	*Unified Modeling Language*
UML-RT	*UML Profile for Real-time Systems*

Capítulo

1

Introdução

1.1 Contextualização

Nos últimos anos, é possível observar a crescente importância dada aos sistemas embarcados. Entende-se por sistema embarcado qualquer dispositivo que inclua um computador programável, mas que não seja um computador de propósito geral (Wolf, 2008). Nesse contexto, verifica-se que esses sistemas têm sido utilizados em uma variedade cada vez maior de equipamentos e desempenhado funções cada vez mais sofisticadas e possivelmente críticas para o funcionamento do sistema. Por exemplo, no domínio automotivo, os veículos *top* de linha do ano de 2006 já possuíam mais de 2000 funções individuais executadas ou controladas por software (Broy, 2006). Além disso, nesse mesmo período, verificou-se que o software já era responsável, sozinho, por cerca de 50% a 70% do custo total de desenvolvimento, sendo boa parte desse montante correspondente aos gastos com manutenção e garantia do software (Broy, 2006). É importante destacar ainda que esses veículos também possuem vários sistemas embarcados desempenhando funções críticas de segurança como, por exemplo, para assistência avançada ao condutor, ou controle do motor e estabilidade do veículo. Entende-se por um sistema embarcado crítico aqueles sistemas embarcados empregados na execução de tarefas muito complexas, como por exemplo, em veículos autônomos (por exemplo, tipos de robôs móveis) e em aeronaves não tripuladas para a realização de missões de monitoramento e controle. Em especial, são considerados sistemas embarcados críticos de segurança aqueles em que uma falha

pode resultar, por exemplo, no ferimento de pessoas, na perda significativa de recursos financeiros ou até mesmo em prejuízos para o meio ambiente (Knight, 2002). Observa-se então uma tendência de que, cada vez mais, os sistemas desenvolvidos para esse domínio sejam impulsionados pela demanda por funcionalidades inovadoras, mudanças rápidas nas plataformas e na infraestrutura dos sistemas, por altos níveis de qualidade e confiança e também por tempos para comercialização mais curtos (Broy, 2006). Desse modo, faz-se necessário adotar medidas para garantir a interoperabilidade desses sistemas e ao mesmo tempo diminuir a dependência de plataformas ou tecnologias específicas, promovendo assim o reúso do projeto arquitetural que pode ao final resultar na diminuição dos custos de desenvolvimento, integração e manutenção desses sistemas.

Em uma outra perspectiva, tem-se que a atenção dada às arquiteturas de software permite aumentar o nível de abstração em que os desenvolvedores podem pensar sobre um sistema ao disponibilizar um modelo que não apresenta os detalhes de implementação e aumenta a independência dos elementos que o compõe (Allen, 1997). Para isso, a arquitetura de um sistema de software apresenta os conceitos ou propriedades fundamentais de um sistema em seu ambiente[1] incorporado em seus elementos, nos relacionamentos que existem entre seus elementos e nos princípios de seu projeto e evolução (ISO, 2010). Desse modo, a existência de uma arquitetura permite ao arquiteto de software concentrar-se na análise e nas decisões que são mais cruciais para a estrutura do sistema e abstrair os demais detalhes de implementação (Allen, 1997).

Nesse contexto, também é possível encontrar a proposição de arquiteturas de referência, ou seja, uma arquitetura que engloba o conhecimento sobre como projetar arquiteturas de sistemas de software de um determinado domínio de aplicação; compreendendo as regras de negócio, os estilos arquiteturais (também definidos algumas vezes como padrões arquiteturais, que permitem inserir atributos de qualidade nas arquiteturas de referência), as melhores práticas de desenvolvimento de software (por exemplo, decisões arquiteturais, restrições do domínio, legislações e padrões) e os elementos de software que apoiam o desenvolvimento de sistemas de software desse domínio (Nakagawa et al., 2011a). Nesse sentido, observa-se que arquiteturas de referência são um importante mecanismo para promover o reúso do projeto arquitetural e o entendimento sobre uma área específica (Zhao e Kearney, 2003). Além disso, elas também contribuem para o aumento da efetividade de arquiteturas concretas de um determinado domínio ao fornecerem um marco inicial para a discussão de futuras mudanças e extensões, ajudando assim a evitar a reinvenção ou revalidação de soluções para problemas já resolvidos (Cloutier et al., 2010). É importante destacar, contudo, que por necessitar de um profundo conhecimento sobre o domínio para

[1]Entende-se por ambiente o contexto que determina a configuração e circunstâncias de todas as influências sobre um sistema como, por exemplo, políticas, economia, ecologia, tecnologias, dentre outras (ISO, 2010).

o qual a arquitetura de referência está sendo proposta, a criação de uma arquitetura de referência é muito mais dispendiosa do que a criação de uma arquitetura concreta, ou seja, de um dado sistema de software (Eickelmann e Richardson, 1996). Mesmo assim, em virtude da relevância de arquiteturas de referência para o desenvolvimento de sistemas de software de um determinado domínio, diversos exemplos podem ser encontrados na literatura como, por exemplo, para o domínio de teste de software (Eickelmann e Richardson, 1996; Nakagawa et al., 2007) e ferramentas de visualização (Nakagawa et al., 2009). Em especial, destaca-se a proposição de arquiteturas de referência para o domínio de sistemas embarcados, uma vez que o uso dessas arquiteturas atende às demandas atuais para o desenvolvimento mais eficiente desses sistemas. De fato, diversas arquiteturas de referência para esse domínio podem ser encontradas na literatura como, por exemplo, para sistemas automotivos (AUTOSAR, 2013; Eklund et al., 2005), sistemas marítimos (Borg, 2011), sistemas aviônicos (Batory et al., 1995) e sistemas para automação de lares (UniversAAL Project, 2011).

Em especial, a representação arquitetural (que neste trabalho será também utilizada como sinônimo do termo descrição arquitetural) de um sistema engloba todo o conjunto de documentos que auxiliam na compreensão da essência e de propriedades chave desse sistema relacionadas ao seu comportamento, composição e evolução, e que, por sua vez, também afetam interesses como viabilidade, utilidade e manutenibilidade desse sistema (ISO, 2010). Desse modo, além de ter um papel fundamental na comunicação de informações relevantes sobre o sistema, a descrição adequada da arquitetura também pode ser utilizada como base para análise e avaliação de implementações alternativas da arquitetura, como documentação e também como entrada para ferramentas de simulação, geração e análise automatizadas (ISO, 2010). Utilizada como documentação, a descrição da arquitetura tem papel fundamental na tomada de decisões e ainda transmite para os futuros mantenedores do sistema conhecimentos importantes sobre a solução original do arquiteto (Clements et al., 2011). Por essa razão, tornar a representação da arquitetura compreensível e acessível a todos os interessados nela é essencial. Nesse contexto, destaca-se o padrão ISO/IEC/IEEE 42010 (ISO, 2010), anteriormente IEEE 1471-2000, que estabelece as melhores práticas para a descrição arquitetural de sistemas de software e padroniza importantes termos e definições relacionados com a descrição de arquiteturas de software.

Vale destacar ainda que este trabalho de dissertação faz parte de projetos de pesquisa maiores, sendo um deles o projeto intitulado "Uma Contribuição ao Estabelecimento e Avaliação de Arquiteturas de Referência para Sistemas Embarcados" (Edital MCT/CNPq 14/2011 - Universal, Processo: 474720/2011-0), financiado pelo CNPq e sob a coordenação da Profa. Dra. Elisa Yumi Nakagawa. Esse projeto de pesquisa tem como objetivo

principal propor um processo para o estabelecimento, representação e avaliação de arquiteturas de referência de sistemas embarcados. Este trabalho também está inserido no contexto do projeto intitulado "ProSA-RAES: Subsídios para o Estabelecimento de Arquiteturas de Referência para Sistemas Embarcados" (Processo N. 2011/23316-8, Vigência: maio/2012 a abril/2014), financiado pela FAPESP e também sob a coordenação da Profa. Dra. Elisa Yumi Nakagawa. Esse projeto tem como objetivo principal o estabelecimento de um processo para a construção, representação e avaliação de arquiteturas de referência para o domínio de sistemas embarcados e aplicação desse processo para a construção de arquiteturas de referência desse domínio. Por fim, este trabalho também está inserido no contexto do INCT-SEC (Instituto Nacional de Ciência e Tecnologia em Sistemas Embarcados Críticos[2]), FAPESP Processo N. 08/57870-9 e CNPQ Processo N. 573963/2008-8, que faz parte de um programa conduzido pelo Ministério da Ciência e Tecnologia (MCT) para a criação de diversos INCTs. Desse modo, o INCT-SEC tem a missão de agregar habilidades, competências e infraestrutura necessárias para o desenvolvimento de sistemas embarcados críticos para áreas estratégicas do país, tais como o controle do meio ambiente, segurança, defesa nacional e agricultura. É também de interesse do INCT-SEC a colaboração da área de Engenharia de Software para a sistematização no desenvolvimento de sistemas embarcados críticos.

1.2 Motivação

Nesse contexto, descrever a arquitetura de referência de modo que todos os interessados nela (como, por exemplo, clientes, programadores e arquitetos de software) possam entendê-la é imprescindível, especialmente porque elas serão a base por meio da qual todo um conjunto de sistemas de software será desenvolvido. No entanto, verifica-se que, na prática, a descrição de arquiteturas de referência ainda é, muitas vezes, realizada informalmente, de modo que cada autor utiliza um meio diferente para representá-las (Oliveira et al., 2010). Em especial, devido ao nível de abstração mais elevado dessas arquiteturas, nem todas as abordagens existentes para descrição de arquiteturas de software são adequadas para a descrição de arquiteturas de referência (Nakagawa e Maldonado, 2008).

Nesse contexto, observa-se que há uma carência de trabalhos que tenham investigado quais abordagens e técnicas são mais adequadas para a descrição de arquiteturas de referência, em especial, para o domínio de sistemas embarcados. No entanto, considerando que o projeto desses sistemas normalmente impõe características específicas do domínio como, por exemplo, restrições rigorosas e requisitos não funcionais complexos, também faz-se necessário investigar como realizar uma descrição adequada dessas arquiteturas.

[2]http://www.inct-sec.org/

Essa investigação poderá contribuir para a área de Sistemas Embarcados propondo meios que promovam a divulgação do conhecimento contido nessas arquiteturas para todas as pessoas e/ou organizações que tenham interesse em utilizá-las.

1.3 Objetivos

O objetivo deste trabalho consiste na proposta de um método para a descrição de arquiteturas de referência do domínio de sistemas embarcados. Para isso, primeiramente, é definido o escopo dessas arquiteturas por meio da identificação das informações, restrições e características específicas de sistemas desse domínio. Em seguida, as principais características e informações que precisam ser consideradas na descrição arquitetural são investigadas para garantir que todos os interesses da arquitetura como, por exemplo, viabilidade, usabilidade, confiabilidade e funcionalidades foram atendidos e podem ser acessados pelos respectivos interessados. Com base nessas informações, é então proposto um método para representar arquiteturas de referência do domínio de sistemas embarcados e um conjunto de diretrizes é elaborado para orientar a utilização correta desse método indicando, por exemplo, em qual contexto o método pode ser adotado e quais interesses podem ser capturados pela descrição arquitetural. Para a avaliação do método, será conduzido um estudo de caso para identificar benefícios e limitações da representação arquitetural construída. Uma análise qualitativa também será conduzida com especialistas da área de Arquitetura de Software para avaliar a viabilidade do método proposto. Espera-se que os resultados deste projeto contribuam para o reúso do conhecimento contido nas arquiteturas de referência propostas para o domínio de sistemas embarcados.

1.4 Organização

Neste capítulo foi apresentado o contexto no qual este trabalho de mestrado está inserido, a motivação para desenvolver a pesquisa proposta e os objetivos do trabalho. O restante desta dissertação está organizado da seguinte forma. No Capítulo 2 são apresentadas as terminologias, os conceitos e as principais teorias relacionados às arquiteturas de software, arquiteturas de referência e descrição dessas arquiteturas. No Capítulo 3 são apresentados conceitos e terminologias básicos da área de Sistemas Embarcados e exemplos de arquiteturas de referência propostas para esse domínio são apresentados e analisados. No Capítulo 4 é apresentado o método proposto, em que as atividades, os produtos de trabalho utilizados e os papéis envolvidos são detalhados. Na sequência, no Capítulo 5 é apresentado o estudo de caso realizado para avaliar a representação arquitetural construída com o apoio do método e também a avaliação conduzida com especialistas

para avaliar a descrição e a viabilidade do método proposto. Por fim, no Capítulo 6 são apresentadas as conclusões deste projeto de mestrado.

Capítulo

2

Arquitetura de Software e Arquitetura de Referência

2.1 Considerações Iniciais

A arquitetura de um sistema de software é composta pelos elementos, pelos relacionamentos existentes entre esses elementos e por aspectos como configuração, restrições, propriedades e requisitos desse sistema (Clements et al., 2011). É possível observar também a proposição de arquiteturas de referência com o objetivo de facilitar e sistematizar o desenvolvimento de sistemas de software para um determinado domínio. Em especial, para permitir que as arquiteturas de software e as arquiteturas referência sejam de fato empregadas, é essencial que elas sejam representadas adequadamente. Nesse contexto, o objetivo desse capítulo é introduzir a teoria básica envolvendo arquiteturas de software e arquiteturas de referência.

Este capítulo está organizado da seguinte forma. Na Seção 2.2 são apresentados conceitos e terminologias essenciais sobre arquiteturas de software. Na Seção 2.3 são apresentados conceitos e teorias essenciais para a compreensão e estabelecimento de arquiteturas de referência. Em seguida, na Seção 2.4, são apresentados detalhes sobre a descrição de arquiteturas de software e arquiteturas de referência. Por fim, na Seção 2.5 são discutidas as considerações finais desse capítulo.

2.2 Arquiteturas de Software

Inicialmente, a palavra arquitetura fazia referência apenas à Arquitetura de Sistemas, designando a estrutura física de computadores. Esse cenário começou a mudar a partir da década de 1990 com o advento da arquitetura de software como uma disciplina separada em virtude do aumento no tamanho e, consequentemente, da complexidade dos sistemas de software (Kruchten et al., 2006). Observou-se a partir de então uma maior preocupação por parte dos engenheiros de software em utilizar princípios de projeto, tais como modularização e ocultamento de informações, com o propósito de obter sistemas de software que apresentassem maior qualidade e custos mais baixos (Kruchten et al., 2006). Nesse contexto, é comum a utilização da arquitetura de software como um dos principais fatores para a determinação do desempenho e da compatibilidade entre vários sistemas de software, uma vez que ela é capaz de preservar a intenção do projetista sobre a estrutura e o comportamento do programa e, assim, fornecer também uma importante proteção contra o envelhecimento do software (Kruchten et al., 2006). Por essa razão, a arquitetura de software também possui papel fundamental na determinação da qualidade e na manutenibilidade do sistema (Wasserman, 1996).

Segundo Babar et al. (2009), as principais contribuições obtidas pela construção de uma arquitetura de software podem ser resumidas em três: (i) servir como um veículo de comunicação entre os interessados, durante a análise e desenvolvimento do sistema; (ii) capturar importantes decisões de projeto que impactam as fases de desenvolvimento subsequentes; e (iii) servir como uma abstração do sistema, a qual pode ser então lida e inspecionada por pessoas e consistirá na base para apoiar o reúso.

Atualmente, a disciplina de arquitetura de software tem sido muito investigada, de modo que existem vários esforços por parte da academia e também da indústria para divulgar e aprofundar os conhecimentos existentes na área. Nesse sentido, vários eventos específicos como, por exemplo, *European Conference on Software Architecture* (ECSA), *Working IEEE/IFIP Conference on Software Architecture*(WICSA) e *ACM Sigsoft International Conference on the Quality of Software Architectures* (QoSA) são referência para os pesquisadores da área.

2.2.1 Conceitos e Terminologia Básicos

O amadurecimento alcançado pela disciplina na última década contribuiu para a definição de vários termos e conceitos que facilitam a comunicação entre pesquisadores e profissionais da área. O termo *arquitetura concreta*, por exemplo, é utilizado para designar a arquitetura de software de um determinado sistema de software. Considerando-se um conjunto de arquiteturas concretas, é possível identificar um conjunto de restrições

comuns sobre a forma e a estrutura dessas arquiteturas o qual é denominado *estilo arquitetural* (Garlan e Perry, 1995). O uso de um determinado estilo no projeto da arquitetura de um sistema de software tem como finalidade facilitar a comunicação entre as pessoas envolvidas no projeto. O estilo em camadas, *pipes&filters* e orientado a serviços[1] são exemplos de estilos arquiteturais que podem ser adotados.

Entende-se por *padrão arquitetural* um conjunto pré-definido de componentes e das responsabilidades e relacionamentos existentes entre eles (Buschmann et al., 1996). O uso de padrões arquiteturais contribui para promover boas práticas no projeto de sistemas, já que apresentam soluções comprovadas para problemas recorrentes (Buschmann et al., 1996). Por exemplo, o padrão arquitetural *Model-View-Controller* (MVC) é bastante utilizado na construção de sistemas *web*. Segundo esse padrão, a arquitetura é dividida em três componentes: (a) *model*, que contém as funcionalidades básicas e os dados; (b) *view*, que exibe as informações para o usuário; e (c) *controller*, que trata as entradas do usuário (Buschmann et al., 1996). Além disso, a existência de um mecanismo de atualização garante a consistência entre esses componentes (Buschmann et al., 1996). É relevante destacar ainda que padrões arquiteturais são importantes para determinados atributos de qualidade da arquitetura, uma vez que podem solucionar questões relacionadas ao desempenho, segurança ou disponibilidade (Bass et al., 2003).

A construção de uma arquitetura de software para um determinado sistema de software envolve conhecer quem são as pessoas e/ou organizações que possuem interesse na construção desse sistema. Os *interessados* no sistema (referenciados na literatura em inglês por *stakeholders*) podem ser desde clientes e usuários até gerentes de projeto e programadores. São considerados *interesses* (do inglês, *concerns*) desde as propriedades ou os requisitos sobre decisões de projeto ou implementação do sistema até considerações sobre a sua operação (ISO, 2010). Segurança, funcionalidade, reusabilidade, performance, custo, portabilidade, usabilidade, confiabilidade, autonomia, comportamento e modularidade são apenas alguns exemplos de interesses que podem existir. Nesse sentido, a construção de uma arquitetura de software envolve a negociação e balanceamento entre os diferentes requisitos funcionais e não funcionais existentes com possíveis soluções que atendam esses interesses (Babar et al., 2009).

Em uma outra perspectiva, tem-se que um *framework* arquitetural estabelece uma prática comum para a criação, interpretação, análise e uso de descrições arquiteturais dentro de um domínio específico de aplicação ou comunidade de interessados, podendo ser utilizado na criação da descrição arquitetural, no desenvolvimento de ferramentas para modelagem da arquitetura, no estabelecimento de processos para facilitar a comunicação

[1]Entende-se por serviço um módulo auto-contido bem definido que fornece funcionalidades de negócio e é independente do estado ou contexto de outros serviços (Papazoglou e Heuvel, 2007)

e a interoperabilidade entre múltiplos projetos e/ou organizações, dentre outros (ISO, 2010). Como exemplo de *framework* arquitetural, pode-se citar o modelo "4+1" proposto por Kruchten (1995) para a representação de arquiteturas de software que será explicado em maiores detalhes na Seção 2.4.

2.3 Arquiteturas de Referência

O modo como os sistemas de software são desenvolvidos mudou muito nos últimos anos. Assim, observa-se cada vez mais que o desenvolvimento dos sistemas atuais além de envolverem vários fornecedores e organizações, também têm suas equipes de desenvolvimento espalhadas em diferentes localidades. Nesse contexto, verifica-se a proposição de arquiteturas de referência principalmente naquelas organizações em que a multiplicidade de fatores atingiu um patamar crítico, fazendo-se necessário adotar novas medidas para facilitar a criação e manutenção desses sistemas (Muller, 2008). Nesse sentido, uma arquitetura de referência pode ser útil para fornecer por exemplo(Muller, 2008): (i) um léxico e taxonomias comuns, que facilitam a comunicação entre os interessados; (ii) uma visão arquitetural comum, que alinha os esforços das várias pessoas e equipes envolvidas; (iii) modularização, que auxilia na divisão de esforços; e (iv) contextos complementares, que garantem a integração desses esforços posteriormente. Considerando-se que as arquiteturas de referência servirão como a base por meio da qual um conjunto de arquiteturas de sistemas de software de um determinado domínio será construído, vale destacar também que o uso de arquiteturas de referência também evita a reinvenção e revalidação de soluções para problemas já resolvidos (Cloutier et al., 2010). Em especial, utiliza-se o termo *instância arquitetural* quando deseja-se destacar que uma arquitetura concreta é derivada de uma arquitetura de referência.

Recentemente, a arquitetura de referência foi definida como uma arquitetura que engloba o conhecimento sobre como projetar arquiteturas concretas para um determinado domínio de aplicação; dessa forma, ela precisa incluir as regras de negócio, os estilos e padrões arquiteturais que abordam atributos de qualidade da arquitetura de referência, as melhores práticas para o desenvolvimento de software (por exemplo, decisões arquiteturais, restrições do domínio, legislações e padrões) além dos elementos de software que apoiam o desenvolvimento de sistemas de software para esse domínio (Nakagawa et al., 2011a).De fato, é possível encontrar arquiteturas de referência para os mais diversos domínios de aplicação como, por exemplo, para o domínio de teste de software (Eickelmann e Richardson, 1996; Nakagawa et al., 2007), sistemas marítimos (Borg, 2011) e automação de lares (UniversAAL Project, 2011). Embora seja, muitas vezes, utilizado como sinônimo de arquitetura de referência, o termo *modelo de referência* designa uma estrutura

que promove o entendimento sobre um determinado domínio ao fornecer um vocabulário comum e apresentar as partes e seus relacionamentos sem, no entanto, considerar detalhes da implementação (Bass et al., 2003). Desse modo, um modelo de referência é uma decomposição padrão para um problema conhecido em partes que cooperativamente solucionam esse problema (Bass et al., 2003). Como exemplo, pode-se citar o modelo de referência OASIS para arquiteturas orientadas a serviços que é utilizado para compreender a essência dessas arquiteturas sem, no entanto, especificar padrões, tecnologias ou outros detalhes mais concretos (OASIS Standard, 2006). Nesse sentido, é possível observar que modelos de referência possuem um nível de abstração ainda mais elevado que arquiteturas de referência, como ilustrado na Figura 2.1. A partir de um modelo de referência e de padrões arquiteturais, arquiteturas de referência podem ser derivadas e essas arquiteturas, quando instanciadas, dão origem a arquiteturas de software concretas.

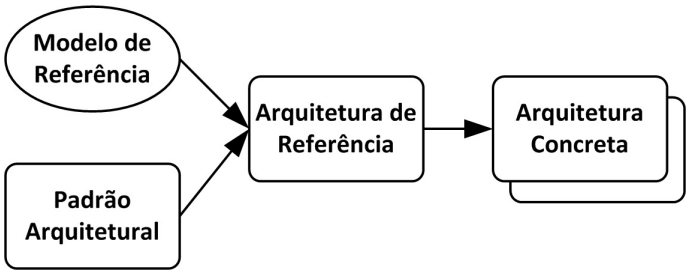

Figura 2.1: Relação entre modelo de referência, padrão arquitetural, arquitetura de referência e arquiteturas concretas (Adaptado de (Bass et al., 2003))

2.3.1 Tipos de Arquiteturas de Referência

Em vista da diversidade de arquiteturas de referência, Angelov et al. (2009) propõem um *framework* para a classificação dessas arquiteturas no qual avaliam-se o seu contexto, objetivos e detalhes de projeto. Para determinar o contexto de uma arquitetura de referência, investigam-se os aspectos de projeto e aplicação que podem afetar seus objetivos de negócio e especificação da arquitetura, sendo esses (Angelov et al., 2009): (i) para quantas organizações a arquitetura de referência interessa, ou seja, se para uma única organização ou múltiplas organizações; (ii) quais são as principais categorias de interessados envolvidos no projeto dessa arquitetura, ou seja, organizações de software, de usuários, centros de pesquisa ou uma organização padronizadora; e (iii) em qual momento ela é construída, ou seja, se antes (denominada preliminar) ou depois (denominada clássica) da existência de arquiteturas concretas do domínio. Para determinar os objetivos de uma arquitetura de referência, avalia-se por qual razão ela é proposta, ou seja, se para padronizar arquiteturas concretas do domínio (visando a interoperabilidade) ou se para servir como um mecanismo facilitador no projeto de arquiteturas concretas desse domínio (visando o fornecimento de

diretrizes para o projeto de novos sistemas na forma de padrões, esquemas, dentre outros). Por fim, para determinar o projeto de uma arquitetura de referência, avalia-se (Angelov et al., 2009): (i) o conteúdo dessas arquiteturas, ou seja, se elas descrevem componentes, interfaces, protocolos, algoritmos e/ou políticas e diretrizes; (ii) o nível de detalhamento com o qual esses elementos são definidos (detalhado, semi-detalhado ou superficial); (iii) o nível de abstração adotado (concreta, semi-concreta ou abstrata); e (iv) o nível de formalização da arquitetura de referência, ou seja, os tipos de técnicas empregadas em sua representação (formais[2], semi-formais[3] ou informais[4]). Baseadas nas combinações para os possíveis valores de cada critério, cinco tipos de arquiteturas de referência foram delineados:

Tipo 1: São arquiteturas de referência *clássicas* que visam a *padronização* e são projetadas para serem implementadas em *múltiplas organizações*. Organizações representativas de usuários e de software devem participar da definição da arquitetura, uma vez que a padronização requer a existência de um consenso entre eles para que a arquitetura seja bem sucedida. Destaca-se também a participação de uma organização padronizadora para facilitar a obtenção desse consenso e o estabelecimento da arquitetura como um padrão. Arquiteturas desse tipo descrevem os componentes superficialmente em alto nível e as interfaces de modo mais detalhado, já que a interoperabilidade é um dos principais objetivos da arquitetura. É importante ressaltar, contudo, que outros elementos também podem estar presentes, mas o excesso de informações pode resultar na diminuição da eficiência de arquiteturas desse tipo. Essa é também uma arquitetura *abstrata* de modo a oferecer a cada organização liberdade para definir os próprios detalhes de implementação. Por fim, elas utilizam-se, no geral, de técnicas *semi-formais* para representação de modo a possibilitar uma especificação clara, ao mesmo tempo em que interessados com pouca experiência em técnicas formais também possam compreendê-las;

Tipo 2: São arquiteturas de referência *clássicas* que visam a *padronização* e são projetadas para serem implementadas em uma *única organização*. Os interessados em arquiteturas desse tipo englobam organizações de potenciais usuários, projetistas e gerentes, que aqui exercem o papel da organização padronizadora. Arquiteturas desse tipo representam componentes e interfaces utilizando técnicas *semi-formais*. Políticas e diretrizes são descritas concretamente. Todos os elementos da arquitetura podem ser descritos em qualquer nível de detalhamento dependendo do contexto da organização. Arquiteturas de

[2]Entende-se por técnicas formais aquelas baseadas em sintaxes e semânticas formalmente definidas e que permitem a descrição não ambígua da arquitetura. Essas técnicas são mais fáceis de serem compreendidas por ferramentas.

[3]Entende-se por técnicas semi-formais aquelas que combinam, em parte, o rigor das linguagens formais à compreensibilidade da linguagem natural.

[4]Entende-se por técnicas informais aquelas mais próximas da linguagem natural. Essas técnicas são mais fáceis de serem compreendidas por seres humanos.

referência desse tipo podem ser *concretas* ou *semi-concretas* devido as escolhas que podem ser feitas em termos de tecnologias, aplicações e padrões com o objetivo de alcançar a padronização desejada dentro da organização;

Tipo 3: São arquiteturas de referência *clássicas* para serem utilizadas como mecanismos *facilitadores* em *múltiplas organizações*. São projetadas por organizações de software em cooperação com organizações de usuários. Arquiteturas desse tipo são construídas para divulgar um produto de software da organização e descrevem seus principais componentes e interfaces, além de fornecerem diretrizes para sua implementação. Desse modo, essas são arquiteturas de referência concretas, ou seja, promovem a tecnologia da organização, com componentes e interfaces definidos superficialmente e diretrizes detalhadas. Podem utilizar de técnicas *semi-formais* para sua representação;

Tipo 4: São arquiteturas de referência *clássicas* para serem utilizadas como mecanismos *facilitadores* em uma *única organização*. São similares às arquiteturas do Tipo 2, mas servem apenas para guiar o projeto e implementação de sistemas dentro de uma organização. Por esse motivo, possuem os mesmos interessados das arquiteturas do Tipo 2. Normalmente, utilizam técnicas *semi-formais* ou *informais* para representação. O projeto de componentes semi-detalhados ou agregados é suficiente para atingir seus propósitos e ela pode ser tanto independente de tecnologia (abstrata) como indicar escolhas tecnológicas preferenciais (semi-concreta ou concreta); e

Tipo 5. São arquiteturas de referência *preliminares* para serem utilizadas como mecanismos *facilitadores* em *múltiplas organizações*. Elas são projetadas para facilitar o projeto de arquiteturas de sistemas que serão necessários no futuro. Por esse razão, elas são normalmente desenvolvidas em centros de pesquisa e envolvem organizações de software e usuários na sua construção. Elas definem detalhadamente os componentes, algoritmos (para apoiar a operação dos componentes) e protocolos (para demonstrar a interação entre componentes) para que se possa demonstrar os aspectos inovadores definidos. Arquiteturas de referência desse tipo são, muitas vezes, abstratas, uma vez que a tecnologia necessária para sua implementação pode ainda não estar disponível. Além disso, porque é preciso convencer seus interessados de suas qualidades para promover a sua adoção no projeto das primeiras arquiteturas concretas, verifica-se o emprego de técnicas *formais* ou *semi-formais* para representá-las.

É importante ressaltar que a aderência de uma arquitetura de referência a um dos tipos definidos no *framework* é uma pré-condição para o seu uso efetivo no projeto de arquiteturas concretas. Além disso, esse *framework* pode servir como um importante suporte para indicar possíveis desvios entre objetivos, contexto e projeto, antes e após a definição de arquiteturas de referência.

2.3.2 ProSA-RA

A investigação dos meios pelos quais as arquiteturas de referência são construídas é importante inclusive para melhorar o entendimento que se tem sobre elas. Pode-se encontrar diferentes trabalhos na literatura relacionados ao projeto de arquiteturas de referência. Por exemplo, Muller (2008) propõe algumas recomendações para criar e manter arquiteturas de referência nas quais se tem, basicamente, que essas arquiteturas devem ser de fácil entendimento e manutenção, além de que devem estar sempre atualizadas. Em outro trabalho, Eklund et al. (2005) aponta a necessidade em se formalizar o processo por meio do qual a arquitetura de referência é projetada, uma vez que processos informais ainda têm sido muito utilizados. Nesse contexto, destaca-se a proposição do ProSA-RA (Nakagawa et al., 2011b, 2009), um processo que visa a construção, representação e avaliação de arquiteturas de referência orientadas a aspectos. Contudo, o processo também tem sido utilizado na construção de arquiteturas de referência de outros domínios como, por exemplo, para ferramentas de visualização (Nakagawa et al., 2009) e sistemas marítimos (Borg, 2011). Os passos para aplicação do ProSA-RA são ilustrados na Figura 2.2 e explicados a seguir.

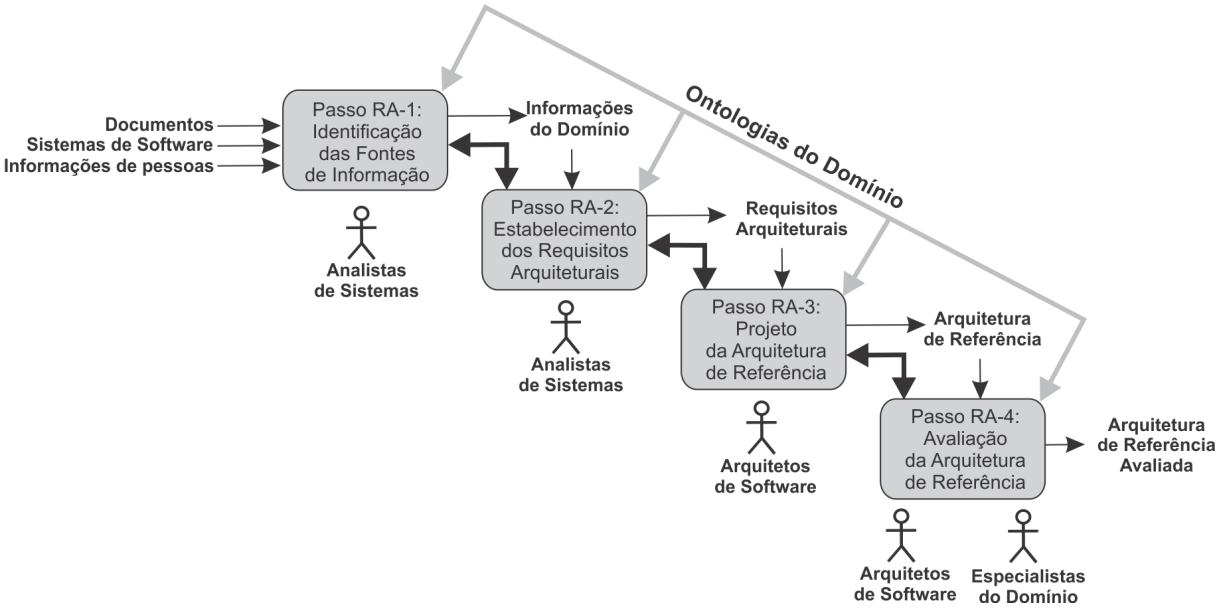

Figura 2.2: Estrutura geral do ProSA-RA (Nakagawa et al., 2009)

Passo RA-1 - Identificação das Fontes de Informação: refere-se à investigação e seleção de fontes de informação com o propósito de levantar informações sobre o domínio para o qual a arquitetura de referência está sendo criada;

Passo RA-2 - Estabelecimento dos Requisitos Arquiteturais: refere-se à identificação dos requisitos da arquitetura de referência com base nas fontes de informações. Para isso, primeiramente identificam-se os requisitos de sistemas do domínio e, em seguida,

estabelecem-se quais são os requisitos específicos da arquitetura de referência, isso é, os requisitos arquiteturais. A seguir, os requisitos arquiteturais são mapeados para os conceitos do domínio alvo. Nesse ponto, identificam-se aqueles conceitos que possuem característica transversal, ou seja, interesses transversais. É denominado interesse transversal o interesse cuja implementação encontra-se espalhada ou entrelaçada com a implementação de outros interesses (Clarke e Baniassad, 2005) como, por exemplo, a persistência. Dessa forma, identificam-se também os aspectos arquiteturais, ou seja, os módulos em nível de arquitetura que agregam interesses transversais;

Passo RA-3 - Projeto da Arquitetura de Referência: a partir dos requisitos arquiteturais e dos conceitos identificados no passo anterior, é estabelecida uma descrição arquitetural da arquitetura de referência, considerando-se também padrões e estilos arquiteturais. A seguir, constrói-se a descrição arquitetural utilizando-se visões arquiteturais. Particularmente, as visões de módulos, de tempo de execução, de implantação e conceitual têm sido propostas para descrever essas arquiteturas (Nakagawa e Maldonado, 2008); contudo, outras visões podem ser relevantes considerando-se diferentes interessados ou o uso e a natureza da arquitetura de referência em questão; e

Passo RA-4 - Avaliação da Arquitetura de Referência: a ideia nesse passo é a utilização de um *checklist* para a avaliação da arquitetura. O *checklist* define uma lista de questões que guiam um revisor para a detecção de defeitos em documentos relacionados às arquiteturas de referência. Usando esse *checklist*, características de qualidade (manutenibilidade, desempenho, segurança, usabilidade, portabilidade e reúso), além da própria descrição da arquitetura são avaliados. Além disso, o *checklist* visa à identificação e eliminação de defeitos relacionados à omissão, ambiguidade, inconsistência na informação, bem como informação estranha e fato incorreto presentes na arquitetura sendo avaliada.

2.3.3 Exemplos de Arquiteturas de Referência

Em virtude da relevância de arquiteturas de referência para o desenvolvimento de sistemas de software de um determinado domínio de aplicação, arquiteturas de referência para os mais diversos domínios podem ser encontradas na literatura. Nesta seção, algumas dessas arquiteturas são brevemente descritas.Foram selecionadas arquiteturas de referência para navegadores *web* (Grosskurth e Godfrey, 2005), ambientes de teste de software (Eickelmann e Richardson, 1996), ambientes de engenharia de software (Nakagawa et al., 2011c) e sistemas de contrato eletrônico (Angelov et al., 2008). Em particular, arquiteturas de referência específicas para o domínio de sistemas embarcados são descritas em maiores detalhes no Capítulo 3.

Considerando-se a crescente popularidade e a evolução de navegadores *web* nos últimos anos como, por exemplo, Internet Explorer da Microsoft[5], Firefox da Mozilla[6] e Google Chrome do Google[7], observa-se que arquiteturas de referência para esse domínio podem ser um importante mecanismo para ajudar os desenvolvedores a compreenderem a relação entre os diferentes requisitos desses sistemas durante o projeto de arquiteturas concretas além de também ajudar os mantenedores a compreenderem melhor o código legado de sistemas antigos (Grosskurth e Godfrey, 2005). Nesse sentido, a arquitetura de referência para navegadores *web* proposta por Grosskurth e Godfrey (2005) e ilustrada na Figura 2.3, é um exemplo de arquitetura de referência clássica derivada de arquiteturas concretas para esse domínio. A arquitetura proposta assemelha-se às arquiteturas de referência do Tipo 4, isso é, arquiteturas de referência clássicas e abstratas para serem utilizadas como mecanismo facilitador em múltiplas organizações e que foram desenvolvidas em um único centro de pesquisa. Técnicas informais foram utilizadas para a representação dessa arquitetura, sendo que apenas os principais componentes e conexões existentes entre eles foram apresentados.

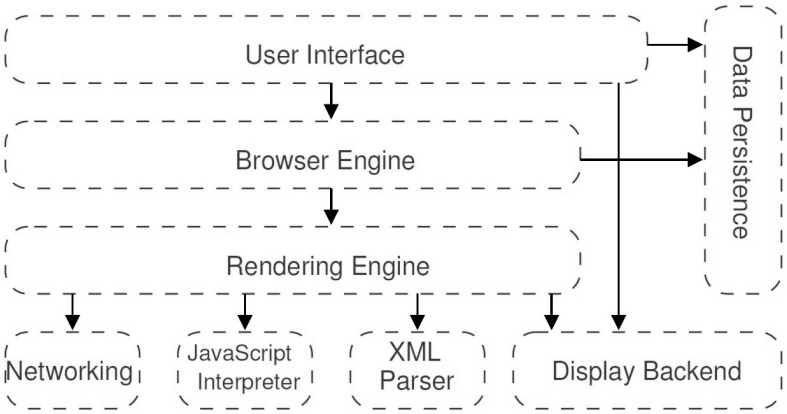

Figura 2.3: Arquitetura de referência para navegadores web (Grosskurth e Godfrey, 2005)

Seguindo um domínio de aplicação mais abrangente, tem-se a proposição da arquitetura de referência RefASSET, uma arquitetura de referência orientada a aspectos para ambientes de engenharia de software (Nakagawa et al., 2011c). Entende-se por um ambiente de engenharia de software um conjunto integrado de ferramentas que cobrem todo o ciclo de vida do software ou pelo menos uma parte dele (Harrison et al., 2000). Em especial, essa arquitetura de referência utiliza conceitos da orientação a aspectos para aprimorar a separação de interesses, resultando em ambientes de engenharia de software reutilizáveis (Nakagawa et al., 2011c). A arquitetura de referência ilustrada na Figura

[5] `http://windows.microsoft.com/en-us/internet-explorer/products/ie/home`
[6] `http://br.mozdev.org`
[7] `http://www.google.co.jp/chrome`

2.4 também pode ser classificada como uma arquitetura de referência do Tipo 4, isso é, uma arquitetura de referência clássica, para ser utilizada como mecanismo facilitador no projeto de novos sistemas para esse domínio, e que identifica os principais módulos e relacionamentos entre esses módulos, além de destacar as principais funcionalidades e serviços que podem ser disponibilizados. Essa arquitetura também foi desenvolvida por um único centro de pesquisa para ser adotada em múltiplas organizações e não define as tecnologias que devem ser adotadas na sua implementação. É importante destacar ainda que a RefASSET é descrita semi-formalmente, utilizando-se a UML.

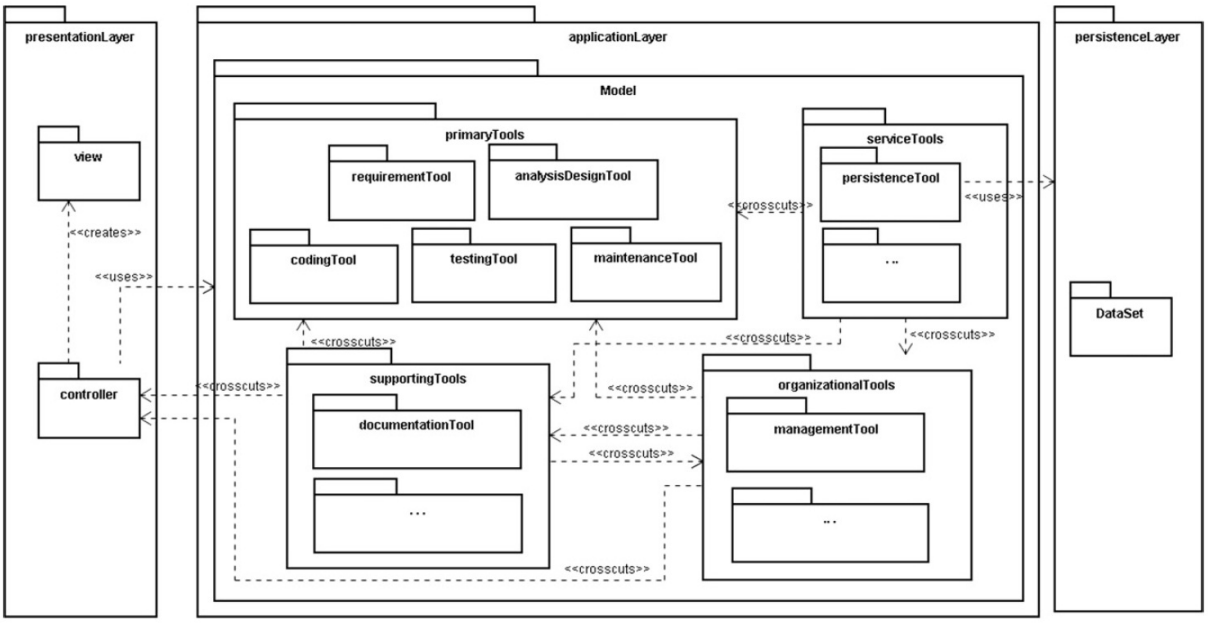

Figura 2.4: Arquitetura de referência orientada a aspectos para ambientes de engenharia de software(Nakagawa et al., 2011c)

Por fim, pode-se destacar também a ERA (do inglês, *E-contracting Reference Architecture*), uma arquitetura de referência para sistemas de contrato eletrônico proposta por Angelov et al. (2008) e ilustrada na Figura 2.5. Um contrato especifica como será dada a troca de serviços, bens e valores bem como os direitos e obrigações de cada uma das partes envolvidas no contrato (Angelov et al., 2008). Nesse sentido, a automatização desse processo possui como finalidade apoiar o estabelecimento, aprovação e gerenciamento de contratos eletrônicos e, com isso, melhorar a eficiência e eficácia do processo que exigia até então a participação intensiva de pessoas (Angelov et al., 2008). A arquitetura de referência proposta servirá como a base por meio da qual arquiteturas concretas para esses sistemas serão analisadas e também como modelo de padronização, facilitando assim a integração desses sistemas e a comunicação de ideias (Angelov et al., 2008). Pode-se então classificar a arquitetura de referência assemelhe-se com uma arquitetura do Tipo 5, isso é, uma arquitetura de referência preliminar para ser utilizada como mecanismo

facilitador em múltiplas organizações. O fato de ser uma arquitetura preliminar faz com ela foque, principalmente, nas características inovadoras propostas pela arquitetura de referência e, portanto, possa não ser considerada na prática para a implementações de sistemas (Angelov et al., 2009). A maior contribuição dessa arquitetura é justamente servir de inspiração para futuras pesquisas a serem realizadas no domínio, identificando limitações que precisam ser resolvidas (Angelov et al., 2009).

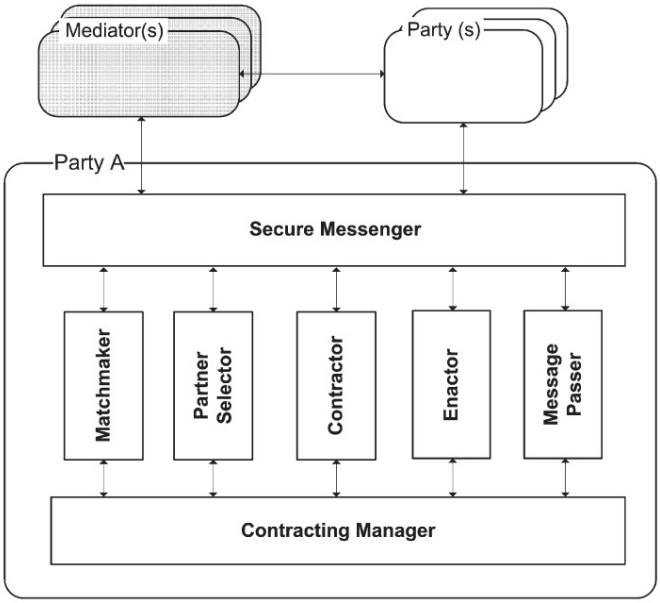

Figura 2.5: Arquitetura de referência para sistemas de contrato eletrônico (Angelov et al., 2008)

2.4 Representação Arquitetural

Enquanto a arquitetura de um sistema de software é abstrata, a representação dessa arquitetura é um artefato tangível que, além de auxiliar na comunicação de informações relevantes sobre um sistema, também pode ser utilizada como documentação, como base para a avaliação de implementações alternativas da arquitetura e ainda como entrada para ferramentas de simulação, geração e análise automatizada do sistema (ISO, 2010). Como documentação, a representação deve capturar informações suficientes para permitir que o projeto da arquitetura de referência seja avaliado com relação aos requisitos arquiteturais necessários. Por esse motivo, é essencial que arquiteturas de software e arquiteturas de referência sejam adequadamente representadas.

Nesse contexto, o padrão ISO/IEC/IEEE 42010 estabelece uma definição comum para os principais elementos que compõem uma descrição arquitetural, sendo esses: visão arquitetural (do inglês, *architectural view*), ponto de vista arquitetural (do inglês, *architectural*

viewpoint), modelo arquitetural (do inglês, *architecture model*), tipo de modelo (do inglês, *model kind*), correspondência (do inglês, *correspondence*), regras de correspondência (do inglês, *correspondence rules*) e decisões arquiteturais (do inglês, *architecture rationale*). Entende-se por uma *visão arquitetural* o artefato que descreve a arquitetura concreta a partir de um determinado ponto de vista do sistema (ISO, 2010). Um *ponto de vista* é um artefato que estabelece as convenções — linguagens, notações, regras de projeto e/ou métodos de modelagem, técnicas de análise, dentre outras — para a construção, interpretação e uso de visões arquiteturais para capturar um conjunto de interesses específicos do sistema (ISO, 2010). Em outras palavras, um ponto de vista expressa um modo de olhar para a arquitetura, enquanto a visão arquitetural é o resultado da aplicação desse ponto de vista em uma arquitetura concreta (ISO, 2010). Uma visão arquitetural é composta por um ou mais *modelos arquiteturais*, os quais utilizam as convenções de modelagem, definidas pelo tipo de modelo, que são apropriadas para os interesses do sistema. O modelo arquitetural pode, por exemplo, ser utilizado para apresentar seletivamente porções do sistema com o objetivo de destacar pontos de interesse. É importante notar que, dentro de uma única descrição arquitetural, o mesmo modelo arquitetural pode fazer parte de uma ou mais visões (ISO, 2010). Uma *correspondência* define uma relação entre elementos da descrição arquitetural como, por exemplo, consistência, rastreabilidade, dependências e compromissos. Cada correspondência é governada por *regras de correspondência* específicas utilizadas para reforçar as relações dentro de uma descrição arquitetural. Desse modo, uma regra de correspondência é reforçada se for possível mostrar uma associação de correspondências que satisfaça a regra. Por outro lado, a regra de correspondência é violada quando for possível mostrar que uma correspondência não satisfaz a regra. Por fim, uma decisão arquitetural registra explicações, justificativas e raciocínio para uma determinada decisão como, por exemplo, razões para a decisão, alternativas e trocas consideradas, e possíveis consequências. Na Figura 2.6, são ilustrados os relacionamentos existentes entre esse conceitos e a descrição arquitetural.

O padrão ISO/IEC/IEEE 42010 também define que um *framework* arquitetural estabelece uma prática comum para a criação, interpretação, análise e uso de descrições arquiteturais dentro de um domínio específico de aplicação ou comunidade de interessados (ISO, 2010). O *framework* arquitetural "4+1" é um exemplo clássico de *framework* estabelecido por Kruchten (1995) que propõe o uso de quatro visões arquiteturais (lógica, de processo, desenvolvimento e física) mais uma quinta visão de cenários ou casos de uso para a representação de arquiteturas de software. A visão lógica descreve a decomposição orientada a objetos do sistema e apresenta essencialmente os requisitos funcionais desse sistema. Nessa visão, a arquitetura do sistema é representada em termos de camadas,

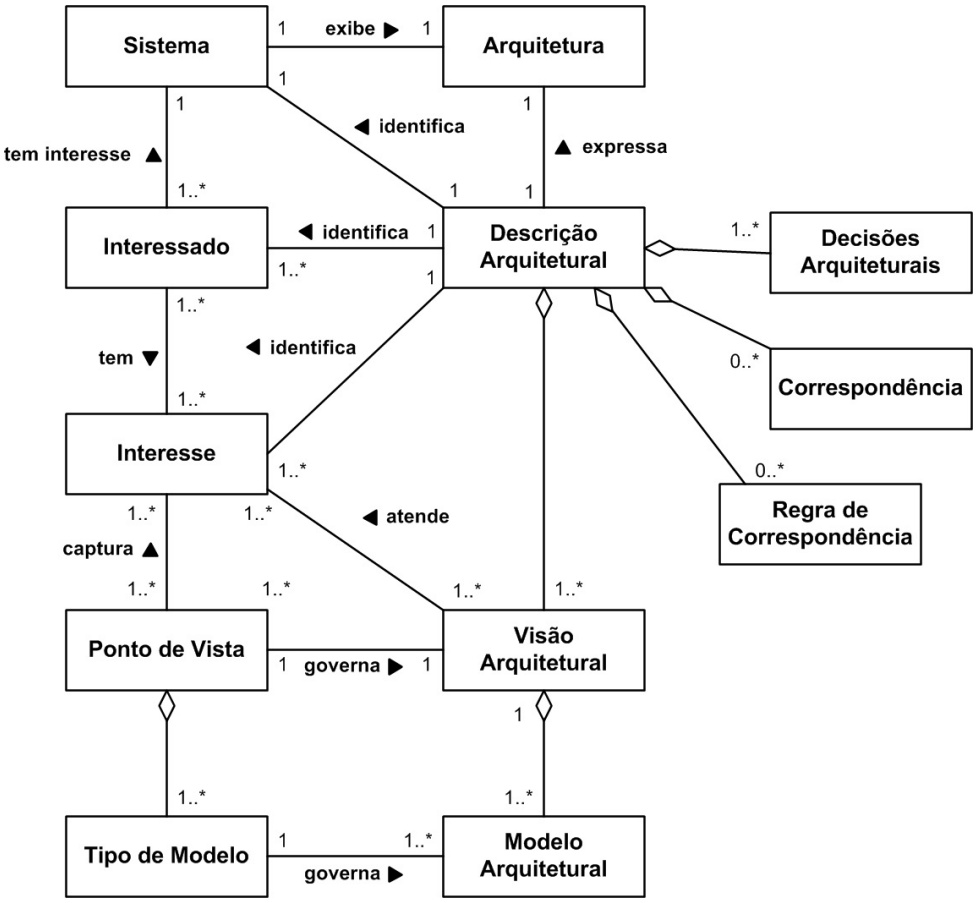

Figura 2.6: Modelo conceitual para a representação de uma arquitetura (ISO, 2010)

subsistemas, pacotes, *frameworks*[8], classes e interfaces mais importantes. Já na visão de processos a arquitetura do sistema é descrita em termos de processos e linhas de execução da aplicação, indicando responsabilidades, colaborações e a alocação de elementos lógicos, podendo apresentar requisitos não funcionais, tais como o desempenho e a disponibilidade. A visão de desenvolvimento descreve a decomposição da arquitetura em subsistemas e foca na organização real dos módulos de software no ambiente de desenvolvimento. Com base nessa visão é possível, por exemplo, alocar os grupos de trabalho, avaliar os custos e o planejamento do projeto, monitorar o progresso do projeto e pensar sobre o reúso, a portabilidade e a proteção do sistema de software. A visão física descreve o mapeamento do software no *hardware* e considera, essencialmente, os requisitos não funcionais do sistema como confiabilidade, desempenho e escalabilidade. Diferentes configurações físicas podem ser adotadas de acordo com os objetivos de desenvolvimento e teste, ou de implantação do sistema para diferentes consumidores. Por fim, a visão de casos de uso reúne um entendimento sobre todo o conjunto de visões e de como elas interagem entre si. A relação

[8]Entende-se por *framework* um conjunto de classes abstratas e concretas que colaboram entre si, que pode ser usado como um modelo para resolver uma família de problemas relacionados, podendo ser estendido para atender um comportamento específico da aplicação.

existente entre essas visões e os interessados de cada uma delas é ilustrada na Figura 2.7. Atualmente, o modelo foi estendido para "N+1" visões refletindo os vários interesses que podem existir em uma arquitetura.

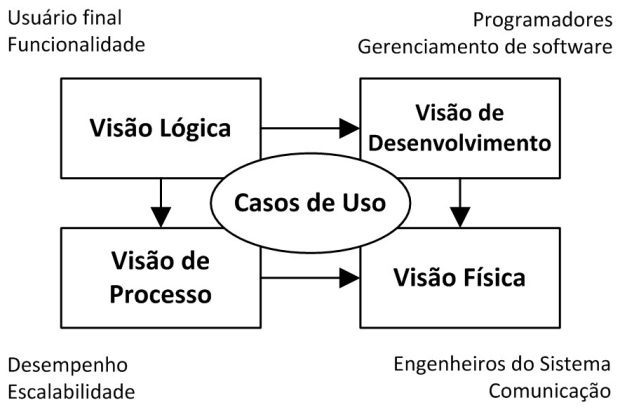

Figura 2.7: Modelo de visões "4+1" (Traduzido de (Kruchten, 1995))

Outro *framework* bastante conhecido é o "*Views and Beyond*" proposto por Clements et al. (2011). Segundo os autores, para representar adequadamente uma arquitetura deve-se primeiro documentar as visões arquiteturais mais relevantes e, em seguida, adicionar uma documentação que se aplica a mais de uma visão. Nesse caso, entende-se por uma visão relevante aquela que melhor satisfaz os interesses dos interessados mais importantes na arquitetura. Para facilitar o entendimento do modelo, Clements et al. (2011) agrupam as visões em três categorias principais: (i) visões modulares, que descrevem a estrutura do sistema como um conjunto de unidades de implementação; (ii) visões de componentes e conectores (C&C), que descrevem a estrutura do sistema no momento em que esse está sendo executado; e (iii) visões de implantação, que descrevem como o sistema relaciona-se com outras estruturas em seu ambiente. Desse modo, sempre é possível classificar uma visão como pertencendo à alguma dessas categorias ou como uma combinação de informações presentes em mais de uma delas.

Uma importante contribuição do padrão IEEE/ISO/IEC 42010 é o estabelecimento de diretrizes para a definição de *frameworks* arquiteturais e descrições arquiteturais compatíveis com o padrão. Por exemplo, para um *framework* arquitetural estar em conformidade com o padrão, ele precisa identificar um ou mais interesses do sistema, um ou mais interessados que possuem esses interesses e um ou mais pontos de vista que abordam esses interesses. Em especial, a especificação de um *framework* deve incluir também condições para a sua aplicabilidade como, por exemplo, um determinado ponto de vista não precisa ser incluído se o interesse que ela captura não foi identificado. Já no caso da descrição arquitetural, para que ela seja compatível com o padrão é preciso que ela identifique, por exemplo, os interessados no sistema e seus respectivos interesses, as regras de correspondência aplicáveis, defina cada ponto de vista utilizado e documente as decisões

arquiteturais realizadas. É importante ressaltar, contudo, que o padrão IEEE/ISO/IEC 42010 não determina a granularidade dos interesses do sistema, ou como esses interesses estão relacionados entre si e com outras declarações do sistema, tais como as necessidades dos interessados, os objetivos ou os requisitos do sistema. Essas informações devem ser esclarecidas por *frameworks* ou métodos arquiteturais específicos (ISO, 2010). No entanto, mesmo sendo largamente reconhecido na literatura o uso de múltiplas visões arquiteturais para representação de arquiteturas de software, ainda não há entre os pesquisadores da área um consenso sobre quais visões são necessárias para descrever uma arquitetura ou sobre quais são os métodos apropriados para expressar cada ponto de vista (ISO, 2010).

2.4.1 Decisões Arquiteturais

Farenhorst e de Boer (2009) destacam que o conhecimento arquitetural tem ganho cada vez mais importância nas discussões sobre reúso, projeto e evolução. As decisões arquiteturais esclarecem, por exemplo, os motivos por trás de uma determinada escolha ou as alternativas consideradas antes da decisão certa ou ótima ter sido tomada. As vantagens em se documentar decisões arquiteturais podem ser observadas em médio e longo prazo, quando a existência de tal documentação previne a necessidade de processos para a recuperação da arquitetura, comuns quando o projeto, a documentação ou até mesmo os criadores da arquitetura não estão mais disponíveis (Kruchten, 2009). Todo o conhecimento arquitetural precisa ser gerenciado para guiar a evolução dos sistemas e prevenir a vaporização do conhecimento (Bosch, 2004). Porém, como a própria evolução natural do sistema de software faz com que algumas decisões de projeto sejam substituídas por novas, o gerenciamento de conhecimento deve ser uma atividade contínua no ciclo de vida para garantir que as mudanças no código e no projeto estejam alinhadas às decisões arquiteturais (Kruchten, 2009).

Diferentes abordagens podem ser empregadas para o gerenciamento de decisões arquiteturais, como por exemplo a criação de uma *wiki* para os envolvidos no projeto da arquitetural ajuda na colaboração e divulgação do conhecimento entre todos os interessados, ou a criação de um *index* de habilidades (do inglês, *skill index*) relacionando as pessoas envolvidas no projeto e as áreas de especialidade ajuda na personalização do conhecimento. Nesse contexto, destaca-se o trabalho de van Heesch et al. (2012) que propõe um *framework* arquitetural composto por quatro pontos de vista para a documentação de decisões arquiteturais, sendo esses: ponto de vista de detalhes da decisão, ponto de vista cronológico de decisões, ponto de vista de relacionamentos entre decisões e ponto de vista de interessados nas decisões. A visão de detalhes da decisão apresenta informações detalhadas sobre decisões individuais como, por exemplo, explicações, alternativas, consequências, dentre outras. A visão cronológica de decisões descreve a evolução das decisões

arquiteturais com o tempo. Nessa visão, as decisões são acrescidas de informações de tipo e data, afim a apresentar todas as suas versões, ou seja, estados assumidos pela decisão, em todas as iterações da arquitetura. A visão de relacionamentos entre as decisões mostra, além do relacionamento, o estado atual de cada decisão de modo a apresentar um quadro do sistema em um momento específico. Por fim, a visão de interessados nas decisões descreve as responsabilidades de cada interessado no processo de tomada de decisão. Visões resultantes desse ponto de vista não apresentam o componente temporal, de modo que apenas as decisões, ações e interessados envolvidos em uma única iteração da arquitetura são apresentados. A informação capturada por esse ponto de vista é importante para personalizar o conhecimento arquitetural.

2.4.2 Linguagens de Descrição Arquitetural

Entende-se por uma linguagem de descrição arquitetural (do inglês, *architectural description language*, ADL) qualquer forma de expressão que possa ser empregada para descrever uma arquitetura (ISO, 2010). Como exemplo de ADLs, pode-se citar as linguagens AADL(SAE, 2011), Rapide (Luckham, 1996), Wright (Allen, 1997), UML (OMG, 2010b) e SysML (OMG, 2010a). Segundo Shaw e Garlan (1994), uma ADL caracteriza-se por fornecer mecanismos que permitam a composição, abstração, reusabilidade, configuração, heterogeneidade e análise da descrição arquitetural. As ADLs, semelhantemente aos *frameworks* arquiteturais, também capturam interesses dos interessados utilizando, para isso, um ou mais tipos de modelo e/ou pontos de vista (ISO, 2010). Nesse sentido, uma ADL pode ser específica, definindo um único tipo de modelo, ou mais abrangente, definindo vários tipos de modelo, os quais por sua vez também podem estar organizados em pontos de vista (ISO, 2010). Normalmente, as ADLs também contam com o apoio de ferramentas especializadas para apoiar a criação, uso e análise automática de seus modelos (ISO, 2010).

Em vista da grande variedade de ADLs que podem ser adotadas e também das características que cada uma delas apresenta, é relevante a existência de meios que permitam a classificação e comparação das diferentes linguagens. Uma boa forma para a classificação de ADLs consiste em avaliar os fundamentos teóricos da linguagem. Segundo esse critério, elas podem ser divididas em duas classes: (i) formais, baseadas em fundamentos matemáticos, cujo principal objetivo é habilitar a verificação automatizada de propriedades da arquitetura do sistema; e (ii) semi-formais, cujo principal objetivo é apoiar a comunicação entre os interessados de uma determinada arquitetura. Cada tipo de linguagem pode contribuir para a descrição arquitetural de maneiras distintas. Nesse sentido, o uso de uma ADL formal permite a definição clara e não ambígua de propriedades de uma arquitetura (Allen, 1997). Para isso, essas ADLs são baseadas em gramáticas bem

definidas e que podem ou não oferecer suporte às notações visuais como, por exemplo, as linguagens WRIGHT (Allen, 1997) e Rapide (Luckham, 1996). Por outro lado, o uso de uma ADL semi-formal como, por exemplo, as linguagens UML (OMG, 2010b) e SysML (OMG, 2010a), permite a criação de uma representação da arquitetura que, muitas vezes, facilita a sua compreensão e comunicação. Por serem baseadas principalmente em notações visuais, ADLs semi-formais não requerem, necessariamente, um conhecimento profundo sobre a linguagem para serem compreendidas ou utilizadas por interessados com diferentes níveis de experiência. No entanto, a falta de uma fundamentação formal desse tipo de linguagem pode resultar, em última instância, na interpretação errônea do modelo ou ainda permitir que incompatibilidades no modelo sejam transmitidas para etapas de desenvolvimento subsequentes.

Nesse contexto, destaca-se o *framework* proposto por Medvidovic e Taylor (2000), que considera como elementos essenciais de uma linguagem de descrição arquitetural os seus componentes, as interfaces desses componentes, os conectores e a configuração arquitetural ou topologia. No Quadro 2.1, são apresentados os elementos de ADLs considerados pelo *framework* proposto. É importante notar que os elementos essenciais de uma ADL estão destacados em negrito. A existência de ferramentas de apoio não é considerada como essencial, segundo o *framework* proposto; porém, pode ser um fator determinante para definir a escolha por uma determinada ADL. Em especial, observa-se também que diversas outras características de componentes, conectores e configurações arquiteturais podem existir e serem desejáveis em linguagens de descrição arquitetural como, por exemplo, a capacidade para especificação de restrições, propriedades não funcionais e tipos, além de atender requisitos como escalabilidade e dinamismo. De modo geral, é possível verificar que o *framework* leva em consideração uma variedade maior de requisitos e recursos presentes nas linguagens de descrição arquitetural para classificar e comparar diferentes linguagens. Desse modo, ele representa uma importante contribuição para identificar as vantagens, limitações e viabilidade de cada linguagem.

Dentre as linguagens de descrição arquitetural existentes, é importante destacar a UML (do inglês, *Unified Modeling Language*) (OMG, 2010b) como uma das mais conhecidas. Por conta da grande variedade de diagramas disponibilizados pela linguagem e também das várias ferramentas que apoiam a sua adoção em diferentes etapas do ciclo de vida do software, a UML tem sido a base para a modelagem de muitos sistemas de software. Em especial, a capacidade de descrição arquitetural da linguagem foi aprimorada com a adição e modificação de elementos da linguagem a partir de sua versão 2.0, promovendo a adoção da UML também para esse fim (Ivers et al., 2004). Em sua versão 2.3, a UML disponibiliza um conjunto de 13 diagramas divididos em três categorias, conforme mostrado na Figura 2.8, para a criação de modelos visuais de sistemas de software (OMG,

1.Recursos de modelagem arquitetural
- **1.1 Componentes**
  - **Interface**
  - Tipos
  - Semântica
  - Restrições
  - Evolução
  - Propriedades não funcionais
- **1.2 Conectores**
  - Interface
  - Tipos
  - Semântica
  - Restrições
  - Evolução
  - Propriedades não funcionais
- **1.3 Configurações Arquiteturais**
  - Compreensibilidade
  - Composição hierárquica
  - Refinamento e rastreabilidade
  - Heterogeneidade
  - Escalabilidade
  - Evolução
  - Dinamismo
  - Restrições
  - Propriedades não funcionais

2. Ferramenta de apoio
- Especificação ativa
- Múltiplas perspectivas
- Análise
- Refinamento
- Escalabilidade
- Geração de código
- Dinamismo

Tabela 2.1: *Framework* para classificação e comparação de ADLs (Medvidovic e Taylor, 2000)

2010b): (i) Diagramas Estruturais, que incluem o Diagrama de Classes, o Diagrama de Objetos, o Diagrama de Componentes, o Diagrama de Estrutura Composta, o Diagrama de Pacotes e o Diagrama de Implantação; (ii) Diagramas Comportamentais, que incluem o Diagrama de Casos de Uso, o Diagrama de Atividades e o Diagrama de Máquina de Estados; e (iii) Diagramas de Interação, que são derivados dos Diagramas Comportamentais e incluem o Diagrama de Sequência, o Diagrama de Comunicação, o Diagrama de Tempo e o Diagrama de Visão Geral da Interação. A UML também oferece mecanismos para a criação de perfis da UML, isso é, subconjuntos da UML adaptados para um deter-

minado fim, que apoiam a modelagem de características particulares como, por exemplo, de tempo-real ou tolerância a falhas (OMG, 2010b).

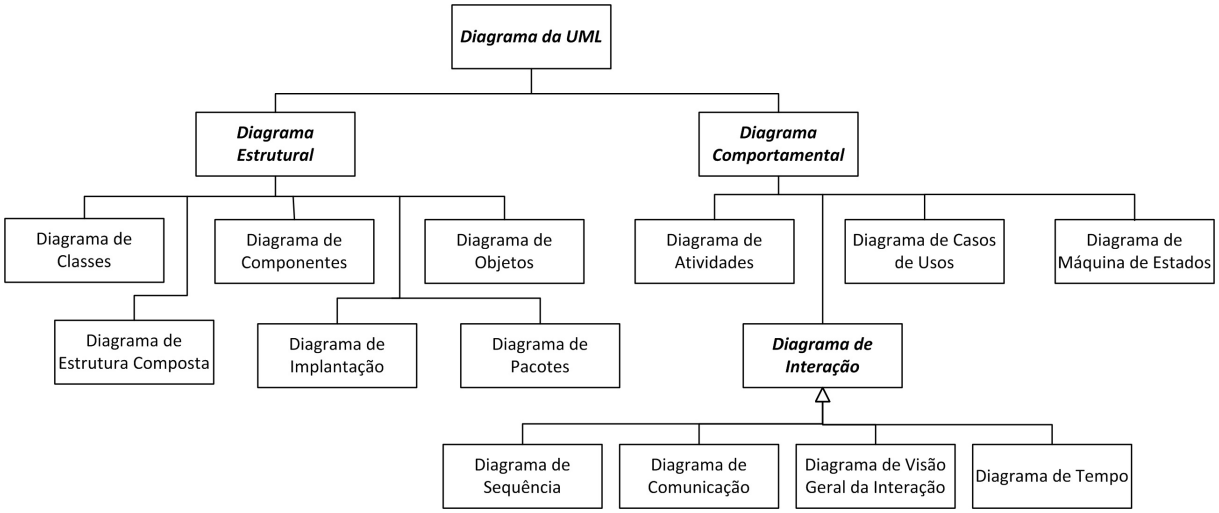

Figura 2.8: Tipos de diagramas da UML (Adaptado de (OMG, 2010b))

Em uma outra perspectiva, enquanto muitas ADLs focam apenas na descrição de interesses estruturais da arquitetura como, por exemplo, componentes, conectores, configuração e comportamentos, recentemente, uma linha de ADLs mais abrangente com foco em uma variedade maior de interesses têm surgido. Dentre elas, destacam-se as linguagens AADL (do inglês, *Architecture Analysis and Design Language*) (SAE, 2011) e SysML (do inglês, *Systems Modeling Language*) (OMG, 2010a). A AADL, por exemplo, é uma linguagem semi-formal projetada para a especificação, análise e integração automatizada de sistemas de software embarcados, críticos e de tempo-real (SAE, 2011). Para isso, a linguagem inclui abstrações de software, hardware e componentes do sistema que permitem mapear o software em elementos computacionais de hardware (Feiler et al., 2006). Além disso, a linguagem permite a descrição de propriedades como proteção, desempenho, tolerância a falhas, segurança, dentre outras. Muitos fatores contribuíram para o sucesso atual alcançado pela AADL, dentre eles: (i) a existência de notações textuais e visuais para a linguagem; (ii) a disponibilidade de várias ferramentas de apoio comerciais e gratuitas[9]; e (iii) a contribuição efetiva de importantes organizações no desenvolvimento da linguagem como, por exemplo, SEI[10], Airbus[11], Honeywell[12], *European Space Agency*[13] e GE[14] (SAE, 2011). A SysML é uma linguagem semi-formal para a especificação, análise,

[9]Uma lista atualizada de ferramentas disponíveis pode ser encontrada em `http://www.aadl.info/aadl/currentsite/start/aadltoolsupport.html` (Acessado em 25/02/2012)
[10]`http://www.sei.cmu.edu`
[11]`http://www.airbus.com`
[12]`http://honeywell.com`
[13]`http://www.esa.int`
[14]`http://www.ge.com`

projeto e verificação de sistemas complexos que podem envolver hardware, software, informação, pessoas, procedimentos e instalações (OMG, 2010a). É importante destacar que a SysML consiste em um perfil da UML para a modelagem de requisitos, comportamentos, estruturas e parâmetros do sistema que são utilizados para integrar com outros modelos de análise da engenharia (OMG, 2010a). A SysML disponibiliza vários tipos de diagramas, conforme ilustrado na Figura 2.9, para apoiar todo o ciclo de vida do software, sendo eles: Diagrama de Atividades, Diagrama de Sequência, Diagrama de Máquina de Estados, Diagrama de Casos de Uso, Diagrama de Definição de Blocos, Diagrama de Blocos Internos, Diagrama de Pacotes, Diagrama Paramétrico e Diagrama de Requisitos (OMG, 2010a). O Anexo A apresenta uma breve introdução sobre a linguagem e seus diagramas. Vale destacar ainda que cada um dos diagramas definidos pela linguagem pode ser compreendido como um tipo de modelo segundo o padrão ISO/IEC/IEEE 42010. Além disso, a disponibilização de construtores de primeira ordem para interessados, interesses, visões e pontos de vista faz com que a SysML permita a criação de novos pontos de vista em conformidade com o padrão ISO/IEC/IEEE 42010.

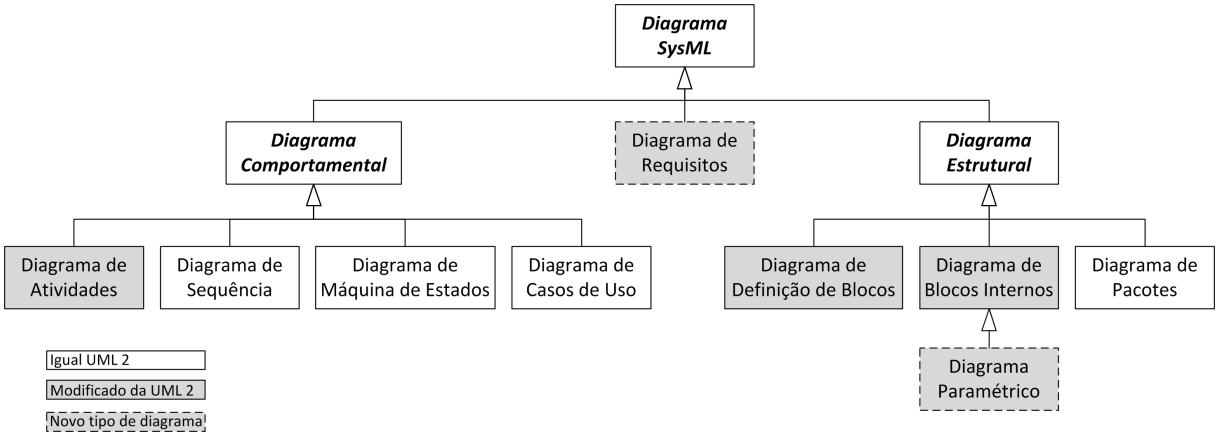

Figura 2.9: Tipos de diagramas da SysML (Adaptado de (OMG, 2010a))

É importante destacar também a possibilidade de utilização da AADL, SysML e UML de forma conjunta para agregar as vantagens oferecidas por cada uma delas e, assim, melhorar a previsibilidade no desenvolvimento de sistemas de software, um requisito de especial importância para sistemas do domínio de embarcados. O uso dessas linguagens em conjunto é vantajoso, pois combina os recursos de modelagem oferecidos pela SysML e UML com a arquitetura em tempo de execução da AADL (Niz, 2007). Desse modo, o uso de várias linguagens em conjunto pode ajudar a esclarecer e melhorar o conhecimento sobre o comportamento e a estrutura de arquiteturas de software. Em especial, vale destacar a proposição da linguagem Acme (Garlan et al., 1997), que embora não se encaixe como uma ADL no sentido estrito da definição apresentada, ainda é uma importante contribuição no

sentido de fornecer uma forma comum para a permutação entre diferentes linguagens de descrição arquitetural e, desse modo, possibilitar a integração de diferentes ferramentas.

2.5 Considerações Finais

Neste capítulo foi apresentada a teoria básica necessária sobre as áreas de arquitetura de software e arquitetura de referência. Exemplos de arquiteturas de referência existentes também foram apresentados. Adicionalmente, importantes definições relacionadas à descrição de arquiteturas de software e arquiteturas de referência foram apresentadas, bem como as principais características e exemplos de linguagens de descrição arquitetural. No próximo capítulo, o domínio de sistemas embarcados será investigado com o objetivo de compreender melhor as características que são específicas desse domínio, além de apresentar exemplos de arquiteturas de referência propostas para esse domínio e a investigação conduzida sobre a descrição dessas arquiteturas.

Capítulo

3

Arquitetura de Sistemas Embarcados

3.1 Considerações Iniciais

Os sistemas embarcados têm ganho cada vez mais destaque e estão cada vez mais presentes nos dias de hoje, com a multiplicação de dispositivos e aplicações. Nos mais diversos ramos como, por exemplo, na medicina, na indústria automobilística e de bens de consumo, sistemas embarcados desempenham tarefas sofisticadas e precisam atender rígidos requisitos tais como de segurança e desempenho, além de outros requisitos impostos pelo mercado, tais como ciclos de inovação cada vez mais curtos e ainda baixo custo de produção. Em vista dos múltiplos requisitos que podem existir, o desenvolvimento e a manutenção de sistemas embarcados têm se tornado cada vez mais complexos. Nesse contexto, arquiteturas de referência para o domínio de sistemas embarcados têm sido propostas com o objetivo de promover o reúso do projeto arquitetural desses sistemas melhorando o entendimento que se tem sobre esse domínio e permitindo o desenvolvimento mais eficiente de novos sistemas.

Neste capítulo, são apresentados os principais conceitos e terminologias do domínio de sistemas embarcados. Na Seção 3.2 é introduzida a área de sistemas embarcados. Na Seção 3.3 são apresentadas algumas arquiteturas de referência propostas para esse domínio e uma discussão sobre essas arquiteturas. Por fim, na Seção 3.5 são apresentadas as considerações finais deste capítulo.

3.2 Sistemas Embarcados

Entende-se por sistemas embarcados aqueles sistemas de software encontrados em dispositivos que incluem um computador programável, mas que não são considerados como um computador de propósito geral (Wolf, 2008). Esses sistemas costumam estar conectados com o ambiente físico por meio de sensores que coletam informações sobre o ambiente e atuadores que exercem uma ação no ambiente (Marwedel, 2011). Diariamente, uma pessoa comum pode interagir com centenas de sistemas embarcados presentes em televisores, celulares, sistemas de GPS e automóveis, apenas para citar alguns. A tendência é que esse número continue aumentando com a modernização dos meios de transporte, comunicação e de bens de consumo disponíveis.

De modo geral, sistemas embarcados precisam oferecer funcionalidades mais sofisticadas daquelas oferecidas por sistemas de propósito geral, uma vez que necessitam de algoritmos e interfaces com o usuário mais complexas (Wolf, 2008). No entanto, outras características também diferenciam o desenvolvimento desses sistemas daqueles de propósito geral. Por exemplo, sistemas embarcados precisam apresentar um alto nível de integração e podem envolver diversas disciplinas de engenharia (como mecânica, elétrica, química e computação) (Mesman et al., 2002). Nesse sentido, destaca-se nos últimos 20 anos um maior impacto do software tanto nas funcionalidades como também no potencial de inovação e diferenciação dos novos produtos de sistemas embarcados (Liggesmeyer e Trapp, 2009). Além dessas características, é importante destacar também o maior importância de requisitos não funcionais no projeto desses sistema como, consumo de energia, tamanho físico, peso e custo, que influenciam nas decisões de projeto e na arquitetura do sistema final. Por exemplo, o baixo consumo de energia é praticamente um requisito padrão dos sistemas embarcados, já que muitos deles serão empregados em dispositivos móveis e, portanto, devem apresentar maior autonomia. Além dele, tem-se ainda restrições de tamanho e peso, também essenciais para garantir a portabilidade, e o baixo custo, para viabilizar a produção em larga escala e incentivar o consumo desses sistemas (Wolf, 2008). Outras características, como restrições de confiabilidade, robustez e proteção, derivam de situações nas quais um certo nível de comportamento autônomo deve ser necessário (Mesman et al., 2002). Nesse sentido, a confiabilidade de um sistema estabelece a probabilidade de que o sistema não irá falhar; a robustez indica com qual segurança, no caso de uma falha, o sistema é capaz de se recuperar; e a segurança indica se, no caso de uma falha, o sistema não causa nenhum dano às pessoas, ao ambiente ou prejuízos financeiros. Vistos em conjunto, todos esses requisitos e restrições tornam o projeto de sistemas embarcados particularmente desafiador.

Uma pesquisa conduzida por Graaf et al. (2005) para descobrir quais são os métodos, ferramentas e técnicas existentes e atualmente em uso no desenvolvimento de sistemas embarcados na indústria[1] revelou que arquitetos de software estavam pouco envolvidos no projeto do sistema e que o desenvolvimento de hardware basicamente dominou o processo de desenvolvimento por conta dos prazos mais longos e de dependências logísticas de fornecedores externos. Consequentemente, o desenvolvimento do software apenas era iniciado quando o desenvolvimento do hardware já estava em um estágio muito avançado no qual mudanças seriam muito caras. Além disso, as propriedades do hardware acabaram por limitar o espaço de solução do software, resultando até mesmo em situações em que o software cumpria o papel de integrador, ou seja, incompatibilidades que deveriam ter sido resolvidos no domínio de hardware foram resolvidas no domínio de software. Para evitar que situações assim continuassem se repetindo, em muitas empresas, os arquitetos de software já estavam começando a participar mais ativamente do projeto no nível do sistema. Resultados da pesquisa também revelaram que devido à falta de sintaxes e semânticas mais bem definidas nas descrições dos requisitos, era muito comum que membros do projeto interpretassem erroneamente diagramas do projeto.

O sucesso alcançado pelos sistemas embarcados está fortemente atrelado à capacidade de desenvolver sistemas de software de alta qualidade dentro do prazo estipulado (Graaf et al., 2005). Segundo Mesman et al. (2002), para conseguir lidar com a complexidade cada vez maior do projeto de sistemas embarcados, será preciso dar maior ênfase na exatidão do fluxo de desenvolvimento, nos recursos das descrições utilizadas e também na exatidão semântica dessas descrições, resultando finalmente na formalização de todo o processo de desenvolvimento. Se as especificações não forem resultado de uma abordagem sistemática e bem definida, então não há nenhuma garantia de que o sistema resultante dessa especificação atende aos requisitos iniciais. Além disso, um fluxo de desenvolvimento melhor definido também estará menos propenso a erros e, consequentemente, melhorará a qualidade dos sistemas desenvolvidos.

Dada a importância da descrição arquitetural para os sistemas embarcados, é preciso que ela seja considerada parte integral do sistema de software e de sua documentação. Considerando-se, em particular, os sistemas embarcados críticos, uma descrição arquitetural explícita é importante mas ainda não suficiente para gerenciar a complexidade de desenvolvimento, manutenção e evolução desses sistemas (Mens et al., 2010). Para gerenciar esses sistemas, ainda é preciso haver uma conexão precisa e rastreável entre a descrição e a implementação do sistema, de modo que qualquer mudança na arquitetura seja diretamente refletida na implementação e vice-versa, ou caso contrário a descrição

[1] A pesquisa entrevistou engenheiros, pesquisadores, arquitetos de software ou do sistema e gerentes das empresas TeamArteche (Espanha), Nokia (Finlândia), Solid (Finlândia), VTT Electronics (Finlândia), Philips PDSL (Holanda), ASML (Holanda), Océ (Holanda) e LogicaCMG (Holanda).

arquitetural se tornará rapidamente obsoleta (Mens et al., 2010). Nesse contexto, mecanismos capazes de garantir que as mudanças arquiteturais preservam os requisitos inicias são necessários, por exemplo, pela análise e verificação da descrição arquitetural final, ou da análise da descrição arquitetural acrescida das mudanças realizadas. Contudo, para isso é preciso fornecer construtores estruturais de primeira ordem para expressar e capturar mudanças arquiteturais durante o desenvolvimento inicial e evolução. Para que as tecnologias existentes possam ser, de fato, adotadas pela indústria no desenvolvimento de sistemas embarcados, também é preciso ter em mente que essas tecnologias, normalmente complexas, precisam estar integradas com tecnologias de mais fácil compreensão principalmente porque o processo de desenvolvimento de software costuma envolver também interessados que não são especialistas da área e que podem apresentar dificuldade em compreender tecnologias complexas discutidas anteriormente, como linguagens de descrição arquitetural formais utilizadas na especificação de requisitos críticos de segurança (Graaf et al., 2005).

Nesse contexto, a disciplina de arquitetura de software pode contribuir para a área de sistemas embarcados no sentido de fornecer importantes recursos para o projeto mais eficiente desses sistemas como arquiteturas de referência e linguagens que possibilitem a descrição e análise arquitetural (Medvidovic, 2005). Cada organização pode aplicar as tecnologias de arquitetura de software mais apropriadas para suas situações específicas, visando otimizar a produtividade, a pontualidade e a qualidade do desenvolvimento de sistemas embarcados (Graaf et al., 2005). Em particular, como normalmente há um número maior de disciplinas envolvidas no desenvolvimento desses sistemas, problemas de comunicação tendem a aparecer com maior facilidade, fazendo-se necessário a utilização de terminologias e padrões que sejam comuns a todas as etapas de desenvolvimento (Mesman et al., 2002). Nesse sentido, a introdução de arquiteturas de referência para o domínio de sistemas embarcados possui grande valor, uma vez que oferece um léxico e taxonomias comuns, além de caracterizar as principais funcionalidades e restrições desses sistemas, promovendo assim o reúso do projeto arquitetural e um melhor entendimento sobre o domínio.

3.3 Arquiteturas de Referência de Sistemas Embarcados

Em virtude da complexidade crescente no desenvolvimento e manutenção dos sistemas de software embarcados, a introdução de arquiteturas de referência pode ser uma solução viável para melhorar e sistematizar o desenvolvimento de sistemas desse domínio. A aplicação de arquiteturas de referência pode trazer muitos benefícios, tais como a integrabilidade entre os sistemas resultantes de instâncias arquiteturais, o reúso de decisões arquiteturais,

a padronização do software básico, a portabilidade mais fácil entre diferentes plataformas, a maior inteligibilidade das descrições arquiteturais pela abstração das funcionalidades e recursos mais importantes e a formalização do processo de desenvolvimento de arquiteturas desse domínio (Eklund et al., 2005). De fato, diversas arquiteturas de referência podem ser encontradas para esse domínio como, por exemplo, para sistemas automotivos (AUTOSAR, 2013; Eklund et al., 2005), sistemas de avião (Batory et al., 1995), sistemas de ciências da terra (NASA ESDS Working Group, 2012), sistemas multiagentes locais (Weyns, 2006) e sistemas para automação de lares (UniversAAL Project, 2011).

3.3.1 Exemplos

Nesta seção, três arquiteturas de referência para o domínio de sistemas embarcados são descritas em mais detalhes, sendo elas: AUTOSAR (AUTOSAR, 2013), ESDS (*Earth Science Data Systems*) (NASA ESDS Working Group, 2012) e para Sistemas Multiagentes Locais (Weyns, 2006). As duas primeiras arquiteturas de referência foram desenvolvidas por consórcios englobando os principais nomes da indústria (fabricantes e fornecedores) e pesquisadores, sendo a primeira uma arquitetura de referência concreta para os sistemas que vão compor os veículos modernos e a segunda uma arquitetura de referência abstrata para guiar o desenvolvimento mais eficiente de sistemas de ciências da terra. Por fim, a terceira arquitetura de referência apresentada é uma arquitetura abstrata para guiar o projeto e a implementação de sistemas multiagentes locais.

AUTOSAR

Atualmente, os carros mais sofisticados já saem de fábrica equipados com uma grande variedade de sistemas embarcados. São sistemas para controle dos *air bags*, do motor, dos freios, do ar condicionado, do sistema de posicionamento global (GPS), contra roubos, da estabilidade do veículo e outras características de segurança (Marwedel, 2011). Frente o surgimento de novas aplicações para sistemas embarcados nos veículos modernos, as arquiteturas automotivas elétricas/eletrônicas que existiam até então mostraram-se ineficientes para gerenciar e ajudar a atingir os requisitos mais rigorosos que estavam surgindo (AUTOSAR, 2013). Essa necessidade ficou particularmente evidente quando os fabricantes de carros de luxo e seus principais fornecedores foram confrontados com requisitos muitas vezes conflitantes sobre aspectos legais do veículo (requisitos ambientais e de proteção), funcionalidades de conforto e entretenimento dos passageiros, requisitos de assistência ao condutor e controle de aspectos dinâmicos da direção, como detecção e proibição dinâmica de estados críticos do veículo para navegação em ambientes de tráfego intenso. A arquitetura de referência AUTOSAR (do inglês, *AUTomotive Open System ARchitecture*), criada por uma parceria entre as principais montadoras (BMW, Bosch, Toyota, Continental, Daimler, Ford, GM, Peugeot, Volkswagen) e fornecedores do setor automobilístico,

foi uma resposta direta a essas dificuldades visando criar uma base para a colaboração de funcionalidade básicas desses sistemas, ao mesmo tempo em que a competição por funções inovadoras é estimulada (AUTOSAR, 2013). De fato, a arquitetura tornou-se um padrão global para o desenvolvimento de sistemas de software embarcados automotivos, provendo especificações para a arquitetura de software desses sistemas, uma metodologia de desenvolvimento e interfaces de aplicação padronizadas (Heidary, 2011). Os objetivos da arquitetura de referência incluem também melhorar a modularidade, escalabilidade, portabilidade, reusabilidade e padronizãção dos sistemas desse domínio. Nesse sentido, a modularização dos elementos de software em um veículo permitirá a customização do software de acordo com requisitos individuais das unidades de controle eletrônico (referenciadas pela sigla em inglês ECU[2]) e suas tarefas. Já a escalabilidade das funções irá assegurar a adaptabilidade de módulos de software comuns à diferentes plataformas de veículos. A portabilidade das funções irá otimizar o uso de recursos disponíveis por toda a arquitetura eletrônica do veículo. A reusabilidade das funções ajudará a melhorar a qualidade e confiabilidade dos produtos e ainda reforçar a logomarca corporativa entre linhas de produto. Para que todos esses objetivos possam ser alcançados, a padronização de interfaces funcionais entre fabricantes e fornecedores e entre diferentes camadas de software é essencial.

O formato padrão proposto pela AUTOSAR para descrever interfaces e outros aspectos necessários para a integração de componentes de software informa (AUTOSAR, 2013): (i) as operação e os dados requeridos e fornecidos; (ii) os requisitos de infraestrutura; (iii) os recursos necessários (por exemplo, memória e tempo de CPU); e (iv) informações específicas sobre a implementação do componente de software. Cada componente de software AUTOSAR é definido de forma a encapsular parte da funcionalidade da aplicação. Embora a arquitetura de referência não prescreva qual deve ser o tamanho de cada componente, que pode ser ajustado de acordo com os requisitos da aplicação, um componente não pode estar distribuído por várias unidades de controle. A implicação direta dessa restrição é de que cada instância de um determinado componente de software AUTOSAR deve ser atribuída a uma única unidade de controle. Uma visão simplificada de componentes e interfaces da AUTOSAR é ilustrada na Figura 3.1. Em especial, as interfaces podem ser independentes –denominada Interface AUTOSAR– ou não –denominada Interface Padronizada– de uma linguagem de programação específica, ECU ou tecnologia de comunicação. Por esse motivo, as interfaces do segundo tipo são normalmente utilizadas entre os módulos de software que estão em um mesma unidade de controle. Um barramento funcional virtual (do inglês, *Virtual Functional Bus*, VFB)

[2]Sistema de software embarcado que consiste em pelo menos uma unidade de processamento e periféricos correspondentes que são colocados em um único compartimento (AUTOSAR, 2010)

é a soma de todos os mecanismos de comunicação e interfaces essenciais para o software básico fornecido pela AUTOSAR em um nível abstrato, ou seja, independente de tecnologia. Quando as conexões entre o componente de software AUTOSAR para um sistema concreto forem definidas, o VFB vai permitir a integração virtual deles em uma etapa inicial de desenvolvimento.

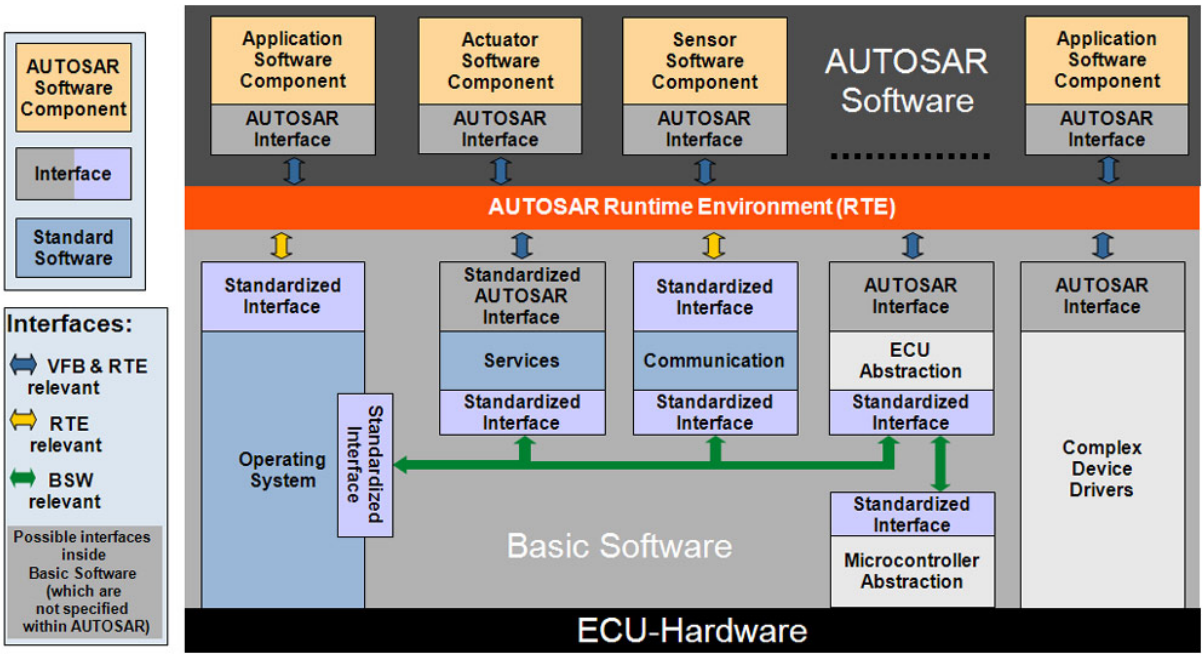

Figura 3.1: Visão simplificada de componentes e interfaces da AUTOSAR (AUTOSAR, 2013)

Para integrar os componentes de software AUTOSAR em uma rede de unidades de controle, a arquitetura de referência fornece formatos de descrição para todo o sistema e também para os recursos e configurações de uma única unidade de controle que são deixadas independentes das descrições de componentes de software (AUTOSAR, 2013). A arquitetura de referência também define uma metodologia e ferramentas de apoio para reunir as informações das várias descrições de elementos de modo a construir um sistema concreto de unidades de controle. Isso inclui a configuração e geração do ambiente de execução (do inglês, *Runtime Environment*, RTE) e do software básico em cada unidade de controle. Da perspectiva do componente de software da AUTOSAR, o ambiente de execução implementa as funcionalidade do barramento funcional virtual em uma unidade de controle específica, mas pode delegar essa tarefa para a camada de software básico o máximo que possível (AUTOSAR, 2013). Nesse sentido, a camada de software básico fornece as funcionalidades de infraestrutura de uma unidade de controle.

Segundo a classificação proposta por Angelov et al. (2009), a AUTOSAR melhor adere às arquiteturas de referência do Tipo 1, ou seja, arquiteturas clássicas visando a padronização dos sistemas desenvolvidos para o domínio automobilístico e com a interoperabilidade

um dos principais objetivos da arquitetura. A AUTOSAR é projetada para ser implementada em várias organizações, nesse caso, montadoras e fornecedores. Embora o nível de abstração adotado pela AUTOSAR (semi-concreta) seja diferente do sugerido para as arquiteturas do Tipo 1 (abstratas), os fabricantes ainda têm liberdade para definir seus próprios detalhes de implementação, garantindo assim a diferenciação dos produtos e estimulando a concorrência. O uso de técnicas formais na representação da arquitetura de referência permite a descrição de requisitos críticos e também a garantia de conformidade com padrões e protocolos de qualidade e segurança.

ESDS

Nas últimas décadas, melhorias nas tecnologias de medição geofísicas resultaram em uma explosão no volume e escopo do conjunto de dados das ciências da terra. A implementação de instrumentos para colher essas medições necessitou do desenvolvimento na mesma proporção de sistemas de dados que permitissem a coleta, o tratamento, o armazenamento e a distribuição do grande volume de dados resultante (NASA ESDS Working Group, 2012). Na medida em que mais desses sistemas foram sendo criados, os recursos comuns entre eles se tornaram cada vez mais evidentes. Além disso, outras razões também foram identificadas, dentre elas (NASA ESDS Working Group, 2012): (i) a necessidade de ampliação da automação do processo, da capacidade computacional e do armazenamento de dados; (ii) a maior complexidade para operação, instrumentos científicos e de dados precisará de um modelo de informação bem definido e melhorado além de assim como de um moderno dicionário de dados online para permitir o gerenciamento, processamento e armazenamento de diversos produtos distribuídos; (iii) a necessidade de uma arquitetura padrão para homogeneizar os dados, os métodos que gerenciam esses dados por diferentes plataformas, e as ferramentas de análise que projetam, geram, validam e entregam conjuntos de dados de qualidade; (iv) a simplificação do uso, exportação e exibição dos dados; e (v) a distribuição e o compartilhamento de serviços por todas as organizações e missões para assegurar o uso de um conjunto de ferramentas comuns em todos os ambientes locais. Nessa perspectiva, a arquitetura de referência NASA ESDS tem como finalidade descrever os recursos e padrões comuns dos sistemas existentes para fornecer diretrizes para a construição de novos sistemas. Essas diretrizes vão permitir que os interessados desses novos sistemas possam planejar e projetar sistemas completos e funcionais de maneira mais rápida e cuidadosa, reduzindo assim os custos e esforços necessários para a implementação (NASA ESDS Working Group, 2012).

Para alcançar os objetivos propostos, os seguintes interesses foram considerados na construção da arquitetura de referência (NASA ESDS Working Group, 2012): (i) independência da implementação; (ii) usabilidade; (iii) confiabilidade; (iv) independência de tecnologia; (v) escalabilidade, para permitir a integração mais fácil do sistema com novas

ferramentas, componentes e outros produtos *off-the-shelf*; (vi) interoperabilidade, para promover a consistência entre os sistemas, facilitar a manutenção dos serviços e também encorajar a adoção de soluções de terceiros; e (vii) segurança, seguindo as diretrizes da NASA. A especificação da arquitetura de referência procura definir os conjuntos de serviços oferecidos de forma desacoplada e, assim, permitir o intercâmbio e substituição desses serviços. Desse modo, cada serviço oferece aos usuários a generalidade e a flexibilidade necessários para compor o sistema que melhor adere aos requisitos do projeto.

A arquitetura de referência é descrita utilizando-se cinco visões arquiteturais, sendo elas (NASA ESDS Working Group, 2012): visão funcional, visão de arquitetura da informação, visão de serviços do sistema e visão de tecnologia. Na visão de funcionalidades, são identificadas e detalhadas as principais funções de alto nível dos sistemas que implementam a arquitetura de referência, com relação aos dados recebidos e tipo de processamento esperados, sem, no entanto, especificar detalhes da implementação. Uma visão geral de como é o fluxo de processamento dos dados dentro de cada função é descrito por meio de um diagrama de atividades. A visão de arquitetura da informação é semelhante a um modelo conceitual e tem como finalidade identificar em um alto nível de abstração as informações que são implementadas dentro do sistema ou trocadas entre eles, além de definir os relacionamentos entre as informações. Desse modo, a visão oferece um vocabulário comum para ser utilizado pelos sistemas que implementam a arquitetura de referência e contribui para maximizar o potencial de interoperabilidade dos sistemas e das informações trocadas entre eles. O modo pelo qual as informações são apresentadas visa não determinar uma solução específica, já que o objetivo é apenas orientar o projeto de sistemas desse domínio. A visão de serviços do sistema tem como objetivo identificar os serviços mais importantes no nível de sistema que são implementados. Baseado no padrão arquitetural orientado a serviços, os serviços são definidos como os pontos fundamentais da interface e da interação entre os componentes (denominados serviços do sistema), assim como entre os componentes e seus usuários (denominados serviços de usuários, pessoas ou aplicações). Em especial, um serviço também pode ser classificado como ambos. Nessa visão, também é feito um mapeamento dos serviços com os requisitos e também com as informações da arquitetura de referência, sendo que nesse último são identificadas as principais informações que apoiam um determinado serviço. Por fim, a visão de tecnologia guia a escolha e seleção de tecnologias e padrões. Considerando-se que o objetivo da arquitetura de referência é maximizar a interoperabilidade dos sistemas e recursos dentro do seu alcance, o uso de tecnologias e os padrões é essencial. No entanto, como a arquitetura de referência não especifica nenhuma tecnologia ou padrão em particular, o projeto e as restrições do sistema deverão guiar a escolha pela tecnologia mais apropriada.

Segundo a classificação proposta por Angelov et al. (2009), a ESDS melhor adere às arquiteturas de referência do Tipo 4, ou seja, arquiteturas de referência clássicas para serem utilizadas como mecanismos facilitadores. Contudo, diferentemente da classificação original, essa arquitetura pode ser utilizada por múltiplas organizações que queiram implementar funcionalidades compatíveis. A ESDS também é uma arquitetura abstrata e utiliza técnicas semi-formais (diagramas de atividades e modelos conceituais) e informais (recomendações) em sua representação.

Sistemas Multiagentes Locais

Um sistema multiagentes locais consiste em um ambiente (distribuído) populado por um conjunto de agentes que cooperam entre si para resolver um problema complexo de maneira descentralizada (Weyns, 2006). Um agente local possui uma posição explícita dentro do ambiente que pode ser acessado apenas localmente pelo agente, ou seja, o agente pode perceber, interagir com outros agentes e atuar apenas dentro de seu contexto local. A funcionalidade do sistema emerge da interação entre os agentes no ambiente, ao invés de ser resultado de uma atuação individual. Nesse contexto, o ambiente consiste em uma parte essencial do sistema, encapsulando recursos e permitindo que os agentes tenham acesso a esses recursos. Em especial, o ambiente consiste no meio em que as informações entre os agentes serão compartilhadas e o comportamento será coordenado. O controle nesses sistemas é, portanto, distribuído entre os agentes com a mediação do ambiente.

A arquitetura de referência para sistemas multiagentes locais é descrita utilizando-se quatro visões arquiteturais, sendo elas: a visão de decomposição em módulos, a visão de dados compartilhados, a visão de componentes colaborativos e a visão de processos comunicativos. Na visão de decomposição em módulos é mostrada a decomposição do sistema em unidades de software gerenciáveis. Na visão de dados compartilhados, o sistema é estruturado como um conjunto de assessores de dados que leem e escrevem informações em vários repositórios de dados compartilhados. Na visão de componentes colaborativos o sistema é apresentado como um conjunto de componentes em tempo de execução que utilizam um conjunto de repositórios de dados compartilhados para desempenhar as funcionalidades requeridas pelo sistema. Na visão de processos comunicativos o sistema é apresentado como um conjunto de unidades executando concorrentemente e a interação entre elas.Complementarmente à representação semi-formal da arquitetura de referência, os elementos também são definidos formalmente, usando a teoria de conjuntos, para permitir a estabelecimento de restrições que devem ser respeitadas por cada elemento. Nesse sentido, para evitar uma complexidade desnecessária, a descrição formal da arquitetura não define os conceitos que introduzem variabilidade e que, portanto, devem ser definidos de acordo com os requisitos de cada instância arquitetural.

Segundo a classificação proposta por Angelov et al. (2009), a arquitetura de referência para sistemas multiagentes locais melhor adere às arquiteturas do Tipo 5, ou seja, é uma arquitetura de referência preliminar para ser utilizada como mecanismo facilitador e foi desenvolvida dentro de um único centro de pesquisa para guiar o projeto e a implementação de sistemas desse domínio. É uma arquitetura abstrata, mas utiliza técnicas semi-formais e formais para convencer seus interessados de suas qualidades e promover sua adoção no projeto de instâncias arquiteturais.

3.4 Representação de Arquiteturas de Referência de Sistemas Embarcados

Descrever as arquiteturas de referência do domínio de sistemas embarcados de modo que todos os seus interessados tenham acesso às informações ali contidas é uma tarefa essencial para que elas sejam de fato utilizadas pelas organizações. Para isso, é relevante investigar maneiras de aprimorar a representação de arquiteturas de referência desse domínio de modo que seja possível alcançar os objetivos de padronização e facilitação no desenvolvimento de sistemas embarcados. Considerando as arquiteturas de referência detalhadas na seção anterior e também de *frameworks* arquiteturais apresentados no Capítulo 2, foi criada a Tabela 3.2 que relaciona os diferentes pontos de vista empregados ou sugeridos para a representação de arquiteturas de software e arquiteturas de referência, em especial para o domínio de sistemas embarcados. Para melhorar a clareza da Tabela 3.2, as fontes são descritas separadamente na Tabela 3.1. Da tabela, é importante notar, por exemplo, que não há uma nomenclatura padrão para os pontos de vista, de modo que existem alguns pontos de vista que capturam as mesmas informações, mas recebem nomes diferentes e são por esse motivo descritos separadamente, como é o caso do ponto de vista de informações do sistema e o ponto de vista conceitual. É possível notar também que não há um consenso entre os diferentes estudos considerados sobre quais visões são mais adequadas para representar arquiteturas de referência, em especial aquelas propostas para o domínio de sistemas embarcados.

Tabela 3.1: Fontes de pontos de vista para representação de arquiteturas de software e arquiteturas de referência

Número	Fonte
1	Visão proposta por Kruchten (1995)
2	Visão utilizada por NASA ESDS Working Group (2012)
3	Visão utilizada por Nakagawa e Maldonado (2008)
4	Visão utilizada por Weyns (2006)
5	Visão proposta por Hofmeister et al. (2000)

Tabela 3.2: Pontos de vista para a representação de arquiteturas de software e arquiteturas de referência

Ponto de Vista	**Fonte**	**Descrição**
Casos de Uso	1, 5	Apresenta em um alto nível de abstração as principais funcionalidades da arquitetura e a interação entre os diferentes atores (sistemas, usuários).
Funcionalidades	2	Identifica em um alto nível de abstração as principais funcionalidades oferecidas pelos sistemas que implementam a arquitetura de referência, seus relacionamentos e possíveis consumidores, sejam eles outras funções, sistemas ou pessoas. Pode incluir uma descrição do fluxo lógico de ações compreendidas em cada função.
Conceitual	3 5	Apresenta a estrutura e o conteúdo das informações gerenciadas dentro da arquitetura referência. Um glossário pode ser utilizado para descrever cada um dos termos existentes na arquitetura sendo, opcionalmente, substituído por uma ontologia do domínio.
Arquitetura da Informação	2	Apresenta a estrutura e o conteúdo das informações gerenciadas dentro da arquitetura de referência.
Visão de Serviços do Sistema	2	Identifica em um alto nível de asbtração o conjunto minimal de serviços que devem ser apoiados por todas as implementações da arquitetura de referência, uma informação essencial para garantir a *interoperabilidade* entre as diferentes instâncias arquiteturais. Pode apresentar também, por meio de tabelas, a relação existente entre os serviços e os requisitos do sistema (casos de uso) ou entre os serviços e informações do sistema (apenas aquelas que apoiam o serviço em questão)
Componentes Colaborativos	4	Descreve a arquitetura de referência como um conjunto de componentes em tempo de execução interagindo e utilizando um conjunto de repositórios de dados compartilhados para executar as funcionalidades requisitadas. Também pode descrever portas e interfaces fornecidas e requeridas por cada componente.
Dados Compartilhados	4	Descreve como a arquitetura de referência é estruturada em componentes que leem e escrevem informações em vários repositórios de dados compartilhados. Descreve portas, interfaces. Os repositórios de dados atuam como mediadores da interação entre diferentes módulos acessando o seu conteúdo. Nesse sentido, o repositório também pode apoiar concorrência e persistência.

Continua...

Tabela 3.2 – Continuação

Ponto de Vista	**Fonte**	**Descrição**
Processos	1	Descreve a arquitetura em termos de processos e linhas de execução da aplicação, indicando responsabilidades, colaborações e a alocação de elementos lógicos, e pode apresentar requisitos não funcionais, tais como o desempenho e a disponibilidade.
Processos Comunicativos	4	Descreve a arquitetura de referência como um conjunto de unidades executando concorrentemente e suas interações. Unidades concorrentes são abstrações de elementos de software mais concretos tais como tarefas, processos e linhas de execução. Também considera os repositórios de dados acessados por essas unidades, mensagens trocadas e interfaces expostas.
Tempo de Execução	3	Apresenta a estrutura do sistema de software quando este está sendo executado. Descreve os componentes presentes em tempo de execução, fluxo de dados e conectores.
Desenvolvimento	1	Descreve a decomposição da arquitetura em subsistemas e foca na organização real dos módulos de software no ambiente de desenvolvimento.
Implantação	3	Descreve as máquinas, os softwares instalados nessas máquinas e conexões de rede utilizadas pelos sistemas de software.
Física	1	Descreve o mapeamento do software no hardware e considera, essencialmente, os requisitos não funcionais do sistema como confiabilidade, desempenho e escalabilidade.
Tecnologia	2	Apresenta requisitos para os padrões e/ou ferramentas comumente utilizados dentro de instâncias da arquitetura de referência com o objetivo de guiar a seleção de tecnologias que sejam adequadas e/ou compatíveis aos propósitos da arquitetura.
Módulos	3, 4, 5	Apresenta os principais módulos funcionais, o fluxo de dados entre esses módulos e as principais interfaces.
Lógica	1	Descreve a decomposição do sistema em módulos e apresenta essencialmente os requisitos funcionais desse sistema. A arquitetura é representada em termos de camadas, subsistemas, pacotes, *frameworks*, classes e interfaces mais importantes.

Visando compreender melhor quais são os elementos necessários para representar adequadamente arquiteturas de referência para o domínio de sistemas embarcados, a aluna conduziu uma revisão sistemática (Guessi et al., 2012), uma técnica proposta pela Engenharia de Software Baseada em Evidências (ESBE) (Dybå et al., 2005) que permite

sumarizar, acessar e interpretar todas as evidências[3] que sejam relevantes para um determinado tópico de pesquisa. A revisão foi conduzida nos meses de outubro de 2011 à janeiro de 2012 e seguiu o processo resumido em três etapas (Kitchenham, 2004): planejamento, condução e documentação. A revisão teve como finalidade responder quatro Questões de Pesquisa (QP):

QP 1: Como arquiteturas para o domínio de sistemas embarcados têm sido representadas?

A QP 1 investigou como arquiteturas para o domínio de sistemas embarcados têm sido descritas atualmente identificando os níveis de formalismo que têm sido adotados.

QP 2: Quais tipos de informação têm sido apresentados na representação de arquiteturas para o domínio de sistemas embarcados?

A QP2 investigou os tipos informações apresentados na descrição dessas arquiteturas como, por exemplo, os interesses considerados e os elementos associados a eles. Com base na resposta obtida para essa pergunta, as linguagens de descrição arquiteturais podem ser analisadas com base nas informações que precisam ser apresentadas na representação de arquiteturas desse domínio.

QP 3: Quais linguagens de descrição arquitetural têm sido adotadas para descrever arquiteturas do domínio de sistemas embarcados?

A identificação das linguagens de descrição arquitetural utilizadas na resposta da QP3 serve como base para a investigação de vantagens e limitações de cada linguagem para a descrição de arquiteturas do domínio de sistemas embarcados.

QP 4: Quais pontos de vista têm sido adotados na construção da descrição arquitetural de arquiteturas do domínio de sistemas embarcados?

A resposta para a QP4 revela quais pontos de vista são mais relevantes para construir a descrição de arquiteturas do domínio de sistemas embarcados, considerando-se os diferentes níveis de abstração que precisam ser representados.

A revisão sistemática considerou como fontes de pesquisa as bases ACM Digital Library[4], IEEEXplore[5], ISI Web of Knowledge[6], ScienceDirect[7], Scopus[8] e SpringerLink[9].

[3]Entende-se por evidência qualquer resultado obtido em um trabalho científico publicado em veículos de publicação tais como periódicos, anais de eventos, livros, relatórios técnicos, dentre outros.

[4]`http://portal.acm.org`

[5]`http://ieeexplore.ieee.org`

[6]`http://apps.isiknowledge.com`

[7]`http://www.sciencedirect.com`

[8]`http://www.scopus.com`

[9]`http://www.springerlink.com`

Como resultado da busca nas bases, um total de 248 estudos primários foram recuperados. No entanto, após a aplicação dos critérios de seleção, apenas 24 estudos primários foram considerados relevantes para responder as questões de pesquisa estabelecidas. É importante ressaltar que arquiteturas de software e modelos de referência também foram incluídos no escopo da revisão sistemática para permitir que fosse feita a comparação entre a descrição de arquiteturas de software, modelos de referência e arquiteturas de referência desse domínio. Os resultados da pesquisa são discutidos a seguir.

A maioria das arquiteturas de software do domínio de sistemas embarcados investigadas, mais especificamente 50% (isso é, 10 das 20 arquiteturas de software analisadas), descreveram a arquitetura formalmente ou utilizando uma combinação de técnicas formais e semi-formais. O uso de uma descrição arquitetural formal tem a vantagem de permitir a geração automática de modelos que também podem ser verificados e validados automaticamente com relação aos requisitos pretendidos para a arquitetura por meio de ferramentas de apoio especiais. Uma vez que o arquiteto de software tenha esses mecanismos disponíveis, será mais fácil avaliar as implicações de diferentes configurações da arquitetura de modo a escolher a que melhor atende aos requisitos. Desse modo, há uma grande possibilidade de evitar o retrabalhado por conta de uma decisão arquitetural ruim, uma vez que erros podem ser detectados ainda em um estágio inicial do ciclo de vida do software. Em especial, no caso de sistemas embarcados críticos de segurança, o uso de formalismos na descrição arquitetural é essencial para garantir a descrição rigorosa e não ambígua de requisitos que são fundamentais para seu funcionamento adequado como, por exemplo, tolerância a falhas e confiabilidade. No que se refere à descrição de arquiteturas de referência, a pesquisa revelou que elas são descritas utilizando-se técnicas semi-formais ou uma combinação de técnicas semi-formais e informais. Segundo Angelov et al. (2009), o uso de técnicas semi-formais é preferível quando é preciso que a arquitetura de referência seja facilmente compreendida por uma variedade maior de interessados, mas ainda ofereça uma especificação clara de seus requisitos. No entanto, o uso de formalismo na descrição de arquiteturas de referência não deve ser descartado, já que eles podem ser úteis para fornecer evidências sobre as qualidades e benefícios que podem ser alcançados pela adoção dessas arquiteturas. Vale destacar também que a escolha por um determinado tipo de abordagem para a descrição de arquiteturas de referência deve ser baseada no objetivo pretendido por essa arquitetura, ou seja, padronização ou mecanismo facilitador, nos tipos de organizações de interessados (técnicos ou não técnicos) e nos interesses que precisam ser capturados. Em particular, com relação à representação de modelos de referência, observa-se que eles são representados apenas informalmente, já que eles apresentam um nível de abstração mais elevado que arquiteturas de referência. A Figura 3.2 sumariza

os resultados encontrados sobre as abordagens utilizadas na descrição de arquiteturas do domínio de sistemas embarcados.

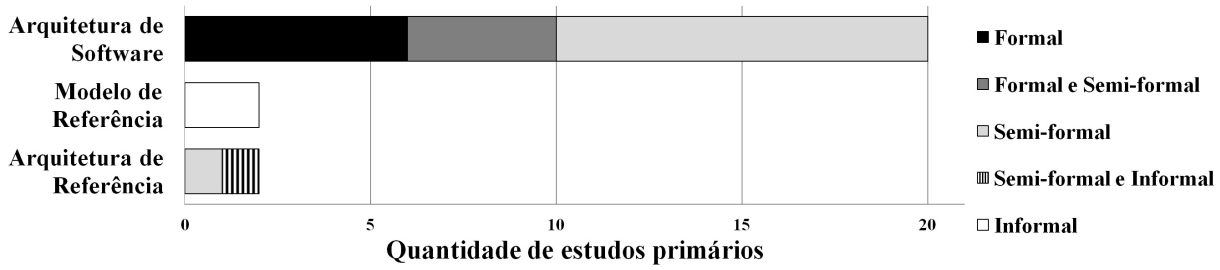

Figura 3.2: Abordagens utilizadas para representação de arquiteturas de sistemas embarcados (Guessi et al., 2012)

Em uma outra perspectiva, é importante identificar os interesses existentes nas arquiteturas do domínio de sistemas embarcados, bem como os tipos de informação que são usualmente associados a cada um deles. Para isso, a revisão sistemática selecionou os interesses que apareceram em pelo menos dois estudos primários e associou com os tipos de informações apresentadas nas descrições arquiteturais. A Tabela 3.3 apresenta os interesses selecionados pela revisão e as informações associadas. Por exemplo, os estudos relacionados com a adaptabilidade da arquitetura apresentaram as seguintes informações: componentes, interações, interfaces, sincronização e temporização. Nesse sentido, a relação apresentada oferece subsídios importantes para a definição dos elementos necessários para a descrição de cada interesse e também orienta a identificação dos pontos de vista mais adequados. Com base nas informações apresentadas, verifica-se que sistemas embarcados precisam atender uma grande variedade de restrições e requisitos, sendo em especial a segurança o interesse encontrado com maior frequência. Nesse contexto, é importante existir meios suficientes para capturar esse interesse na descrição de arquiteturas desse domínio. Em seguida, os interesses que ainda não foram abordados ou descritos neste projeto são discutidos.

O conceito chave para a adaptabilidade de arquiteturas de software é permitir o ajuste dessas arquiteturas de acordo com mudanças no ambiente, especialmente no caso de sistemas embarcados em robôs e veículos não tripulados. Para que isso seja possível, em primeiro lugar, é preciso que a descrição do componente especifique o modo como ele interage com outros componentes e quais serviços são requisitados e fornecidos por ele (Borde et al., 2009). Com base nessas informações, Khalgui et al. (2009) sugere então a definição de um sistema de controle, obtido pela existência de restrições de precedência entre os componentes, para controlar a execução do software de acordo com ordens de controle bem definidas. Tal sistema de controle é responsável por checar e interpretar a evolução do ambiente, por meio da leitura de dados pelos sensores, antes da tomada de uma decisão como, por exemplo, ativar atuadores no equipamento. Isso implica em uma maior

Tabela 3.3: Interesses e tipos de informação associados

Interesse	**Informações**
adaptabilidade	componentes, interações, interfaces, sincronização, temporização
exatidão	componentes, configuração, restrições, fluxo de dados, interações, processes, sincronização, *threads*
confiabilidade	componentes, configuração, conectores, interfaces, temporização
tolerância a falhas	componentes, conectores, fluxo de dados, interfaces, temporização
interoperabilidade	componentes, fluxo de dados, interações, interfaces, concorrência, sincronização
reúso do conhecimento	camadas de abstração, conceitos, decisões arquiteturais
manutenibilidade	componentes, concorrência, eventos, interações, interfaces, estados
desempenho	componentes, conectores, fluxo de dados, eventos, interações, interfaces
consumo de energia	fluxo de dados
confiança	concorrência, restrições, eventos, interações, estados
segurança	componentes, conectores, restrições, fluxo de dados, interações, interfaces, sincronização, temporização
temporização	comportamentos, configuração, restrições, fluxo de dados, processos, sincronização, *threads*

capacidade de processamento para permitir a análise desses dados em tempo-real. Nesse sentido, sempre que uma mudança ocorrer em tempo de execução, o sistema deve estar apto a adaptar suas funcionalidades para garantir a segurança de todo o sistema. Para que isso seja possível, contudo, deve haver uma capacidade de processamento suficiente para permitir os cálculos e ações em tempo-real. Além disso, observa-se uma relação próxima entre os interesses de segurança, confiabilidade, dependabilidade e adaptabilidade.

A exatidão das descrições arquiteturais, ou seja, a descrição clara e precisa da arquitetura ajuda a identificar os componentes que estão ativos em um determinado instante de tempo e pode oferecer informações importantes sobre o comportamento esperado desses componentes durante o projeto deles. Nesse sentido, a temporização das informações é essencial para afirmar precisamente qual o comportamento do sistema em um dado instante de tempo. Para que isso seja possível, Bernardo et al. (2001) sugerem a utilização de técnicas formais que possibilitam a descrição detalhada da estrutura do sistema, além de possibilitar a análise de propriedades ainda no nível arquitetural, o que permite detectar a existência de erros em uma etapa do desenvolvimento em que correções são ainda relativamente baratas e abrir espaço para otimizações no projeto.

Considerando-se em especial o caso de aplicações móveis, têm-se que o baixo consumo de energia é um requisito essencial. Em particular, Tan et al. (2003) apontam dois aspectos considerados no nível arquitetural que tem grande influência no consumo de energia final em sistemas embarcados: (i) seleção do estilo arquitetural, uma vez que cada estilo fornece

trocas interessantes entre eficiência, portabilidade, manutenibilidade, dentre outras; e (ii) projeto de arquiteturas concretas a partir do *framework* fornecido pelo estilo arquitetural. Observa-se então que o consumo de energia final apresentado depende também de como a arquitetura concreta do sistema de software é implementada. Por exemplo, podem ser aplicadas otimizações no compilador que reduzem o consumo de energia.

Com relação à descrição das arquiteturas de referência apresentadas na Seção 3.3.1 e Seção 3.3.2, observa-se que a interoperabilidade é um dos principais interesses já que ela vai permitir que as arquiteturas concretas derivadas dessas arquiteturas de referência sejam compatíveis entre si. Além disso, o reúso do conhecimento arquitetural é também um importante interesse dessas arquiteturas, sendo um pré-requisito para melhorar o desenvolvimento de sistemas de software desse domínio já que contribui para o projeto mais eficiente desses sistemas. Nesse sentido, é importante disponibilizar mecanismos suficientes para capturar as fundamentações das decisões arquiteturais como parte integrante da descrição arquitetural com o objetivo de reduzir os esforços necessários para criação a derivação de arquiteturas concretas do domínio. Em especial, Biehl e Törngren (2010) propõem que essas decisões arquiteturais sejam capturadas como regras gerais, possivelmente organizadas em bibliotecas, que uma vez criadas possam ser aplicadas em diferentes locais da arquitetura concreta. Desse modo, é possível reduzir ainda mais o esforço necessário para a documentação dessas decisões. Em especial, a existência de um repositório com as decisões arquiteturais formais e informais tomadas pelo arquiteto de software melhoram o processo criativo, uma vez que podem ser utilizadas para explorar novas soluções, medir as consequências de uma determinada decisão, além de servir como material de estudo para arquitetos em treinamento (Mesman et al., 2002).

A revisão sistemática também levantou quais são as linguagens de descrição arquitetural utilizadas atualmente para descrição de arquiteturas do domínio de sistemas embarcados (ver Figura 3.3). Da Figura, é possível identificar as linguagens AADL (SAE, 2011) e UML(OMG, 2010b) como sendo as principais linguagens empregadas. A UML é a linguagem de descrição arquitetural mais comumente utilizada por ser, geralmente, melhor compreendida por todos os interessados e por existirem diversas ferramentas de apoio a linguagem em diferentes etapas do desenvolvimento de software. Já a AADL é uma linguagem especificamente desenvolvida para o domínio de sistemas embarcados. Além dessas linguagens já mencionadas, também foram encontradas na revisão sistemática outras linguagens semi-formais, sendo essas: ACOL (Langsweirdt et al., 2010), REAL (Gilles e Hugues, 2010), Acme, CCM (*Common Object Request Broker Architecture Component Model*) (OMG, 2011), EAST-ADL (*Electronic Architecture and Software Tools ADL*) (ATESST, 2010), SysML (OMG, 2010a), UMT-RT (*UML Profile for Real-time Systems*), MARTE (*UML Profile for Modeling and Analysis of Real-time and Embedded Systems*)

(OMG, 2009) e SAG (*Software Architecture Graph*) (Tan et al., 2003). Adicionalmente, também foram encontradas várias linguagens de descrição arquitetural formais, sendo essas: MetaH (*Avionics Architecture Description Language*) (Honeywell, 2000), FSM (*Finite State Machines*), SDL (*Specification and Description Language*) (ITU, 1994), NCES (*Net Condition/Event Systems*), ASDL (*Architecture Structure Description Language*) (Eixelsberger e Gall, 1998), DRTSADL (*Distributed/Embedded Real-Time System ADL*) (Ye et al., 2008) e PADL (*Process Algebraic Architectural Description Language*) (Bernardo et al., 2001). Por fim, além das linguagens já mencionadas, também observou-se a adoção de abordagens informais, principalmente para a descrição de modelos de referência. Mas apesar da variedade de linguagens encontradas pela revisão sistemática, poucas delas são de fato adotadas na indústria como, por exemplo, AADL, UML, SysML e EAST-ADL. A participação ativa de importantes organizações industriais na criação dessas linguagens pode ser apontado como um fator determinante para a aceitação dessas linguagens pela indústria. Em vista da variedade de abordagens formais, semi-formais e informais que podem ser adotadas, o uso de um conjunto de linguagens de descrição arquitetural compatíveis entre si pode ser a resposta para melhorar a consistência entre diferentes níveis de abstração de uma arquitetura. Para que isso seja possível, contudo, é preciso que existam mecanismos confiáveis para integrar as diferentes linguagens.

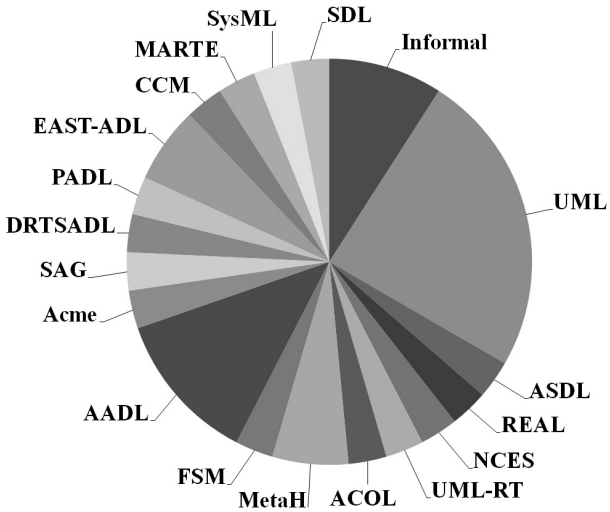

Figura 3.3: Linguagens de descrição arquitetural utilizadas na representação de arquiteturas de sistemas embarcados(Guessi et al., 2012)

Por fim, com relação aos pontos de vista adotados na construção da descrição arquitetural, é importante ressaltar que alguns estudos não utilizaram uma nomenclatura padrão. Considerando-se o fato de que a UML é a linguagem de descrição arquitetural mais comumente encontrada nos estudos analisados pela revisão sistemática e também a relevância dessa linguagem para o domínio de arquitetura de software, os pontos de vista encontrados foram separados de acordo com as categorias de diagramas estabeleci-

das pela UML, sendo essas (OMG, 2010b): (i) pontos de vista estruturais, que descrevem uma visão estática da arquitetura; (ii) pontos de vista comportamentais, que descrevem o comportamento da arquitetura de modo geral; e (iii) pontos de vista de interação, derivadas da categoria anterior, mas que adicionam mais detalhes aos aspectos de interação. Além dessas, um quarto ponto de vista, chamado ponto de vista conceitual, também foi considerado para classificar aqueles casos em que apenas uma descrição dos conceitos da arquitetura ou vocabulário é apresentado. A revisão verificou que os modelos de referência para o domínio de sistemas embarcados descrevem apenas uma visão conceitual da arquitetura. Já as arquiteturas de referência analisadas apresentam visões estruturais e/ou comportamentais para destacar funcionalidades e recursos importantes. Em especial, o nível de detalhamento considerado na definição desses pontos de vista pode variar de acordo com os objetivos da arquitetura de referência. Assim, se o principal objetivo de uma arquitetura de referência é garantir a interoperabilidade entre as instâncias arquiteturais, então diretrizes suficientes devem ser disponibilizadas para que todas as instâncias sigam o mesmo padrão. Por exemplo, interfaces chave da AUTOSAR foram descritas concretamente, enquanto o modelo de informação apresentado em ESDS ajuda a definir um vocabulário comum a ser seguido entre as instâncias arquiteturais. Nesse sentido, o uso e o apoio que se espera da arquitetura referência influenciam na escolha dos pontos de vista a serem construídos. No caso de arquiteturas de referências, em particular, os pontos de vista precisam ser cuidadosamente analisados já que o excesso de documentação pode prejudicar e até mesmo diminuir a aplicabilidade dessas arquiteturas. Por outro lado, não existem restrições para os pontos de vista utilizados na descrição de arquiteturas de software do domínio de sistemas embarcados, devendo ser escolhidos aqueles que melhor satisfizerem os tipos de informação que se deseja descrever. Em particular, destaca-se a adoção de pontos de vista estruturais (70%, isso é, 14 das 20 arquiteturas de software analisadas), seguidos por pontos de vista de interação (35%, isso é, sete das 20 arquiteturas de software analisadas), pontos de vista comportamentais (30%, isso é, seis das 20 arquiteturas de software analisadas) e pontos de vista conceituais (5%, isso é, uma das 20 arquiteturas de software analisadas).

De modo geral, verifica-se que a descrição adequada de arquiteturas de software e arquiteturas de referência é essencial, principalmente para o domínio de sistemas embarcados. No entanto, observa-se a falta de consenso sobre quais linguagens de descrição arquitetural são mais apropriadas para representar arquiteturas de referência desse domínio ou, ainda, quais pontos de vista podem ser adotados para capturar adequadamente os interesses dessas arquiteturas. Além disso, a definição do nível de detalhes da descrição de arquiteturas de referência também precisa ser investigado. Em especial, parece relevante a adoção de um conjunto de linguagens complementares para a descrição de arquiteturas de

referência do domínio de sistemas embarcados considerando, possivelmente, os diferentes níveis de abstração que podem ser necessários.

3.5 Considerações Finais

Neste capítulo foi introduzida a teoria básica sobre sistemas embarcados necessária para o desenvolvimento dessa dissertação. Em seguida, três importantes exemplos de arquiteturas de referência para o domínio foram apresentados a fim de ilustrar alguns dos benefícios que podem ser alcançados com a adoção dessas arquiteturas. Em seguida, a descrição de arquiteturas de software, arquiteturas de referência e modelos de referência para o domínio de sistemas embarcados foram investigadas e importantes características foram identificadas. No próximo capítulo, o método proposto para representação de arquiteturas de referência para o domínio de sistemas embarcados é descrito.

Capítulo

4

ProSA-Re: Um Método para a Representação de Arquiteturas de Referência de Sistemas Embarcados

4.1 Considerações Iniciais

Em vista da importância e também da complexidade dos sistemas embarcados na atualidade, observa-se que a proposição e utilização de arquiteturas de referência para sistemas desse domínio pode ser uma importante contribuição para a promoção e comunicação do conhecimento arquitetural contido nessas arquiteturas. Em especial, representar adequadamente a arquitetura de referência, ou seja, criar uma descrição que seja capaz de capturar os interesses mais importantes e que também seja de fácil compreensão para os seus interessados, é fundamental para que ela seja, de fato, empregada para o desenvolvimento de arquiteturas de software. Nesse contexto, o método ProSA-Re é proposto visando sistematizar a representação de arquiteturas de referência para o domínio de sistemas embarcados.

Este capítulo é organizado da seguinte forma. Na Seção 4.2, os elementos que compõem o ProSA-Re são detalhados e o fluxo de atividades para a elaboração da representação arquitetural é apresentado. Na sequência, na Seção 4.3, são apresentadas as considerações finais deste capítulo.

4.2 Descrição do ProSA-Re

O método ProSA-Re tem como objetivo apoiar o projeto e representação de arquiteturas de referência para o domínio de sistemas embarcados. O foco no domínio de sistemas embarcados visa capturar particularidades desse domínio para a elaboração de descrições arquiteturais que sejam: (i) abrangentes, ou seja, capazes de atender as necessidades de uma grande variedade de interessados; (ii) compactas, no sentido de que viabilizem a fácil manutenção, rastreabilidade e evolução do conjunto de visões que descrevem a arquitetura; e (iii) padronizada, para promover a comunicação e o entendimento homogêneo das informações apresentadas.

Foi utilizada a linguagem SPEM (do inglês, *Software and Systems Process Engineering Meta-Model Specification*) (OMG, 2008) para a descrição do método, considerando a relevância e a maturidade dessa linguagem para a definição de métodos e processos de software. De modo geral, essa linguagem define quatro elementos principais para a descrição de métodos e processos, sendo esses: (i) tarefas, que definem um trabalho sendo executado com um objetivo específico; (ii) papéis, que definem os indivíduos envolvidos na execução dessas tarefas; (iii) produtos de trabalho, que definem os artefatos utilizados como entrada e saída nessas tarefas; e (iv) diretrizes, que definem os artefatos que apoiam a execução dessas tarefas e a criação dos produtos de trabalho. É importante destacar que a linguagem também permite a definição de diferentes estados para cada produto de trabalho de modo a permitir a representação da evolução desse produto durante a execução do método. O conceito de atividade também é introduzido pela linguagem SPEM para agrupar e sistematizar um conjunto de tarefas relacionadas. O Anexo B apresenta mais informações sobre os elementos dessa linguagem. Nesse contexto, o ProSA-Re compreende quatro atividades principais ilustradas na Figura 4.1. As atividades são brevemente introduzidas a seguir:

- *Definição do escopo* tem como finalidade identificar os interessados na arquitetura de referência e seus respectivos interesses;
- *Elaboração das visões arquiteturais* tem como finalidade selecionar o conjunto apropriado de pontos de vista para representar a arquitetura de referência de sistemas embarcados e criar o respectivo conjunto de visões arquiteturais;
- *Representação das decisões arquiteturais* deve ser realizada em conjunto com as duas atividades anteriores e tem como finalidade documentar as decisões arquiteturais; e
- *Avaliação da representação* tem como finalidade mesclar os documentos produzidos pelo método e submeter a representação a uma avaliação. O objetivo dessa atividade é melhorar a qualidade dos documentos criados.

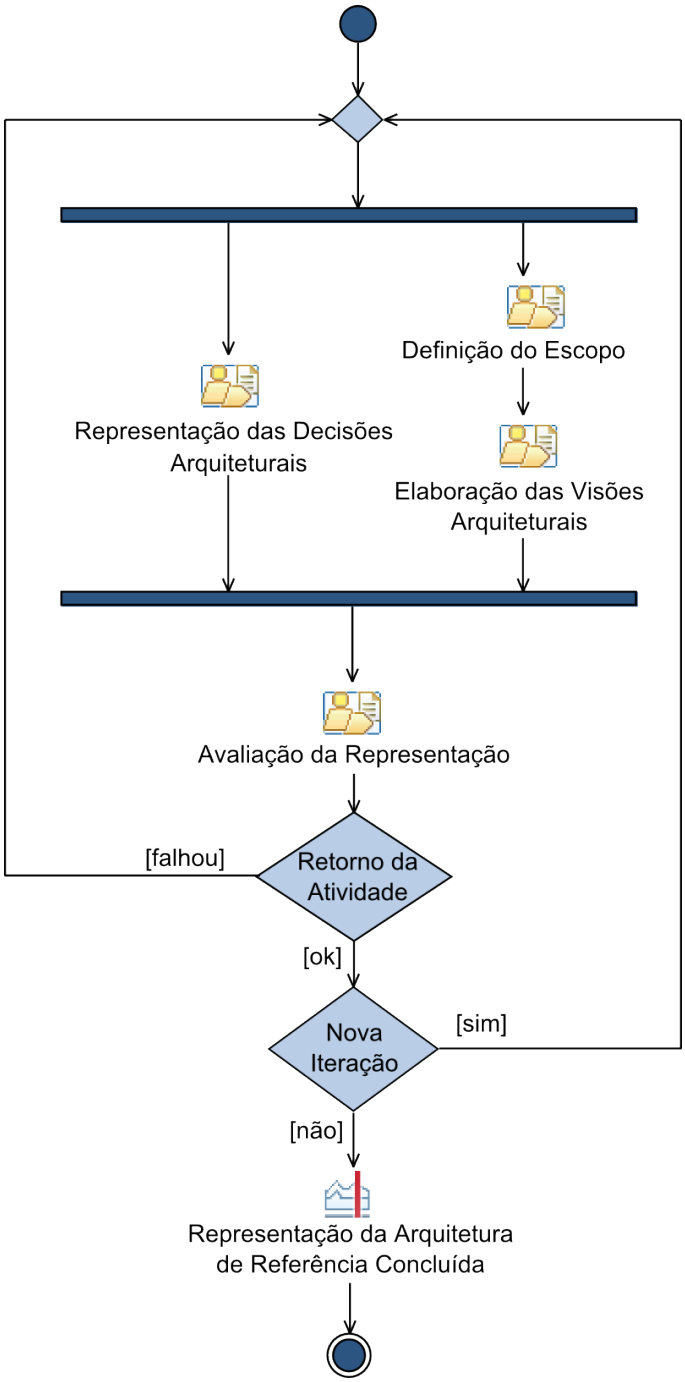

Figura 4.1: Fluxo de trabalho do ProSA-Re

A principal vantagem para a realização de uma avaliação ainda na fase de representação da arquitetura de referência é a garantia de que erros não sejam propagados para iterações subsequentes. No entanto, não é objetivo dessa atividade sobrecarregar a equipe com uma avaliação detalhada e dispendiosa, podendo ser substituída por uma avaliação de alto nível da representação arquitetural. Em especial, o esforço necessário para executar essa avaliação pode ser minimizado com o uso de ADLs que permitam a verificação automática das visões arquiteturais construídas. De acordo com o resultado da avaliação da

representação arquitetural, um dos seguintes fluxos de trabalho pode ser tomado: no caso de falha, será preciso revisar as atividades anteriores para corrigir os problemas encontrados; caso contrário, será preciso decidir então se uma nova iteração do método é necessária para, por exemplo, adicionar novas visões ou refinar o conjunto de visões já construídas, ou se ela é suficiente para alcançar os propósitos das arquitetura de referência. Em ambos os casos, as decisões são tomadas em reuniões de acompanhamento que reunem os principais envolvidos na elaboração da arquitetura de referência. O objetivo dessas reuniões é, portanto, garantir que a representação esteja de acordo com os objetivos da arquitetura de referência, além de manter a equipe de projeto e representação da arquitetura de referência sincronizada. Quando a representação da arquitetura de referência for considerada satisfatória, o processo arquitetural poderá avançar para a próxima fase do ProSA-RAES. Nas subseções a seguir, são apresentados os produtos de trabalho utilizados no ProSA-Re (Seção 4.2.1), os papéis envolvidos (Seção 4.2.2), as tarefas propostas (Seção 4.2.3) e as diretrizes para guiar a execução do método (Seção 4.2.4).

4.2.1 Produtos de Trabalho Utilizados

SPEM distingue o produto de trabalho em artefato (que se trata de um produto de trabalho tangível) e resultado final (que se trata de uma agregação de outros artefatos que pode ser disponibilizado para terceiros). Durante a execução do ProSA-Re, um total de 13 produtos de trabalho podem ser consumidos e/ou produzidos, dentre eles a lista de interessados na arquitetura de referência, a lista de interesses, a lista de pontos de vista que podem ser selecionados, além da representação das respectivas visões arquiteturais. É importante destacar que a quantidade real de produtos de trabalho consumidos/produzidos pelo método varia de acordo com os pontos de vista selecionados, de modo que apenas um subconjunto dessas representações serão, de fato, elaboradas. Uma breve descrição de cada um deles é apresentada a seguir.

- **i) Artefatos**
  - *Lista de interessados* é o conjunto de pessoas, organizações ou grupos que possuem interesse em uma arquitetura de referência de sistemas embarcados. Uma vez que as arquiteturas de referência possuem uma grande variedade de interessados, é relevante considerar apenas aqueles diretamente relacionados ao projeto da arquitetura de referência ou que sejam representativos do domínio para o qual ela está sendo proposta. Durante a execução do ProSA-Re, esse artefato pode assumir um dos seguintes estados: *padrão* (para destacar que é a lista fornecida originalmente pelo método) ou *modificado* (para destacar que a lista foi adaptada);

- *Lista de interesses* é o conjunto de interesses dos interessados na arquitetura de referência a ser proposta. A lista padrão fornecida pelo método é resultado da revisão sistemática apresentada na Seção 3.4 e inclui: *confiabilidade*, *desempenho*, *disponibilidade*, *escalabilidade*, *interoperabilidade*, *manutenibilidade*, *proteção*, *segurança*, *usabilidade* e *variabilidade*. A lista de interesses pode ser alterada pela execução do ProSA-Re, de modo a capturar os interesses específicos da arquitetura de referência a ser elaborada. Nesse sentido, durante a execução do ProSA-Re, esse artefato pode assumir um dos seguintes estados: *padrão* (para indicar que é a lista fornecida originalmente pelo método) ou *modificado* (para indicar que a lista foi adaptada);
- *Lista de Pontos de Vista* é o conjunto de pontos de vista candidatos que podem ser empregados para a descrição das arquiteturas de referência. O ProSA-Re propõe um conjunto de 14 pontos de vista classificados em quatro grupos. A Tabela 4.1 descreve cada ponto de vista. Durante a execução do ProSA-Re, esse artefato pode assumir um dos seguintes estados: *padrão* (para indicar que é a lista fornecida originalmente pelo método) ou*modificado* (para indicar que a lista foi adaptada);
- *Relatório de Inconsistências* é o documento que registra todos os problemas conhecidos da representação da arquitetura de referência e pode ser utilizado como guia para a revisão das atividades;
- *Representação da Visão Cronológica de Decisões* é o documento que descreve a visão cronológica de decisões;
- *Representação da Visão de Detalhe das Decisões* é o documento que descreve a visão de detalhe das decisões;
- *Representação da Visão de Interessados nas Decisões* é o documento que descreve a visão de interessados nas decisões;
- *Representação da Visão de Relacionamentos entre Decisões* é o documento que descreve a visão de relacionamentos entre decisões;
- *Representação das Visões de Implantação* é o documento que descreve visões de implantação da arquitetura de referência;
- *Representação das Visões de Código Fonte* é o documento que descreve visões de código fonte da arquitetura de referência;
- *Representação das Visões em Tempo de Execução* é o documento que descreve visões em tempo de execução da arquitetura de referência; e
- *Representação das Visões Transversais* é o documento que descreve visões transversais da arquitetura de referência.

- **ii) Resultado Final**
  - *Representação da Arquitetura de Referência* consiste no principal produto de trabalho produzido pelo ProSA-Re e é utilizado para expressar as arquiteturas de referência. Durante a execução do ProSA-Re, esse artefato assume um dos seguintes estados: *aprovação pendente*, *aprovado* ou *reprovado*.

Tabela 4.1: Descrição dos pontos de vista

Ponto de vista	**Descrição**
	Grupo 1: Pontos de vista transversais
Funcionalidades	Identifica, em um alto nível de abstração, quais são as principais funcionalidades, seus relacionamentos e possíveis consumidores, por exemplo, funções, sistemas ou pessoas. Um fluxo lógico de ações também pode ser acrescentado para detalhar cada funcionalidade.
Conceitual	Apresenta a estrutura e (opcionalmente o conteúdo) das informações gerenciadas pela arquitetura de referência. Um glossário pode ser utilizado para descrever cada um dos termos utilizados sendo, opcionalmente, substituído por uma ontologia do domínio.
Serviços do sistema	Identifica em um alto nível de abstração o conjunto minimal de serviços —funções— que precisam ser apoiados por todas as implementações da arquitetura de referência. Essa informação é útil para garantir a interoperabilidade entre diferentes instâncias arquiteturais. Opcionalmente, podem ser utilizadas tabelas para descrever a relação entre os serviços e requisitos arquiteturais, ou entre os serviços e os conceitos do sistema que apoiam o serviço em questão.
Variabilidade	Descreve quais são e como os elementos de uma arquitetura de referência podem ser exercitados para a configuração de instâncias arquiteturais. Esse ponto de vista pode destacar, por exemplo, alternativas (por meio de relacionamentos ou-exclusivo), elementos opcionais e elementos abstratos, ou seja, que precisam ser concretizados.
	Grupo 2: Pontos de vista em tempo de execução
Componentes colaborativos	Descreve a arquitetura de referência como um conjunto de componentes interativos que utilizam um conjunto de repositórios de dados compartilhados para executar funcionalidades em tempo de execução. Nesse ponto de vista podem ser descritas portas e interfaces fornecidas e requeridas por cada componente além do fluxo de dados entre eles.

Continua...

Tabela 4.1 – Continuação

Ponto de vista	**Descrição**
Dados compartilhados	Descreve como a arquitetura de referência é estruturada em componentes que leem e escrevem por meio de portas e interfaces informações em vários repositórios de dados compartilhados. Nesse contexto, os repositórios atuam como mediadores da interação entre diferentes módulos acessando o seu conteúdo.
Processos	Descreve a arquitetura de referência como um conjunto de unidades concorrentes de execução — abstrações de elementos de software mais concretos, tais como tarefas, processos e linhas de execução — e suas interações. Também mostra os repositórios de dados acessados por essas unidades, troca de mensagens e interfaces expostas.
Grupo 3: Pontos de vista de implantação	
Desenvolvimento	Descreve a decomposição da arquitetura de referência em subsistemas com foco na organização real dos módulos de software no ambiente de desenvolvimento. Esse ponto de vista serve de base para a distribuição de trabalho entre equipes de desenvolvimento, avaliação de custo e planejamento, portabilidade e segurança.
Implantação	Descreve as máquinas, os softwares instalados nessas máquinas e as conexões de rede utilizadas em instâncias da arquitetura de referência.
Físico	Descreve o mapeamento do software no hardware. Esse ponto de vista enfatiza os requisitos não funcionais da arquitetura de referência como confiabilidade, desempenho e escalabilidade.
Tecnologia	Apresenta requisitos para os padrões e/ou ferramentas comumente utilizadas em instâncias arquiteturais com a finalidade de guiar a seleção de tecnologias compatíveis e/ou adequadas aos objetivos da arquitetura de referência.
Grupo 4: Ponto de vista de código fonte	
Estrutural	Delimita o contexto de implantação (partes de hardware e software disponíveis, além de outros recursos externos com os quais a arquitetura de referência precisa interagir) do contexto da aplicação que será desenvolvida. Esse ponto de vista também mostra a interação com o ambiente e pode conter ainda um breve descrição de cada módulo em que são detalhados seus recursos.
Módulos	Apresenta os principais módulos funcionais, o fluxo de dados entre esses módulos e as principais interfaces.

Continua...

Tabela 4.1 – Continuação

Ponto de vista	**Descrição**
Lógico	Descreve a decomposição em módulos da arquitetura de referência e apresenta, essencialmente, os requisitos funcionais dessa arquitetura. A arquitetura de referência pode ser descrita em termos de camadas, subsistemas, pacotes, *frameworks*, classes e interfaces.

4.2.2 Papéis Considerados

A definição de um papel descreve a pessoa ou organização que participa da execução de um determinado conjunto de tarefas e que também é responsável por um conjunto de produtos de trabalho. O ProSA-Re identifica três papéis como sendo os mais relevantes para o projeto e descrição de arquiteturas de referência de sistemas embarcados, sendo apresentados a seguir:

Arquiteto funcional: é o arquiteto ou o interessado com extenso conhecimento no domínio de sistemas embarcados. A atuação desse papel está relacionada à definição dos principais recursos, funções e restrições regendo o projeto e definição da arquitetura de referência independentemente de qualquer detalhe técnico ou tecnológico. A Figura 4.2 apresenta o conjunto de tarefas em que o arquiteto funcional é o executor primário e o conjunto de produtos de trabalho sob sua responsabilidade;

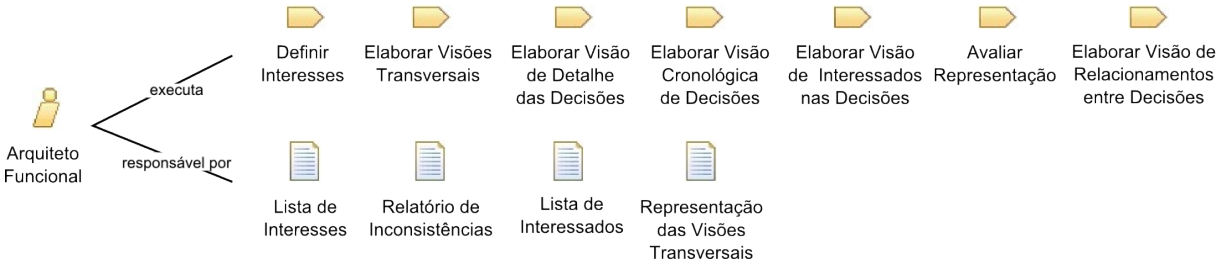

Figura 4.2: Participação do arquiteto funcional

Arquiteto técnico: é o arquiteto com o conhecimento técnico para projetar a arquitetura de referência. A atuação desse arquiteto está relacionada ao refinamento do projeto arquitetural feito pelo arquiteto funcional. O arquiteto técnico também pode auxiliar na seleção de tecnologias para garantir que os objetivos da arquitetura de referência sejam alcançados. A Figura 4.3 apresenta o conjunto de tarefas em que o arquiteto técnico é o executor primário e o conjunto de produtos de trabalho sob sua responsabilidade; e

Desenvolvedor: é a pessoa ou grupo responsável pela implementação da arquitetura de referência. O desenvolvedor possui vasto conhecimento das tecnologias selecionadas pelo arquiteto técnico ou que são necessárias para alcançar os objetivos da arquitetura

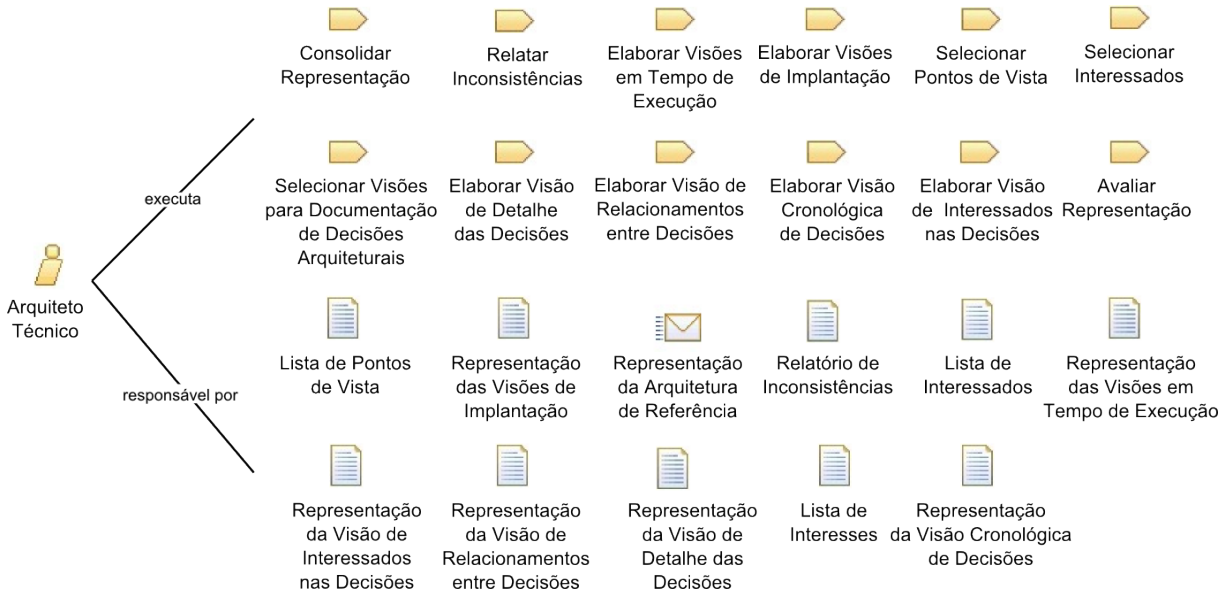

Figura 4.3: Participação do arquiteto técnico

de referência. A Figura 4.4 apresenta o conjunto de tarefas em que o desenvolvedor é o executor primário e o conjunto de produtos de trabalho sob sua responsabilidade.

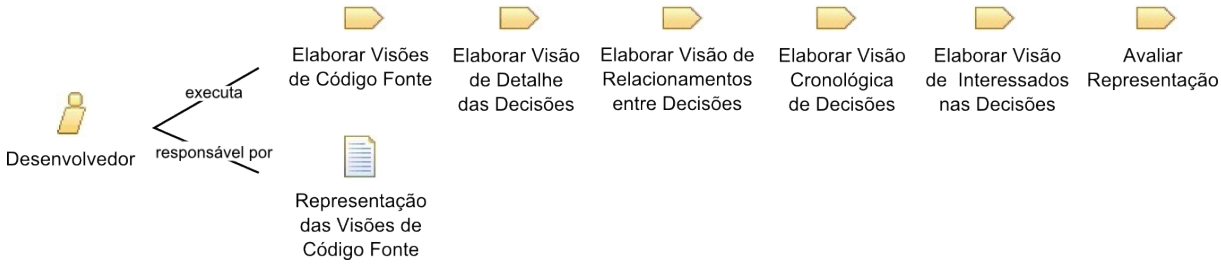

Figura 4.4: Participação do desenvolvedor

4.2.3 Tarefas Planejadas

O ProSA-Re propõe um conjunto de 15 tarefas separadas nas quatro atividades apresentadas anteriormente. É importante destacar que todas as tarefas do método são repetíveis, ou seja, podem ser executadas mais de uma vez sobre um mesmo conjunto de produtos de trabalho. Contudo, a frequência com a qual cada atividade será executada depende dos objetivos da arquitetura de referência.

A atividade de **Definição do Escopo** possui como objetivo delinear o contexto do projeto e representação da arquitetura de referência. O fluxo de trabalho dessa atividade é mostrado na Figura 4.5. Em seguida, cada tarefa que compõe essa atividade é detalhada e também ilustrada na Figura 4.6.

i) *Selecionar Interessados:* compreende a seleção dos principais interessados na arquitetura de referência. Arquitetos técnicos podem ser apoiados por arquitetos funcionais

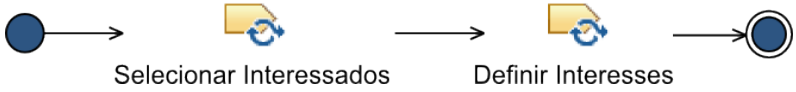

Figura 4.5: Fluxo de trabalho para a atividade de Definição do Escopo

na execução dessa tarefa. A lista de interessados é a entrada para essa tarefa, sendo atualizada em sua execução; e

ii) *Definir Interesses:* compreende a identificação dos interesses que precisam ser capturados na representação da arquitetura de referência. Arquitetos técnicos podem ser auxiliados por arquitetos funcionais na execução dessa tarefa. A lista de interessados modificada e a lista de interesses padrão são as entradas e a lista de interesses modificada é saída dessa tarefa.

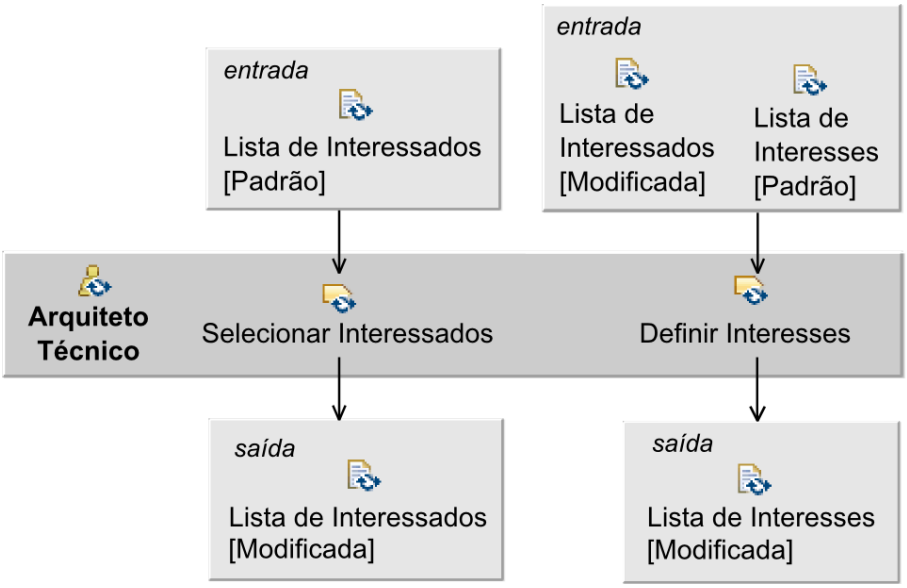

Figura 4.6: Detalhes da atividade de Definição do Escopo

A atividade de **Elaboração das Visões** possui como finalidade elaborar a representação da arquitetura de referência (ou seja, as visões arquiteturais) com base na lista inicial de pontos de vista apresentados na Tabela 4.1. Do conjunto de pontos de vista arquiteturais apresentados, precisam ser construídos apenas aqueles que são interessantes para uma determinada arquitetura de referência. Desse modo, mais de um grupo de pontos de vista pode ser selecionado, com uma ou mais visões arquiteturais de cada grupo construídas. O ProSA-Re define a sequência para a elaboração dos grupos de visões de acordo com o nível de abstração das informações apresentadas em cada grupo, partindo de informações de alto nível sobre a arquitetura de referência (por exemplo, ponto de vista de dados compartilhados e de processos) para informações de nível de abstração mais baixo, ou seja, mais próximas do código fonte (por exemplo, ponto de vista de desenvolvimento e

estrutural). O fluxo de trabalho dessa atividade é mostrado na Figura 4.7. Em seguida, cada tarefa que compõe essa atividade é detalhada e também ilustrada na Figura 4.8.

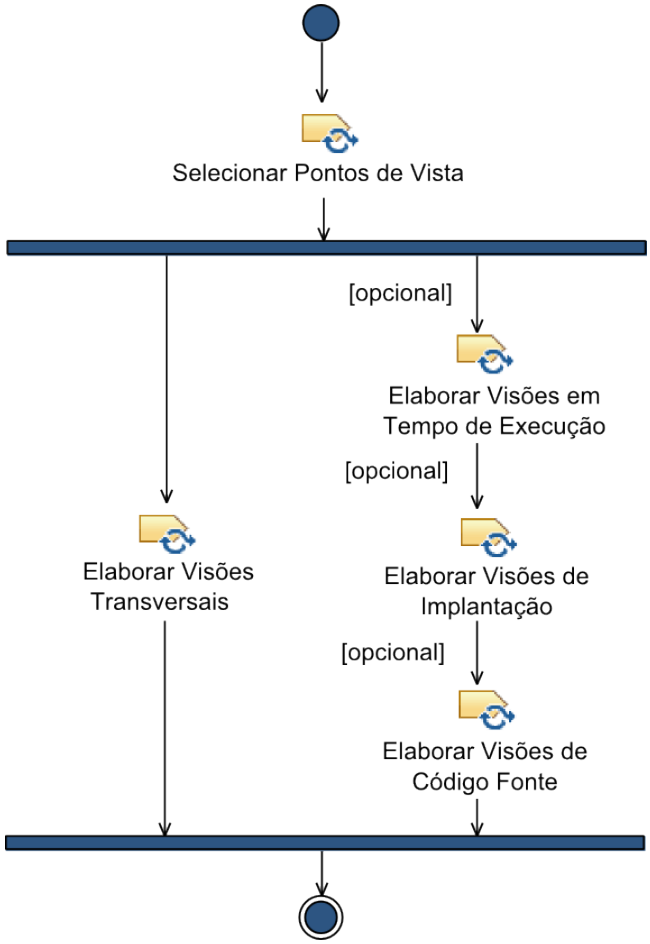

Figura 4.7: Fluxo de trabalho para a atividade de Elaboração das Visões

i) *Selecionar Pontos de Vista:* tem como objetivo selecionar o conjunto de pontos de vista mais adequados para representar a arquitetura de referência baseando-se na lista de interesses definida anteriormente. O ProSA-Re fornece uma lista de pontos de vista padrão para ser utilizada como entrada nessa tarefa, sendo então atualizada por sua execução;

ii) *Elaborar Visões Transversais:* tem como objetivo criar as visões transversais da arquitetura de referência (Grupo 1 da Tabela 4.1). Visões desse grupo contêm informações gerais sobre funções, domínio e escopo da arquitetura de referência, formando a espinha dorsal para o estabelecimento de todos os outros grupos de visões. Por essa razão, o ProSA-Re propõe a inclusão de pelo menos uma visão desse grupo na representação da arquitetura de referência, de modo que esse grupo é o único obrigatório. Além disso, visões desse grupo devem ser elaboradas paralelamente às visões de outros gru-

pos para garantir a consistência das informações representadas. A representação das visões transversais é a saída dessa tarefa;

iii) *Elaborar Visões em Tempo de Execução:* tem como objetivo criar as visões em tempo de execução da arquitetura de referência (Grupo 2 da Tabela 4.1). A partir das visões transversais elaboradas anteriormente, as visões desse grupo detalham o comportamento esperado pela arquitetura de referência especificando como os elementos que a compõe interagem em tempo de execução. Essa tarefa é opcional, pois pode ser necessária uma maior maturidade das visões transversais para iniciar a elaboração desse grupo de visões. Contudo, essa tarefa ainda pode ser executada em iterações futuras, caso seja identificada a necessidade. A representação das visões em tempo de execução é a saída dessa tarefa;

iv) *Elaborar Visões de Implantação:* tem como objetivo criar as visões de implantação da arquitetura de referência (Grupo 3 da Tabela 4.1). A partir das visões em tempo de execução elaboradas anteriormente, as visões desse grupo descrevem o mapeamento da arquitetura de referência em elementos físicos ou de hardware. Essa tarefa também é opcional, pois pode ser necessária uma maior maturidade das visões em tempo de execução para iniciar a elaboração desse grupo de visões. Contudo, essa tarefa ainda pode ser executada em iterações futuras, caso seja identificada a necessidade. A representação das visões de implantação é a saída dessa tarefa. O arquiteto técnico pode ser apoiado pelos desenvolvedores para executar essa tarefa; e

v) *Elaborar Visões de Código Fonte:* tem como objetivo criar as visões de código fonte da arquitetura de referência (Grupo 4 da Tabela 4.1). A partir das visões de implantação elaboradas anteriormente, as visões desse grupo descrevem como a arquitetura de referência pode ser mapeada em elementos lógicos. Essa tarefa também é opcional, pois pode ser necessária uma maior maturidade das visões de implantação para iniciar a elaboração desse grupo de visões. Contudo, essa tarefa ainda pode ser executada em iterações futuras, caso seja identificada a necessidade. A representação das visões de código fonte é a saída dessa tarefa. Desenvolvedores podem ser auxiliados por arquitetos técnicos na execução dessa tarefa.

A atividade **Representação das Decisões Arquiteturais** possui como objetivo capturar e registrar o conhecimento tácito envolvido na construção e representação de arquiteturas de referência, buscando facilitar o reúso e manutenção dessas arquiteturas. Por exemplo, a razão para a seleção de um determinado interessado ou ponto de vista, para a determinação do nível de abstração empregado ou ainda para a adoção de um padrão arquitetural específico. O fluxo de trabalho dessa atividade é mostrado na Figura

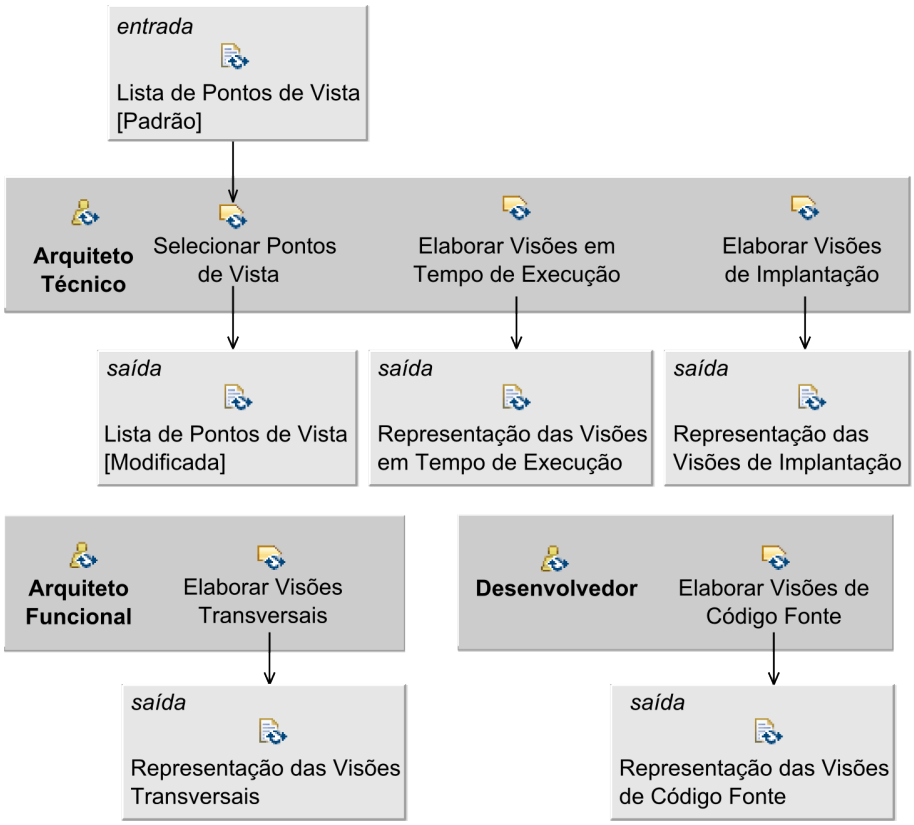

Figura 4.8: Detalhe da atividade de Elaboração das Visões

4.9. Em seguida, cada tarefa que compõe essa atividade é detalhada e também ilustrada na Figura 4.10.

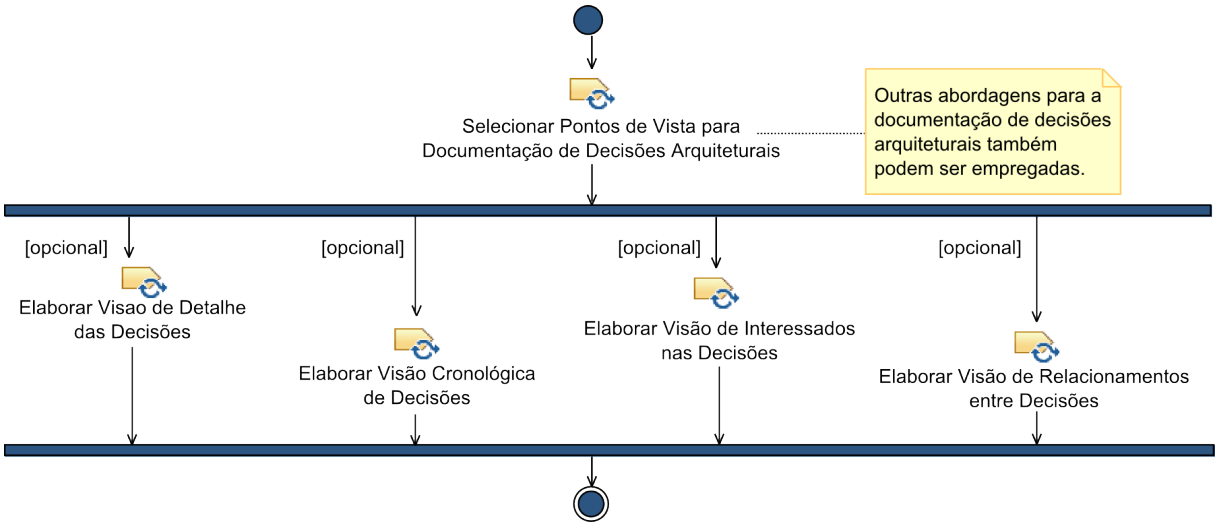

Figura 4.9: Fluxo de trabalho para a atividade de Representação das Decisões Arquiteturais

i) *Selecionar Pontos de Vista para Documentação de Decisões Arquiteturais:* compreende a seleção do conjunto de pontos de vista que irão documentar diferentes aspectos

da criação, uso e evolução do conhecimento tácito. ProSA-Re adota um *framework* arquitetural específico para documentação de decisões arquiteturais (van Heesch et al., 2012), por se tratar de uma abordagem completa; alternativamente, abordagens simples como, por exemplo, a criação de uma *wiki* para colaboração e divulgação do conhecimento entre todos os interessados, ou a criação de um *index* de habilidades (do inglês, *skill index*) para personalização do conhecimento podem ser empregadas (Kruchten, 2009). A lista de pontos de vista modificada é a única saída dessa atividade;

ii) *Elaborar Visão de Detalhe das Decisões:* cria a visão com informações detalhadas sobre decisões arquiteturais individuais. A representação da visão de detalhe das decisões é a única saída dessa tarefa;

iii) *Elaborar Visão Cronológica de Decisões:* cria a visão que descreve a evolução das decisões arquiteturais. Nessa visão, são apresentadas todas as versões de cada decisão, ou seja, o histórico de todos os estados que cada decisão já assumiu. A representação da visão cronológica de decisões é a única saída dessa tarefa;

iv) *Elaborar Visão de Interessados nas Decisões:* cria a visão descrevendo as responsabilidades dos interessados no processo de tomada de decisão, o que é importante para a personalização do conhecimento arquitetural, ou seja, a documentação de quem detém um conhecimento específico. A representação da visão de interessados nas decisões é a única saída dessa tarefa; e

v) *Elaborar Visão de Relacionamentos entre Decisões:* cria a visão que detalha o relacionamento entre decisões arquiteturais e seus respectivos estados em um dado momento. A representação da visão de relacionamentos entre decisões é a única saída dessa tarefa.

A atividade de **Avaliação da Representação** tem como objetivo mesclar todos os documentos relacionados à descrição da arquitetura de referência e submeter essa representação a uma avaliação arquitetural. O fluxo de trabalho dessa atividade é mostrado na Figura 4.11. Em seguida, cada tarefa que compõe essa atividade é detalhada e também ilustrada na Figura 4.12.

i) *Consolidar Representação:* tem como objetivo unificar todos os produtos de trabalho produzidos —visões arquiteturais, glossário e decisões arquiteturais— em um único produto de trabalho para organizar a representação. A representação da arquitetura de referência é a saída dessa tarefa;

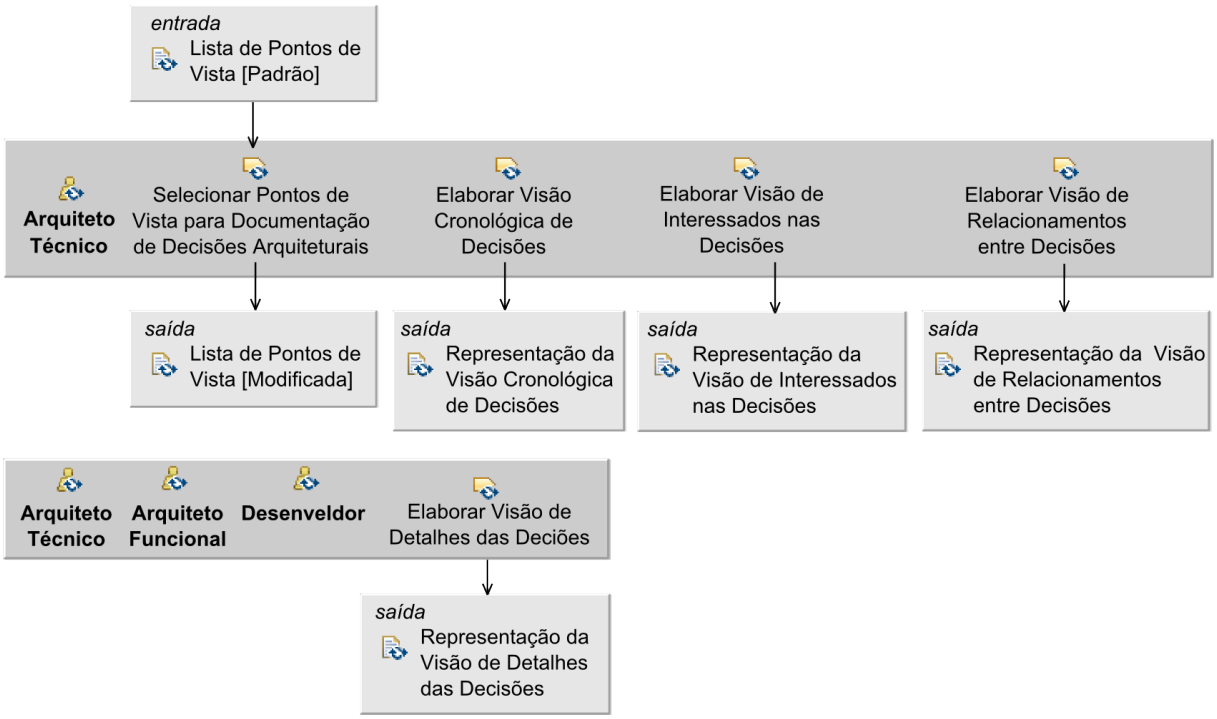

Figura 4.10: Detalhe da atividade de Representação das Decisões Arquiteturais

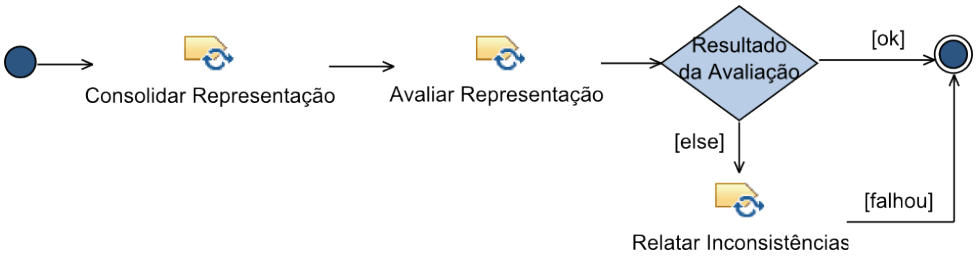

Figura 4.11: Fluxo de trabalho para a atividade de Avaliação da Representação

ii) *Avaliar Representação:* tem como objetivo passar a representação por uma avaliação para verificar a sua completude, consistência e efetividade em capturar interesses relevantes para a arquitetura de referência sendo construída. Por exemplo, um *checklist* poderia ser utilizado para avaliar a representação com relação ao conteúdo e interesses considerados. Vale destacar que o *chechlist* é uma alternativa com ótimo custo-benefício para a avaliação de arquiteturas de referência, já sendo explorada para a avaliação de arquiteturas de software (Barcelos e Travassos, 2006) e com trabalhos em andamento para a sua aplicação na avaliação de arquiteturas de referência (Nakagawa et al., 2013). Se falhas forem encontradas, essa tarefa desencadeia a revisão das atividades anteriores para correção ou realização de ajustes na representação;

iii) *Relatar Inconsistências:* é uma tarefa dirigida a eventos que é executada apenas em caso de falha da avaliação da representação. Nesse caso, todos os problemas encontrados são documentados em um relatório de inconsistências, o qual será utilizado

como guia de uma iteração para revisão das atividades do método. Se algum problema não puder ser resolvido nessa revisão, essa informação deve ser registrada na representação da arquitetura de referência.

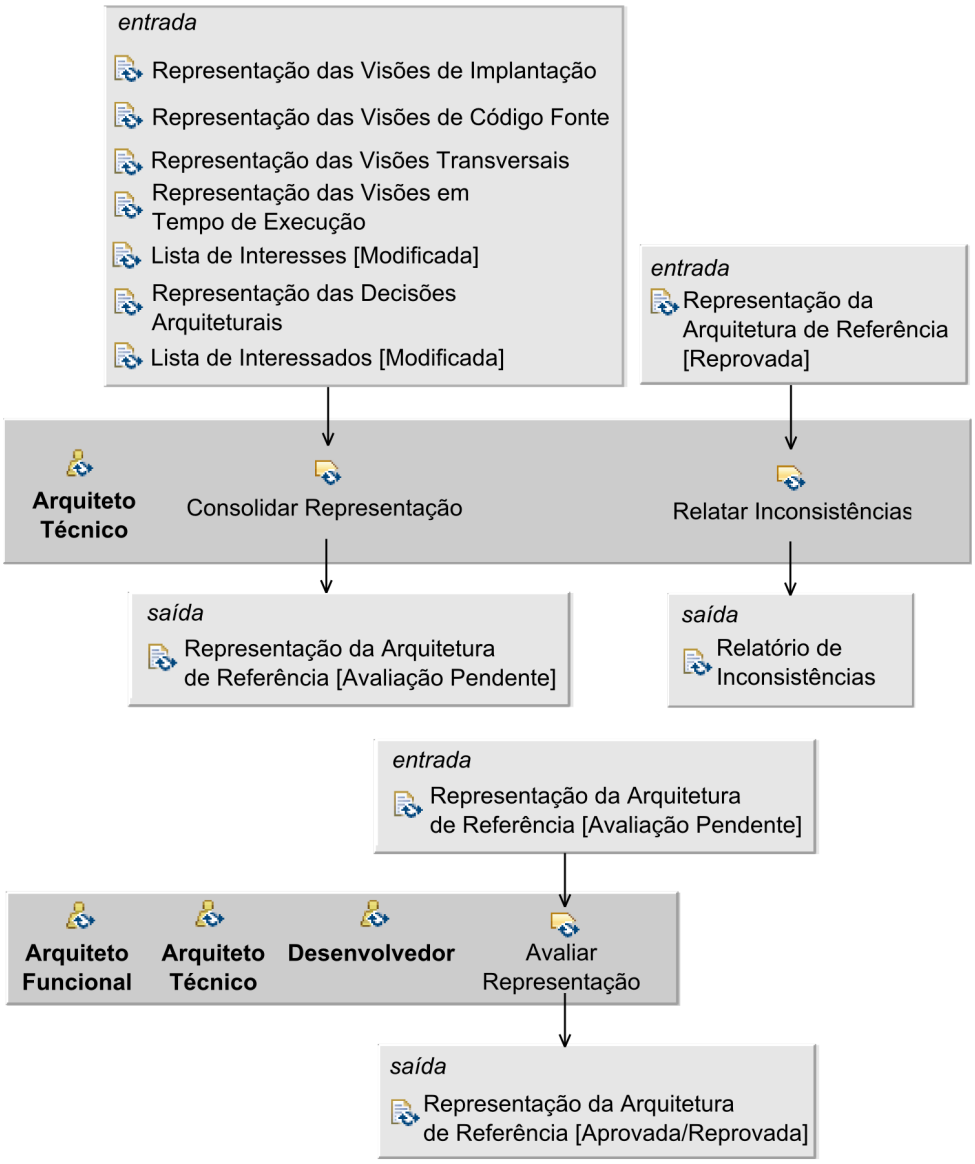

Figura 4.12: Detalhe da atividade de Avaliação da Representação

4.2.4 Diretrizes para Utilização

Nesta seção são apresentadas as diretrizes para a utilização do ProSA-Re. Essas diretrizes contém orientações importantes sobre como executar as tarefas do método e também modelos que podem ser empregados como base para a criação dos produtos de trabalho sugeridos.

Orientações para aplicação do ProSA-Re fornece direções sobre como aplicar o método para a representação de uma arquitetura de referência de sistemas embarcados. Essas orientações sugerem, por exemplo, como o conjunto de visões descrevendo a arquitetura pode evoluir de acordo com a maturidade da arquitetura de referência, além de estabelecer a sequência por meio da qual o conjunto de visões pode ser construído. A representação dessa abordagem recebeu o nome de Cone Arquitetural para o Ciclo de Vida de Arquiteturas de Referência.

Descrição: A primeira atividade do ProSA-Re consiste na definição do escopo da arquitetura de referência. Com base nessa informação, é possível então identificar quais interesses são relevantes para a arquitetura de referência e selecionar os pontos de vista que capturam esses interesses. Com a lista de pontos de vista a serem construídos delineada, a elaboração de cada visão arquitetural pode ser iniciada. No início, o ProSA-Re recomenda que apenas visões transversais sejam construídas, uma vez que elas formarão a espinha dorsal para o desenvolvimento da arquitetura de referência. Ao passo em que a arquitetura de referência for avançando em seu ciclo de vida, mais visões poderão ser adicionadas ao conjunto inicial de modo iterativo e incremental (conforme Figura 4.13). Em especial, o método sugere a seguinte sequência para a elaboração das visões de outros grupos: primeiro visões em tempo de execução, seguidas por visões de implantação e, por fim, visões de código fonte (conforme Figura 4.14). De acordo com essa abordagem, o nível de detalhes com o qual cada visão é descrita deverá aumentar com a maturidade da arquitetura de referência. Em paralelo a todas essas atividades, o ProSA-Re recomenda a documentação das decisões arquiteturais mais relevantes.

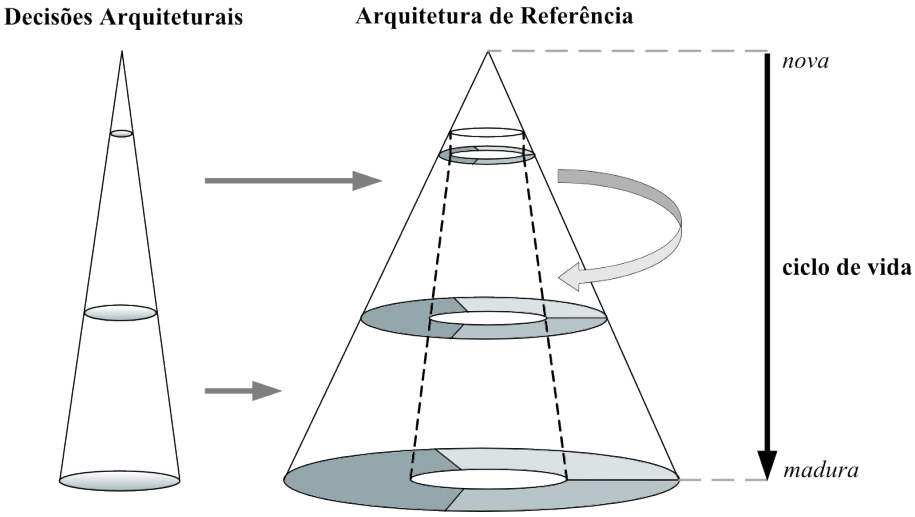

Figura 4.13: Cone Arquitetural para o Ciclo de Vida de Arquiteturas de Referência

Orientações para seleção dos pontos de vista e construção das visões arquiteturais: guiam a seleção do conjunto adequado de pontos de vista para representação de arquiteturas de referência do domínio de sistemas embarcados.

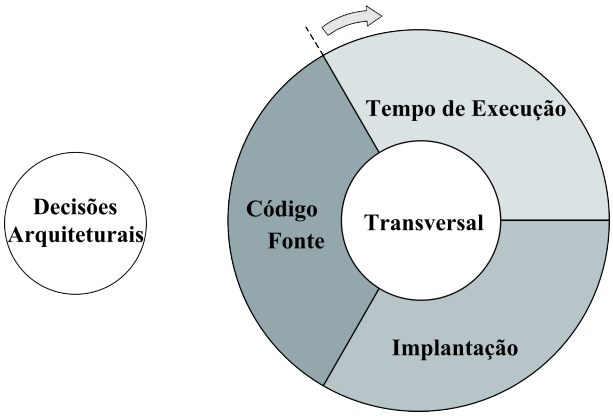

Figura 4.14: Corte horizontal do Cone Arquitetural

Descrição: a seleção dos pontos de vista arquiteturais depende principalmente dos interessados e de seus respectivos interesses. Uma vez que a arquitetura de referência é construída visando atender um conjunto heterogêneo de interessados, é importante selecionar um conjunto abrangente de pontos de vista. Contudo, deve-se evitar a adição de informações desnecessárias, uma vez que isso dificulta a manutenção e utilização da representação arquitetural. Por isso, encontrar o balanço adequado entre quais informações incluir na representação é uma tarefa bastante complexa. Nesse contexto, o ProSA-Re sugere alguns passos para auxiliar os arquitetos na seleção do conjunto de pontos de vista:

i) Identificação dos pontos de vista que capturam interesses relevantes da arquitetura de referência: seleciona na lista de pontos de vista aqueles que estão diretamente relacionados a esses interesses. Visões representando a mesma informação, porém, com diferentes níveis de abstração, podem aumentar a complexidade da documentação. Contudo, essa é uma característica particularmente interessante para arquiteturas de referência, uma vez que elas precisam ser acessíveis para uma grande variedade de interessados. Por exemplo, a visão estrutural e a visão lógica capturam, essencialmente, a mesma informação, porém a visão estrutural é mais abstrata que a visão lógica;

ii) Classificação de cada ponto de vista de acordo com o nível de detalhe: categoriza o conjunto de pontos de vista de acordo com o nível de detalhes necessários para descrevê-lo (por exemplo, visão geral, em mais detalhes ou detalhado). Essa classificação depende dos objetivos da arquitetura de referência. Assim, se o objetivo da arquitetura de referência for a padronização, então pode ser preciso descrever as visões detalhadamente. Por outro lado, se o objetivo da arquitetura de referência for servir como mecanismo facilitador, então apenas uma visão geral de cada visão pode ser suficiente. Vale destacar ainda que a maturidade da arquitetura de referência também influencia no nível de detalhes com o qual é possível descrever cada visão,

uma vez que um nível de detalhes mais elevado apenas pode ser alcançado com a maturidade da arquitetura de referência; e

iii) Distribuição das visões pelo ciclo de vida da arquitetura de referência: o estágio atual da arquitetura de referência em seu no ciclo de vida determina quais tipos de informações podem ser descritas. Por exemplo, segundo o Cone Arquitetural apresentado na Figura 4.13, em um estágio inicial do ciclo de vida é possível elaborar visões que capturam informações de alto nível sobre a arquitetura de referência, como a visão conceitual e a visão de funcionalidades, enquanto em um estágio mais avançado do ciclo de vida já é possível criar visões mais complexas e menos abstratas da arquitetura de referência, como a visão de processos, a visão estrutural e a visão lógica. Nesse contexto, o ProSA-Re sugere começar o projeto da arquitetura de referência apenas com visões transversais, uma vez que elas fornecem a espinha dorsal para a descrição de todas as outras visões da arquitetura de referência. Quando houver uma base conceitual bem definida, a elaboração dos outros grupos de visões pode ser iniciada. É importante destacar ainda que o conjunto de visões descrevendo a arquitetura de referência pode evoluir para acomodar mudanças nas necessidades dos interessados e refinar a descrição arquitetural.

Orientações para documentação das visões arquiteturais: guia a elaboração do conjunto de visões descrevendo as arquiteturas de referência de sistemas embarcados.

Descrição: O ProSA-Re propõe que o conjunto de visões seja construído iterativamente, com mais detalhes ou visões sendo incluídos a cada iteração. A Figura 4.14 apresenta a sequência para construção do conjunto de visões de acordo com o grupo de pontos de vista. É importante destacar ainda que visões com um alto nível de detalhes devem ser resultado de várias iterações e sucessivos refinamentos nos artefatos criados. A descrição de cada visão deve incluir as características, as interfaces e os comportamentos de seus elementos.

Como a quantidade de informações que podem ser descritas em cada visão está diretamente relacionada à expressividade da linguagem de descrição arquitetural escolhida, é importante selecionar linguagens que sejam adequadas para o domínio de sistemas embarcados. A partir dos resultados da revisão sistemática conduzida no contexto deste trabalho e que revelou a UML como a linguagem semi-formal mais amplamente utilizada, o ProSA-Re sugere a adoção da linguagem SysML, uma vez de que se trata de um padrão OMG que combina o rigor das técnicas da UML à compreensibilidade da linguagem natural. Desse modo, é interessante considerar a SysML para a representação de arquiteturas de referência do domínio de sistemas embarcados, especialmente porque também existe a possibilidade de que, em versões futuras, sejam incluídas restrições e semânticas formais à

linguagem para aumentar a precisão dos seus modelos. O Anexo A apresenta uma breve introdução sobre a linguagem e seus diagramas.

Na Tabela 4.2 são descritas quais técnicas da SysML (versão 1.2) podem ser empregadas para a elaboração de cada ponto de vista selecionado pelo ProSA-Re. Também são mostradas na tabela técnicas adicionais, como diagramas extras e uso de elementos específicos, que podem acrescentar detalhes relevantes na descrição de cada ponto de vista. Para a elaboração dos diagramas sugeridos pelo método, é recomendada a utilização de uma ferramenta para a criação dos diagramas. Atualmente, diversas ferramentas de apoio à linguagem SySML podem ser encontradas. Por exemplo, as ferramentas *Enterprise Architect*[1] (da empresa Sparx Systems) e *MagicDraw*[2] (da empresa No Magic) disponibizam um *plugin* para modelagem em SysML, a ferramenta *Rhapsody Developer*[3] (da IBM) permite a criação de diagramas nessa linguagem, e o *Papyrus*[4] é uma extensão para o Eclipse[5] para modelagem de sistemas, inclusive em SysML.

Tabela 4.2: Técnicas para representação de pontos de vista usando a SysML

Ponto de vista	**Técnica Principal**	**Técnica Auxiliar**	**Elementos**
	Grupo 1: Pontos de vista transversais		
Funcionalidade	Diagrama de Casos de Uso	Diagrama de Atividades	
Conceitual	Diagrama de Definição de Blocos		
Serviços do sistema	Diagrama de Definição de Blocos		
Variabilidade	Diagrama de Definição de Blocos		
	Grupo 2: Pontos de Vista em Tempo de Execução		
Componentes colaborativos	Diagrama de Blocos Internos	Diagrama de Atividades	
Dados compartilhados	Diagrama de Blocos Internos	Diagrama de Atividades, Diagrama de Máquina de Estados	
Processos	Diagrama de Blocos Internos	Diagrama de Máquina de Estados	

Continua...

[1]Disponível em `http://www.sparxsystems.com.au` (Acessado em 21/01/2013)
[2]Disponível em `http://www.nomagic.com/products/magicdraw.html` (Acessado em 21/01/2013)
[3]Disponível em `http://www-142.ibm.com/software/products/us/en/ratirhapfami` (Acessado em 21/01/2013)
[4]Disponível em `http://www.eclipse.org/modeling/mdt/papyrus` (Acessado em 21/01/2013)
[5]Disponível em `http://www.eclipse.org` (Acessado em 21/01/2013)

Tabela 4.2 – Continuação

Ponto de vista	Técnica Principal	Técnica Auxiliar	Elementos
	Grupo 3: Pontos de Vista de Implantação		
Desenvolvimento	Diagrama de Componentes		
Implantação	Diagrama de Pacotes		
Físico	Diagrama de Pacotes		Alocação
Tecnologia	Textual		
	Grupo 4: Pontos de Vista de Código Fonte		
Estrutural	Diagrama de Definição de Blocos		
Módulos	Diagrama de Componentes		
Lógico	Diagrama de Definição de Blocos		

***Checklist* para avaliação da representação de arquiteturas de referência de sistemas embarcados:** é a proposta de um conjunto de questões sobre a organização e conteúdo da representação. Além de questões específicas sobre o domínio de sistemas embarcados como, por exemplo, questões sobre segurança e interoperabilidade, o *checklist* também contém questões para avaliar se a descrição da arquitetura de referência atende as principais recomendações do padrão ISO/IEC/IEEE 42010 (ISO, 2010). Por exemplo, podem ser incluídas questões para verificar a consistência do vocabulário utilizado, a rastreabilidade dos elementos entre os diferentes diagramas, a utilidade das visões arquiteturais construídas, os objetivos da arquitetura de referências e também se a descrição dos principais riscos e impactos da arquitetura de referência foram detalhados. Para minimizar o esforço necessário para a aplicação desse *checklist*, as questões podem ser separadas e distribuídas aos diferentes interessados na arquitetura de referência. Vale destacar ainda que um *checklist* está sendo elaborado por pesquisadores e alunos do grupo de pesquisa ao qual este trabalho faz parte.

Referências: é a literatura de apoio ao projeto e representação da arquitetura de referência. Pode incluir, por exemplo, leis, padrões e restrições que sejam específicas para o domínio de sistemas embarcados ou para a arquitetura de referência. Essas informações são importantes para aumentar a usabilidade da arquitetura de referência e podem ser úteis, inclusive, para justificar decisões arquiteturais tomadas.

Modelo para descrição de decisões arquiteturais: apresenta uma estrutura básica de um formulário para documentar a motivação e justificativa de decisões arquiteturais. Diferentes modelos de formulários podem ser encontrados na literatura, sendo que o modelo

proposto por van Heesch et al. (2012) é bastante abrangente. Esse modelo é apresentado na Tabela 4.3. É importante ressaltar que esse formulário ainda pode ser adaptado.

Tabela 4.3: Modelo para descrição de decisões arquiteturais (van Heesch et al., 2012)

Seção	Descrição
Nome	Um breve identificados da decisão que será utilizado como chave em outras visões.
Estado atual	Descreve o estado atual da decisão
Grupos de decisão	Uma decisão pode estar associada a um ou mais grupos, os quais compartilham características específicas. Decisões pode ser agrupadas, por exemplo, por subsistemas, interesses ou interessados.
Problema	As circunstâncias que levaram o arquiteto a optar por uma decisão dentre outras alternativas.
Decisão	O resultado da decisão ou solução encontrada.
Alternativas	Soluções alternativas que também foram consideradas no processo de tomada da decisão.
Decisões relacionadas	Lista de todas as decisões relacionadas a esta decisão.
Interesses relacionados	Lista de todos os interesses tratados nessa decisão.
Histórico	O histórico de mudanças de estado da decisão descrita como, por exemplo, quando ela foi proposta, aprovada, dentre outros.

Modelo para representação de arquiteturas de referência de sistemas embarcados: considerando que a documentação da arquitetura de referência é bastante importante para a sua usabilidade e manutenibilidade, é relevante definir uma estrutura básica para o documento que descreve a arquitetura de referência até para padronizar a documentação construída com o apoio do ProSA-Re. Recomenda-se que uma representação arquitetural completa apresente (ISO, 2010):

i) Nome ou identificação para a arquitetura de referência;

ii) Visão Geral: descreve as principais características e objetivos da arquitetura de referência;

iii) Sumário: apresenta a organização do documento e facilita o seu uso;

iv) Introdução: apresenta explicações sobre o projeto da arquitetura de referência, o contexto em que ela está inserida, suas características e limitações, além de considerações sobre o ambiente de aplicação;

v) Objetivos: apresenta e detalha os principais objetivos da arquitetura de referência e como se espera atendê-los;

vi) Interessados: apresenta os principais interessados na arquitetura de referência ou os interessados em potencial, que podem estar interessados em uma instância arquitetural, para os quais a documentação é destinada;

vii) Interesses: apresenta uma descrição para cada interesse identificado a partir dos interessados selecionados e também dos objetivos da arquitetura de referência;

viii) Visões: apresenta a representação da arquitetura de referência a partir do conjunto de pontos de vista selecionados. É interessante iniciar essa seção pelas visões arquiteturais de nível de abstração mais elevado e acrescentar gradualmente visões arquiteturais de nível de abstração mais baixo;

ix) Decisões Abertas: descreve as decisões ou questões que não foram incluídas ou tratadas pela arquitetura de referência, mas que devem ser consideradas em instâncias arquiteturais específicas ou de acordo com as tecnologias disponíveis atualmente;

x) Material de Apoio: apresenta qualquer material que possa auxiliar na compreensão da representação arquitetural como, por exemplo, padrões, leis e ou restrições que são específicas para o domínio; e

xi) Histórico de Mudanças: registra todas as alterações feitas no documento desde a data de sua criação.

É importante destacar que outros modelos para documentação da representação arquitetural existem e também podem ser adotados como, por exemplo, o modelo proposto por Eklund et al. (2012) que separa a documentação da arquitetura de referência em dois documentos complementares, mas com propósitos e públicos distintos. Nesse sentido, enquanto o primeiro documento captura apenas os conceitos e detalhes essenciais sobre a arquitetura de referência e é destinado a um conjunto diversificado de interessados, o segundo documento captura todos os detalhes técnicos necessários para a instanciação da arquitetura de referência e é destinado, principalmente, à arquitetos e desenvolvedores.

Modelo para representação de visões arquiteturais: considerando que a visão arquitetural representa a arquitetura de referência a partir de um determinado ponto de vista, é importante capturar informações suficientes sobre essa visão de modo a facilitar o seu entendimento. Por esse motivo, o ProSA-Re também propõe uma estrutura predefinida para o documento que descreve a visão arquitetural até para padronizar a documentação elaborada com o seu apoio. Baseado no modelo proposto por Clements et al. (2011), o ProSA-Re recomenda que a descrição de cada visão arquitetural inclua as seguintes informações:

i) Interesses: descreve os interesses capturados em cada visão;

ii) Visão Primária: apresenta o diagrama principal que ilustra essa visão. Opcionalmente, pode ser incluída uma descrição dos elementos apresentados na visão primária e também uma breve descrição da notação utilizada, se ela não for padrão;

iii) Uma ou mais visões adicionais: apresenta um ou mais diagramas secundários que adicionam informações complementares à visão primária e enriquecem a visão arquitetural. Opcionalmente, também podem ser incluídas as descrições dos elementos apresentados em cada visão adicional e as notações utilizadas; e

iv) Histórico de Mudanças: registra todas as alterações feitas na visão arquitetural desde a data de sua criação.

4.3 Considerações Finais

Neste capítulo, foi apresentado o ProSA-Re, um método para a construção da representação de arquiteturas de referência de sistemas embarcados. O conjunto de atividades proposto pelo método visa a construção de representações arquiteturais que sejam abrangentes, compactas e padronizadas para facilitar a comunicação entre todos os diferentes interessados que podem existir na arquitetura de referência. Observa-se que para a utilização do método já existe um conjunto de ferramentas que apoiam, dentre outras funcionalidades, a elaboração dos diagramas sugeridos, sendo algumas inclusive software livre. Com a finalidade de avaliar a capacidade do método para a elaboração de representações para arquiteturas de referência de sistemas embarcados que sejam adequadas foi conduzido um estudo de caso. Foi também realizada uma avaliação qualitativa com especialistas da área de Arquitetura de Software para avaliar a viabilidade do ProSA-Re. Os resultados das duas avaliações são apresentados no próximo capítulo.

Capítulo

5

Avaliação do ProSA-Re

5.1 Considerações Iniciais

Uma avaliação completa do ProSA-Re deve considerar não apenas o conjunto de atividades propostas (perspectiva dos executores do método), como também o seu potencial para a criação de representações mais adequadas para arquiteturas de referência do domínio de sistemas embarcados (perspectiva dos usuários da representação). Por esse motivo, a avaliação do método foi dividida em duas etapas. Na primeira etapa, um estudo de caso foi conduzido para identificar benefícios e limitações da representação arquitetural construída com o apoio do método. Na segunda etapa, uma avaliação com especialistas da área de arquitetura de software foi realizada para identificar possíveis benefícios e limitações de sua adoção em projetos reais.

Este capítulo é organizado da seguinte maneira. Na Seção 5.2, é apresentado o projeto conduzido para exemplificar a aplicação do método. Na Seção 5.3, é apresentada a avaliação da representação da arquitetura de referência construída com o apoio do ProSA-Re. Na Seção 5.4, é apresentado os resultados da avaliação conduzida com especialistas da área de arquitetura de software. Por fim, na Seção 5.5 são apresentadas as considerações finais deste capítulo.

5.2 Aplicação do Método na Representação de uma Arquitetura de Referência

Uma vez que o escopo deste trabalho não contempla o projeto de uma arquitetura de referência, a viabilidade das atividades e do conjunto de pontos de vista propostos pelo método foram avaliados por meio da elaboração da representação de uma arquitetura de referência para o domínio de sistemas embarcados disponível na literatura. Weyns (2006) representou sua arquitetura de referência para sistemas multi-agentes locais utilizando a abordagem proposta por Clements et al. (2011) para representação de arquiteturas de software. Com base no projeto original da arquitetura de referência, uma nova representação foi construída seguindo-se a sequência de tarefas propostas pelo ProSA-Re para elaboração das visões. Essa atividade levou em consideração os materiais disponibilizados pelo ProSA-Re, como lista de interesses, lista de pontos de vista e modelos de documentos. É importante destacar ainda que não foram feitas alterações ao projeto inicial da arquitetura de referência, uma vez que o propósito dessa atividade foi construir uma representação arquitetural que fosse equiparável à representação original. A representação elaborada nesta dissertação é apresentada no Apêndice C.

Segundo o ProSA-Re, a representação da arquitetura de referência deve ser resultado de várias iterações do método. Contudo, para garantir que a nova representação correspondesse, de fato, ao projeto original, partiu-se do conjunto de informações já disponibilizado pela representação original para selecionar os respectivos pontos de vista que seriam mais adequados para descrever a arquitetura de referência. Nesse sentido, observou-se situações em que uma mesma visão da representação original contribuiu para a elaboração de uma ou mais visões na nova representação. Por exemplo, a seção de modelo integrado da representação original foi a base para a elaboração da visão conceitual e da visão de variabilidade, sendo que todas as visões arquiteturais da representação original contribuíram para a elaboração da visão de variabilidade, uma vez que essas informações encontravam-se espalhadas originalmente. A Tabela 5.1 apresenta a estrutura geral de cada representação arquitetural. Da tabela, é possível observar, por exemplo, que a representação original organizou o conjunto de visões em pacotes para cada componente principal da arquitetura de referência, ou seja, o agente e o ambiente de aplicação. Em comparação, na nova representação, cada visão arquitetural representa ambos os componentes em um mesmo diagrama. Outra observação importante está relacionada à variedade de visões arquiteturais construídas seguindo-se as recomendações do ProSA-Re. Embora a quantidade seja maior — inicialmente, haviam apenas quatro visões arquiteturais enquanto na nova representação foram elaboradas sete — nenhuma informação estranha foi acrescentada e a quantidade de diagramas foi mantida. Mesmo considerando o maior número de visões

arquiteturais, houve uma redução de cerca de 30% no tamanho da representação (passou de 40 para 28 páginas[1]), o que pode ser resultado da maior expressividade da linguagem utilizada. Vale destacar apesar de algumas visões arquiteturais receberem o mesmo nome em ambos os documentos, o conteúdo de cada uma delas não é necessariamente o mesmo, uma vez que diferentes técnicas foram empregadas e algumas informações podem ter sido transferidas para outras seções do documento.

Tabela 5.1: Tabela comparativa de informações apresentadas em cada representação da arquitetura de referência

Original	Com apoio do ProSA-Re
1. Introdução	1. Introdução
2. Fundamentação e Contextualização	2. Objetivos
2.1 Explicação das características da arquitetura de referência e dos requisitos do domínio da arquitetura de referência	3. Interesses
2.2 Características e requisitos do domínio de aplicação da arquitetura de referência	4. Visões Arquiteturais
2.3 Organização da documentação da arquitetura de referência	4.1 Visão Conceitual
3. Modelo Integrado para Sistemas Multi-agentes Locais	4.2 Visão Estrutural
3.1 Modelo do Ambiente	4.3 Visão de Dados Compartilhados
3.2 Modelo do Agente	4.4 Visão de Componentes Colaborativos
4. Visão de Decomposição em Módulos	Visão adicional: Interfaces
4.1 Sistema Multi-agente Local	Visão Adicional: Funcionalidade de Percepção
4.2 Agente	Visão Adicional: Funcionalidade de Ação
4.3 Ambiente de Aplicação	Visão Adicional: Funcionalidade de Comunicação
5. Visão de Dados Compartilhados	4.5 Visão de Processos
5.1 Agente	Visão Adicional
5.2 Ambiente de Aplicação	4.6 Visão de Detalhes das Decisões Arquiteturais
6. Visão de Componentes Colaborativos	4.7 Visão de Variabilidade

Continua...

[1]Para efeito de comparação, considerou-se o tamanho da representação original no formato utilizado no estudo de caso, que padronizou os documentos utilizados.

Tabela 5.1 – Continuação

6.1 Percepção e Geração da Representação	Visão Adicional: Variabilidade de Conceitos
6.2 Tomada de Decisão e Interação	5. Decisões Abertas
6.3 Comunicação e Serviço de Comunicação	6. Histórico de Mudanças.
7. Visão de Processos	
7.1 Percepção, Interação e Comunicação	

A partir da elaboração dessa representação, foi possível identificar algumas diferenças importantes entre o ProSA-Re e o método para representação de arquiteturas de software proposto por Clements et al. (2011). Destaca-se, por exemplo, a nova proposta de organização das informações apresentadas. Nesse sentido, a principal diferença consiste na sugestão de visões arquiteturais específicas para capturar variabilidade e decisões arquiteturais, cujas informações encontravam-se anteriormente distribuídas para cada visão arquitetural. Na nova proposta, essas visões são consideradas transversais, uma vez que a sua definição se estende a todo o conjunto de visões. Como resultado, observa-se que a nova organização proposta pelo ProSA-Re implica em uma maior centralização das informações, o pode facilitar a manutenção, acessibilidade e compreensibilidade da arquitetura de referência.

5.3 Avaliação da Representação Arquitetural Construída com o Método

O objetivo dessa avaliação é investigar se a representação elaborada com o auxílio do ProSA-Re oferece, de fato, benefícios que justifiquem a sua adoção. Em particular, é relevante investigar a eficiência e usabilidade da representação em transferir o conhecimento que se tem sobre a arquitetura de referência de maneira consistente, visando ainda identificar quais são e como aspectos da representação colaboraram para os resultados observados. Para isso, foi conduzido um estudo de caso, uma técnica de pesquisa indicada para investigar fenômenos dentro de seu contexto real, especialmente quando os limites entre esse fenômeno e o seu contexto não estão claros (Yin, 1994).

5.3.1 Planejamento do Estudo de Caso

O projeto dessa avaliação é baseada na proposta de um estudo de caso embarcado único, na qual um único estudo de caso é conduzido, mas diferentes unidades de análise são

observadas (Gray, 2004). O relatório detalhado sobre o estudo de caso pode ser encontrado no Anexo D.

A execução do estudo de caso contou com a participação de quatro sujeitos. De modo geral, os sujeitos selecionados são alunos de mestrado e doutorado do grupo de Engenharia de Software do Instituto de Ciências Matemáticas e de Computação (ICMC/USP). Com base na quantidade de participantes foi definido o tamanho dos grupos, sendo dois sujeitos escolhidos aleatoriamente para compor o grupo A, que recebeu a representação original, e os outros dois sujeitos para compor o grupo B, que recebeu a representação criada com o auxílio do método ProSA-Re. No início do estudo de caso, cada sujeito recebeu uma cópia da representação a ser analisada e um *link* para o questionário *online*. O questionário foi composto por quatro partes comuns aos dois tipos de representação analisadas e uma quinta parte específica para o conjunto de visões arquiteturais utilizadas em cada representação. É importante notar que os sujeitos tiveram um prazo máximo de três semanas para responder o questionário. O projeto final é ilustrado na Tabela 5.2.

Tabela 5.2: Projeto do estudo de caso

	Objeto	
Sujeito	**Original**	**ProSA-Re**
Sujeito 1		X
Sujeito 2		X
Sujeito 3	X	
Sujeito 4	X	

5.3.2 Resultados do Estudo de Caso

A análise dos resultados obtidos no estudo de caso estão separadas de acordo com o tipo de representação analisada. Na sequência, é apresentada a análise comparativa dos resultados obtidos.

5.3.2.1 Análise do Grupo A

Na parte I e II do questionário foi solicitado dos sujeitos que completassem a leitura da representação da arquitetura de referência e respondessem a um conjunto de questões sobre o seu projeto, devendo informar também a dificuldade encontrada para responder cada questão. A Tabela 5.3 sumariza as respostas dos participantes do grupo A. Da tabela, é possível observar que os dois participantes apresentaram apenas um desempenho regular na parte II do questionário, tendo acertado menos da metade das questões propostas. A resposta dos participantes na questão 15, referente aos motivos para a decisão de criação de um repositório de conhecimento, foi desconsiderada uma vez que eles não discutiram os motivos para tal decisão. De modo geral, os participantes notaram um nível de dificuldade

intermediário (notas dois e três) para responder o conjunto de questões propostas, embora o sujeito 3 tenha encontrado mais dificuldade que o sujeito 4.

Tabela 5.3: Sumário das respostas dos participantes do grupo A

	Sujeito 3	Sujeito 4
Tempo gasto na Parte I	06:30	00:40
Tempo gasto na Parte II	00:23	00:50
Quantidade de acertos	2	3
Quantidade de erros	5	4
Questões que não soube responder	0	0

Na parte III do questionário, referente ao conjunto de questões específicas sobre a percepção da representação A, cada participante avaliou o conjunto de visões apresentadas com relação a três critérios: (i) utilidade, para medir o quanto uma visão arquitetural ajudou a responder as questões da parte II; (ii) esforço, para medir o trabalho necessário para uma visão arquitetural ser adequadamente analisada ou escaneada por uma informação específica; e (iii) clareza, para indicar o quanto uma visão arquitetural é confusa a ponto de não comunicar efetivamente aspectos relevantes da arquitetura de referência. A Figura 5.1 ilustra os resultados obtidos. É importante ressaltar que apesar do modelo integrado não ser uma visão arquitetural, ele também é incluído nessa avaliação uma vez que apresenta informações importantes sobre a arquitetura de referência. Da figura, é possível observar, por exemplo, que o modelo integrado recebeu a maior avaliação do sujeito 3 com relação à utilidade, mas também foi considerada a mais trabalhosa e confusa. Por outro lado, o sujeito 4 avaliou todas as visões com o mesmo grau de utilidade e teve uma opinião oposta com relação à clareza do modelo integrado que, na sua opinião, não é confuso. O sujeito 4 também indicou a visão de decomposição em módulos como sendo a mais trabalhosa.

Na sequência, ainda na parte III do questionário, os participantes avaliaram a percepção geral da representação. Os participantes destacaram as atividades de leitura e busca por informações específicas na representação como sendo as que mais tomaram tempo no estudo de caso. O sujeito 3 destacou a variedade de visões arquiteturais apresentada como essencial para o seu entendimento, sendo que a visão primária e a descrição textual também foram importantes. Para o sujeito 4, a visão primária e a variedade de visões foram ambas essenciais para o seu entendimento, sendo a descrição textual também foi importante. Desse modo, todos os aspectos da visão arquitetural merecem atenção. Sobre a localização das informações no documento, o sujeito 4 informou que conseguiu encontrar todas as informações solicitadas, enquanto o sujeito 3 afirmou que não. Nesse sentido, como o sujeito 3 não respondeu nenhuma questão da parte II com a alternativa "Não sei", esse resultado indica que a representação não deve ter sido compreendida corretamente

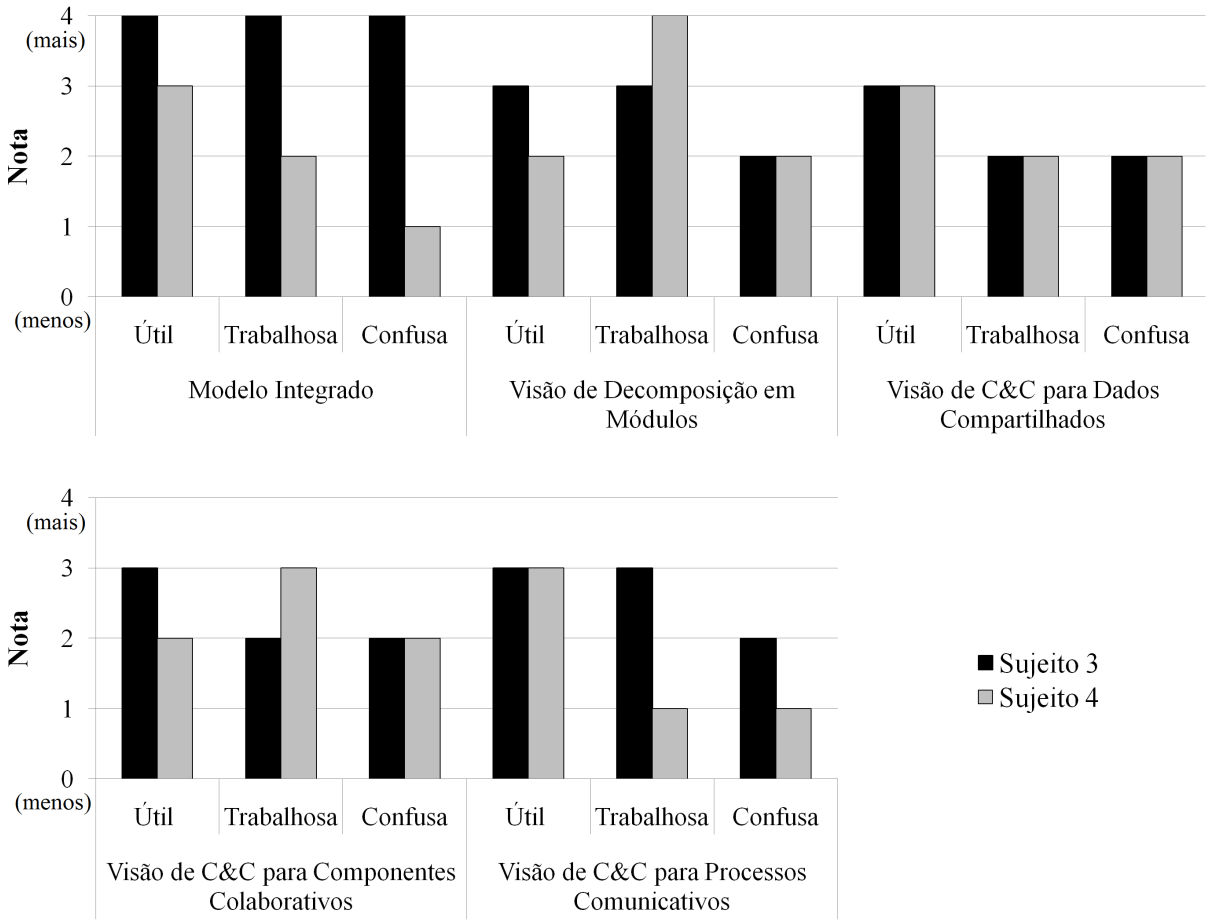

Figura 5.1: Avaliação do conjunto de visões arquiteturais da representação original

pelo sujeito. Dentre as dificuldades apontadas pelos participantes para responder as questões da parte II, ambos destacaram o tamanho da representação, tendo sido inclusive uma nova organização sugerida pelo sujeito 4. Também foi solicitado dos participantes que escolhessem e justificassem o conjunto de aspectos positivos e negativos da representação arquitetural analisada. A Figura 5.2 ilustra os resultados obtidos. Novamente, o tamanho e a organização da representação foram destacados como aspectos negativos, sendo os diagramas e a completude da representação considerados aspectos positivos. Porém, é importante observar que não houve consenso entre os sujeitos sobre as informações textuais. Destaca-se também a sugestão para usar mais tabelas e/ou diagramas para diminuir as informações textuais apresentadas. Em relação às considerações que poderiam melhorar o entendimento sobre a arquitetura de referência e sua representação, o sujeito 3 selecionou a consulta a um especialista no domínio e na arquitetura de referência, além de mais tempo para a leitura e resposta do questionário. Contudo, considerando-se que os participantes tiveram um período de três semanas para a realização do estudo de caso e o tempo total gasto pelo mesmo sujeito nas atividades de leitura e compreensão, acredita-se que tempo suficiente tenha sido disponibilizado para a participação no estudo de caso. Por outro

lado, o sujeito 4 considerou que uma melhor organização das informações e a inclusão de outras visões poderiam melhorar o entendimento da representação arquitetural.

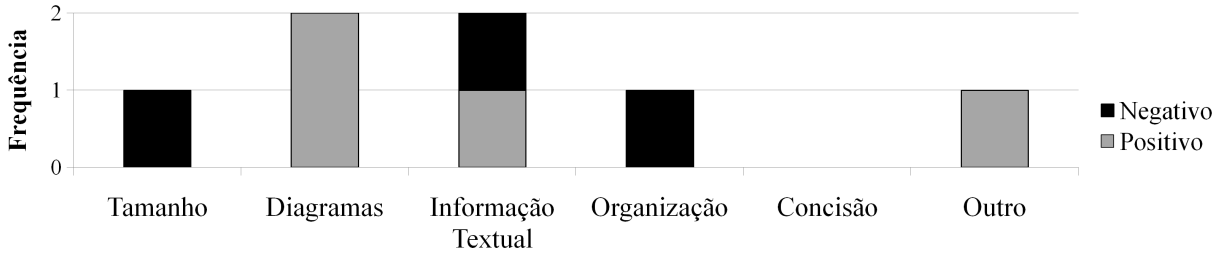

Figura 5.2: Avaliação de aspectos da representação original

Por fim, na parte IV do questionário foram feitas perguntas para avaliar as habilidades pessoais dos sujeitos. Os resultados obtidos nessa parte são ilustrados na Figura 5.3. Da figura, é possível observar que os participantes do grupo A apresentam sólidos conhecimentos em engenharia de software, arquitetura de software e arquitetura de referência. Além disso, eles possuem um conhecimento intermediário sobre UML e SysML.

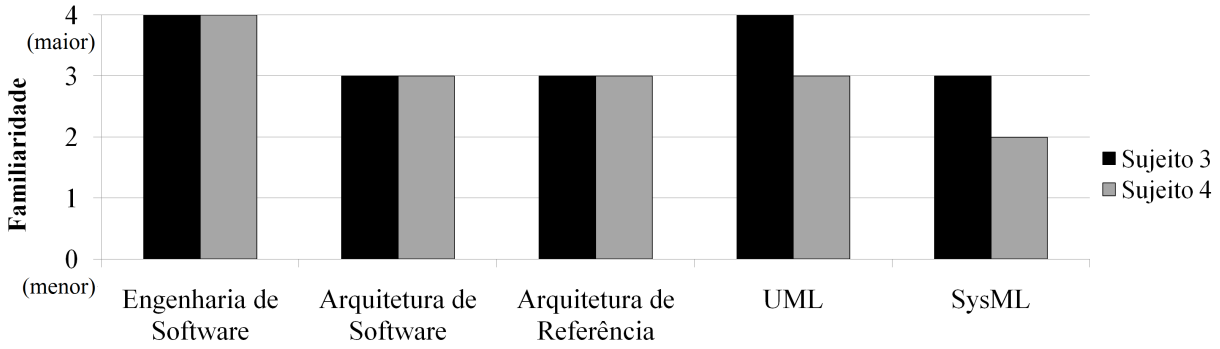

Figura 5.3: Nível de conhecimento dos participantes do grupo A

5.3.2.2 Análise do Grupo B

Na parte I e II do questionário foi solicitado dos sujeitos que completassem a leitura da representação da arquitetura de referência e respondessem a um conjunto de questões sobre o seu projeto, devendo informar também a dificuldade encontrada para responder cada questão. A Tabela 5.4 sumariza as respostas dos participantes do grupo B sobre a representação construída com o apoio do ProSA-Re. Da tabela, observa-se que um dos participantes obteve um ótimo desempenho na parte II do questionário acertando seis das sete questões propostas. Além disso, ambos os sujeitos apresentaram uma taxa de erros baixa, mesmo o sujeito 2 indicando que não sabia responder três das questões propostas. De modo geral, os participantes notaram um nível de dificuldade moderado para responder o conjunto de questões, sendo que o sujeito 2 encontrou mais dificuldade que o sujeito 1.

Tabela 5.4: Sumário das respostas dos participantes do grupo B

	Sujeito 1	Sujeito 2
Tempo gasto na Parte I	01:30	00:50
Tempo gasto na Parte II	01:00	00:40
Quantidade de acertos	6	3
Quantidade de erros	1	1
Questões que não soube responder	0	3

Na parte III do questionário, referente ao conjunto de questões específicas sobre a percepção da representação B, cada participante avaliou o conjunto de visões apresentadas com relação a três critérios: (i) utilidade, para medir o quanto uma visão arquitetural ajudou a responder as questões da parte II; (ii) esforço, para medir o trabalho necessário para uma visão arquitetural ser adequadamente analisada ou escaneada por uma informação específica; e (iii) clareza, para indicar o quanto uma visão arquitetural é confusa a ponto de não comunicar efetivamente aspectos relevantes da arquitetura de referência. A Figura 5.4 ilustra os resultados obtidos. Para o sujeito 1, por exemplo, com exceção da visão de variabilidade, todas as outras visões foram bastante úteis (notas três e quatro). Ele também considerou as visões de componentes colaborativos e de detalhes de decisões arquiteturais as mais trabalhosas. Por outro lado, o sujeito 2 considerou apenas as visões de componentes colaborativos e de variabilidade como úteis, e destacou as visões de componentes colaborativos e de detalhes de decisões arquiteturais como trabalhosas, embora em menor proporção.

Na sequência, na parte III do questionário sobre a percepção geral da representação, os sujeitos destacaram as atividades de leitura e busca por informações específicas como sendo as que tomaram mais tempo no estudo de caso. Com relação aos aspectos importantes da representação arquitetural, o sujeito 1 considerou que todos eles como essenciais para o entendimento, ou seja, diagramas, variedade de visões e descrição textual. Por outro lado, o sujeito 2 considerou apenas a visão primária como essencial, sendo a descrição textual de importância intermediária e a variedade de visões como sendo de pouco importância. Esse resultado é coerente com a resposta desse sujeito na avaliação das visões arquiteturais, uma vez que apenas duas das sete visões arquiteturais foram úteis para ele. Além disso, enquanto o sujeito 1 informou que conseguiu encontrar todas as informações solicitadas, o sujeito 2 destacou que não. De fato, na questão seguinte, o sujeito 2 explica que teve muita dificuldade em encontrar as informações solicitadas porque o documento era muito longo, enquanto para o sujeito 1 o fato de as questões terem sido apresentadas em uma sequência diferente daquela em que apareceram no documento dificultou muito a busca por informações específicas. Contudo, uma vez que o questionário foi projetado para ser aplicado nos dois tipos de representação avaliados, não era possível e nem interessante que

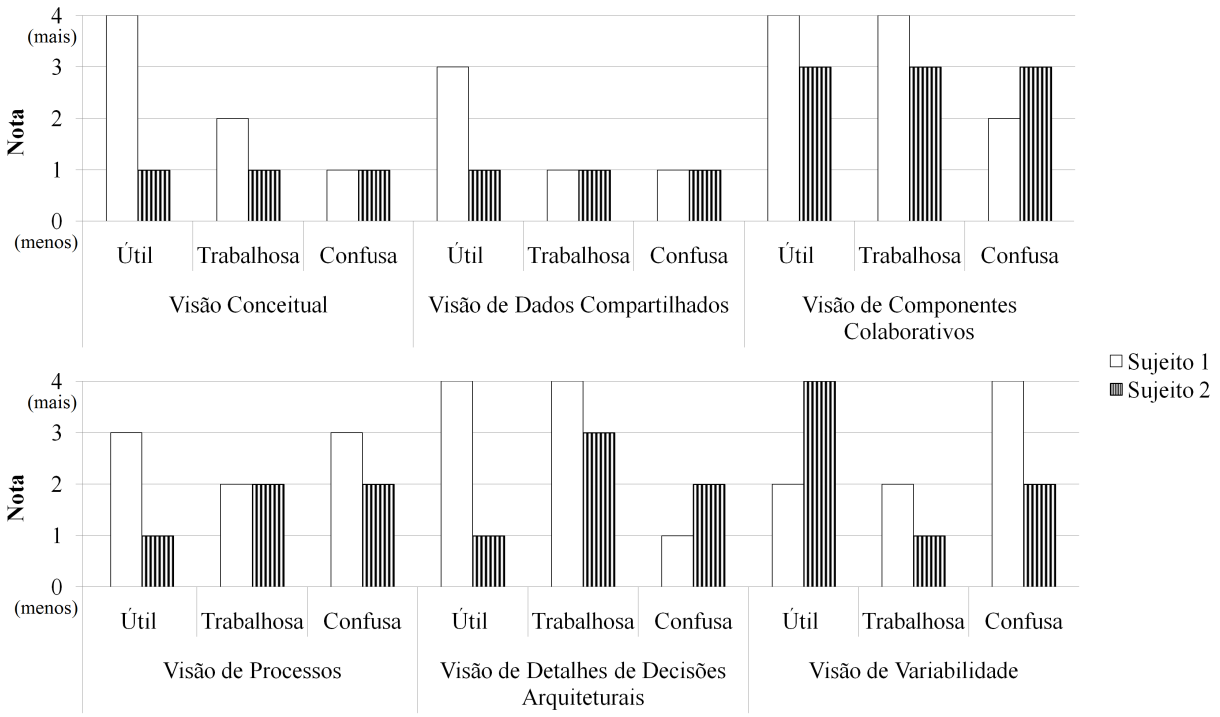

Figura 5.4: Avaliação do conjunto de visões arquiteturais da representação elaborada com auxílio do ProSA-Re

o questionário apresentasse as questões na mesma ordem do texto, já que a facilidade de acesso era um dos aspectos sob avaliação. Os participantes também escolheram o conjunto de aspectos positivos e negativos da representação arquitetural analisada. Os resultados dessas perguntas são ilustrados na Figura 5.5. Da figura, é possível observar que o tamanho e a notação empregados foram destacados como aspectos negativos, mas os participantes apresentaram informações opostas sobre os diagramas e a informação textual apresentada. É importante destacar ainda que a organização da representação foi indicada como sendo um dos aspectos positivos pelo sujeito 1. Sobre as considerações que poderiam melhorar o entendimento da arquitetura de referência e sua representação, os sujeitos indicaram a consulta a especialistas do domínio, em SysML, em arquitetura de referência e também a disponibilização de mais tempo para leitura e resposta do questionário.

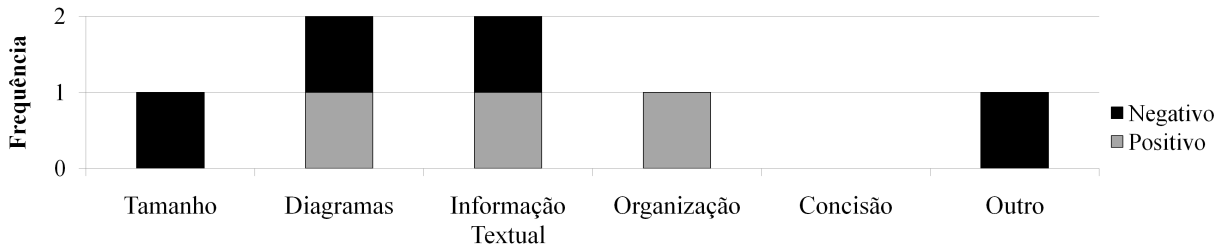

Figura 5.5: Avaliação de aspectos da representação elaborada com auxílio do ProSA-Re

Por fim, na parte IV do questionário foram feitas perguntas para avaliar as habilidades pessoais dos sujeitos. Os resultados obtidos são ilustrados na Figura 5.6. Da figura,

observa-se que o sujeito 1 apresenta sólidos conhecimentos em engenharia de software, arquitetura de software, arquitetura de referência e UML, possuindo conhecimento intermediário em SysML. Já o sujeito 2 possui sólidos conhecimentos em engenharia de software e UML, possuindo conhecimento intermediário nos tópicos sobre arquitetura de software e pouco conhecimento em SysML. Nesse sentido, a falta de conhecimentos em SysML pode ter sido um dos motivos para a dificuldade encontrada para a compreensão dos diagramas apresentados.

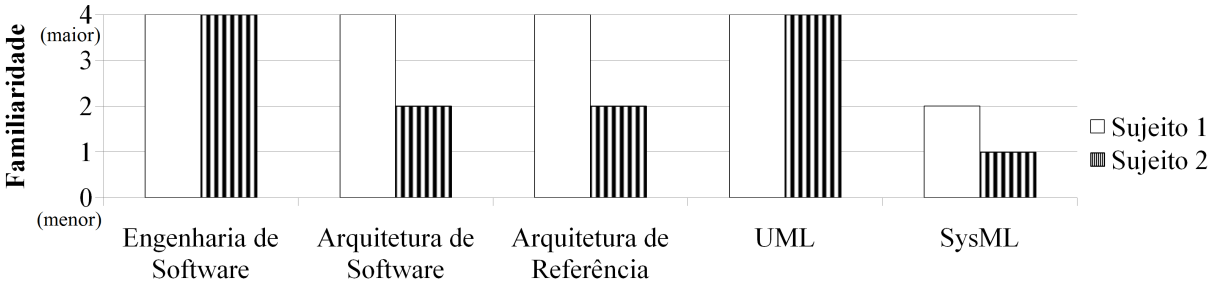

Figura 5.6: Nível de conhecimento dos participantes do grupo B

5.3.2.3 Análise Comparativa

A partir dos resultados obtidos pelos grupos A e B, é possível observar que um entendimento integral da arquitetura de referência apenas pode ser alcançado pela combinação dos três aspectos da representação arquitetural, ou seja, visões primárias, variedade de visões e descrição textual. Nesse sentido, a proposta do método ProSA-Re, em que cada visão deve apresentar uma visão primária e, se necessário, incluir visões adicionais para refinar o conhecimento inicial, permite uma organização melhor das informações na representação arquitetural, permitindo a criação de visões com diferentes níveis de detalhes dentro de um mesmo ponto de vista. A partir dos resultados obtidos no grupo A, é possível verificar que a proposta do método ProSA-Re atende às solicitações dos participantes e pode contribuir para um melhor entendimento da representação arquitetural.

Em uma outra perspectiva, considerando que as descrições textuais têm como propósito complementar e enriquecer as informações apresentadas nos diagramas, é interessante não sobrecarregar esse aspecto da representação. Nesse sentido, foi observado que a adoção de uma linguagem com maior expressividade permitiu a redução da descrição textual na representação arquitetural elaborada com o auxílio do ProSA-Re. Nesse sentido, embora os dois grupos tenham destacado o tamanho da representação arquitetural como um aspecto negativo, e em particular o tamanho da descrição textual, é importante destacar que a representação analisada pelo grupo B é 30% menor que a representação analisada pelo grupo A. Contudo, o uso de uma nova linguagem, nesse caso da SysML, ainda requer treinamento para que a linguagem possa ser corretamente empregada e compreendida por

todos os usuários da representação. Nesse sentido, pode ser interessante complementar a representação arquitetural com uma breve introdução sobre os principais elementos da linguagem, além de incluir uma legenda em cada diagrama construído. Apesar disso, os diagramas da representação B receberam nota menor no aspecto confusão, o que consiste em um importante avanço no sentido de promover a comunicação entre os diferentes interessados.

Por fim, vale destacar também que a variedade de visões arquiteturais apresentadas na representação B também foram destacadas como um importante aspecto positivo, já que permitiram a organização das informações em seções específicas e menores. De fato, observa-se que as visões da representação B foram, de modo geral, menos trabalhosas que as visões na representação original. Esse resultado é uma importante evidência de que o método ProSA-Re pode colaborar, de fato, para uma representação arquitetural mais compacta.

5.3.3 Discussão dos Resultados da Avaliação da Representação Arquitetural

A representação elaborada com o auxílio do método ProSA-Re apresenta importantes vantagens para a representação de arquiteturas de referência como, por exemplo, uma melhor organização das informações e menor tamanho da documentação. Contudo, a dificuldade apresentada pelos participantes para compreender as notações utilizadas indica que ainda é preciso aprimorar a visão primária das visões arquiteturais construídas, além de apresentar uma explicação mais completa da notação utilizada.

Dentre as limitações desse estudo de caso, destacam-se a falta de mais representações arquiteturais construídas com o apoio do ProSA-Re e também a aplicação do questionário *online*. Nesse sentido, seria interessante avaliar com qual frequência os participantes consultaram a arquitetura de referência, para ter uma ideia mais precisa de quanto tempo foi gasto, de fato, procurando informações na documentação, além de permitir um entendimento mais amplo sobre o como isso foi realizado pelos participantes. Essas informações seriam úteis para otimizar a organização das informações na representação.

De modo geral, espera-se que os resultados obtidos nesse estudo de caso motivem a adoção do método e, assim, permitam a replicação desse estudo de caso com outras arquiteturas de referência para o domínio de sistemas embarcados e possa também alcançar uma variedade maior de interessados.

5.4 Avaliação da Descrição do Método

Com o objetivo de avaliar a descrição do ProSA-Re e a viabilidade de sua adoção em projetos de pesquisa reais, foi realizada uma pesquisa de opinião com dois especialistas da área de arquitetura de software. Essa abordagem foi preferida uma vez que por limitações de tempo e pessoal não foi possível aplicar o método no projeto e representação de mais arquiteturas de referência por diferentes arquitetos. Por esse motivo, a abordagem escolhida se baseia na experiência e vivência desses especialistas para identificar possíveis vantagens e limitações que poderiam surgir da aplicação do método. O Anexo D apresenta o questionário utilizado para a coleta de informações.

5.4.1 Resultados da Avaliação com Especialistas

Na parte I do questionário foram feitas questões sobre as habilidades de cada especialista consultado e cujos resultados são apresentados na Figura 5.7. Observa-se que ambos os especialistas apresentam pouco ou algum conhecimento no domínio de sistemas embarcados, mas apresentam um conhecimento avançado sobre o domínio de arquiteturas de software e arquiteturas de referência. De fato, ambos trabalharam recentemente com alguma arquitetura de referência, sendo que o primeiro especialista consultado também tem experiência com a criação de arquiteturas de referência. O primeiro especialista possui grande experiência com a arquitetura de referência AUTOSAR e também com arquiteturas de referência para *sites* governamentais. Já o segundo possui experiência com a arquitetura de referência para comércio eletrônico da Microsoft. Vale destacar também que os especialistas apresentam conhecimento mediano a avançado na linguagem UML, mas possuem pouca experiência com a linguagem SysML.

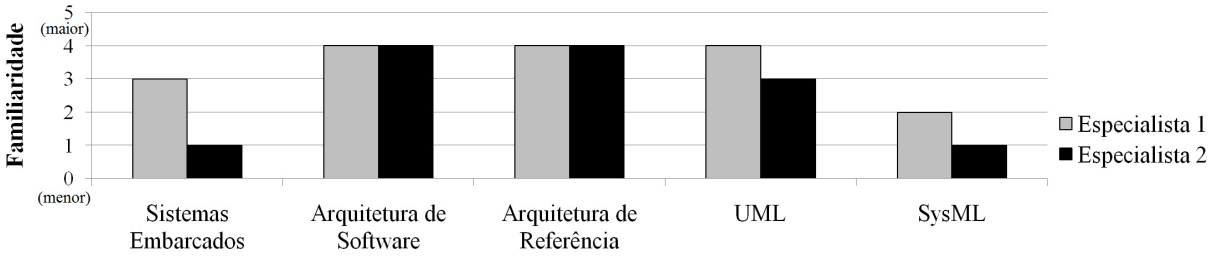

Figura 5.7: Nível de conhecimento dos especialistas consultados

Na parte II do questionário foram feitas perguntas para identificar a motivação por trás da proposta de um método para representação arquitetural específico para arquiteturas de referência do domínio de sistemas embarcados, sendo que as respostas obtidas de ambos os especialistas confirmam a relevância de tal proposta. O primeiro especialista destaca que arquiteturas de referências possuem interessados e interesses que diferem de

arquiteturas de software. O especialista também destaca que o papel da arquitetura de referência no desenvolvimento de sistemas de software e o cenário de aplicação dessas arquiteturas também são diferentes quando comparados ao de arquiteturas de software. Por exemplo, arquiteturas de referência precisam apoiar a construção de instâncias arquiteturais, enquanto isso não é requisitado de arquiteturas de software. Segundo o especialista, a representação de arquiteturas de referência e também o como elaborar essas representações precisam apoiar essas características. Para o segundo especialista, o desenvolvimento de sistemas embarcados pode ser bastante beneficiado pela adoção de arquiteturas de referência, pois o problema, ressalta ele, reside na falta arquiteturas de referência para esse domínio e também na falta de qualidade das arquiteturas de referência existentes. Nesse contexto, um método de apoio à representação de arquiteturas de referência pode fornecer os mecanismos necessários para garantir que a representação da arquitetura de referência seja elaborada corretamente, possibilitando que a arquitetura de referência seja útil durante todo o ciclo de desenvolvimento de software.

Na parte III do questionário foram feitas questões para avaliar a descrição do método com relação à clareza das informações apresentadas (descrição de entradas, saídas e tarefas). As respostas obtidas são ilustradas na Figura 5.8. Enquanto o primeiro especialista notou a falta de algumas informações, o segundo achou a descrição bastante completa. Dentre as sugestões de melhorias estão a descrição mais clara de entradas e saídas para cada tarefa, o detalhamento de mudanças nos estados dos produtos de trabalho, a elaboração de exemplos para cada atividade além do oferecimento de diretrizes mais precisas para a adoção do método.

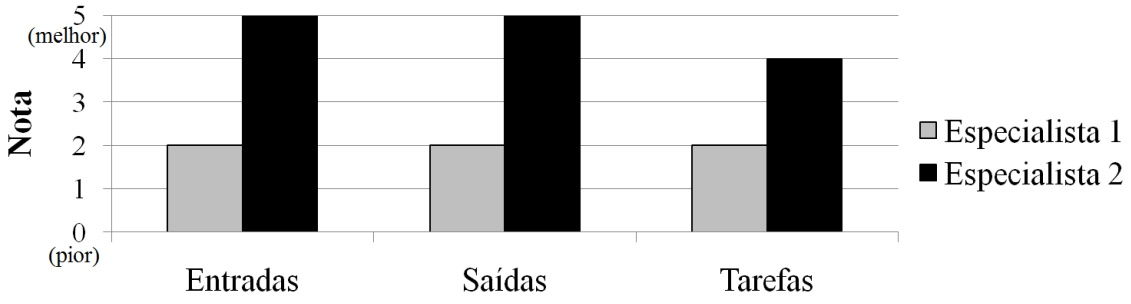

Figura 5.8: Avaliação da descrição do método por especialistas

Na parte IV do questionário, os especialistas avaliaram como seria uma provável representação construída com auxílio do ProSA-Re. Embora fosse preciso ver o método em ação para melhor opinar sobre a representação que poderia ser construída com o seu auxílio (opinião do primeiro especialista), o segundo especialista afirma que a existência do método pode colaborar para a criação de uma representação que adere melhor aos requisitos existentes, os quais são bastante afetados pelo contexto em que a arquitetura de referência está inserida. Para o primeiro especialista, interessados com pouca experi-

ência em engenharia de software podem ter dificuldade em entender as técnicas adicionais sugeridas pelo método como o diagrama de atividades ou o diagrama de máquina de estados. Contudo, segundo destaca o segundo especialista, o ponto de vista conceitual, por exemplo, pode ser mais facilmente compreendido por interessados com menor experiência em engenharia de software, uma vez que apresentam informações de alto nível sobre a arquitetura de referência e os seus objetivos. Com relação aos pontos de vista do primeiro grupo, o segundo especialista afirma de que se trata de um grupo essencial para todas as arquiteturas de referência de sistemas embarcados, sendo que o primeiro especialista destaca a importância do ponto de vista de variabilidade para as arquiteturas de referência. Considerando os demais grupos de pontos de vista, ambos os especialistas afirmam que eles também colaboram para o entendimento da arquitetura de referência, sendo que a importância de cada um deles depende da arquitetura de referência a ser construída. Dentre as ferramentas de apoio à linguagem SysML, os especialistas possivelmente adotariam as ferramentas Papyrus (uma extensão para o Eclipse), Rhapsody Developer (da empresa IBM) e Enterprise Architect (da empresa Sparx Systems), pesando nessa escolha o fato de essas ferramentas serem disponibilizadas pelas organizações na qual eles trabalharam e, portanto, já conhecidas.

Na parte V do questionário, os especialistas avaliaram possíveis benefícios e limitações que poderiam surgir com a utilização do método em projetos reais. O principal benefício destacado é a representação mais abrangente da arquitetura, sendo que o segundo especialista também afirma que o método pode aumentar a produtividade da atividade de representação, diminuir o esforço necessário para entender a representação, melhorar a clareza dos diagramas e ainda facilitar a manutenção e evolução da representação construída. Por outro lado, ambos os especialistas consultados concordam que o método também pode aumentar a burocracia para a criação da representação. Além disso, o primeiro especialista discorda do segundo em alguns pontos quando afirma que o método pode dificultar a manutenção e evolução da arquitetura de referência. Esse especialista também ressalta que os pontos de vista selecionados são insuficientes para capturar os interesses específicos de arquiteturas de referência e de sistemas embarcados. Sobre a influência do método na representação de arquiteturas de referência, os especialistas acreditam que ele pode apresentar um impacto moderado, uma vez que para ter um impacto maior seria preciso mostrar como as visões e pontos de vista tratam os interesses que são específicos para arquiteturas de referência e sistemas embarcados. Por exemplo, se um ponto de vista ajudar a pensar sobre o desempenho e a confiabilidade ou sobre o relacionamento entre artefatos de software e hardware, então o método pode significar um grande impacto.

Por fim, na parte VI do questionário, os especialistas opinaram sobre a viabilidade e perspectivas de utilização do ProSA-Re em projetos reais. Os dois especialistas consulta-

dos concordam que uma abordagem iterativa e incremental, inclusive para a representação de arquiteturas de referência, é o melhor modo de criar uma solução viável, uma vez os ambientes continuamente em mudança exigem atenção contínua para capturar adequadamente o seu conteúdo. Por esse motivo, ao adotar uma abordagem incremental é possível aplicar as lições aprendidas em cada ciclo. Em virtude das limitações encontradas anteriormente, o primeiro especialista acredita que o método pode contribuir parcialmente para a comunidade de sistemas embarcados, de modo que uma contribuição maior poderia ser alcançada com a realização das melhorias sugeridas e uma motivação mais clara da necessidade de cada atividade do método. De acordo com o segundo especialista consultado, o método apresenta grande potencial e foca em um problema atual muito claro do domínio de sistemas embarcados. Além disso, esse especialista ressalta que a abordagem incremental com poucos papéis é muito clara e concisa, tendo concordado com as representações e decisões tomadas.

5.4.2 Discussão dos Resultados da Avaliação com Especialistas

Os resultados da avaliação realizada com especialistas forneceram um importante panorama para modificações e melhoramentos no método com relação ao conjunto de atividades propostas e também ao material de apoio disponibilizado. De modo geral, optou-se por não definir detalhadamente os passos necessários para executar cada tarefa para conferir maior liberdade aos arquitetos que trabalharão na elaboração da representação arquitetural. É importante destacar que várias melhorias sugeridas foram incorporadas à definição do método apresentada no capítulo anterior.

Seria interessante também que a avaliação do método tivesse considerado a opinião de pessoas que empregaram o método na prática para a representação de arquiteturas de referência de sistemas embarcados. Contudo, essa abordagem não foi adotada uma vez que o método proposto ainda não foi utilizado fora do escopo deste projeto de pesquisa devido, principalmente, às limitações de tempo e pessoal disponível. No entanto, espera-se que isso seja realizado a partir da adoção do método para a representação de outras arquiteturas de referência para o domínio de sistemas embarcados, o que deverá contribuir ainda mais para o amadurecimento do conjunto de atividades proposto.

5.5 Considerações Finais

Neste capítulo, foi apresentado o estudo de caso conduzido para avaliação da representação arquitetural obtida com o auxílio do ProSA-Re. Em seguida, também foi apresentada a investigação feita com especialistas da área de arquitetura de software para avaliar a descrição e a relevância do método. Observa-se que os resultados alcançados nessas

avaliações mostram importantes contribuições para a usabilidade e compreensibilidade da representação de arquiteturas de referência de sistemas embarcados, sendo que melhorias na descrição do método foram implementadas para facilitar a sua correta utilização pela comunidade.

Capítulo

6

Conclusão

6.1 Caracterização da Pesquisa Realizada

Sistemas embarcados têm ganho cada vez mais destaque ao desempenharem funções mais sofisticadas e complexas. Em especial, observa-se que os sistemas desenvolvidos para esse domínio também devem ser capazes de se adaptar às rápidas mudanças de plataformas e infraestrutura, além de atender níveis cada vez elevados de qualidade e confiança em ciclos de inovação cada vez menores. Nesse contexto, têm-se que a proposição de arquiteturas de referência pode representar uma importante contribuição para o domínio de sistemas embarcados ao promoverem o reúso do projeto arquitetural e o entendimento sobre os sistemas de software construídos. De fato, diversas arquiteturas de referência podem ser encontradas na literatura para o domínio de sistemas embarcados. Em especial, descrever a arquitetura de referência de modo que todos os interessados nela possam compreendê-la é essencial, principalmente porque elas serão a base por meio da qual todo um conjunto de sistemas de software será desenvolvido. No entanto, verifica-se que, na prática, a descrição de arquiteturas de referência ainda é, muitas vezes, realizada informalmente. Além disso, existe uma carência de trabalhos que tenham investigado quais abordagens e técnicas são mais adequadas para a descrição de arquiteturas de referência, em especial, para o domínio de sistemas embarcados, inclusive sobre o como realizar uma descrição adequada dessas arquiteturas.

Nesse contexto, neste projeto de mestrado foi estabelecido um método para a representação de arquiteturas de referência de sistemas embarcados que compreende desde a identificação do escopo dessas arquiteturas e dos principais tipos de informação do domínio que precisam ser capturados, até a elaboração das visões arquiteturais e a avaliação da representação. O método denominado ProSA-Re inclui a sistematização do conjunto de tarefas que precisam ser executadas para a construção da representação arquitetural, os papéis envolvidos nessa construção e também o conjunto de produtos de trabalho consumidos e produzidos pela execução dessas tarefas. O método contém também diretrizes e orientações para guiar a sua adoção durante o ciclo de vida da arquitetura de referência.

O método foi avaliado tanto pela perspectiva dos usuários da representação da arquitetura de referência quanto pela perspectiva dos executores do método, ou seja, arquitetos e desenvolvedores envolvidos na construção da representação. Os resultados obtidos em ambas as perspectivas evidenciaram aspectos positivos que podem ser alcançados com o apoio do método e colaboraram para uma melhor descrição das tarefas e diretrizes propostas. Espera-se que os resultados obtidos pelo estudo de caso contribuam para promover a adoção do método em diferentes e novos projetos e que as representações elaboradas com o auxílio do ProSA-Re ajudem a melhorar e promover o entendimento sobre arquiteturas de referência construídas para o domínio de sistemas embarcados.

Vale destacar que este projeto foi desenvolvido em parceria com o Prof. Dr. Flavio Oquendo[1], especialista na área de Arquitetura de Software na Université de Bretagne Sud, França, onde ele coordena o grupo de pesquisa em arquitetura de software ARCHLOG[2] do laboratório de pesquisa IRISA. O Prof. Flavio tem conduzido pesquisas na área de linguagens, processos e ferramentas para o projeto, construção e evolução de sistemas de software baseados em arquiteturas orientadas a serviço e tecnologias baseadas em componentes, visando aplicações em domínios de informação e plataformas diferentes. Nesse contexto, o Prof. Flavio contribuiu com importantes conhecimentos sobre o desenvolvimento de sistemas de software e experiência na aplicação de linguagens para descrição arquitetural.

6.2 Contribuições

Como principais contribuições deste trabalho, tem-se:

- Contribuição para a área de Sistemas Embarcados com um método que visa facilitar e aprimorar a representação de arquiteturas de referência desse domínio ao estabele-

[1] http://www-irisa.univ-ubs.fr/flavio.oquendo/
[2] http://www-irisa.univ-ubs.fr/ARCHWARE/ARCHLOG/

cer um conjunto de atividades, pontos de vista e interesses que auxiliam na criação de representações arquiteturais adequadas;

- Contribuição para a área de Arquitetura de Software com um método que sistematiza a representação de arquiteturas de referência e que, como citado anteriormente, não havia sido ainda investigado pela comunidade científica;

- Contribuição para projetos de pesquisa maiores, mais especificamente para o INCT-SEC, para o projeto "Uma Contribuição ao Estabelecimento e Avaliação de Arquiteturas de Referência para Sistemas Embarcados" e para o projeto "ProSA-RAES: Subsídios para o Estabelecimento de Arquiteturas de Referência para Sistemas Embarcados"; e

- Publicação de artigos científicos em conferências e periódicos de relevância internacional para a área de concentração, visando divulgar os resultados obtidos pela condução deste trabalho.

6.3 Dificuldades e Limitações

Dentre as principais dificuldades encontradas para a realização deste trabalho, pode-se destacar a necessidade de investigação e estudo das linguagens SPEM (empregada na definição do método) e SysML (empregada nos diagramas da representação construída com o apoio do ProSA-Re). Além disso, a maior dificuldade encontrada foi para a análise e compreensão da arquitetura de referência, descrita em uma tese de doutorado, e que foi selecionada como objeto no estudo de caso. Um estudo detalhado sobre o projeto dessa arquitetura de referência foi necessário para garantir que a representação arquitetural construída com o auxílio do ProSA-Re correspondia, de fato, à arquitetura de referência proposta originalmente.

Em particular, como foi preciso garantir a consistência entre as duas representações analisadas no estudo de caso, ou seja, garantir que as duas representações analisadas continham as mesmas informações, a elaboração da representação com o auxílio do ProSA-Re foi baseada na representação arquitetural original já disponível. Por esse motivo, nem todos as atividades propostas pelo método foram aplicadas como, por exemplo, a definição de escopo, sendo que a criação das visões arquiteturais foi baseada nas informações já documentadas na representação original, como a visão de detalhes de decisões arquiteturais. Assim, não foi possível conduzir uma investigação mais precisa sobre a real extensão do método e seu impacto na representação arquitetural construída com o seu apoio, porém a arquitetura de referência selecionada permitiu a criação de uma descrição abrangente,

no sentido de que várias visões arquiteturais puderam ser construídas e diferentes níveis de detalhes puderam ser representados.

6.4 Trabalhos Futuros

A realização deste projeto possibilitou identificar diferentes e importantes direções para trabalhos futuros a serem desenvolvidos sobre a representação de arquiteturas de referência de sistemas embarcados. Dentre eles, destacam-se:

- Entender melhor a relação entre os pontos de vista e os interessados na arquitetura de referência;
- Identificar os interessados na arquitetura de referência, para viabilizar a seleção mais informada de um conjunto minimal de interesses que precisam ser capturados pela descrição arquitetural;
- Relacionar os pontos de vista selecionados pelo método ProSA-Re aos interesses identificados pela revisão sistemática para a representação das arquiteturas do domínio de sistemas embarcados;
- Consolidar os resultados obtidos nesta dissertação com a proposição de um *framework* arquitetural para a representação arquiteturas de referência de sistemas embarcados. Esse *framework* pode assegurar, dentre outras coisas, a rastreabilidade entre os elementos descritos em diferentes pontos de vista e também servir para empacotar todas as possíveis descrições arquiteturais que podem existir, sendo compatível com o padrão ISO/IEC/IEEE 42010;
- Investigar a possibilidade de utilização do ProSA-Re para a representação de outras arquiteturas de referência para o domínio de sistemas embarcados e também de outros domínios;
- Investigar como o ProSA-Re pode ser melhor integrado ao ProSA-RAES;
- Criar um perfil SysML para especializar os elementos dessa linguagem com estereótipos específicos para o domínio de sistemas embarcados como, por exemplo, hardware, software, processos, etc.;
- Investigar a adoção de blocos de restrição da linguagem SysML para a análise de arquiteturas de referência de sistemas embarcados;

- Investigar técnicas da SysML que podem ser empregadas para a representação de decisões arquiteturais, em especial considerando o *framework* arquitetural proposto por van Heesch et al. (2012); e
- Utilizar o ProSA-Re para o projeto e representação de arquiteturas de referência na indústria.

Vale destacar ainda que a condução desta pesquisa também motivou a inscrição da aluna no doutorado, no qual ela pretende continuar as pesquisas na área de arquitetura de software, em virtude de ser uma área que ainda precisa ser mais investigada e consolidada.

Referências

Allen, R. J. *A Formal Approach to Software Architecture.* Tese de Doutoramento, Carnegie Mellon University, 1997.

Angelov, S.; Grefen, P.; Greefhorst, D. A Classification of Software Reference Architectures: Analyzing Their Success and Effectiveness . In: *Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2009)*, Cambridge, Reino Unido: IEEE Computer Society, 2009, p. 141–150.

Angelov, S.; Trienekens, J.; Grefen, P. An e-contracting reference architecture. *Journal of Systems and Software*, v. 81, n. 11, p. 1816–1844, 2008.

ATESST EAST-ADL Domain Model Specification v. 2.1. [*On-line*], *World Wide Web*, Disponível em `http://www.east-adl.info/repository/EAST-ADL2.1/EAST-ADL-Specification_2010-06-30.pdf` (Acessado em 23/01/2013), 2010.

AUTOSAR AUTOSAR Glossary. [*On-line*], *World Wide Web*, Disponível em `http://autosar.org/download/R4.0/AUTOSAR_TR_Glossary.pdf` (Acessado em 23/01/2013), 2010.

AUTOSAR AUTOSAR (AUTomotive Open System ARchitecture). [*On-line*], *World Wide Web*, disponível em `http://www.autosar.org/` (Acessado em 21/01/2013), 2013.

Babar, M. A.; Dingsøyr, T.; Lago, P.; van Vliet, H., eds. *Software Architecture Knowledge Management: Theory and Practice.* Springer, 279 p., 2009.

Barcelos, R. F.; Travassos, G. H. ArqCheck: Uma abordagem para inspeção de documentos arquiteturais baseada em checklist. In: *V Simpósio Brasileiro de Qualidade de Software (SBQS 2006)*, Vitória-ES, 2006, p. 175–189.

Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice.* Addison-Wesley, 528 p., 2003.

Batory, D.; Coglianese, L.; Goodwin, M.; Shafer, S. Creating reference architectures: an example from avionics. In: *Symposium on Software Reusability (SSR 95)*, Seattle, EUA, 1995, p. 27–37.

Bernardo, M.; Ciancarini, P.; Donatiello, L. Detecting architectural mismatches in process algebraic descriptions of software systems. In: *Working IEEE/IFIP Conference on Software Architecture (WICSA 2001)*, Amsterdam, Holanda, 2001, p. 77–86.

Biehl, M.; Törngren, M. An executable design decision representation using model transformations. In: *XXXVI EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010)*, Lille, França, 2010, p. 131 –134.

Borde, E.; Haik, G.; Pautet, L. Mode-based reconfiguration of critical software component architectures. In: *Design, Automation Test in Europe Conference Exhibition (DATE 2009)*, Dresden, Alemanha, 2009, p. 1160–1165.

Borg, Z. *A Reference Architecture for Marine Systems.* Dissertação de Mestrado, University of Kaiserslautern and Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Alemanha, 2011.

Bosch, J. Software architecture: the next step. In: *I European Workshop on Software Architecture (EWSA 2004)*, 2004, p. 194–1999 (LNCS v. 3047).

Broy, M. Challenges in Automotive Software Engineering. In: *XXXI International Conference on Software Engineering (ICSE 2006)*, Shanghai, China, 2006, p. 33–42.

Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. *Pattern-oriented Software Architecture: A System of Patterns*, v. 1. John Wiley & Sons, 1996.

Clarke, S.; Baniassad, E. *Aspect-oriented analysis and design: The theme approach.* Addison-Wesley, 2005.

Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Merson, P.; Nord, R.; Stafford, J. *Documenting Software Architectures: Views and Beyond.* 2 ed. Addison-Wesley, 512 p., 2011.

Cloutier, R.; Muller, G.; Verma, D.; Nilchiani, R.; Hole, E.; Bone, M. The Concept of Reference Architectures. *Systems Engineering*, v. 13, n. 1, p. 14–27, 2010.

Dybå, T.; Kitchenham, B.; Jorgensen, M. Evidence-based software engineering for practitioners. *IEEE Software*, v. 22, n. 1, p. 58–65, 2005.

Eickelmann, N. S.; Richardson, D. J. An evaluation of software test environment architectures. In: *Proc. of the XVIII International Conference on Software Engineering (ICSE 1996)*, Berlin, Alemanha: IEEE Computer Society, 1996, p. 353–364.

Eixelsberger, W.; Gall, H. Describing software architectures by system structure and properties. In: *XXII International Computer Software and Applications Conference (COMPSAC 1998)*, Vienna, Austria, 1998, p. 106–111.

Eklund, U.; Örjan Askerdal; Granholm, J.; Alminger, A.; Axelsson, J. Experience of introducing reference architectures in the development of automotive electronic systems. *SIGSOFT Softw. Eng. Notes*, v. 30, n. 4, p. 1–6, 2005.

Eklund, U.; Jonsson, N.; Eriksson, A.; Bosch, J. A Reference Architecture Template for Software-Intensive Embedded Systems. In: *Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (WICSA/ECSA 2012)*, Helsinki, Finlândia, 2012, p. 104–111.

Farenhorst, R.; de Boer, R. C. Knowledge Magament in Software Architecture: State of the Art. In: Babar, M. A.; Dingsøyr, T.; Lago, P.; van Vliet, H., eds. *Software Architecture Knowledge Management Theory and Practice*, Springer, p. 21–38, 2009.

Feiler, P. H.; Gluch, D. P.; Hudak, J. J. *The Architecture Analysis & Design Language (AADL): An Introduction.* Relatório Técnico CMU/SEI-2006-TN-011, Carnegie Mellon University, Disponível em `http://www.sei.cmu.edu/reports/06tn011.pdf` (Acessado em 23/01/2013), 2006.

Friedenthal, S.; Moore, A.; Steiner, R. *A Practical Guide to SysML: The Systems Modeling Language.* Morgan Kaufmann, 560 p., 2008.

Garlan, D.; Monroe, R. T.; Wile, D. Acme: An Architecture Description Interchange Language. In: *VII Centre for Advanced Studies on Collaborative Research (CASCON 1997)*, Toronto, Canadá, 1997, p. 1–15.

Garlan, D.; Perry, D. Introduction to the special issue on software architecture. *IEEE Transactions on Software Engineering*, v. 21, n. 4, p. 269–274, 1995.

Gilles, O.; Hugues, J. A mde-based optimisation process for real-time systems. In: *XIII IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC 2010)*, Carmona, Espanha, 2010, p. 50 –57.

Graaf, B.; van Dijk, H.; van Deursen, A. Evaluating an embedded software reference architecture - industrial experience report. In: *IX European Conf. on Software Maintenance and Reengineering (CSMR 2005)*, Manchester, Reino Unido, 2005, p. 354–363.

Gray, D. E. *Doing Research in the Real Word*. SAGE Publications, 422 p., 2004.

Grosskurth, A.; Godfrey, M. W. A Reference Architecture for Web Browsers. In: *XXI IEEE International Conference on Software Maintenance (ICSM 2005)*, Budapeste, Hungria, 2005, p. 1–4.

Guessi, M.; Oquendo, F.; Nakagawa, E. Y. Architectural Description of Embedded Systems: a Systematic Review. In: *III International ACM Sigsoft Symposium on Architecting Critical Systems Federated with CompArch (ISARCS 2012)*, Bertinoro, Itália, (submetido), 2012, p. 1–10.

Harrison, W.; Ossher, H.; Tarr, P. Software engineering tools and envrionments: a roadmap. In: *XXII International Conference on Software Engineering (ICSE 2000)*, Limerick, Irlanda, 2000, p. 261–277.

van Heesch, U.; Avgeriou, P.; Hilliard, R. A documentation framework for architecture decisions. *Journal of Systems*, v. 85, p. 796–820, 2012.

Heidary, K. Autosar technical overview and future development roadmap. In: *III AUTOSAR Open Conference*, Frankfurt, Alemanha, 2011, p. 1–13.

Hofmeister, C.; Nord, R.; Soni, D. *Applied Software Architecture*. 1ª. Addison Wesley, 397 p., 2000.

Honeywell Technical and Historical Overview of MetaH. [*On-line*], *World Wide Web*, Disponível em `http://aadl.sei.cmu.edu/aadl/documents/Technical%20and%20Historical%20Overview%20of%20MetaH.pdf` (Acessado em 21/01/2013), 2000.

ISO ISO/IEC/IEEE 42010 - Recommended Practice for Architectural Description of Software-Intensive Systems. 2010.

ITU Recommendation Z.100, Specification and Description Language (SDL). [*On-line*], *World Wide Web*, Disponível em `http://www.sdl-forum.org/` (Acessado em 21/01/2013), 1994.

Ivers, J.; Clements, P.; Garlan, D.; Nord, R.; Schmerl, B.; Silva, J. R. O. *Documenting Component and Connector Views with UML 2.0*. Relatório Técnico CMU/SEI-2004-TR-008, Disponível em `http://www.sei.cmu.edu/reports/04tr008.pdf` (Acessado em 23/01/2013), 2004.

Khalgui, M.; Hanisch, H.-M.; Gharbi, A. Model-checking for the functional safety of control component-based heterogeneous embedded systems. In: *IEEE Conference on Emerging Technologies Factory Automation (ETFA 2009)*, 2009, p. 1 –10.

Kitchenham, B. A. *Procedures for performing systematic reviews.* Relatório Técnico TR/SE-0401 e NICTA Technical Report 0400011T.1, Keele University and National ICT Australia Ltd, 2004.

Knight, J. C. Safety critical systems: challenges and directions. In: *International Conference on Software Engineering (ICSE 2002)*, Orlando, USA, 2002, p. 547–550.

Kruchten, P. Architectural Blueprints - The "4+1"View Model of Software Architecture. *IEEE Software*, v. 12, n. 6, p. 42–50, 1995.

Kruchten, P. Documentation of Software Architecture from a Knowledge Management Perspective – Design Representation. In: Babar, M. A.; Dingsøyr, T.; Lago, P.; van Vliet, H., eds. *Software Architecture Knowledge Management Theory and Practice*, Springer, p. 39–57, 2009.

Kruchten, P.; Obbink, H.; Stafford, J. The Past, Present and Future of Software Architecture. *IEEE Software*, v. 23, n. 2, p. 22–30, 2006.

Langsweirdt, D.; Boucké, N.; Berbers, Y. Architecture-driven development of embedded systems with acol. In: *XIII IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW 2010)*, Carmona, Espanha, 2010, p. 138 –144.

Liggesmeyer, P.; Trapp, M. Trends in Embedded Software Engineering. *IEEE Software*, v. 3, n. 3, p. 19–25, 2009.

Luckham, D. C. Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Orderings of Events. In: *Partial Order Methods Workshop (POMIV 1996)*, New Jersey, USA, 1996, p. 1–25 (*DIMACS*, v.29).

Marwedel, P. *Embedded System Design.* 2 ed. Springer, 389 p., 2011.

Medvidovic, N. Software architectures and embedded systems: a match made in heaven? *IEEE Software*, v. 22, n. 5, p. 83–86, 2005.

Medvidovic, N.; Taylor, R. N. A Classification and Comparison Framework for Software Architecture Description Languages. *IEEE Transactions on Software Engineering*, v. 26, n. 1, p. 70–93, 2000.

Mens, T.; Magee, J.; Rumpe, B. Evolving Software Architecture Descriptions of Critical Systems. *Computer*, v. 43, p. 42–48, 2010.

Mesman, B.; Spaanenburg, B.; Brinksma, E.; Deprettere, E.; Verhulst, E.; Timmer, F.; van Gageldonk, H.; Eggermont, L. D. J.; van Leuken, R.; Krol, T.; Hendriksen, W. *Embedded Systems Roadmap 2002*. Utrecht, Holanda: Technology Foundation (STW), 178 p., 2002.

Muller, G. A Reference Architecture Primer. [*On-line*], *World Wide Web*, Relatório Técnico do Gaudi Project, disponível em `http://www.gaudisite.nl/ReferenceArchitecturePrimerPaper.pdf` (Acessado em 21/01/2013), 2008.

Nakagawa, E. Y.; Antonino, P. O.; Becker, M. Reference Architecture and Product Line Architecture: A Subtle But Critical Difference. In: *V European Conference on Software Architecture (ECSA 2011)*, Essen, Alemanha, 2011a, p. 207–211 (LNCS v. 6903).

Nakagawa, E. Y.; Barbosa, E. F.; Fioravanti, M. L.; Maldonado, J. C. ProSA-RA: A process for the design, representation, and evaluation of aspect-oriented reference architectures. *Journal of Systems and Software*, p. 1–40, (submetido), 2011b.

Nakagawa, E. Y.; Ferrari, F. C.; Sasaki, M. M. F.; Maldonado, J. C. An aspect-oriented reference architecture for Software Engineering Environments. *Journal of Systems and Software*, v. 84, n. 10, p. 1670–1684, 2011c.

Nakagawa, E. Y.; Galster, M.; Guessi, M.; Santos, J. F. A Checklist-based Framework for Reference Architecture Evaluation. In: *European Conference on Software Architecture (ECSA 2013)*, Montpellier, France, (a ser submetido), 2013, p. 1–10.

Nakagawa, E. Y.; Maldonado, J. C. Reference Architecture Knowledge Representation: An Experience. In: *III Internacional Workshop on Sharing and Reusing Architectural Knowlegde (SHARK 2008)*, Leipzig, Alemanha, 2008, p. 51–54.

Nakagawa, E. Y.; Martins, R.; Felizardo, K.; Maldonado, J. C. Towards a process to design aspect-oriented reference architectures. In: *XXXV Latin American Informatics Conference (CLEI'2009)*, Pelotas, Brasil, 2009, p. 1–10.

Nakagawa, E. Y.; Oquendo, F.; Becker, M. RAModel: A Reference Model for Reference Architectures. In: *X Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) & VI European Conference on Software Architecture (ECSA)*, 2012, p. 297–301.

Nakagawa, E. Y.; Simão, A. S.; Ferrari, F.; Maldonado, J. C. Towards a reference architecture for software testing tools. In: *Proc. of the XIX International Conference on Software Engineering and Knowledge Engineering (SEKE 2007)*, Boston, EUA, 2007, p. 1–6.

NASA ESDS Working Group NASA ESDS Reference Architecture v 1.1. [*On-line*], *World Wide Web*, disponível em `http://earthdata.nasa.gov/sites/default/files/field/document/ESDS%20Reference%20Architecture%20v1.1.pdf` (Acessado em 21/01/2013), 2012.

Niz, D. *Diagrams and Languages for Model-Based Software Engineering of Embedded Systems: UML and AADL.* Relatório técnico, SEI, Disponível em `http://www.sei.cmu.edu/library/assets/UML_AADL_Comparison.pdf` (Acessado em 23/01/2013), 2007.

OASIS Standard *OASIS: Reference model for service oriented architecture.* Relatório Técnico SOA-RM, Disponível em `http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf` (Acessado em 23/01/2013), 2006.

Oliveira, L. B. R.; Felizardo, K. R.; Feitosa, D.; Nakagawa, E. Y. Reference Models and Reference Architectures Based on Service-Oriented Architecture: A Systematic Review. In: *IV European Conference on Software Architecture (ECSA 2010)*, Copenhagen, Dinamarca, 2010, p. 1–16.

OMG Software and Systems Process Engineering Meta-Model Specification v 2.0. [*On-line*], *World Wide Web*, Disponível em `http://www.omg.org/spec/SPEM/2.0/PDF` (Acessado em 21/01/2013), 2008.

OMG UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems. [*On-line*], *World Wide Web*, Disponível em `http://www.omg.org/spec/MARTE/1.0/PDF/` (Acessado em 21/01/2013), 2009.

OMG Systems Modeling Language v 1.2. [*On-line*], *World Wide Web*, Disponível em `http://www.omgsysml.org/` (Acessado em 21/01/2013), 2010a.

OMG Unified Modeling Language v. 2.3. [*On-line*], *World Wide Web*, Disponível em `http://www.omg.org/spec/UML/2.3/` (Acessado em 21/01/2013), 2010b.

OMG CORBA CCM. [*On-line*], *World Wide Web*, Disponível em `http://www.omg.org/spec/CORBA/3.2/` (Acessado em 21/01/2013), 2011.

Papazoglou, M. P.; Heuvel, W.-J. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, v. 16, n. 3, p. 389–415, 2007.

SAE Architecture Analysis & Design Language. [*On-line*], *World Wide Web*, disponível em `http://www.aadl.info/` (Acessado em 21/01/2013), 2011.

Shaw, M.; Garlan, D. *Characteristics of higher-level languages for software architecture.* Relatório Técnico CMU/SEI-87-TR-02, Carnegie Mellon University, Disponível em `http://www.sei.cmu.edu/reports/94tr023.pdf` (Acessado em 23/01/2013), 1994.

Tan, T. K.; Raghunathan, A.; Jha, N. K. Software architectural transformations: A new approach to low energy embedded software. In: *Conference on Design, Automation and Test in Europe (DATE 2003)*, Munich, Alemanha, 2003, p. 1–6.

UniversAAL Project The UniversAAL Reference Architecture. [*On-line*], *World Wide Web*, disponível em `http://www.universaal.org/images/stories/deliverables/D1.3-B.pdf` (Acessado em 21/01/2013), 2011.

Wasserman, A. I. Towards a discipline of software engineering. *IEEE Software*, v. 13, n. 6, p. 23–31, 1996.

Weyns, D. *An Architecture-Centric Approach for Software Engineering with Situated Multiagent Systems.* Tese de Doutoramento, Katholieke Universiteit Leuven, Disponível em `http://people.cs.kuleuven.be/~danny.weyns/PhD/PhD-thesis-Danny-Weyns.pdf` (Último acesso 15/01/2013), 2006.

Wolf, W. *Computers as Components – Principle of Embedded Computing System Design.* 2 ed. Morgan Kaufman, 507 p., 2008.

Ye, J.; Dong, W.; Qi, Z. A method to generate embedded real-time system test suites based on software architecture specifications. In: *IX International Conference for Young Computer Scientists (ICYCS 2008)*, Zhang Jia Jie, Hunan, 2008, p. 2325–2329.

Yin, R. K. *Case Study Research: Design and Methods.* 2ª ed. Thousand Oaks, CA: SAGE Publications, 1994.

Zhao, W.; Kearney, D. Deriving Architectures of Web-Based Applications. In: *Proc. of the VI Asia-Pacific Web Conference (APWeb 2003)*, Xian, China, 2003, p. 301–312 (LNCS v.2642).

Anexo

A

SysML

SysML is a general-purpose graphical modeling language that supports the analysis, specification, design, verification, and validation of complex systems. This annex content is extracted from Friedenthal et al. (2008) and OMG (2010a), and introduces the main characteristics of SysML. The language is intended to help specify and architect systems and specify its components that can then be designed using other domain-specific languages such as UML for software design and VHDL for hardware design.

SysML is specified using a combination of UML modeling techniques and precise natural language. Nevertheless, the use of more formal constraints and semantics may be applied in future versions of the language to further increase the precision of the language. The version 2.3 of the SysML profile is organized into the following discrete language units that extend the UML language:

- *Requirements* specify textual requirements and their relationships to each other and to models;
- *Blocks* define system structure and properties;
- *Activities* define extensions to UML activities to support continuous behavior;
- *Constraint Blocks* that add parametric models; and
- *Ports and Flows* extend the UML structural model and add support to definition of information, matter, and energy flows between system elements.

Overall, SysML defines nine diagram types categorized into behavior diagrams, structural diagrams, and requirements diagram. Each diagram type is briefly explained below.

The **Block Definition Diagram** represents structural elements called blocks, and their composition and classification (modification of UML class diagram). The *Block* can be used to represent hardware, software, facilities, personnel, or any other system element. The block definition diagram also introduces a new type of block called a constraint block, which defines constraints in terms of equations and their parameters.

The **Internal Block Diagram** represents interconnections and interfaces between the parts of a block (modification of UML composite structure diagram). A usage (or part) is the same as a role. The parts in the diagram are indicated by the colon (:) notation. A part enables the same block to be reused in many different contexts and be uniquely identified by its usage. Each part may have unique behaviors, properties, and constraints that apply to its particular usage. It is called an internal block diagram because it represents the internal structure of a higher level block. Ports specify interaction points with other parts and are represented by small squares on the boundary of the parts. Connectors define how the parts connect to one another via their ports and are represented as the lines between the ports. Parts can also be connected without ports as indicated by some of the interface definitions.

The **Parametric Diagram** represents constraints on property values, such as performance, reliability, or weight, which can be used to support engineering analysis (not in UML). The parametric diagram shows a network of constraints (equations). Each constraint is a usage of a constraint block defined in the block definition diagram. The parametric diagram is expected to be executed in an engineering analysis tool to provide the results of the analysis. This may be a separate specialized analysis tool that is not provided by the SysML modeling tool, such as a simple spreadsheet or a high-fidelity performance simulation depending on the need. The analysis results from the execution then provide values that can be incorporated back into the SysML model.

The **Package Diagram** represents the organization of a model in terms of packages that contain model elements (same as UML package diagram). Each package contains a set of model elements, which can be related to model elements in another package. However, model organization enables each model element to be uniquely identified by the package that contains it. This facilitates understandability, access control, and change management of the model.

The **Requirements Diagram** captures the requirements hierarchies, in which each requirement can relate to other requirements, design elements, and test cases using relationships such as derive, satisfy, verify, refine, and trace. These relationships can be used to establish requirements traceability with a high degree of granularity. The direction of

the arrows in this diagram shows the dependency from leafs to root requirements, in the sense that if the root requirement changes, all derived requirement should also change.

The **Activity Diagram** represents behavior in terms of the ordering of actions based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs (modification of UML activity diagram). The activity diagramis effective for communicating continuous types of behaviors.

The **Sequence Diagram** represents behavior in terms of a sequence of messages exchanged between parts (same as UML sequence diagram). The sequence diagram is effective for communicating discrete types of behavior.

The **State Machine Diagram** represents behavior of an entity in terms of its transitions between states triggered by events (same as UML state machine diagram). A state machine can specify the lifecycle behavior of a block in terms of its states and transitions, and are often used with sequence and activity diagrams.

The **Use Case Diagram** represents functionality in terms of how a system or other entity is used by external entities (i.e., actors) to accomplish a set of goals (same as UML use case diagram). The use cases describe the high-level goals of the system.

The semantics of the language enable a modeler to develop an integrated model where model elements on one diagram can be related to model elements on other diagrams. The diagrams enable capturing and viewing the information in the model repository to help specify, design, analyze, and verify systems. The repository information can be imported and exported to exchange model data via the XMI standard and other exchange mechanisms.

Finally, it is also worth highlighting that the SysML also supports tabular representations as a complement to diagram representations to capture model information such as allocation tables.

Anexo

B

SPEM

This annex presents an introduction of the main characteristics of SPEM (Software and Systems Process Engineering Meta-Model), an UML meta-model for methods and processes definition. It provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting (OMG, 2008). The content of this annex is extracted from the OMG standard specification of the language. In Section B.1 we present a brief overview of the language objectives. In Section B.2 we show a definition for the language main elements.

B.1 Overview

SPEM's goal is to provides process engineers with mechanisms to consistently and effectively manage whole families of related processes. For this, SPEM uses a set of relationships for reuse and variability to realize inheritance and crosscutting semantics which allows a process engineer to maintain consistent families of processes, that may be specialized to a given project type but may also present variations of the same base method and process content. The results of this approach are variants of specific processes based on the same core method content and process structures but applied with different levels of detail and scale.

In this context, we observe that the main characteristic of SPEM is the separation of reusable method content from its application in processes. Method Content is fun-

damentally described by defining Task Definitions organized into Steps, having Work Product Definitions as input and output, and performed by Roles Definitions. While the method content provides step-by-step explanations describing how specific development goals are achieved independent of the placement of these steps within a development lifecycle; on the other hand, processes take these method content elements and relate them into partially-ordered sequences that are customized to specific types of projects, defining a particular use for these method elements. These use relationships are responsible for the realization of traceability from processes to their underlying method content, allowing changes in the methods to be reflected in all processes using it.

Although a Process aims to describe the work to be performed in a development project, it keeps certain decisions that are too project specific open. For example, the process definition may define which tasks and activities has multiple occurrences or is repeatable, but does not say how many occurrences and how many repeats/iterations it will have. These decisions have to be made by a project manager when planning a concrete project, project phase, or project iterations to identify the elements that need to be instantiated multiple times, i.e. allowing parallel execution, or repeatedly, i.e. being executed sequentially.

B.2 Elements

SPEM defines four essential elements for methods and processes creation, namely:

i) Tasks: defines work being performed for a particular goal;

ii) Roles: defines individual involved in the task execution;

iii) Work Products: defines artifacts used as input and output for tasks; and

iv) Guidelines: define artifacts supporting tasks execution and work products creation.

Overall, a Task Definition encompasses details on the objectives and alternatives for a given task, specifying roles involved and work products produced and consumed by the task execution. As stated before, a Role Definition specify the individual or team performing the work described by the Task Definition or many Role Definitions participating and collaborating with each other during the Task Definition's execution. A Work Product Definition describes elements of the method content that can be used, modified, and produced by Task Definitions. In fact, Roles use Work Products to perform Tasks and produce Work Products in the course of performing Tasks. Frequently, work products may represent: tangible work that may serve as a basis for defining reusable assets (named Artifact); non-tangible work that are a result or state, or informally defined (named

Outcome); or it can pack other Work Products, and be delivered to an internal or external party (named Deliverable). A key differentiator of outcomes versus artifacts is that outcomes are not candidates for harvesting as reusable assets. In particular, a particular Work Product can also assume different states, which permits capturing the "flow of state" that many key development work products go through. Finally, SPEM also enables the classification of method and processes content into different categories such as View, Grouping, Practice, Discipline, Domain, and others to improve organization. Table B.1 summarizes the language elements used in this project.

Tabela B.1: Description of SPEM elements

Element	Notation	Description
`Guidance`		It provides additional information related to method content as well as for specific processes. SPEM also defines many different kinds — e.g. such as roadmaps, templates, term definitions, concepts, checklists, examples, and reports — for which perhaps a specific structure and type of content is assumed
		Method Content
`Work Product Definition`		It is an element that is used, modified, and produced by Task Definitions. Work Product Definitions can be related to other Work Product Definitions.
`Role Definition`		It is an element that specify a set of related skills, competencies, and responsibilities. Roles are used by Task Definitions to define who performs them as well as to define a set of Work Product Definitions that they are responsible for.
`Task Definition`		It establishes work being performed by Roles Definition instances. A Task is associated to input and output of Work Products.
`Category`		It establishes different kinds of elements grouping, such as disciplines or domain, that helps navigating through the method content.
		Process Content
`Work Product Use`		It represents a Work Product Definition in the context of one specific Activity. One Work Product Definition can be represented by many Work Product Uses each within the context of an Activity with its own set of relationships.

Continued...

Tabela B.1 – Continued

Element	Notation	Description
`Role Use`		It represents a Role Definition in the context of one specific Activity. One role can be represented by many Role Uses, each within the context of an Activity with its own set of relationships.
`Task Use`		It represents a Task Definition in the context of one specific Activity. One Task Definition can be represented by many Task Uses each within the context of an Activity with its own set of relationships.
`Activity`		It represents a grouping of nested elements such as other Activity instances, Task Uses, Role Uses, Milestones, etc. But, it is not only a "high-level" grouping of work, since it also defines a namespace for these elements. Thus, it enables instances of specific elements to define different relationships and textual documentation properties according to different activities. In addition, each activity may contain pre- and post-conditions to define constraints that need to be valid before the described work can begin or before it can be declared as finished.
`Milestone`		It establishes a significant event in a development project, such as a major decision, completion of a deliverable, or meeting of a major dependency (like completion of a project phase). It can be used to refer to both the event itself and the point in time at which the event is scheduled to happen.

Apêndice

C

Reference Architecture for Situated Multiagents Systems

In this chapter, we give an overview of the reference architecture for situated multiagent systems. This reference architecture integrates the agent and environment functionalities and maps them onto abstract system decomposition, i.e. software elements and relationships between them. This abstracted system composition provides a blueprint for architectural design of the family of self-managing applications. It is important to notice that this reference architecture is originally defined in Weyns (2006).

C.1 Introduction

A reference architecture embodies a set of architectural best practices gathered from the design and development of a family of applications with similar characteristics and system requirements. Applying the reference architecture to develop new software architectures will yield valuable input that can be used to update and refine the reference architecture. As such, a reference architecture provides a means for large-scale reuse of architectural design.

This document describes the various aspects of the reference architecture design for situated multiagent systems. Example domains are mobile and ad-hoc networks, automated transportation systems, and robotics. First, the objectives of the reference architecture

are listed. Next, architectural views of the reference architecture are presented. This documentation covers eight views of the reference architecture. The documentation of each view starts with a brief explanation of its goal and a general description (elements and relationships between elements). Additional views or references to supporting material may also be displayed in the description of each view. It is important to notice that this reference architecture uses SysML[1] to describe each view diagram.

C.2 Objectives

The general goal of the reference architecture is to support the architectural design of self-managing applications with situated multiagent systems. Concrete motivations for the reference architecture are:

- *Integration of mechanisms.* The mechanisms of adaptivity for developing self-managing applications are described separately. However, to build a concrete application these mechanisms have to work together. The reference architecture integrates the different mechanisms. It defines how the functionalities of the various mechanisms are allocated to software elements of agents and the environment and how these elements interact with one another.

- *Blueprint for architectural design.* The reference architecture generalizes common functions and structures from various experimental applications we have studied and built. This generalized architecture provides a reusable design artifact, it facilitates deriving new software architectures for systems that share the common base more reliably and cost effectively. On the one hand, the reference architecture defines constraints that incarnate the common base. On the other hand, the architecture defines variation mechanisms that provide the necessary variability to instantiate software architectures for new systems.

- *Reification of knowledge and expertise.* The reference architecture embodies the knowledge and expertise we have acquired during our research. It conscientiously documents the knowhow obtained from this research. As such, the reference architecture offers a vehicle to study and learn the advanced perspective on situated multiagent systems we have developed.

[1] Available at http://www.omg.org/spec/SysML/1.3/

C.3 Concerns

Because the stakeholders of situated multiagent systems have different demands on the quality of the software, two requirements were selected as most relevant: *flexibility* (i.e., the ability to adapt to variable operating conditions) and *openness* (i.e., the ability to cope with parts that come and go during execution). To meet both requirements, the agent needs to be self-manageable to cope with dynamic and changing operating conditions. Because global access to resources can be difficult or even infeasible to achieve, in this case the systems are also required to actuate locally.

C.4 Architectural Views

In this section we describe the reference architecture for a situated multiagent system. We present a total of six views of the reference architecture, namely: conceptual, structural, shared data, collaborative components view, architectural decisions, and variability view.

C.4.1 Conceptual View

In this view, we present the most relevant concepts used in this documentation. Explanations shown in parentheses of some concepts in the reference architecture use the Packet-World as exemplification. A representation of the relationship among these concepts is shown in C.1.

Action laws (or a-laws): restrict influences invoked by the Agents.

Agent: is an autonomous problem solving entity in the system. An agent encapsulates its state and controls its behavior. Agents are situated in an environment which they can perceive and in which they can act and interact with one another.

Application environment (AE): is the part of the environment that has to be designed for a concrete multiagent system application. It acts as the medium in which agents share information and coordinate their behavior.

Communication Language: determines the format of the messages, i.e. the subsequent fields the message is composed of.

Communication Laws (or c-laws): impose constraints on the exchange of messages.

Communication Protocol: specifies a set of possible sequences of messages.

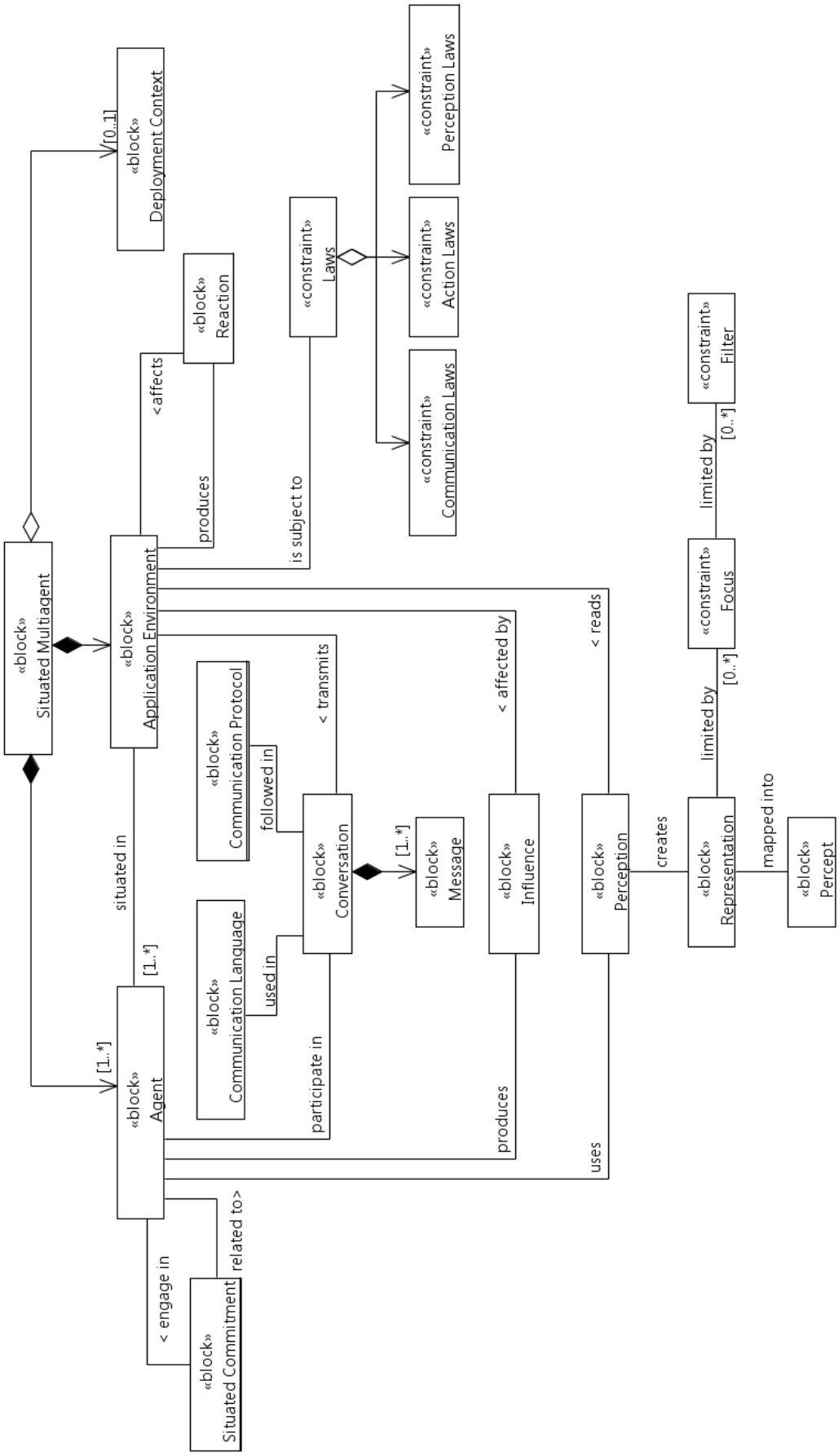

Figura C.1: Conceptual view for the situated multiagent reference architecture

Conversation: term used to refer to an ongoing communicative interaction. It starts with the initial message of a communication protocol. At each stage of the conversation there is a limited set of possible messages that can be exchanged. Terminal states determine when the conversation comes to an end.

Deployment context (DC): is the given hardware and software and external resources with which the multiagent system interacts (sensors and actuators, a printer, a network, a database, a web service, etc.). It is highly application dependent. The internal structure of the deployment context and its functioning are not considered in the reference architecture.

Description: enables an agent to interpret representations derived from the observation of the environment.

Domain Ontology: domain ontology is used to define the terminology for the application domain that will serve as a basis for the definition of the knowledge of the agents and the state of the application environment. Typically, it also defines a shared vocabulary of words that agents use in messages.

Filter: enables an agent to direct its attention within the sensed information.

Focus: enables an agent to observe the environment for a particular type of information (e.g., a focus to observe objects and a focus to sense gradient fields).

Influence: actions produced by the agents and which attempt to modify the course of events in the environment. There are two kinds of influences: (i) the ones that attempt to modify the state of the Application Environment, and (ii) the ones that attempt to modify the state of the Deployment Context.

Message: contains a field with a unique identifier of the ongoing conversation to which the message belong, fields with the identity of the sender and the identities of the addressees of the message, a field with the performative —word that incites an addressee to perform a particular action, e.g. request, propose, and accept — of the message, and a field with the content of the message.

Observation: enable the multiagent system to collect data from resources in the deployment context.

Operation: used by the application environment as internal representations for the influences invoked by the agents. Operations can represent influences that target the manipulation of the state of the application environment, or influences that target the manipulation of elements in the deployment context.

Percept: a description of the sensed environment in a form of data elements that can be used to update the agent's current knowledge.

Perception laws (or p-laws): restrict what the agent is able to perceive. For example, it can be used to restrain the amount of information that has to be processed to improve efficiency.

Reactions: actions produced by the environment which result in state changes of the environment.

Representation: is a data structure that represents elements or resources in the environment (e.g., a description of the sensed objects in an agent's vicinity). Interpreting maps a representation to a percept.

Role: represents a coherent part of an agent's functionality in the context of an organization. Roles have a well-known name that is part of the domain ontology and that is shared among the agents in the system. Sharing this name enables agents to set up collaborations via message exchange.

Situated commitment: is an engagement of an agent to give preference to the actions of a particular role in the commitment. Agents typically commit relative to one another in collaboration, but an agent can also commit to itself, e.g. when a vital task must be completed. Commitments have a well-known name that is part of the domain ontology and that is shared among the agents in the system. Sharing this name enables agents to set up collaborations via message exchange.

C.4.2 Structural View

This view separates the implementation context (i.e., parts of hardware and software available, and other external resources with which the system interacts) from the application (which will be developed). A brief explanation of each module features is also included.

C.4.2.1 Primary View

Figure C.2 shows the structural view for the situated multiagent system.

Elements

Communication Service is the module of the *Application Environment* responsible for handling the exchange of messages in the agent system. It provides the necessary infrastructure to buffer messages, and to deliver messages to the appropriate agents. The exchange of messages is regulated by a set of applicable communication laws. It makes use of a (distributed) message transfer system provided by the deployment context to actually

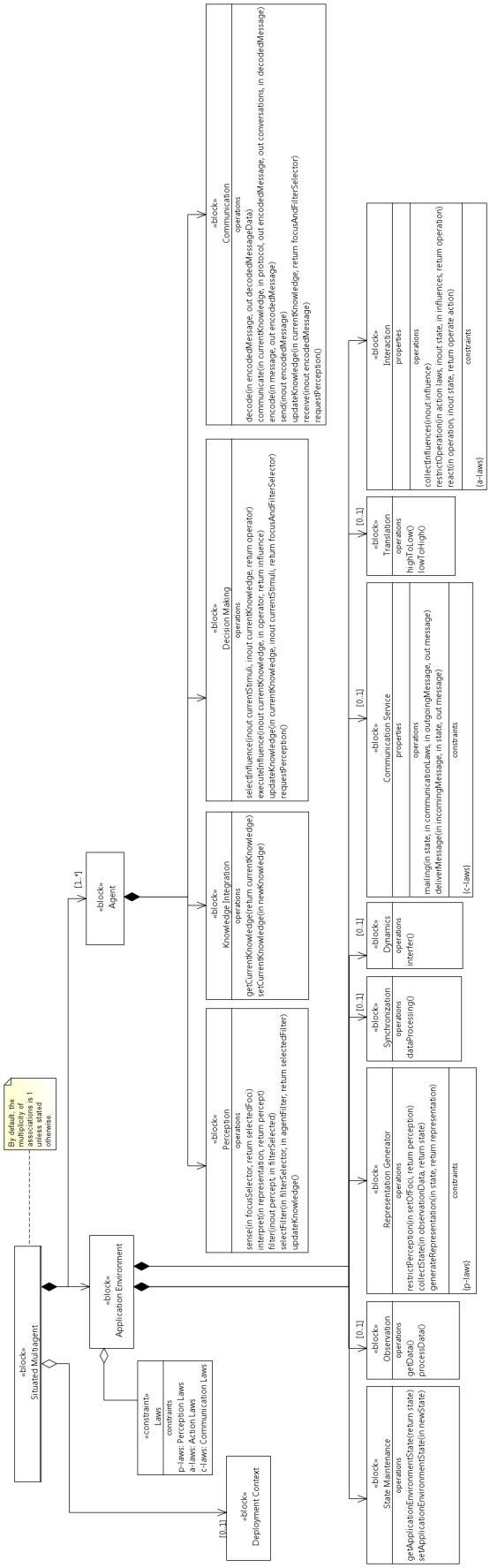

Figura C.2: Structural view of the reference architecture for multiagent systems

transmit the messages; and it uses the translation module to convert the high-level message descriptions into low-level communication primitives of the deployment context.

Communication is the module of the *Agent* responsible for providing the functionality for exchanging messages with other agents according to well-defined communication protocols. Message exchange enables agents to share information and to set up collaborations. The communication module processes incoming messages, and produces outgoing messages according to well-defined communication protocols. It uses the agent's current knowledge to make appropriate decisions.

Decision Making is the module of the *Agent* responsible for providing the functionality for selecting influences to realize the agent's tasks, and for invoking the influences in the *Application Environment*. To enable situated agents to set up collaborations, behavior-based action selection mechanisms are extended with the notions of role and situated commitment. It uses the agent's current knowledge to make appropriate decisions.

Dynamics is the module of the *Application Environment* responsible for maintaining activities that happen independently of the *Agents* and the *Deployment Context*. The dynamics module directly accesses the state of the application environment and maintains this state according to its application specific definition.

Interaction is the module of the *Application Environment* responsible for dealing with agents' actions – *influences* and *reactions* – in the environment. *Agents*' influences are subject to action laws. For influences that relate to the application environment, the interaction module calculates the reaction of the influences resulting in an update of the state of the application environment. Influences that relate to the deployment context are passed to the translation module that converts the influences invoked by the agents into low-level action primitives in the deployment context.

Knowledge Integration is the module of the *Agent* responsible for providing the functionality to access and update the agent's current knowledge [2].

Observation & Data Processing is the module of the *Application Environment* responsible for providing the functionalities to observe the deployment context. Data collected from resources in the deployment context is returned to the requester, i.e. the representation generator. Rather than delivering raw data retrieved from resources in the deployment context, this module can also process this data (e.g., through sorting, sensors calibration, data correction, and/or data interpolation).

Perception is the module of the Agent responsible for providing the functionality for selective perception that permits the agent to direct its perception according to its current tasks. This module provides a Request interface that enables *Decision Making* and *Communication* to request a perception of the environment. These modules pass on a

[2] We use current knowledge as a synonym for the agent's actual state.

focus and filter selector to sense the environment according to their current needs. Sensing results in a representation of the sensed environment, which the perception module later maps into a percept, that the selected set of filters further reduces according to the criteria specified by the filters.

Representation Generator is the module of the *Application Environment* responsible for providing the agents with the functionalities to perceive the environment selectively according to perception laws. When the agent senses the environment, the Representation Generator uses the current state of the *Application Environment* and possibly state collected from the *Deployment Context* to produce a representation for the agent.

State Maintenance is the module of the *Application Environment* responsible for providing the functionality to other modules to access and update the state of the application environment. This state usually includes an abstraction of the deployment context (e.g. topology of a network) and additional state (e.g. virtual marks situated on a map of the physical environment, agent's identities and positions, and tags for coordination purposes).

Synchronization & Data Processing is the module of the *Application Environment* responsible for monitoring application-specific parts of the *Deployment Context* and keeping the corresponding representation in the state of the application environment up-to-date. It converts the resource data observed from the *Deployment Context* into a format that can be used to update the state of the application environment.

Translation is the module of the *Application Environment* responsible for converting the high-level influences of agents into low-level actions in the deployment context.

C.4.3 Shared Data View

This viewpoint describes how the reference architecture is structured into components reading and writing information into various shared data repositories through ports and interfaces. Data repositories act as mediators of the interaction between different modules accessing its content. This repository can meet different concerns, for example concurrency and persistence.

C.4.3.1 Primary View

Figure C.3 shows the shared data view for the situated multiagent system.

Elements

The data accessors— runtime instances of the corresponding modules —of the Agent are Perception, Decision Making and Communication. The accessors share the Current

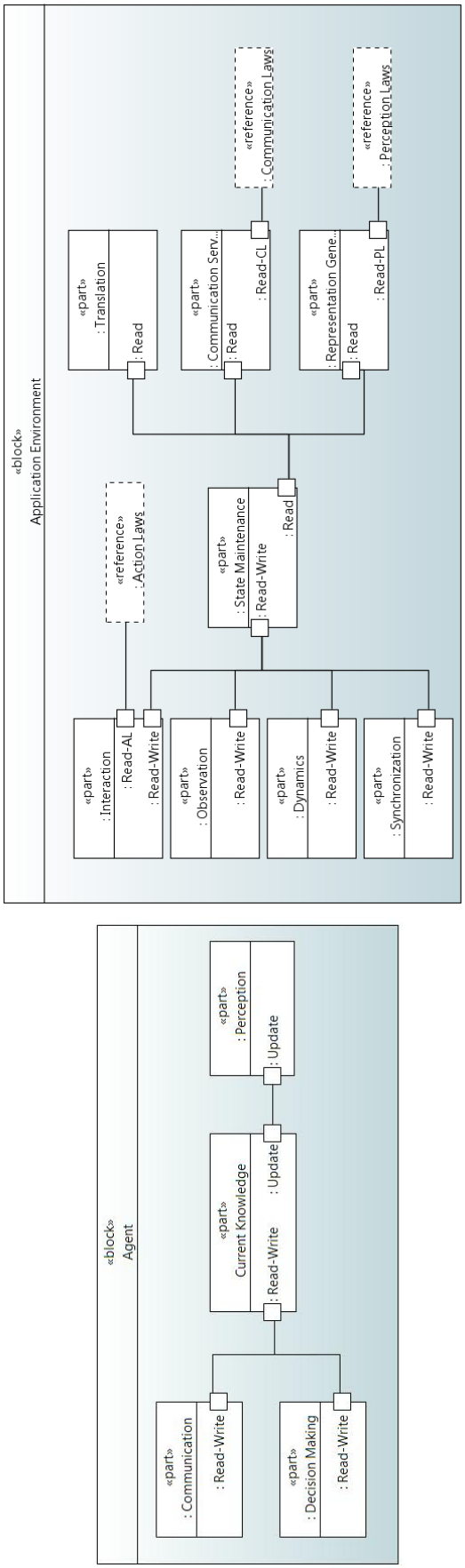

Figura C.3: Shared data view for the multiagent system

Knowledge repository, which provides the functionality to access and update the agents' current knowledge.

The current knowledge repository exposes two interfaces. The provided interface `Update` enables the perception component to update the agents knowledge according to information derived from sensing the environment. The `Read-Write` interface enables the communication and decision making component to access and modify the agent's current knowledge. Data stored in the current knowledge repository refers to state perceived in the environment, to state related to the agent's roles and situated commitments, and possibly other internal state that is shared among the data accessors. To update the agent's knowledge of the environment, both the communication and decision making components can trigger the perception component to sense the environment.

The Laws repository contains the various laws that are defined for the application at hand. This repository is divided in three sub-repositories: one with the perception laws, one with the action laws, and one with communication laws. Each sub-repository is attached to the component responsible for the corresponding functionality. The laws repository exposes three interfaces to read the various types of laws: Read-AL, Read-PL, and Read-CL. These provided interfaces enable the attached components to consult the respective types of laws.

C.4.4 Collaborative Components View

This viewpoint describes the reference architecture as a set of components interacting and using a set of shared data repositories (data flow) to perform the required functionality at runtime. Data repositories act as mediators of the interaction between different modules accessing its content, and can meet different concerns, for example concurrency and persistence. The interaction during runtime is realized by the use of provided and required ports and interfaces, which are depicted in the primary view. The additional view 1 shows the interfaces realization by the agent, application environment, and deployment context. The additional views details in terms of incoming and outgoing data flows the internal behavior of the elements that collaborate to realize a particular set of functionalities, namely: perception, action, and communication.

C.4.4.1 Primary View

Figure C.4 shows the collaborative components view for the situated multiagent system.

Elements

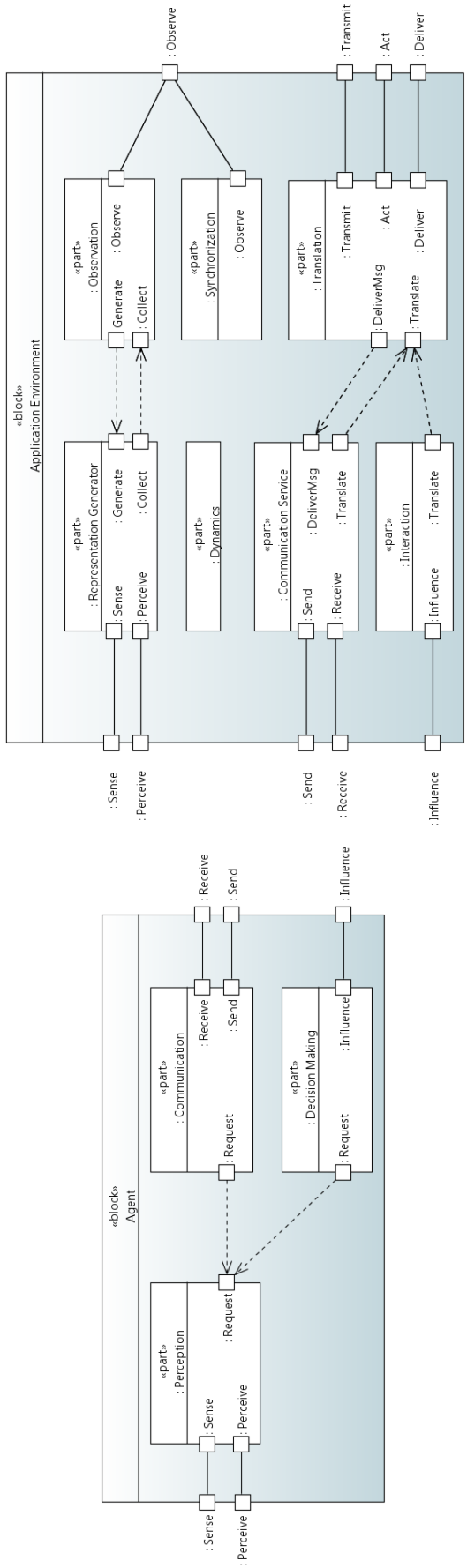

Figura C.4: Collaborative components view for the multiagent system

The `Sense` interface enables an agent to sense the environment, `Send` enables an agent to send messages to other agents, and `Influence` enables an agent to invoke influences in the environment. These interfaces are provided by the application environment.

The application environment requires the interface `Perceive` to pass on representations (resulting from sensing the environment) to the agent, and the interface `Receive` to deliver messages. Furthermore, the application environment requires the interface `Observe` from the deployment context to observe particular resources (based on perception requests of agents), `Transmit` to send messages, and `Act` to modify the state of external resources (based on influences invoked by agents).

Finally, the deployment context requires the interface `Deliver` from the application environment to deliver the incoming messages to the agents.

C.4.4.2 Additional View: Interface Realizations

Fig C.5 shows the interface realizations for the situated multiagent system.

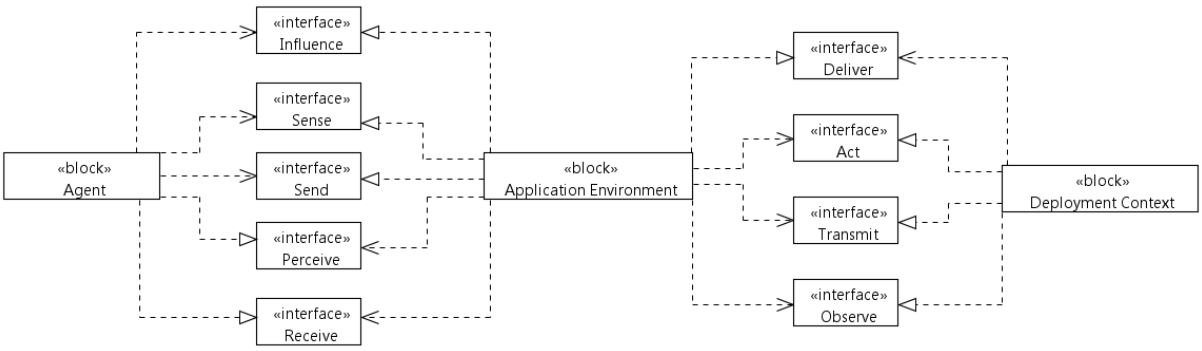

Figura C.5: Distribution of interface realizations in the situated multiagent system

Elements

The Agent provides the interfaces `Perceive` and `Receive`, and requires the interfaces `Influence`, `Send`, and `Sense`.

The Application Environment provides the `Influence`, `Send`, `Sense`, and `Deliver` interfaces, and requires the `Perceive` and `Receive` to interact with the Agent, and the `Act`, `Transmit`, and `Observe` to interact with the Deployment Context.

The Deployment Context requires the `Deliver` interface and provides the interfaces `Act`, `Transmit`, and `Observe`.

C.4.4.3 Additional View: Perception Functionality

In this view, we discuss the collaborating components of perception (see Figure C.6), i.e. the Perception component of the Agent and the Representation Generator of the Application Environment. We follow the logical thread of successive activities that take

place from the moment an agent takes the initiative to sense the environment until the percept is available to update the agent's current knowledge.

Elements

`Sensing` takes a focus selector and the set of foci of the agent, and selects the corresponding foci to produce a perception request. `Perception Restriction` takes the set of foci of a perception request, the current state of the application environment, and generates according to the set of perception laws, a perception scope. A perception scope delineates the scope of perception for the perception request according to the constraints imposed by the perception laws. For example, for `request(i, sense-objects())` which is a perception request of an agent in the Packet-World with identity `i` that senses the visible objects in its vicinity, a possible perception scope is visibleObjects`(agent(i),position(3,6),5)`. In this particular example, the number 5 refers to the default sensing range for visible perception of five cells.

`State Collection` collects the observable state for the perception scope, given the current state of the environment. In particular, state collection selects the subset of state elements of the application environment for the given perception scope (observed state appl. env.), and produces– if applicable –an observation to collect data from the deployment context within the scope of perception.

`Representation Generation` takes the observed state of the application environment together with the state observed from the deployment context (observed state depl. context) and produces a representation.

`Interpreting` uses the agent's set of descriptions to interpret the given representation. Interpreting results in a percept for the agent. `Filter Selection` selects a subset of filters from the agent's set of filters according to the given filter selector. `Filtering` filters the percept of the agent according to the set of selected filters. Finally, the filtered percept is used to update the agent's current knowledge.

C.4.4.4 Additional View: Action Functionality

In this view, we discuss the collaborating components of interaction (see Figure C.7), i.e. the Decision Making component of the Agent and the Interaction component of the Application Environment. For the discussion of the collaborating components, we follow the successive activities that take place from the moment an agent initiates decision making until the effects of the selected influence are completed.

Elements

`D-Knowledge Update` enables the agent to update its current knowledge of the environment. `D-Knowledge Update` takes the agent current knowledge and the set of *current stimuli* and produces a focus and filter selector. We distinguish between two types of

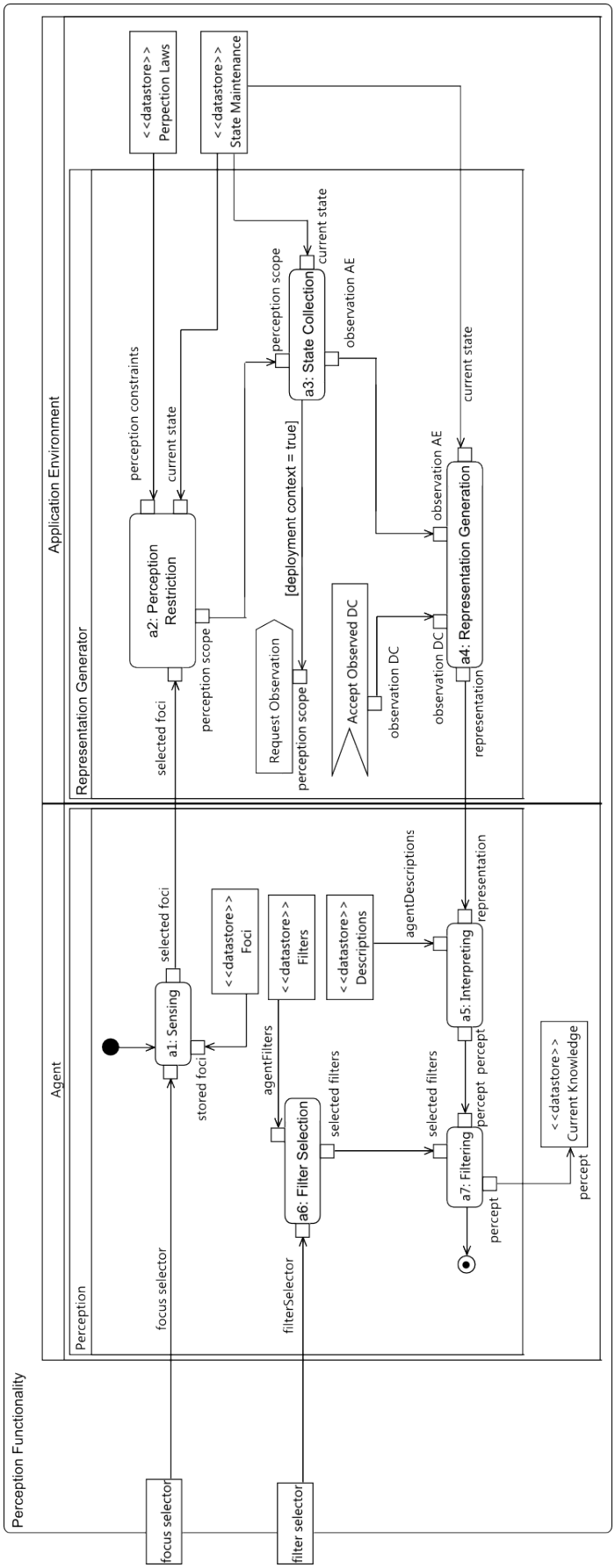

Figura C.6: Collaborating components for the perception functionality

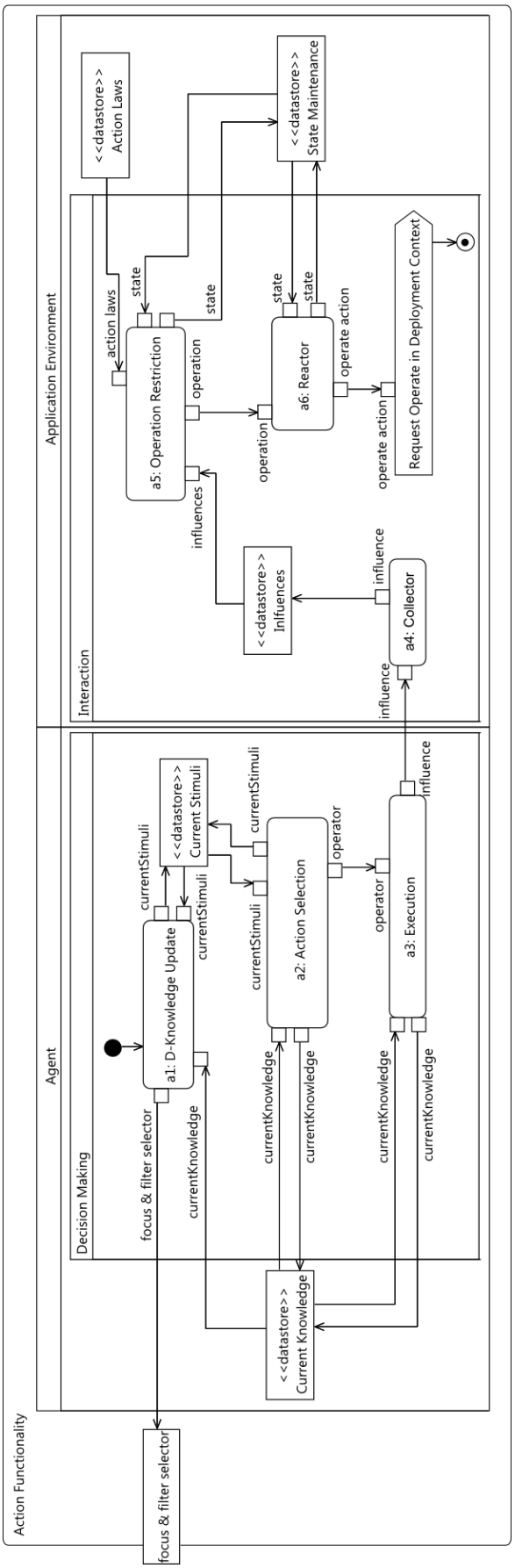

Figura C.7: Collaborating components for the action functionality

stimuli: external and internal stimuli. An external stimulus refers to a factor in the environment that drives the agent's decision making (an example is a gradient field that guides an agent towards a particular location). An internal stimulus is an internal factor that affects an agent's decision making (an example is the agent's available energy to act in the environment). The focus and filter selector is passed to the perception module that senses the environment to produce a new percept and update the agent's knowledge. `Action Selection` takes the current stimuli and the knowledge of the agent to select an operator and update the set of current stimuli. An operator is an internal representation used by the agent to represent the selected action. Decision making converts the operator into an influence that is invoked in the environment; see the discussion of `Execution` below.

To select suitable operators, the action selection component encapsulates a behavior-based action selection mechanism. In general, a behavior-based action selection mechanism consists of a set of behavior modules. Each behavior module is a relatively simple computation module that tightly couples sensing to action. An arbitration or mediation scheme controls which behavior-producing module has control and selects the next action of the agent. To enable situated agents to set up collaborations, behavior-based action selection mechanisms are extended with the notions of role and situated commitment. Behavior modules that represent a coherent part of an agent's functionality in the context of an organization are denoted as a role. A role is defined as a 5-tuple:

$$\mathfrak{R}\text{: } \langle name, stimuli, operators, select \rangle$$

Roles have a well-known *name* that is shared among agents in the system. The function *select* maps the set of stimuli to a set of *h operator*; *pref-factor* i tuples, one for each operator of the set of operators that can be selected by the role. A *pref-factor* determines the relative preference for selecting the accompanying operator. The *h operator*; *pref-factor i* tuples are used by the arbitration schema to determine which role has control and which operator is selected for execution.

A situated commitment is defined as a 7-tuple:

$$\mathfrak{C}\text{: } \langle name, rel - set, act - con, status, rolemap \rangle$$

As for roles, situated commitments have a well-known *name*. Explicitly naming roles and commitments enables agents to set up collaborations, reflected in mutual commitments. The relations set $rel - set$ contains the identity of the related agent(s) in the situated commitment. The context describes contextual properties of the situated commitment such as descriptions of objects in the local environment. $Act{-}con$ and $deact{-}con$ are the activation and deactivation conditions that determine the *status* of the situated

commitment. When the activation condition becomes true, the situated commitment is activated. The behavior of the agent will than be biased according to the specification of the *rolemap*. The *rolemap* specifies the relative weight of the preference factors of the operators of different roles. In its simplest form, the rolemap narrows the agent's action selection to operators in one particular role. An advanced example is a *rolemap* that biases the operator selection towards the operators of one role relative to the preference factors of operators of a number of other roles of the agent. As soon as the deactivation condition becomes true, the situated commitment is deactivated and will no longer affect the behavior of the agent.

For the chosen action selection mechanism the operators and the various stimuli have to be specified. Furthermore, the definition of the agent's roles and situated commitments have to be completed. For each role, the select function has to be defined (see the definition of a role above). For each situated commitment, the *relation set*, the *activation* and *deactivation* conditions, and the *rolemap* have to be defined (see the definition of situated commitment above). Finally, the roles and situated commitments have to be integrated with the arbitration mechanism of the selected behavior-based action selection mechanism.

`Execution` takes the selected operator and the current knowledge of the agent and produces an influence that is invoked in the environment. `Execution` decouples the agent's internal representation of activity in the environment from the influences that are available to the agent to access and modify the state of the environment.

The `Collector` collects influences invoked by the agents, and adds the influences to the set of pending influences in the agent system. `Operation Restriction` selects an influence from the set of pending influences and converts the influence into an operation. For example, for `influence(i, step(North))` which is an influence produced by an agent in the Packet-World with identity i that attempts to make a step on the grid in the direction North, a possible operation is `move(agent(i),position(3,6),destination(2,6))`. The selection of influences is based on an influence selection policy that specifies the ordering of influences, taking into account the current state of the environment. Simple policies are first-in-first-out and random selection. Advanced forms of influence ordering can be defined based on a combination of criteria, such as the types of influences, the types of agents that have invoked the influences, and even the identities/priorities of the invoking agents. `Operation Restriction` applies the set of action laws to the selected operation, given the current state of the environment. The action laws impose restrictions on the kind of manipulations agents can perform in the environment.

The `Reactor` applies the operation to the current state of the application environment and produces an operation to act in the deployment context if applicable. The execution of

the operation modifies the state of the application environment and produces an operation in the deployment context that is passed to the translation module.

C.4.4.5 Additional View: Communication Functionality

In this view, we show how the functionality for communication is realized between the Communication component of the Agent and the Communication Service of the Application Environment (see Figure C.8). We discuss the collaboration between the components by following the successive activities that take place from the moment an agent takes the initiative to send a message until the addressees have received the message and can react to it.

Elements

We start with `Communicating` that handles the agent's communicative interactions. The communicating component processes incoming messages, and produces outgoing messages according to well-defined communication protocols. Communicative interactions can modify the agent's state possibly affecting the agent's selection of influences; a typical example is the activation and deactivation of situated commitments.

A communication protocol consists of a series of protocol steps. A protocol step is a tuple $\langle condition, effect \rangle$. Condition is a Boolean expression that determines whether the step is applicable. Expressions can take into account the agent's current knowledge, the actual set of conversations that contains the agent's history of ongoing interactions, the set of available protocols, and possibly the decoded message data of a received message. We distinguish between three types of protocol steps: conversation initiation, conversation continuation, and conversation termination. A conversation initiation step starts a new conversation according to a particular protocol. An agent initiates a conversation based on its current knowledge, possibly taking into account the decoded message data of a message received from another agent that started the interaction. A conversation continuation performs a step in an ongoing conversation. A conversation continuation may deal with a received message without directly responding to it, it may immediately react with a reply message, or it may pick up a conversation after a break. Finally, a conversation termination concludes an ongoing conversation. The termination of a conversation can be induced by changing circumstances in the environment or it can directly result from a preceding step of the conversation.

`C-KnowledgeUpdate` enables the agent to update its current knowledge according to its ongoing conversations. The knowledge update component takes the agent current knowledge and the set of conversations and produces a focus and filter selector that is passed to the perception module that senses the environment to produce a new percept and update the agent's knowledge.

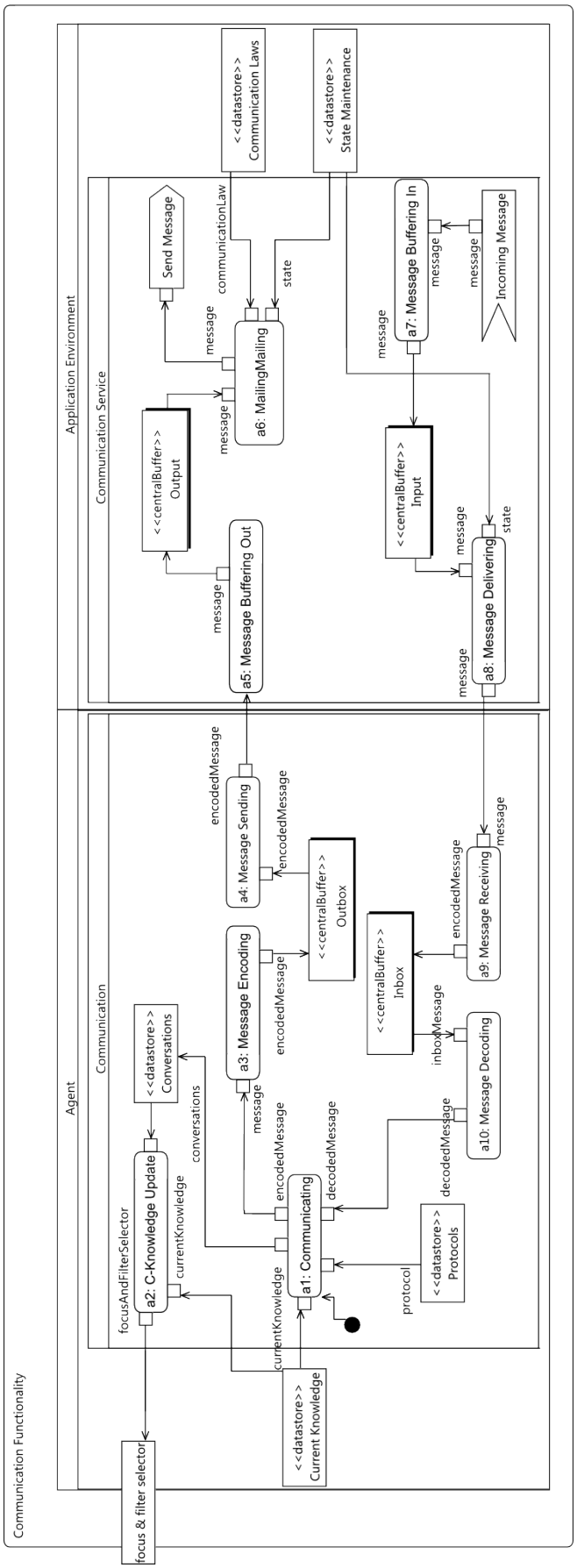

Figura C.8: Collaborating components for the communication functionality

`MessageEncoding` encodes newly composed message data into messages and puts the messages in the outbox buffer of the agent. Message encoding is based on the communication language that is shared among the agents in the system.

`MessageSending` selects a message from the set of pending messages in the outbox buffer and passes it to the communication service. A message selection policy imposes an application specific ordering on the transmission of pending messages from the output buffer of the application environment. A simple policy is first-in-first-out. Advanced forms of ordering can be defined based on a combination of criteria, such as ordering based on the type of messages and the type of agents that have sent the messages. In addition, the policy can take into account runtime information such as the current load of the network to select messages.

`MessageBufferingOut` collects messages sent by agents and puts them in the output buffer of the communication service. `Mailing` selects a message from the output buffer given a message selection policy, and applies the set of communication laws to the selected message taking into account the current state of the environment. To determine the order in which messages are sent, the selection policy takes into account the current state of the environment.

`Mailing` passes the messages to the translation component that handles the conversion of the messages for transmission.

`MessageBufferingIn` collects incoming messages and puts them in the input buffer of the environment. In principle, a selection policy has to be defined for the input buffer of the communication service. However, since the message stream between agents is already regulated by the message selection policy of the output buffer, a simple static selection policy such as first-in-first-out can be used for selecting messages from the input buffer.

`MessageDelivering` delivers the messages of the input buffer to the appropriate agents.

`MessageReceiving` accepts messages and puts them in the agent's inbox.

Finally, `MessageDecoding` selects a message from the agent's inbox and decodes the message according to the given communication language. The decoded message data of the selected message is passed to the communicating component that will process it.

C.4.5 Process View

This view describes the reference architecture as a set of concurrently executing units and their interactions. Concurrent units can be abstractions of concrete software elements such as tasks, processes, or execution threads. This viewpoint takes into account data repositories accessed by these units, exchanged messages, and exposed interfaces. The

relationship in this view is attachment that indicates which connectors are connected to which concurrent units and repositories.

The communicating processes view explains which portions of the system operate in parallel and is therefore an important artifact to understand how the system works and to analyze the performance of the system. Furthermore, the view is important to decide which components should be assigned to which processes. Actually, we present the communicating processes view as a number of core components and overlay them with a set of concurrently executing units and their interactions.

C.4.5.1 Primary View

Figure C.9 shows the processes view for the situated multiagent system.

Elements

We make a distinction between active processes that run autonomously, and reactive processes that are triggered by other processes to perform a particular task. The discussion of the elements in this view is divided in four parts. Successively, we zoom in on the communicating processes of perception, interaction, and communication, and the independent processes of the application environment.

Perception. The Perception Process of agent is a reactive process that can be activated by the Decision Making Process and the Communication Process. Once activated, the perception process requests the Representation Generator Process to generate a representation. The representation generator process collects the required state from the State repository of the application environment, and optionally it requests the Observation Process to collect additional data from the deployment context. State collection is subject to the perception laws. The observation process returns the observed data to the representation generator process as soon as it becomes available. Subsequently, the representation genera- tor integrates the perceived state and generates a representation that is returned to the perception process of the agent. The perception process converts the representation to a percept that it uses to update the agent's Current Knowledge. Finally, the requesting process can read the updated state of the agent. The current knowledge repository can provide a notification mechanism to inform the decision making and communication process when a state update is completed.

Interaction. The Decision Making Process is an active process of agent that selects and invokes influences in the environment. The Interaction Process collects the concurrently invoked influences and converts them into operations. The execution of operations is subject to the action laws of the system. Operations that attempt to modify state of the application environment are executed by the interaction process, operations that attempt to modify state of the deployment context are forwarded to the Appl.Env-DC Translation

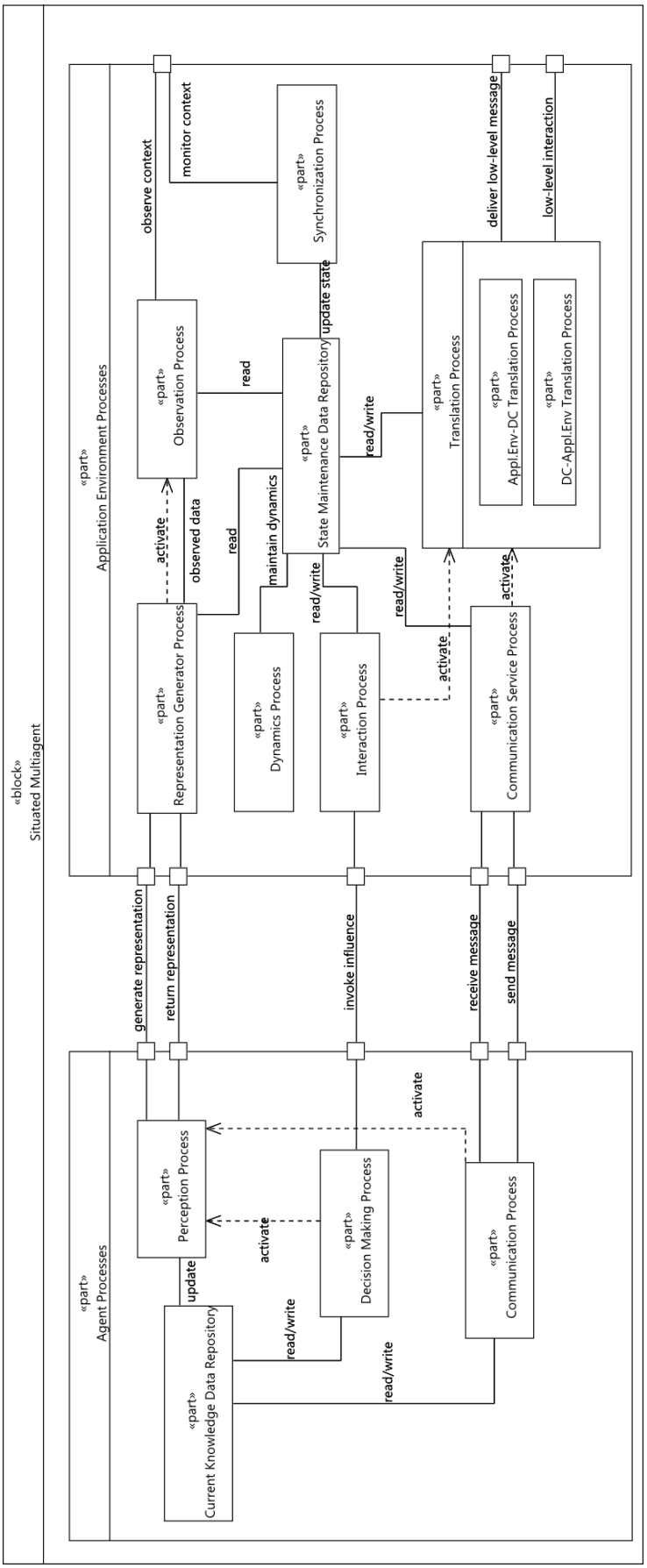

Figura C.9: Processes view of the situated multiagent system

Process. This translation process converts the operations into low-level interactions in the deployment context.

Communication. The Communication Process is an active process that handles the communicative interactions of the agent. Newly composed messages are passed to the Communication Service Process that applies the communication laws and subsequently passes the messages to the Appl.Env-DC Translation Process. This latter process converts the messages into low-level interactions with the deployment context. Furthermore, the DC-Appl.Env Translation Process collects low-level messages from the deployment context, converts the messages into a format understandable for the agents, and forward the messages to the communication service process. The communication process delivers the messages to the communication process of the appropriate agent. Independent Processes in the Application Environment. The Synchronization Processes are active processes that monitor application specific parts of the deployment context and keep the corresponding state of the application environment up-to-date. Ongoing Activities are active processes that maintain application specific dynamics in the application environment.

Independent Processes in the Application Environment. The Synchronization Processes are active processes that monitor application specific parts of the deployment context and keep the corresponding state of the application environment up-to-date. Ongoing Activities are active processes that maintain application specific dynamics in the application environment.

C.4.5.2 Additional View: Processes Behavior

Figure C.10 shows a state machine highlighting the behavior of processes depicted earlier, thus indicating which processes execute in parallel and which processes are activated along the execution of the situated multiagent system. We use the following convention to name transitions: transitions that represent an operation call are labeled with the name of the operation followed by "()", and transitions that represent a signal or event are labeled with the name of the event or signal.

The observation process is reactive, it collects data from the deployment context when requested by the representation generator. The translation processes are also reactive, they provide their services on command of other processes. Finally, synchronization processes and ongoing activities are active processes that act largely independent of other processes in the system. Synchronization processes monitor particular dynamics in the deployment context and keep the corresponding representations up-to-date in the state of the application environment. Ongoing activities represent dynamics in the application environment that happen independent of agents and the deployment context. These pro-

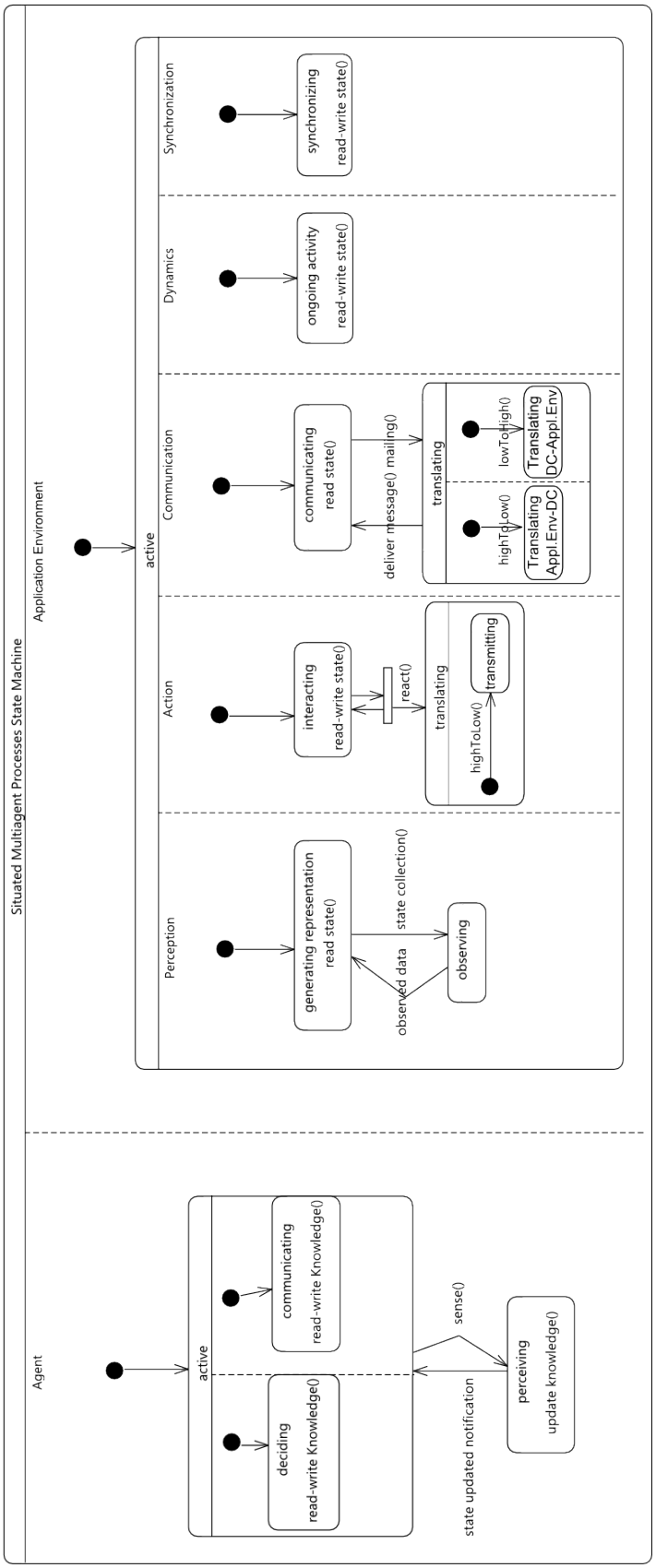

Figura C.10: Processes additional view of the situated multiagent system

cesses are responsible to maintain the state the application environment according to the ongoing dynamics.

C.4.6 Architectural Decisions Detail View

AD 1: Decentralized control among situated agents.

Arguments: It is essential to cope with the inherent locality of the activity, which is a characteristic of the target applications of the reference architecture.

Implications: Requires more communication, which may affect performance. In addition, because decision making is based on local information only, this may also lead to suboptimal system behavior.

AD 2: Self-management of agents.

Arguments: It enables a system to manage dynamic and changing operating conditions autonomously.

Implications: Systems designed to cope with many disturbances generally need redundancy, usually to the detriment of performance, and vice-versa.

AD 3: Module Decomposition.

Arguments: It is necessary to achieve quality attributes (e.g., changeable parts of a system are encapsulated in separate modules, supporting modifiability; and the separation of functionality of a system that has a higher performance requirement from other functionality). It supports efficient design and implementation of systems with the reference architecture.

Implications: This decomposition results in a clear organization of the architecture.

AD 4: Selective perception.

Arguments: Selective perception enables an agent to focus its attention to the relevant aspects in the environment according to its current tasks. By selecting an appropriate set of foci and filters, the agent directs its attention to the current aspects of its interest, and adapts it attention when the operating conditions change.

Implications: When selecting actions and communicating messages with other agents, decision making and communication typically request perceptions to update the agent's knowledge about the environment.

Related Decisions: D2

AD 5: Coordination between Decision Making and Communications.

Arguments: The separation of functionality for coordination (via communication) from the functionality to perform actions to complete tasks enables to complete the agent's tasks more efficiently. For example, agents can send each other messages with requests for information that enable them to act more purposefully. Decision making and communication also coordinate during the progress of collaboration. Collaborations are typically established via message exchange. Once collaboration is achieved, the communication module activates a situated commitment. This commitment will affect the agent's decision making towards actions in the agent's role in the collaboration. This continues until the commitment is deactivated and the collaboration ends.

Implications: Requires a careful design, but presents several advantages, such as clear design, improved modifiability, and reusability. Two particular advantages of separating communication from performing actions are: (1) it allows both functions to act in parallel, and (2) it allows both functions to act at a different pace. In many applications, sending messages and executing actions happen at different tempo. Also enables agents to reconsider the coordination of their behavior while they perform actions, improving adaptability and efficiency.

AD 6: Module decomposition of the application environment.

Arguments: The decomposition of the application environment yields a flexible modularization that can be tailored to a broad family of application domains.

Implications: Minimizing the overlap of functionality among modules, helps the architect to focus on one particular aspect of the functionality of the application environment. It supports reuse, and it further helps to accommodate change and to update one module without affecting the others.

Related Decisions: D3

AD 7: Creation of a current knowledge repository.

Arguments: Enables the communication and decision making components to share data and to communicate indirectly, which allows both components to act in parallel and at a different pace. The laws repository encapsulates the various laws as first-class elements in the agent system, which avoids that laws are scattered over different components of the system.

Implications: Design each component encapsulating its own state and providing interfaces through which other elements can get access to particular information. Because a lot of state is shared between the components of an agent (examples are the state that is derived from perceiving the environment and the state of situated commitments), such a design would increase dependencies among the components or imply the duplication of state in different components. Furthermore, such duplicated state must be kept synchronized among the components. If performance is a high-ranked quality, laws may be hard coded in the various applicable modules.

Related Decisions: Improve efficiency and adaptability. The shared data style results in low coupling between the various elements, improving modifiability and reuse. The state repository enables the various components of the application environment to share state and to communicate indirectly. This avoids duplication of data and allows different components to act in parallel. Explicitly modeling laws may induce an important computational overhead.

AD 8: Abstract from Deployment Context.

Arguments: Depending on the specific application requirements, distribution can take different forms. For a distributed application, the deployment context consists of multiple processors deployed on different nodes that are connected through a network.

Alternatives: The same instance of the application environment subsystem is deployed on each node, or specific instances are instantiated on different nodes, e.g., when different types of agents are deployed on different nodes.

Implications: Variability support.

AD 9: Collaboration of Perception and Representation Generator Components.

Arguments: Provides the functionality for selective perception, which results from the collaboration of the various components. Each component provides a clear-cut functionality, while the coupling between the component is kept low. Concepts such as foci, descriptions, filters, and laws are first-class in the reference architecture. Selective perception allows an agent to adapt its perception according to its current tasks. The reference architecture supports adaptation of perception laws according the changing circumstances in the environment. Both these properties contribute to the flexibility of the system.

Implications: Helps to improve modifiability and reusability.

AD 10: Collaboration of Decision Making and Interaction Components.

Arguments: Provides the functionality for action execution in the agent system. Each component provides a clear-cut functionality, while the coupling between the components is kept low. Behavior-based action selection enables agents to behave according to the situation in the environment, and change their behavior with changing circumstances. The notions of a role and situated commitment enable agents to set up collaborations. The duration of a situated commitment can be regulated based on conditions in the local context in which the collaborating agents are placed. Both action laws and the influence selection policy can be defined in terms of the actual conditions in the environment, contributing to the flexibility of the system.

Implications: Helps to improve modifiability, reusability, openness, and flexibility. In accordance with locality requirement.

AD 11: Collaboration of the Communication and Communication Service components.

Arguments: Provides the functionality for message exchange in the agent system. Direct communication allows situated agents to exchange information and set up collaborations. Coordination through message exchange is complementary to mediated coordination via marks in the environment (e.g. pheromone-based coordination). The various components in the collaboration are assigned clear-cut responsibilities and coupling amongst components is kept low. Communication defined in terms of protocols puts the focus of communication on the relationship between messages. Message selection policy and communication laws provide mechanisms to regulate the message stream in the agent system and are imposed according to changing conditions in the environment.

Implications: Helps to improve flexibility.

AD 12: Define two kinds of processes: active and reactive.

Arguments: Active processes represent loci of continuous activity in the system. By letting active processes run in parallel, different activities in the system can be handled concurrently. Reactive processes, on the other hand, are only activated and occupy resources when necessary.

Implications: Improve efficiency.

AD 13: Provide agents with an active process for decision making and one for communication.

Arguments: This allows these processes to run in parallel. Because the perception process is reactive, the agent only senses the environment when required for decision making and communicative interaction, thus being only activated when necessary.

Implications: Improve efficiency. Communication among the processes happens indirectly via the current knowledge repository.

AD 14: Provide the application environment with separate processes to collect and process perception requests, handle influences, and provide message transfer.

Related Decisions: D12

C.4.7 Variability View

This viewpoint describes which are and how elements of the reference architecture can be exercised to build instances of the reference architecture. This view highlights two kinds of variation mechanisms: omission, which highlights optional elements and abstraction, which highlights elements that need further specification and must be implemented according to a particular application. By default, mandatory elements are noted with a thicker outline and bold font. Conversely, optional elements use the regular notation of thinner outline. Abstract elements are highlighted with an italic font.

C.4.7.1 Primary View

Figure C.11 shows the variability view of the situated multiagent system.

Elements

For agents that do not communicate via message exchange the communication module can be omitted. An example is an ant-like agent system in which the agents communicate via the manipulation of marks in the environment.

For applications that do not interact with external resources, the observation, synchronization, and translation modules can be omitted. For such applications, the environment is entirely virtual and as such only consists of the application environment. In addition, if the multiagent system applications with an application environment that does not have to maintain dynamics independent of agents, the dynamics module can also be omitted.

The three interfaces between the application environment and the deployment context must be concretized according to the application at hand. An observation primitive indicates which resource is observed and what type of information should be observed. Action primitive indicates the target resource and the type of action. Communication primitives enable to transmit low-level formatted messages via the deployment context.

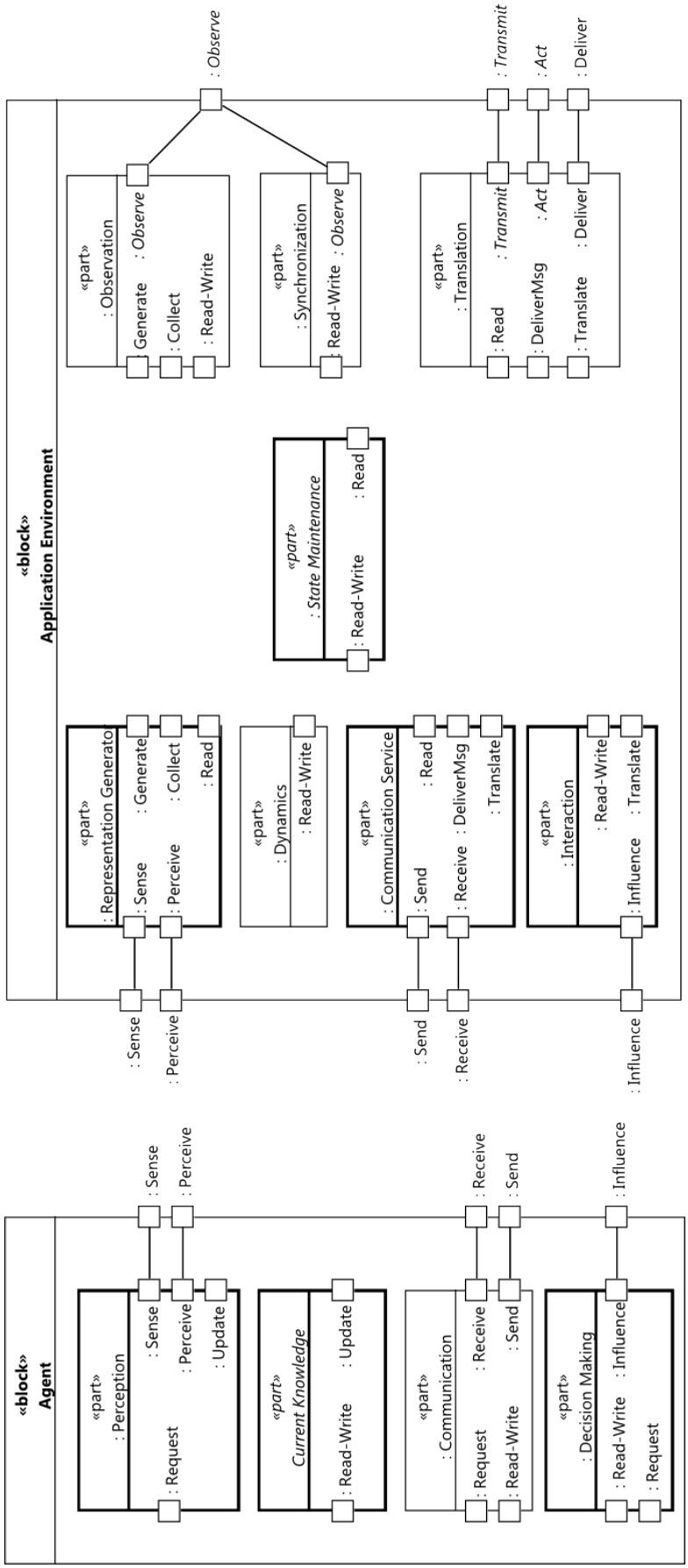

Figura C.11: Variability view of the situated multiagent system

A low-level formatted message is a data structure that represents a message exchanged between a sender and one or more addressees and that can be transmitted via the deployment context.

Definition of current knowledge includes the definition of the state of the agent and the specification of the knowledge repository. The definition of the state of the agent has to comply to the ontology that is defined for the multiagent system application. The specification of the knowledge repository includes various aspects such as the specification of a policy for concurrency, specification of possible event mechanisms to signal data consumers, support for persistency of data, and support for transactions. The concrete interpretation of these aspects depends on the specific requirements of the application at hand.

The definition of state includes the definition of the actual state of the application environment and the specification of the state repository. The state definition has to comply to the ontology that is defined for the application domain. The specification of the state repository includes various aspects such as the specification of a policy for concurrency, specification of possible event mechanisms to signal data consumers, support for persistency of data, and support for transactions.

The parts of the deployment context for which a representation has to be maintained in the application environment have to be defined. The deployment context may provide a notification mechanism to inform synchronization processes about changes, or the processes may poll the deployment context according to specific time schemes.

C.4.7.2 Additional View: Concepts Variability

Figure C.12 depicts the variability view of concepts previously defined in the conceptual view.

Elements

Definition of Foci and Focus Selectors. The definition of the foci in the agent system includes the specification of the kind of data each focus targets, together with the scoping properties of each focus. The definition of focus selectors includes the specification of the various combinations of foci that can be used to sense the environment.

Definition of Filters and Filter Selectors. The definition of the filters in the agent system includes the specification of the kind of data each filter aims to filter and the specific properties of each filter. The definition of filter selectors includes the specification of the various combinations of filters that can be used to filter percepts.

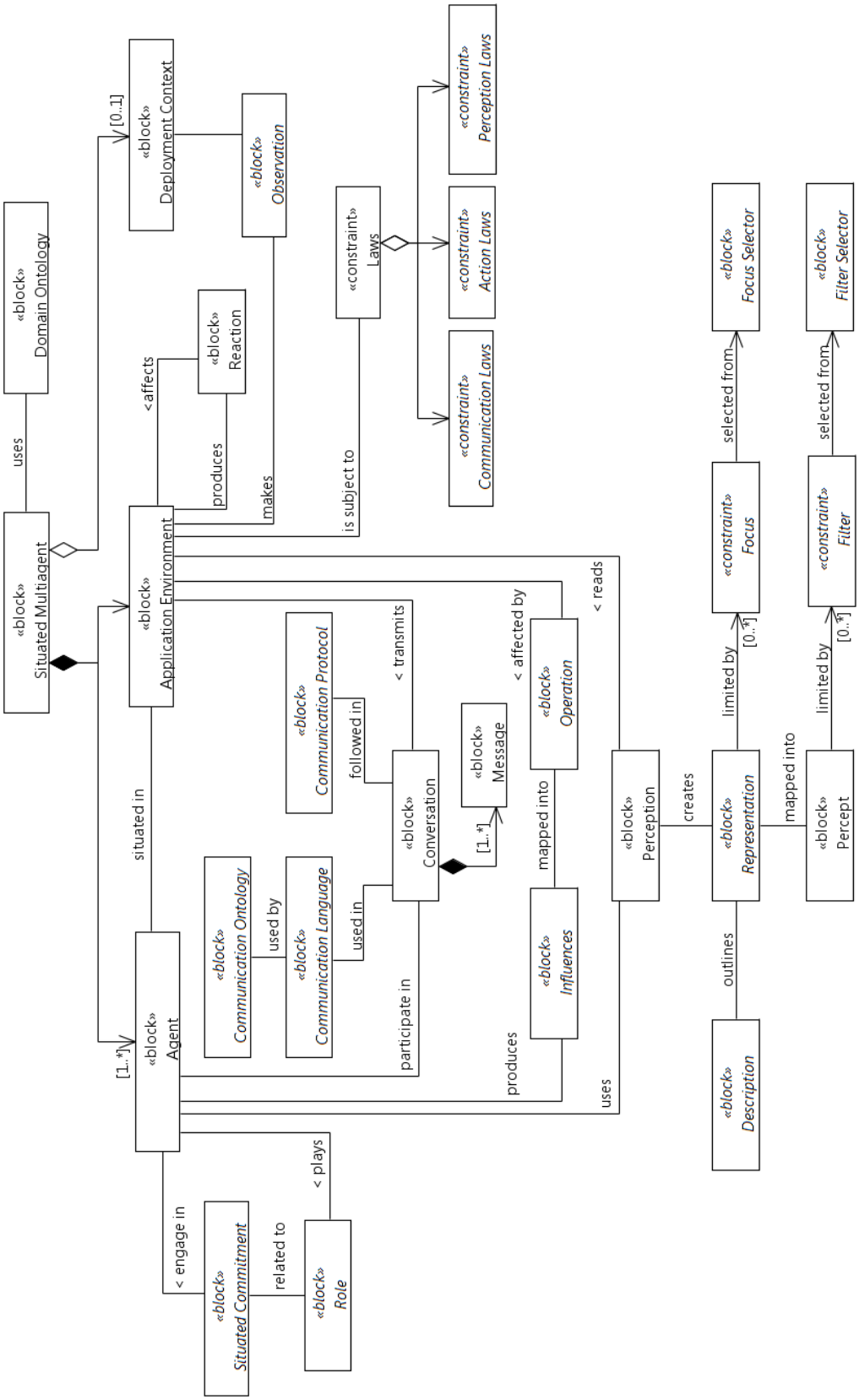

Figura C.12: Concepts variability view

Definition of the Communication Language and the Ontology. The definition of the communication language includes the specification of identities for agents and conversations, the specification of the various performatives of the language, and the format of the content of messages. The definition of the ontology for communication includes the specification of the vocabulary of words that represent the domain concepts used in messages and the relationships between the concepts. The ontology for communication is typically a part of the integral domain ontology.

Definition of Communication Protocols. The definition of a concrete communication protocol includes the specification of (1) the conditions for the agent to initiate the protocol, (2) the conditions to continue the interaction in the various stages of the protocol, (3) the conditions to terminate the interaction. For each situation, the protocol has to define the concrete actions that have to be executed. Actions include the processing of received messages, the composition of new messages, and the update of the current knowledge. An important aspect of this latter is the activation/deactivation of situated commitments. State charts are one possible approach to specify a communication protocol.

Definition of Representation. Representations are defined by means of data structures that represent elements and resources in the environment. The definition of representations must comply with the ontology defined for the domain.

Definition of Influences. The definition of an influence includes the specification of an operation that is provided by the application environment and that can be invoked by the agents.

Definition of Roles and Situated Commitments. Each role in the agent system is defined by a unique name and a description of the semantics of the role in terms of the influences that can be selected in that role as well as the relationship of the role to other roles in the agent system. Each situated commitment in the agent system is defined by a unique name and a description of the semantics of the commitment in terms of roles defined in the agent system.

Definition of Observations. The definition of an observation includes the specification of the kind of data to be observed in the deployment context together with additional properties of the observation.

Definition of Descriptions. Descriptions enable an agent to interpret representations derived from the observation of the environment. A description can be defined as a template that specifies a particular pattern of a representation. Interpreting a

representation then comes down to searching for matches between the description template and the examined representation. Each match yields data of a percept that is used to update the agent's state, possibly after some preceding filtering.

Definition of Operations. The concrete definition of operations is closely related to the definition of the state of the application environment.

Definition of Action Laws. The constraints imposed by an action law can be defined relative to the actual state of the environment. For example, when an agent injects a tuple in network, the distribution of the tuple can be restricted based on the actual cost for the tuple to propagate along the various links of the network.

Definition of Perception Laws. The constraints imposed by a perception law can be defined relative to the actual state of the environment. For example, restrictions on the observation of local nodes in a network can be defined as a function of the actual traffic on the network.

Definition of Communication Laws. The constraints imposed by a communication law can be defined relative to the actual state of the environment. For example, the delivering of a broadcast message in a network can be restricted to addressees that are located within a particular physical area around the sender.

Definition of Ongoing Activity Process. Each ongoing activity in the application environment an active process has to be defined. Processes of ongoing activities may run independently, or they may monitor and react to particular events in the application environment. Examples of the former are a clock, and the evaporation of digital pheromones. Examples of the latter are a timer that is triggered by a particular event, and the diffusion of new pheromone that is dropped by an agent.

C.5 Open Decisions

This reference architecture did not explicitly address human-software interaction, because it heavily depends on the application domain, which determines the role of humans in the multiagent systems. For example, in some applications humans can play the role of agents and interact directly or via an intermediate wrapper| with the application environment, while in others humans can be part of the deployment context with which the multiagent system application interacts.

This reference architecture design also abstracted from distribution, although it is an important concern in many application domains. This decision was preferred because introducing distribution in the reference architecture would result in a more specific design

that is tailored toward such applications. Some functionality provided by the application environment may be limited to the local context (e.g. observation of the deployment context may be limited to resources of the local deployment context); other functionalities may be integrated (partially or complete) by appropriate middleware. Examples are support for message transfer in a distributed setting, or for a distributed pheromone infrastructure. If the required support is not available, it has to be developed and integrated with the rest of the application logic. By abstracting from distribution, this reference architecture is also valuable for non-distributed applications, such as simulations.

The communication service of the Application Environment can be provided with additional services, such as a yellow-page service to enable agents to find particular service providers. But, since such services are uncommon for situated agent systems, they are not included in the reference architecture.

In addition, crosscutting concerns (such as security, monitoring, and logging, which usually crosscut several architecture modules) were not tackled in this reference architecture design because they are hardly explored in the domain and are open research problems.

C.6 Change History

9, 2012	Creation
	Conceptual View added
	Structural View added
	Architectural decisions details view added
10, 2012	Conceptual view updated
	Structural view updated
	Collaborative components view added
	Architectural decisions details view updated
11, 2012	Collaborative components view updated
	Variability View added
	Deployment View added
	Process View added
	Architectural decisions details view updated
	Documentation Review

Apêndice

D

Relatório da Avaliação do Método

Neste apêndice é apresentado o planejamento do estudo de caso conduzido para avaliar a representação construída com o auxílio do ProSA-Re e também o planejamento da avaliação realizada com especialistas. Este apêndice é organizado da seguinte maneira. Na Seção D.1 é apresentado o contexto e a motivação para a avaliação do método, sendo também delineados os objetivos para a sua realização. Em seguida, na Seção D.2 é apresentado o planejamento do estudo de caso, incluindo a descrição dos procedimentos empregados para coleta e análise de dados e também as respostas obtidas. Na Seção D.3 é apresentado o planejamento da avaliação com especialistas, também incluindo os procedimentos empregados para coleta e análise de dados.

D.1 Contextualização e Motivação

A representação adequada de arquiteturas de referência é essencial para que elas sejam, de fato, empregadas na criação de arquiteturas de software de um determinado domínio. Considerando-se, em particular, o domínio de sistemas embarcados, observa-se que algumas características, tais como desempenho, consumo de energia e segurança, exercem grande influência no projeto arquitetural e, portanto, precisam ser capturadas pela descrição. Observa-se, contudo, que a representação de arquiteturas de referência é muitas vezes conduzida de maneira informal, sendo que cada autor utiliza um meio diferente para representá-las (Oliveira et al., 2010).

Visando sistematizar e padronizar a representação de arquiteturas de referência, em especial para o domínio de sistemas embarcados, é proposto o ProSA-Re, que estabelece um conjunto de atividades e diretrizes para a construção de representações adequadas para esse domínio. O objetivo do método é, portanto, garantir que determinados interesses relevantes para o domínio de sistemas embarcados sejam capturados pela representação, de modo a promover a comunicação e o reúso do conhecimento arquitetural contido nessas arquiteturas.

Nesse contexto, é importante avaliar se a representação elaborada com o auxílio do ProSA-Re oferece, de fato, benefícios que justifiquem a sua adoção. Para isso, é relevante investigar a eficiência das representações construídas com o seu apoio em transferir o conhecimento arquitetural de maneira consistente. Deseja-se com isso identificar também quais são e como aspectos da representação arquitetural colaboraram para os resultados obtidos. Para avaliar a representação arquitetural construída, foi desenvolvido um estudo de caso, uma técnica de pesquisa indicada para investigar fenômenos dentro de seu contexto real, especialmente quando os limites entre esse fenômeno e o seu contexto não estão claros (Yin, 1994). Em uma outra perspectiva, também é importante avaliar o método e a sua descrição com relação à viabilidade de aplicação em um projeto real. Para tanto, uma avaliação qualitativa sobre o método foi realizada com especialistas da área de arquitetura de software.

D.2 Planejamento do Estudo de Caso

Nesta seção são descritas as questões de pesquisa, a seleção dos sujeitos e objetos de estudo, os procedimentos para extração e análise de dados e também os riscos à validade do estudo de caso e as medidas tomadas para diminuir esses riscos. O planejamento é detalhado para que o estudo de caso possa ser replicado por outros pesquisadores.

Questões de Pesquisa (QP): são as perguntas que se deseja responder por meio da realização desse estudo de caso, sendo essas:

QP1: A representação da arquitetura de referência de sistemas embarcados elaborada com o ProSA-Re é mais eficiente? Deseja-se saber com essa questão se a representação elaborada requer menos esforço para leitura e análise.

QP2: A representação da arquitetura de referência de sistemas embarcados elaborada com o ProSA-Re promove o conhecimento de maneira consistente? Deseja-se saber com essa questão se os usuários da representação apresentam o mesmo entendimento sobre a arquitetura e se esse entendimento é melhor na nova representação.

Seleção dos Sujeitos: corresponde aos procedimentos adotados para selecionar os participantes do estudo de caso. Os perfis desejados para os sujeitos nesse estudo de caso são o de estudantes, pesquisadores ou especialistas das áreas de arquitetura de software, arquitetura de referência e/ou sistemas embarcados. Convites eletrônicos deverão ser enviados para todos os sujeitos em potencial que se encaixarem nesse perfil.

Objetos de Estudo: são as duas representações arquiteturais para o sistema de multiagentes locais, sendo a primeira a representação original elaborada por Weyns (2006) e a segunda a representação elaborada com o auxílio do ProSA-Re apresentada no Apêndice C.

Procedimentos para Coleta de Dados: os dados serão coletados por meio de um questionário respondido individualmente. O questionário é dividido em quatro partes. Na primeira parte, avalia-se o esforço necessário para a leitura da representação. Para isso, o sujeito informará o tempo dispendido nessa atividade. Na segunda parte, avalia-se a capacidade de compreensão do documento pelo sujeito por meio de algumas questões sobre o conteúdo e o projeto da arquitetura de referência. Ao todo, são propostas sete questões sobre a arquitetura de referência. O tempo dispendido nessa atividade e também o nível de dificuldade encontrado em cada questão serão coletados para servir de indicativo sobre a capacidade da representação arquitetural em transmitir de maneira consistente o conhecimento contido nela. Na sequência, a terceira parte do questionário pretende capturar percepções individuais sobre a representação arquitetural como, por exemplo, quais visões arquiteturais contribuíram para um melhor entendimento da arquitetura de referência, quais seções foram confusas ou difíceis de entender e ainda quais tipos de apresentação são mais eficazes (diagramas, textos ou tabelas). Por fim, na quarta parte são feitas algumas questões sobre o nível de conhecimento dos sujeitos em assuntos abordados pela arquitetura de referência. O modelo do questionário é apresentado na Seção D.2.1.

Procedimentos para Análise: os sujeitos serão divididos em dois grupos, sendo que o primeiro grupo irá analisar a representação original da arquitetura de referência, enquanto o segundo grupo irá analisar a representação construída com o apoio do ProSA-Re. É importante destacar que por conta dessa divisão, a terceira parte do questionário sobre a avaliação da representação) será adaptada para cada tipo de documento. Os resultados obtidos em cada grupo serão analisados separadamente e, em seguida, será feita uma análise comparativa dos resultados.

Projeto do Estudo de Caso: foi considerado um estudo de caso embarcado único (Gray, 2004). Segundo esse projeto, um único estudo de caso é conduzido, porém com diferentes

unidades de análise sendo observadas — correspondentes a cada um dos sujeitos do estudo de caso. O primeiro grupo será utilizado como controle do estudo de caso, permitindo que os resultados obtidos com a nova representação possam sem comparados.

Validade do Estudo de Caso: diferentes riscos foram identificados para esse estudo de caso. A seguir, para cada risco é apresentada uma breve explicação e as medidas de contenção tomadas são descritas.

- *Validade Interna:* esse risco está relacionado à realização de inferências sobre os dados. Para minimizar os riscos à validade interna, os sujeitos foram divididos em dois grupos, sendo que cada um recebeu um tipo de representação. Essa divisão permitiu que inferências fossem realizadas por meio da comparação dos resultados obtidos por cada grupo;

- *Validade Externa:* esse risco está relacionado à dificuldade de generalização dos resultados obtidos no estudo de caso a outras arquiteturas de referências. Para minimizar os riscos à validade externa, a seleção da arquitetura de referência foi baseada em características tipicamente presentes na população de arquiteturas de referência definidas pelo modelo de referência proposto por Nakagawa et al. (2012);

- *Validade de Construção:* esse risco está relacionado principalmente à dificuldade em se definir quais são os elementos sendo investigados. Para diminuir esse risco, no início do planejamento do estudo de caso, as principais concepções, isso é, o entendimento do pesquisador sobre os conceitos sob investigação, foram claramente definidos. Também será construída uma cadeia de evidências durante a coleta dos dados para permitir que outros pesquisadores possam validar os resultados obtidos;

- *Validade de Objetividade:* esse risco está relacionado ao viés nos resultados do estudo de caso adicionado pelo pesquisador. Para minimizar esse risco, a opinião e supervisão de especialistas durante todo o processo de avaliação foram fundamentais para a validação dos resultados e conclusões alcançadas. Para evitar que as questões sobre o projeto da arquitetura de referência beneficiassem a nova representação, as questões foram elaboradas a partir da representação original. Além disso, as representações foram padronizadas em um mesmo formato para não evidenciar para os sujeitos qual objeto estava sendo analisado; e

- *Confiabilidade:* esse risco está relacionado à validade dos resultados obtidos no estudo de caso. Para melhorar a confiabilidade do estudo de caso desenvolvido, as questões de pesquisa e o planejamento do estudo de caso foram documentados para que outros pesquisadores possam replicar o estudo de caso. Uma cadeia de

evidências também será criada para documentar as decisões e inferências realizadas sobre os dados durante o processo de coleta e análise das informações, possibilitando assim a reconfirmação dos resultados por outros pesquisadores.

Estudo Piloto: o planejamento e os questionários para coleta de dados foram validados e refinados em um estudo de caso piloto do qual participaram nove alunos de mestrado e doutorado do programa de pós-graduação do Instituto de Ciências Matemáticas e de Computação (ICMC/USP). É importante destacar que os alunos participaram voluntariamente do estudo e não apresentam qualquer relação com o projeto do ProSA-Re.

O estudo piloto foi conduzido por três alunos de doutorado e seguiu o planejamento do estudo de caso apresentado anteriormente com a única diferença de que os participantes tiveram um limite máximo de duas horas para completar o questionário na presença dos avaliadores. Por conta dessa restrição, o objeto de estudo foi reduzido, de modo que apenas um subconjunto da arquitetura de referência foi avaliado pelos participantes.

D.2.1 Modelo de Questionário para Avaliação da Representação da Arquitetura de Referência

Esta seção apresenta o conjunto de questões elaborado para avaliar a representação da arquitetura de referência para sistemas multiagentes locais. O questionário é estruturado em quarto partes:

- Na primeira parte, é solicitada a leitura completa da descrição da arquitetura de referência disponibilizada. Nenhum conhecimento prévio sobre a arquitetura de referência é assumido;
- Na segunda parte, são propostas algumas questões específicas sobre o projeto e conteúdo da arquitetura de referência analisada;
- Na terceira parte, pede-se que o sujeito avalie a representação com base em sua percepção; e
- Na última e quarta parte, são propostas questões gerais para mapear o conhecimento prévio dos sujeitos e que poderão ajudar em uma melhor análise dos dados.

Parte I: Leitura

O objetivo dessa parte do questionário é familiarizar o sujeito com a documentação. Acredita-se que essa atividade seja suficiente para que o sujeito tenha uma visão geral sobre o seu conteúdo.

1. Escreva o tempo total gasto para leitura da representação.
Resposta: _____

Parte II: Compreensão

O objetivo dessa parte do questionário é avaliar a compreensão do sujeito sobre a arquitetura de referência analisada. Espera-se que essa atividade seja suficiente para avaliar a usabilidade e eficiência da representação em transmitir o conhecimento contido na arquitetura de referência. As questões foram definidas de modo a cobrir toda a representação arquitetural.

Recomendações:

- Cada participante no estudo de caso deverá cronometrar o tempo total gasto para responder essa parte do questionário;
- Cada participante poderá consultar livremente a representação para responder cada uma das questões propostas. Em especial, o participante deve se empenhar para responder cada questão corretamente. Contudo, se ele não conseguir encontrar a informação solicitada ou se não tiver certeza sobre a opção escolhida, o sujeito deve selecionar a opção "Não sei"; e
- Ao final de cada questão, o participante deverá indicar ainda o nível de dificuldade encontrado para encontrar a informação solicitada.

2. Escreva o horário em que iniciou essa parte do questionário. Opcionalmente, o participante pode anotar o tempo total gasto.
Resposta: _____

3. Escolha a alternativa correta:
a) Um Agente é um interessado da arquitetura de referência
b) É preciso haver um Contexto de Implantação em todas as instâncias arquiteturais
c) O funcionamento do Ambiente de Aplicação é sujeito à restrições
d) Um Agente apenas pode enviar mensagens para outros Agentes por meio de troca de mensagens.
e) Não sei.

4. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 3?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

5. Quais dos módulos do Ambiente de Aplicação podem ser omitidos em instâncias arquiteturais? I-Tradução, II-Dinamismo, III - Serviço de Comunicação.
a) II
b) II, III
c) I, II
d) I, II, III
e) Não sei

6. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 5?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

7. Quais das seguintes portas são fornecidas pelo módulo Gerador de Representação do Ambiente de Aplicação? I-Sentir, II-Gerar, III-Ler.
a) I
b) I, II
c) II, III
d) I, II, III
e) Não sei

8. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 7?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

9. Quais dos seguintes aspectos não são abordados no projeto da arquitetura de referência, ou seja, são decisões arquiteturais abertas? I-Distribuição;II-Decomposição em Módulos; III-Interação Usuário-Computador; IV-Implantação.
a) I, III
b) II, III
c) II, III, IV
d) Todas as anteriores
e) Não sei

10. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 9?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

11. Quais dos seguintes processos estão relacionados à funcionalidade de comunicação?

- ☐ a) Gerando representação
- ☐ b) Observando/Processo de observação
- ☐ c) Sincronizando/Processo de sincronização
- ☐ d) Interagindo/Processo de Interação
- ☐ e) Traduzindo/Processo de tradução
- ☐ f) Não sei

12. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 11?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

13. Qual das seguintes declarações são falsas?

- ☐ a) O Contexto de Implantação depende do domínio de aplicação.
- ☐ b) Foco permite que um agente observe o ambiente em busca de um tipo de informação específica.
- ☐ c) Percepção é uma descrição do ambiente observado na forma de elementos de dados que podem ser usados para atualizar o conhecimento atual do agente.
- ☐ d) Conhecimento atual é um sinônimo para integração de conhecimento.
- ☐ e) Influências são ações produzidas pelo ambiente de aplicação.
- ☐ f) Não sei

14. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 13?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

15. Por favor, explique os motivos para a decisão de criar um repositório de conhecimento atual.

Não ultrapasse o espaço disponível

16. Com qual nota você avalia a dificuldade para encontrar a informação solicitada na questão 15.5?

	1	2	3	4	
Fácil	☐	☐	☐	☐	Difícil

17. Escreva o horário em que terminou essa parte do questionário. Opcionalmente, anote o tempo total gasto.
Resposta: _____

Gabarito: 3) C; 5) C; 7) B; 9) A; 11) E; 13) E.

Parte III: Percepção

Nessa parte do questionário, solicita-se que o participante avalie o conjunto de visões arquiteturais com relação à utilidade, esforço e clareza. Uma visão deve ser considerada útil se ajudou a responder alguma das questões anteriores sobre o projeto da arquitetura de referência. Uma visão deve ser considerada trabalhosa se exige uma quantidade de tempo razoável para ser adequadamente analisada ou escaneada por uma informação específica. Por fim, uma visão deve ser considerada confusa se é incapaz de comunicar aspectos relevantes da arquitetura de referência e que, provavelmente, não foi completamente compreendida.

18. Qual tipo de arquitetura de referência está sendo avaliada? A indicação de tipo de documento avaliado aparece na primeira página.
a) A
b) B

De acordo com a resposta do sujeito na questão 18, um dos seguintes conjuntos de perguntas será disponibilizado referente à representação A ou B.

Percepção da Representação A

A1. Modelo integrado

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

A2. Visão de decomposição em módulos

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

A3. Visão de Componentes e Conectores para Dados Compartilhados

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

A4. Visão de Componentes e Conectores para Componentes Colaborativos

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

A5. Visão de Componentes e Conectores para Processos Comunicativos

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

Percepção da Representação B

B1. Visão Conceitual

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

B2. Visão de Dados Compartilhados

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

B3. Visão de Componentes Colaborativos

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

B4. Visão de Processos

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

B5. Visão de Detalhes de Decisões Arquiteturais

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

B6. Visão de Variabilidade

	1 (menos)	2	3	4 (mais)
a) Útil	☐	☐	☐	☐
b) Trabalhosa	☐	☐	☐	☐
c) Confusa	☐	☐	☐	☐

Parte III: Continuação

19. Na sua opinião, qual das seguintes atividades tomou mais tempo?

a) Leitura
b) Buscar por informação específica
c) Entender os diagramas
d) Nenhuma das anteriores (se você gastou a mesma quantidade de tempo em todas as atividades)
e) Outra: _____

20. Como você avalia a importância de cada aspecto abaixo para o seu entendimento da representação?

	1 (indiferente)	2	3	4 (essencial)
a) Visão Primária/Diagramas	☐	☐	☐	☐
b) Variedade de visões apresentada	☐	☐	☐	☐
c) Descrição textual de cada visão	☐	☐	☐	☐

21. Você encontrou todas as informações requisitadas na documentação?

a) Sim
b) Não

22. Por favor, descreva as principais dificuldades que você encontrou para responder as questões.

Não ultrapasse o espaço disponível

23. Quais dos seguintes aspectos você destaca como positivos na representação da arquitetura de referência?

- ☐ a) Tamanho
- ☐ b) Diagramas
- ☐ c) Informação Textual
- ☐ d) Organização
- ☐ e) Concisão
- ☐ f) Outro: _____

24. Quais dos seguintes aspectos você destaca como negativos na representação da arquitetura de referência?

- ☐ a) Tamanho
- ☐ b) Diagramas
- ☐ c) Informação Textual
- ☐ d) Organização
- ☐ e) Concisão
- ☐ f) Outro: _____

25. Por favor, explique sua resposta nas questões sobre aspectos positivos e negativos da representação.

Não ultrapasse o espaço disponível

26. Na sua opinião, quais das seguintes considerações poderiam melhorar o seu entendimento sobre a representação analisada?

- ☐ a) Atividade em grupo
- ☐ b) Consultar um especialista no domínio
- ☐ c) Consultar um especialista em SysML
- ☐ d) Consultar um especialista na arquitetura de referência
- ☐ e) Mais tempo disponível paa leitura e resposta do questionário
- ☐ f) Melhor organização das informações
- ☐ g) Inclusão de outras visões
- ☐ h) Outro: _____

Parte IV: Habilidades Pessoais

Essa parte do questionário tem como objetivo mapear o conhecimento do participante sobre os temas envolvidos na representação arquitetural para ajudar na análise dos dados.

27. Como você avalia seu nível de experiência em engenharia de software?

	1	2	3	4	
Nada familiar	☐	☐	☐	☐	Muito familiar

28. Como você avalia seu nível de experiência em arquitetura de software?

	1	2	3	4	
Nada familiar	☐	☐	☐	☐	Muito familiar

29. Como você avalia seu nível de experiência em arquitetura de referência?

	1	2	3	4	
Nada familiar	☐	☐	☐	☐	Muito familiar

30. Como você avalia seu nível de experiência em UML?

	1	2	3	4	
Nada familiar	☐	☐	☐	☐	Muito familiar

31. Como você avalia seu nível de experiência em SysML?

	1	2	3	4	
Nada familiar	☐	☐	☐	☐	Muito familiar

D.2.2 Respostas Obtidas na Avaliação da Representação Arquitetural

Essa seção apresenta o conjunto de respostas obtidas na avaliação da representação da arquitetura de referência. A Tabela D.1 apresenta as respostas obtidas na parte geral do questionário, enquanto as Tabelas D.2 e D.3 separam as respostas obtidas na parte III do questionário referentes à representação A e B, respectivamente.

Tabela D.1: Respostas gerais obtidas no questionário

Questão	Sujeito 1	Sujeito 2	Sujeito 3	Sujeito 4
		Parte I: Leitura		
1 (HH:MM)	01:30	01:00	06:30	00:40
		Parte II: Compreensão		
2 (HH:MM)	15:00	-	14:25	16:20
3	C	B	B	D
4	3	2	3	2
5	D	C	C	D
6	3	2	4	2
7	B	B	A	B
8	3	2	3	2
9	A	A	A	A
10	3	3	4	3
11	E	Não sei	A, C, D	E
12	3	4	4	1
13	E	Não sei	C	D
14	3	4	3	2

Continua...

Tabela D.1 – Continuação

Questão	Sujeito 1	Sujeito 2	Sujeito 3	Sujeito 4
15	Possibilita que os componentes de comunicação e decisão compartilhem dados e se comuniquem indiretamente, permitindo que ambos os componentes atuem paralelamente e com ritmos diferentes. O repositório de dados encapsula as várias leis como elementos de primeira classe no sistema do agente, evitando assim que as leis sejam espalhadas por diferentes componentes do sistema.	Não sei	O repositório de conhecimento atual contém dados que são compartilhados entre diferentes assessores de dados. Os dados armazenados nesse repositório estão relacionados ao estado do ambiente percebido, ao estado relacionado aos papéis do agente e compromissos locais e, possivelmente, outros estados internos que são compartilhados. Os componentes de comunicação e tomada de decisão podem ler e escrever dados no repositório. O componente de percepção mantém o conhecimento do agente sobre o ambiente. Para atualizar esse conhecimento, os componentes de comunicação e tomada de decisão podem ativar o componente de percepção para sentir o ambiente.	O repositório de conhecimento atual contém dados que são compartilhados entre diferentes assessores de dados. Os dados armazenados nesse repositório estão relacionados ao estado do ambiente percebido, ao estado relacionado aos papéis do agente e compromissos locais e, possivelmente, outros estados internos que são compartilhados. Os componentes de comunicação e tomada de decisão podem ler e escrever dados no repositório. O componente de percepção mantém o conhecimento do agente sobre o ambiente. Para atualizar esse conhecimento, os componentes de comunicação e tomada de decisão podem ativar o componente de percepção para sentir o ambiente.
16	2	4	2	1
17	15:50	00:40	14:48	17:10
		Parte III: Percepção		
18	B	B	A	A
19	B	A	B	A
20 a)	4	4	3	4

Continua...

Tabela D.1 – Continuação

Questão	Sujeito 1	Sujeito 2	Sujeito 3	Sujeito 4
20 b)	4	1	4	4
20 c)	4	2	3	3
21	Sim	Não	Não	Sim
22	Algumas questões foram difíceis de responder porque elas estavam fora de ordem no documento. Eu não me senti confortável em afirmar que minha resposta estava completamente correta.	Documento muito longo. Foi difícil encontrar todas as informações.	Procurar por uma informação específica, porque o texto estava muito longo.	Falta de uma descrição de elementos mais específica. É importante entender parte da arquitetura de referência antes de se aprofundar na descrição de alguns elementos. Se uma descrição maior é necessária, é melhor conseguir um entendimento do todo e depois se aprofundar. Eu recomendaria fazer uma discriminação clara entre informações essenciais e informações adicionais.
23	B, C	D	B, C	B, Completude
24	Algumas notações no diagrama são confusas	A, B, C	A	C, D

Continua...

Tabela D.1 – Continuação

Questão	Sujeito 1	Sujeito 2	Sujeito 3	Sujeito 4
25	Podemos discutir depois.	O documento está muito longo e alguns diagramas não estão claros.	O texto está muito longo. Algumas partes poderiam ter sido inseridas como anexos. Se eu quiser saber de mais detalhes, eu posso ver o anexo.	Positivo: Bom conjunto de diagramas e informações apresentadas. Bom conteúdo, abrange todos (ou a maioria) dos aspectos da arquitetura de referência. Negativo: Muita informação textual, algumas podem ser representadas com outros tipos como tabelas ou diagramas. Boa organização, mas muita informação de uma vez, representando um processo difícil para compreensão real do conteúdo.
26	B, C, D	B, C, E	B, D, E	F, G
		Parte IV: Habilidades Pessoais		
27	4	4	4	4
28	4	2	3	3
29	4	2	3	3
30	4	4	4	3
31	2	1	3	2

Tabela D.2: Respostas obtidas para percepção da representação original

Questão	**Sujeito 3**	**Sujeito 4**
A1 a)	4	3
A1 b)	4	2
A1 c)	4	1
A2 a)	3	2
A2 b)	3	4
A2 c)	2	2
A3 a)	3	3
A3 b)	2	2
A3 c)	2	2
A4 a)	3	2
A4 b)	2	3
A4 c)	2	2
A5 a)	3	3
A5 b)	3	1
A5 c)	2	1

Tabela D.3: Respostas obtidas para percepção da representação elaborada com auxílio do ProSA-Re

Questão	**Sujeito 1**	**Sujeito 2**
B1 a)	4	1
B1 b)	2	1
B1 c)	1	1
B2 a)	3	1
B2 b)	1	1
B2 c)	1	1
B3 a)	4	3
B3 b)	4	3
B3 c)	2	3
B4 a)	3	1
B4 b)	2	2
B4 c)	3	2
B5 a)	4	1
B5 b)	4	3
B5 c)	1	2

Continua...

Tabela D.3 – Continuação

Questão	Sujeito 1	Sujeito 2
B6 a)	2	4
B6 b)	2	1
B6 c)	4	2

D.3 Planejamento da Avaliação com Especialistas

Nesta seção é apresentado o modelo de questionário adotado para a avaliação da descrição do método por especialistas da área de arquitetura de software.

D.3.1 Modelo de Questionário para Avaliação do ProSA-Re

Essa seção apresenta o conjunto de questões elaborado para a avaliação da descrição e das atividades do método ProSA-Re por especialistas. O questionário é estruturado em quarto partes:

- Na primeira parte, é feito um mapeamento do conhecimento dos especialistas para uma melhor análise das respostas obtidas;
- Na segunda parte, investiga-se a opinião dos especialistas sobre as motivações de um método para a representação de arquiteturas de referência, em especial para o domínio de sistemas embarcados;
- Na terceira parte, investiga-se a opinião dos especialistas sobre a descrição do método ProSA-Re;
- Na quarta parte, investiga-se a opinião dos especialistas sobre uma provável representação construída com auxílio do ProSA-Re;
- Na quinta parte, os especialistas avaliam possíveis benefícios e limitações que podem surgir com a utilização do método em projetos reais; e
- Na sexta e última parte, investiga-se a opinião dos especialistas sobre a viabilidade e perspectivas de utilização do ProSA-Re.

Parte I: Habilidades Pessoais

O objetivo dessa parte do questionário é mapear o conhecimento do participante para ajudar na análise das respostas obtidas.

1. Como você avalia seu nível de experiência em sistemas embarcados?

	1	2	3	4	5	
Iniciante	☐	☐	☐	☐	☐	Especialista

2. Como você avalia o seu nível de experiência em arquitetura de software?

	1	2	3	4	5	
Iniciante	☐	☐	☐	☐	☐	Especialista

3. Como você avalia o seu nível de experiência em arquitetura de referência?

	1	2	3	4	5	
Iniciante	☐	☐	☐	☐	☐	Especialista

4. Você já construiu uma arquitetura de referência?
a) Sim
b) Não

5. Você utilizou uma arquitetura de referência recentemente?
a) Sim
b) Não

6. Por favor, diga pelo menos uma arquitetura de referência com a qual você possui experiência.

Não ultrapasse o espaço disponível

7. Como você avalia o seu nível de experiência em UML?

	1	2	3	4	5	
Iniciante	☐	☐	☐	☐	☐	Especialista

8. Como você avalia o seu nível de experiência em SysML?

	1	2	3	4	5	
Iniciante	☐	☐	☐	☐	☐	Especialista

Parte II: Motivação

Essa parte da avaliação tem como finalidade capturar percepções individuais sobre a proposta de um novo método para representação de arquiteturas de referência, em especial, para o domínio de sistemas embarcados.

10. Na sua opinião, é relevante ter um método específico para a representação de arquiteturas de referência de sistemas embarcados?
a) Sim
b) Em termos
c) Não

11. Por favor, justifique sua resposta anterior.

Não ultrapasse o espaço disponível

Parte III: Descrição

Essa parte da avaliação investiga a percepção do participante com relação à descrição do método.

12. Dado que o método define as entradas para cada tarefa, como você avalia a descrição desse elemento?

	1	2	3	4	5	
Confusa	☐	☐	☐	☐	☐	Clara

13. Dado que o método define as saídas para cada tarefa, como você avalia a descrição desse elemento?

	1	2	3	4	5	
Confusa	☐	☐	☐	☐	☐	Clara

14. Dado que o método define os papéis envolvidos na execução de cada tarefa, como você avalia a descrição desse elemento?

	1	2	3	4	5	
Confusa	☐	☐	☐	☐	☐	Clara

15. Considerando toda a documentação do método ProSA-Re, como você avalia a sua descrição? Com essa questão, queremos verificar se você considera a informação disponível suficiente para viabilizar a aplicação do método em um projeto real.

	1	2	3	4	5	
Insuficiente	☐	☐	☐	☐	☐	Completa

16. Por favor, escreva aqui sugestões de melhorias para a descrição do método.

Não ultrapasse o espaço disponível

Parte IV: Representação

Essa parte da avaliação captura percepções individuais sobre prováveis representações arquiteturais construídas com o auxílio do método ProSA-Re.

17. Considerando a documentação do método, quais são as suas expectativas com relação à representações construídas com o seu apoio? De qual modo você acredita que o método poderia contribuir para melhorar a descrição atual dessas arquiteturas?

Não ultrapasse o espaço disponível

18. Na sua opinião, as representações construídas com o apoio do ProSA-Re podem ser úteis para diferentes tipos de interessados? Você acredita que interessados com diferentes níveis de experiência podem acessar a representação arquitetural elaborada?

Não ultrapasse o espaço disponível

19. Considerando a categoria de pontos de vista transversais proposta pelo ProSA-Re, com quais das seguintes afirmações você concorda?

- ☐ a) Essa categoria de pontos de vista é essencial para todas as arquiteturas de referência de sistemas embarcados.
- ☐ b) Os pontos de vista incluídos nessa categoria colaboram para a compreensão do projeto da arquitetura de referência.
- ☐ c) As técnicas sugeridas para construir cada ponto de vista são suficientes.
- ☐ d) Os pontos de vista incluídos nessa categoria são irrelevantes para o domínio de sistemas embarcados.
- ☐ e) As técnicas sugeridas não capturam todos os aspectos relevantes dos pontos de vista incluídos nessa categoria.
- ☐ f) Outro: _____

20. Considerando a categoria de pontos de vista em tempo de execução proposta pelo ProSA-Re, com quais das seguintes afirmações você concorda?

- ☐ a) Essa categoria de pontos de vista é essencial para todas as arquiteturas de referência de sistemas embarcados.
- ☐ b) Os pontos de vista incluídos nessa categoria colaboram para a compreensão do projeto da arquitetura de referência.
- ☐ c) As técnicas sugeridas para construir cada ponto de vista são suficientes.
- ☐ d) Os pontos de vista incluídos nessa categoria são irrelevantes para o domínio de sistemas embarcados.
- ☐ e) As técnicas sugeridas não capturam todos os aspectos relevantes dos pontos de vista incluídos nessa categoria.
- ☐ f) Outro: _____

21. Considerando a categoria de pontos de vista de código fonte proposta pelo ProSA-Re, com quais das seguintes afirmações você concorda?

- ☐ a) Essa categoria de pontos de vista é essencial para todas as arquiteturas de referência de sistemas embarcados.
- ☐ b) Os pontos de vista incluídos nessa categoria colaboram para a compreensão do projeto da arquitetura de referência.
- ☐ c) As técnicas sugeridas para construir cada ponto de vista são suficientes.
- ☐ d) Os pontos de vista incluídos nessa categoria são irrelevantes para o domínio de sistemas embarcados.
- ☐ e) As técnicas sugeridas não capturam todos os aspectos relevantes dos pontos de vista incluídos nessa categoria.
- ☐ f) Outro: _____

22. Considerando a categoria de pontos de vista de implantação proposta pelo ProSA-Re, com quais das seguintes afirmações você concorda?

- ☐ a) Essa categoria de pontos de vista é essencial para todas as arquiteturas de referência de sistemas embarcados.
- ☐ b) Os pontos de vista incluídos nessa categoria colaboram para a compreensão do projeto da arquitetura de referência.
- ☐ c) As técnicas sugeridas para construir cada ponto de vista são suficientes.
- ☐ d) Os pontos de vista incluídos nessa categoria são irrelevantes para o domínio de sistemas embarcados.
- ☐ e) As técnicas sugeridas não capturam todos os aspectos relevantes dos pontos de vista incluídos nessa categoria.
- ☐ f) Outro: _____

23. Considerando as quatro categorias de pontos de vista (além da categoria de decisões arquiteturais), você acredita que elas sejam suficientes para descrever arquiteturas de referência do domínio de sistemas embarcados?
a) Sim
b) Em termos
c) Não

24. Por favor, escreva aqui comentários e sugestões para o conjunto de categorias proposto ou sobre as técnicas selecionadas.

Não ultrapasse o espaço disponível

25. Considerando a lista de ferramentas que oferecem apoio à criação de diagramas na linguagem SysML, quais delas você provavelmente adotaria?

☐ a) Papyrus (Eclispe)
☐ b) Rhapsody Developer (IBM)
☐ c) Enterprise Architect (Sparx Systems)
☐ d) MagicDraw (No Magic)
☐ e) Outro: _____

26. Por favor, justifique sua resposta anterior.

Não ultrapasse o espaço disponível

Parte V: Benefícios e Limitações

Essa parte da avaliação tem como finalidade identificar a opinião de especialistas sobre sobre aspectos positivos e negativos que podem surgir com a adoção do ProSA-Re.

27. Por favor, selecione potenciais benefícios que a adoção do método ProSA-Re pode trazer para o processo de desenvolvimento de arquiteturas de referência de sistemas embarcados.

- ☐ a) Maior produtividade da atividade de representação
- ☐ b) Menor burocracia para criação da representação
- ☐ c) Menor esforço para compreender a representação criada
- ☐ d) Maior clareza dos diagramas criados
- ☐ e) A arquitetura de referência é mais fácil de compreender
- ☐ f) A representação da arquitetura de referência é abrangente
- ☐ g) A representação elaborada é mais fácil de manter e evoluir
- ☐ h) Outro: _____

28. Por favor, selecione potenciais limitações que a adoção do método ProSA-Re pode trazer para o processo de desenvolvimento de arquiteturas de referência de sistemas embarcados.

- ☐ a) Menor produtividade da atividade de representação
- ☐ b) Maior burocracia para criação da representação
- ☐ c) Maior esforço para compreender a representação criada
- ☐ d) Diagramas criados são mais complexos
- ☐ e) A arquitetura de referência é mais difícil de compreender
- ☐ f) Menor liberdade para a criação de representações específicas
- ☐ g) A manutenção e evolução da representação da arquitetura de referência é mais complicada
- ☐ h) Os pontos de vista são insuficientes para capturar interesses específicos da arquitetura de referência
- ☐ i) Os pontos de vista são insuficientes para capturar interesses específicos de sistemas embarcados
- ☐ j) Outro: _____

29. O quanto você acredita que a representação de arquiteturas de referência de sistemas embarcados podem ser beneficiadas pela adoção do ProSA-Re?

	1	2	3	4	5	
Pouco	☐	☐	☐	☐	☐	Muito

30. Por favor, justifique sua resposta anterior.

Não ultrapasse o espaço disponível

Parte VI: Conclusão

Essa parte da avaliação tem como finalidade identificar a percepção geral do especialista sobre o método ProSA-Re.

31. Na sua opinião, é interessante ou até mesmo viável aplicar uma abordagem incremental para a representação da arquitetura de referência de sistemas embarcados?
a) Sim
b) Em termos
c) Não

32. Por favor, justifique sua resposta anterior.

Não ultrapasse o espaço disponível

33. Você adotaria o ProSA-Re se tivesse que construir a representação de uma arquitetura de referência para o domínio de sistemas embarcados?
a) Sim
b) Talvez
c) Não

34. Na sua opinião, o ProSA-Re é uma contribuição importante para a comunidade de sistemas embarcados?
a) Sim
b) Em termos
c) Não

35. Na sua opinião, o ProSA-Re pode melhorar as perspectivas de disseminação de arquiteturas de referência do domínio de sistemas embarcados?
a) Sim
b) Em termos
c) Não

36. Por favor, justifique sua resposta anterior.

Não ultrapasse o espaço disponível

37. Por favor, escreva aqui comentários e sugestões para melhorar o ProSA-Re.

Não ultrapasse o espaço disponível