Cássio Capucho Peçanha

RASO: uma Ontologia Sobre Requisitos para o Desenvolvimento de Sistemas Adaptativos

Vitória, ES

2019

Cássio Capucho Peçanha

RASO: uma Ontologia Sobre Requisitos para o Desenvolvimento de Sistemas Adaptativos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2019

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Peçanha, Cássio Capucho, 1991-
P364r RASO: uma ontologia sobre requisitos para o desenvolvimento de sistemas adaptativos / Cássio Capucho Peçanha. - 2019.
82 f. : il.

Orientador: Vítor Estêvão Silva Souza.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Software. 2. Engenharia de software. 3. Ontologia. 4. Sistemas Adaptativos. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

Cássio Capucho Peçanha

RASO: uma Ontologia Sobre Requisitos para o Desenvolvimento de Sistemas Adaptativos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 07 de agosto de 2019:

Prof. Dr. Vítor E. Silva Souza
Orientador

Renata Silva Souza Guizzardi, Ph.D.
Universidade Federal do Espírito Santo

Julio Cesar Sampaio do Prado Leite, Ph.D.
Pontifícia Universidade Católica do Rio de Janeiro

Vitória, ES
2019

Agradecimentos

Existes várias pessoas que tiveram um papel crucial nesta minha caminhada, alguns em especial preciso destacar, pois foram de suma importância neste período.

Inicialmente gostaria de agradecer a minha amiga Sabrina Panceri, quem me incentivou inicialmente a entrar nessa jornada e seguiu nela comigo.

Gostaria de agradecer ao meu marido Michel Beccalli, que me deu todo apoio durante este período.

Agradeço também a minha psicóloga Luana Sartori, quem aguentou todas as crises de ansiedade e me proveu condições de manter minha saude mental até este momento.

Agradeço ao meu orientador Vítor Souza, por todo aprendizado, toda paciência e motivação que me fizeram concluir mais essa etapa.

Obrigado a todos.

Resumo

Existe um interesse crescente em softwares que podem adaptar seu comportamento para lidar com desvios entre seus resultados e seus requisitos em tempo de execução. Um mapeamento sistemático da literatura sobre abordagens de auto-adaptação com base em modelos de requisitos revelou mais de 200 artigos sobre o assunto.

No entanto, ainda não há uma representação formal e explícita dos conceitos neste domínio, o que pode levar a problemas de comunicação, aprendizagem, resolução de problemas, interoperabilidade, etc. Para fazer uma descrição clara e precisa deste domínio, este trabalho propõe a uma ontologia de requisitos para sistemas adaptativos (RASO, *the Requirements-based Adaptive Systems Ontology*).

A RASO foi construída utilizando-se de um método bem estabelecido de Engenharia de Ontologia, baseia-se em uma ontologia de fundamentação e reutiliza conceitos de outras ontologias relacionadas a software. A ontologia foi avaliada por meio do mapeamento de construtos das abordagens mais referenciadas da literatura para seus conceitos, criando assim um caminho para a interoperabilidade entre eles.

Palavras-chaves: Sistemas Adaptativos. Requisitos. Modelos. Ontologia.

Abstract

There is growing interest in software that can adapt their behavior to deal with deviations between their outcome and their requirements at runtime. A systematic mapping of the literature on self-adaptation approaches based on requirements models revealed over 200 papers on this subject.

However, there is still a lack of a formal and explicit representation of the concepts in this domain, which can lead to problems in communication, learning, problem-solving, interoperability, etc. To make a clear and precise description of this domain, this work proposes RASO: the Requirements-based Adaptive Systems Ontology.

RASO was built using a well-established Ontology Engineering method, is grounded on a foundational ontology and reuses concepts from other software-related ontologies. The ontology was evaluated by mapping constructs from the most referenced approaches from the literature to its concepts, thus creating a path for interoperability among them.

Keywords: Adaptive Systems · Requirements · Models · Ontology.

Lista de ilustrações

Figura 1 – Fases e processos de apoio do método SABiO (FALBO, 2014). 26
Figura 2 – Modelo de SEON e o relacionamento das ontologias (DUARTE et al., 2018). 28
Figura 3 – Visão de SEON mostrando conceitos e relações utilizadas por RASO. . 30
Figura 4 – Visão geral do protocolo de mapeamento sistemático para a RASO. . . 36
Figura 5 – Distribuição de publicações ao longo dos anos. 37
Figura 6 – Distribuição de publicações por continente. 37
Figura 7 – Distribuição de publicações por país. 38
Figura 8 – Modelo de objetivos KAOS para um sistema de lavanderia (BARESI; PASQUALE; SPOLETINI, 2010). 41
Figura 9 – Modelo de adaptive goals para o sistema de lavanderia (BARESI; PASQUALE; SPOLETINI, 2010). 41
Figura 10 – Modelo contextual de Adaptive STS descrevendo um cenário de cuidados médicos (DALPIAZ; GIORGINI; MYLOPOULOS, 2012). 44
Figura 11 – Visão geral do *framework* Zanshin em ação. (SOUZA, 2012) 46
Figura 12 – Modelo de Requisitos do Sistema Agendador de Reuniões, criado com a abordagem Zanshin (SOUZA, 2012) 46
Figura 13 – Representação gráfica dos conceitos de modelagem apresentados por Morandini et al. (2017). 49
Figura 14 – Modelo de infraestrutura de adaptação (GOLDSBY et al., 2008). . . . 51
Figura 15 – Modelo de adaptação de S1 para S2 (GOLDSBY et al., 2008). 52
Figura 16 – A Requirements-based Adaptive Systems Ontology (RASO). 55

Lista de tabelas

Tabela 1 – Questões de pesquisa definidas para o Mapeamento Sistematico. 34
Tabela 2 – Terms used in the search string. 35
Tabela 3 – Critérios de Inclusão e Exclusão de artigos do Mapeamento Sistemático. 36
Tabela 4 – Os 10 artigos mais citados do mapeamento e suas respectivas abordagens. 39
Tabela 5 – Verificação das Questões de Competência de RASO. 60
Tabela 6 – Validação da RASO pela instanciação de seus conceitos de acordo com abordagens mais populares para o desenvolvimento de sistemas adaptativos (parte 1). 61
Tabela 7 – Validação da RASO pela instanciação de seus conceitos de acordo com abordagens mais populares para o desenvolvimento de sistemas adaptativos (parte 2). 62
Tabela 8 – Artigos de Controle (parte 1). 78
Tabela 9 – Artigos de Controle (parte 2). 79
Tabela 10 – Artigos de Controle (parte 3). 80

Lista de abreviaturas e siglas

CI	Critério de Inclusão
CORE	Core Ontology for Requirement Engineering
ER	Engenharia de Requisitos
MAPE	Monitoring, Analysis, Planning and Execution
MDRC	Monitor-Diagnose-Reconcile-Compensate
OWL	Ontology Web Language
QC	Questão de Competência
QoS	Quality of Service
QP	Questão de Pesquisa
RASO	Requirements-based Adaptive Systems Ontology
RRA	Runtime Requirement Artifacts
RRO	Runtime Requirements Ontology
RRT	Requirements at Runtime
RSRO	Reference Software Requirements Ontology
SEON	Software Engineering Ontology Network
SPO	Software Process Ontology
STS	Socio-technical System
SWO	Software Ontology
UFO	Unified Foundational Ontology
UML	Unified Modeling Language

Sumário

1 INTRODUÇÃO 19
1.1 Motivação 19
1.2 Objetivos 20
1.3 Método 20
1.4 Organização da Dissertação 21

2 REVISÃO DA LITERATURA 23
2.1 Sistemas Adaptativos 23
2.2 Fundamentação Ontológica 25
2.3 Trabalhos Relacionados 30

3 MAPEAMENTO SISTEMÁTICO: REQUISITOS PARA SISTEMAS ADAPTATIVOS 33
3.1 FLAGS 40
3.2 RELAX 41
3.3 Adaptive STS 42
3.4 Zanshin 45
3.5 A abordagem aplicada ao Necesity 46
3.6 Tropos4AS 47
3.7 QoS-aware Middleware 48
3.8 LoREM 49

4 RASO 53

5 AVALIAÇÃO DO TRABALHO 59
5.1 Verificação e Validação da Ontologia 59
5.2 Ameaças à Validade 63

6 CONSIDERAÇÕES FINAIS 67

REFERÊNCIAS 69

APÊNDICES 75

APÊNDICE A – ARTIGOS DE CONTROLE 77

1 Introdução

Este trabalho refere-se à produção de uma ontologia de referência de domínio chamada RASO — *Requirements-based Adaptive Systems Ontology* — e construída a partir das necessidades observadas durante pesquisas sobre modelos de requisitos para sistemas adaptativos e elaborada a partir de um mapeamento sistemático da literatura desta área. O presente capítulo apresenta uma visão geral da pesquisa realizada.

1.1 Motivação

Nas últimas décadas, as pesquisas sobre sistemas adaptativos têm ganhado evidência e consequentemente um número crescente de trabalhos tem sido realizado. Atuando de modo a gerenciar a crescente complexidade de sistemas, tais trabalhos são caracterizados por propostas de sistemas capazes de monitorar seu ambiente e adaptar seu comportamento de modo a satisfazer seus requisitos (SOUZA, 2012). Normalmente, tais sistemas são constituídos de um sistema base, definido para cumprimento dos propósitos principais, e um controlador adaptativo, responsável pelo monitoramento do sistema base e, através de análise de contexto, verificar a satisfação dos requisitos e realizar as adaptações adequadas.

Parte destes esforços estão direcionados à Engenharia de Requisitos para Sistemas Adaptativos (BARESI; PASQUALE; SPOLETINI, 2010; WHITTLE et al., 2010; DALPIAZ; GIORGINI; MYLOPOULOS, 2012; SOUZA et al., 2011), buscando evidenciar quais adaptações são possíveis e de que modo estas podem ser realizadas (CHENG et al., 2009a). A Engenharia de Requisitos contempla processos por meio dos quais se descobre o propósito do sistema, identificam-se as partes interessadas e suas necessidades e documenta-se todo o processo. O processo de elicitação de requisitos envolve a compreensão do domínio e do problema a ser solucionado, bem como todos os itens que possam estar relacionados com este (YANG et al., 2014).

Devido à sua complexidade e suas particularidades, sistemas adaptativos necessitam de uma atenção especial sobre o processo de adaptação. Assim, engenheiros de requisitos necessitam identificar as situações que requerem adaptações, o sistema que será monitorado, o que adaptar, quando adaptar e como adaptar (YANG et al., 2014).

Atualmente, na literatura científica, muitas pesquisas fazem o uso de modelos de requisitos para desenvolvimento de sistemas adaptativos, mas até o momento não há uma conceituação compartilhada amplamente aceita e consolidada sobre o assunto. Portanto, a área apresenta alguns problemas a serem tratados:

- Não há na literatura uma formalização amplamente aceita para a área de requisitos para sistemas adaptativos que possa ser utilizada como modelo de referência de domínio;
- Não existe na literatura uma representação ontológica que abranja todos os principais conceitos mapeados atualmente;
- Devido às diversas abordagens fruto de pesquisas recentes não possuírem uma base compartilhada de conceitos, a falta de consenso na área gera ambiguidade sobre os termos utilizados.

Assim sendo, foi realizado um mapeamento sistemático de estudos científicos sobre o uso de modelo de requisitos para sistemas adaptativos. Sobre este resultado foi produzida uma ontologia de referência que possa ser utilizada como vocabulário formal para o domínio. Detalhes sobre os procedimentos metodológicos serão apresentadas posteriormente.

1.2 Objetivos

Este trabalho possui o objetivo geral de oferecer uma representação formal para o uso de modelo de requisitos para sistemas adaptativos, com uma descrição precisa e bem fundamentada dos conceitos do domínio em questão, eliminando dúvidas e ambiguidades sobre estes. Este objetivo foi decomposto em:

- Entendimento do domínio em questão, buscando conceitos compartilhados sobre o assunto por meio de estudos sobre a literatura científica existente;
- Construção de um modelo de referência sobre o domínio e sua aplicação numa análise das abordagens baseadas em requisitos para construção de sistemas adaptativos.

1.3 Método

Para alcançar os objetivos deste trabalho, foram realizadas as seguintes atividades:

- Revisão da literatura: esta etapa se iniciou com estudos encontrados de formas não sistemáticas a fim de prover um conhecimento inicial sobre o assunto. Buscando conhecer o estado da arte sobre a utilização de modelos de requisitos para sistemas adaptativos, foi realizado em seguida um mapeamento sistemático da literatura. Esta pesquisa foi utilizada como meio principal para elicitação de informações, visto que esta se mostra como método amplamente aceito dentro da literatura para aquisição de conhecimento. Para isso, foram seguidas as diretrizes apresentadas por Kitchenham e Charters (2007);

- Construção da ontologia: após realizado o processo de mapeamento sistemático, os resultados serviram como base para construção de uma ontologia de referência de domínio denominada RASO (*Requirements-based Adaptive Systems Ontology*), baseada na ontologia de fundamentação UFO (GUIZZARDI, 2005) e desenvolvida de acordo com as diretrizes do método SABiO (FALBO, 2014), que também apoiou a integração de ontologias já existentes e bem consolidadas neste contexto;

- Avaliação da ontologia: conforme indicado nas diretrizes do método SABiO, uma ontologia de referência pode ser avaliada pelos processos de verificação e validação (processos descritos por Falbo (2014) no método SABiO). A verificação realizada se dá pelo cumprimento das questões de competência levantadas (GRÜNINGER; FOX, 1995). A validação foi realizada por meio da instanciação da ontologia RASO com as principais e mais populares arquiteturas propostas na literatura que fazem uso de modelo de requisitos para sistemas adaptativos, apresentadas em nosso mapeamento sistemático.

1.4 Organização da Dissertação

Neste primeiro capítulo, foram apresentadas de forma sucinta as ideias iniciais que compõem esta dissertação, evidenciando a origem de nossa pesquisa e a motivação para realização. Também descrevemos os métodos aplicados, e que serão detalhados nos próximos capítulos. Além deste, esta dissertação apresenta os seguintes capítulos:

- Capítulo 2 – Revisão da Literatura: neste capítulo é apresentada toda nossa base para condução desta pesquisa. A discussão é iniciada apresentando abordagens baseadas em requisitos para sistemas adaptativos e assim são trabalhados os alvos principais de nossa pesquisa, os sistemas adaptativos. Aqui se faz presente uma descrição do estado-da-arte e os principais métodos associados. Seguido, neste mesmo capítulo, está nossa fundamentação ontológica, descrevendo os principais conceitos das ontologias utilizadas de base, e o método utilizado que permitiu o reuso destas ontologias e apoiou a construção da RASO.

- Capítulo 3 – Mapeamento Sistemático: apresenta uma descrição completa e o método aplicado para realização do mapeamento sistemático sobre sistemas adaptativos baseados em modelo de requisitos, fonte de nossos dados utilizados para criação de nossa ontologia. As informações extraídas deste mapeamento também serviram como auxílio no processo de validação da ontologia.

- Capítulo 4 – RASO (*Requirements-based Adaptive Systems Ontology*): neste capítulo é detalhado todo o processo de construção da ontologia, envolvendo desde o mapeamento sistemático, aplicação do método SABiO para construção e reuso de

ontologias, até o resultado final, a ontologia RASO apresentada em seu modelo conceitual.

- Capítulo 5 – Avaliação da Ontologia: este capítulo descreve o processo de verificação e validação da ontologia, seguindo o método SABiO e utilizando o fruto do mapeamento sistemático de apoio. Assim, pretende-se garantir a qualidade da ontologia construída e a certeza do cumprimento dos requisitos propostos.

- Capítulo 6 – Conclusão: nesta etapa são expostas as conclusões obtidas nesta pesquisa e evidenciadas as principais contribuições da mesma, além de apresentar possibilidades de continuação e possíveis trabalhos futuros.

O trabalho descrito nesta dissertação gerou uma publicação no *Workshop de Engenharia de Requisitos* (PEÇANHA; DUARTE; SOUZA, 2018), selecionada como um dos 5 melhores artigos do *workshop*, posteriormente estendida e publicada no periódico *Cadernos do IME* (PEÇANHA et al., 2018).

2 Revisão da Literatura

Neste capítulo é apresentado o estado-da-arte sobre os principais temas desta pesquisa e uma análise dos trabalhos relacionados, a fim de prover um embasamento teórico para contextualização e análise das discussões apresentadas nos capítulos que seguem. O capítulo está dividido em duas seções: a Seção 2.1 aborda Sistemas Adaptativos, explicando sobre os conceitos envolvidos e sua estrutura, enquanto a Seção 2.2 trata da Fundamentação Ontológica aplicada ao desenvolvimento da ontologia proposta, incluindo o método de desenvolvimento, sua ontologia de fundamentação e as ontologias reutilizadas neste contexto. A Seção 2.3 discute trabalhos relacionados, identificando a lacuna que RASO visa preencher nas pesquisas sobre sistemas adaptativos.

2.1 Sistemas Adaptativos

Atualmente, com a expansão da informação, sistemas de software tem de lidar com questões como integração de novas tecnologias e contínua evolução, além de atuarem em ambientes abertos, dinâmicos e imprevisíveis. Percebe-se também o crescente envolvimento de seres humanos e organizações nas estruturas e operações do sistema, à medida que aspectos da vida humana são automatizados ou assistidos por programas de computador. Como consequência, estes sistemas não apenas crescem em tamanho como em variabilidade e, assim, expandindo o número de requisitos para suprir a incertezas do ambiente. Estas questões afetam os sistemas, não apenas em sua fase de desenvolvimento, mas também em sua manutenção e evolução, mesmo após sua entrada em operação, surgindo assim a necessidade de novas abordagens que suportem mudanças (SOUZA, 2012).

Para isso, são necessários sistemas mais flexíveis, versáteis, resilientes, recuperáveis, e auto-otimizáveis, sistemas que se adaptam à mudança operacional de contexto, ambiente ou falha em sua própria operação. Assim, auto-adaptação tem se tornado um importante tópico de pesquisa (CHENG et al., 2009a; LEMOS et al., 2013).

O elemento que faz a auto-adaptação possível é comumente um software. Como a adaptação está associada a um contexto específico, ela se emprega a diversas áreas de aplicação e tecnologias, tais como: interfaces de usuário, sistemas embarcados, robôs móveis e autônomos e sistemas multiagentes. Em muitas áreas alguns aspectos da auto-adaptação foram introduzidos em sua própria perspectiva, mas somente nas últimas décadas iniciaram as tentativas de construir sistemas auto-adaptativos em domínios de aplicações específicos. No entanto, a adaptação adequada ainda é um desafio e pouco esforço foi feito para criar abordagens apropriadas de Engenharia de Software para fornecer funcionalidade de auto-adaptação (CHENG et al., 2009a).

Como definido por Laddaga e Robertson (2004), um software auto-adaptável avalia seu comportamento e assim, quando não consegue realizar as ações para que foi especificado ou ainda quando necessita de um melhor desempenho, realiza mudanças para que contemplem estas situações. Isso implica que um software possa ter diversas maneiras de cumprir seu propósito e possui conhecimento suficiente para avaliar e conduzir mudanças em tempo de execução. Nele, devem ser incluídas funcionalidades para avaliar seu comportamento e desempenho, bem como a capacidade de planejar e reconfigurar suas operações.

Como exposto por Souza (2012), muitos pesquisadores utilizam os termos autônomo e auto-adaptativo de forma intercambiável. Estes termos possuem semelhanças, mas são distintos no sentido de que, a computação autônoma possui uma maior abrangência, incluindo desde aplicação até o *hardware*, enquanto sistemas auto-adaptativos são voltados a aplicação e *middleware*. Outro ponto exposto se refere à motivação das pesquisas, as quais apontam que um sistema autônomo comumente está direcionado ao custo de manutenção e possui soluções voltadas a questões arquitetônicas para automatização de tarefas de manutenção, enquanto o sistema auto-adaptativo direciona-se sobre contextos sociais abertos com alto grau de incerteza e suas soluções focam em requisitos do usuário e seu ambiente. Nesse sentido, a pesquisa em sistemas adaptativos teria um escopo mais amplo, considerando todo o processo de desenvolvimento de software para a operação.

O presente estudo está direcionado a soluções sobre sistemas auto-adaptativos. Porém, como exposto anteriormente, devido ao uso indiscriminado de alguns termos, a pesquisa realizada para elicitação de informações bibliográficas da área necessitou de explorar um número maior de termos de forma a garantir uma maior cobertura na descoberta de soluções da área.

Para que um sistema auto-adaptativo proveja uma adaptação é necessário que possua um retorno sobre sua execução. Comumente observado nas abordagens, de forma explícita ou não, o elemento responsável pela adaptação é um tipo de ciclo de realimentação (*feedback loop*) que executa os papéis de monitoramento, diagnóstico, entre outras funções. Em (KEPHART; CHESS, 2003a) temos como exemplo um ciclo MAPE, representando suas quatro atividades: monitorar, analisar, planejar e executar.

Embora com grande relevância para as propostas, por vezes o *feedback loop* é um artefato negligenciado e, recentemente, pesquisadores têm destacado a necessidade de uma maior exploração do *feedback loop* em projetos de sistemas adaptativos. Diferentes abordagens são desenvolvidas e, devido à amplitude da área, diversas formas de *feedback loops* podem ser elaborados de acordo com cada uma destas abordagens (SOUZA, 2012).

Em síntese, os ciclos de realimentação preveem meios para que os sistemas adaptativos possam monitorar indicadores e quando o sistema em questão não está funcionando adequadamente, tomar as medidas corretivas realizar adaptação.

Como em qualquer outro sistema, a adaptação também deve ser descrita na especificação de requisitos e a ausência desta informação pode impactar sobre a entrega e qualidade do sistema produzido (ROSS; SCHOMAN, 1977). A Engenharia de Requisitos é o primeiro estágio do ciclo de vida do desenvolvimento de *software* e, diferente da tradicional, a Engenharia de Requisitos para Sistemas Adaptativos foca na definição da lógica de adaptação, apontando quais mudanças no ambiente e no próprio sistema devem ser monitoradas, o que adaptar, quando adaptar e como adaptar.

Neste desenvolvimento, engenheiros devem levar em consideração a lógica de domínio e a lógica de adaptação (YANG et al., 2014). A definição de requisitos deve conter todas as informações necessárias para estabelecer uma base para os estágios subsequentes no desenvolvimento do sistema. É preciso informar qual o propósito do sistema, relacionar com condições atuais ou previstas, estabelecer recursos principais e por quais quais critérios de satisfação deverão ser mantidos, além de informar como o sistema deverá ser construído. A negligência com estas questões poderá recair sobre o projeto na forma de atraso no desenvolvimento, no alto custo de produção e resultar em insatisfação do cliente (ROSS; SCHOMAN, 1977).

Como apresentado por Duarte (2016) em sua pesquisa, o requisito é uma entidade complexa que pode existir durante todo o ciclo de vida de um software, tanto em tempo de desenho quanto em tempo de execução e pode ser representado de diversas formas dentro do processo de modelagem conceitual. Levando em consideração que estes são originados dos objetivos das partes interessadas, os requisitos podem ser descritos de várias formas, tornando-se artefatos. Atuando com sistemas adaptativos, o requisito deverá ser processado e consumido por uma máquina durante a execução do sistema, e por isso, o mesmo deverá ser descrito em uma linguagem que permita um raciocínio da máquina sobre ele (DUARTE, 2016).

Existem na literatura diversas abordagens que fazem o uso de modelo de requisitos para sistemas adaptativos, e.g. (SOUZA, 2012; BARESI; PASQUALE; SPOLETINI, 2010; SAWYER et al., 2010; MORANDINI et al., 2017; GOLDSBY et al., 2008), e cada uma delas possui métodos distintos de realizar a adaptação. Algumas das principais abordagens reveladas no mapeamento sistemático aqui realizado, foram apresentadas e descritas no Capítulo 3. Estas abordagens foram utilizadas nesta pesquisa como material de embasamento e fonte de dados para construção da ontologia RASO, apresentada no Capítulo 4, bem como auxílio na avaliação da mesma, demonstrada no Capítulo 5.

2.2 Fundamentação Ontológica

A ontologia RASO foi construída utilizando o método SABiO: *Systematic Approach for Building Ontologies* (FALBO, 2014), uma abordagem sistemática para construção

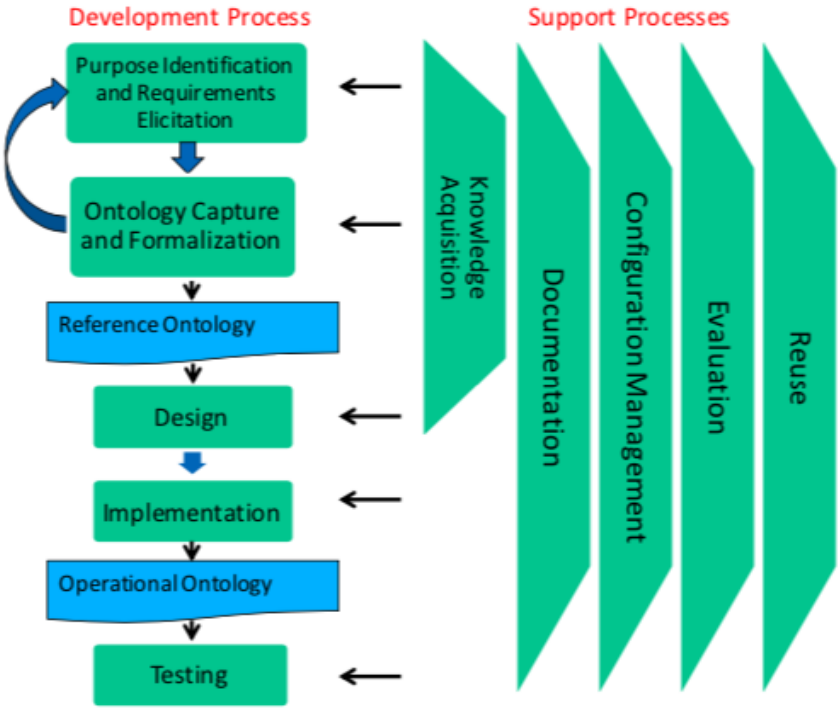

Figura 1 – Fases e processos de apoio do método SABiO (FALBO, 2014).

de ontologias, um método de engenharia de ontologias bem estabelecido. SABiO foi escolhido pois foca no desenvolvimento de ontologias de domínio e foi utilizado com sucesso no desenvolvimento de várias ontologias no domínio de Engenharia de Software (e.g., (BRINGUENTE; FALBO; GUIZZARDI, 2011; RUY et al., 2016)), sendo assim, seus objetivos vão ao encontro com os desta pesquisa.

SABiO é um método composto de cinco fases e cinco processo de apoio, como apresentado na Figura 1:

- Identificação do propósito e elicitação de requisitos: esta primeira fase do método busca identificar a finalidade da ontologia e suas intenções de uso. Em seguida deve-se realizar a elicitação de requisitos, que, assim como requisitos de software, podem ser divididos em requisitos funcionais e requisitos não-funcionais. Os requisitos funcionais devem ser descritos na forma de questões de competência que a ontologia deverá responder (GRÜNINGER; FOX, 1995), assim podemos definir o escopo da ontologia e obtermos uma forma de avaliação. Requisitos não-funcionais estão associados a características e qualidades não relacionadas ao conteúdo da ontologia;

- Captura e formalização da ontologia: Nesta fase, conceitos e relações relevantes para o domínio são identificados e organizados. SABiO sugere que esta análise seja realizada baseada em uma ontologia de fundamentação e indica a construção de um modelo gráfico, que se torna um instrumento muito importante para apoiar a comunicação com especialistas de domínio. Nesta etapa um modelo conceitual deve ser construído

e SABiO sugere o uso de OntoUML. O processo de captura de ontologia é guiado pelas questões de competência e é fortemente apoiado pelo processo de aquisição de conhecimento, que pode ser extraído de especialistas de domínio ou até de fontes de conhecimento consolidadas, como livros, padrões internacionais e modelos de referência. O resultado desta etapa é a ontologia de referência de domínio. A elicitação de requisitos e captura da ontologia de RASO são descritos no Capítulo 4;

- Design: uma vez produzida a ontologia de referência, os passos a partir desta etapa buscam a produção da versão operacional para ser usada por aplicações de computador. Na fase de design, a especificação conceitual da ontologia de referência é transformada em uma especificação de design. Diferente da ontologia de referência, a ontologia operacional não são focadas em adequação da representação, mas são projetadas de modo a garantir as propriedades computacionais desejáveis;

- Implementação: assim como no modelo tradicional da Engenharia de Software, a implementação de ontologias trabalha a codificação da ontologia para a linguagem operacional escolhida na fase anterior para que seja computacionalmente interpretada;

- Teste: nesta última etapa ocorre a verificação e validação de forma dinâmica do comportamento da ontologia operacional, através de um conjunto finito de testes, analisados sobre o comportamento esperado em relação às questões de competência. São realizados principalmente testes de caixa preta.

As etapas anteriores são acompanhadas por processos de suporte amplamente aceitos na Engenharia de Software, como aquisição do conhecimento, reutilização, documentação, gerência de configuração e avaliação. As duas primeiras fases do processo de desenvolvimento do SABiO buscam a produção de uma ontologia de domínio de referência, dado o interesse desta pesquisa, para a construção da RASO foram realizadas as duas primeiras fases do método SABiO.

SABiO reconhece explicitamente a importância do uso de ontologias de fundamentação no processo de desenvolvimento de ontologias para melhorar a qualidade, representatividade e formalidade da ontologia. Assim, a ontologia RASO, aqui produzida, foi fundamentada na *Unified Foundational Ontology* (UFO) (GUIZZARDI; FALBO; GUIZZARDI, 2008), uma ontologia de fundamentação bem estabelecida.

UFO foi desenvolvida baseada em várias teorias das áreas de Lógica Filosófica, Ontologias Formais, Linguística e Psicologia Cognitiva. UFO é utilizada para prover a semântica para ontologias de domínio através de um conjunto abrangente de categorias de entidades que buscam lidar com as especificidades do domínio alvo, permitindo um mapeamento sólido de entidades e relações existentes no mundo real. Como SABiO, UFO tem sido empregada com sucesso no desenvolvimento de diversas ontologias no

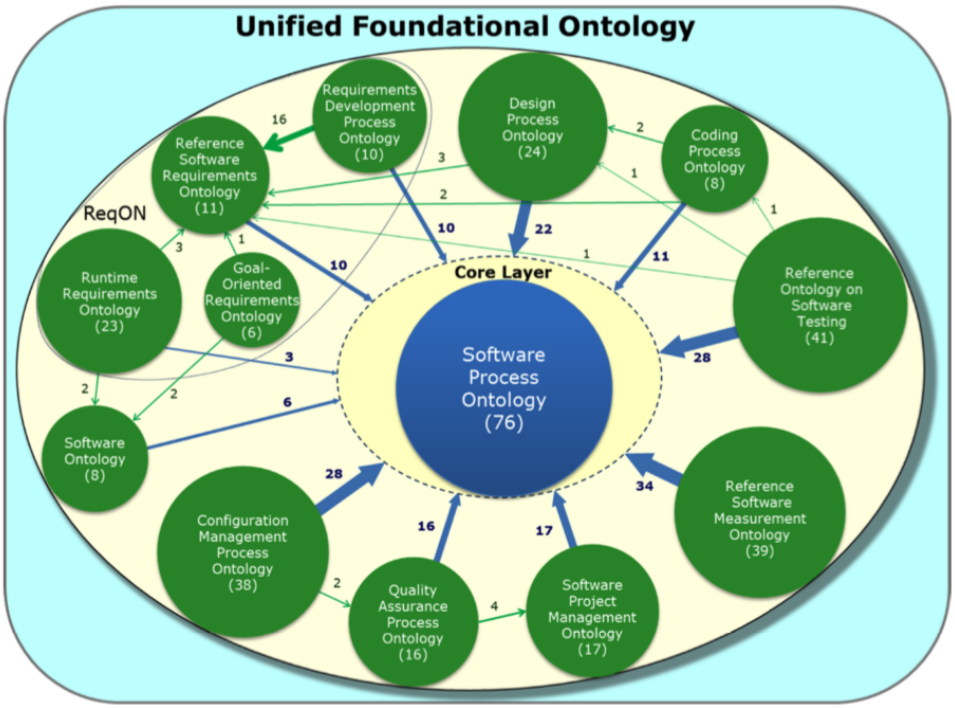

Figura 2 – Modelo de SEON e o relacionamento das ontologias (DUARTE et al., 2018).

domínio da Engenharia de Software (por exemplo, (BRINGUENTE; FALBO; GUIZZARDI, 2011), (RUY et al., 2016)), incluindo os que foram reutilizados/estendidos neste trabalho.

O SABiO também sugere que as ontologias existentes em domínios relacionados devem ser reutilizadas. No contexto da Engenharia de Software, devido ao vasto domínio da área, a representação deste em uma única ontologia se tornaria inviável, assim, sob-domínios são tratados de forma isolada e consequentemente ocorrem problemas para integração sobre o domínio principal. A *Software Engineering Ontology Network* (SEON) (RUY et al., 2016) é uma rede de ontologias de referência no contexto de Engenharia de Software, que busca tratar problemas relacionados a integração do conhecimento e oferece meios de facilitar a construção e integração de novas ontologias deste domínio à rede, mantendo uma semântica consistente entre os conceitos e relações que a compõem. A Figura 2 apresenta uma visão geral sobre a SEON,[1] fruto de atualização realizada por Duarte et al. (2018). Nota-se que a mesma é criada a luz de UFO e cada um dos círculos representa uma ontologia componente da rede. Ao centro se encontra a ontologia central das relações. As ontologias mais próximas da borda são as ontologias de domínio, já integradas à SEON. A proporção dos círculos representa a quantidade de elementos pertencentes a cada ontologia e a espessura das linhas de relação se faz pela quantidade de relações entre elas.

Assim, ontologias que fazem parte da SEON foram reutilizadas pela RASO, facilitando uma possível futura integração com a SEON. O conjunto de ontologias reutilizadas inclui:

[1] A especificação completa encontra-se em <https://nemo.inf.ufes.br/projects/seon/>.

- *Software Ontology* (SwO): uma ontologia sobre softwares e sua natureza, descrevendo sua constituição, como sistemas de software, programas e código;

- *Reference Software Requirements Ontology* (RSRO): uma ontologia de referência sobre requisitos de software, buscando a definição de requisito, trabalhando com distinção de requisitos funcionais e não funcionais, requisitos como artefatos, entre outros;

- *Runtime Requirements Ontology* (RRO) (DUARTE et al., 2018): uma ontologia sobre requisitos em tempo de execução que, neste trabalho, possui grande relevância;

- *Software Process Ontology* (SPO) (BRINGUENTE; FALBO; GUIZZARDI, 2011): uma ontologia sobre processos de software que se encontra como peça central em SEON, unificando as ontologias anteriormente citadas.

Decidiu-se reutilizar conceitos dessas ontologias pois seus domínios estão intrinsecamente relacionados ao domínio de sistemas adaptativos baseados em requisitos. Além disso, são todas baseadas em UFO, garantindo compatibilidade no nível fundamental. Entre essas ontologias, a RRO tem um papel proeminente, pois descreve o uso de artefatos de requisitos em tempo de execução, o que é fundamental para os sistemas adaptativos. Dada a proximidade da RRO, poderia até se dizer que RASO é uma extensão de RRO.

A Figura 3 mostra os conceitos de SEON reutilizados pela RASO, usando um diagrama de classes UML principalmente para visualização (para uma discussão aprofundada e uma caracterização mais formal, consulte (DUARTE et al., 2018)). Como mostrado, esses conceitos derivam da noção de Artifact de SPO, ou seja, um Object (no sentido de UFO) intencionalmente feito para servir um determinado propósito no contexto de um produto ou organização de software. Estamos particularmente interessados em três tipos de Artifact: Software Items — partes do software produzido durante o processo de software —, Documents — qualquer informação escrita ou pictórica relacionada ao desenvolvimento de software, geralmente apresentado em um formato pré-definido — e Information Items — qualquer informação relevante produzida durante o processo de software para uso humano.

Conforme definido em SwO (DUARTE et al., 2018), um Software System é um Software Item que *intends to implement* uma System Specification, que é o Document contendo um conjunto de requisitos para um sistema, definindo suas funções e características desejadas de forma abstrata, sem restringir seu comportamento. Software Systems são *constituted of* Programs. Um Program é um Software Item que visa produzir resultados através da execução em um computador, conforme prescrito pela Program Specification, um Document que descreve a estrutura e as funções de um Program. Finalmente, um Program é *constituted of* Code, um Software Item representando um conjunto de instruções de computador e definições de dados expressas em uma linguagem de programação. Um

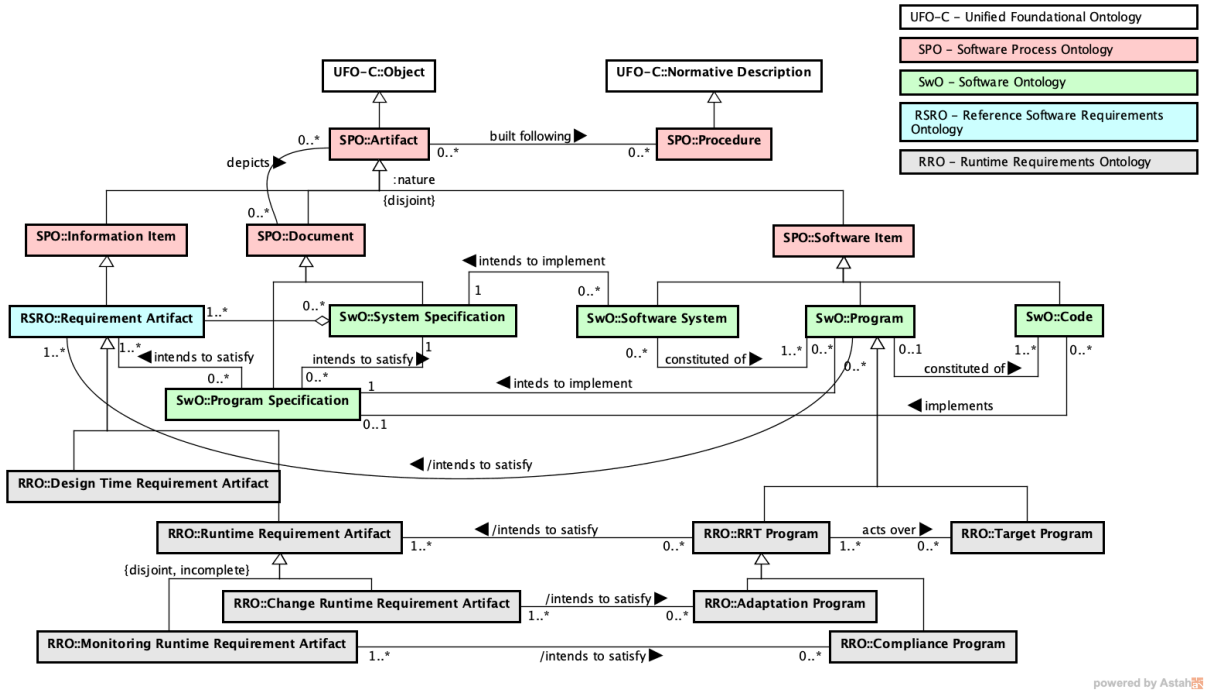

Figura 3 – Visão de SEON mostrando conceitos e relações utilizadas por RASO.

Program é *constituted by* Code, mas não é idêntico a Code. Code pode ser alterado (por exemplo, para corrigir um bug) sem alterar a identidade de seu Program, que está ancorado na propriedade essencial do programa: sua Program Specification pretendida, que o Code *implements*.

Dado nosso foco em abordagens de sistemas adaptativos baseados em requisitos, reutilizamos da RSRO o conceito de Requirement Artifact, um Information Item que descreve o requisito de um *stakeholder* (conceitos também presentes no RSRO, mas fora deste escopo), provavelmente como resultado de alguma atividade de documentação de requisitos no processo de Engenharia de Requisitos. No contexto de sistemas adaptativos, estamos particularmente interessados em um tipo especial de Requirement Artifact especificado em RRO, o Runtime Requirement Artifact (RRA). Ao contrário de Design Time Requirement Artifacts, RRAs são manipulados executando Programs, ou seja, em tempo de execução. Tais Programs são ditos RRT (Requirements at Runtime) Programs, que *intend to satisfy* um ou mais RRAs, *acting over* outros Programs em execução, referidos como Target Programs. RRO especifica dois tipos particulares de uso de requisitos em tempo de execução: Compliance Programs *intending to satisfy* Monitoring RRAs e Adaptation Programs *intending to satisfy* Change RRAs.

2.3 Trabalhos Relacionados

Qureshi, Jureta e Perini (2011) propôs uma nova versão da *Core Ontology for Requirement Engineering* (CORE) (JURETA; MYLOPOULOS; FAULKNER, 2008; JU-

RETA; MYLOPOULOS; FAULKNER, 2009), introduzindo dois novos conceitos (*context* e *resource*) e relações (*relegation* e *influence*) no topo da CORE (que diz respeito à Engenharia de Requisitos em geral) para representar adequadamente as possíveis mudanças que podem ocorrer nos requisitos, em tempo de execução. Os autores afirmam que combinando os novos elementos com os que são originalmente utilizados pela linguagem de modelagem de objetivos Techne (JURETA et al., 2010), eles são capazes de dar suporte à definição do problema de adaptação de requisitos de tempo de execução.

No entanto, a ontologia de Qureshi, Jureta e Perini (2011) não representa conceitos comuns às abordagens baseadas em requisitos para o desenvolvimento de sistemas adaptativos, como requisitos de monitoramento e de adaptação, por exemplo. Tal fato é confirmado pela análise das abordagens selecionadas no mapeamento sistemático da literatura, apresentado no Capítulo 3. Em vez disso, a ontologia inclui conceitos e relações que não estão fortemente relacionadas a esse domínio (ou seja, não foram comumente encontradas nas abordagens selecionadas), como *resource* e *relegation*.

Soares et al. (2016) propõem uma ontologia central para auxiliar no elicitação e especificação de requisitos para sistemas adaptativos, com base na ontologia de Qureshi, Jureta e Perini (2011) — que consideraram incompletas —, acrescentando conceitos extraídos das modelagem de dimensões para sistemas adaptativos (ANDERSSON et al., 2009). Além de herdar os supracitados problemas de (QURESHI; JURETA; PERINI, 2011), a ontologia não é apropriadamente baseada em uma ontologia de fundamentação, é apresentada apenas em sua forma operacional (em OWL) e inclui conceitos pertencentes à área de sistemas sensíveis ao contexto (alegando que eles subsumem sistemas adaptativos, uma afirmação que neste trabalho mostra-se questionável). Em nosso trabalho, propomos uma ontologia de referência, fundamentada em UFO e focada exclusivamente nos conceitos do domínio de sistemas adaptativos.

Reinhartz-Berger, Sturm e Wand (2013) propõem um modelo conceitual de comportamento de software baseado na ontologia de fundamentação de Bunge (1977), que pode ser usado para modelar o comportamento esperado de um sistema (seus requisitos) e comportamentos alternativos que o programa em execução pode executar, apoiando assim as decisões de auto-adaptação, comparando alternativas sobre: (1) como eles atendem aos requisitos; e (2) o esforço necessário para mudar comportamentos. Seu trabalho usa uma abordagem ontológica para modelar o comportamento do sistema, a fim de promover a auto-adaptação, ou seja, é uma abordagem baseada em requisitos para o desenvolvimento de sistemas adaptativos. O trabalho aqui apresentado fornece uma ontologia para o domínio de requisitos para sistemas adaptativos, isto é, descreve conceitos usados por diferentes abordagens (incluindo (REINHARTZ-BERGER; STURM; WAND, 2013), que poderiam ser instanciadas).

Finalmente, em relação à inclusão de um mapeamento sistemático da literatura (cf.

Capítulo 3) como parte do método utilizado neste trabalho, uma revisão sistemática da literatura neste domínio havia já sido conduzida por Yang et al. (2014), com o objetivo de investigar quais métodos de modelagem, atividades, atributos de qualidade, domínios de aplicação e tópicos de pesquisa foram estudados na área de Engenharia de Requisitos para sistemas adaptativos e quão bem esses estudos foram transmitidos. O mapeamento conduzido como parte deste trabalho, embora mais superficial do que uma revisão, apresenta algumas semelhanças com a revisão de Yang et al. (2014), como por exemplo o método de pesquisa (KITCHENHAM; CHARTERS, 2007). Foi considerada a possibilidade de usar o estudo como a base de conhecimento para construir a RASO em vez de executar um novo. No entanto, visto este ter sido conduzido há cinco anos, decidiu-se que um novo estudo seria necessário para não negligenciar os avanços recentes que foram alcançados nessa área de pesquisa. Além disso, atualizar sua revisão teria demandado muito mais esforço do que produzir um novo mapeamento.

O estudo de mapeamento sistemático aqui produzido mostrou que quase um quinto das abordagens baseadas em requisitos para o desenvolvimento de sistemas adaptativos faz uso de ontologias como parte de seus métodos propostos. Entre eles, CORE (JURETA; MYLOPOULOS; FAULKNER, 2008; JURETA; MYLOPOULOS; FAULKNER, 2009) é o trabalho mais citado. No entanto, como mencionado anteriormente, a CORE representa o domínio da Engenharia de Requisitos em geral, enquanto a RASO se concentra na Engenharia de Requisitos para sistemas adaptativos (com base em RSRO (DUARTE et al., 2018) pelas razões discutidas na Seção 2.2). Isso mostra um interesse crescente no uso de ontologias neste domínio e uma lacuna de ontologias especificamente desenvolvidas para sistemas adaptativos, que a RASO pretende preencher.

3 Mapeamento Sistemático: Requisitos para Sistemas Adaptativos

Uma revisão sistemática da literatura é um meio de identificar, avaliar e interpretar as publicações relevantes a uma área de pesquisa em particular. Estudos que compõem o *corpus documental* de revisão sistemática são considerados estudos primários e uma revisão sistemática é uma forma de estudo secundário. Considerando que durante a análise inicial seja descoberto que o tópico de pesquisa é muito amplo ou que exista pouco conteúdo específico sobre o tema (situação aplicada a esta pesquisa em questão), torna-se mais apropriada a realização de um mapeamento sistemático (KITCHENHAM; CHARTERS, 2007).

Um mapeamento sistemático permite que evidências de um domínio possam ser visualizadas a partir de um alto nível de granularidade. São projetados para estabelecer se existem evidências de pesquisa sobre um tópico e fornecer uma indicação de quantidade de evidências. Como resultado é possível identificar áreas que carecem da realização de uma revisão sistemática ou áreas onde um estudo primário seria mais apropriado (KITCHENHAM; CHARTERS, 2007).

Buscando identificar e compreender os conceitos envolvidos em abordagens que utilizam modelos de requisitos para o desenvolvimento de sistemas adaptativos, foi realizado um mapeamento sistemático da literatura (KITCHENHAM; CHARTERS, 2007; PETERSEN; VAKKALANKA; KUZNIARZ, 2015) e analisado o estado-da-arte neste campo em particular, a fim de desenvolver a ontologia RASO. Para a realização deste mapeamento sistemático, foram seguidas as diretrizes apresentadas por Kitchenham e Charters (2007).

Assim, foram estabelecidos dois objetivos, sendo o principal levantar conceitos utilizados em modelos de requisitos para sistemas adaptativos para construção de uma ontologia, e tendo como objetivo secundário levantar informações sobre o estado-da-arte.

Definido o objetivo, houve a definição das questões de pesquisa (QP). Esta é a parte mais importante do mapeamento sistemático e conduz toda a metodologia de revisão sistemática, pois, a busca deverá identificar estudos primários relevantes às QPs, a extração de dados deve prover material suficiente a responder às QPs e análise dos dados devem sintetizar os dados de forma a responder as QPs (KITCHENHAM; CHARTERS, 2007). As questões definidas a partir do objetivo estão representadas na Tabela 1.

Em um trabalho anterior, Duarte et al. (2018) realizou um estudo sobre a área de Requisitos em Tempo de Execução com o objetivo de desenvolver a Runtime Requirements

Tabela 1 – Questões de pesquisa definidas para o Mapeamento Sistematico.

ID	**Questão de Pesquisa**	**Motivação**
QP1	Quais são os indicadores sobre sistemas adaptativos que aparecem nos modelos de requisitos?	Identificar e mapear conceitos na área a fim de construir conhecimento compartilhado necessário para o desenvolvimento da ontologia.
QP2	Que tipo de formalismo é usado para apoiar os modelos?	Identificar a maturidade semântica dos modelos encontrados. Descobrir trabalhos relacionados à ontologia que será proposta.
QP3	Que tipo de modelos foram usados?	Identificar tipos de modelos e sua popularidade nessa comunidade.
QP4	Que tipo de pesquisas foram realizadas?	Avaliar o atual estágio de maturidade da área de requisitos para sistemas adaptativos.
QP5	Quando e onde os documentos foram publicados?	Identificar mais informações e grupos de pesquisa.

Ontology (RRO, cf. seções 2.2 e Capítulo 4). Durante este processo, foi realizado um mapeamento sistemático, usado como pilar para a construção da ontologia. Tal pesquisa apresenta um escopo mais amplo que a pesquisa aqui apresentada, buscando qualquer uso de modelos de requisitos em tempo de execução. Esse mapeamento classificava publicações por propósito de uso de requisitos em tempo de execução, separando-os em duas categorias principais: *Monitor Requirements*, que propõem verificar se os requisitos definidos no tempo de design estão sendo atingidos em tempo de execução e *Change Requirements* em que os requisitos são usados não apenas como diretrizes para o monitoramento, mas também como regras sobre como o sistema deve se adaptar para continuar satisfazendo seus requisitos.

Baseamos nosso protocolo de mapeamento nesses resultados anteriores considerando as publicações que apresentam algum modelo de adaptação, essas, categorizadas como *Change Requirements* (86 artigos mapeados), utilizadas então como documentos de controle, ou seja, publicações relevantes ao nosso contexto de pesquisa e que nossa *string* de busca deve retornar. Estes artigos são apresentados no Apêndice A.

Também consideramos o mesmo conjunto de fontes de publicação. Estes bancos de artigos apresentam mecanismos de busca que facilitam a utilização das *strings* de buscas criadas e favorecem a organização e obtenção dos resultados. A *string* de busca é a união de diversos termos que representam o tema a ser pesquisado e tem por objetivo atingir o maior número possível de artigos relevantes a serem retornados das bases de pesquisa. A *string* de busca modifica sua sintaxe a fim de atender as necessidades de entrada de cada banco, mas permanece com a mesma semântica. Os bancos aqui utilizados foram:

- ACM Digital Library (http://portal.acm.org);
- Engineering Village (http://www.engineeringvillage.com);

Tabela 2 – Terms used in the search string.

Id	Group of terms
ρ	"requirements model" OR "requirements engineering" OR "requirements analysis" OR "requirements reasoning" OR "goal model" OR "gore" OR "goal analysis" OR "goal reasoning"
α	"adaptive system" OR "adaptive systems" OR "self-tuning" OR "runtime adaptation" OR "self-adaptive" OR "self-adaptation" OR "self-optimization" OR "self-adaptivity" OR "software adaptation" OR "self-configuration" OR "self-healing" OR "self-protection"

- IEEE Xplore (http://ieeexplore.ieee.org);
- Science Direct – Elsevier (http://www.elsevier.com);
- Scopus (http://www.scopus.com);

Com base nos requisitos da RASO, a busca foi elaborada em um processo iterativo, no qual os termos que se referiam ao uso de modelos de requisitos (ρ) e à aplicação no desenvolvimento de sistemas adaptativos (α) foram sendo refinados a partir da verificação da ocorrência dos artigos de controle nos resultados. Os meta-dados (título, resumo, palavras-chave) dos artigos de controle ausentes eram analisados e novos termos incluídos na *string* de busca, até que todos os artigos de controle estivessem presentes no resultado da busca. A *string* de busca final é o conjunção dos dois grupos de termos mostrados na Tabela 2, i.e., $\rho \wedge \alpha$.

Uma vez de posse de uma *string* de busca satisfatória, prosseguiu-se com o protocolo de mapeamento sistemático. A Figura 4 apresenta uma visão geral deste processo de busca e filtragem. A partir do primeiro resultado, obtivemos como retorno geral 784 artigos, sendo 111 no IEEE Xplore, 21 no Science Direct, 287 no Engineering Villag, 40 no ACM Digital Library e 325 no Scopus. Os resultados foram unificados e removidos os artigos duplicados (mesmo artigo retornado em diferentes bases), resultando em 385 artigos no total. Os artigos resultantes foram analisados (em título, resumo e palavras-chave) dentro do especificado no protocolo e eliminados pelo não cumprimento do critério de inclusão (CI1) ou da satisfação de algum critério de exclusão (CE1–CE6), apresentados na Tabela 3.

Após aplicar os critérios de exclusão, 161 artigos foram excluídos do mapeamento. Em seguida aplicamos o critério de inclusão, considerando se o artigo apresentasse algum tipo de modelo de requisitos para sistemas adaptativos, o que excluiu outros 57 artigos. Os critérios de seleção foram aplicados em duas fases — primeiro considerando apenas os metadados (título, resumo e palavras-chave), depois lendo o artigo completo — resultando em 224 publicações selecionadas.

A Figura 5 apresenta a distribuição de publicações ao longo dos anos. Nota-se que embora exista registro de publicações desde o ano de 1991, apenas a partir de 2001

Tabela 3 – Critérios de Inclusão e Exclusão de artigos do Mapeamento Sistemático.

ID	**Critério**
CI1	O artigo apresenta algum modelo de requisitos para sistemas adaptativos, não necessariamente diagramático.
CE1	Não possui resumo.
CE2	É apenas um resumo.
CE3	Não está escrito em inglês.
CE4	É uma cópia ou uma versão mais antiga de outra publicação que já foi considerada.
CE5	Não é um estudo primário.
CE6	Não é possível obter a versão completa da publicação.

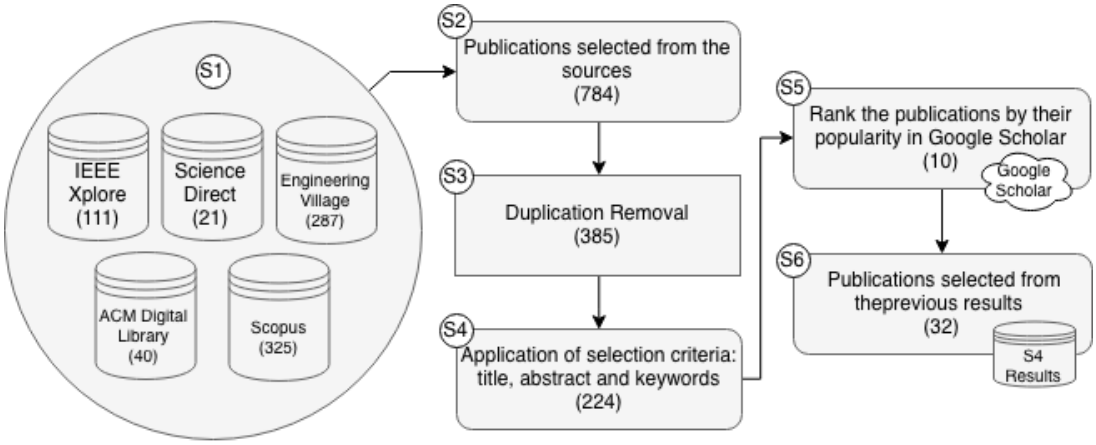

Figura 4 – Visão geral do protocolo de mapeamento sistemático para a RASO.

verificamos a recorrência do tema e (de modo geral) o crescente número de publicações da área, com grande destaque nos anos de 2012 e 2015.

Também nota-se, a partir da distribuição de publicações por continente (representada na Figura 6) a origem europeia da maior parte das publicações. O continente americano está segmentado em América do Norte e América Latina de modo a evidenciar o nível de contribuição dos mesmos. Ao analisar a distribuição de publicações por países (representada na Figura 7), destacam-se Itália (34 publicações), Reino Unido (33 publicações), China (23 publicações), Estados Unidos (20 publicações) e Brasil (16 publicações), como os países com maior número de contribuições.

Devido ao grande número de publicações selecionadas, propõe-se um novo filtro a fim de selecionar um conjunto menor de publicações que seriam estudadas como parte da elicitação de requisitos e conceitos ontológicos para a RASO. Dado que uma ontologia deveria fornecer uma conceituação que é compartilhada por uma determinada comunidade, classificamos as publicações por sua popularidade (citações), como segue:

1. O número total de citações foi extraído do Google Scholar pesquisando pelo título

Figura 5 – Distribuição de publicações ao longo dos anos.

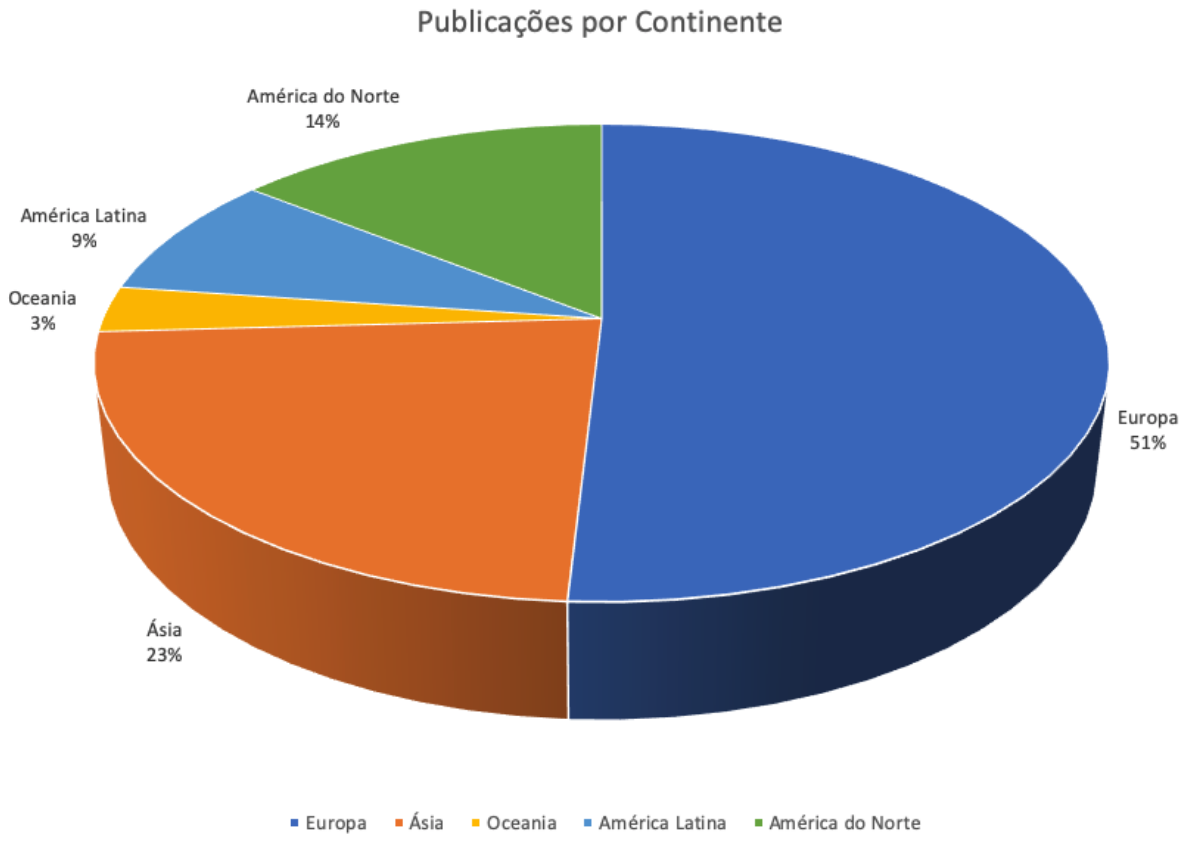

Figura 6 – Distribuição de publicações por continente.

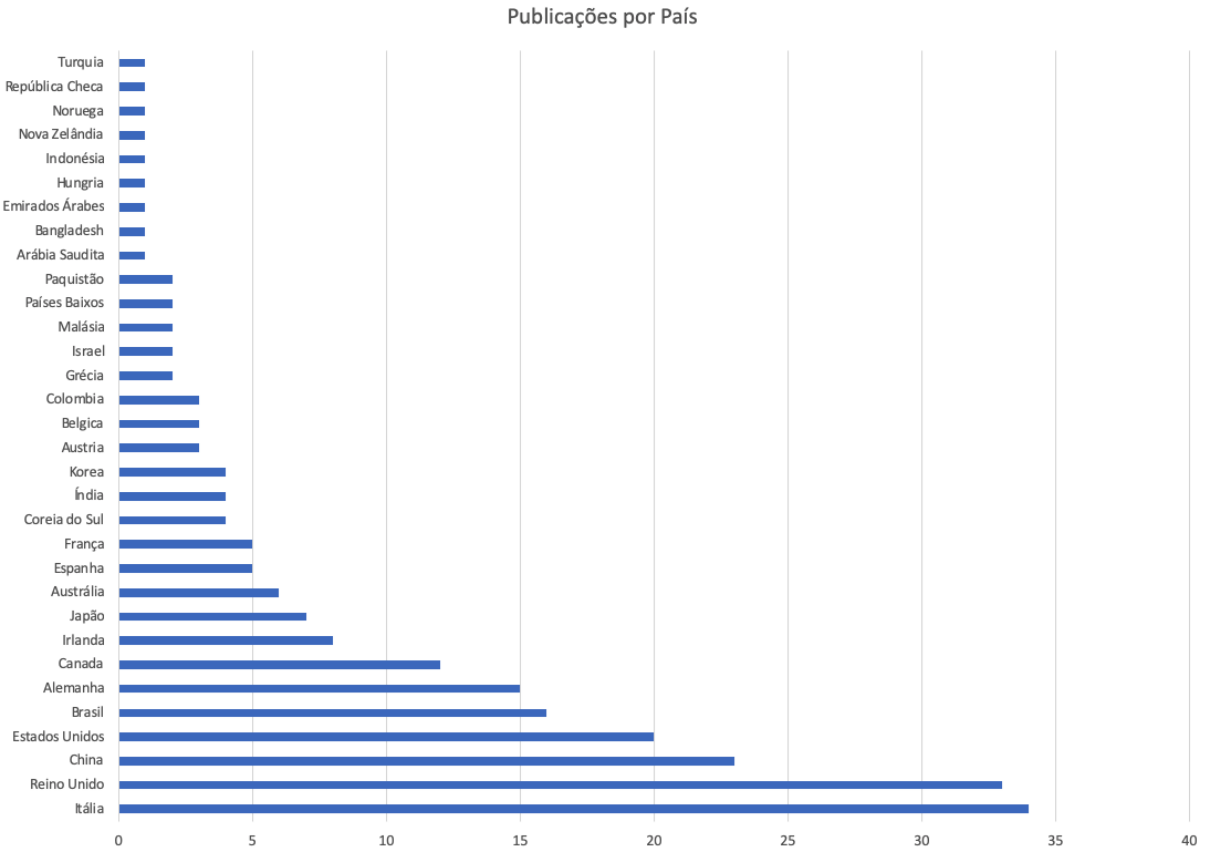

Figura 7 – Distribuição de publicações por país.

da publicação;

2. A idade de cada publicação foi calculada subtraindo o ano de publicação do ano atual;

3. O número de citações foi dividido pela idade, normalizando-se para citações por ano;

4. As publicações foram classificadas das mais citadas para as menos citadas.

Foram selecionados os artigos mais citados, atendendo a um ponto de corte, sendo que o último selecionado deveria possuir mais de 50% do coeficiente do primeiro. Totalizando assim 10 artigos selecionados como principais. A lista de artigos selecionados é exibida pela Tabela 4, indicando que 8 diferentes abordagens baseadas em requisitos para o desenvolvimento de sistemas adaptativos foram representadas pelos 10 principais artigos.

Por fim, retornou-se à lista completa de (224) publicações em busca de outros trabalhos abordando cada uma dessas abordagens selecionadas, a fim de fornecer um conteúdo mais completo para sua compreensão. A pesquisa foi realizada usando o nome dos autores dos 10 artigos principais e retornou 22 novas publicações para serem analisadas, a saber:

Tabela 4 – Os 10 artigos mais citados do mapeamento e suas respectivas abordagens.

Paper	**Coefficient**	**Approach**
Fuzzy goals for requirements-driven adaptation (BARESI; PASQUALE; SPOLETINI, 2010)	24,62	FLAGS
Requirements-aware systems: A research agenda for RE for self-adaptive systems (SAWYER et al., 2010)	23,00	RELAX
A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty (CHENG et al., 2009b)	22,33	RELAX
Adaptive socio-technical systems: A requirements-based approach (DALPIAZ; GIORGINI; MYLOPOULOS, 2012)	20,00	Adaptive STS
RELAX: A language to address uncertainty in self-adaptive systems requirement (WHITTLE et al., 2010)	19,87	RELAX
Awareness requirements for adaptive systems (SOUZA et al., 2011)	18,00	Zanshin
Ambient Assisted Living system for in-home monitoring of healthy independent elders (BOTIA; VILLA; PALMA, 2012)	17,66	Unnamed approach applied to Necesity
Engineering requirements for adaptive systems (MORANDINI et al., 2017)	14,00	Tropos4AS
QoS-aware middleware for ubiquitous and heterogeneous environments (NAHRSTEDT et al., 2001)	13,52	QoS-aware middleware
Goal-based modeling of Dynamically Adaptive System requirements (GOLDSBY et al., 2008)	12,90	LoREM

- 8 novos artigos sobre a abordagem Zanshin: (SOUZA et al., 2013; SOUZA; LAPOUCHNIAN; MYLOPOULOS, 2012a; ANGELOPOULOS; SOUZA; MYLOPOULOS, 2014; TALLABACI; SOUZA, 2013; SOUZA; LAPOUCHNIAN; MYLOPOULOS, 2012b; SOUZA; MYLOPOULOS, 2011a; SOUZA; MYLOPOULOS, 2015; INGOLFO; SOUZA, 2013);

- 8 novos artigos sobre a abordagem RELAX: (RAMIREZ; CHENG, 2011; AHMAD; BELLOIR; BRUEL, 2015; RAMIREZ et al., 2012a; RAMIREZ et al., 2012b; FREDERICKS; DEVRIES; CHENG, 2014; AHMAD et al., 2012; CHENG; RAMIREZ; MCKINLEY, 2013; FREDERICKS; CHENG, 2015);

- 3 novos artigos sobre a abordagem Tropos4AS: (MORANDINI; PENSERINI; PERINI, 2009; MORANDINI; PERINI; MARCHETTO, 2011; MORANDINI; MARCHETTO; PERINI, 2011);

- 3 novos artigos sobre a abordagem FLAGS: (PASQUALE; SPOLETINI, 2011; BARESI; PASQUALE, 2010; PASQUALE; BARESI; NUSEIBEH, 2011).

O conjunto final de 32 publicações foi usado não apenas para construir a RASO (cf. Capítulo 4), mas também para validá-la (Capítulo 5). A seguir, resumimos brevemente cada uma das abordagens selecionadas.

3.1 FLAGS

Baresi, Pasquale e Spoletini (2010) afirmam que a auto-adaptação isoladamente já é um requisito, mas também impacta em outros requisitos do sistema e ambos devem ser reconhecidos durante o processo de elicitação de requisitos e representados de modo coerente e homogêneo. Assim sendo, a abordagem FLAGS (BARESI; PASQUALE; SPOLETINI, 2010) se apresenta como inovadora, com base em KAOS (LAMSWEERDE et al., 1991), mas acrescido de adaptive goals, para realização de contramedidas de adaptação. FLAGS também propõe *crisp goals* (especificadas em Lógica Temporal Linear) cuja satisfação pode ser facilmente avaliada (Booleanos), e *fuzzy goals* que são especificadas usando *fuzzy constraints*, principalmente associadas a requisitos não-funcionais. A principal diferença entre *crisp goals* e *fuzzy goals* é que o primeiro são requisitos firmes, enquanto os últimos são mais flexíveis (BARESI; PASQUALE; SPOLETINI, 2010).

Para fornecer semântica para *fuzzy goals*, o FLAGS inclui operadores relacionais e operadores temporais *fuzzy*. Sempre que uma *fuzzy membership function* é introduzida em FLAGS, sua forma deve ser definida considerando as preferências das partes interessadas. Isso especifica exatamente quais valores são considerados próximos do valor desejado. Além disso, em FLAGS, as *adaptive goals* definem contramedidas a serem executadas quando as metas não são satisfeitas, usando as regras de Evento-Condição-Ação. A abordagem permite a definição de *adaptive goals* que executam um conjunto de *adaptation actions* que podem mudar o modelo de objetivos do sistema de diferentes maneiras — adicionar / remover / modificar objetivos ou agentes, flexibilizar um objetivo, etc. — e em diferentes níveis — de forma transitória ou permanente (BARESI; PASQUALE; SPOLETINI, 2010).

A Figura 8 apresenta um modelo de objetivos KAOS para um sistema de lavanderia. O objetivo geral do sistema é a lavagem de roupas (G1) que por sua vez é dividido em sub-objetivos: configurar a máquina de lavar (G1.1), completar o ciclo de lavagens(G1.4), consumir menor quantidade de energia (G1.2) e manter baixo número de roupas a serem lavadas (G1.3). A Figura 9 expõe os *adaptive goals* do sistema de lavanderia. Se o objetivo G1.4.1 for violado porque a máquina de lavar desligou repentinamente durante um ciclo de lavagem, deve-se ligar a máquina novamente e restaurar o sistema em um estado onde o tambor já esteja preenchido, o sabão deve ser adicionado e o programa ajustado novamente. Esse é o objetivo que determina como o sistema deve se adaptar aplicando uma contramedida adequada (adaptive goal). Cada contramedida está associada a um evento que a desencadeia uma execução (objetivo G1.4 violado), uma condição para a sua ativação

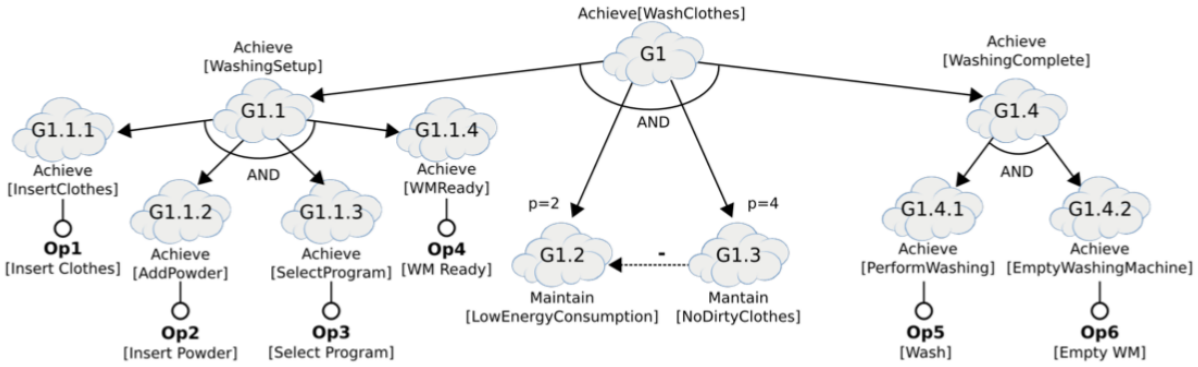

Figura 8 – Modelo de objetivos KAOS para um sistema de lavanderia (BARESI; PASQUALE; SPOLETINI, 2010).

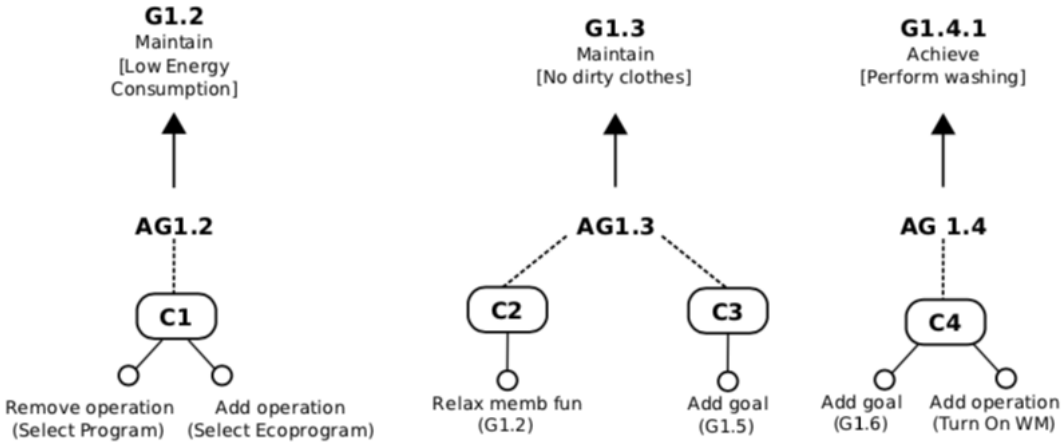

Figura 9 – Modelo de adaptive goals para o sistema de lavanderia (BARESI; PASQUALE; SPOLETINI, 2010).

real (a máquina de lavar se desligar), um objetivo a ser alcançado (ligar a lavadora de roupas) e uma sequência de ações básicas a serem executadas no modelo de objetivos e no sistema subjacente, para atingir o objetivo (BARESI; PASQUALE; SPOLETINI, 2010).

O *adaptive goal* AG1.4.1 está associado ao objetivo G1.4.1 e é operacionalizado por meio da contramedida C4, que é ativada quando a máquina de lavar desliga durante um ciclo de lavagem. O objetivo desta contramedida é restaurar a satisfação do objetivo G1.4.1, ligando a máquina de lavar roupa e restaurando-a em um estado no qual o tambor está cheio, mas o pó deve ser adicionado novamente e o programa deve ser selecionado, esta contramedida adiciona assim o objetivo G1.6 (BARESI; PASQUALE; SPOLETINI, 2010) .

3.2 RELAX

A linguagem RELAX (SAWYER et al., 2010; WHITTLE et al., 2010) visa capturar a incerteza declarativamente com operadores modais, temporais, ordinais e fatores de

incerteza fornecidos pela linguagem, capturando a incerteza na forma como os requisitos podem ser cumpridos, principalmente devido a fatores ambientais. Ao contrário das abordagens orientadas por objetivos, ele pressupõe que as especificações de requisitos em linguagem natural estruturada, contendo *SHALL statements* (instruções que seguem o modelo "o sistema deve ...") que especificam o que o sistema deve fazer, estejam disponíveis antes de sua conversão para as especificações RELAX. Os operadores modais disponíveis, SHALL e MAY...OR, especificam, respectivamente, que os requisitos devem ser mantidos ou que existem requisitos alternativos (variabilidade).

No RELAX, os pontos de flexibilidade/incerteza são especificados declarativamente, permitindo *designs* baseados em regras, planejamento, etc., bem como suportando adaptações imprevistas. Alguns requisitos são considerados invariáveis — eles precisam ser satisfeitos, não importa a circunstância. Outros requisitos tornam-se mais flexíveis para manter a sua satisfação usando os operadores RELAX do tipo AS POSSIBLE (por exemplo, AS EARLY AS POSSIBLE, AS CLOSE AS POSSIBLE, etc.). Por causa disso, o RELAX precisa de uma lógica com incerteza embutida para capturar sua semântica. Os autores escolheram a *Fuzzy Branching Temporal Logic* para esse propósito. Baseia-se na ideia de conjuntos difusos, o que permite funções graduais de associação. Operadores temporais, como EVENTUALLY e UNTIL, permitem um componente temporal nas especificações de requisitos em RELAX.

Cheng et al. (2009b) integram o LoREM (Seção 3.8) e o RELAX, adicionando à mistura uma abordagem para explorar sistematicamente a incerteza do ambiente para o qual o sistema adaptativo será implantado usando modelagem de ameaças em KAOS. Quando uma ameaça de objetivo é identificada, existem três possíveis estratégias de mitigação que podem ser aplicadas: (a) adicionar sub-objetivos para lidar com a condição da ameaça; (b) usar o RELAX para adicionar flexibilidade à definição do objetivo; (c) criar novas metas de alto nível que captem o objetivo de corrigir a falha. A última estratégia funciona como um ciclo de realimentação (*feedback loop*) que adapta o sistema sempre que o objetivo falhar em tempo de execução.

3.3 Adaptive STS

Um sistema sociotécnico (*socio-technical system* ou STS) consiste em uma interação de seres humanos, organizações e sistemas tecnológicos. Assim, estes sistemas são pouco controláveis e seu contexto pode mudar inesperadamente, atingindo seus atores e impactando no alcance de seus objetivos. Buscando uma solução, Dalpiaz, Giorgini e Mylopoulos (2012) propõem uma arquitetura para STSs, que possibilita a capacidade de se auto reconfigurar, buscando uma melhor configuração. A arquitetura realiza um loop MDRC (*monitor-diagnose-reconcile-compensate*), monitorando o comportamento dos

atores e mudanças de contexto, reconhecendo falhas e baixo desempenho (comparando com o comportamento esperado para o alcance de seus requisitos) e, em caso de falhas, busca meios de resolução e promove a compensação a fim de alcançar o comportamento desejado. Se tratando de um sistema sociotécnico, as ações humanas são consideradas e a autonomia dos participantes não podem ser controladas (DALPIAZ; GIORGINI; MYLOPOULOS, 2012).

A proposta de arquitetura foi construída orientada a requisitos, utilizando meta-modelos estendidos de Tropos (BRESCIANI et al., 2004) para diagnóstico de falhas e identificação de alternativas para cumprimento dos requisitos. O meta-modelo de Tropos foi acrescido de condições de acionamento e completude, especificação de planos e tempo de espera, permitindo assim o monitoramento e a criação de regras de compensação, base da adaptação. Como sua criação é voltada a STSs, alguns itens em particular se destacam na abordagem: a arquitetura permite não apenas a compreensão do sistema, mas também dos atores humanos; o mecanismo de diagnóstico compreende violação de dependências entre atores humanos; mecanismos de planejamento compreende tanto planos de execução como dependência entre atores; e os atores são autônomos, assim podem acatar ou recusar um sugestão proposta. O diagnóstico é realizado baseado nas intenções dos atores e em dependências sociais, sempre que o objetivo de um ator é acionado, o mesmo deve responder com uma intenção sobre este objetivo, seja executando planos ou interagindo com outros atores, mantendo assim, uma rede de dependências ao longo da execução do sistema (DALPIAZ; GIORGINI; MYLOPOULOS, 2012).

A Figura 10 mostra um exemplo, no qual o objetivo `g1` (*g1: Pre breakfast morning routine*) inclui um link de decomposição em `g10` (*g10 : Take medicine*), e `g10` é necessário para atingir `g1` somente se o contexto `c2` for válido (*c2: the patient suffers from a chronic disease*). O objetivo `g11` (*g1: Regulate glucose*) é uma alternativa válida para atingir a meta `g10` decomposta em alternativas (*OR*) somente se o contexto `c5` se mantiver (*c5:the patient suffers from diabetes*). Seguindo este modelo, uma regra de ativação é composta por um evento acionador e uma pré-condição. O objetivo é ativado quando o evento acionador acontece, se a condição prévia é válida. Este exemplo inclui ainda regras de ativação para o objetivo principal `g1`, ativado quando o alarme toca (*activation rule g1:alarm plays sound*), enquanto `g18` (*g18: Call helper*) é acionado quando o alarme toca, desde que o paciente esteja sozinho em casa e não seja autônomo (*activation rule g18: alarm play sound, if contexts c1 and c4 hold*; *c1:patient is alone at home*; *c4:patient is not autonomous*).

Objetivos declarativos são atingidos somente se sua condição de realização for cumprida e as condições de realização são expressas como estados no modelo de contexto. Neste exemplo, uma possível condição de realização para `g6` (*g6: Routinecheck*) é que um relatório médico eletrônico é enviado ao centro médico. Os objetivos declarativos lidam com a incerteza, pois permitem que os atores atinjam metas por meio de formas

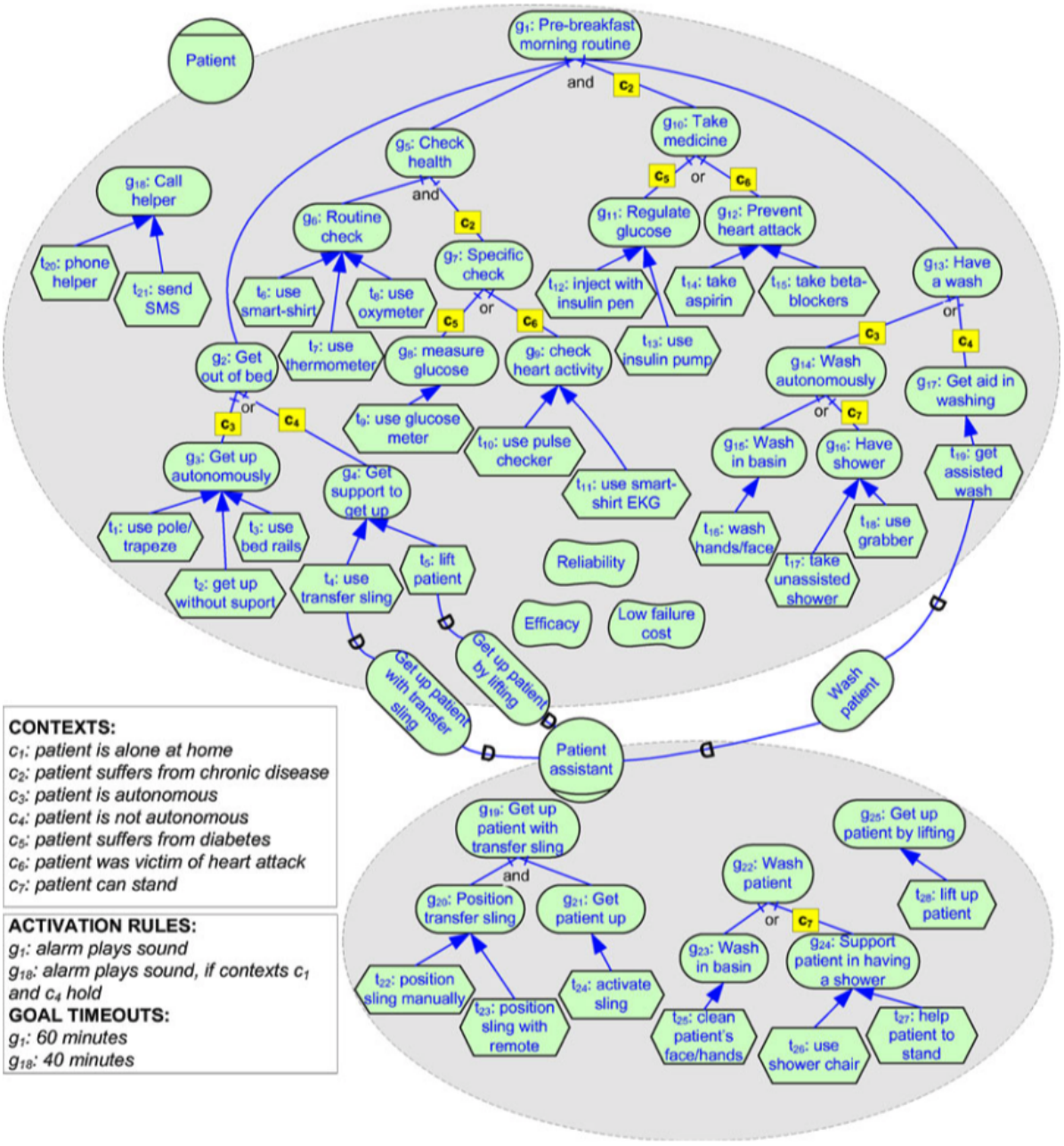

Figura 10 – Modelo contextual de Adaptive STS descrevendo um cenário de cuidados médicos (DALPIAZ; GIORGINI; MYLOPOULOS, 2012).

imprevistas. Referente ao tempo de espera, os limites de tempo estão associados a objetivos principais para definir o tempo máximo que um agente pode utilizar para atingir um objetivo. Neste exemplo, a meta `g1` deve ser alcançada dentro de 60 min (*goal timeout g1: 60 minutes*), enquanto a meta `g18` dentro de 40 min (*goal timeout g18: 40 minutes*) (DALPIAZ; GIORGINI; MYLOPOULOS, 2012).

3.4 Zanshin

Zanshin (SOUZA et al., 2011; SOUZA, 2012) é um *framework* baseado em Engenharia de Requisitos (ER) criado para o desenvolvimento de sistemas adaptativos. Sua principal ideia é fazer com que elementos do ciclo de realimentação (*feedback loop*) responsáveis por fornecer adaptabilidade sejam representados explicitamente nos modelos de requisitos. Para fazer isso, o Zanshin representa os requisitos do sistema usando os conceitos da *Core Ontology for Requirements Engineering* (CORE) (JURETA; MYLOPOULOS; FAULKNER, 2008; JURETA; MYLOPOULOS; FAULKNER, 2009), mas com uma sintaxe similar a i*.

Seus modelos são acrescidos com novos elementos, chamados *Awareness Requirements* (*AwReqs*) — requisitos que se referem ao estado de outros requisitos de um sistema de software em tempo de execução, ou seja, responsáveis por representar as partes do sistema que as partes interessadas querem que sejam capaz de adaptar — e *Evolution Requirements* (*EvoReqs*) — requisitos que descrevem como outros requisitos devem se adaptar/evoluir em resposta a uma falha do *AwReq*, ou seja, eles agem diretamente sobre os requisitos do sistema por meio de um conjunto de estratégias de adaptação. Quando combinados com construtos clássicos de GORE (*goals*, *softgoals*, *tasks*, *AND-OR refinements* e assim por diante), esses novos elementos são capazes de representar estratégias de monitoramento e adaptação durante a execução de um sistema de software.

A Figura 11 apresenta o funcionamento do *framework* Zanshin. Inicialmente o sistema alvo (*Target System*), base a qual será acrescida a funcionalidade de adaptação, é preparada de forma a prover um log indicando quando instancias de um requisito sofrerem mudanças. A partir deste log, juntamente com a especificação dos requisitos, o componente responsável pelo monitonamento (*Monitor*) realiza a análise sobre os status dos *AwReqs* e a satisfação dos mesmos, consequentemente, as mudanças de status acionam o componente de adaptação (*Adapt*), que por sua vez, decide quais as adaptações a serem executadas. A saída do componente de adaptação é uma lista de operações de *EvoReqs* enviadas ao sistema alvo. Assim, são realizadas ações com base nas instruções dadas pelas operações dos *EvoReqs*, representadas pela *Evolution API* implementada no sistema alvo (SOUZA; MYLOPOULOS, 2013).

A Figura 12 representa o modelo de requisitos de um sistema Agendador de Reunião criado como uma prova de conceito para o *framework* Zanshin e usado no Capítulo 4 para ilustrar os conceitos de RASO. Os *AwReqs* aparecem no modelo de requisitos do Agendador de Reuniões como pequenos círculos em negrito com setas, apontando para os elementos do sistema que precisam ter seus estados monitorados. Os *EvoReqs* não são representados graficamente no modelo de requisitos, no entanto, eles são implementados como um artefato de código que será executado pelo Zanshin quando um *AwReq* falhar. Por exemplo, se, por algum motivo, uma reunião não puder ser agendada adequadamente

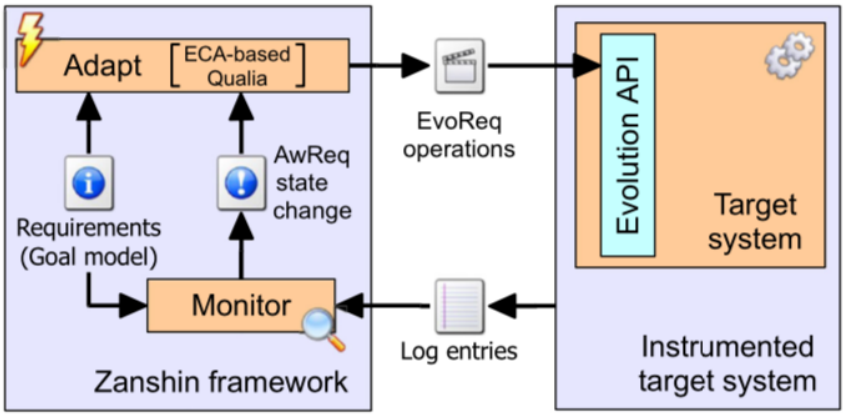

Figura 11 – Visão geral do *framework* Zanshin em ação. (SOUZA, 2012)

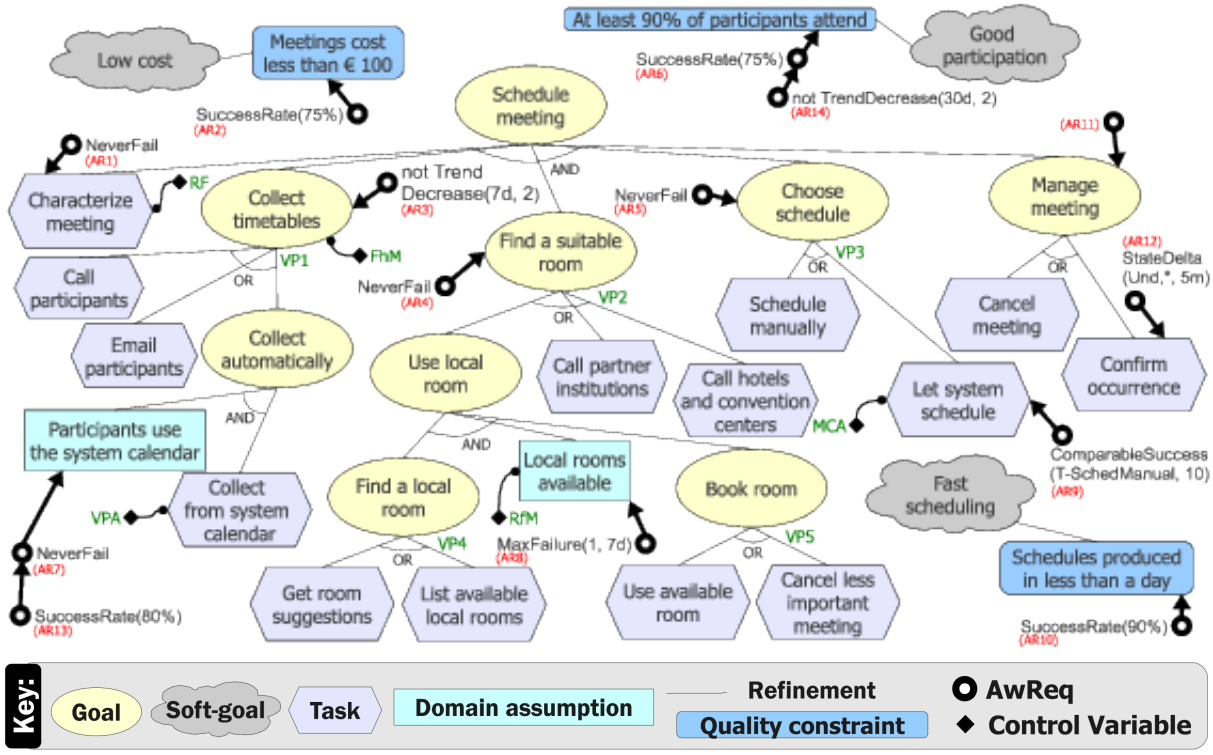

Figura 12 – Modelo de Requisitos do Sistema Agendador de Reuniões, criado com a abordagem Zanshin (SOUZA, 2012)

no sistema, AR1 (NeverFail, no canto superior esquerdo do modelo) acionará o *EvoReq* Retry Characterize Meeting, que fará com que o sistema execute uma reversão, aguarde 5 segundos e tente agendar a reunião novamente.

3.5 A abordagem aplicada ao Necesity

Necesity (BOTIA; VILLA; PALMA, 2012) é um sistema Ambiente de Vivência Assistido (*Ambient Assisted Living*) que foi projetado para monitorar os idosos que vivem sozinhos e querem continuar vivendo de forma independente. Um sistema capaz de detectar

quedas ou problemas semelhantes por meio de redes de sensores com dados distribuídos, a fim de obter dados sobre a atividade domiciliar do idoso. Técnicas Sensíveis ao Contexto processam os dados do sensor, fornecendo imagens do contexto do idoso. Uma das principais contribuições está relacionada com a capacidade do sistema de adaptar seu comportamento ao do idoso monitorado com base em um conjunto de regras de decisão.

Em Necesity, a adaptabilidade leva em conta os padrões comportamentais dos monitorados, utilizando técnicas de análise de dados para permitir que o sistema ajuste seu desempenho, assim, o sistema pressupõe que, se um houve evento em um passado recente e o mesmo evento ocorrer novamente, este é um comportamento normal. Por exemplo: este sistema está apto a reconhecer quedas a partir das especificações registradas e executar medidas, mas, considerando que o monitorado realiza exercícios se agachando (ou sentando) no chão, o sistema reconhecerá esta como um comportamento padrão e não acionará as contramedidas especificadas sobre quedas.

3.6 Tropos4AS

Em Tropos4AS (*Tropos for Adaptive Systems*), Morandini et al. (2017) propõem extensões para a fase de projeto arquitetônico do Tropos (BRESCIANI et al., 2004) para modelar sistemas adaptativos baseados no modelo *Belief-Desire-Inention* (BDI) como uma arquitetura de referência (RAO; GEORGEFF, 1995). Tropos4AS possui um modelo de requisitos que pode ser diretamente mapeado para protótipos de software com arquitetura orientada a agentes, e assim ser executado para validação e refinamento dos requisitos.

A estrutura de Tropos é baseada em três principais aspectos de modelagem, buscando capturar o conhecimento de requisitos específicos, reduzir a lacuna entre abstrações de tempo-requisitos e uma representação em tempo de execução dos requisitos para adaptação. Estes aspectos são:

1. um modelo de objetivos incluindo informações sobre tipos de objetivos e condições de satisfação associadas aos mesmos;

2. um modelo de ambiente que represente os elementos que cercam o sistema em diferentes situações e como eles podem afetar a satisfação dos objetivos do sistema;

3. um modelo de falha para dar suporte a engenheiros para indicar estados de negócios indesejados e permitir que eles desenvolvam procedimentos de recuperação, tanto para antecipar falhas previsíveis quanto para se recuperar de falhas.

A abordagem introduz novos tipos de objetivos — *maintain-goals*, *achieve-goals* e *perform-goal* — e uma nova relação *inhibit* entre objetivos que especificam que um

objetivo (o inibidor) tem que ser parado em ordem para que outro objetivo (o inibido) seja alcançado/mantido.

O Tropos4AS também permite que os designers modelem elementos não intencionais usando diagramas de classes UML,[1] especificando recursos que pertencem a um agente e os que pertencem ao ambiente. A abordagem também permite a modelagem de estados indesejáveis (defeituosos), que são conhecidos como possíveis em tempo de execução e devem acionar a adaptação do sistema. No nível operacional, os modelos de objetivos são mapeados para Jadex,[2] plataforma utilizada para implementação em tempo de execução.

A Figura 13 apresenta uma representação gráfica dos conceitos de modelagem de um robô de limpeza. A parte (a) apresenta o modelo de objetivos estendido com tipos de objetivos e novos relacionamentos. De acordo com o exemplo, para o objetivo de manter a bateria carregada (Battery loaded) tem-se a classificação de *maintain-goals*. A parte (b) representa o modelo de ambiente com condições e artefatos, neste exemplo apresentando a condição de manutenção (MaintainCond.) que indica que a carga da bateria deve se manter superior a 10% (charge>10%) . A parte (c) expõe o modelo de falha, especificando falhas, erros e atividades de recuperação. Assim, sobre o objetivo associado a carga de bateria (Battery loaded) existe uma falha referente à bateria descarregada (battery discharged), associada ao erro onde a estação de carregamento estaria muito distante (ERROR: Charging station too distant) e em decorrência temos o objetivo associado referente a uma possível ação de recuperação (Recovery act.) e um plano (Recovery activity 2) para retomada do seu objetivo que foi quebrado.

3.7 QoS-aware Middleware

QoS-aware middleware (NAHRSTEDT et al., 2001) é uma solução para fornecer suporte a qualidade de serviço (*quality of service* ou QoS), que é exigido por uma nova geração de aplicações sensíveis ao QoS. Ele apresenta quatro aspectos-chave de um *QoS-aware middleware system*: *QoS specification* (responsável pela descrição do comportamento do aplicativo e parâmetros de QoS), *QoS translation and compilation* (responsável por traduzir o comportamento especificado da aplicação em configurações de aplicações candidatas para diferentes condições de recursos), *QoS setup* (responsável por selecionar e instanciar apropriadamente uma configuração específica) e *QoS adaptation* (responsável por se adaptar em tempo de execução às variações de recursos).

A operação de *QoS-aware middleware systems* é definida como: primeiro, gerar apropriadas especificações de QoS; segundo, traduzir e compilar múltiplas configurações de aplicações para o mesmo aplicativo a ser executado em ambientes heterogêneos; seguido por

[1] The Unified Modeling Language, <http://www.uml.org/>.
[2] A BDI Agent System, <http://jadex-agents.informatik.uni-hamburg.de/>.

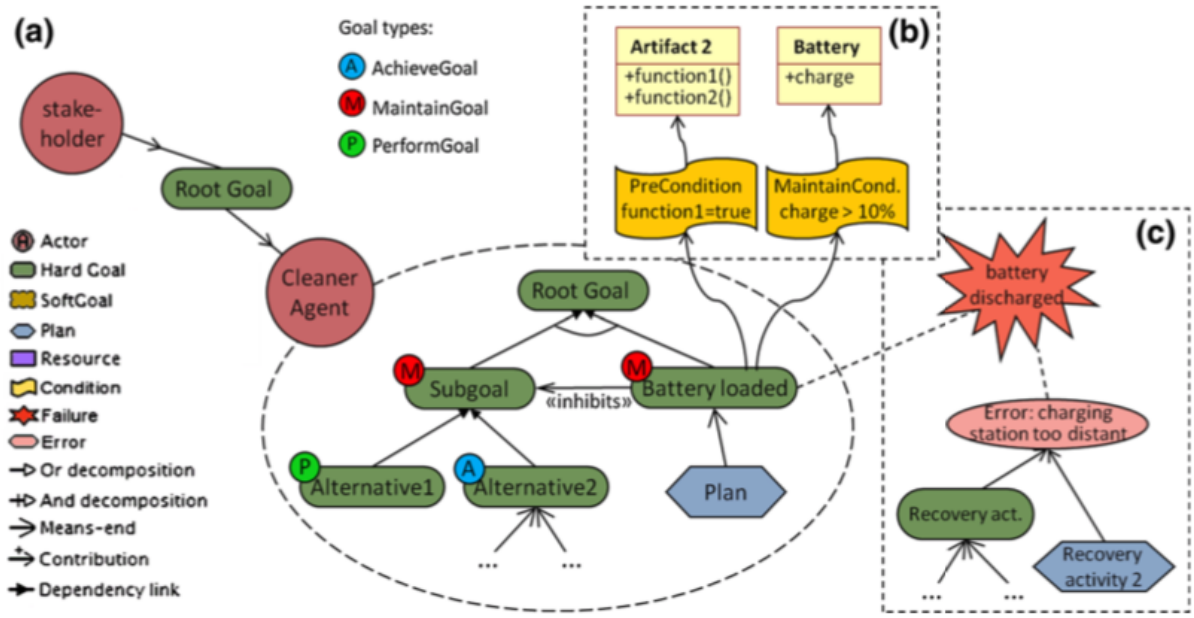

Figura 13 – Representação gráfica dos conceitos de modelagem apresentados por Morandini et al. (2017).

uma seleção de uma configuração apropriada e descobrindo os componentes da aplicação participante; e, finalmente, adaptar QoS em vários níveis.

3.8 LoREM

Em 2005, o trabalho de Berry, Cheng e Zhang (2005) definiu quatro níveis de Engenharia de Requisitos (ER) para modelagem de sistemas adaptativos. Goldsby et al. (2008) propõem então a abordagem LoREM (*Levels of RE for Modeling*), que reifica estes níveis originais, para descrever o trabalho de modelagem de ER feito por desenvolvedores de sistemas adaptativos. Cada nível corresponde ao trabalho de um tipo diferente de desenvolvedor para construir o modelo de objetivos especificando seus requisitos:

1. Nível 1: desenvolvedores de sistemas identificam os objetivos do sistema e os cursos dos sistemas que são adequados para os domínios que satisfazem os objetivos;

2. Nível 2: desenvolvedores de cenários de adaptação criam o conjunto de cenários de adaptação, que representam as transições entre os sistemas atual e o alvo de destino em tempo de execução, incluindo os requisitos de monitoramento, tomada de decisão e adaptação;

3. Nível 3: preocupa-se em identificar a infraestrutura de adaptação necessária para suportar os cenários previamente identificados;

4. Nível 4: pesquisa feita para melhorar os métodos e técnicas usadas nos outros níveis.

Goldsby et al. (2008) afirmam que a engenharia de requisitos para sistemas adaptativos possui três preocupações principais: condições de monitoramento para adaptação, ações de adaptação necessárias para busca do objetivo e processos de decisão para associar as condições de monitoramento às adaptações apropriadas. Dentre estas preocupações, as abordagens atuais focam em especificar condições de monitoramento e não se preocupam com desenvolvimento de modelos dos últimos dois pontos. Assim, Goldsby et al. (2008) propõem dois processos complementares para a modelagem destes tópicos no processo de ER para um sistema adaptativo. LoREM modela os requisitos de um sistema adaptativo utilizando modelos de objetivo i* (YU et al., 2011). Os modelos de objetivos i* representam os objetivos das partes interessadas, o comportamento não-adaptativo do sistema (lógica de negócios), o comportamento adaptativo e as necessidades de mecanismo de adaptação, sendo que cada um desses modelos aborda as três preocupações de ER (condições de monitoramento, procedimento de tomada de decisão e possíveis adaptações) do ponto de vista de um desenvolvedor específico.

Deste modo, a abordagem oferece um processo orientado a aplicação que pressupõe um conjunto maduro de mecanismos de adaptação que permite que um sistema adaptativo se adapte dinamicamente, e também apresenta um processo orientado por tecnologia que pressupõe um conjunto menos maduro de mecanismos de adaptação, restringindo o alcance do comportamento adaptativo. A Figura 14 apresenta a infraestrutura de adaptação do modelo desenvolvido. A Figura 15 exemplifica o modelo de adaptação do comportamento S1 para o comportamento S2.

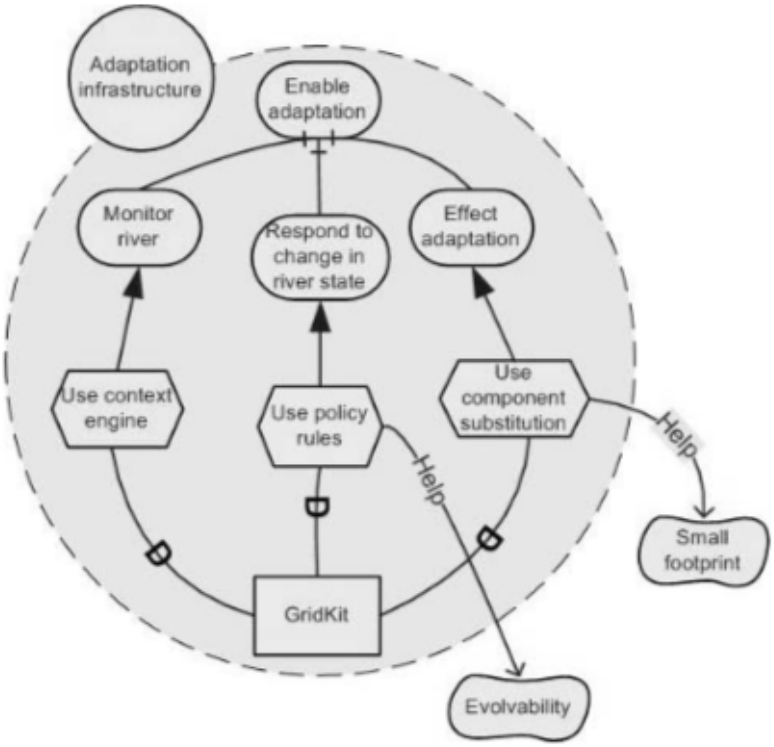

Figura 14 – Modelo de infraestrutura de adaptação (GOLDSBY et al., 2008).

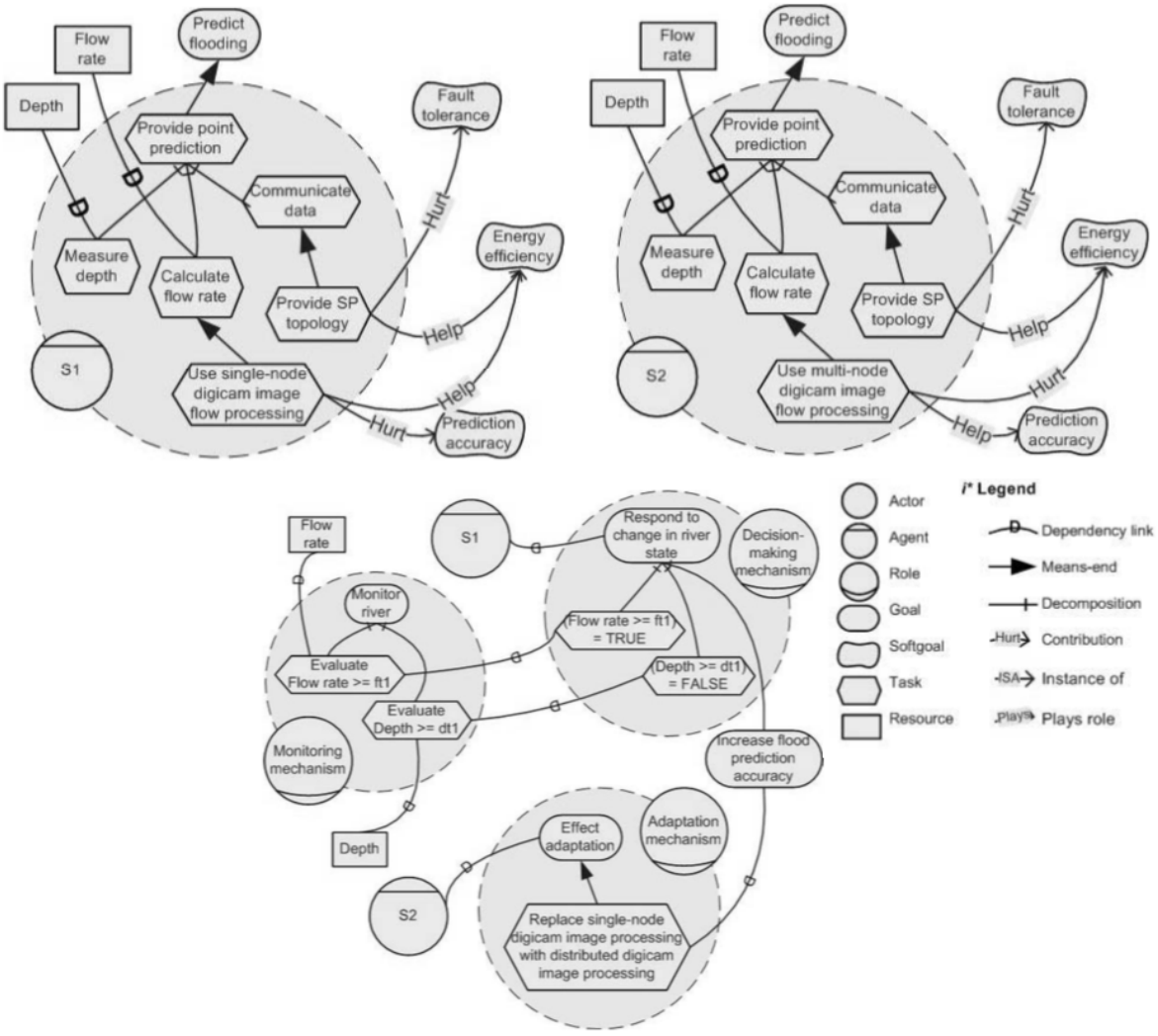

Figura 15 – Modelo de adaptação de S1 para S2 (GOLDSBY et al., 2008).

4 RASO

Neste capítulo é apresentada a *Requirements-based Adaptive Systems Ontology* (RASO), uma ontologia de domínio sobre abordagens baseadas em requisitos para sistemas adaptativos. Seguindo os primeiros passos do método SABiO, foi definido o propósito de RASO: ser uma representação formal e explícita dos conceitos para sistemas adaptativos baseados em modelo de requisitos. Para além disso, foram identificadas as intenções de uso, apresentas a seguir:

1. Servir de base conceitual para resolução de problemas de interoperabilidade sobre diferentes abordagens para desenvolvimento de sistemas adaptativos baseados em requisitos, possibilitando uma melhor integração e viabilizando o desenvolvimento de novas abordagens;
2. Servir como um vocabulário bem fundamentado para melhorar a compreensão e compartilhamento de conhecimento do domínio de requisitos para sistemas adaptativos, sendo especialmente útil para engenheiros de requisitos que trabalham neste campo;
3. Servir como ferramenta de suporte para criação ou reengenharia de abordagens baseadas em requisitos para o desenvolvimento de sistemas adaptativos;
4. Servir como modelo conceitual para a criação e / ou integração de ferramentas que suportam os usos pretendidos descritos em (1) e (2) acima.

Uma vez que sua finalidade e usos pretendidos foram definidos, os requisitos para a RASO foram levantados e documentados na forma de Questões de Competência (CQs). As CQs apresentadas foram elicitadas através de reuniões semanais realizadas com pesquisadores especialistas nas áreas de requisitos em tempo de execução, sistemas adaptativos e ontologias. As QCs foram definidas baseando-se nos resultados apresentados no mapeamento sistemático (cf. Capítulo 3), este sendo utilizado como base de informação consensual e delimitador para definição de escopo e requisitos da ontologia, refletindo diretamente as questões de competência. Devido a periodicidade dos encontros, observa-se na definição das CQs um processo altamente interativo de elicitação/documentação de requisitos e captura/formalização de ontologias. A cada nova reunião as CQs definidas anteriormente puderam ser refinadas, e a partir do aprimoramento das questões mais básicas obteve-se questões de maior complexidade. Os CQs da RASO são:

- **CQ1:** O que é um sistema adaptativo?
- **CQ2:** Quais são os componentes de um sistema adaptativo?

- **CQ3:** Como as adaptações são realizadas em um sistema adaptativo?
- **CQ4:** Como os sistemas adaptativos são especificados?
- **CQ5:** Quais requisitos o sistema adaptativo manipula em tempo de execução?

Como requisitos não-funcionais da RASO, foi definido que:

- **NFR1**: A ontologia proposta deve ser formalizada com base em uma ontologia de fundamentação.
- **NFR2**: A ontologia proposta deve representar o conhecimento consensual presente na literatura referente ao domínio da ontologia.
- **NFR3**: A ontologia proposta deve ser capaz de ser reutilizada em partes ou totalmente por qualquer outra ontologia de domínio fundamentada em UFO.

A Figura 16 mostra o modelo conceitual da RASO, relacionando conceitos de seu domínio com os reutilizados da *Software Engineering Ontology Network* (SEON, cf. Seção 2.2). A seguir, mostramos como a RASO fornece uma descrição clara e precisa desse domínio. Para exemplificar os conceitos da RASO, utilizaremos o Agendador de Reuniões, sistema que exemplifica a abordagem Zanshin (cf. Seção 3.4), ilustrado na Figura 12.

Um Adaptive System é um tipo especial de Software System, constituido de (relação *constituted of*) um ou mais Adaptive Feedback Loop Control Programs (*feedback loop* ou *controller*) e programas base (*base programs*, geralmente referenciados na literatura como sistemas alvo ou *target systems* (SOUZA, 2012)), representados em RASO como Adaptable Target Program. No exemplo, a implementação do controlador do Zanshin é uma instância de Adaptive Feedback Loop Control Program e o Agendador de Reuniões representa um Adaptable Target Program.

Uma vez que um *base program* é integrado a um *feedback loop* (em outras palavras, torna-se adaptativo, usando qualquer abordagem dada para o desenvolvimento de sistemas adaptativos (SOUZA; MYLOPOULOS, 2011b)), ele se torna um Adaptive System. Assim, os programas que precisam executar alguma adaptação serão os Adaptable Target Programs. Em nosso exemplo, essa integração consiste em implementar no sistema base (Agendador de Reuniões) uma API de *callback* especificada pelo Zanshin, que é chamada pelo controlador do Zanshin em tempo de execução, formando o Adaptive System "Agendador de Reuniões adaptativo".

Como Programs, ambos Adaptable Target Program e Adaptive Feedback Loop Control Program pretendem implementar (relação *intend to implement*) um Program Specification para satisfazer (relação *satisfy*) Requirement Artifacts. O primeiro implementa o que informalmente chamamos de funções principais do sistema, por exemplo, agendar reuniões,

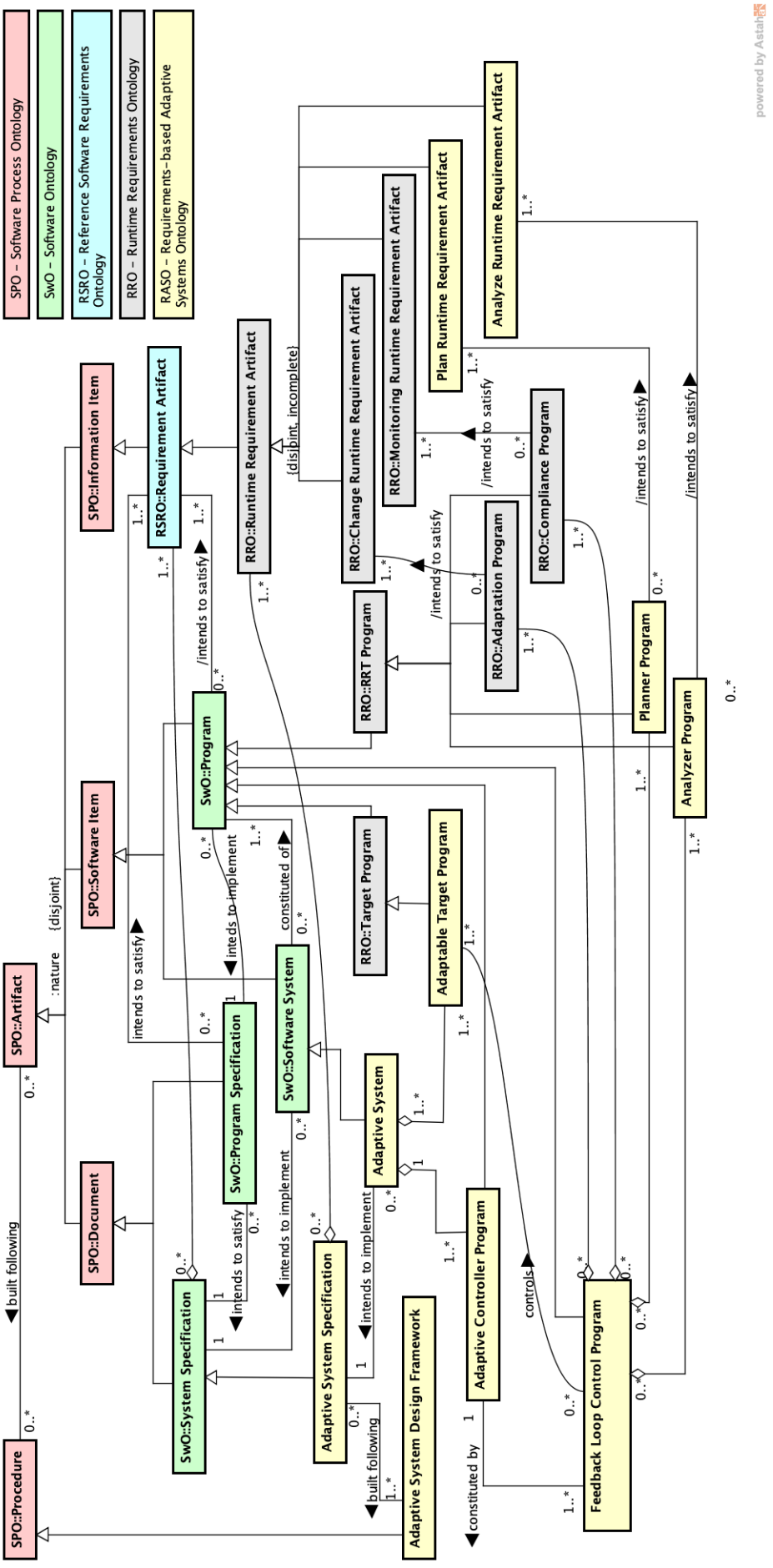

Figura 16 – A Requirements-based Adaptive Systems Ontology (RASO).

enquanto o segundo implementa funções de adaptação, ou seja, monitorar as funções principais e adaptá-las se necessário, formando um *feedback loop* (por exemplo, o MAPE loop (KEPHART; CHESS, 2003b)). Por exemplo, se o Agendador de Reuniões adaptativo detectar uma baixa taxa de participação em reuniões, ele poderia enviar e-mails para os participantes convidados para coletar corretamente seus horários.

Um Software Systems pretende implementar (*intends to implement*) uma especificação do sistema (System Specification) (DUARTE et al., 2018). Como tal, um Adaptive System pretende implementar (*intends to implement*) uma especificação do sistema adaptativo Adaptive System Specification. Em particular, além de especificar as principais funções do sistema, uma especificação de sistema adaptativo (Adaptive System Specification) é composta de (relação *is composed of*) um ou mais Runtime Requirement Artifacts (RRAs) referentes às funções de auto-adaptação do sistema. Este tipo particular de especificação é construída seguindo (*built following*) pelo menos um Adaptive System Design Framework, que SPO (BRINGUENTE; FALBO; GUIZZARDI, 2011) considera um Procedure.

Por exemplo, o Agendador de Reuniões adaptativo implementa sua especificação base (coletar horários, reservar sala, agendar reunião, etc.), especificada na forma de um modelo de objetivos. Além disso, por meio do Zanshin (Adaptive System Design Framework), incrementa seu modelo de requisitos com RRAs como, por exemplo, o *Awareness Requirement* que verifica se há baixa participação em reuniões agendadas (AR6 na Figura 12) e o *Evolution Requirement* associado que modifica a forma de coletar horários dos participantes para evitar conflitos de horário e aumentar a participação.

Finalmente, as funções de auto-adaptação do Adaptive System Specification são implementadas por meio dos Adaptive Feedback Loop Control Programs dos Adaptive Systems, que são constituídos por (relação *constituted by*) quatro tipos particularidades de RRT (Requirements at Runtime) Programs:

- Compliance Programs, responsáveis pelo monitoramento de outros programas, pretendem satisfazer (relação *intends to satisfy*) os Monitoring RRAs da especificação;
- Analyzer Programs, responsáveis pela analise dos dados monitorados, pretendem satisfazer (relação *intends to satisfy*) os Analyze RRAs da especificação;
- Planner Programs, responsáveis pelo planejamento (forma como será realizada) da adaptação, pretendem satisfazer (relação *intends to satisfy*) os Plan RRAs da especificação; e
- Adaptation Programs, responsáveis pela realização da adaptação, pretendem satisfazer (relação *intends to satisfy*) os Change RRAs da especificação.

Novamente, ilustrando os conceitos a partir do Agendador de Reuniões adaptativo,

a abordagem Zanshin fornece uma implementação de um Adaptive Feedback Loop Control Program baseado na plataforma Eclipse[1] e constituída por pacotes OSGi[2], dentre os quais o *Monitoring Bundle* (Compliance Program e Analyzer Program) monitora a satisfação de *Awareness Requirements* e envia relatórios para o *Adaptation Bundle* (Planner Program and Adaptation Program), que analisam *Evolution Requirements* para se adaptar.

A ilustração apresentada ao longo deste capítulo mostra que RASO representa adequadamente os conceitos relativos à abordagem Zanshin, uma das abordagens baseadas em requisitos para desenvolvimento de sistemas adaptativos. No entanto, meios mais sistemáticos de avaliação são necessários para garantir uma representação mais ampla do domínio em questão. Apresentamos o resultado da avaliação de RASO no próximo capítulo.

[1] <http://www.eclipse.org>
[2] <http://www.osgi.org>

5 Avaliação do Trabalho

Neste capítulo detalhamos como a *Requirements-based Adaptive Systems Ontology* (RASO) foi avaliada. A Seção 5.1 descreve os resultados de atividades de Verificação & Validação realizadas, enquanto a Seção 5.2 discute ameaças à validade das atividades de avaliação conduzidas.

5.1 Verificação e Validação da Ontologia

Buscando avaliar a RASO, foram aplicadas técnicas de verificação e validação, como instituído pelo método SABiO. Para a etapa de verificação, SABiO sugere uma tabela que mostre os elementos da ontologia necessários para responder às questões de competência (QCs) que foram levantadas, demonstrando assim que a ontologia satisfaz os requisitos conforme documentado. Para validação, a ontologia deve ser instanciada usando entidades do mundo real, demonstrando que é capaz de representar bem o domínio em questão.

A Tabela 5 mostra como cada uma das QCs predefinidas é respondida por conceitos e relações da RASO. Essa tabela também pode ser usada como uma ferramenta de rastreabilidade, suportando o gerenciamento de mudanças da ontologia. Vale a pena enfatizar que todos os conceitos de RASO são mencionados na Tabela 5. Isso mostra que eles são necessários e suficientes para atender aos requisitos da ontologia.

Para validação, estudamos as 32 publicações sobre as oito diferentes abordagens selecionadas em nosso mapeamento sistemático da literatura (cf. Seção 3). Em seguida, identificamos instâncias de conceitos da RASO (e alguns conceitos-chave da RRO também) em cada uma das abordagens, produzindo a tabela de instanciação, apresentada nas tabelas 6 e 7. As instâncias de elementos das abordagens podem ser apresentadas de forma explicita, como por exemplo o Adaptive System Design Framework, que representa um Framwork que possibilita a implementação de um sistema adaptativo, é apresentado em Dalpiaz, Giorgini e Mylopoulos (2012) como o Adaptive STS, em Baresi, Pasquale e Spoletini (2010) como o FLAGS, em Sawyer et al. (2010) como RELAX, em Morandini et al. (2017) como Tropos4AS e em Souza et al. (2011) como Zanshin, todos nomeados como abordagens que possibilitam a adaptação em um sistema alvo. O sistema alvo passível a adaptação, nomeado em RASO como Adaptable Target Program é um que sistema isoladamente possui suas funções padrão e, juntamente do Adaptive System Design Framework, permite a implementação de mecanismos de adaptação. Nas publicações encontramos estas implementações de sistemas descritas de forma explicita, principalmente em exemplos, como a Smart Home apresentada em Dalpiaz, Giorgini e Mylopoulos (2012),

Tabela 5 – Verificação das Questões de Competência de RASO.

CQ	**Concepts and *Relations***
CQ1 - O que é um sistema adaptativo?	Adaptive System *subtype of* Software System;
	Adaptive System *intends to implement* Adaptive System Specification.
CQ2 - Quais são os componentes de um sistema adaptativo?	Adaptive System *subtype of* Software System;
	Software System *constituted of* Program;
	Adaptive System *constituted of* Adaptable Target Program and Adaptive Feedback Loop Control Program;
	Adaptable Target Program *subtype of* Target Program *subtype of* Program;
	Adaptive Feedback Loop Control Program *subtype of* Program.
CQ3 - Como as adaptações são realizadas em um sistema adaptativo?	Adaptive Feedback Loop Control Program *controls* Adaptable Target Program.
	{Compliance Program, Analyzer Program, Planner Program, Adaptation Program} *part of* Adaptive Feedback Loop Control Program;
CQ4 - Como os sistemas adaptativos são especificados?	Adaptive System Specification *subtype of* System Specification;
	Runtime Requirement Artifact *part of* Adaptive System Specification;
	Adaptive System Specification *built following* Adaptive System Design Framework.
CQ5 - Quais requisitos o sistema adaptativo manipula em tempo de execução?	Adaptive System *constituted of* Adaptable Target Program and Adaptive Feedback Loop Control Program;
	{Compliance Program, Analyzer Program, Planner Program, Adaptation Program} *part of* Adaptive Feedback Loop Control Program;
	Compliance Program *intends to satisfy* Monitoring RRA;
	Analyzer Program *intends to satisfy* Analyze RRA;
	Planner Program *intends to satisfy* Plan RRA;
	Adaptation Program *intends to satisfy* Change RRA;
	{Monitoring RRA, Analyze RRA, Plan RRA, Change RRA} *subtype of* Runtime Requirement Artifact.

o Laundry System em Baresi, Pasquale e Spoletini (2010), o Ambiente Assisted Living (ALL) em Sawyer et al. (2010), o iCleaner em Morandini et al. (2017) e o Meeting Scheduler em Souza et al. (2011).

Para além dos elementos explícitos, alguns elementos são apresentados de forma implícita, em Baresi, Pasquale e Spoletini (2010) encontra-se o Adaptive Goal, em que suas funções representam 4 elementos da ontologia RASO. O Adaptive Goal de Baresi, Pasquale e Spoletini (2010), possui Triggers que especifica critérios de monitoramento sobre um requisito (em RASO apresentada como Monitoring Runtime Requirement Atifact) com base em Condition (em Raso apresentados como Analyze Runtime Requirement Artifact) responsáveis pela análise dos dados realizadas e acionam Actions (em RASO apresentados como Change Runtime Requirement Artifact) que especifica a adaptação a ser realizada quando este não esta de acordo com o especificado, que por sua vez são definidos de acordo com o Objective (em RASO apresentados como Plan Runtime Requirement Artifact) responsável pela definição de como a adaptação será realizada. Estes elementos possuem as mesmas funções que outros elementos presentes nas publicações selecionadas, como por exemplo: os Triggers de FLAGS (BARESI; PASQUALE; SPOLETINI, 2010) apresentam funções semelhantes as Satisfaction Conditions de Tropos4AS (MORANDINI et al., 2017), as Conditions de FLAGS apresentam funções semelhantes a Plans de Tropos4AS e Actions

Tabela 6 – Validação da RASO pela instanciação de seus conceitos de acordo com abordagens mais populares para o desenvolvimento de sistemas adaptativos (parte 1).

Concept	**Approach for the Development of Adaptive Systems**			
Adaptive System Design Framework	**Adaptive STS**	**FLAGS**	**LoREM**	**Unnamed approach applied to Necesity**
Adaptive System	Combination of the Smart Home example with the adaptation controller	Combination of the Laundry System example with the adaptation controller	The GridStix system	The Necesity system and the Context Management Middleware
Adaptive System Specification	Plan specifications for the Smart Home example	Goal model specification for the Laundry System example	Goal model specification for GridStix	Adaptive timers algorithm
Adaptable Target Program	Smart Home example implementation	Laundry System example implementation	GridStix's steady-state system implementation	The Necesity system implementation
Adaptive Feedback Loop Control Program	Self-Reconfiguration component (Monitor-Diagnose-Reconcile-Compensate cycle)	FLAGS framework implementation	Adaptation infrastructure	Context Management Middleware
Analyze Runtime Requirement Artifact	—	Condition (Adaptive Goal)	—	—
Analyzer Program	—	Data Collector	—	—
Plan Runtime Requirement Artifact	—	Objective (Adaptive Goal)	—	—
Planner Program	—	Monitor	—	—
Compliance Program	Monitor component	Monitor	Monitoring mechanism	Monitoring software
Monitoring Runtime Requirement Artifact	Fulfillment conditions	Triggers (Adaptive Goal)	Decision-making mechanism	Set of decision rules
Adaptation Program	Reconfiguration component	Monitor	Adaptation mechanism	Adapter component
Change Runtime Requirement Artifact	Activation rules	Actions (Adaptive Goal)	Adaptive step	Parameters which are set based on the history of use of the system

em FLAGS apresentam funções semelhantes a Recovery Activities em Tropos4AS, e por isso foram mapeados com o mesmo elemento em RASO. Note que o traço (—) em uma célula não significa que o conceito não está presente de fato na abordagem respectiva, mas sim que não é explicitamente nomeado.

A instanciação bem-sucedida da RASO com entidades dessas abordagens é um indicador da adequação da ontologia proposta como um modelo de referência desse domínio. As Tabelas 6 e 7 também mostram que a RASO, como modelo conceitual, não apresenta os problemas discutidos por Guizzardi (GUIZZARDI; FALBO; GUIZZARDI, 2008):

Tabela 7 – Validação da RASO pela instanciação de seus conceitos de acordo com abordagens mais populares para o desenvolvimento de sistemas adaptativos (parte 2).

Concept	**Approach for the Development of Adaptive Systems**			
Adaptive System Design Framework	**QoS-aware Middleware**	**RELAX**	**Tropos4AS**	**Zanshin**
Adaptive System	Combination of a video streaming application with the QoS Middleware	—	Combination of the iCleaner example with the Tropos4AS Middleware	Combination of the Meeting Scheduler example with the Eclipse/OSGi-based implementation
Adaptive System Specification	Qos Specification	Goal model for the Ambient Assisted Living example	Goal model for the iCleaner example	Goal model for the Meeting Scheduler example
Adaptable Target Program	A video streaming application	Ambient Assisted Living (ALL) example	iCleaner example implementation	Meeting Scheduler example implementation
Adaptive Feedback Loop Control Program	QoS Middleware (QoS-aware resource management)	—	Tropos4AS Middleware (MAPE Loop)	The Eclipse/OSGi-based implementation (MAPE Loop)
Analyze Runtime Requirement Artifact	QoS parameter	Specification using REL operator	Failure Conditions or Drop Conditions	—
Analyzer Program	QoS translation and compilation	—	—	—
Plan Runtime Requirement Artifact	—	—	Plan	—
Planner Program	QoS setup	—	—	—
Compliance Program	Observer component	—	Monitoring BDI agent	Monitoring OSGi bundle
Monitoring Runtime Requirement Artifact	—	Specification using MON operator	Satisfaction conditions	Awareness Requirement
Adaptation Program	QoS adaptation	—	Adaptation BDI agent	Adaptation OSGi bundle
Change Runtime Requirement Artifact	Application adaptation policies	Relaxed requirements	Recovery activities	Evolution Requirement

- *Sobrecarga de Constructos (Construct Overload)*: os conceitos de RASO representam uma única entidade nas abordagens, ou seja, não são sobrecarregados com significado;
- *Excesso de Constructos (Construct Excess)*: todos os conceitos de RASO têm instâncias na maioria das abordagens, ou seja, não há conceitos desnecessários na ontologia em relação ao domínio;
- *Redundância de Constructos (Construct Redundancy)*: não há dois conceitos de RASO que sistematicamente apontem para o mesmo conceito nas diferentes abordagens, ou seja, os conceitos da ontologia não são redundantes;
- *Incompletude (Incompleteness)*: as entidades relacionadas ao domínio de sistemas adaptativos nas diferentes abordagens foram todas mapeadas para conceitos de

RASO, ou seja, a ontologia é completa em relação ao seu escopo dado.

Como tal, ela pode servir como base conceitual para resolver problemas de interoperabilidade entre diferentes abordagens para o desenvolvimento de sistemas adaptativos, sendo que as tabelas 6 e 7 podem servir como ponto de partida para o mapeamento de conceitos entre as oito abordagens selecionadas. A RASO também pode ser usada para desenvolver ou integrar ferramentas de software existentes neste domínio, como feito, por exemplo, no domínio de medição de software (FONSECA; BARCELLOS; FALBO, 2017).

Outro uso interessante da ontologia seria realizar análises ontológicas das linguagens de modelagem usadas nessas abordagens, oferecendo recomendações para esclarecer sua semântica e garantir expressividade, como feito, por exemplo, no domínio de arquiteturas corporativas orientadas a serviços (NARDI; FALBO; ALMEIDA, 2014). As tabelas 6 e 7 ainda fornecem referências sobre as oito abordagens selecionadas, pois elas também podem ser avaliadas com relação à lucidez, solidez, laconicidade e completude (GUIZZARDI; FALBO; GUIZZARDI, 2008), como realizado com a RASO acima. Por exemplo, não foi possível instanciar nenhum dos conceitos de Program para a abordagem RELAX, pois os artigos definem a linguagem de especificação, mas não fornecem uma implementação. Como outro exemplo, a abordagem QoS-aware Middleware não torna explícito seu Monitoring Runtime Requirement Artifacts. Esse feedback pode ajudar a melhorar essas abordagens.

Em geral, a avaliação da RASO mostra que ela pode atuar como um vocabulário bem fundamentado do domínio do modelagem de sistemas adaptativos baseados em requisitos, melhorando a comunicação, a aprendizagem e a resolução de problemas neste domínio. No entanto é preciso destacar as ameaças à validade da ontologia aqui proposta.

5.2 Ameaças à Validade

O protocolo utilizado no mapeamento sistemático apresentado no Capítulo 3 buscou, dentre outras funções, reduzir o número de possíveis fragilidades no desenvolvimento do presente estudo. Todavia, se faz necessário destacar elementos que podem influenciar a validade dos resultados e, para isso, utilizamos o trabalho de Wohlin et al. (2012), que propõe uma lista de ameaças à validade, utilizada como guia neste trabalho. A lista de ameaças apresentadas na proposta de Wohlin et al. (2012), não é aplicável a todos os experimentos, mas pode servir de lista de verificação. A lista é dividida em 4 seções: Ameaças a Conclusões, Ameaças Internas, Ameaças a Construção e Ameaças Externas.

Ameaças à validade de conclusões relacionam-se a questões que afetam a capacidade de obter a conclusão correta sobre as relações entre o tratamento e o resultado de um experimento, e nesta pesquisa se aplica em:

- Confiabilidade das medidas: a validade de um experimento está ligada à confiabilidade das medidas, que podem depender de muitos fatores. Neste trabalho o quesito em questão pode ser aplicado à definição das strings de busca, pois os resultados dos artigos selecionados do mapeamento sistemático são diretamente dependentes da mesma;

- Confiabilidade da implementação do tratamento: na etapa do mapeamento sistemático, o tópico se relaciona ao modelo de pesquisa de cada buscador utilizado nas bases de dados, em que cada caso se utiliza de uma sintaxe distinta e a customização da string de busca para cada buscador deve manter a mesma semântica, a fim de garantir a busca do mesmo elemento independente da plataforma;

- Irrelevâncias aleatórias em ambiente experimental: ainda sobre o mapeamento sistemático, situações não consideradas no planejamento do mapeamento poderiam ter impactado de forma negativa na busca em bases de dados, se não identificadas durante a realização do experimento como, por exemplo, alguns erros de busca decorrentes da plataforma dos buscadores; problemas com o cache do navegador, viciando pesquisa; e erros ao exportar os resultados de busca;

- Heterogeneidade aleatória dos objetos: sempre existe heterogeneidade no grupo de estudos. Se o grupo for muito heterogêneo, existe o risco de que a variação devido às diferenças individuais seja maior do que devido ao tratamento. Considerando os objetos em questão como artigos resultantes da pesquisa nas bases de dados, foram definidos durante o mapeamento sistemático critérios de inclusão e exclusão de publicações, criando assim uma homogeneidade entre os objetos de análise, enquanto a ação pode ser considerada positiva para se criar uma padrão de análise, o mesmo também pode impactar no resultado, visto que os elementos são selecionados a partir de uma perspectiva restrita.

Ameaças à validade interna são influências que podem afetar a variável independência em relação à causalidade, sem o conhecimento do pesquisador. Assim, estas podem se apresentar como ameaças à conclusão em relação a uma possível causalidade entre tratamento e resultado (WOHLIN et al., 2012). Nesta pesquisa pode-se relacionar com as ameaças que se aplicam a desenvolvimentos em grupos únicos. Neste caso, o desenvolvimento do trabalho com reuniões semanais realizadas no grupo de pesquisa, com pesquisadores especialistas na área do estudo em questão. Não possuindo um outro grupo participante da construção do projeto, pode haver problemas para determinar se o *modus operandi* ou outro fator causou o resultado observado.

A ameaça à validade de construção diz respeito à generalização do resultado do experimento para o conceito por trás do experimento. Algumas ameaças estão relacionadas ao design do experimento, outras a fatores sociais:

- Design do experimento: abrangem questões relacionadas ao design do experimento e sua capacidade de refletir o construto a ser estudado. Considerando as estratégias metodológicas aqui empreendidas, as quais foram inspiradas no trabalho previamente realizado por Duarte (2016), é possível que modelo anterior possua particularidades não identificadas (ainda que os trabalhos apresentem forte similaridade) que reflitam de maneira negativa sobre o objeto de estudo desta pesquisa;

- Ameaças sociais à validade: estão relacionadas a questões referentes ao comportamento dos sujeitos e dos experimentadores, visto que estes podem agir de forma diferente ao proposto por algum fator social. Por exemplo, parte do trabalho foi desenvolvido com reuniões semanais com especialistas de domínio que já estiveram engajados em pesquisas similares e, portanto, poderiam desenvolver projeções em relação à ontologia, antes dos resultados iniciais do mapeamento sistemático serem realizados.

Ameaças à validade externa são condições que limitam a capacidade de generalizar os resultados de experimentos. No contexto desta pesquisa, por exemplo, pode ocorrer na seleção das publicações consideradas como relevantes e impactar na ontologia produzida a partir dos mesmos. Embora tenham sido utilizados meios de prevenir em relação a vícios e vieses, a seleção de publicações pode não refletir, de fato, o cenário atual no que se refere a sistemas adaptativos que se utilizam de modelo de requisitos e, por consequência, produzir uma ontologia que não abranja todas os modelos de adaptação propostos.

6 Considerações Finais

Este trabalho apresentou a RASO, uma ontologia de referência de domínio sobre o desenvolvimento de sistemas adaptativos baseado em requisitos. RASO foi construída usando o método de Engenharia de Ontologias SABiO, reutilizando conceitos de ontologias existentes em Engenharia de Software e construída com a ontologia de fundamentação UFO. A elicitação de requisitos e a captura da ontologia foram baseadas em publicações selecionadas a partir de um mapeamento sistemático da literatura referente a esse domínio. A ontologia foi avaliada por meio da verificação de que seu modelo conceitual proposto responde a todas as suas Questões de Competência e instanciando entidades do mundo real extraídas das abordagens mais citadas a partir do mapeamento sistemático da literatura utilizando os conceitos definidos pela RASO.

Dado como realizou-se a elicitação e validação de RASO, considera-se que ela representa com sucesso o estado-da-arte das abordagens baseadas em requisitos para o desenvolvimento de sistemas adaptativos, alcançando seus propósitos de servir como um vocabulário bem fundamentado para melhorar a compreensão e o conhecimento compartilhado neste domínio e como base conceptual para resolver problemas de interoperabilidade ou para a criação ou reengenharia de abordagens para o desenvolvimento de sistemas adaptativos.

Como trabalho futuro, considera-se:

- Estudar as abordagens retornadas em nosso mapeamento sistemático que não ficaram entre as 10 mais citadas, a fim de validar ainda mais a RASO ou complementar seu modelo conceitual;

- Ampliar o escopo e incluir na ontologia conceitos relacionados aos modelos que representam os requisitos para sistemas adaptativos;

- Fornecer uma caracterização mais formal da RASO, incluindo eventuais axiomas que não foram analisados neste trabalho, simulando a ontologia como forma de validação mais detalhada;

- Integrar adequadamente a RASO à rede de ontologias de Engenharia de Software SEON, mencionada na Seção 2.2;

- Construir uma versão operacional da RASO, a fim de implementar um protótipo sobre a interoperabilidade entre diferentes abordagens baseadas em requisitos para sistemas adaptativos;

- Combinar RASO com a Goal-Oriented Requirements Ontology (GORO) (NEGRI et al., 2017), realizando análise e reengenharia da abordagem Zanshin, incluindo sua linguagem de modelagem e implementação de seu Adaptive Feedback Loop Control Program;
- Construir um método para avaliar sistemas adaptativos, bem como criar métricas para medir qualitativamente esses sistemas.

Referências

AHMAD, M.; BELLOIR, N.; BRUEL, J.-M. Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems. *Journal of Systems and Software*, Elsevier, v. 107, p. 50–70, 2015. Citado na página 39.

AHMAD, M. et al. Using relax, sysml and kaos for ambient systems requirements modeling. *Procedia Computer Science*, Elsevier, v. 10, p. 474–481, 2012. Citado na página 39.

ANDERSSON, J. et al. Modeling dimensions of self-adaptive software systems. In: *Software Engineering for Self-Adaptive Systems*. [S.l.]: Springer, 2009. v. 5525, p. 27–47. Citado na página 31.

ANGELOPOULOS, K.; SOUZA, V. E. S.; MYLOPOULOS, J. Dealing with multiple failures in zanshin: a control-theoretic approach. In: ACM. *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.], 2014. p. 165–174. Citado na página 39.

BARESI, L.; PASQUALE, L. Adaptive goals for self-adaptive service compositions. In: IEEE. *Web Services (ICWS), 2010 IEEE international conference on*. [S.l.], 2010. p. 353–360. Citado na página 39.

BARESI, L.; PASQUALE, L.; SPOLETINI, P. Fuzzy Goals for Requirements-driven Adaptation. In: *Proc. of the 18th IEEE International Requirements Engineering Conference*. [S.l.]: IEEE, 2010. p. 125–134. Citado 8 vezes nas páginas 11, 19, 25, 39, 40, 41, 59 e 60.

BERRY, D. M.; CHENG, B. H. C.; ZHANG, J. The Four Levels of Requirements Engineering for and in Dynamic Adaptive Systems. In: *Proc. of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality*. [S.l.: s.n.], 2005. p. 95–100. Citado na página 49.

BOTIA, J. A.; VILLA, A.; PALMA, J. Ambient Assisted Living system for in-home monitoring of healthy independent elders. *Expert Systems with Applications*, Elsevier, v. 39, n. 9, p. 8136–8148, 2012. Citado 2 vezes nas páginas 39 e 46.

BRESCIANI, P. et al. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, Springer, v. 8, n. 3, p. 203–236, 2004. Citado 2 vezes nas páginas 43 e 47.

BRINGUENTE, A. C. O.; FALBO, R. A.; GUIZZARDI, G. Using a Foundational Ontology for Reengineering a Software Process Ontology. *Journal of Information and Data Management*, Brazilian Computer Society, v. 2, n. 3, p. 511–526, 2011. Citado 4 vezes nas páginas 26, 28, 29 e 56.

BUNGE, M. *Treatise on basic philosophy, Vol. 3: Ontology I: The furniture of the world*. [S.l.]: Reidel Publishing Company, 1977. Citado na página 31.

CHENG, B. H.; RAMIREZ, A.; MCKINLEY, P. K. Harnessing evolutionary computation

to enable dynamically adaptive systems to manage uncertainty. In: IEEE. *Combining Modelling and Search-Based Software Engineering (CMSBSE), 2013 1st International Workshop on*. [S.l.], 2013. p. 1–6. Citado na página 39.

CHENG, B. H. C. et al. Software engineering for self-adaptive systems: A research roadmap. In: *Software Engineering for Self-Adaptive Systems*. [S.l.]: Springer, 2009. v. 5525, p. 1–26. Citado 2 vezes nas páginas 19 e 23.

CHENG, B. H. C. et al. A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty. In: *Proc. of the 12th International Conference on Model Driven Engineering Languages and Systems (MODELS 09)*. [S.l.]: Springer, 2009. p. 468–483. Citado 2 vezes nas páginas 39 e 42.

DALPIAZ, F.; GIORGINI, P.; MYLOPOULOS, J. Adaptive socio-technical systems: a requirements-based approach. *Requirements Engineering*, Springer, v. 18, n. 1, p. 1–24, 2012. Citado 7 vezes nas páginas 11, 19, 39, 42, 43, 44 e 59.

DUARTE, B. B. *RRO Uma Ontologia Sobre o Uso de Requisitos de Software em Tempo de Execução*. Dissertação (Mestrado) — Dissertação de Mestrado, Universidade Federal do Espírito Santo, 2016. Citado 2 vezes nas páginas 25 e 65.

DUARTE, B. B. et al. Ontological Foundations for Software Requirements with a Focus on Requirements at Runtime. *Applied Ontology*, IOS Press, preprint, n. preprint, p. 1–33, 2018. Citado 7 vezes nas páginas 11, 28, 29, 32, 33, 56 e 77.

FALBO, R. A. SABiO: Systematic Approach for Building Ontologies. In: *Proc. of the Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*. [S.l.]: CEUR, 2014. Citado 4 vezes nas páginas 11, 21, 25 e 26.

FONSECA, V. S.; BARCELLOS, M. P.; FALBO, R. de A. An ontology-based approach for integrating tools supporting the software measurement process. *Science of Computer Programming*, Elsevier, v. 135, p. 20–44, 2017. Citado na página 63.

FREDERICKS, E. M.; CHENG, B. H. An empirical analysis of providing assurance for self-adaptive systems at different levels of abstraction in the face of uncertainty. In: IEEE. *IEEE/ACM 8th International Workshop on Search-Based Software Testing (SBST)*. [S.l.], 2015. p. 8–14. Citado na página 39.

FREDERICKS, E. M.; DEVRIES, B.; CHENG, B. H. Autorelax: automatically relaxing a goal model to address uncertainty. *Empirical Software Engineering*, Springer, v. 19, n. 5, p. 1466–1501, 2014. Citado na página 39.

GOLDSBY, H. J. et al. Goal-Based Modeling of Dynamically Adaptive System Requirements. In: *Proc. of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*. [S.l.]: IEEE, 2008. p. 36–45. Citado 7 vezes nas páginas 11, 25, 39, 49, 50, 51 e 52.

GRÜNINGER, M.; FOX, M. Methodology for the Design and Evaluation of Ontologies. In: *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*. [S.l.: s.n.], 1995. Citado 2 vezes nas páginas 21 e 26.

GUIZZARDI, G. *Ontological foundations for structural conceptual model*. Tese (Doutorado)

— University of Twente, The Netherlands, 2005. Citado na página 21.

GUIZZARDI, G.; FALBO, R. d. A.; GUIZZARDI, R. S. S. Grounding software domain ontologies in the unified foundational ontology (ufo): The case of the ode software process ontology. In: *Proc. of the 11th Iberoamerican Conference on Software Engineering (CIbSE)*. [S.l.: s.n.], 2008. p. 127–140. Citado 3 vezes nas páginas 27, 61 e 63.

INGOLFO, S.; SOUZA, V. E. S. Law and adaptivity in requirements engineering. In: IEEE PRESS. *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.], 2013. p. 163–168. Citado na página 39.

JURETA, I.; MYLOPOULOS, J.; FAULKNER, S. Revisiting the core ontology and problem in requirements engineering. In: IEEE. *International Requirements Engineering, 2008. RE'08. 16th IEEE*. [S.l.], 2008. p. 71–80. Citado 4 vezes nas páginas 30, 31, 32 e 45.

JURETA, I. J. et al. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: IEEE. *Requirements Engineering Conference (RE), 2010 18th IEEE International*. [S.l.], 2010. p. 115–124. Citado na página 31.

JURETA, I. J.; MYLOPOULOS, J.; FAULKNER, S. A core ontology for requirements. *Applied Ontology*, IOS Press, v. 4, n. 3-4, p. 169–244, 2009. Citado 4 vezes nas páginas 30, 31, 32 e 45.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE, n. 1, p. 41–50, 2003. Citado na página 24.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE, v. 36, n. 1, p. 41–50, jan 2003. Citado na página 56.

KITCHENHAM, B. A.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007. Citado 3 vezes nas páginas 20, 32 e 33.

LADDAGA, R.; ROBERTSON, P. Self adaptive software: A position paper. In: CITESEER. *SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems*. [S.l.], 2004. v. 31, p. 19. Citado na página 24.

LAMSWEERDE, A. van et al. The KAOS Project: Knowledge Acquisition in Automated Specification of Software. In: *Proc. of the AAAI Spring Symposium Series, Stanford University*. [S.l.]: AAAI, 1991. p. 59–62. Citado na página 40.

LEMOS, R. de et al. Software engineering for self-adaptive systems: A second research roadmap. In: *Software Engineering for Self-Adaptive Systems II*. [S.l.]: Springer, 2013. p. 1–32. Citado na página 23.

MORANDINI, M.; MARCHETTO, A.; PERINI, A. Requirements comprehension: A controlled experiment on conceptual modeling methods. In: IEEE. *Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on*. [S.l.], 2011. p. 53–60. Citado na página 39.

MORANDINI, M.; PENSERINI, L.; PERINI, A. Operational semantics of goal models in adaptive agents. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. *Proceedings of The 8th International Conference on*

Autonomous Agents and Multiagent Systems-Volume 1. [S.l.], 2009. p. 129–136. Citado na página 39.

MORANDINI, M. et al. Engineering requirements for adaptive systems. *Requirements Engineering*, Springer, v. 22, n. 1, p. 77–103, 2017. Citado 7 vezes nas páginas 11, 25, 39, 47, 49, 59 e 60.

MORANDINI, M.; PERINI, A.; MARCHETTO, A. Empirical evaluation of tropos4as modelling. *iStar*, Citeseer, v. 766, p. 14–19, 2011. Citado na página 39.

NAHRSTEDT, K. et al. QoS-Aware Middleware for Ubiquitous and Heterogeneous Environments. *IEEE Communications Magazine*, IEEE, v. 39, n. 11, p. 140–148, 2001. Citado 2 vezes nas páginas 39 e 48.

NARDI, J. C.; FALBO, R. de A.; ALMEIDA, J. P. A. An Ontological Analysis of Service Modeling at ArchiMate's Business Layer. In: IEEE. *Proc. of the 18th International IEEE Enterprise Distributed Object Computing Conference*. [S.l.], 2014. p. 92–100. Citado na página 63.

NEGRI, P. P. et al. Towards an ontology of goal-oriented requirements. In: *Proc. of the 20th Ibero-American Conference on Software Engineering, Requirements Engineering track*. [S.l.: s.n.], 2017. Citado na página 68.

PASQUALE, L.; BARESI, L.; NUSEIBEH, B. Towards adaptive systems through requirements@runtime. In: *6th Workshop on Models@run.time*. [S.l.: s.n.], 2011. Citado na página 39.

PASQUALE, L.; SPOLETINI, P. Monitoring fuzzy temporal requirements for service compositions: Motivations, challenges and experimental results. In: IEEE. *Requirements Engineering for Systems, Services and Systems-of-Systems (RESS), 2011 Workshop on*. [S.l.], 2011. p. 63–69. Citado na página 39.

PEÇANHA, C. C. et al. RASO: an Ontology on Requirements for the Development of Adaptive Systems. *Cadernos do IME - Série Informática*, UERJ, v. 41, n. 1, p. 57–76, 2018. Citado na página 22.

PEÇANHA, C. C.; DUARTE, B. B.; SOUZA, V. E. S. RASO: an Ontology on Requirements for the Development of Adaptive Systems. In: *Proc. of the 21st Workshop on Requirements Engineering (WER 2018)*. Rio de Janeiro, RJ, Brasil: PUC-RIO, 2018. p. 1–14. Citado na página 22.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, Elsevier, v. 64, p. 1–18, 2015. Citado na página 33.

QURESHI, N. A.; JURETA, I. J.; PERINI, A. Requirements engineering for self-adaptive systems: Core ontology and problem statement. In: SPRINGER. *International Conference on Advanced Information Systems Engineering*. [S.l.], 2011. p. 33–47. Citado 2 vezes nas páginas 30 e 31.

RAMIREZ, A. J.; CHENG, B. H. Automatic derivation of utility functions for monitoring software requirements. In: SPRINGER. *International Conference on Model Driven Engineering Languages and Systems*. [S.l.], 2011. p. 501–516. Citado na página 39.

RAMIREZ, A. J. et al. Relaxing claims: Coping with uncertainty while evaluating assumptions at run time. In: SPRINGER. *International Conference on Model Driven Engineering Languages and Systems*. [S.l.], 2012. p. 53–69. Citado na página 39.

RAMIREZ, A. J. et al. Automatically relaxing a goal model to cope with uncertainty. In: SPRINGER. *International Symposium on Search Based Software Engineering*. [S.l.], 2012. p. 198–212. Citado na página 39.

RAO, A. S.; GEORGEFF, M. P. BDI Agents: From Theory to Practice. In: *Proc. of the 1st International Conference on Multiagent Systems*. San Francisco, CA, USA: AAAI, 1995. Citado na página 47.

REINHARTZ-BERGER, I.; STURM, A.; WAND, Y. Comparing functionality of software systems: An ontological approach. *Data & Knowledge Engineering*, Elsevier, v. 87, p. 320–338, 2013. Citado na página 31.

ROSS, D. T.; SCHOMAN, K. E. Structured analysis for requirements definition. *IEEE transactions on Software Engineering*, IEEE, n. 1, p. 6–15, 1977. Citado na página 25.

RUY, F. B. et al. SEON: A Software Engineering Ontology Network. In: SPRINGER. *Proc. of the 20th International Conference on Knowledge Engineering and Knowledge Management*. [S.l.], 2016. p. 527–542. Citado 2 vezes nas páginas 26 e 28.

SAWYER, P. et al. Requirements-Aware Systems: A research agenda for RE for self-adaptive systems. In: *Proc. of the 18th IEEE International Requirements Engineering Conference*. [S.l.]: IEEE, 2010. p. 95–103. Citado 5 vezes nas páginas 25, 39, 41, 59 e 60.

SOARES, M. et al. Core ontology to aid the goal oriented specification for self-adaptive systems. In: *New Advances in Information Systems and Technologies*. [S.l.]: Springer, 2016. p. 609–618. Citado na página 31.

SOUZA, V. E. S. *Requirements-based Software System Adaptation*. Tese (Doutorado) — University of Trento, Italy, 2012. Citado 8 vezes nas páginas 11, 19, 23, 24, 25, 45, 46 e 54.

SOUZA, V. E. S. et al. Requirements-driven software evolution. *Computer Science-Research and Development*, Springer, v. 28, n. 4, p. 311–329, 2013. Citado na página 39.

SOUZA, V. E. S.; LAPOUCHNIAN, A.; MYLOPOULOS, J. (requirement) evolution requirements for adaptive systems. In: IEEE PRESS. *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.], 2012. p. 155–164. Citado na página 39.

SOUZA, V. E. S.; LAPOUCHNIAN, A.; MYLOPOULOS, J. Requirements-driven qualitative adaptation. In: SPRINGER. *OTM Confederated International Conferences"On the Move to Meaningful Internet Systems"*. [S.l.], 2012. p. 342–361. Citado na página 39.

SOUZA, V. E. S. et al. Awareness Requirements for Adaptive Systems. In: *Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.]: ACM, 2011. p. 60–69. Citado 5 vezes nas páginas 19, 39, 45, 59 e 60.

SOUZA, V. E. S.; MYLOPOULOS, J. From awareness requirements to adaptive

systems: A control-theoretic approach. In: IEEE. *2011 2nd International Workshop on Requirements@ Run. Time (RE@ RunTime)*. [S.l.], 2011. p. 9–15. Citado na página 39.

SOUZA, V. E. S.; MYLOPOULOS, J. From Awareness Requirements to Adaptive Systems: a Control-Theoretic Approach. In: *Proc. of the 2nd International Workshop on Requirements@Run.Time*. [S.l.]: IEEE, 2011. p. 9–15. Citado na página 54.

SOUZA, V. E. S.; MYLOPOULOS, J. Designing an adaptive computer-aided ambulance dispatch system with zanshin: an experience report. *Software: Practice and Experience*, Wiley, v. 45, n. 5, p. 689–725, may 2013. Citado na página 45.

SOUZA, V. E. S.; MYLOPOULOS, J. Designing an adaptive computer-aided ambulance dispatch system with zanshin: an experience report. *Software: Practice and Experience*, Wiley Online Library, v. 45, n. 5, p. 689–725, 2015. Citado na página 39.

TALLABACI, G.; SOUZA, V. E. S. Engineering adaptation with zanshin: an experience report. In: IEEE PRESS. *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.], 2013. p. 93–102. Citado na página 39.

WHITTLE, J. et al. Relax: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering*, Springer, v. 15, n. 2, p. 177–196, 2010. Citado 3 vezes nas páginas 19, 39 e 41.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012. Citado 2 vezes nas páginas 63 e 64.

YANG, Z. et al. A systematic literature review of requirements modeling and analysis for self-adaptive systems. In: SPRINGER. *International Working Conference on Requirements Engineering: Foundation for Software Quality*. [S.l.], 2014. p. 55–71. Citado 3 vezes nas páginas 19, 25 e 32.

YU, E. S. K. et al. *Social Modeling for Requirements Engineering*. 1st. ed. [S.l.]: MIT Press, 2011. Citado na página 50.

Apêndices

APÊNDICE A – Artigos de Controle

As tabelas que se seguem apresentam uma listagem dos 86 artigos de controle utilizados como guias no pameamento sistemático desta pesquisa. Estes artigos são provenientes da pesquisa de Duarte et al. (2018). O detalhamento do processo do mapeamento sistemático está descrito no Capítulo 3.

Tabela 8 – Artigos de Controle (parte 1).

Código	**Título do Artigo**
AC1	(Requirement) evolution requirements for adaptive systems
AC2	A lightweight framework for dynamic GUI data verification based on scripts
AC3	A middleware and policy-based adaptation framework to simplify software evolution: An implementation on unit trust enterprise system
AC4	A proactive approach for context-aware self-adaptive mobile applications to ensure Quality of Service
AC5	A requirements-based approach for the design of adaptive systems
AC6	A Runtime evaluation methodology and framework for autonomic systems
AC7	A tool for monitoring and maintaining system trustworthiness at runtime
AC8	ACon: A learning-based approach to deal with uncertainty in contextual requirements at runtime
AC9	Adaptive agent model: An agent interaction and computation model
AC10	Adaptive Goals for Self-Adaptive Service Compositions
AC11	Adaptive Reasoning in the Presence of Imperfect Security Requirements
AC12	An automated approach to monitoring and diagnosing requirements
AC13	An Empirical Analysis of Providing Assurance for Self-Adaptive Systems at Different Levels of Abstraction in the Face of Uncertainty
AC14	Are your sites down? Requirements-driven self-tuning for the survivability of web systems
AC15	Automated mapping from goal models to self-adaptive systems
AC16	Automatic Derivation of Utility Functions for Monitoring Software Requirements
AC17	Continuous adaptive requirements engineering: An architecture for self-adaptive service-based applications
AC18	Dealing with multiple failures in zanshin: A control-theoretic approach
AC19	Dealing with softgoals at runtime: A fuzzy logic approach
AC20	Efficient parallel reasoning on fuzzy goal models for run time requirements verification
AC21	Feedback-aware requirements documents for smart devices
AC22	Flexibility requirements in real-world process scenarios and prototypical realization in the care domain
AC23	From a goal-oriented methodology to a BDI agent language: The case of Tropos and alan
AC24	From awareness requirements to adaptive systems: A control-theoretic approach
AC25	Fuzzy goals for requirements-driven adaptation
AC26	Goal-driven adaptation of service-based systems from runtime monitoring data
AC27	GODA: A goal-oriented requirements engineering framework for runtime dependability analysis
AC28	Implicit Priorities in Adaptation Requirements
AC29	Incremental runtime-generation of optimisation problems using RAG-controlled rewriting
AC30	Intention-oriented modelling support for GORE in elastic cloud applications

Tabela 9 – Artigos de Controle (parte 2).

Código	**Título do Artigo**
AC31	Intention-oriented programming support for runtime adaptive autonomic cloud-based applications
AC32	Investigation of feature run-time conflicts on goal model-based reuse
AC33	Market-aware requirements
AC34	Mitigating the obsolescence of quality specifications models in service-based systems
AC35	Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems
AC36	Modeling Self-adaptive Software Systems with Learning Petri Nets
AC37	Operational semantics of goal models in adaptive agents
AC38	Privacy arguments: Analysing selective disclosure requirements for mobile applications
AC39	QoS-aware middleware for ubiquitous and heterogeneous environments
AC40	Rationalism with a dose of empiricism: Case-based reasoning for requirements-driven self-adaptation
AC41	Rationalism with a dose of empiricism: Combining goal reasoning and case-based reasoning for self-adaptive software systems
AC42	Reasoning about adaptive requirements for self-adaptive systems at runtime
AC43	Reconciling system requirements and runtime behavior
AC44	Refining goal models by evaluating system behaviour
AC45	RELAX: A language to address uncertainty in self-adaptive systems requirement
AC46	RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems
AC47	Relaxing Claims: Coping with Uncertainty While Evaluating Assumptions at Run Time
AC48	RE-PREF: Support for REassessment of PREFerences of Non-functional Requirements for Better Decision-Making in Self-Adaptive Systems
AC49	Requirements engineering for adaptive Service Based Applications
AC50	Requirements engineering for self-adaptive systems: Core ontology and problem statement
AC51	Requirements models for design- and runtime: A position paper
AC52	Requirements reflection: requirements as runtime entities
AC53	Requirements tracing to support change in dynamically adaptive systems
AC54	Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems
AC55	Requirements-driven adaptive security: Protecting variable assets at runtime
AC56	Requirements-driven qualitative adaptation
AC57	Requirements-driven self-optimization of composite services using feedback control
AC58	Requirements-driven software evolution
AC59	Runtime goal models: Keynote

Tabela 10 – Artigos de Controle (parte 3).

Código	**Título do Artigo**
AC60	Runtime models for automatic reorganization of multi-robot systems
AC61	Run-time requirements verification for reconfigurable systems
AC62	Run-time verification of MSMAS norms using event calculus
AC63	SACRE: A tool for dealing with uncertainty in contextual requirements at runtime
AC64	Self-adaptation through incremental generative model transformations at runtime
AC65	Self-explanation in adaptive systems based on runtime goal-based models
AC66	Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop
AC67	Self-tuning of software systems through goal-based feedback loop control
AC68	Social software product lines
AC69	Software Product Line Engineering for Developing Self-Adaptive Systems: Towards the Domain Requirements
AC70	Software self-reconfiguration: A BDI-based approach
AC71	Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks
AC72	Synthesis of observers for autonomic evolutionary systems from requirements models
AC73	The Generation and Evolution of Adaptation Rules in Requirements Driven Self-Adaptive Systems
AC74	The reassessment of preferences of non-functional requirements for better informed decision-making in self-adaptation
AC75	The requirements problem for adaptive systems
AC76	Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems
AC77	Towards a goal-driven approach to action selection in self-adaptive software
AC78	Towards adaptive systems through requirements@runtime?
AC79	Towards an ontology of Requirements at Runtime
AC80	Towards Dynamic Evolution of Self-Adaptive Systems Based on Dynamic Updating of Control Loops
AC81	Towards Requirements-driven Autonomic Systems Design
AC82	Towards variability design as decision boundary placement
AC83	Uncertainty handling in goal-driven self-optimization - Limiting the negative effect on adaptation
AC84	Uncertainty modeling of self-adaptive software requirement
AC85	Using requirements traceability links at runtime - A position paper
AC86	Visualizing the Analysis of Dynamically Adaptive Systems Using i* and DSLs