



Pós-Graduação em Ciência da Computação

Monique Conceição Soares

An Ontology to aid the Goal-oriented Requirements Elicitation and Specification for Self-Adaptive Systems



Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

<http://cin.ufpe.br/~posgraduacao>

RECIFE

2017

Monique Conceição Soares

An Ontology to aid the Goal-oriented Requirements Elicitation and Specification for Self-Adaptive Systems

Ph.D. Thesis presented to the Post-Graduation Program in Computer Science of the Informatics Centre of the Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Philosophy Doctor in Computer Science.

Advisor: **Jaelson Freire Brelaz de Castro**

Co-Advisor: **Ph.D. Carla T.L.L. Silva Schuenemann**

RECIFE
2017

Catálogo na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S676o Soares, Monique Conceição
An ontology to aid the goal-oriented requirements elicitation and specification for self-adaptive systems / Monique Conceição Soares. – 2017.
245 f.: il., fig., tab.

Orientador: Jaelson Freire Brelaz de Castro.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Engenharia de software. 2. Engenharia de requisitos. 3. Ontologia. I. Castro, Jaelson Freire Brelaz de (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2018-031

Monique Conceição Soares

**An Ontology to aid the Goal-oriented Requirements Elicitation and
Specification for Self-Adaptive Systems**

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação.

Aprovado em: 11/09/2017

Orientador: Prof. Dr. Jaelson Freire Brelaz de Castro

BANCA EXAMINADORA

Prof. Dr. Frederico Luiz Gonçalves de Freitas
Centro de Informática / UFPE

Prof. Dr. Robson do Nascimento Fidalgo
Centro de Informática / UFPE

Prof. Dr. João Baptista da Silva Araújo Junior
Departamento de Informática / Universidade Nova Lisboa

Profa. Dra. Fernanda Maria Ribeiro de Alencar
Departamento de Eletrônica e Sistemas / UFPE

Prof. Dr. Vitor Estevão Silva Souza
Departamento de Informática / UFES

I dedicate my PhD thesis to my mother, Marinalva Maria de Santana.

Acknowledgements

I thank God for my life offered to Him, for all the Grace given to me, for the things allowed to me, and for the people I have in my life, specially Raphael Figueiredo and Rebecca Soares. And I thank Our Lady, for being my mother and owner.

I thank to my mom and brothers to be my family and Viviane Figueiredo for be like a sister for me. I thank to every person of the Santana and Soares families.

I thank the Catholic Community Benedict XVI - Maanaim, for be my family and my home. I thank Elizama, Manu Sampaio and Juliana Martins.

I thank all my friends, specially Oberdan.

I thank the friends of APG (Associação de Pós-Graduação - Postgraduate Association) for tolerate me, especially Bruno Maciel, Elaine, Hilário and Paulo Henrique.

I thank the members of the LER (Laboratório de Engenharia de Requisitos - Laboratory of Requirements Engineering) specially Gabriela Guedes.

I thank the Center of Informatics for the great structure and the best professors and employees.

I thank Carla to be an incredible person, with a huge heart and a dedicated and a worried adviser, as well as a wonderful friend. And I thank Jaelson for the guidance.

I also thank the examiners members of this thesis, Dr. Fernanda Alencar, Dr. Fred Freitas, Dr. João Araújo, Dr. Robson Fidalgo and Dr. Vítor Souza, for their valuable collaboration and contribution.

*"The man of science will really help humanity if he keeps 'the sense of man's
transcendence over the world and of God's transcendence over man. "
(St. John Paul II)*

Abstract

Self-Adaptive Systems (SAS) can adapt their own behavior in response to context information or changes in the environment and also in response to their own behavior. The interest in requirements engineering for SAS has grown in recent years, but despite this, some works involving requirements specification of these systems do not guide requirements elicitation. Goal-oriented requirements engineering (GORE) modeling languages are widely used to specify requirements for SAS. There are GORE modeling languages specifically proposed for the SAS domain and each of them presents a fixed and small set of concepts. Ontologies can be used to overcome the limitation of concepts, since they can help in the representation of concepts within a domain, as well as in the communication and specification of requirements. The purpose of this thesis is to provide a richer set of SAS concepts to guide the elicitation and specification of requirements for such systems. An ontology for SAS is proposed, as well as a process to guide the use of the ontology for eliciting and specifying requirements for SAS. The unique core ontology for requirements for SAS in literature does not cover all main concepts that SAS involves, like the modeling dimensions and a feedback loop. In order to achieve the objective, first, two systematic literature reviews (SLRs) were performed to analyze the work involving knowledge representation for SAS and context-aware systems. A total of twenty-three studies were selected in both. Then, three GORE modeling languages for SAS were analyzed - Tropos4AS, AdaptiveRML and Design Goal Model - to identify the concepts that these languages are able to represent. It was observed that the analyzed languages do not represent most of the concepts involved in the SAS domain. With the results of both SLRs and the analysis of the GORE modeling languages, an ontology was proposed to aid the requirements engineer to perform the elicitation and specification of SAS. To create the ontology, three methodologies were used: Uschold and Gruninger's, METHONTOLOGY and SABiO. The proposed ontology covers the main concepts of self-adaptive systems, such as the feedback loop concepts, context, the modeling dimensions for SAS, and goal-oriented requirements. It was also proposed a process for the use of the ontology. The ontology evaluation was based on six criteria: comprehensiveness, verification, validation, utility, easiness of use and accordance. Our ontology is embracing in comparison to the related works selected in both SLRs. The ontology was verified and validated by instantiating a multimedia news system. The usefulness and easiness of use of both the ontology and the process were evaluated by case study and a survey, where requirements engineers used the process to instantiate an ambulance dispatch system. This evaluation found that the ontology is useful, although the process is not so easy to use. Another survey was answered by SAS specialists to evaluate the accordance of the ontology, who agreed with the concepts of the ontology.

Keywords: Ontology. Specification. Elicitation. Self-adaptive systems. Requirements engineering. Goal-oriented.

Resumo

Sistemas auto-adaptativos (Self-Adaptive Systems - SAS) conseguem adaptar o próprio comportamento em resposta a informações de contexto ou mudanças no ambiente e também em resposta ao próprio comportamento. O interesse em engenharia de requisitos para SAS tem crescido nos últimos anos, mas apesar disto, os trabalhos que envolvem especificação de requisitos desses sistemas não guiam a elicitação de requisitos. Linguagens de modelagem de engenharia de requisitos orientados a objetivos (GORE) são muito utilizadas para especificar requisitos para SAS. Existem linguagens de modelagem GORE que foram propostas especificamente para o domínio de SAS e cada uma apresenta um conjunto fixo e pequeno de conceitos. Ontologias podem ser utilizadas para superar essa limitação, já que elas ajudam na representação de conceitos dentro de um domínio, bem como na comunicação e especificação de requisitos. O objetivo desta tese é fornecer um conjunto mais rico de conceitos para SAS para orientar a elicitação e a especificação de requisitos para tais sistemas. Uma ontologia para SAS é proposta, bem como um processo para orientar o uso da ontologia para elicitação e especificação de requisitos para SAS. A única ontologia core para requisitos para SAS na literatura não abrange todos os principais conceitos que SAS envolve, como as dimensões de modelagem e um feedback loop. Para atingir o objetivo, em primeiro lugar, foram realizadas duas revisões sistemáticas de literatura (RSLs) para analisar o trabalho que envolve a representação do conhecimento para SAS e sistemas sensíveis ao contexto. Um total de vinte e três estudos foram selecionados em ambos. Então, três linguagens de modelagem GORE para SAS foram analisadas - Tropos4AS, AdaptiveRML e Design Goal Model - para identificar os conceitos que essas linguagens podem representar. Observou-se que as linguagens analisadas não representam a maioria dos conceitos envolvidos no domínio SAS. Com os resultados das SLRs e da análise das linguagens de modelagem GORE, foi proposta uma ontologia para ajudar o engenheiro de requisitos a realizar a elicitação e a especificação de SAS. Para criar a ontologia, foram utilizadas três metodologias: Uschold e Gruninger, METHONTOLOGY e SABiO. A ontologia proposta abrange os principais conceitos de sistemas auto-adaptativos, como os conceitos de feedback loop, contexto, dimensões de modelagem para SAS e requisitos orientados a objetivos. Também foi proposto um processo para o uso da ontologia. A avaliação da ontologia foi baseada em seis critérios: abrangência, verificação, validação, utilidade, facilidade de uso e conformidade. Nossa ontologia é mais abrangente que os trabalhos relacionados selecionados em ambas RSLs. A ontologia foi verificada e validada através da instanciação de um sistema de notícias multimídia. A utilidade e facilidade de uso tanto da ontologia quanto do processo foram avaliadas por estudo de caso e um survey, onde os engenheiros de requisitos usaram o processo para instanciar um sistema de despacho de ambulância. Esta avaliação constatou que a ontologia é útil, embora o processo não seja tão fácil de usar. Outra pesquisa foi

respondida por especialistas em SAS para avaliar a conformidade da ontologia, os quais concordam com os conceitos da ontologia.

Palavras-chaves: Ontologia. Especificação. Elicitação. Sistemas auto-adaptativo. Engenharia de requisitos. Orientação a objetivos.

List of Figures

Figure 1 – MAPE-K.	30
Figure 2 – Context model.	34
Figure 3 – Requirements elicitation and analysis process.	36
Figure 4 – Design Goal Model Elements.	38
Figure 5 – Tropos4AS Elements.	38
Figure 6 – AdaptiveRML Elements.	39
Figure 7 – METHONTOLOGY states and activities.	43
Figure 8 – Procedure for a formal approach to ontology design and evaluation. . .	45
Figure 9 – SABiO's Processes.	47
Figure 10 – Number of studies per year.	52
Figure 11 – Ontology Languages adopted by the papers.	56
Figure 12 – Studies by RE activities.	59
Figure 13 – Application Domain for SAS studies.	61
Figure 14 – Amount of citations.	66
Figure 15 – Application Domain for CAS studies.	67
Figure 16 – Core Ontology for SAS.	68
Figure 17 – Goal model for the ZNN.com modeled using the Design Goal Model. .	70
Figure 18 – Goal model for the ZNN.com modeled using the Tropos4AS.	71
Figure 19 – Goal model for the ZNN.com modeled in AdaptiveRML.	72
Figure 20 – Goal Modeling Dimensions.	85
Figure 21 – Change Modeling Dimensions.	86
Figure 22 – Mechanism Modeling Dimensions.	86
Figure 23 – Effect Modeling Dimensions.	87
Figure 24 – MAPE-K adaptation cycle.	87
Figure 25 – First version of Onto4SASGORE.	88
Figure 26 – Context Elements.	89
Figure 27 – Core Onto4SASGORE in OntoUML.	97
Figure 28 – OntoUML model of the Onto4SASGORE Goal related elements. . . .	97
Figure 29 – OntoUML model of the Onto4SASGORE MAPE related elements. . .	98
Figure 30 – OntoUML model of the Onto4SASGORE Context related elements. . .	99
Figure 31 – OntoUML model of the Onto4SASGORE Modeling Dimensions related elements.	100
Figure 32 – The Onto4SASGORE process.	104
Figure 33 – Requirements Discovery Sub-process.	105
Figure 34 – SQWRL Tab in Protégé	130
Figure 35 – New SQWRL rule screen	131

Figure 36 – Result for CQ71	132
Figure 37 – Result for CQ74	132
Figure 38 – Graph with answers of affirmative 1.	150
Figure 39 – Graph with answers of affirmative 2.	150
Figure 40 – Graph with answers of affirmative 3.	151
Figure 41 – Graph with answers of affirmative 4.	151
Figure 42 – Graph with answers of affirmative 5.	152
Figure 43 – Graph with answers of affirmative 6.	153
Figure 44 – Graph with answers of affirmative 7.	153
Figure 45 – Graph with answers of affirmative 8.	154
Figure 46 – Graph with answers of affirmative 9.	154
Figure 47 – Graph with answers of affirmative 10.	155
Figure 48 – Graph with percentage to affirmative 1.	159
Figure 49 – Graph with percentage to affirmative 2.	160
Figure 50 – Graph with percentage to affirmative 3.	160
Figure 51 – Graph with percentage to affirmative 4.	160
Figure 52 – Graph with percentage to affirmative 5.	161
Figure 53 – Graph with percentage to affirmative 6.	161
Figure 54 – Graph with percentage to affirmative 7.	161
Figure 55 – Systematic Literature Review Phases.	175
Figure 56 – Selection process stages.	177
Figure 57 – Selection Process Stages.	184

List of Tables

Table 1 – Search String.	52
Table 2 – Studies identification.	53
Table 3 – Groups or organizations by studies.	55
Table 4 – Ontology Language by study.	56
Table 5 – Problems addressed by each study.	57
Table 6 – Requirements Engineering activities by studies.	60
Table 7 – Search String.	63
Table 8 – Studies and references.	64
Table 9 – Concepts by studies.	66
Table 10 – GORE Languages x Modeling Dimensions.	74
Table 11 – Tropo4AS x Core Ontology.	75
Table 12 – AdaptiveRML x Core Ontology.	76
Table 13 – Design Goal Model x Core Ontology.	77
Table 14 – Modeling Dimensions x Core Ontology.	78
Table 15 – Attributes of Onto4SASGORE	95
Table 15 – Attributes of Onto4SASGORE	96
Table 16 – Concepts and CQ related to activity 1.	107
Table 17 – Concepts and CQ related to activity 2.	107
Table 18 – Concepts and CQ related to activity 3.	108
Table 19 – Concepts and CQ related to activity 4.	109
Table 20 – Concepts and CQ related to activity 5.	110
Table 21 – Study S1 concepts vs Onto4SASGORE concepts	121
Table 22 – Study S2 concepts vs Onto4SASGORE concepts	122
Table 23 – Study S3 concepts vs Onto4SASGORE concepts	122
Table 24 – Study S4 concepts vs Onto4SASGORE concepts	123
Table 25 – Study S5 concepts vs Onto4SASGORE concepts	123
Table 26 – Study S6 concepts vs Onto4SASGORE concepts	124
Table 27 – Study S7 concepts vs Onto4SASGORE concepts	124
Table 28 – Study S8 concepts vs Onto4SASGORE concepts	125
Table 29 – Study S9 concepts vs Onto4SASGORE concepts	125
Table 30 – Study 10 concepts vs Onto4SASGORE concepts	126
Table 31 – Study 11 concepts vs Onto4SASGORE concepts	127
Table 32 – Study 12 concepts vs Onto4SASGORE concepts	128
Table 33 – Study 13 concepts vs Onto4SASGORE concepts	129
Table 34 – Results for perceived usefulness	153
Table 35 – Results for perceived ease of use	155

Table 36 – Search String.	176
Table 37 – Inclusion and Exclusion Criteria.	178
Table 38 – Quality Assessment.	179
Table 39 – Studies by quality score.	180
Table 40 – Data extraction form.	185
Table 41 – Onto4SASGORE Concepts and References	240
Table 41 – Onto4SASGORE Concepts and References	241
Table 41 – Onto4SASGORE Concepts and References	242
Table 41 – Onto4SASGORE Concepts and References	243
Table 41 – Onto4SASGORE Concepts and References	244
Table 41 – Onto4SASGORE Concepts and References	245

List of abbreviations and acronyms

CAS Context-Aware Systems

DAS Dynamic Adaptive System

GORE Goal-oriented Requirements Engineering

IBM International Business Machines

OWL Ontology Web Language

RE Requirement Engineering

SAS Self-adaptive System

SLR Systematic Literature Review

SQWRL Semantic Query-Enhanced Web Rule Language

SWRL Semantic Web Rule Language

TAM Technology Acceptance Model

W3C World Wide Web Consortium

List of symbols

\equiv	Concept equivalence
\sqsubseteq	Concept inclusion
\exists	Existential restriction
\forall	Universal restriction
\neg	Negation of concepts
\sqcup	Disjunction of concepts
\sqcap	Intersection of concepts

Contents

1	INTRODUCTION	20
1.1	Context	20
1.2	Problem Statement	23
1.3	Objectives	24
1.4	Overview of the Thesis	25
1.5	Research Methodology	26
1.6	Organization of the Thesis	28
2	BACKGROUND	29
2.1	Self-adaptive Systems	29
2.1.1	MAPE-K	30
2.1.2	Modelling Dimensions	31
2.1.3	Context	33
2.2	Requirements Engineering	34
2.2.1	Traditional Requirements Elicitation and Specification	34
2.2.2	GORE Specification for Self-adaptive Systems	36
2.3	Ontology	39
2.3.1	Reference Ontology	40
2.3.2	Operational Ontology	41
2.3.3	Methodologies to Create Ontologies	43
2.4	Conclusion	49
3	ANALYSIS OF THE STATE OF THE ART	51
3.1	Ontologies for SAS: A Systematic Literature Review	51
3.2	Ontologies for CAS: A Systematic Literature Review	63
3.3	Discussion of the SLRs	67
3.4	Analysis of Requirements Modeling Languages for SAS	68
3.4.1	GORE Languages versus Modeling Dimensions	72
3.4.2	GORE Languages versus the Core Ontology for SAS	74
3.5	Conclusion	78
4	ONTO4SASGORE: AN ONTOLOGY FOR SELF-ADAPTIVE SYS- TEMS GOAL-ORIENTED REQUIREMENTS	80
4.1	Purpose Identification and Requirements Elicitation	80
4.1.1	Objective	81
4.1.2	Scope	81

4.1.3	Motivating Scenarios	81
4.1.4	Informal Competence Questions	81
4.2	Knowledge Acquisition	84
4.3	Conceptualization	89
4.3.1	Concepts	90
4.3.2	Relationships	93
4.3.3	Attributes	94
4.3.4	OntoUML	96
4.4	Formalization	100
4.4.1	Competence questions	100
4.5	Implementation	101
4.6	Evaluation	101
4.7	Conclusion	102
5	THE ONTO4SASGORE PROCESS	103
5.1	Overview of the Onto4SASGORE Process	103
5.2	Goal-oriented Requirements Discovery Sub-process	104
5.2.1	Activity 1	106
5.2.2	Activity 2	107
5.2.3	Activity 3	108
5.2.4	Activity 4	108
5.2.5	Activity 5	109
5.3	Goal-oriented requirements classification and organization	112
5.4	Goal-oriented Requirements Prioritization and Negotiation	114
5.5	Goal-oriented requirements specification	115
5.6	Conclusion	118
6	EVALUATION	119
6.1	Comprehensiveness	119
6.2	Verification and Validation	130
6.3	Usefulness and Easy of Use	147
6.3.1	Results and Discussions	149
6.4	Accordance	156
6.4.1	Results and Discussions	156
6.5	Conclusion	162
7	CONCLUSION	163
7.1	Contributions	164
7.2	Limitations	166
7.3	Future Work	166

REFERENCES	168
APPENDIX A – ONTOLOGY FOR SAS: SLR PROTOCOL	175
APPENDIX B – ONTOLOGY FOR CAS: SLR PROTOCOL	182
APPENDIX C – COMPETENCE QUESTIONS	186
APPENDIX D – ONTO4SASGORE PROCESS WITHOUT PRO- TÉGÉ	196
APPENDIX E – ONTO4SASGORE PROCESS WITH PROTÉGÉ .	214
APPENDIX F – CLEANER ROBOT DESCRIPTION	238
APPENDIX G – WHERE DID THIS COME FROM?	240

1 INTRODUCTION

In this Chapter, we introduce the research presented in this thesis. In Section 1.1, we contextualize the research by presenting the description of the key concepts presented in this thesis. The problem statement is in Section 1.2 and the objectives of the work are in Section 1.3. The overview of this thesis is in Section 1.4 and the the adopted research methodology is in Section 1.5. The organization of this document is in Section 1.6.

1.1 Context

According to Brun et al. (2009), self-adaptivity is the ability of the system to adjust its own behavior in response to the environment. Self-adaptive systems are capable of dealing with a continuously changing environment and emerging requirements that may be unknown at design-time. The "self" prefix indicates that the system decides autonomously how to adapt or be organized to accommodate changes in their context and environment (BRUN et al., 2009). In other words, self-adaptive systems are context-aware and their adaptation interferes in the context, whereas the context contributes for their adaptation.

A Self-adaptive System (SAS) evaluates its own behavior and changes its performance when the evaluation indicates that it is not doing what it is intended to do, or when better functionality or performance is possible (SALEHIE; TAHVILDARI, 2009).

According to Andersson et al. (2009), changes are the cause of adaptation. When there is a change in the system's requirements on runtime or in the environment in which it is deployed, it can cause a system self-adaptation. The system reaction to change is the mechanism for self-adaptation and the impact of the system adaptation is the effect of this adaptation (ANDERSSON et al., 2009).

Context-aware systems (SCHILIT; ADAMS; WANT, 1994) adapt according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes in such things over time. A system with these capabilities can examine the computing environment and react to changes in the environment. Three important aspects of the context are: where you are, who you are with, and what resources are nearby (SCHILIT; ADAMS; WANT, 1994). Context-aware technology (JEONG-DONG; SON; BAIK, 2012) roots its core in the ability to conduct self-adjustment of conditions, or the ability to detect changes to conditions and to deliver relevant information to users. It is vital for the realization of this context-aware technology to be able to provide such environment and technologies that invisibly and automatically process the contextual information received from sensors (JEONG-DONG; SON; BAIK, 2012).

Feedback loops provide the generic mechanism for self-adaptation. A feedback loop typically involves four key activities (BRUN et al., 2009): collect, analyze, decide, and

act. The feedback cycle starts with the collection of relevant data from environmental sensors and other sources that reflect the current state of the system. The diagnosis then analyzes the data to infer trends and identify symptoms. The planning attempts to predict the future to decide on how to act on the executing system and its context through actuators or effectors (BRUN et al., 2009).

The software engineering for SAS community, through the first roadmap paper (CHENG et al., 2009b), has identified important challenges on SAS. One of the most important challenges refers to the points of variation that can be understood as modeling dimensions - that can represent a wide range of system properties. Andersson et al. (2009) defined a set of modeling dimensions for self-adaptive software systems. The identified dimensions were organized into four categories: the self-adaptive goals of the system, the causes or triggers of self-adaptation, the mechanisms used to adapt, and the effects of those mechanisms on the system.

The second roadmap paper, authored by Lemos et al. (2013), summarized the state-of-the-art and identified research challenges when developing, deploying, managing and evolving self-adaptive software systems. Specifically, they focused on development methods, techniques, and tools that are required when dealing with software intensive systems that are self-adaptive in their nature. According to Lemos et al. (2013), dynamic environments change the system's goals. As a consequence, it is necessary to propose means to fully comprehend the characteristics of self-adaptive systems and the key characteristics of their lifecycles to enhance design and modeling, optimization, and enactment of such systems.

Requirement Engineering (RE) is the process of discovering the purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. RE draws on the cognitive and social sciences to provide both theoretical grounding and practical techniques for eliciting and modelling requirements (NUSEIBEH; EASTERBROOK, 2000).

The requirements elicitation activity involves understanding the application domain, the specific problem to be solved, the organizational needs and constraints and the specific facilities required by the system stakeholders (DERMEVAL et al., 2015).

The requirements modeling or specification - the construction of abstract descriptions that are amenable to interpretation - is a fundamental activity in RE. The modeling notation and partial models produced are used as drivers to prompt further information gathering (NUSEIBEH; EASTERBROOK, 2000).

Different from traditional approaches (YANG et al., 2014), RE for SAS focuses more on defining adaptation logic, since SAS need adaptation mechanisms. Thus, during RE for SAS, engineers must address what changes in the environment and in the system to be monitored, what to adapt, when to adapt and how to adapt (YANG et al., 2014).

A Systematic Literature Review (SLR) performed by Yang et al. (2014) investigated

the modeling approaches, requirements engineering activities, requirements quality attributes, application domains and research topics for the self-adaptive systems domain. The results of this SLR depict that many goal-oriented modeling approaches, including KAOS (LAMSWEEERDE, 2003), i* (YU, 2009) and Tropos (BRESCIANI et al., 2004), are popular to model requirements for self-adaptive systems.

Goal-oriented models have proven to be effective in the specification of self-adaptive systems, monitoring and choosing between adaptive behaviors (MACÍAS-ESCRIVÁ et al., 2013).

In recent years, some Goal-oriented Requirements Engineering (GORE) works have been proposed to specify requirements for SAS (AHMAD et al., 2013). For instance, we can cite Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (DGM) (PIMENTEL et al., 2014). These are modeling methods proposed by experienced research groups in GORE and SAS fields. They involve the construction of a visual modeling language specifically for SAS, i.e. they are not general purpose goal modeling approaches, and they were developed for the SAS domain. These approaches are also investigated in this thesis.

Such approaches offer very important elements to specify self-adaptive systems requirements, but they do not guide the discovery of these requirements not even cover all SAS relevant concepts, like the modeling dimensions (SOARES et al., 2016). According to Nuseibeh and Easterbrook (2000), the choice of modeling approach affects the set of phenomena that can be modeled, and may even restrict what a requirements engineer is capable of observing.

Thus, a modeling approach should include all known SAS characteristics. We claim that an ontology for SAS could help to define such characteristics, because ontology is a standard way to represent concepts and relationships on a specific domain, i.e., it is a good way to formalize knowledge (SOUAG et al., 2015). An ontology can be used and shared within a specific domain and it has been used to support requirements engineering activities (DERMEVAL et al., 2015). A predefined ontology can make the work of requirements engineering much easier and faster, because ontologies also have been proved useful to support knowledge reusability (SOUAG et al., 2015). It has been used in requirements engineering to solve or minimize different kinds of problems. It is used, for example, to specify more complete and unambiguous requirements, to achieve consistent analysis on the requirements, to explicitly model the domain knowledge, to manage the requirements knowledge and the requirements changes, among other benefits (DERMEVAL et al., 2015).

1.2 Problem Statement

The goal-oriented modeling languages analyzed in this thesis, Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PIMENTEL et al., 2014), allow the goal-oriented specification of the requirements for SAS, but they do not present any guide to elicit these requirements. Although these languages were specifically created to model GORE requirements for SAS, they also do not allow the specification of some key concepts for SAS, like the modeling dimensions identified by Andersson et al. (2009) and all elements involved in an adaptation cycle like the proposed by Kephart and Chess (2003), which can generate an insufficient specification.

In order to reduce the negative effects of insufficient specifications, or requirements not completely defined, or yet, the dynamic and changing requirements, on the RE processes, ontologies can be used (CASTAÑEDA et al., 2010).

The results of the SLR about the applications of ontology in RE, performed by (DERMEVAL et al., 2015) show that the predominant RE activity supported by ontology is the specification. This result shows that ontologies may be used as means to formally specify requirements engineering activities. There is also an interest in the use of ontologies in the elicitation activity aiming to provide some guidance to realization of the requirements elicitation. For instance, by representing domain knowledge with the aid of ontologies in order to guide the elicitation activity.

The nature of requirements engineering involves capturing knowledge from diverse sources, including many stakeholders with their own interests and points of view. There are, therefore, many potential uses of ontologies in RE (CASTAÑEDA et al., 2010). It is well known that requirements elicitation and specification is decisive for gathering the needs and goals of the system's stakeholders. However, it is not a trivial task to elicit all information necessary for the development of self-adaptive systems. To support requirements elicitation for SAS, there is a need of an approach that takes full advantage of SAS-related domain knowledge.

There is an interest in the use of ontologies in the elicitation activity aiming to provide some guidance to perform requirements elicitation. For instance, by representing domain knowledge with the aid of ontologies in order to guide the elicitation activity (DERMEVAL et al., 2015). A domain ontology can be used as a way of facilitating the understanding between stakeholders. RE involves several activities to generate consistent and complete requirements representation and specification (CASTAÑEDA et al., 2010).

We advocate the ontology for self-adaptive systems need to cover the change, mechanisms and effects dimensions defined by Andersson et al. (2009). These concepts are essential for system self-adaptation and, therefore, they need to be represented in an ontology to support the requirements activities of SAS. So, to support the SAS specifications, it is necessary to represent an adaptation cycle and the modeling dimensions (ANDERSSON

et al., 2009) in an ontology for SAS.

The work presented by Qureshi, Jureta and Perini (2011) proposes a core ontology for SAS. This core ontology is formed by the following concepts: Goal, Softgoal, Quality Constraint, Task, Evaluation, Context, Resource and Domain Assumption. It presents the context and resource concepts but it does not present all the key concepts related to the requirements specification for self-adaptive systems. For example, it does not contemplate the modeling dimensions proposed by Andersson et al. (2009) neither an adaptation cycle (KEPHART; CHESS, 2003) which is responsible for controlling the execution of the adaptation and it has activities like monitor, analyze, plan and execute.

The modeling dimensions for SAS deals with adaptation and in the best of our knowledge, they are the only source in the literature that defines the dimensions necessary to model SAS. Each modeling dimension describes a particular aspect of the system that is relevant for self-adjustment. They allow that key aspects of different self-adaptive systems to be easily identified. They are necessary that the ontology for SAS be able to represent these modeling dimensions.

For the above, there is a need for more complete ontology for SAS. It is necessary to define a core ontology, contemplating the modeling dimensions as well as supporting an adaptation cycle, and define a domain ontology with all concepts to represent the context, the modeling dimensions and an adaptation cycle in order to specify more complete GORE requirements for SAS. It also is necessary to propose a process to use the ontology, otherwise it would be hard to elicit and specify gore requirements for SAS.

1.3 Objectives

In this context, we believe that it is necessary to develop an ontology to aid the goal-oriented requirements activities related to elicitation and specification of self-adaptive systems. Based on this, the main objective of this thesis is:

Research objective *To support the requirements elicitation and specification activities for SAS.*

From the objective, it was stated the research question investigated by this thesis:

Research question *What are the essential concepts and relationships required to specify GORE requirements for self-adaptive systems?*

To achieve this goal, it is necessary to understand the self-adaptive systems domain by identifying the concepts required to represent SASs. The work presented in this thesis intends to bring contributions for the area of requirements engineering for self-adaptive systems. So, the specific objectives of this work are:

- Analyze the existing ontologies related to some characteristics of SAS;
- Analyze the existing ontologies related to some characteristics of context-awareness;
- Systematically organize the knowledge of SAS by analyzing some goal-oriented requirements languages proposed to specify self-adaptive systems;
- Define a core ontology with the main concepts to be considered for the self-adaptive systems domain by using as base the core ontology for SAS (QURESHI; JURETA; PERINI, 2011), adding the *change*, *mechanism*, *effect* and *adaptation cycle* concepts and expanding the *context* concept. The first version of this core ontology is in Soares et al. (2016);
- Define a domain ontology for self-adaptive systems, with concepts, relationships, axioms and competence questions;
- Define a process to use the ontology with a GORE elicitation guide and a specification template;
- Evaluate the ontology and the process.

1.4 Overview of the Thesis

In order to achieve our objective, we analyzed three goal-oriented modeling approaches that involve a modeling language proposed specifically to specify SAS. However, we noticed that these approaches do not have all the concepts necessary to specify the goal-oriented requirements of SAS. Then, we performed two SLR to analyze the works in the literature that deal with knowledge representation, SAS and context. From this, we built the Onto4SASGORE, an ontology for goal-oriented requirements elicitation and specification for SAS. It was built according to the steps proposed by (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997), (FALBO, 2014) and Uschold and Gruninger (1996). We also present a process to use the ontology. Its main objective is to support requirements engineers to discover GORE requirements during the elicitation activity in a development process of SAS, as well as to specify the elicited requirements. In order to create the Onto4SASGORE, we used concepts of the core ontology for SAS proposed by Qureshi, Jureta and Perini (2011), for being, to the best of our knowledge, the only formal core ontology for requirements for SAS and present some concepts that have to be considered. Our proposal also have reused knowledge from the CA_{5W1H}Onto which uses the 5W1H approach to propose an ontology for context (JEONG-DONG; SON; BAIK, 2012).

The Onto4SASGORE also presents concepts about the modeling dimensions for SAS defined by (ANDERSSON et al., 2009) that can represent a wide range of SAS properties and, to represent the adaptation cycle, we use MAPE-K (KEPHART; CHESS, 2003).

It consists of a monitor, an analyzer, a planner, and an executor, that share a common knowledge base (BRUN et al., 2009). Those are the works that found our ontology, but we have also used the 23 works selected in two SLR and the three goal-oriented modeling languages elements. The modeling languages are the Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and the Design Goal Model (PIMENTEL et al., 2014).

The Onto4SASGORE process, a process to use the Onto4SASGORE ontology, uses the ontology and a SAS description as input and it produces a GORE requirements specification as output. The Onto4SASGORE process is composed of a sub-process and three more activities. The sub-process is a goal-oriented requirements discovery process and the activities are goal-oriented requirements classification and organization, goal-oriented requirements prioritization and negotiation and goal-oriented requirements specification. This process aims to assist the requirements engineer in performing activities such as GORE requirements elicitation and specification when in the self-adaptive domain. It offers a guide for GORE requirements elicitation and a GORE requirements specification template.

1.5 Research Methodology

To investigate the phenomenon in the literature in order to know how the researchers have tried the subject, we conducted two systematic literature reviews. An analysis of existing goal oriented modeling languages was performed in order to know the concepts that goal oriented modeling languages, have created specifically for self-adaptive systems. The ontology was proposed following some existing methodologies for the creation of ontologies. We also define a process to allow the use of the ontology by requirements engineer to elicit and specify goal requirements for SAS. To evaluate the ontology, we used six criteria: comprehensiveness, accordance, verification, validity, usefulness and ease of use. This work followed eight steps:

- Step 1: Analyzing the existing goal modeling languages for self-adaptive systems. First, we analyzed the languages proposed to specify goals of self-adaptive systems. We performed a comparative study in order to evaluate whether the goal modeling languages proposed to model SAS (Tropos4AS, Adaptive RML and Design Goal Model) cover the modeling dimensions for SAS proposed by Andersson et al. (2009). Thereby, we get the concepts represented by each goal modeling language and the modeling dimensions.
- Step 2: Performing a systematic literature review about ontologies and SAS. We investigated studies about ontologies and self-adaptive systems to identify and understand the ontologies proposed for the self-adaptive systems domain. We have

selected 13 studies that propose ontologies to the domain of self-adaptive systems. Note that some studies present concepts and relationships for just one characteristic of self-adaptive systems. Others, like the core ontology to requirements engineering for SAS by Qureshi, Jureta and Perini (2011), do not present concepts to cover all the modeling dimensions by Andersson et al. (2009) neither an adaptation cycle.

- Step 3: Performing a systematic literature review about ontologies and Context-Aware Systems (CAS). We investigated studies about ontologies and context-aware systems to identify and understand the research about context and knowledge representation. We have selected 10 studies that propose ontologies to the context domain. This SLR present works for one characteristic of self-adaptive systems, the context awareness.
- Step 4: Building the core ontology. We build a core ontology based on the core ontology for SAS requirements proposed by Qureshi, Jureta and Perini (2011) with more general concepts related to SAS. The new elements came from the analysis of the CA_{5W1H}Onto (JEONG-DONG; SON; BAIK, 2012) as well as the modeling dimensions for SAS identified by Andersson et al. (2009). As a result, we added more context information, adaptation, changes and mechanisms concepts in the core ontology.
- Step 5: Building a domain ontology. With the results of our SLRs, the analysis obtained from step 1, the core ontology built in step 4 and the MAPE-K (to adaptation control) proposed by (KEPHART; CHESS, 2003), we built a domain ontology in order to aid the goal requirements activities for SAS. The ontology presents the concepts of modeling dimensions, context, adaptation control, requirements and goal modeling.
- Step 6: Defining a process. We defined the steps of a process to use the ontology. The process offers a goal-oriented requirements elicitation guidance and a specification template. The input of the process is the ontology and a SAS description.
- Step 7: Evaluation. We used six criteria to evaluate our work. They are: comprehensiveness, accordance, verification, validity, usefulness and ease of use. The studies selected in the SLRs were used to evaluate the comprehensiveness criteria. The usability criteria was evaluated by using a case study with requirements engineers. The usefulness and ease of use was evaluated through a survey with requirements engineers. A survey also was used to evaluate the accordance criteria, with adaptive specialists. The verification criteria was evaluated by answering the competence questions of the ontology with its own elements (concepts, relationships and axioms). To evaluate the validity criteria, we instantiate the ontology with a real world system.

1.6 Organization of the Thesis

The remain of this thesis is organized as following:

- Section 2 - Background. In this Section, we review the concepts used to found this work, requirements engineering, self-adaptive systems and ontology.
- Section 3 - Analysis of the State of the Art. In this Section, we present the two SLRs performed and an analysis of the goal oriented modeling languages for SAS.
- Section 4 - Onto4SASGORE. In this Section, we present an ontology to aid the GORE requirements elicitation and specification for self-adaptive systems.
- Section 5 - Process. In this Section, we present the process of using the Onto4SASGORE ontology.
- Section 6 - Evaluation. In this Section, we present the comprehensiveness, verification, validity, accordance, usefulness and ease of use evaluations of our ontology.
- Section 7 - Conclusion. In this Section, we conclude our work, stating the contributions and limitations of the research and describe future works.

2 BACKGROUND

In this Chapter, we present the theoretical background of the research. So, we focus on self-adaptive systems, requirements engineering, requirements engineering for self-adaptive systems and ontology. In Section 2.1, we explain about adaptation cycle, modeling dimensions and context, vital subjects for SAS. In Section 2.2, we present the research about requirements engineering and GORE for SAS. While in Section 2.3, we explain about ontology. Section 2.4 concludes this Section.

2.1 Self-adaptive Systems

To deal with the increasing complexity of software systems and uncertainty of their environments, software engineers have turned to self-adaptivity. Self-adaptive systems are capable of dealing with a continuously changing environment and emerging requirements that may be unknown at design-time. The “self” prefix indicates that the systems decide autonomously (i.e., without or with minimal interference) how to adapt or organize to accommodate changes in their contexts and environments (BRUN et al., 2009).

Self-adaptive systems are highly context dependent. Whenever the system’s context changes the system has to decide whether it needs to adapt. Whenever the system decides to adapt, this may prompt the need for verification activities to provide continual assessment. Moreover, depending on the dynamics of change, verification activities may have to be embedded in the adaptation mechanism (CHENG et al., 2009b). Self-adaptive systems may change their state quickly to respond to context or property changes (CHENG et al., 2009b).

The generic mechanism for self-adaptation is provided by feedback loops (BRUN et al., 2009). A feedback loop typically involves four key activities: collect, analyze, decide, and act. The feedback cycle starts with the collection of relevant data from environmental sensors and other sources that reflect the current state of the system. The diagnosis then analyzes the data to infer trends and identify symptoms. The planning attempts to predict the future to decide on how to act on the executing system and its context through actuators or effectors (BRUN et al., 2009).

According to Lemos et al. (2011), we need to establish the foundations that enable the systematic development, deployment, management and evolution of future generations of self-adaptive software systems.

There is the challenge of defining models that can represent a wide range of system properties (LEMOS et al., 2011). So, Andersson et al. (2009) define the modeling dimensions.

Three fundamental aspects to self-adaptive systems are: the feedback loop, i.e. the adaptation cycle, that the systems work on it; the context information, that is one of the power supplies of the adaptation cycle, allowing the systems to know the environment and know themselves; and the points of variations that are the particular characteristics of the systems, captured as modeling dimensions.

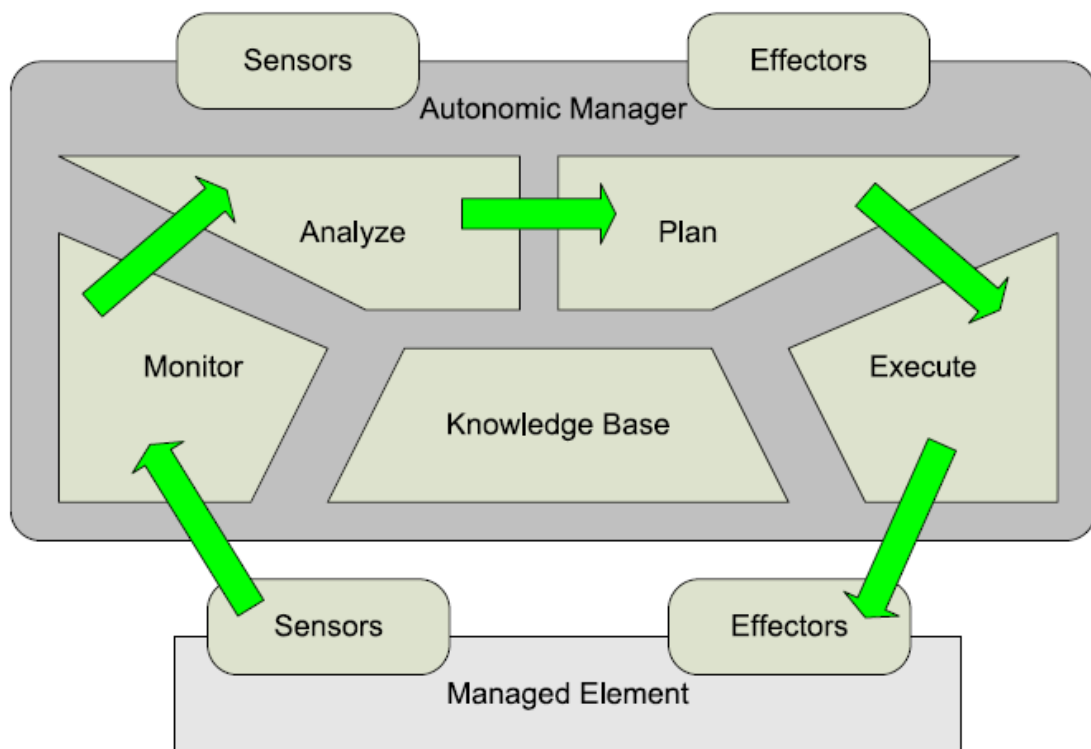
In the next sections, we present each of them in detail.

2.1.1 MAPE-K

Adaptation control can be performed by a simple sequence of four activities: monitor, analyze, plan, and execute (MAPE). Together, these activities form a feedback control system from control theory (LEMOS et al., 2013).

The International Business Machines (IBM) has developed a technique called MAPE-K (KEPHART; CHESS, 2003), this was used to create several other tools used in the development of autonomous systems. Figure 1 shows the MAPE-K cycle.

Figure 1 – MAPE-K.



Source: Brun et al. (2009)

The MAPE-K activities are listed below:

Monitor

- The monitor function collects the details from the managed resources, via touch-points, and correlates them into symptoms that can be analyzed. The details can include topology information, metrics, configuration property settings and so on.
- The monitor function aggregates, correlates and filters these details until it determines a symptom that needs to be analyzed.

Analyze

- Autonomic managers must be able to perform complex data analysis and reasoning on the symptoms provided by the monitor function.
- Influenced by stored knowledge data.
- The analyze function provides the mechanisms to observe and analyze situations to determine if some change needs to be made.

Plan

- Structures the actions needed to achieve goals and objectives.
- The plan function creates or selects a procedure to enact a desired alteration in the managed resource.
- The plan function can take on many forms, ranging from a single command to a complex workflow.

Execute

- Changes the behavior of the managed resource using effectors, based on the actions recommended by the plan function.

Knowledge Base

- Standard data shared among the monitor, analyze, plan and execute functions
- The shared knowledge includes data such as topology information, historical logs, metrics, symptoms and policies.
- The autonomic manager itself creates the knowledge.

2.1.2 Modelling Dimensions

The common point in self-adaptive systems is that design decisions are moved towards runtime to control dynamic behavior and that an individual system reasons about its state and environment (BRUN et al., 2009).

Cheng et al. (2009b) advocate that there is a lack of consensus among researchers and practitioners on the points of variation among SAS, so they refer to these points of variations as modelling dimensions. Each dimension describes a particular aspect of the system that is relevant for self-adaptation. The dimensions are presented by (ANDERSSON et al., 2009) in term of four groups: Goals, Change, Mechanisms and Effects.

Goals - Goals are objectives the system under consideration should achieve. The dimensions related to goals are: evolution, flexibility, duration, multiplicity and dependency. The evolution dimension defines the system's capability to manage and create new goals during its lifetime. The flexibility is related to the level of uncertainty associated with the goal specification, which may range over three values: rigid, constrained, and unconstrained. The duration dimension is concerned with the validity of a goal throughout the system's life-time. It may range from temporary to persistent. The multiplicity dimension is related with the number of goals associated with the self-adaptability aspects of a system. A system can have a single goal or multiple goals. Finally, the dependency dimension, in case a system has multiple goals, captures how the goals are related to each other. The dependency dimension can be either independent or dependent.

Change - Changes are the cause of adaptation. Whenever the system's context changes, the system has to decide whether it needs to adapt. The dimensions related to change are: source, type, frequency, anticipation. The source dimension identifies the origin of the change, which can be either external or internal to the system. Internal change might be in: application, middleware or infrastructure. The type dimension refers to the nature of change. It can be functional, non-functional, and technological. The frequency is concerned with how often a particular change occurs, and it can range from rare to frequent. Anticipation dimension captures whether change can be predicted ahead of time. It can be foreseen (taken care of), foreseeable (planned for), and unforeseen (not planned for).

Mechanisms - This set of dimensions captures the system's reactions towards change, which means that they are related to the adaptation process itself. The dimensions of mechanisms are: type, autonomy, organization, scope, duration, timeliness and triggering. The type dimension captures whether adaptation is related to the parameters of the system's components or to the structure of the system. The adaptation can be parametric or structural, or a combination of these. The autonomy identifies the degree of outside intervention during adaptation. The range of this dimension goes from autonomous to assisted. The organization dimension captures whether adaptation is performed by a single component (centralized), or distributed amongst several components (decentralized). The scope dimension identifies whether adaptation is localized or involves the entire system. It can range from local to global. The duration dimension refers to the period of time in which the system is self-adapting, it can be for short (seconds to hours), medium (hours to months), or long (months to years) term. The timeliness dimension captures whether the time period for performing self-adaptation can be guaranteed, and it ranges from best-

effort to guaranteed. For last, the triggering identifies whether the change that initiates adaptation is event-trigger or time-trigger.

Effects - This set of dimensions captures what is the impact of adaptation upon the system. The effects dimensions are: criticality, predictability, overhead and resilience. The criticality dimension captures the impact upon the system in case the self-adaptation fails. The range of values associated is harmless, mission-critical, and safety-critical. The predictability dimension identifies whether the consequences of self-adaptation can be predictable in both value and time. Since predictability is associated with guarantees, the degree of predictability can range from non-deterministic to deterministic. In overhead dimension is captured the negative impact of system adaptation upon the system's performance. The overhead can range from insignificant to system failure. The resilience dimension is related to the persistence of service delivery that can justifiably be trusted, when facing changes. The degree of resilience can range from resilient to vulnerable.

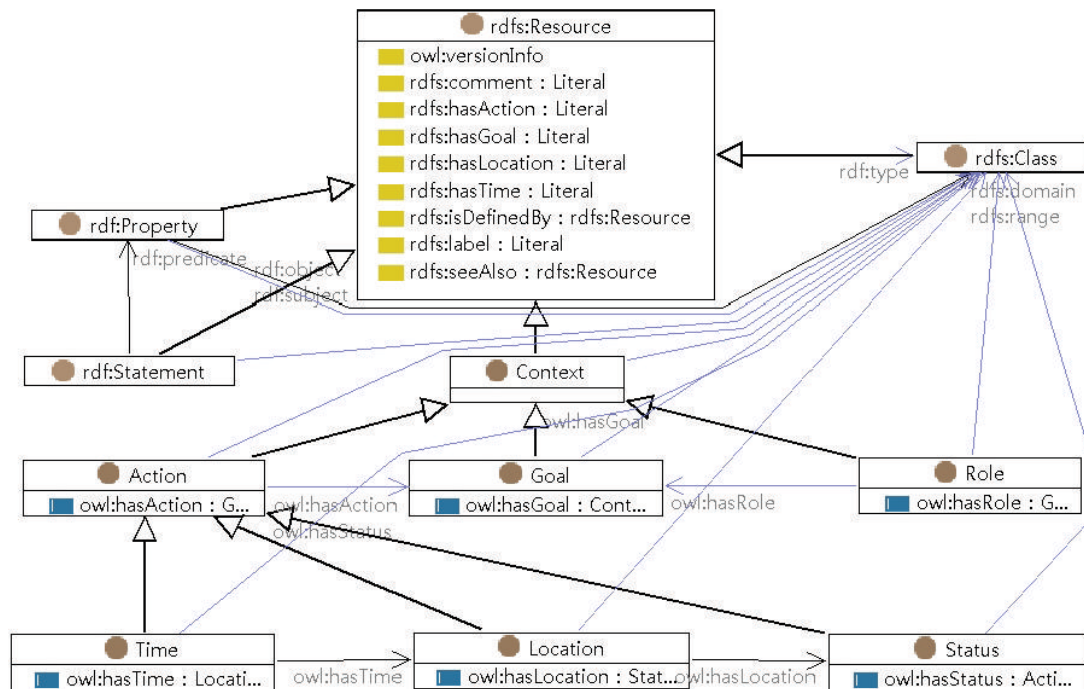
2.1.3 Context

Context-aware systems adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes to the environment. Three important aspects of context are: where you are, who you are with, and what resources are nearby (SCHILIT; ADAMS; WANT, 1994). Context-aware technology roots its core in the ability to conduct self-adjustment of conditions, or the ability to detect changes to conditions and to deliver relevant information to users. It is vital to realization of this context-aware technology to be able to provide such environment and technologies that invisibly and automatically process the contextual information received from sensors (JEONG-DONG; SON; BAIK, 2012).

Context is a very important aspect of self-adaptive systems. Self-adaptive systems take into account the context in its self evaluation to decide and perform the adaptation.

The work presented in Jeong-Dong, Son and Baik (2012) introduces an ontology for context which comprises the 5W1H maxim, i.e. the why, who, what, where, when, and how attributes. Figure 2 presents a class diagram of the ontology for context. A context is a resource and it has an action, a goal and a role. An action has a time, a location, and a status.

Figure 2 – Context model.



Source: Jeong-Dong, Son and Baik (2012)

2.2 Requirements Engineering

2.2.1 Traditional Requirements Elicitation and Specification

One of the clearest definitions of RE, according to Nuseibeh and Easterbrook (2000) is provided by Zave (1997):

"Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

Nuseibeh and Easterbrook (2000) advocate that RE needs to be sensitive to how people perceive and understand the world around them, how they interact, and how the sociology of the workplace affects their actions. RE draws on the cognitive and social sciences to provide both theoretical grounding and practical techniques for eliciting and modelling requirements:

- Cognitive psychology provides an understanding of the difficulties people may have in describing their needs.
- Anthropology provides a methodological approach to observing human activities that helps to develop a richer understanding of how computer systems may help or hinder those activities.

- Sociology provides an understanding of the political and cultural changes caused by computerization.
- Linguistics is important because RE is largely about communication. Tools from linguistics can also be used in requirements elicitation, for instance to analyze communication patterns within an organization.

RE is concerned with interpreting and understanding stakeholder terminology, concepts, viewpoints and goals. RE is concerned with an understanding of beliefs of stakeholders (epistemology), with the question of what is observable in the world (phenomenology), and with the question of what can be agreed on as objectively true (ontology) (NUSEIBEH; EASTERBROOK, 2000). Hence, it is important include the modelling technique, because it affects the set of phenomena that can be modelled, and may even restrict what a requirements engineer is capable of observing (NUSEIBEH; EASTERBROOK, 2000). The core RE activities are:

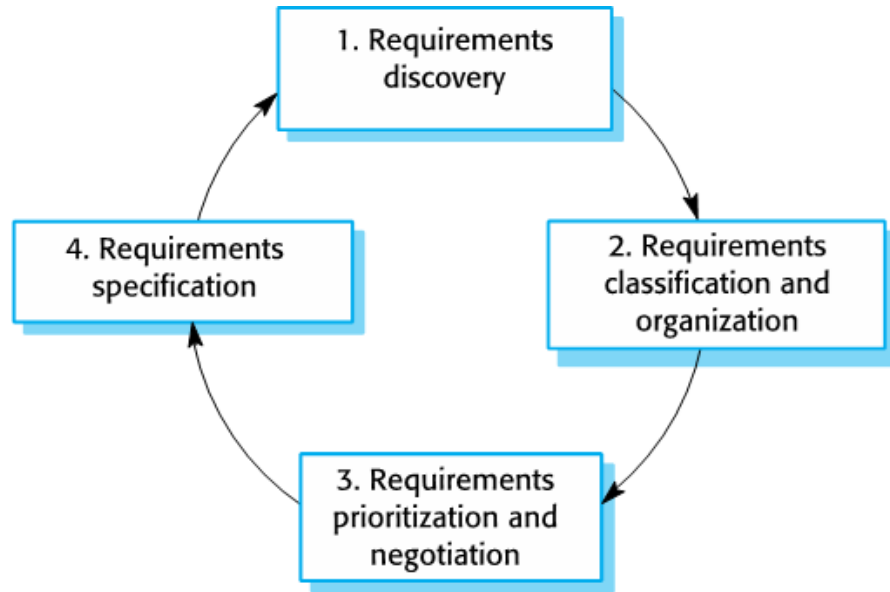
- Eliciting requirements,
- Modelling and analyzing requirements,
- Communicating requirements,
- Agreeing requirements, and
- Evolving requirements.

Sommerville (2010) presents a requirements elicitation and analysis process. It includes: Requirements discovery, Requirements classification and organization, Requirements prioritization and negotiation, Requirements specification as pictured in Figure 3. It is an iterative process that can be represented as a spiral of activities.

The Process Activities are described by Sommerville (2010) as:

- Requirements discovery: Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organization: Groups related requirements and organizes them into coherent clusters.
- Prioritization and negotiation: Prioritizing requirements and resolving requirements conflicts.
- Requirements specification: Requirements are documented and input into the next round of the spiral.

Figure 3 – Requirements elicitation and analysis process.



Source: Sommerville (2010).

The requirements elicitation and analysis involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints. It may involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called stakeholders (SOMMERVILLE, 2010).

Every kind of system obtains benefits from the traditional RE, but some needs some adaptation for the RE be more effective. For example, Ossada et al. (2012) propose a guide to direct the software engineers to capture and define the requirements for embedded systems.

For SAS, a number of requirements modeling languages have been proposed.

2.2.2 GORE Specification for Self-adaptive Systems

Different from traditional RE, RE for SAS focuses more on defining adaptation logic, since SAS need adaptation mechanisms. Thus, during RE for SAS, engineers must address what changes in the environment and the system themselves to be monitored, what to adapt, when to adapt and how to adapt, taking both domain logic and adaptation logic into account (YANG et al., 2014).

The results of the SLR performed by Yang et al. (2014) indicate that GORE approaches are very popular to model requirements for SAS. In this thesis, we chose three GORE approaches created specifically to model self-adaptive systems: the Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), the Adaptive RML (QURESHI; JURETA; PERINI, 2012) and the Design Goal Model (PIMENTEL, 2015).

Note that, there are more goal-oriented modeling languages to specify goal requirements for SAS, such as the languages proposed by Goldsby et al. (2008), LoREM (the Levels of RE for Modeling), FLAGS (Fuzzy Live Adaptive Goals for Self-adaptive systems) proposed by Baresi, Pasquale and Spoletini (2010), and RELAX (WHITTLE et al., 2010). FLAGS and LoREM are extensions of general-purpose modeling languages. LoREM allows to model the requirements of a Dynamic Adaptive System (DAS) using i^* goal models, and the FLAGS generalizes the KAOS model, to add adaptive goals to embed adaptation countermeasures, and fosters self-adaptation by considering requirements as live, runtime entities. However, in this thesis we chose the Tropos4AS, Design Goal Model (DGM) and AdaptiveRML because besides being GORE modeling, they were created specifically for SAS domain and present a visual modeling language.

The work presented by Whittle et al. (2010) is also a goal-based requirements model that explicitly captures where adaptations are needed, documents the level of flexibility supported during adaptation, and takes into account environmental uncertainty. RELAX is a textual language for dealing with uncertainty in DAS requirements which allows requirements to be temporarily relaxed if necessary to support adaptation (CHENG et al., 2009a).

The RELAX has modal, temporal and ordinal operators, and uncertainty factors. The modal operators are: SHALL; MAY...OR (behavior). The temporal operators are: EVENTUALLY; UNTIL; BEFORE, AFTER; IN; AS EARLY, LATE AS POSSIBLE (event); AS CLOSE AS POSSIBLE TO [frequency]. The ordinal operators are: AS CLOSE AS POSSIBLE TO [quantity]; AS MANY, FEW AS POSSIBLE. The uncertainty factors are: ENV (environment); MON (properties monitored); REL (relationship); DEP (dependencies) (WHITTLE et al., 2010). As RELAX is a textual language, we consider it can be used to support any goal oriented modeling language for SAS.

Design Goal Model

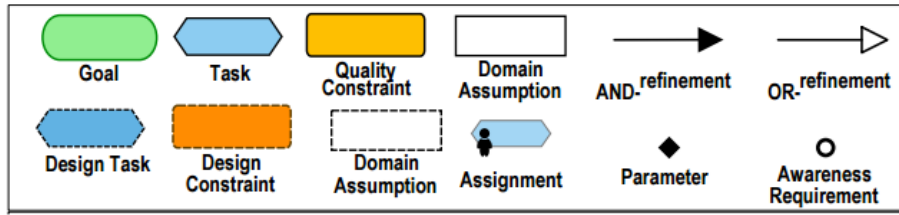
Pimentel et al. (2014) propose an extended form of goal model, named Design Goal Model (DGM) where architects can incrementally refine the original requirements by considering behavioral alternatives. Through the definition of alternative refinements for the same design element, this model captures the inherent variability of the design space, where different solutions for a given problem can be explored.

In DGM, requirements can be represented as goals, tasks, domain assumptions (DAs) and quality constraints (QCs) as illustrated in Figure 4. The space of alternatives for goal satisfaction is represented by Boolean AND/OR refinements relationships with obvious semantics. Besides these elements, DGM can represent concepts like parameters, assignment and awareness requirements (SOUZA et al., 2011).

Tropos4AS

The objective of Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008) is to define a process and a tool-supported design framework to develop self-adaptive systems. This

Figure 4 – Design Goal Model Elements.



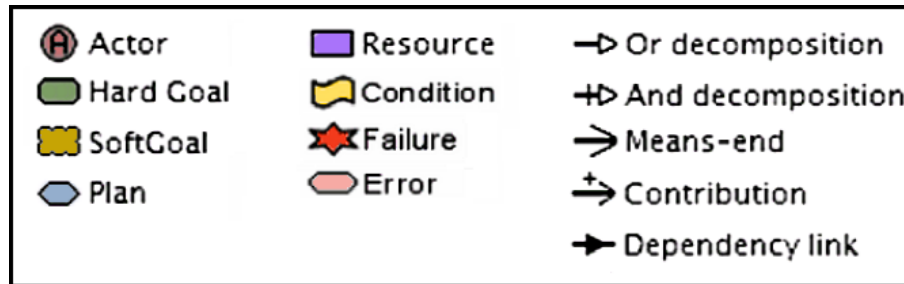
Source: Pimentel (2015).

approach, founded on the Tropos methodology (BRESCHIANI et al., 2004), offers support to the designer to capture, specify and detail characteristics specific to self-adaptive systems.

Tropos4AS proposes to complement the Tropos agent goal model with the possibility to model undesirable (faulty) states, which are known to the designer for possibly leading to system failure. The Tropos4AS supports the detailed specification of goals, the relationship with the environment the system will be deployed in, and taking into account possible failures and corresponding recovery actions.

Tropos4AS can capture the following concepts: actor, goal, softgoal, plan, resource, condition, failure, error, and relationships like AND/OR decomposition, means-end, contribution and dependency link. Figure 5 presents the Tropos4AS elements.

Figure 5 – Tropos4AS Elements.



Source: Own.

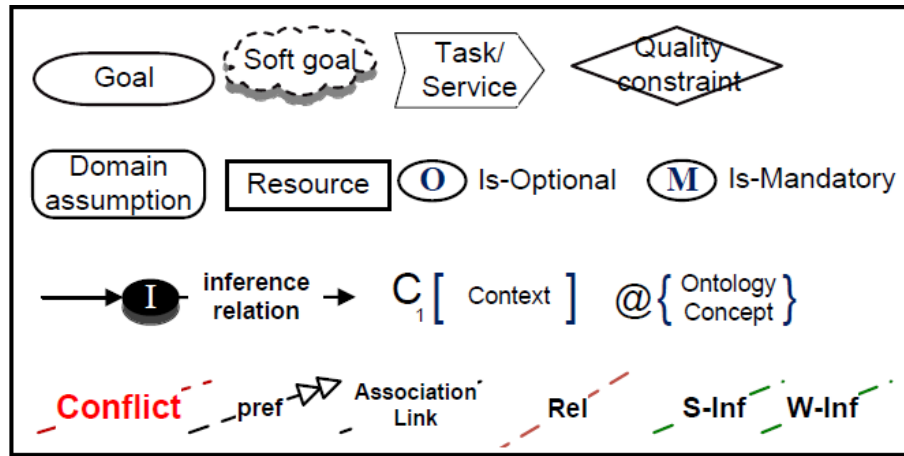
AdaptiveRML

The AdaptiveRML is a modeling approach for the representation of early requirements for SAS (QURESHI; JURETA; PERINI, 2012). It uses a core ontology for RE (JURETA; MYLOPOULOS; FAULKNER, 2008) to extend the goal-oriented perspective allowing to model optional requirements, preferences, and to treat non-functional requirements in terms of approximations and quality constraints.

This approach is built on *Techne* (JURETA et al., 2010) by adding two new concepts, namely, context and resource, and two relations, i.e. relegation and influence. The *Techne* basic elements models are requirements, modeled as natural language propositions that are labeled as domain assumptions, goals, quality constraints, or tasks.

AdaptiveRML represents concepts like goal, softgoal, task, service, quality constraint, domain assumption, resource, optional, mandatory, context, ontology concept and relationships like conflict, preference, association link, relegation, S-inf, W-inf and inference relation as viewed in Figure 6. The influence relation is added between a set of requirements or task, where the achievement of the former has strong influence (S-inf) or weak influence (W-inf) to the achievement of others. The relegation relation is used between tasks that have conflicting context conditions, resource availabilities or quality criteria, so that a task can relegate another with the aim to avoid failure in achieving a critical requirement.

Figure 6 – AdaptiveRML Elements.



Source: Own.

2.3 Ontology

Ontology can be defined as a specification of a conceptualization (GUBER, 1993). More precisely, ontology is an explicit formal specification of how to represent entities that exist in a given domain of interest and the relationships that hold among them. In general, for an ontology to be useful, it must represent a shared, agreed upon conceptualization (CASTAÑEDA et al., 2010).

Ontologies provide a skeletal structure for a knowledge base. An ontology allows that two systems built with their knowledge based on a common ontology, share a common structure, making easier to merge and share the knowledge bases. The use of ontologies suggests a possible approach to building shareable knowledge bases (SWARTOUT et al., 1996).

Ontology is a standard way to represent concepts and relationships of a specific domain, i.e., it is a good way to formalize knowledge (SOUAG et al., 2015). An ontology can be used and shared within a specific domain, and it has been used to support requirements engineering (RE) activities (DERMEVAL et al., 2015). A predefined ontology can

make the work of RE much easier and faster (SOUAG et al., 2015). Ontologies described in formal languages can be used by different applications to share and store knowledge (KEPLER et al., 2006).

There are some kinds of ontologies. The core ontology is a mid-term ontology that is not as specific as a domain ontology but also not so domain-independent as a foundational ontology (DUARTE et al., 2016). A meta-ontology describing a set of real-world categories allowing to be used to talk about reality is a foundational ontology. The axiomatization of this meta-ontology must represent the laws that define that nomological world space of reality. This meta-ontology is constructed using the theories developed by formal ontology in philosophy (GUIZZARDI, 2007). Examples of foundational ontologies is the DOLCE (MASOLO et al., 2003) and UFO Guizzardi (2007).

Domain ontologies are developed based on foundational ontologies. Concepts and relations in a domain ontology are previously analyzed according of a foundational ontology. An ontological analysis provides a sound foundation for modeling concept, if assumed that such concepts are aimed at representing reality (FALBO, 2014). Reference and Operational ontologies are domain ontologies.

2.3.1 Reference Ontology

Reference ontology is a domain ontology with the sole objective of making the best possible description of the domain in reality, regarding a certain level of granularity and viewpoint. It is a solution-independent specification (conceptual model) with the aim of making a clear and precise description of domain entities for communication, learning and problem-solving (FALBO, 2014). One example of language to create a conceptual model of the reference ontology is the OntoUML. OntoUML is a UML class diagram profile that incorporates important foundational distinctions made by the Unified Foundational Ontology (UFO).

OntoUML has the following constructs: Kind, Subkind, Phase, Role, Category, RoleMixin, and Mixin. This language also is based on Unified Modeling Language (UML), and this is proposed by Guizzardi (2005).

The UFO foundational ontology provides ontological foundations for the philosophical categories of *universals* and *individuals*. *Universals* are represented in conceptual modeling by using the constructs of *Types (classes, classifiers)* and *Individuals* are represented by the constructs of the *Types instances*. The instances of *universals* are *substantials*, that is similar to what is termed *object*. The follow, it is the distinction of the categories specializing substantial universal by Guizzardi (2005).

Kinds and Subkinds - Kind represents a substance sortal that supplies a principle of identity for its instances. Kinds can be specialized in other rigid subtypes that inherit their supplied principle of identity named subkinds. In general, the Subkind can be omitted in

conceptual specifications without loss of clarity. Every object in a conceptual specification using this profile must be an instance of a Kind, directly or indirectly.

Phases - Phase in this profile represent the phased-sortals phase.

Roles - Role represent the phased-sortals role. Roles and Phases are anti-rigid universals and cannot appear in a conceptual model as a supertype of a Kind. However, sometimes subtyping is wrongly used in conceptual modeling to represent alternative allowed types that can fulfill a role.

Mixins - Conceptual modeling types classified as Mixins represent the dispersive universals. Mixin types are perceived to be of great importance in structuring the specification of conceptual models. They can represent top-most types such as Thing, Entity, Element. We use the stereotype «category» to represent a rigid mixin that subsumes different kinds. In contrast, some mixins are anti-rigid and represent abstractions of common properties of roles. These types are stereotyped as «roleMixin» and represent dependent anti-rigid non-sortals. We use the stereotype « mixin » to represent non-rigid non-sortals. Mixins cannot appear in a conceptual model as subclasses of kinds, phases or roles. Moreover, rigid mixins (categories) cannot be subsumed by anti-rigid ones, i.e., by role mixins. A mixin must always be depicted as an abstract class in a UML conceptual specification.

Once users have already agreed on a common conceptualization, operational versions (machine-readable ontologies) of a reference ontology can be implemented (FALBO, 2014).

2.3.2 Operational Ontology

Contrary to reference ontologies, operational ontologies are designed with the focus on guaranteeing desirable computational properties (FALBO, 2014). One example of language to implement operational ontology is the Ontology Web Language (OWL). OWL¹ is a Semantic Web language designed to represent rich and complex knowledge about individual or groups of objects, and relations between them. It is a computational logic-based language and it can be exploited by computer programs to verify the consistency of that knowledge or to make implicit knowledge explicit.

OWL is based on description logics since 2004. It is a recommendation of the World Wide Web Consortium (W3C) and provides three increasingly expressive sub-languages designed for use by specific communities of implementers and users. OWL Lite supports classification hierarchy and simple constraints. OWL DL supports maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computed) and decidability (all computations will finish in finite time). OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF² with no computational guarantees. OWL DL is based on the description logic ALC-SHOIN -D

¹ <https://www.w3.org/OWL/>

² <https://www.w3.org/RDF/>

and its subset. OWL 2 DL was standardized by the World Wide Web Consortium (W3C) in 2009 (and updated in 2012).

Protegé, a tool to create ontologies, uses the Manchester OWL syntax to represent the DL syntax. Then, special mathematical symbols such as \exists , \forall , \sqcap , \sqcup and \neg have been replaced by more intuitive keywords such as **some**, **only**, **and**, **or** and **not**. The OWL constructors for these symbols are, *someValuesFrom*, *allValuesFrom*, *intersectionOf*, *unionOf* and *complementOf*. To represent the **exclusive or**, i.e., the *oneOf* OWL constructor, it is used the $\{a\} \sqcup \{b\}$ in DL and $\{a, b\}$ in Manchester OWL (HORRIDGE et al., 2006).

Semantic Web Rule Language (SWRL)³ is a rule language based on OWL. Semantic Query-Enhanced Web Rule Language (SQWRL) (Semantic Query-enhanced Web Rule Language) is built on SWRL.

SWRL⁴ is an expressive OWL-based rule language. It allows users to write rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities than OWL alone. Semantically, SWRL is built on the same description logic foundation as OWL and provides similar strong formal guarantees when performing inference. The rules in SWRL consist of positive conjunctions of atoms. Informally, a SWRL rule may be read as meaning that if all the atoms in the antecedent are true, then all atoms in the consequent must also be true. SWRL does not support negated atoms or disjunction. SWRL's additional expressivity comes at the expense of decidability. That is, while OWL reasoners are guaranteed to terminate when classifying an OWL ontology, inference with SWRL rules is not. Depending on the underlying inference engine and the nature of a particular ontology and associated SWRL rules, the decidability caveat may have more theoretical than practical implications. As a general point, however, one should aim to stay within OWL if possible and only use SWRL when its additional expressive power is required.

SQWRL (O'CONNOR; DAS, 2009) takes a standard SWRL rule antecedent and efficiently treats it as a pattern specification for a query. SQWRL uses SWRL's built-in facility as an extension point. Using built-ins, it defines a set of operators that can be used to construct retrieval specifications. Standard SWRL serialization mechanisms can be used so queries can be stored in OWL ontologies.

The use of a reasoner allows the formal verification of the ontology. There are several reasoners⁵. Pellet, Fact++ and the Hermit are examples of reasoners and all of them are supported by Protegé⁶.

Pellet is the first reasoner that supported all of OWL DL SHOIN(D) and has been extended to OWL2 (SROIQ(D)). It supports OWL 2 profiles. It reasons ontologies through

³ <https://www.w3.org/Submission/SWRL/>

⁴ <https://github.com/protegeproject/swrlapi/wiki/SWRLLanguageFAQ>

⁵ <http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/>

⁶ <http://protege.stanford.edu/>

Jena ⁷ well as OWL-API ⁸ interfaces. Pellet also supports the explanation of bugs (ABBURU, 2012).

FaCT++ reasoner uses the same algorithm as in FaCT (Fast Classification of Terminologies), but with a different internal structure. It is implemented in C++ and it can be used as a description logic classifier and for modal logic satisfiability testing. The latest version of FaCT++ supports OWL and is based on the description logic SROIQ (ABBURU, 2012).

HermiT is an OWL 2 DL reasoner, one of the few such systems that attempts (modulo bugs) to fully and correctly support the OWL 2 DL specification. HermiT is the first publicly available OWL reasoner. It can check the OWL files to determine the consistency of the ontologies and to identify the hierarchical relationships between the classes. It also provides the faster process for classifying the ontologies (ABBURU, 2012).

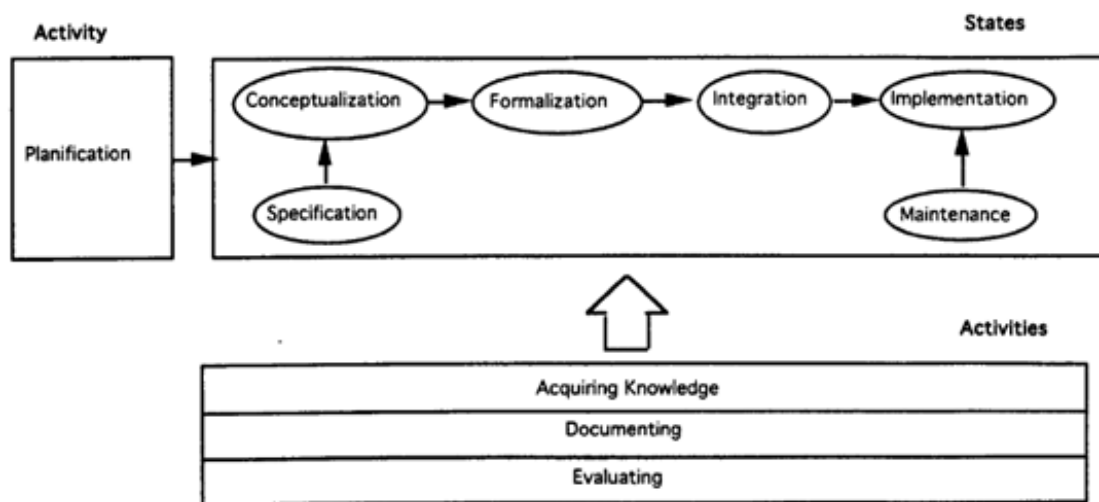
2.3.3 Methodologies to Create Ontologies

There are many methodologies to create ontologies. In this section, we present the methodologies adopted to create the ontology proposed in this thesis.

METHONTOLOGY

It allows to build ontologies from scratch. Figure 7 shows the states and activities in METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997). It consists of the following steps:

Figure 7 – METHONTOLOGY states and activities.



Source: Fernandez-Lopez, Gomez-Perez and Juristo (1997).

⁷ <https://jena.apache.org/>

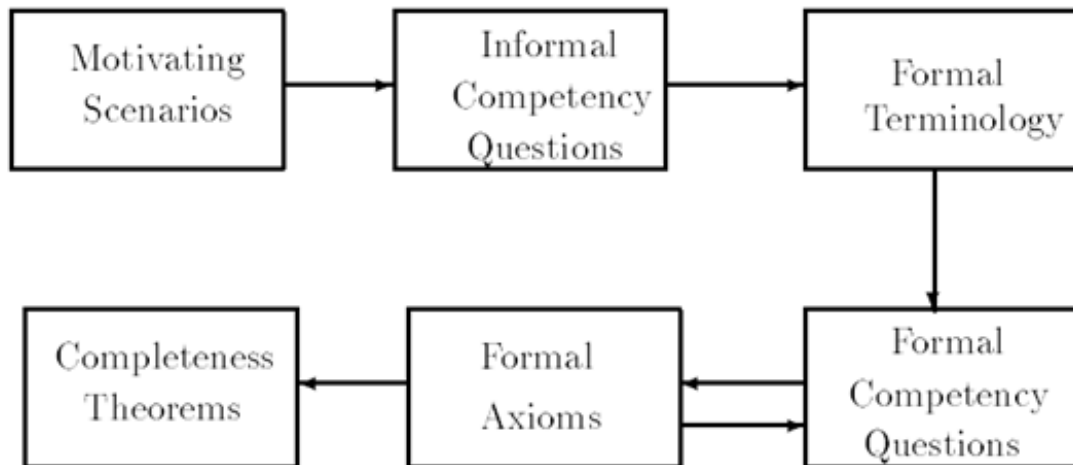
⁸ <http://owlapi.sourceforge.net/>

1. Specification - The goal of the specification phase is to produce either an informal, semi-formal or formal ontology specification document written in natural language, using a set of intermediate representations or using competency questions, respectively. In the specification phase, the following informations should be included: the purpose of the ontology, the level of formality of the implemented ontology, Scope, which encompasses the set of terms to be represented, its characteristics and granularity.
2. Knowledge Acquisition - Knowledge acquisition is an independent activity in the ontology development process. In this phase, the sources of knowledge are chosen to compose the base of knowledge of the ontology, and a first glossary with terms potentially relevant and their meaning is elaborated and refined.
3. Conceptualization - In this activity, the glossary of terms, initialized in the previous phase is completed. Concepts are described using data dictionary, which describes and gathers all the useful and potentially usable domain concepts, their meanings, attributes, and instances.
4. Integration - It is possible to consider reuse of definitions already built in other ontologies instead of starting from scratch. In this case, they propose the following: Inspect meta-ontologies to select those that better fit the conceptualization phase.
5. Implementation - Ontologies implementation requires the use of an environment that supports the meta-ontology and ontologies. The result of this phase is the ontology codified in a formal language.
6. Evaluation - Evaluation means to carry out a technical judgment of the ontologies, their software environment and documentation with respect to a requirements specification document during each phase and between phases of their life cycle. The output for this activity is many evaluation document describing how the ontology has been evaluated, the techniques used, the kind of errors found in each activity, and the sources of knowledge used in the evaluation.
7. Documentation - The documentation phase is executed in parts after each phase. In METHONTOLOGY, each phase produces a document. In fact, after the specification phase, you get a requirements specification document; after the knowledge acquisition phase, a knowledge acquisition document; after the conceptualization, a conceptual model document that includes a set of intermediate representations that describe the application domain; after the formalization, a formalization document; after the integration, an integration document; after the implementation, the implementation document; and during the evaluation, an evaluation document.

Ushold

Methodology for the design and evaluation of integrated ontologies. It uses first-order logic as the formal language. Figure 8 shows the production to design and evaluation of an ontology. It consists of six steps (USCHOLD; GRUNINGER, 1996):

Figure 8 – Procedure for a formal approach to ontology design and evaluation.



Source: Uschold and Gruninger (1996).

1. Capture of motivating scenarios - The development of ontologies is motivated by scenarios that arise in the applications. The motivating scenarios are story problems or examples which are not adequately addressed by existing ontologies. A motivating scenario also provides a set of intuitively possible solutions to the scenario problems.
2. Formulation of informal competency questions - Given the motivating scenario, a set of queries will arise which place demands on an underlying ontology. These queries are expressiveness requirements that are in the form of questions. An ontology must be able to represent these questions using its terminology, and be able to characterize the answers to these questions using the axioms and definitions. The informal CQ also provides an initial evaluation of the ontology.
3. Specification of the terminology of the ontology within a formal language such as first-order logic.
 - a) Informal Terminology - Given the informal competency questions, we can extract the set of terms used in expressing the question, these will form the basis for the specification of the terminology in a formal language.
 - b) Specification of Formal Terminology - A formal ontology is a formal description of objects, properties of objects, and relations among objects. For every informal competency question, there must be objects, attributes, or relations in the

proposed ontology or proposed extension to an ontology, which are intuitively required to answer the question.

4. Formulation of formal competency questions using the terminology of the ontology
 - Once the competency questions have been posed informally and the terminology of the ontology has been defined, the competency questions are defined formally as an entailment or consistency problem with respect to the axioms in the ontology. The axioms in the ontology provide the core axioms applicable to all objects and relations within the ontology as well as the definition of classes of objects. All terms in the statement of the formal competency questions must be included in the terminology of the ontology.
5. Specification of axioms and definitions for the terms in the ontology within the formal language
 - The axioms in the ontology specify the definitions of terms in the ontology and constraints on their interpretation; they are defined as first-order sentences using the predicates of the ontology. Axioms must be provided to define the semantics, or meaning, of these terms. The questions are used to evaluate the completeness of the sets of axioms in any particular axiomatisation.
6. Justification of the axioms and definitions by proving characterization theorems
 - Once the competency questions have been formally stated, we must define the conditions under which the solutions to the questions are complete. This forms the basis for completeness theorems for the ontology. Completeness theorems can also provide a means of determining the extensibility of an ontology, by making explicit the role that each axiom plays in proving the theorem.

SABiO

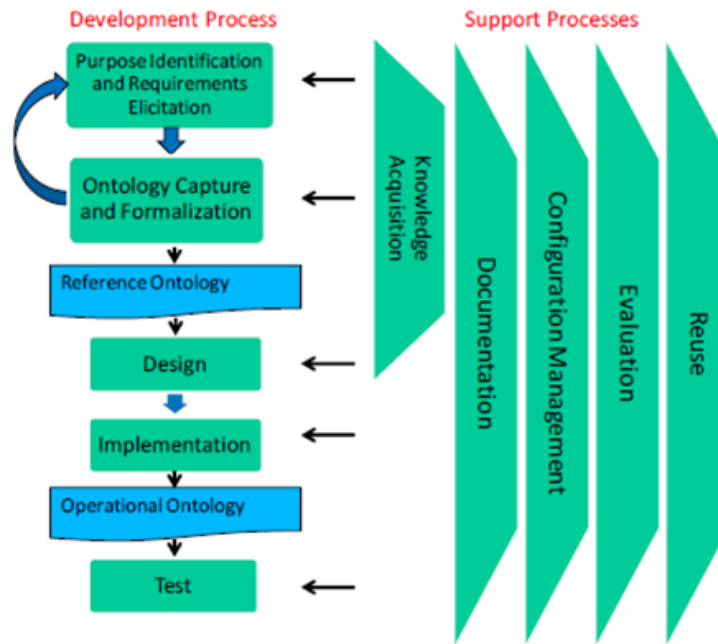
SABiO (FALBO, 2014) has five phases on the SABiO's Development Process and five support processes as shown in Figure 9. SABiO supports the development of both types of domain ontologies. If someone is interested in building a reference domain ontology, then the ontology project shall accomplish only the first three activities of the development process. If someone wants to build an operational domain ontology, then the ontology project has to perform the entire development process.

SABiO's Development Process

SABiO's Development Process has five phases: Ontology Purpose Identification and Requirements Elicitation, Ontology capture and formalization, Design, Implementation and Test.

1. Ontology Purpose Identification and Requirements Elicitation

Figure 9 – SABiO’s Processes.



Source: Falbo (2014).

The first phase in SABiO’s development process is Purpose Identification and Requirements Elicitation. This phase comprises three activities that occur in an iterative way. The activity of identifying the ontology purpose and its intended uses. To elicit its requirements. These should take the intended uses into account, and can be stated as competency questions. Lastly, to define the ontology scope and provide a way for its evaluation.

2. Ontology Capture and Formalization

The goal is to capture the domain conceptualization based on the ontology competence. The relevant concepts and relations should be identified and organized. A graphical model is a key instrument for supporting communication and it should be accompanied by a dictionary of terms. This phase is supported by knowledge acquisition activities. Knowledge can be elicited with domain experts and from sources of consolidated knowledge. This phase also defines the axioms specifying constraints and definitions of derived concepts. The process of defining axioms should be guided by the competency questions. The axioms in the ontology must be necessary and sufficient to express the competency questions and to characterize their solutions. After this phase, the reference ontology is created.

3. Ontology Design

In the design phase, the conceptual specification of the reference ontology should be transformed into a design specification. The design phase, thus, is necessary to

bridge the gap between the conceptual modeling of reference ontologies and the coding of them in terms of a specific operational ontology language.

4. **Ontology Implementation**

The implementation phase regards implementing the ontology in the chosen operational language.

5. **Ontology Test**

In SABiO, ontology test refers to dynamic verification and validation of the behavior of the operational ontology on a finite set of test cases, against the expected behavior regarding the competency questions. In this sense, SABiO's testing phase is driven by the competency questions. A test case comprises an implementation of a competency question as a query in the chosen operational language, plus instantiation data from the fragment of the ontology being tested and the expected result. Validation testing can also be performed by using the operational ontology in actual software applications, according to the intended uses originally proposed to the ontology. Validation testing should be performed by ontology users.

SABiO's Support Processes

SABiO considers five main support processes: knowledge acquisition, documentation, configuration management, evaluation and reuse.

1. **Knowledge Acquisition Process**

Knowledge acquisition occurs mainly in the initial phases of the ontology development process. Domain experts are the main source for knowledge acquisition. Other important sources of knowledge are consolidated bibliographic material, such as classical books and international standards, reference models, system models, existing ontologies, and ontology patterns.

2. **Reuse Process**

Regarding ontological resources, there are four main types of resources: existing domain ontologies, core ontologies, foundational ontologies, and ontology patterns. Reuse of core ontologies are made mainly by means of specialization. New concepts, relations, properties and axioms can then be introduced in the domain ontology. Foundational ontologies can be reused by means of specializations and by analogy. Reuse can also be achieved by means of ontology patterns (OPs). Finally, during ontology test, test cases can be reused.

3. **Documentation Process**

Documentation is a process that has to occur in parallel with the others. SABiO suggests three main documents to be produced as the documentation of an ontology

project. As a result of the first two phases of the development process, a Reference Ontology Specification should be produced. SABiO suggests that the organization defines naming conventions and rules for commenting the resulting code. For documenting the ontology test phase, a test document shall be produced, including test cases and test results. SABiO also suggests the use of wikis for documentation, especially for documenting reference ontologies.

4. Configuration Management Process

The main documents proposed by SABiO, as well as the source code of the operational ontologies must have their configuration managed.

5. Evaluation Process

SABiO's evaluation process comprises two main perspectives: ontology verification and ontology validation. Paraphrasing the definitions of software verification and validation, the concerns of ontology verification and validation (V&V) are the following: - Ontology Verification: aims to ensure that the ontology is being built correctly, in the sense that the output artifacts of an activity meet the specifications imposed on them in previous activities. - Ontology Validation: aims to ensure that the right ontology is being built, that is, the ontology fulfills its specific intended purpose. During the ontology capture and formalization phase, the reference ontology should also be verified whether it meets the requirements posed as competency questions. This can be done by means of technical reviews, applying expert judgment. A table indicating which ontology elements (concepts, relations, properties and axioms) are able to answer each competency question should be built. The purpose of this table is also for further verification. The competency questions play an essential role in the verification of the ontology completeness, especially when considering its axioms. Concerning ontology validation, the participation of domain experts and ontology users is essential. Ontology users have to evaluate whether the ontology is adequate for their intended uses. For validating the reference ontology with domain experts, the use of a graphical notation is very important, since generally they are not able to read formal specifications. Besides expert judgment, another relatively easy way to validate a reference ontology is by means of instantiation. The reference ontology should be able to represent real world situations.

2.4 Conclusion

In this Chapter, we presented the self-adaptive systems description with their main features, including the MAPE-K cycle, the modeling dimensions by Andersson et al. (2009) and an ontology for context by Jeong-Dong, Son and Baik (2012). We also presented a

explanation about requirements engineering area and the goal oriented field for the SAS context. Lastly, the explanation about ontologies e subjects related were accomplished.

3 ANALYSIS OF THE STATE OF THE ART

In this Chapter, we present two SLRs that we have performed, one of them is related to SAS domain and selected 13 works, and the other commits with CAS domain and selected 10 works, totalizing 23 works analyzed as related works. In Section 3.1, we present the SLR of ontologies for SAS domain. While in Section 3.2, we present the SLR of context domain. Then, in Section 3.3, we discuss the SLRs. Section 3.4 presents an analysis about the modeling languages for SAS and in Section 3.5, we summarize this Section.

3.1 Ontologies for SAS: A Systematic Literature Review

The SLR presented here was performed following the guidelines proposed by Brereton et al. (2007) and an overview of the protocol can be found in the Appendix A. In this section, we present a overview of the results.

This SLR involves these research questions:

RQ01: Which groups and organizations are most active in ontology for SAS research?

RQ02: Which languages are used to describe the ontology?

RQ03: Which research problems are addressed by the ontology?

RQ04: Which requirements engineering activities are considered by the ontology?

RQ05: Which application domains are involved?

RQ06: Does the analyzed ontology present just core concepts?

RQ07: Does the ontology have a tool to support RE activities?

RQ08: Which concepts are presented in the ontology?

The search string, presented in Table 1, returned 2162 papers and this SLR resulted in 12 selected papers. This study was based on automatic and manual search. The databases used in the automated search were: ACM, IEEE, SpringerLink, Science Direct, ISI Web of Science and Scopus. The automated search was performed in the search engines considering the Computer Science area in January 2016. No start date was used. The manual search was conducted by analyzing all available proceedings of the major conferences, workshops and journals in the requirements engineering and self-adaptive systems areas, including: RE, ICSE, SEAMS, CAiSE, REFSQ, SAC, REJ and JSS.

In February 2017, we performed again a manual search in the same conferences, workshops and journals. We have searched by editions published between January 2016 and January 2017 because this was the period of time not covered by the previous search. Our search did not return any paper satisfying any inclusion criteria. However, by doing an informal search (Google Scholar), we have found two papers that satisfy the inclusion criteria. They are the works presented in Soares et al. (2016) and Duarte et al. (2016).

Table 1 – Search String.

(

(ontology OR “knowledge representation” OR ontological) AND

(“self-adaptive system” OR “adaptive system” OR “dynamic system”)

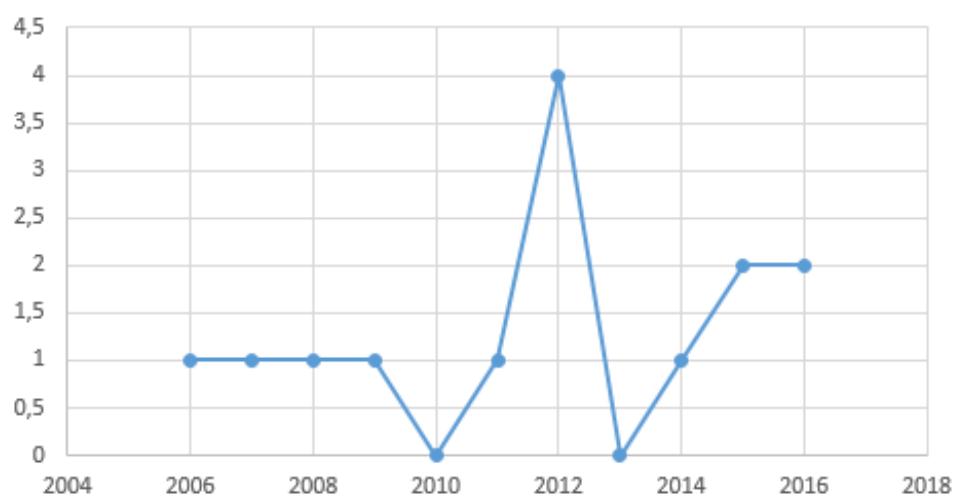
)

Source: Own.

This SLR resulted in the selection and analysis of 14 papers. The studies presented in Vassev and Hinchey (2015a) and Vassev and Hinchey (2015b) are very similar and, for our research, they answer the research question in the same way. We considered the study presented by Vassev and Hinchey (2015b) because it is a journal. The 13 studies identified are presented in Table 2. This table also presents the year of publication, the paper type (conference or journal), the language used in the study to describe the ontology and whether the ontology presented in the study has tool support.

Figure 10 presents the number of studies per year. We can notice that the year with more publications was 2012, with a total of 4 published studies (30.7%). The years 2010 and 2013 had no published study. It is worth pointing out that this research had no time restriction. The selected studies were published between 2007 and 2016 (this search was conducted in January 2016 and updated in January 2017). 2006, 2007, 2008, 2009, 2011 and 2014 had only one published study (7.7%) each and 2015 and 2016 had two published studies (15.3%) each.

Figure 10 – Number of studies per year.



Source: Own.

There was a great interest in the topic in 2012, reaching the maximum number of publications, but in 2013 this interest decreased the number of publications to 0. In 2014,

Table 2 – Studies identification.

ID	Ref.	Year	Paper Type	Ontology Language	Tool Support	Evaluation
S1	(VASSEV; HINCHEY, 2015b)	2015	Journal	Knowlang	Yes	Case Study
S2	(WEICHHART; NAUDET, 2014)	2014	Conference	OWL	No	None
S3	(SUDHA et al., 2007)	2007	Conference	OWL	No	None
S4	(QURESHI; PERINI, 2009)	2009	Conference	None	No	None
S5	(TRAN; CIMIANO; ANKOLEKAR, 2006)	2006	Conference	OWL	No	None
S6	(QURESHI; JURETA; PERINI, 2011)	2011	Conference	None	No	None
S7	(SERBEDZIJA; FAIRCLOUGH, 2012)	2012	Journal	UML	Yes	Case Study
S8	(PANTSAR-SYVÄNIEMI et al., 2012)	2012	Journal	OWL	Yes	Case Study
S9	(GACEK; GIESE; HADAR, 2008)	2008	Conference	None	No	None
S10	(YUAN; MALEK, 2012)	2012	Conference	None	No	None
S11	(RAMIREZ; JENSEN; CHENG, 2012)	2012	Conference	None	No	None
S12	(SOARES et al., 2016)	2016	Conference	None	No	None
S13	(DUARTE et al., 2016)	2016	Conference	UML	No	Case Study

Source: Own.

2015 and 2016, the number of publications on this topic began to increase again, but it is still difficult to predict whether there is a tendency of interest in this topic by researchers.

We can notice that 3 studies, S1, S7 and S8 (23%) presented evaluation of the work, and the three used a case study to validate their study.

We present the results organized according to the Research Questions (RQ).

RQ01: Which groups and organizations are most active in ontology for SAS research?

This research question is related to the groups that are working with ontology and SAS.

This RQ aims to conduct a survey of the regional distribution of the studies and identify the groups and organizations that are researching ontologies for self-adaptive systems. The 13 papers are related to 19 groups. This happened because some included papers belong to more than one group, like S2 that belongs to 3 groups, specifically in Germany, Austria and Luxembourg. Likewise, S9 belongs to three groups in UK, Germany and Israel. Fondazione Bruno Kessler - IRST, Software Engineering Research Group, Trento, Italy has two studies S6 and S4. Table 3 presents the studies distributed by groups.

When we distribute the studies by country, we can extract twelve countries: Israel, Luxembourg, India, USA, Austria, Belgium, Italy, Germany, Ireland, UK, Brazil and Finland. From that, we can observe that the Germany has four studies (S3, S6, S8 and S10), corresponding to 30.7% of the selected studies. Italy, UK, EUA and Brazil have two studies each (S4 and S6, S7 and S9, S10 and S11, and S12 and S13, respectively), corresponding to 15.3% of the selected studies. Last, India, Austria, Israel, Luxembourg, Belgium, Finland and Ireland have one study each (S4, S3, S10, S3, S7, S9, S1, respectively), corresponding to 7.7% of the selected studies. Therefore, we can observe that Germany is the most active country in this research topic.

RQ02: Which languages are used to describe the ontology?

This RQ has the objective to identify which ontology description languages are used to describe ontologies in the selected studies. We identified three languages in the studies: OWL ¹, Knowlang (VASSEV; HINCHEY, 2015b) and UML ². 46.15% of the studies do not use a specific language to describe ontologies (S4, S6, S9, S10, S11 and S12). 30.77% of the studies use OWL to describe the ontology (S2, S3, S5 and S8), 15.38% of the studies (S7 and S13) use UML to represent the ontology and only 7.7% of the studies (S1) use Knowlang to specify the ontology. OWL is the most popular ontology description language in the literature for this domain. It can describe the concepts, relationships between the concepts and the axioms. In fact, seven out the studies use some description language and four of them (S2, S3, S5 and S8) describe the ontology by using OWL, as shown in Figure 11. Six out the studies do not use language to describe the ontology.

Table 4 shows the ontology language and the studies that used it. UML was created to model system specification, an objective different from describing ontologies; it is used to model the structure and behavior of object-oriented systems. The Knowlang is a knowledge representation language for self-adaptive systems.

Dermeval et al. (2014) present a systematic literature review about the use of ontologies in requirements engineering and show a result of 50% of the studies using OWL as description language. Note that the SLR presented in Dermeval et al. (2014), in particular, did not explore the use of ontology for SAS requirements engineering, but the use of

¹ <https://www.w3.org/OWL/>

² <http://www.uml.org/>

Table 3 – Groups or organizations by studies.

Groups or Organizations	Related Studies
CA Labs, CA Inc. Yokneam, Israel	S9
Centre de Recherche Public Henri Tudor, Luxembourg	S2
Centre for Development of Advanced Computing (CDAC), Chennai, Índia	S3
Department of Computer Science, George Mason University, McLean, Virginia, USA	S10
Dept. of Communications Engineering - Business Informatics, Johannes Kepler University Linz, Austria	S2
FNRS & Louvain School of Management, University of Namur, Belgium	S6
Fondazione Bruno Kessler - IRST, Software Engineering Research Group, Trento, Italy	S6,S4
Fraunhofer FIRST, Germany	S7
Hasso Plattner institute at the University of Potsdam, Germany	S9
Institute AIFB, University of Karlsruhe, Germany	S5
Lero—the Irish Software Engineering Research Center, University of Limerick, Limerick, Ireland	S1
Liverpool John Moores University, UK	S7
Madras Institute of Technology, Anna University, Chennai, Índia	S3
Metasonic AG, Pfaffenhofen, Germany	S2
Michigan State University, East Lansing, MI, EUA	S11
School of Computing Science University of Newcastle Newcastle upon Tyne, UK	S9
VTT Technical Research Centre of Finland	S8
Federal University of Pernambuco, Brazil	S12
Ontology & Conceptual Modeling Research Group (Nemo) – Department of Informatics – Federal University of Espirito Santo (Ufes), Brazil	S13

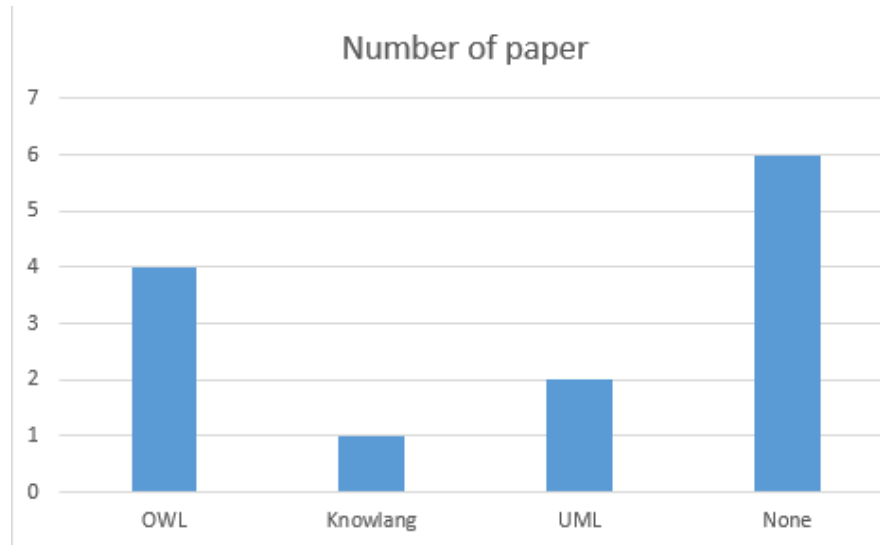
Source: Own.

ontology in the requirements engineering activities in general.

RQ03: Which research problems are addressed by the ontology?

This research question is about the objective of each ontology. It is related to the problem the ontology is concerned to solve. This RQ provides an overview about the problems that are already addressed by the existing ontologies and to detect those problems that

Figure 11 – Ontology Languages adopted by the papers.



Source: Own.

Table 4 – Ontology Language by study.

Ontology Language	Related Studies
OWL	S2, S3, S5 and S8
None	S4, S6, S9, S10, S11 and S12
Other	S1 (Knowlang), S7 and S13 (UML)

Source: Own.

have not been addressed yet. Table 5 presents the problem addressed by each study.

S1 presents a formal approach and demonstrates how knowledge representation and reasoning help to establish the vital connection between knowledge, perception, and actions that comprehend self-adaptive behavior.

S2 describes the application of the Complex Adaptive Systems extended Ontology of Enterprise Interoperability in the Knowledge Life Cycle framework. It enables agents in a dynamic system to communicate semantically unambiguously to solve interoperability related problems.

S3 study introduces a context-aware and coordination middleware framework for Ubiquitous Environment using Ubiquitous Semantic Space. Ubiquitous Semantic Space uses ontologies to define the semantics of various concepts. Using ontologies facilitates different agents in the environments to have a common semantic understanding of different contexts.

S4 proposes a characterization of adaptive requirements. It investigates how available techniques for representing domain properties and techniques for modelling stakeholders' goals and preferences can provide a practical support to the analyst while capturing

Table 5 – Problems addressed by each study.

Problem	Related Study
Knowledge representation and reasoning for awareness and self-adaptive behavior.	S1
Modelling dynamic aspects and active agents that are placed in a complex adaptive business ecosystem.	S2
Modeling ubiquitous semantic space.	S3
Characterization of adaptive requirements	S4
Difficulties in the acquisition of model information and the lack of control and transparency of the system's adaptive behavior.	S5
Representation of the requirements problem on the requirements engineering for SAS.	S6
Monitoring the psychological status of the user.	S7
Heterogeneity and complexity of computing, communication, and software technologies in smart environments.	S8
Developing a new system to handle all aspects of the evolution autonomously.	S9
It is a comprehensive taxonomy to classify and characterize research efforts in Self-protecting software systems arena.	S10
It is a taxonomy of potential sources of uncertainty at the requirements, design, and execution phases of DAS.	S11
Requirements specification for SAS	S12
Requirements at runtime	S13

Source: Own.

adaptive requirements.

Study S5 introduces an ontology-based approach that allows adaptation customized to different requirements. The proposal has been designed to maximize extensibility by the use of de-facto standard languages, terminologies and ontology modeling principles. It provides a conceptualization of objects and relations that are relevant for the adaptation of hypermedia resources to the user context.

S6 defines a minimal set of concepts and relations needed to formulate the requirements problem, its solutions, the changes in its formulation that arise from changes in the operating context, requirements, and resource availability.

S7 describes a number of conceptual issues associated with smart adaptive technology. The biocybernetic loop describes different approaches for monitoring the status of the user from physiological sensors to the behavior. Pervasive adaptive systems are concerned with the construction of smart technologies capable of adapting to the needs of the individual in real time.

S8 introduces an innovative adaptation framework for the situation-based and self-adaptive applications of smart environments. The framework embodies a novel architecture, generic ontologies for context, security, and performance management, and dynamic models for performing runtime reasoning and adaptation.

S9 presents a conceptual model for self-adaptation capturing the concepts and overall interweaving process of self-adaptation. It defines the required roles and responsibilities, along with the various process artifacts and maps them onto the current practice in ITIL change management, evaluating similarities and differences. It also discusses the implications for the co-existence of self-adaptation and change management and the potential for optimization for the different activities.

S10 reports a study and analysis of the literature in the self-protecting software systems area. Self-protection is when the security threats are detected and mitigated at runtime and it has been identified as one of the essential traits of self-management for autonomic computing systems. This study surveys research on self-protecting software systems in order to evaluate relevant research approaches and to identify the taxonomies and schemes against which such research efforts may be classified and organized. The main contribution is a taxonomy to classify and characterize research efforts in the adaptive security arena.

Study S11 revisits the concept of uncertainty from the perspective of a DAS, proposes a taxonomy of potential sources of uncertainty at the requirements, design, and execution phases, and identifies a preliminary set of promising techniques that can mitigate different types of uncertainty. The study also introduces a template to facilitate the organization and reuse of the proposed taxonomy of uncertainty.

S12 presents a core ontology for self-adaptive systems in order to facilitate the requirements elicitation and specification. It presents a comparative study with three GORE modeling languages for SAS, the Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PIMENTEL, 2015), by using the modeling dimensions by Andersson et al. (2009). The core ontology presents the modeling dimensions concepts.

Finally, S13 study introduces the Runtime Requirements Ontology, a domain ontology that intends to represent the nature and context of Requirements at Runtime. The paper also presents an evaluation of the ontology by using verification and validation techniques.

The S1 study dealt with knowledge representation and reasoning. The S2 problem is about modeling of dynamic aspects and the S3 addresses the modeling ubiquitous semantic space. The S4 and S6 dealt with requirements and the S5 study is concerned with adaptive behavior of the systems. The S7 study dealt with the monitoring of the user and the S8 study is concerned with smart environments. The S9 dealt with autonomously evolution and the studies S10 and S11 present a taxonomy. S12 and S13 present solutions for requirements

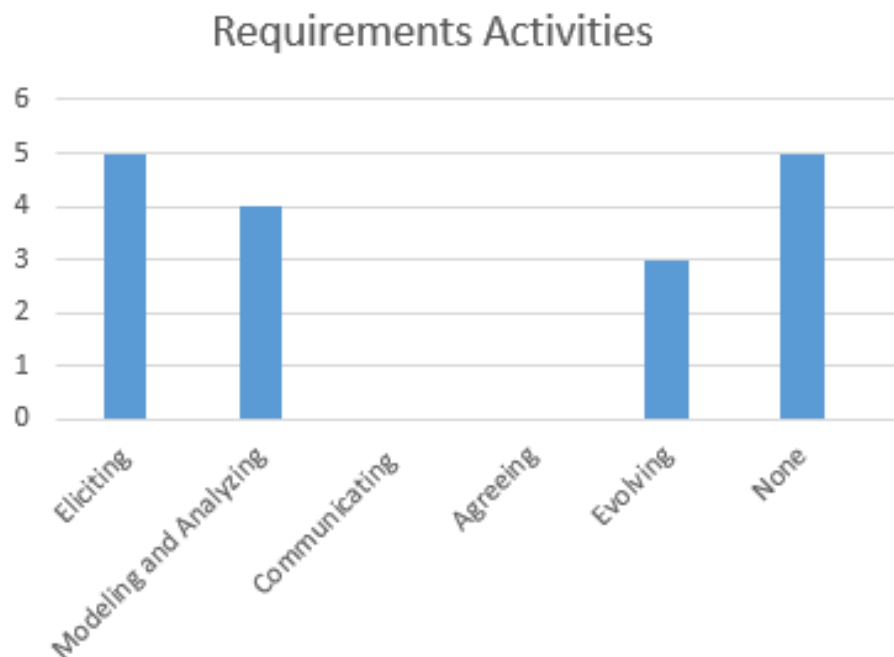
RQ04: Which requirements engineering activities are considered by the ontology?

This research question is related to requirements engineering process activities that the ontology is created to support them. This RQ investigates the main requirements engineering activities (Eliciting Requirements, Modelling and analyzing requirements, Communicating requirements, Agreeing requirements or Evolving requirements according Nuseibeh and Easterbrook (2000)) supported by the existing ontologies in the self-adaptive systems context.

We discovered that S4, S6, S9, S12 and S13 (38.5%) consider the eliciting activity and 30.77% of the studies (S4, S6, S10 and S11) consider the modeling and analyzing requirements activities. S8, S9 and S11 (23.07%) consider the evolving activity. Note that 38.5% of the studies (S1, S2, S3, S5 and S7) did not consider any requirements engineering activity. Moreover, both communicating and agreeing requirements activities were not considered in any study.

Figure 12 shows the quantity of studies by requirements engineering activities.

Figure 12 – Studies by RE activities.



Source: Own.

The papers that do not mention any requirements engineering activity (S1, S2, S3, S5 and S7) are, therefore, not related to RE, but benefits the domain of SAS. The other eight papers (S4, S6, S8, S9, S10, S11, S12 and S13) cite the requirements elicitation, modeling and analysis, and evolve activities. The most cited activity is the requirements elicitation (S4, S6, S9, S12 and S13). Table 6 presents the requirements engineering activities by

paper.

Table 6 – Requirements Engineering activities by studies.

Requirements Engineering Activities	Related Studies
Eliciting Requirements	S4, S6, S9, S12 and S13
Modelling and analyzing requirements	S4, S6, S10 and S11
Communicating requirements	0
Agreeing requirements	0
Evolving requirements	S8, S9, S11
None	S1, S2, S3, S5 and S7

Source: Own.

RQ05: Which application domains are involved?

The application domain focused by each paper is listed in the following: The S1 study has the *Science Cloud* as application, S2 uses a *Enterprise System*, S3 uses the *e-Health Ubicom* application, S4 uses the *Travel booking software* application, S5 uses the *Personalized portal* and S6 uses a *Travel booking software*. S7 uses three application domains, a *music player*, a *cognitive monitor* and a *reflective vehicle*. S8 uses the *Smart building maintenance*, S9 uses the *ITIL Change Management*. S10 uses an *Online Banking System* and S11 uses an *Intelligent vehicle systems*. S12 uses *SAS domain* and S13 uses a *Meeting Scheduler*. We can notice that the S4 and S6 studies use a travel booking application, S5 and S10 use a web application and S7 and S11 use a automotive application. Figure 13 presents the application domains used in the studies.

RQ06: Does the analyzed ontology present just core concepts?

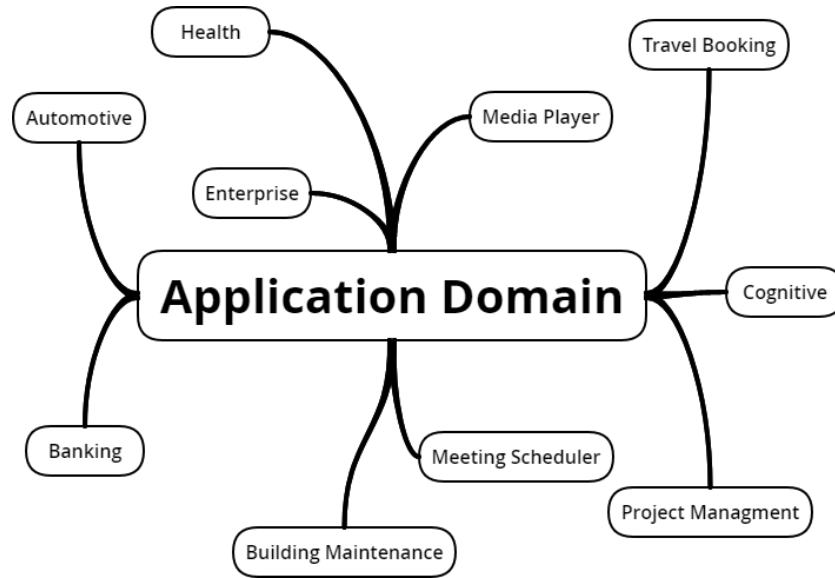
Core ontology is an ontology that presents a minimal set of concepts and the most general concepts of a domain. The objective of this question is to understand if the ontologies are complete or present just a minimum set of concepts allowing their expansion. A core ontology can be specialized in a complementary manner in order to be used in practice.

The S2, S6 and S12 studies present general concepts for the ontology. It means that 23% of the papers present a core ontology. A core ontology can and should be extended to include the concepts and relationships needed for the domain. S2 presents a core ontology but it also presents an expansion of it.

RQ07: Does the ontology have a tool to support RE activities?

This research question is related to tool support for using the ontology. This RQ aims to identify studies that support requirements engineering activities for self-adaptive systems through an ontology in an automated manner.

Figure 13 – Application Domain for SAS studies.



Source: Own.

We can notice that 23% of the papers offer tool support (papers S1, S2 and S8). A tool support can facilitate the use of the ontology. The studies S2 and S8 had their ontology described by OWL and the S1 used Knowlang as description language.

RQ08: Which concepts are presented in the ontology?

This RQ is intended to discover the concepts covered by the ontologies present in the studies.

The concepts of S1 are related to self-adaptation capabilities: Policies (goal, situations, situation relations, and conditions), Events, Actions, Situations, Resource, User, Software, and Phenomenon (communication, time, capacity and knowledge on goal, action, event, behavior, task, policy, state, and situation).

S2 presents the following concepts: systems, behavior, objective, function, environment, model, system to build, interface, relation, interaction, structure, system element, state, memory, time, attractor, agent, Complex Adaptive Systems - CAS (set of agents).

S3 presents just three concepts. They are: Agent, Space and Device. While S4 presents the concepts of: When (flexibility), Where (variability), Plans - How (uncertainty, context and behaviours).

S5 presents the following concepts: Context, Task, Presentation, Environment, System, Domain, Service, Agent and User. While S6 presents the concepts of: Context, Communicated information, Resource, Domain assumption, Goal (Functional goal, Quality constraint and Softgoal), Task and Evaluation.

S7 presents the following concepts: Device (Sensor, Actuator), Feature (User, Context), Feature (Simple, Complex, Sense, Analyse). While S8 presents the concepts of: Context,

ActiveObject (preference, event), Agent, Service, Rule, Person, Role, Quality and Trigger.

S9 presents the following concepts: Sensors, Monitors, Implementation team, Analyzers, Approver, Planer, Change Advisory Board, Approvers, Scheduler, Process Owner, Change Manager, and Affected Users.

S10 presents the concepts of: Monitor & Detect, Analyze & Characterize, Plan & Adapt, Network, Host, App / Software, Services, Abstract Architecture, Development Time, Runtime, Confidentiality, Integrity, Availability, Local Only, Centralized, Decentralized, Human-Driven, Heuristics-Driven, Algorithm-Driven, Other, Reactive, Proactive, Neutral / Hybrid, System Boundary, System Internals, Neutral / Hybrid, Proxy/-Containment, Redundancy, Diversity, Recomposition, Rejuvenation.

S11 presents the concepts of: Requirements (Missing Requirement, Ambiguous Requirement, Falsifiable Assumption, Unsatisfiable, Requirements, Requirements Interactions, Doubt, Claim, Changing Requirements), Design, Unexplored Alternatives, Untraceable Design, Risk, Misinformed Tradeoff Analysis, Inadequate Design, Unverified Design, Inadequate Implementation, Latent Behavior), Run-Time (Effector, Sensor Failure, Sensor Noise, Imprecision, Inaccuracy, Unpredictable Environment, Ambiguity, Non-Specificity, Inconsistency, Incomplete Information).

S12 presents the following concepts: Communicated Information, Context, Resource, Role, action, Location, Status, Time, Goal (Functional goal, Quality constraint, Soft-goal), Task, Domain Assumption, Change, Mechanism, Effect, Evaluation, Comparative evaluation, individual evaluation.

S13 presents the following concepts: Requirement, Software Function Universal, Program, Software Function, Loaded Program Copy, Program Copy Execution, Observable State, Requirement Artifact, Runtime Requirement Artifact, Compliance Program Copy Execution, Monitoring Runtime Requirement Artifact, Adaptation Program Copy Execution, Change Runtime Requirement Artifact.

S1 and S7 studies present the user concept. It considers the interaction between the user and the system. S2, S3, S5 and S8 studies present the agent concept to represent the interaction in smart environment. The context concept is considered by S4, S5, S6, S7, S8 and S12 studies, i.e., 46,15% of the studies have context concept.

S9, S12 and S13 studies present a concept related to change. S9 study presents the change manager concept. S1, S4, S6 and S12 studies have the concept of goal, which facilitates the goal-oriented modeling. The concept of task is present in S1, S5, S6 and S12 studies and it can facilitate the representation of a plan. The S9 study presents the planer concept and S4 study presents the how concept.

Only the S7, S9 and S11 studies present the concept of sensor. The concept of actuator is considered by the S7 (actuator) and S11 (effector) studies. Although these studies present ontologies for SAS, some of them do not have concepts that clearly express change or adaptation, as in S1, S2, S3, S4 and S5 studies.

The unique studies that do not clearly show context-related concepts are the studies S9, S11 and S13. The studies S7, S9 and S10 do not clearly consider the concept of goal, although the studies S9 and S10 cite requirements activities. Despite studies S1 to S6 plus S11 do not cite requirements activities, they do present goal concepts.

From the studies that cite requirements engineering activities, only the studies S8 and S13 use a language to specify the ontology. The study S8 uses OWL and the study S13 uses UML.

Context is a decisive concept for SAS domain. Then, to know the others concepts related to context, we performed a SLR to CAS domain. It is presented in the next section.

3.2 Ontologies for CAS: A Systematic Literature Review

The SLR presented here was performed following the guidelines proposed at Brereton et al. (2007) and an overview of the protocol can be found in the Appendix B. In this section, we present an overview of the results.

The search string, presented in Table 7, returned 653 papers and this SLR resulted in 10 papers. This study is based on automatic and manual search. The databases used in the automated search were: ACM, IEEE, Springer, Science Direct, ISI Web of Science and Scopus. The automated search was performed in the search engines considering the Computer Science area in May 2016. No start date was used.

Table 8 presents 10 studies identified. The table also presents the paper title and the respective reference.

Table 7 – Search String.

“ontolog” AND “context – awaresystem”*

Source: Own.

This SLR has three research questions. This SLR has less questions than the previous SLR because we were only interested in the **context** term, since the other questions of interest had already been answered by the other SLR. The three research questions:

RQ01: Which concepts are presented in the ontology?

RQ02: Which languages are used to describe the ontology?

RQ03: What application domains are involved in the paper?

We present the results organized according to the Research Questions (RQ).

RQ01: Which concepts are presented in the ontology?

Table 8 – Studies and references.

ID	Title	Ref.
S14	An Ontology based Context-aware Model for Semantic Web Services	(HU; LI, 2009)
S15	A Semantic Web Based System for Context Metadata Management	(STEFANOV; HUANG, 2009)
S16	iConAwa - An intelligent context-aware system	(YILMAZ; ERDUR, 2012)
S17	A service-oriented middleware for building context-aware services	(GU; PUNG; ZHANG, 2005)
S18	Refinement Strategies for Correlating Context and User Behavior in Pervasive Information Systems	(JAFFAL; GRAND; KIRSCH-PINHEIRO, 2015)
S19	Ontology-based context modeling to facilitate reasoning in a context-aware system: A case study for the smart home	(HOQUE et al., 2015)
S20	Smart-context: A context ontology for pervasive mobile computing	(MOORE; HU; WAN, 2008)
S21	Towards an Extensible Context Ontology for Ambient Intelligence	(PREUVENEERS et al., 2004)
S22	Towards an Ontology for Context Representation in Groupware	(VIEIRA; TEDESCO; SALGADO, 2005)
S23	CA5W1HOnto: Ontological Context-Aware Model Based on 5W1H	(JEONG-DONG; SON; BAIK, 2012)

Source: Own.

The S14 study presents the following concepts: User Context (Social Info, Organization Info, Operation History, Age, Agenda, Contact Info, Mood, and Permission Profile); Platform Context (physical device such as network device, computing device, and IO device, and computational resource such as code, middleware and operating system); Environment Context (“location” (city, country, room, street), “time” (year, month, day), "network" (delay, bandwidth), and “physical environment” (temperature, lighting, noise, situation); Service Context (including Contextual property (current overload, current distance) and Quality of Service (QoS) property (cost, response time, latency, throughput, capability).

The S15 study presents the following concepts. Sensor (motion, GPS, acoustic, eletromagnetic, chemical, humidity, orientation), SpatialThing, Region, Person, Agent, OnlineAccount, User, PoliticalRegion, Point, Location, Device, MobilePhone.

The concepts presented by S16 study are: Location (latitude, longitude), Time, PersonalProfile, Device, Network, Preference. While the concepts present in S17 study are: Person, Location, CompEntity, Activity, service, Application, Device, Network, Agent.

The S18 study presents the following concepts: User (Activity, Role, Profile), Environment (Time, Location), Network, Device.

The S19 study presents the following concepts: Time, Person, Activity (Deduced Activity), Device (Sensor, Smart Device, Appliance), Space (Outside, Inner Space), Environment.

The Spatio-Temporal, Device, Resource, Person, Preference, Schedule, Activity, Task and Role concepts are presented in the S20 study.

The S21 study presents the following concepts: User (activity, role, profile, mood, task), Platform (software (rendering engine, operating system, virtual machine, middleware), hardware (resource, device)), Service (service profile, service model, service grounding), Environment (location (relative, absolute, address), time, environmental condition (temperature, pressure, humidity, lighting, noise)).

The S22 study presents the following concepts: InteractionContext (SharedArtifactsContext, ApplicationContext), OrganizationalContext (ProjectContext, AgentContext (HumanAgentContext, SoftwareAgentContext), GroupContext, TaskContext, RoleContext), PhysicalContext (DeviceContext, LocationContext, TimeContext, ConditionContext).

The S23 study presents the following concepts: Role (Actor, Organization, System), Goal (Personal Goal, Role Goal, Nonfunctional Goal, Functional Goal), Action (Atomic Action, Composite Action, Expectation (Failure), Input, Output, Effect, Precondition, contextual, trustworthy), Status (Atomic Status, Composite Status), location (Atomic Location, Composite Location), and time (Start Time, End Time, Repetition Time).

The Location concept is represented in eight studies and the Time concept is represented in seven studies. The Role and Activity concepts are represented in five studies. The Environment, Network, Person and User are represented in four studies. The Agent and Task are represented in three studies. Table 9 presents the ten most cited concepts and the studies that cite them. Figure 14 presents the concepts and the studies that cite each concept.

RQ02: Which languages are used to describe the ontology?

Eight from ten studies use OWL to describe the ontology, they are S14, S15, S16, S17, S19, S21, S22 and S23. One of ten studies uses RDF to describe the ontology, it is S20. One of ten studies does not use a description language to describe the ontology, it is S18.

RQ03: What application domains are involved in the paper?

We can note that the majority of the domain application used to illustrate the ontologies on studies use applications that need some interaction with the user. Figure 15 presents the application domains used in the studies.

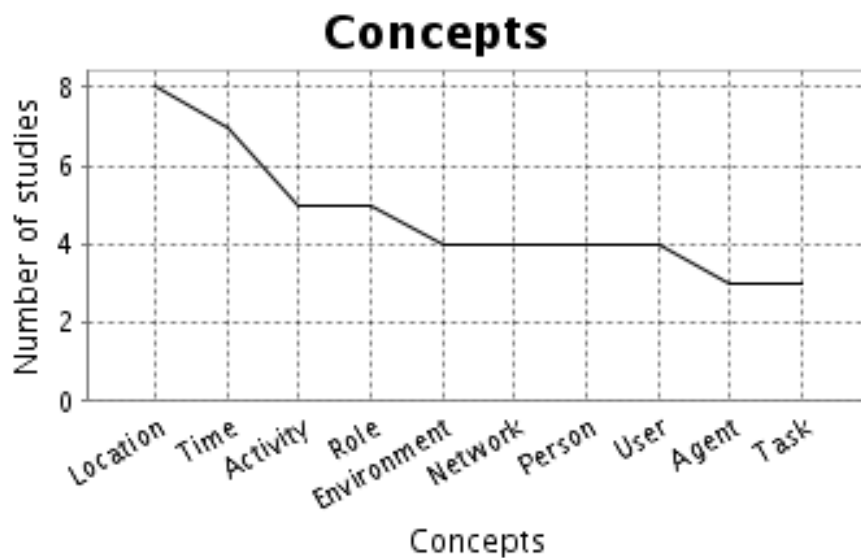
Analyzing the studies in order to notice how many of the most cited concepts they present, the result is that the S14 study has five of the ten most cited concepts, S15 study has four of the ten most cited concepts, S16 study has three of the ten most cited concepts, S17 study has five of the ten most cited concepts, S18 study has seven of the

Table 9 – Concepts by studies.

Concept	Cited by
Location	S14, S15, S16, S17, S18, S21, S22, S23
Time	S14, S16, S18, S19, S21, S22, S23
Activity	S17, S18, S19, S20, S21
Role	S18, S20, S21, S22, S23
Environment	S14, S18, S19, S21
Network	S14, S16, S17, S18
Person	S15, S17, S19, S20
User	S14, S15, S18, S21
Agent	S15, S17, S22
Task	S20, S21, S22

Source: Own.

Figure 14 – Amount of citations.



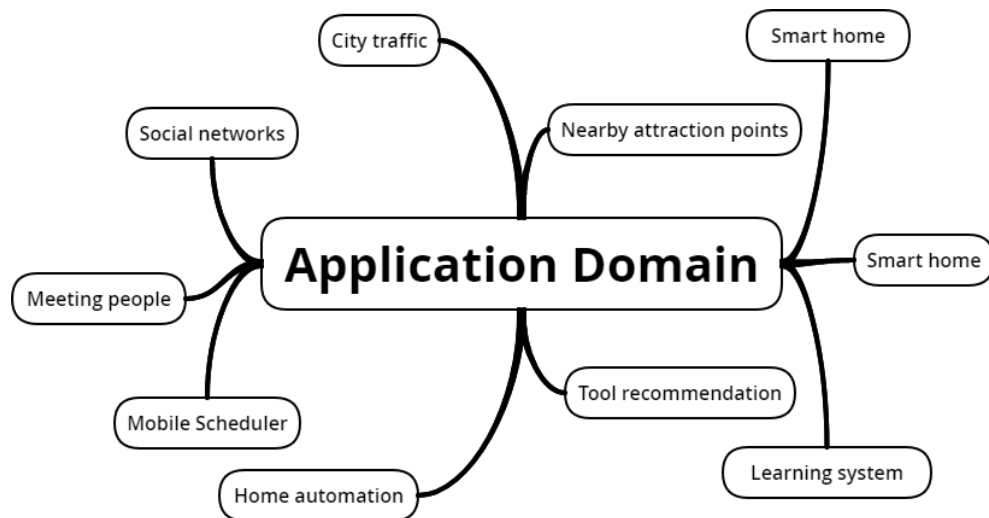
Source: Own.

ten most cited concepts, S19 study has four of the ten most cited concepts, S20 study has four of the ten most cited concepts, S21 study has seven of the ten most cited concepts, S22 study has five of the ten most cited concepts and S23 study has three of the ten most cited concepts. S18 and S21 represent the most quantity of the ten more cited concepts.

The S18, S21, S22 and S23 are the studies that present larger quantity of the most cited concepts.

Analyzing the studies regarding the four most cited concepts (Location, Time, Activity and Role), the result is that the S14 study presents two of the four most cited concepts, S15 study presents one of the four most cited concepts, S16 study presents one of the

Figure 15 – Application Domain for CAS studies.



Source: Own.

four most cited concepts, S17 study presents two of the four most cited concepts, S18 study presents four of the four most cited concepts, S19 study presents two of the four most cited concepts, S20 study presents two of the four most cited concepts, S21 study presents four of the four most cited concepts, S22 study presents three of the four most cited concepts and S23 study presents three of the four most cited concepts.

Note that some concepts are present in the GORE modeling languages, presented in Section 2.2.2, but they are not in SLR studies, like as **Condition**, **Failure**, **Error** and **Conflict**. While some concepts are in SLR studies but not in languages, for example, **Behavior**, **Environment**, **Sensor**, **Effector**, **Analyze**, **Trigger**, **Monitor** and **Change**.

3.3 Discussion of the SLRs

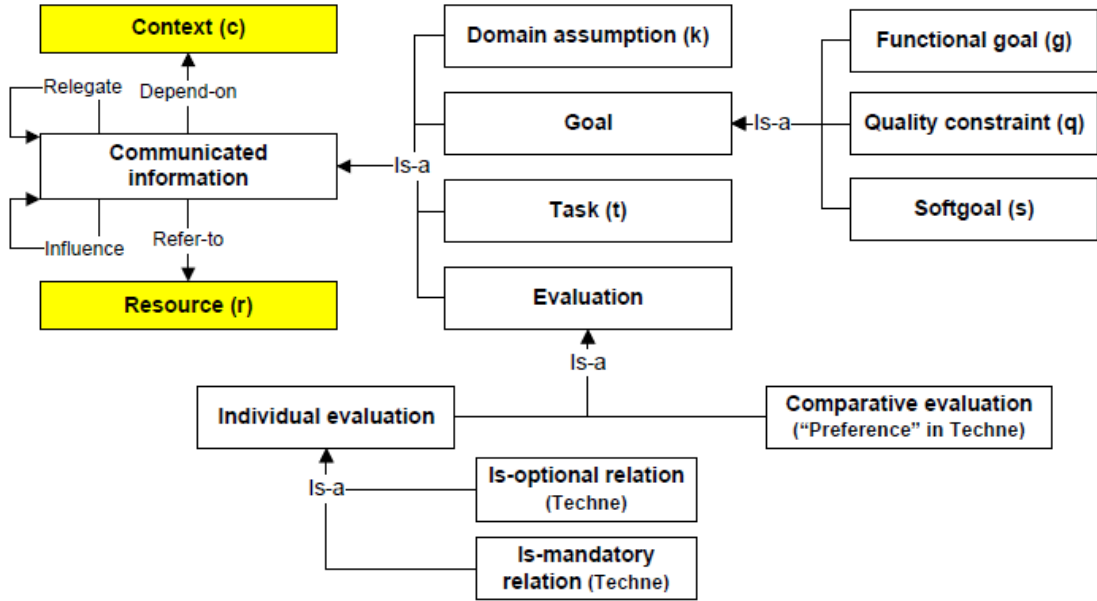
We presented the results of two systematic literature reviews, one of them concerned in the use of ontologies for SAS and another for CAS. We obtained 13 studies for SAS domain and 10 studies for CAS domain. Then, we extracted information from these studies to answer some research questions.

All studies present some features of self-adaptive systems, such as context, change and adaptation concepts, but no one covers all the concepts involved in SAS domain.

The little quantity of selected papers (23 in total) indicates there are few works concerning ontologies for self-adaptive systems. The SLRs allow us to know the concepts used in this studies. We perceive the limitation of these works in representing concepts for a self-adaptive system, motivating a creation of a more complete ontology for this kind of system.

Figure 16 shows the core ontology presented by Qureshi, Jureta and Perini (2011), it is presented in S6 study.

Figure 16 – Core Ontology for SAS.



Source: Qureshi, Jureta and Perini (2011).

S6 study presents the core ontology to requirements engineer for SAS (QURESHI; JURETA; PERINI, 2011), it is formed by the following concepts: **Goal**, **Softgoal**, **Quality Constraint**, **Task**, **Evaluation**, **Context**, **Resource** and **Domain Assumption**; and the following relationships: *Is-a*, *Depend-on*, *Refer-to*, *Relegate* and *Influence*. It added two new concepts (**Context** and **Resource**) and two new relations (*Relegation* and *Influence*) to the core ontology for Requirements Engineering proposed by Jureta, Mylopoulos and Faulkner (2008). These extensions were performed to accommodate the changes that might occur at runtime, which not only demands adaptation (i.e. dynamically changing from one requirements problem to another) but also requires an update to the specification (i.e. refinement of requirements).

S6 study was the only ontology for requirements for SAS found in the literature. It presents some limitations. These limitations will be known throughout this chapter.

In the sequel, we analyze some modeling languages in order to know their limitations and do perceive if they capture context, adaptation and modeling dimensions.

3.4 Analysis of Requirements Modeling Languages for SAS

We modeled a self-adaptive news service example using three modeling languages that involve a goal modeling language proposed specifically to specify SAS: Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), Adaptive RML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PIMENTEL et al., 2014). During this activity, we no-

ticed that these languages have several interesting concepts, like **Failure** and **Error** (Tropos4AS), but they do not cover all the concepts necessary to specify the requirements of SAS (SOARES et al., 2016) like the **Mechanism** concept from the modeling dimensions (ANDERSSON et al., 2009).

The Software Engineering for Adaptive and Self-Managing Systems (SEAMS) research community³ provides a repository of examples, challenge problems, and solutions that the software engineering for self-adaptive systems community can use to motivate research, exhibit solutions and techniques, and compare results.

For this work, we chose the Znn.com news service since it has adaptive requirements adequate to our objectives. The main goal of ZNN.com is to serve multimedia news content to its customers. Therefore, we present the Znn.com description.

“Znn.com uses a load balancer to balance requests across a pool of replicated servers, the size of which is dynamically adjusted to balance server utilization against service response time. A set of client processes makes stateless content requests to one of the servers. Let us assume we can monitor the system for information such as server load and the bandwidth of server-client connections. Assume further that we can modify the system, for instance, to add more servers to the pool or to change the quality of the content. We want to add self-adaptation capabilities that will take advantage of the monitored system and adapt the system to fulfill Znn.com objectives. The business objectives at Znn.com are to serve news content to its customers within a reasonable response time range while keeping the cost of the server pool within its operating budget. From time to time, due to highly popular events, Znn.com experiences spikes in news requests that it cannot serve adequately, even at maximum pool size. To prevent unacceptable latencies, Znn.com opts to serve minimalist textual content during such peak times instead of providing its customers zero service. The Znn.com system administrators (sys-admins) adapt the system using two actions: adjust the server pool size or switch content mode. When the system comes under high load, the sys-admins may increase the server pool size until a cost-determined maximum is reached, at which point the sys-admin would switch the servers to serve textual content. If the system load drops, the sys-admin may switch the servers back to multimedia mode to make customers happy in combination with reducing the pool size to reduce operating cost. The adaptation decision is determined by observations of overall average response time versus server load. Specifically, four adaptations are possible, and the choice depends not only on the conditions of the system, but also on business objectives: Switch the server content mode from multimedia to textual; Switch the server content mode from textual to multimedia; Increment the server pool size; and Decrement the server pool size.”

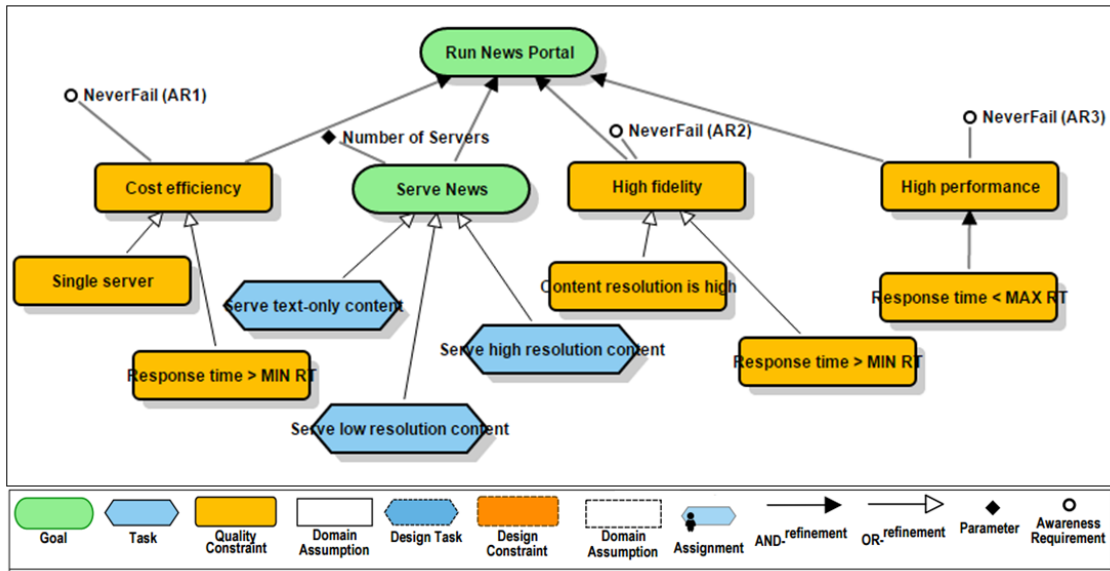
DGM

The Design Goal Model for the ZNN.com example is presented in Figure 17. In this model example, the “Run News Portal” goal needs the “Serve News” goal to be satisfied

³ See <https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/>.

and the “Cost efficiency”, “High fidelity” and “High performance” quality constraints to be met. The “Serve News” goal is satisfied if one of the three tasks is satisfied, “Serve high resolution content”, “Serve low resolution content” or “Serve text-only content”. The “Cost efficiency” quality constraint limits the server or the response time. The “High fidelity” quality constraint limits the response time or the content resolution and the “High performance” quality constraint limits the response time. The DGM allows modeling the “Number of Servers” as parameter and three awareness requirements like “NeverFail (AR1)”, “NeverFail (AR2)” and “NeverFail (AR3)” related to the following quality constraints: “Cost efficiency”, “High fidelity” and “High performance” respectively.

Figure 17 – Goal model for the ZNN.com modeled using the Design Goal Model.

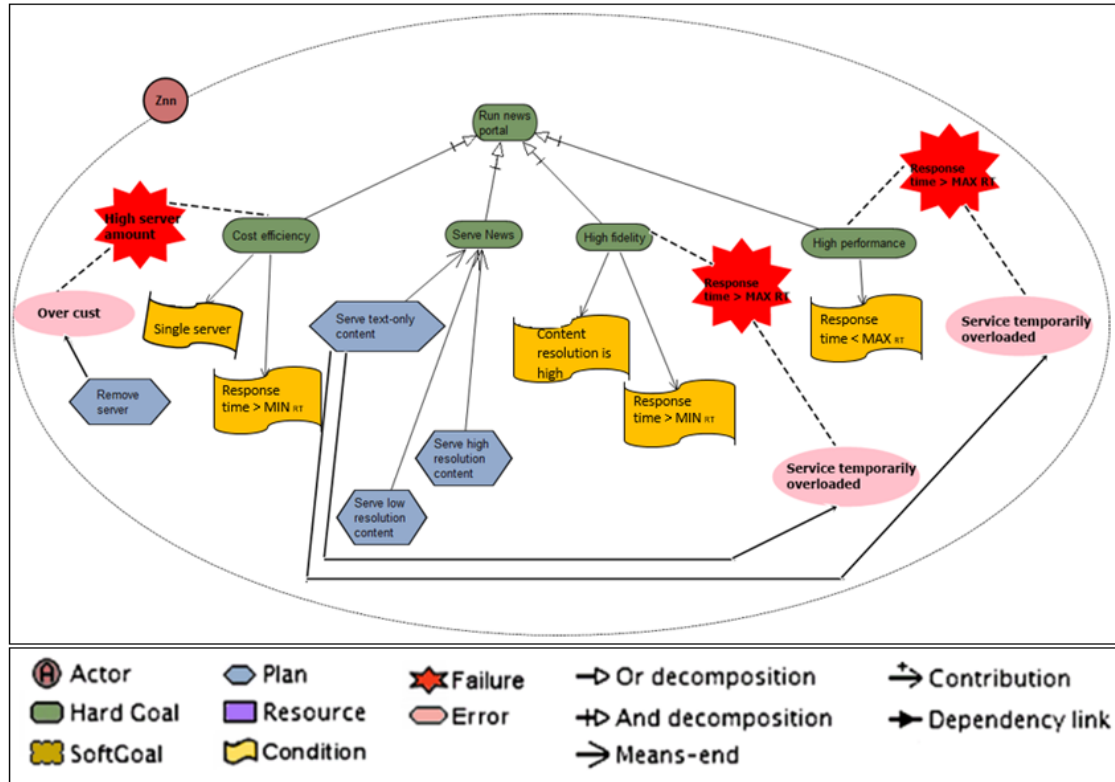


Source: Pimentel (2015).

Tropos4AS

The Znn.com example modeled using the Tropos4AS is illustrated in Figure 18. In this model, the “Run News Portal” goal depends on the “Serve News”, “Cost efficiency”, “High fidelity” and “High performance” goals to be satisfied. The “Serve News” goal is satisfied if one of the three tasks is satisfied: “Serve high resolution content”, “Serve low resolution content” or “Serve text-only content”. The “Cost efficiency” goal has two conditions, “single server” and “response time > MINRT”. The “High fidelity” goal has two conditions, “content resolution is high” and “response time > MINRT” and the “High performance” goal has a condition “response time < MAXRT”. The Tropos4AS allows anticipating a failure as “High server amount” linked to “Cost efficiency” goal. This failure can bring an error as “over cost”, allowing the system to adapt by performing the “Remove server” task. The “High fidelity” and the “High performance” goals also can anticipate a failure as “Response time > MAXRT” bring towards an error like “Service temporarily overloaded” with the adaptation “Serve text-only content” task.

Figure 18 – Goal model for the ZNN.com modeled using the Tropos4AS.



Source: Own.

AdaptiveRML

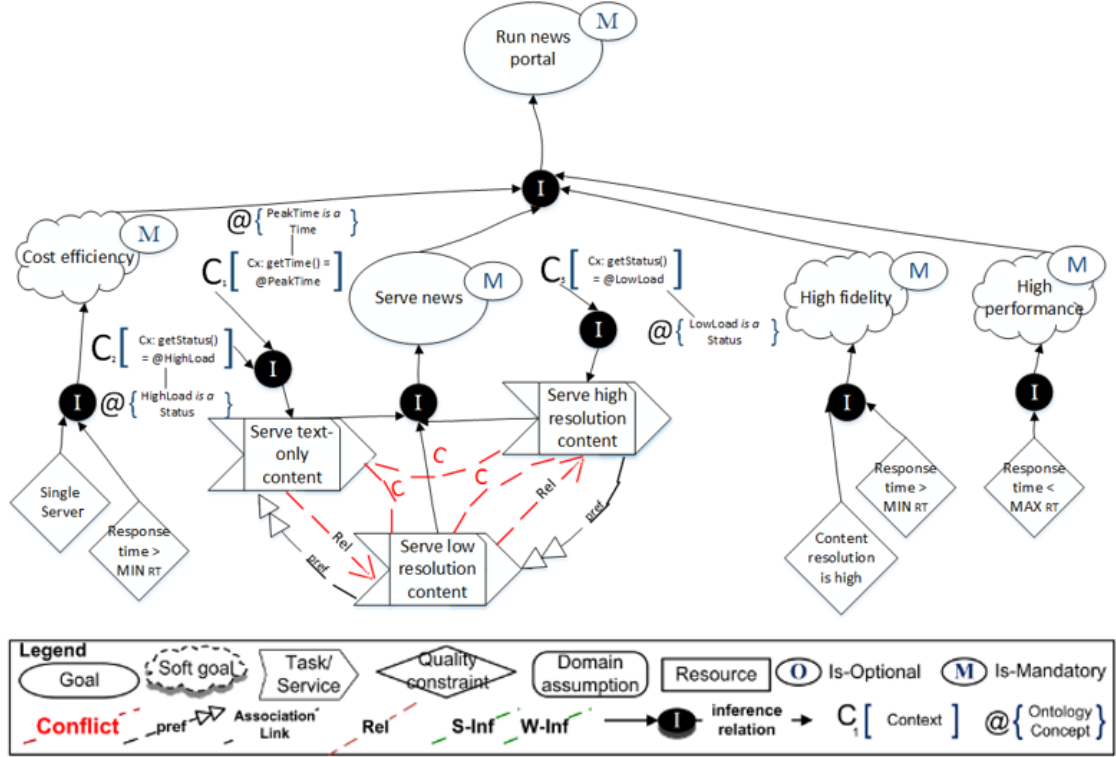
As represented in Figure 6, the Znn.com example modeled using the AdaptiveRML has the following structure: The “Run News Portal” goal needs the “Serve News” goal and the “Cost efficiency”, “High fidelity” and “High performance” softgoals to be satisfied. The “Serve News” goal in turn is satisfied if one of the three tasks be realized, “Serve high resolution content”, “Serve low resolution content” or “Serve text-only content”. The “Cost efficiency” softgoal limits the server or the response time. The “High fidelity” softgoal limits the response time or the content resolution and the “High performance” softgoal limits the response time.

The influence relation does not apply for the Znn.com example. In Figure 19, “Serve text-only content” has a relegate relation with “Serve low resolution content”, which in turn has a relegate relation with “Serve high resolution content”, when context conditions changes. The “Serve text-only content” task has two contexts: C1 - PeakTime and C2 - HighLoad. PeakTime is a time context and HighLoad is a status context (from server) and they both are ontology concepts. The “Serve high resolution content” has one context, the LowLoad, a status context.

Conflicts can arise between inconsistent or contradictory requirements or tasks node. Figure 19 shows conflict between the "Serve text only content", "Serve low resolution content" and "Serve high resolution content" tasks. The preference relation models when it

is more desirable satisfying one requirement than other. Figure 19 presents the preference relation between the "Serve high resolution content" and "Serve low resolution content" tasks and also between the "Serve low resolution content" and "Serve text-only content" tasks.

Figure 19 – Goal model for the ZNN.com modeled in AdaptiveRML.



Source: Own.

3.4.1 GORE Languages versus Modeling Dimensions

We analyzed these languages in order to perceive whether they can capture context (JEONG-DONG; SON; BAIK, 2012), adaptation and modeling dimensions (ANDERSSON et al., 2009) concepts.

The AdaptiveRML has a **Context** element which facilitates the representation of context in this language. The other languages do not have a **Context** element, that makes it harder to represent **Context** in these languages.

We also look for elements to capture monitoring, analysis, planning and execution activities. Tropos4AS present a **plan** element, to be used in the planning activity, DGM present a **task** element, similar to **plan** element and a **AwarenessReq** used to monitoring. There are no others constructs to represent the other adaptation activities in the Tropos4AS neither the AdaptiveRML or Design Goal Model.

To analyze whether the three languages can represent the modeling dimensions, we create a table for better viewing. Table 10 shows all modeling dimensions (rows) and the

goal-oriented modeling languages to SAS (columns). Each pair of modeling dimension and goal-oriented modeling language is divided into two columns. The first column, labeled as "S" (Satisfy), indicates if the language attends or not the dimension. The second column, labeled as "C" (Concept), indicates the concept that represents that dimension in the language. In the case of the "S" column, the possible answers are "Y" for Yes, "N" for No and "-" for undecided.

The cell marked with "Y" indicates that the goal modeling language has elements capable of representing that modeling dimension. The cell marked with "N" indicates that the goal modeling language has no elements capable of representing that modeling dimension. The cell is marked with a "-" indicates that it is not clear if the goal modeling language is capable of representing the modeling dimension.

To perform that analysis, firstly, we investigated each dimension modeling with the intention of fully understand the concepts presented for them. Then, the elements and concepts of Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), Adaptive RML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PIMENTEL, 2015) were analyzed. The results of the comparative study are presented in Table 10. From the general analysis presented in this table, we can notice that none of the goal modeling languages can represent all modeling dimensions of the four groups (Goals, Change, Mechanisms, and Effects).

Almost all elements in the Goal dimension have representative elements and concepts in the modeling languages (not surprisingly, since they are GORE languages). Some elements in the change and effects dimensions can be represented by these languages. Most elements in the Mechanisms dimension are not clearly represented in at least two goal-oriented modeling languages from the three investigated.

Some concepts' descriptions of the goal-oriented modeling languages look like the modeling dimensions' description, but do not cover all elements of the modeling dimensions (presented in Andersson et al. (2009)), and then they were marked with "-". Accordingly, it is not clear if they cannot be represented or if they depend of the software engineer experience to model the dimension, compromising the documentation and evolution of the system.

With this comparative study, it is possible to conclude that the three goal modeling languages studied have difficulties to represent the Mechanisms dimension, but the other groups (Goal, Change and Effects) have a good representation in these languages. However, it is necessary to know what information needs to be elicited to model the system.

These languages can capture some concepts that represent some of the various characteristics of SAS such as goal, softgoal, task/service, quality constraint, domain assumption, assignment, parameter, awareness requirement, plan, resource, condition, failure, error, conflict, preference, relegation, influence, context and ontology concept. However, there are some fundamental characteristics to the representation of SAS that are not covered by the languages. Every SAS comply with an adaptation cycle to complete their

Table 10 – GORE Languages x Modeling Dimensions.

Modeling Dimensions	Goal Modeling Languages					
	Tropos4AS		AdaptiveRML		Design Goal Model	
Goals	S	C	S	C	S	C
Evolution	-		-		Y	Evolution Requirements
Flexibility	Y	Softgoal	Y	Soft goal	Y	Softgoal
Duration	Y	Condition	Y	Quality Constraint	Y	Quality Constraint
Multiplicity	Y	Hard goal	Y	Goal	Y	Goal
Dependency	Y	Dependency	Y	Inference Relation	Y	AND-Refinement
Change	S	C	S	C	S	C
Source	Y	Hard goal	-		Y	Awareness Requirements
Type	Y	Goal	Y	Goal	Y	Goal
Frequency	Y	Condition	Y	Quality Constraint	Y	Quality Constraint
Anticipation	Y	Failure/Error	N		Y	Awareness Requirements
Mechanisms	S	C	S	C	S	C
Type	-		-		-	
Autonomy	-		N		N	
Organization	-		-		-	
Scope	-		Y	Context	-	
Duration	-		-		-	
Timeliness	N		N		N	
Triggering	N		N		N	
Effects	S	C	S	C	S	C
Criticality	Y	Contribution	Y	Influence Relation	Y	AND/OR-Refinement
Predictability	Y	Error/Plan	-		-	
Overhead	Y	Contribution	Y	Influence Relation	Y	AND/OR-Refinement
Resilience	Y	Error/Plan	N		N	

Source: Own.

adaptation routine and context is part of SAS reality. Therefore, it is necessary that concepts such as sensor, actuator, monitor, evaluation, change, mechanism and effects are captured and represented in any SAS specification. The concepts regarding monitoring (sensor, actuator and monitor) and measurements (evaluation) are essential to SAS, as well as the concepts regarding adaptation (change, mechanism and effects).

In summary, the three languages present the following concepts, according to their elements: **Goal**, **Softgoal**, **Plan**, **Task**, **Quality Constraint**, **Domain Assumption**, **Actor**, **Resource**, **Context**, **Failure**, **Error**, **Conflict** and **Condition**. We note that the **Condition** concept can be interpreted as a **Quality Constraint**.

3.4.2 GORE Languages versus the Core Ontology for SAS

Although, there is a core ontology for SAS (QURESHI; JURETA; PERINI, 2011), it does not cover the concepts present in the GORE modeling languages for SAS, and it also does

not cover the modeling dimensions or the adaptation cycle.

The core ontology has the following concepts: **Goal**, which it is specialized on **Functional Goal (FG)**, **Quality Constraint (QC)** and **Softgoal (SG)**, **Task**, **Domain Assumption**, **Evaluation**, **Context** and **Resource**.

Table 11 presents the Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008) elements compared to the core ontology (QURESHI; JURETA; PERINI, 2011) concepts. We can perceive that not all concepts of the core ontology can be represented by the language neither all language elements have a representation in the core ontology.

Tropos4AS has the following concepts: **Hard Goal**, **Softgoal**, **Plan**, **Resource**, **Condition**, *Or decomposition*, *And decomposition*, *Means-end*, *Contribution*, *Dependency link*, **Actor**, **Failure** and **Error**.

The Tropos4As elements are listed in the row and the concepts of the core ontology are in the column. Each cell of the table marked with "X" means that the element of the Tropos4AS is represented in the core ontology. The Hard Goal, Softgoal, Plan, Resource, Condition, Or decomposition, And decomposition, Means-end, Contribution and Dependency link are represented in the core, while Actor, Failure and Error are not. The Hard Goal is represented by FG, Softgoal by SG, Plan by Task, Resource by Resource, Condition by QC, Or decomposition by Domain Assumption, And decomposition by SG, Means-end by FG, Contribution by SG and Dependency link by FG as presented in Table 11. On the other hand, **Evaluation** and **Context** are not represented in Tropos4AS.

Table 11 – Tropo4AS x Core Ontology.

Tropos4AS	Core Ontology for SAS Concepts							
	Goal			Task	Domain Assumption	Evaluation	Context	Resource
	FG	QC	SG					
Actor								
Hard Goal	X							
Softgoal			X					
Plan				X				
Resource								X
Condition		X						
Failure								
Error								
Or decomposition					X			
And decomposition			X					
Means-end	X			X				
Contribution			X					
Dependency link	X							

Source: Own.

Table 12 presents the AdaptiveRML (QURESHI; JURETA; PERINI, 2012) elements against the core ontology (QURESHI; JURETA; PERINI, 2011) concepts. We can per-

ceive that not all concepts of the core ontology can be represented by the language neither all language concepts have a representation in the core ontology.

AdaptiveRML has the following elements: **Goal**, **Softgoal**, **Task/Service**, **Quality Constraint**, **Domain Assumption**, **Resource**, *Is-Optional*, *Is-Mandatory*, *inference relation*, **Context**, **Ontology concept**, **Conflit** and *pref*, *Association Link*, *Rel*, *S-Inf* and *W-Inf*.

Table 12 shows the AdaptiveRML elements listed in the row and the concepts of the core ontology in the column. Each cell of the table marked with "X" means that the element of the AdaptiveRML is represented in the core ontology. The Goal, Softgoal, Task/Service, Quality Constraint, Domain Assumption, Resource, Is-optional, Is-mandatory, Context, and Association link are represented in the core, while Inference Relation, Ontology Concept, Conflict, *pref*, *Rel*, *S-Inf* and *W-Inf* are not. The Goal is represented by FG, Softgoal by SG, Task/Service by Task, Quality Constraint by QC, Domain Assumption by Do-main Assumption, Resource by Resource, Is-Optional and Is-Mandatory by SG, Context by Context, Association Link by FG as detailed in the Table 12. On the other hand **Evaluation** concept is not represented in AdaptiveRML.

Table 12 – AdaptiveRML x Core Ontology.

AdaptiveRML	Core Ontology for SAS Concepts							
	Goal			Task	Domain Assumption	Evaluation	Context	Resource
	FG	QC	SG					
Goal	X							
Softgoal			X					
Task/Service				X				
Quality constraint		X						
Domain assumption					X			
Resource								X
Is-Optional			X					
Is-Mandatory			X					
inference relation								
Context							X	
Ontology Concept								
Conflict								
<i>pref</i>								
Association Link	X							
<i>Rel</i>								
<i>S-Inf</i>								
<i>W-Inf</i>								

Source: Own.

Table 13 presents the Design Goal Model (PIMENTEL, 2015) elements against the core ontology (QURESHI; JURETA; PERINI, 2011) concepts. We can perceive that not

all concepts of the core ontology can be represented by the language neither all language elements is based on concept of the core ontology.

Design Goal Model has the following elements: **Goal, Task, Quality Constraint, Design Task, Design Quality/Constraint, Domain Assumption, And-refinement, Or-refinement, Parameter, Awareness Requirements, Flow Expressions.**

Table 13 shows the Design Goal Model elements listed by row and the concepts of the core ontology in the column. Each cell of the table marked with "X" means that the element of the Design Goal Model is represented in the core ontology. The Goal, Task, Quality Constraint, Design Task, Design Quality/Constraint, Domain Assumption, And-refinement and OR-refinement are represented in the core, while Parameter, Awareness Requirements and Flow Expressions are not. The Goal is represented by FG, Task by Task, Quality Constraint by QC, Design Task by Task, Design Quality Constraint by QC, Domain Assumption by Domain Assumption, AND-refinement and OR-refinement by SG as detailed in Table 13.

Table 13 – Design Goal Model x Core Ontology.

	Core Ontology for SAS Concepts							
Design Goal Model	Goal			Task	Domain Assumption	Evaluation	Context	Resource
	FG	QC	SG					
Goal	X							
Task				X				
Quality Constraint		X						
Design Task				X				
Design Quality/Constraint		X						
Domain Assumption					X			
AND-refinement			X					
OR-refinement			X					
Parameter								
Awareness Requirements								
Flow Expressions								
Evolution Requirements								

Source: Own.

The only core ontology concept not captured by any language is the **Evaluation** concept. On the other hand, the languages can capture several concepts that are not present in the core ontology. Besides, the core ontology (QURESHI; JURETA; PERINI, 2011) does not cover the modeling dimensions (ANDERSSON et al., 2009). In fact, we compared each element of the modeling dimensions' groups to the core ontology concepts and just the goal and change groups can be represented by the core ontology, but the mechanisms and effects groups cannot, as presented in Table 14.

We concluded that the core ontology for self-adaptive systems (QURESHI; JURETA; PERINI, 2011) does not cover all the elements necessary for the specification of self-

Table 14 – Modeling Dimensions x Core Ontology.

Modeling Dimensions for SAS	Core Ontology for SAS
Goals are objectives the system under consideration should achieve. The dimensions related to goals are: evolution, flexibility, duration, multiplicity and dependency.	Goal (Functional Goal, Quality Constraint, Softgoal)
Change is the cause for adaptation. Whenever the system's context changes, the system has to decide whether it needs to adapt. The dimensions related to change are: source, type, frequency, anticipation.	Goal (Functional Goal, Quality Constraint, Softgoal) and Context.
What is the reaction of the system towards change. They are related to the adaptation process itself. The dimensions of mechanisms are: type, autonomy, organization, scope, duration, timeliness and triggering.	-
What is the impact of adaptation upon the system. The effects dimensions are: criticality, predictability, overhead and resilience.	-

Source: Own.

adaptive systems requirements. It lacks essential concepts for the self-adaptive systems domain, such as the modeling dimensions that are considered by the SAS community as essential for modeling SAS. In fact, the work of Andersson et al. (2009) has 116 citations according to Google Scholar.

3.5 Conclusion

This Chapter presented two systematic literature reviews to identify the studies about ontologies for self-adaptive systems and context-aware systems. We identified 23 studies that present some knowledge representation for these kind of systems. Some research questions were answered and several concepts were identified. We also presented an analysis of requirements modeling languages for SAS. It offers a knowledge base to create our ontology and elements to evaluate the comprehensiveness criteria of our ontology.

We know that self-adaptive systems are not easy to specify because of their dynamic features. We observed that each of the three modeling languages analyzed presents different elements from each other but they are unable to capture some important characteristics of these systems. Moreover, the core ontology for SAS, proposed by Qureshi, Jureta

and Perini (2012) cannot represent all the modeling dimensions neither the MAPE-K cycle or the context. The modeling dimensions mean the points of variation in self-adaptive systems, a very important feature to adaptation. The MAPE-K represents the adaptation cycle and all SAS are context dependable.

The works obtained by the SLRs also do not cover all of these characteristics. Hence, there is a need for a standard set of concepts to be identified and represented in the specification of SAS. There is a need for a knowledge body to capture every feature of SAS, facilitating the SAS requirements elicitation and specification. Therefore, we believe that an ontology is helpful to organize the knowledge about self-adaptive systems, in order to facilitate: (i) early identification and elicitation of system requirements, parameters, context, adaptive actions, monitoring and measurements; (ii) modeling of complex SAS with, possibly, varying properties and alternatives; (iii) sharing common understanding of self-adaptive knowledge among different stakeholders. Therefore, in this thesis, we define an ontology to aid the requirements elicitation and specification of self-adaptive systems covering the elements present in the GORE modeling languages, MAPE-K adaptation cycle and the modeling dimensions.

4 ONTO4SASGORE: AN ONTOLOGY FOR SELF-ADAPTIVE SYSTEMS GOAL-ORIENTED REQUIREMENTS

In this Chapter, we present the ontology proposed to aid the elicitation and specification of goal-oriented requirements for self-adaptive systems. The three methodologies presented in Section 2.3.3 have some intersection points, and some unique features. As intersection points there is the fact that they are very useful to create ontologies from scratch. As unique features, there is the focus on competence questions stated by Uschold and Gruninger (1996), the consideration of the differences between reference and operational ontology defined by SABiO (FALBO, 2014), and the good division and description of the ontology development phases proposed by METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997). Therefore, we decided to use the best practice of the three methodologies to create our ontology. Documentation is a support process in SABiO and METHONTOLOGY, and we decided to use it as a part of each ontology development phase, and we used a wiki to facilitate the documentation. In Section 4.1, we present the ontology purpose identification and ontology requirements elicitation and in Section 4.2, we present the works used to knowledge acquisition to our ontology. In Section 4.3, we present the ontology conceptualization, in Section 4.4, the formalization of the competence questions and axioms from the ontology. The implementation phase is presented in Section 4.5 and lastly, evaluation of the ontology is in Section 4.6. In Section 4.7, we conclude this Chapter.

4.1 Purpose Identification and Requirements Elicitation

This phase has the same name of the first activity of SABiO (FALBO, 2014), but it has some resemblance to the objective and scope phases of METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997) and the motivating scenarios and informal competence questions phases of Uschold and Gruninger (1996). In this phase, we can set the objective and scope for the Onto4SASGORE, finding the motivating scenarios and creating the informal competence questions. As results of this phase, we have the objective, scope, motivating scenarios and the informal competence questions for the proposed ontology.

4.1.1 Objective

The main objective of our ontology is to support requirements engineers to discover goal-oriented requirements during the elicitation activity in a development process of SAS, as well as specifying the elicited requirements.

Onto4SASGORE provides a knowledge base about the concepts related to self-adaptive domain. This ontology was created to support the elicitation and specification of GORE requirements for self-adaptive systems. It can also help the development of modeling languages and tools. The ontology is a knowledge body for self-adaptive domain in the literature. It should harmonize the self-adaptive domain terminology spread in the existing modeling languages for SAS and help requirements engineers to communicate with each other.

4.1.2 Scope

The scope of this ontology is the SAS domain, its requirements (functional, nonfunctional, and referring to changes) and relations among them.

4.1.3 Motivating Scenarios

The following motivating scenarios demonstrate cases where a ontology for adaptation can help to specify the system:

- Cleaner Robot (Appendix F);
- Dispatching Ambulance System (SILVASOUZA; MYLOPOULOS, 2015);
- Server of multimedia content (ZNN.com - Section 3.4);

4.1.4 Informal Competence Questions

General competence questions can generate other more specific competence questions, so that, in this activity, we create the general competence questions represented by GCQ and in the formalization phase (Section 4.4), we create all competence questions and formalize them.

GCQ01: What is the system goal?

GCQ02: How does the system achieve its goal(s)?

GCQ03: How does the system self-adapt to achieve its goal(s)?

GCQ04: Do environment changes have impact on the system?

GCQ05: What are the system changes that can change an action?

GCQ06: What happens when the self-adaptability fails?

GCQ07: Any change occur because of the time?

GCQ08: Any change occur because of the location?

GCQ09: How does the system act when an error appears?

GCQ10: How does the system act with non-anticipated changes?

After the creation of these ten general competence questions, we develop them in more specific competence questions, as follows.

CQ01: What are the sensors in the scope of the system?

CQ02: What is the sensor status?

CQ03: What are the effectors in the scope of the system?

CQ04: What are the effects of the system?

CQ05: What are the actors of the system?

CQ06: What are the roles played by the actor?

CQ07: What are the actions held by the actor?

CQ08: What are the actions of the system?

CQ09: What are the Effects of the action?

CQ10: What is the predictability of the effect?

CQ11: What is the resilience of the effect?

CQ12: What is the criticality of the effect?

CQ13: What is the overhead of the effect?

CQ14: What are the resources of the system?

CQ15: What are the Plans of the system?

CQ16: What are the Change Plans of the system?

CQ17: What are the Symptoms of the system?

CQ18: What are the Change Requests of the system?

CQ19: What are the goals of the system?

CQ20: What are the Functional Goals of the system?

CQ21: What are the actions that achieve the functional goal X?

CQ22: What are the Non Functional Goals of the system?

CQ23: What are the actions that contribute to the Non functional goal?

CQ24: What are the Hardgoals of the system?

CQ25: What are the Tasks of the system?

CQ26: What are the Softgoals of the system?

CQ27: What are the Quality Constraints of the system?

CQ28: What is the duration of the goal?

CQ29: What is the flexibility of the goal?

CQ30: What is the dependency of the goal?

CQ31: What is the evolution of the goal?

- CQ32: What are the inputs of the system?
- CQ33: What is the input of the monitor?
- CQ34: What is the input of the analyze?
- CQ35: What is the input of the plan?
- CQ36: What is the input of the execute?
- CQ37: What is the input of the effector?
- CQ38: What are the outputs of the system?
- CQ39: What is the output of the sensor?
- CQ40: What is the output of the monitor?
- CQ41: What is the output of the analyze?
- CQ42: What is the output of the plan?
- CQ43: What is the output of the execute?
- CQ44: What are the preconditions of the Action X?
- CQ45: What are the preconditions of the monitor?
- CQ46: What are the preconditions of the analyze?
- CQ47: What are the preconditions of the plan?
- CQ48: What are the preconditions of the execute?
- CQ49: What are the changes of the system?
- CQ50: What is the frequency of the change?
- CQ51: What is the type of the change?
- CQ52: What is the anticipation of the change?
- CQ53: What is the source of the change?
- CQ54: What are the contexts involved in the system?
- CQ55: What is the goal of the context?
- CQ56: What is the location of the context?
- CQ57: What is the time of the context?
- CQ58: What is the action of the context?
- CQ59: What is the role of the context?
- CQ60: What is the status of the context?
- CQ61: What are the mechanisms of the system?
- CQ62: What is the type of the mechanism?
- CQ63: What is the timeliness of the mechanism?
- CQ64: What is the triggering of the mechanism?
- CQ65: What is the autonomy of the mechanism?
- CQ66: What is the organization of the mechanism?
- CQ67: What is the scope of the mechanism?
- CQ68: What is the duration of the mechanism?
- CQ69: What is the interface of the mechanism?
- CQ70: What is the goal of the plan?

- CQ71: How many goals are there in the system?
- CQ72: How many functional goals are there in the system?
- CQ73: How many non-functional goals are there in the system?
- CQ74: How many personal goals are there in the system?
- CQ75: How many role goals are there in the system?
- CQ76: How many softgoals are there in the system?
- CQ77: How many quality constraints are there in the system?
- CQ78: How many changes are there in the system?
- CQ79: How many mechanisms are there in the system?
- CQ80: How many mechanisms are there in Change X?
- CQ81: What are the mechanisms of Change X?
- CQ82: How many effects are there in the system?
- CQ83: How many effects does the Change X have?
- CQ84: What are the effects produced by Change X?
- CQ85: How many actors are there in the system?
- CQ86: How many contexts are there in the system?
- CQ87: How many resources are there in the system?
- CQ88: How many sensors are there in the system?
- CQ89: How many effectors are there in the system?
- CQ90: What are the goals of static evolution?
- CQ91: What are the goals of dynamic evolution?
- CQ92: How does the system achieve the goals?
- CQ93: How does the system achieve the goal X?

4.2 Knowledge Acquisition

Knowledge Acquisition is a phase in METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997) and a support process in SABiO (FALBO, 2014). In this phase, we can look for documents, books, papers, others ontologies, taxonomies, international standards and material about the domain that can be used as knowledge source to the proposed ontology. As result of this phase, we have enough material to be used as knowledge source to build the ontology.

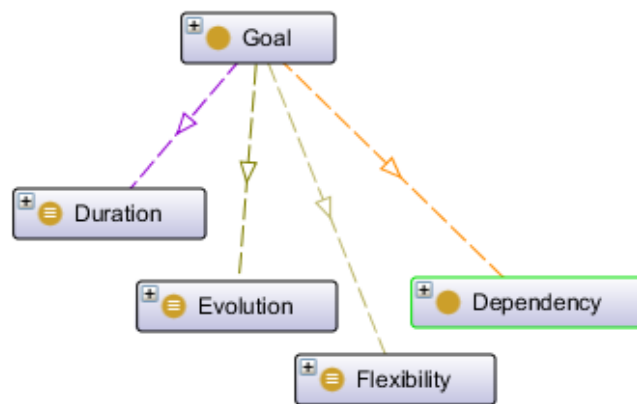
To compose the knowledge acquisition, we consulted the roadmap papers presented in Cheng et al. (2009b) and in Lemos et al. (2011). They do a mapping of the software engineering for self-adaptive systems and they present more than a thousand of citations, which are relevant works to self-adaptive domains. The second roadmap explains about the modeling dimensions (ANDERSSON et al., 2009) needed when modeling SAS and the MAPE-K adaptation cycle (COMPUTING et al., 2006). Besides these works, we used the core ontology to preforming RE for SAS (QURESHI; JURETA; PERINI, 2011) since it

is, for the best of our knowledge, the unique core ontology for this domain proposed on literature.

SAS are dependent on context. So, we also used an ontology for context. We choose the CA5W1HOnto (JEONG-DONG; SON; BAIK, 2012) because it considers the 5W1H technique. Salehie and Tahvildari (2009) advocate that 5W1H facilitates the requirements elicitation for SAS. We used the goal-oriented modeling languages proposed for SAS, namely Tropos4AS (MORANDINI; PERINI; MARCHETTO, 2011), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PIMENTEL et al., 2014). We have also used the SLR's results presented on Section 3. The core ontology (QURESHI; JURETA; PERINI, 2011) is a work selected in the SLR about the SAS domain (Section 3.1) and the CA5W1HOnto (JEONG-DONG; SON; BAIK, 2012) was selected in the SLR about the context-aware systems domain (Section 3.2). Thus, our knowledge acquisition sources were composed by studies in literature about software engineering for self-adaptive systems, ontologies, twenty three papers from the SLR about ontologies for SAS and CAS and three GORE modeling languages for SAS.

AS presented in Section 2.1.2, Andersson et al. (2009) defined a set of modeling dimensions for SAS that can represent a wide range of system properties. They are classified in four groups. The first group is associated with self-adaptability aspects of the system goals, the second one involves the dimensions associated with the causes of self-adaptation, the third group addresses the dimensions associated with the mechanisms to achieve self-adaptability, and the last group comprises the dimensions related to the effects of self-adaptability upon a system. The dimensions related to **Goals** are **Evolution**, **Flexibility**, **Duration**, and **Dependency** as presented in Figure 20. The figures from 20 to 23 were generated using Protégé.

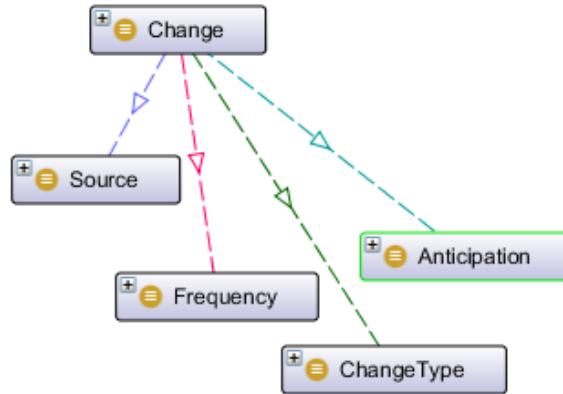
Figure 20 – Goal Modeling Dimensions.



Source: Own.

Figure 21 presents the dimensions related to **Change**, they are **Source**, type (**ChangeType**), **Frequency**, and **Anticipation**.

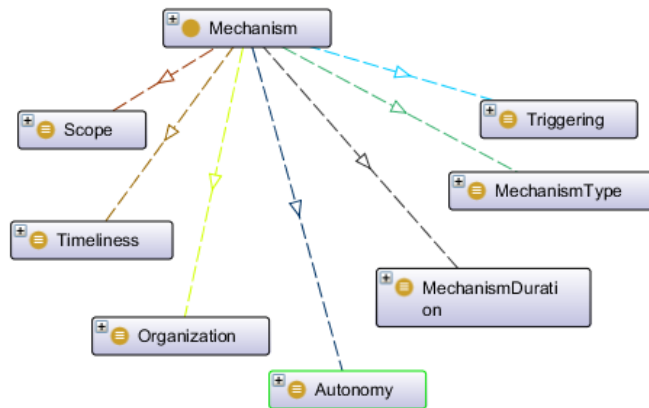
Figure 21 – Change Modeling Dimensions.



Source: Own.

The dimensions of **Mechanisms** are type (**MechanismType**), **Autonomy**, **Organization**, **Scope**, duration (**MechanismDuration**), **Timeliness** and **Triggering**, they can be visualized in Figure 22.

Figure 22 – Mechanism Modeling Dimensions.



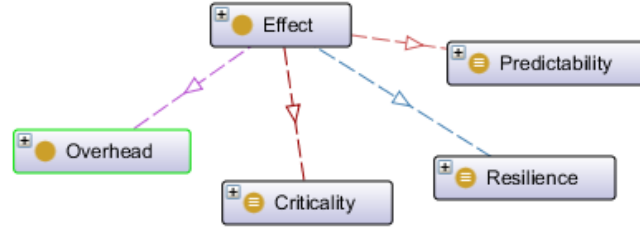
Source: Own.

The **Effects** dimensions are **Criticality**, **Predictability**, **Overhead** and **Resilience**, as depicted in Figure 23.

The study performed by Soares et al. (2016) presented a comparative study of three goal modeling languages proposed for self-adaptive systems to discover whether they can represent all the modeling dimensions for SAS (presented in Section 3.4). The study concluded the modeling languages studied, Tropos4AS (MORANDINI; PERINI; MARCHETTO, 2011), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PI-MENTEL et al., 2014) cannot represent all the modeling dimensions for SAS.

We also considered the MAPE-K (monitor, analyze, plan, execute) cycle (KEPHART; CHESS, 2003), which is an adaptation cycle and any self-adaptive system operates based

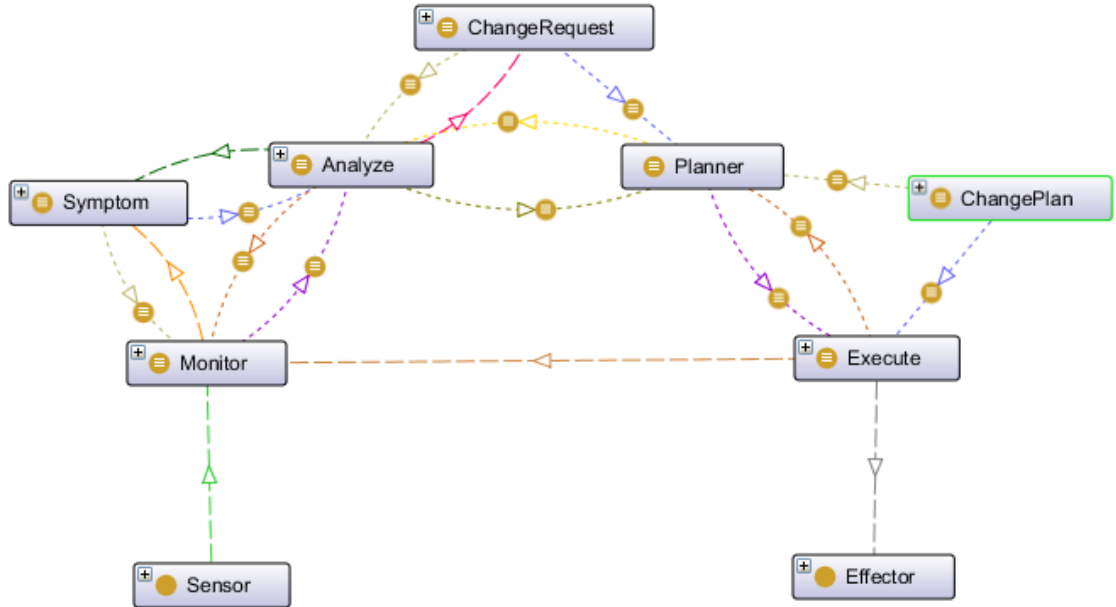
Figure 23 – Effect Modeling Dimensions.



Source: Own.

on an adaptation cycle. Figure 24 presents the MAPE cycle. **Sensor** has a relationship with **Monitor**. **Monitor** is related with **Analyze** through the **Symptom** that is a **Resource** as **Change Request** that does the link between **Analyze** and **Plan**. **Plan** and **Execute** are linked by **Change Plan**. **Execute** has a relationship with **Monitor** and **Effector**.

Figure 24 – MAPE-K adaptation cycle.

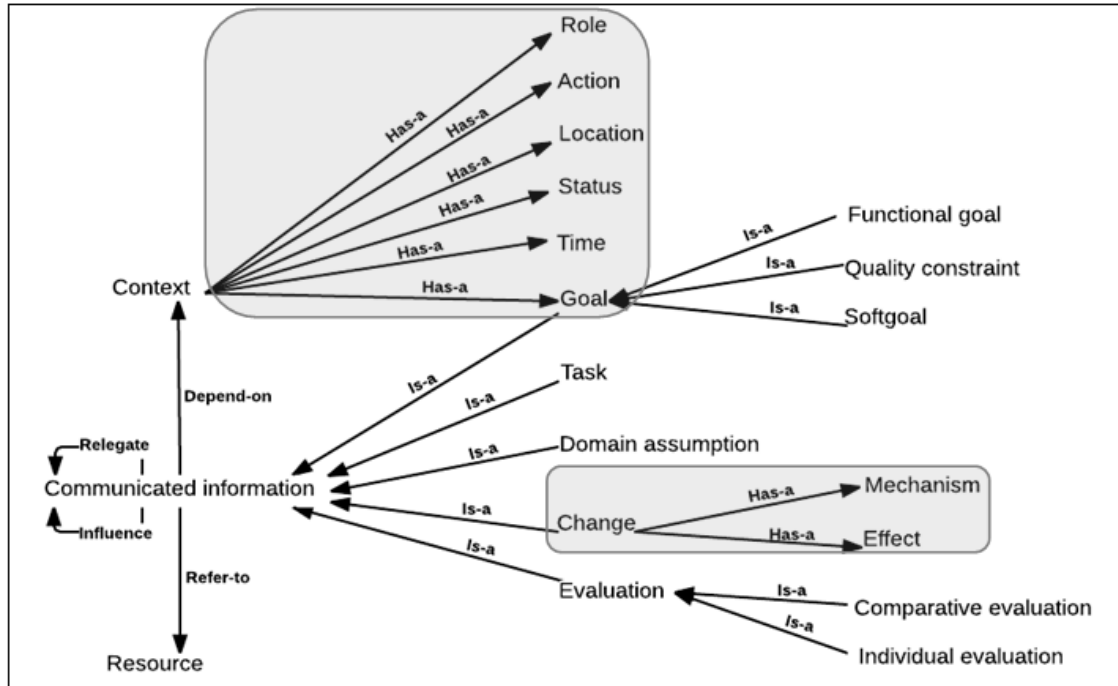


Source: Own.

The core ontology by Qureshi, Jureta and Perini (2011) has the following main concepts: **Goal**, **Task**, **Evaluation**, **Domain Assumptions**, **Resource** and **Context** and to the best of our knowledge, it is the unique formal core ontology to perform RE for SAS. The work presented by Soares et al. (2016) presents an extension of the core ontology (QURESHI; JURETA; PERINI, 2011). This extension adds the modeling dimensions, **Change**, **Mechanism** and **Effect** (**Goal** already was part of) and context elements, as depicted in Figure 25.

The CA5W1HOnTo (JEONG-DONG; SON; BAIK, 2012) uses the 5W1H approach to propose an ontology for context. We drive for greater attention to build the context con-

Figure 25 – First version of Onto4SASGORE.

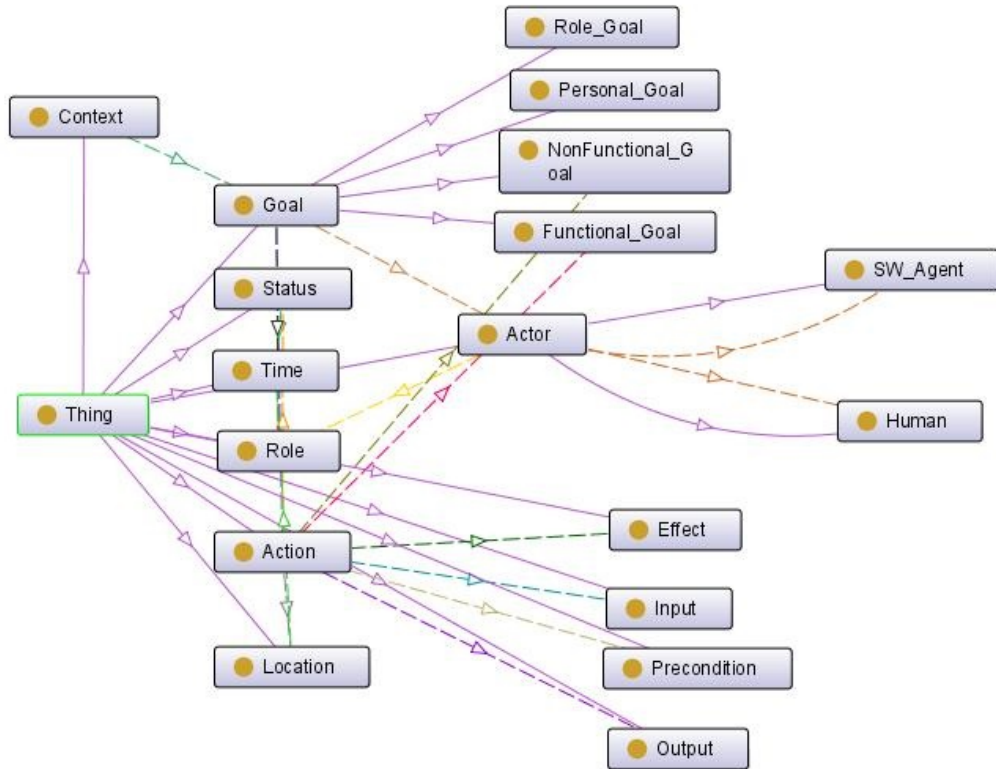


Source: Soares et al. (2016).

cept, so our ontology adopts the following concepts of the CA5W1HOnto model: **Action**, **Status**, **Goal**, **Location**, **Role** and **Time**. Figure 26 shows the context elements modeled on Protégé.

Behind analyzing these four works (core ontology for SAS, CA5W1HOnto, modeling dimensions and MAPE-K), with their concepts and relationships as well as by analyze together the modeling languages, the Onto4SASGORE was defined. Several concepts are shared for more than one work. With the results of the knowledge acquisition step, the concepts and relationships were defined and structured in the next section. Onto4SASGORE was constructed from scratch by reusing knowledge from different sources.

Figure 26 – Context Elements.



Source: Own.

4.3 Conceptualization

Conceptualization is a METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997) phase with the same name and it is in the Ontology Capture and Formalization phase in SABiO (FALBO, 2014). In this phase, we capture the concepts, relations and the properties of the ontology based on the competence questions. SABiO suggests that concepts and relations in a reference domain ontology are analyzed in the light of a foundation ontology. Then, we also chose the foundational ontologies to use in this phase. In the reuse support process in SABiO, it explains that the reuse can be done by specialization of core ontologies, or by analogy using the foundation concepts and relations for deriving the structure of a portion of the ontology, and the reuse by ontological analysis. SABiO also cited the use of OntoUML as modeling language as example to performing the ontological analysis. So, lastly, in this phase, we perform the reuse by specialization, analogy and ontological analysis. In the end of this phase, we will have the concepts, relations, properties and an OntoUML model.

In the conceptualization phase, we describe the concepts and relationships of the proposed ontology. Every concept identified on knowledge acquisition is part of Onto4SASGORE. First, we present the concepts structured in a glossary with their respective description. Second, we define the relationship between the concepts. Lastly, we present

the properties of the Onto4SASGORE. We choose the CORE (Core Ontology for Requirements Engineering) (JURETA; MYLOPOULOS; FAULKNER, 2008), DOLCE (MA-SOLO et al., 2003) and the UFO (GUIZZARDI, 2007) as foundational ontologies and we use the OntoUML to create a graphical model.

4.3.1 Concepts

Action - How the adaptable artifacts can be changed and which adaptation can be appropriate to be applied in a given condition (SALEHIE; TAHVILDARI, 2009). It is performed by actor.

Actor - Who intervene on the system. Who performs the adaptation. **Human** and **SW__Agent** are actors.

Agent - it is a existentially independent concrete individual (real entities possessing an unique identity) that does not have temporal parts, and persists in time while keeping its identity (GUIZZARDI et al., 2014).

Analyze - It compares event data against patterns in the knowledge base to diagnose symptoms and stores the symptoms for future reference (KEPHART; CHESS, 2003). It observes and analyzes situations to determine if some change needs to be made (COMPUTING et al., 2006).

Behavior - System procedure in relation to the context. Range of the actions of the system.

Communicated Information - It is every information to be communicated in the system. Domain Assumption, Evaluation and Goal are kinds of Communicated Information.

Domain Assumption - It is an indicative property, describing the environment as it is and in spite of the system-to-be (JURETA; MYLOPOULOS; FAULKNER, 2008). It is a particular information taken as truth about the domain.

Evaluation - Stakeholders' preferences over requirements (JURETA; MYLOPOULOS; FAULKNER, 2008). It is about the system to assess a possible alternative.

Task - It is an action/function that should be executed (JURETA et al., 2015).

Goal - It is a propositional variable that describes a desired state (JURETA et al., 2015). It is an objective of the system. Functional Goal, Non-functional Goal, Hardgoal and Softgoal are kinds of Goal. A goal has (ANDERSSON et al., 2009) evolution, flexibility, duration, multiplicity and dependency .

Functional Goal - It is a goal that does not refer to a quality and it refers to a *perdurant*. *Perdurant* is a concrete individual that is composed of temporal parts (JURETA; MYLOPOULOS; FAULKNER, 2008). Functional Goal refers to function (GUIZZARDI et al., 2014).

Non-Functional Goal - It is a goal that refers to a quality (JURETA; MYLOPOULOS; FAULKNER, 2008). Non-Functional Goal refers to quality. Quality constraint is a

Non-Functional Goal (GUIZZARDI et al., 2014).

Quality Constraint - It is a goal with an acknowledged shared quality space among the stakeholders (JURETA; MYLOPOULOS; FAULKNER, 2008). It exists if it is desired that a quantitative variable gets the value (in the range of values that) the requirement assigns to it (JURETA et al., 2015). Quality constraints are Non-Functional goal that specifies crisp quality region (GUIZZARDI et al., 2014).

Hardgoal - It is a goal that is satisfied by a given set of situations (GUIZZARDI et al., 2014).

Softgoal - It is a goal vague for agreed success and it does not have a shared quality space among stakeholders (JURETA; MYLOPOULOS; FAULKNER, 2008). It refers to a desirable range of values for a variable in which the range is only vaguely specified (JURETA et al., 2015). We are not able to determine a priori the set of situations that satisfies a softgoal (GUIZZARDI et al., 2014).

Evolution - The change of the system's goal within the lifetime of the system. Goal evolution can be **static** (changes are not expected) or **dynamic** (the goal can change at runtime) (ANDERSSON et al., 2009).

Flexibility - The goal is flexible in the way it is expressed. The goal statement provides flexibility for dealing with uncertainty. Goal flexibility can be **rigid** (not flexible), or **constrained**, or **unconstrained** (ANDERSSON et al., 2009).

Duration - The validity of a goal throughout the system's lifetime. It can be **temporary** (short, medium or long term) or **persistent** (ANDERSSON et al., 2009).

Multiplicity - Number of goals associated with the self-adaptability aspects of a system. It can be simple or multiple (ANDERSSON et al., 2009).

Dependency - Relation between goals. It can be **independent** or **dependent** (**complementary** or **conflicting**) (ANDERSSON et al., 2009).

Context - It is a set of information that is presupposed by the stakeholders to hold when they communicate particular requirements (QURESHI; JURETA; PERINI, 2011). It is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves (JEONG-DONG; SON; BAIK, 2012). Every requirement depends on one or more context (QURESHI; JURETA; PERINI, 2011).

Change - It is the cause for adaptation (ANDERSSON et al., 2009). A change has a source, type, frequency and anticipation.

Source - The source of change. It can be **external** or **internal** (ANDERSSON et al., 2009).

Type - The nature of change. It can be **functional**, **non-functional**, or **technological** (ANDERSSON et al., 2009).

Frequency - The frequency with which a change occurs. It can be **rare** or **frequent** (ANDERSSON et al., 2009).

Anticipation - The prediction of a change. It can be **foreseen**, **foreseeable** or **unforeseen** (ANDERSSON et al., 2009).

Effect - It is the impact of adaptation upon the system (ANDERSSON et al., 2009). A effect has a criticality, predictability, overhead and resilience.

Criticality - impact upon the system in case the self-adaptation fails. It can be **harmless**, **mission-critical** or **safety-critical** (ANDERSSON et al., 2009).

Predictability - Prediction of the consequences of adaptation. It can be **non-deterministic** or **deterministic** (ANDERSSON et al., 2009).

Overhead - The impact of system adaptation upon the quality of services of the system. It can be **insignificant** or **failure** (ANDERSSON et al., 2009).

Resilience - The persistence of service delivery that can justifiably be trusted, when facing changes. It can be **resilient** or **vulnerable** (ANDERSSON et al., 2009).

Effector - It enables state changes for a resource (COMPUTING et al., 2006). It executes the change.

Environment - The system as a whole and the local where the system is inserted.

Execute - It schedules and performs the necessary changes on change plan (COMPUTING et al., 2006).

Input - It is a resource used for an action to produce something.

Location - Where the need for change is. To locate the problem that needs to be resolved by adaptation (SALEHIE; TAHVILDARI, 2009).

Mechanism - It is the reaction of the systems towards change (ANDERSSON et al., 2009). Interface Mechanism is a kind of Mechanism. A mechanism has a type, autonomy, organization, scope, duration, timeliness and triggering.

Type - Relation of adaptation. It can be **parametric** or **structural** (ANDERSSON et al., 2009).

Autonomy - The degree of outside intervention during adaptation. It can be **autonomous** or **assisted** (ANDERSSON et al., 2009).

Organization - How the adaptation is done. It can be **centralized** or **decentralized** (ANDERSSON et al., 2009).

Scope - Where the adaptation is localized. It can be **local** or **global** (ANDERSSON et al., 2009).

Duration - How long the adaptation lasts. It can be short, medium or long term (ANDERSSON et al., 2009).

Timeliness - Guarantee of time period for performing self-adaptation. It can be **best-effort** or **guaranteed** (ANDERSSON et al., 2009).

Triggering - Association of the change that triggers adaptation. It can be **event-trigger** or **time-trigger** (ANDERSSON et al., 2009).

Monitor - It senses the system and its context, filters the accumulated sensor data, and stores relevant events (KEPHART; CHESS, 2003). It collects the details from resources via touchpoints, and correlates them into symptoms. The monitor determines a symptom to be analyzed (COMPUTING et al., 2006).

Output - The resource that some action have produced.

Precondition - Condition that must be met before the action is carried out.

Resource - Something that is available for use or that can be used for support. Change request, plan and symptom are resources.

Change Request - It describes the necessary or desirable modifications. It is generated by Analyze (COMPUTING et al., 2006).

Plan - It is a sequence of actions that it specifies the necessary change (COMPUTING et al., 2006). It serves to interpret the symptoms and devises a plan to execute the change the system through its effectors (KEPHART; CHESS, 2003). Change Plan is a plan.

Change Plan - Set of changes (COMPUTING et al., 2006).

Symptom - Something that is happening with the system.

Role - It is a function performed by an actor.

Sensor - It collects data from the executing system and its context about its current state (KEPHART; CHESS, 2003). It exposes information about the state and state transitions of a resource (COMPUTING et al., 2006). It is the hardware equipment or a software used to perceive the environment.

Status - Particular situation of an information at a given moment.

Time - When the change need to be applied (SALEHIE; TAHVILDARI, 2009).

Touchpoint - The interface to an instance of a resource, such as an operating system or a server. A touchpoint implements sensor and effector behavior for the managed resource, and maps the sensor and effector interfaces to existing interfaces (COMPUTING et al., 2006).

Appendix G presents a table with the concepts and references that use the concepts.

4.3.2 Relationships

The relationships between the concepts of the ontology are described as follows. A context has a status (hasStatus), a role (hasRole), a time (hasTime), a location (hasLocation) and a goal (hasGoal). A location and status has a (hasTime) time. A goal has a duration (hasDuration), an actor (hasActor), a dependency (hasDependency), an evolution (hasEvolution), a flexibility (hasFlexibility) and a role (hasRole).

We represent a functional goal, a non-functional goal, personal goal, a quality constraint, a role goal and a softgoal by (subclass of) goal, i.e. they are subclasses of goal.

An execute performs (performsChange) a Change and updates (updates) the monitor. A change has an anticipation (hasAnticipation), and a source (hasSource), a type

(hasChangeType), and a frequency (hasFrequency). The change happens by a reacting (hasMechanism) mechanism and produces an effect (hasEffect). The mechanism has a type (hasType), an intervention (hasIntervention), an organization (hasOrganization), a scope (hasScope), a duration (hasDuration); it has a control if the time period for performing self-adaptation can be guaranteed (hasTimeliness), and it has a knowledge of whom triggers (hasTriggering) the adaptation.

The effect has a criticality (hasCriticality), a predictability (hasPredictability), an overhead (hasOverhead), and a resilience (hasResilience). The sensor monitors (monitors) the system behavior and environment. Monitor correlates (correlates) a symptom.

Action achieves (achieves) a functional goal and it contributes to (contributes) a non-functional goal. An action has an input (hasInput), an output (hasOutput), and a precondition (hasPrecondition). An action also changes the status (changeStatus) of a resource and it has tasks (hasTask). Execute performs an action (performsAction) and a change (performsChange). It is executed by (executesThrough) an effector.

An actor plays (plays) a role. The software (SW_Agent) has a Sensor (hasSensor). The sensor has a status (hasSensorStatus), it is interfaced by (interfacedBy) touchpoint, it monitors (monitors) the behaviour and the environment (monitorEnvironment) and sends data to (sendsDataTo) monitor. A touchpoint manage resource (managesResource). A touchpoint also manages a mechanism (managesMechanism). A failure can generates (generatesError) a error.

Communicated information depends on (dependsOn) context and it refers to (refersTo) resource. A context has a goal (hasGoal), an action (hasAction), a role (hasRole), a time (hasTime), a status (hasStatus), a actor (hasActor) and a location (hasLocation).

The analyze generates a change request (generatesCR). The analyze also analyzes (analyzes) the symptom and it sends the change request to (sendCRTTo) a plan. A plan generates a change plan (generatesCP). A plan and a goal have an action (hasAction). A plan achieves a goal (achievesGoal) and it sends the change plan to (sendCPTTo) execute.

The system analyzes (analyzes) the behavior and the environment, it has a plan (hasPlan), and the effector executes (executes) this plan. The system includes (includes) system requirements, and the system requirements include (include) communicated information.

4.3.3 Attributes

Ontologies also present attributes. They are the properties of ontologies' concepts. Table 15 presents the Onto4SASGORE attributes.

Table 15 – Attributes of Onto4SASGORE

Concept	Attribute	Value Type
Plan	Policy Information	Literal
Sensor	input	Literal
Sensor	output	Literal
Sensor	name	Literal
Sensor	description	Literal
Interface Mechanism	commands	Literal
Interface Mechanism	configurationFiles	Literal
Interface Mechanism	events	Literal
Interface Mechanism	logs	Literal
Interface Mechanism	APIs	Literal
Touchpoint	type	boolean
Time	repetition_Time	integer
Time	start_Time	dateTime
Time	end_Time	dateTime
Resource	resourceType	Literal
Resource	topologies	Literal
Resource	users	Literal
Resource	hosts	Literal
Resource	status	Literal
Resource	metrics	Literal
Resource	identification	Literal
Resource	configuration	Literal
Resource	throughput	Literal

Table 15 – Attributes of Onto4SASGORE

Concept	Attribute	Value Type
Resource	capacity	Literal
SW_Agent	hasMultiplicity	Boolean

Source: Own.

4.3.4 OntoUML

We used Menthor¹ to model the new core Onto4SASGORE presented in Figure 27, and the MAPE, Context, Goal and Modeling Dimensions present in Figures from 28 to 30. Menthor allows modeling ontologies by a conceptual (ontology) modeling language, the OntoUML.

The core ontology presented by Soares et al. (2016) was updated to include the elements from the MAPE cycle. Figure 27 presents the new core Onto4SASGORE. It is formed by the core concepts of the goal-oriented requirements, context, MAPE and modeling dimensions.

According the Figure 27, a **Functional Goal**, **Non Functional Goal**, **Softgoal**, **Hardgoal** are sub kinds of **Goal** and a **Quality Constraint** is a sub kind of a **Non Functional Goal**. A **Goal**, in turn, is a **Communicated Information** as well as a **Domain Assumption** and a **Evaluation**. All **Communicated Information** refers to a **Resource** and depends on **Context**. A **Context** has a **Goal**, a **Status**, a **Role**, a **Location**, a **Time** and an **Action**. **Execute**, **Planner**, **Analyze** and **Monitor** are **Action**. A **Sensor** sends data to **Monitor**, the **Monitor** act before **Analyze**, **Planner** and **Execute**. The **Analyze** act before **Planner** and **Execute** and after **Monitor**. The **Planner** act before **Execute** and after **Monitor** and **Analyze**. The **Execute** act after **Monitor**, **Analyze** and **Planner**. The **Execute** executes through an **Effector** and it also updates the **Monitor** and performs **Change**. A **Change** can be a kind of **Goal** and all **Change** has an **Effect** and a **Mechanism**.

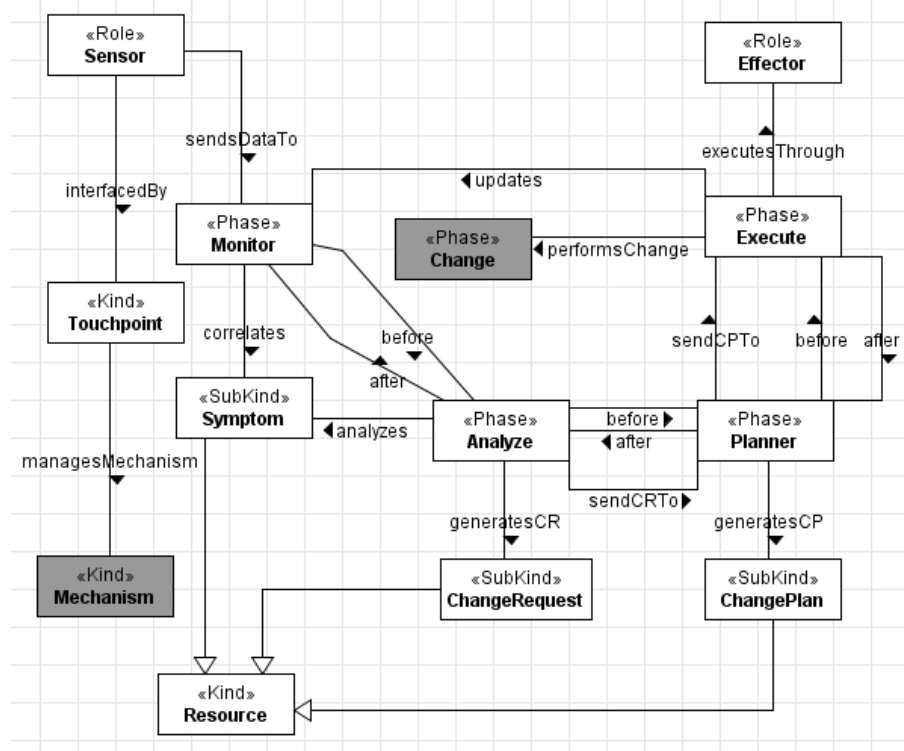
We also modeled the four parts of the core in separated and put the other concepts related to them. The modelling of each part is presented following.

We model the **Goal**, **Evaluation** and **Domain Assumption** as a **Communicated Information**. **Functional Goal**, **Non Functional Goal**, **Hardgoal** and **Softgoal** are subkinds of **Goal**. **Quality constraint** is a subkind of **Non Functional Goal**. A **Communicated Information** depends on **Context** and refers to **Resource**. The element of intersection with other parts of the ontology is the **Goal**. The concepts related to **Goal** are represented in Figure 28.

¹ <http://www.menthor.net/>

Change Plan are subkinds of **Resource**. The **Sensor** is interfaced by the **Touchpoint** and manages the **Mechanism**. The **Change** and **Mechanism** concepts are interception points with others part of the ontology.

Figure 29 – OntoUML model of the Onto4SASGORE MAPE related elements.

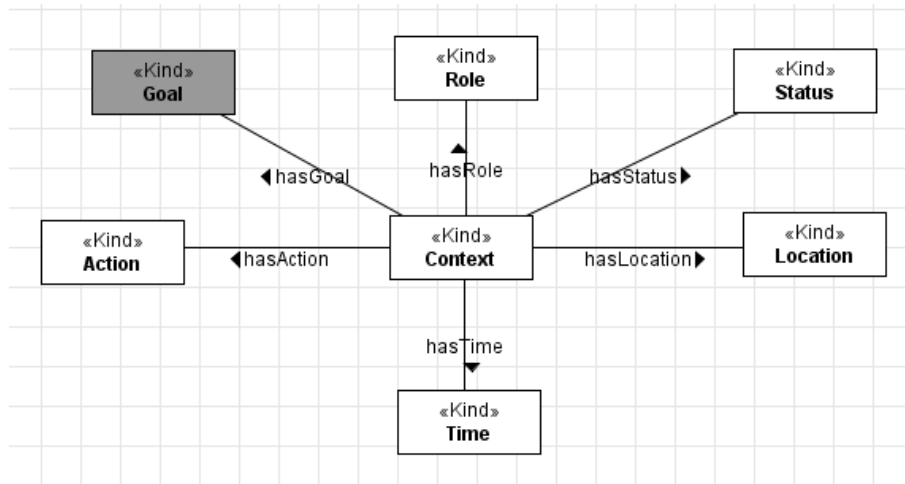


Source: Own.

A **Context** can have an **Action**, a **Role**, a **Status**, a **Location**, a **Time** and a **Goal**. Figure 30 shows the concepts related to **Context**. The **Goal** is a concept that link this part to other part of the ontology.

A **Goal** has a **Evolution**, a **Duration**, a **Flexibility**, a **Multiplicity** and a **Dependency**. **Dependent** and **Independent** as subkinds of **Dependency**, **Conflicting** and **Complementary** are subkinds of **Dependent**. **Constrained**, **Rigid** and **Unconstrained** are subkinds of **Flexibility**. **Temporary** and **Persistent** are subkinds of **Duration**. **Static** and **Dynamic** are subkinds of **Evolution**. **Change** is a phase of **Goal** and it has an **Effect** and a **Mechanism** and it can have an **Anticipation**, a **Frequency**, a type (**ChangeType**), a **Source**. **Foreseen**, **Foreseable** and **Unforeseen** are subkinds of **Anticipation**. **Frequent** and **Rare** are subkinds of **Frequency**. **Non Functional** and **Functional** are subkinds of type (**ChangeType**). **Internal** and **External** are subkinds of **Source**. An **Effect** can have an **Overhead**, a **Resilience**, a **Predictability** and a **Criticality**. **Failure** and **Insignificant** are subkinds of **Overhead**. **Resilient** and **Vulnerable** are subkinds of **Resilience**. **Deterministic** and **Non Deterministic** are subkinds of **Predictability**. **Harmless**, **Mission-Critical** and **Safety-Critical** are subkinds of **Criticality**. A **Mechanism** can have an **Autonomy**, a **Timeliness**, a **Triggering**, a duration (**MechanismDu-**

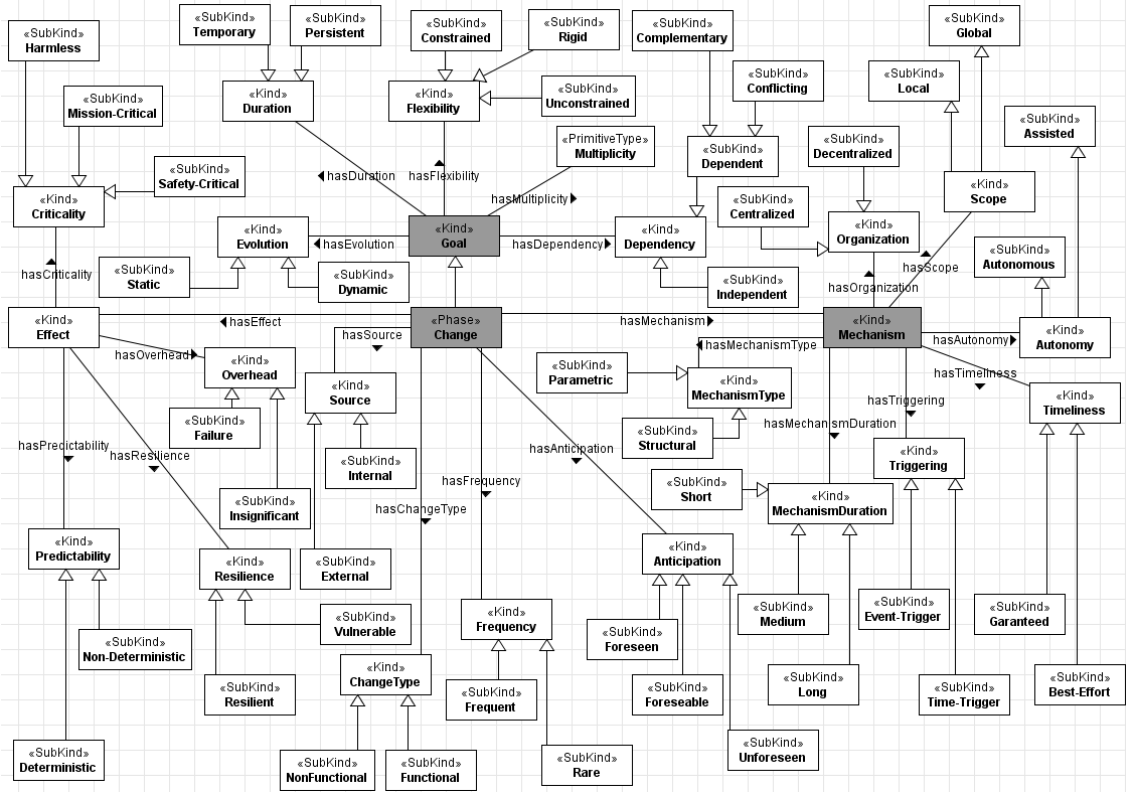
Figure 30 – OntoUML model of the Onto4SASGORE Context related elements.



Source: Own.

ration), a type (**MechanismType**), a **Organization** and a **Scope**. **Autonomous** and **Assisted** are subkinds of **Autonomy**. **Garanteed** and **Best-Effort** are subkinds of **Timeliness**. **Event-Trigger** and **Time-Trigger** are subkinds of **Triggering**. **Short**, **Medium** and **Long** are subkinds of duration (**MechanismDuration**). **Parametric** and **Strutural** are subkinds of type (**MechanismType**). **Centralized** and **Decentralized** are subkinds of **Organization**. **Local** and **Global** are subkinds of **Scope**. Figure 31 shows an OntoUML model for modeling dimensions.

Figure 31 – OntoUML model of the Onto4SASGORE Modeling Dimensions related elements.



Source: Own.

4.4 Formalization

This phase is present in Uschold and Gruninger (1996) and in SABiO (FALBO, 2014). In Uschold and Gruninger (1996), it is presented as Formal Terminology, Formal Competence Questions and Formal Axioms. In SABiO, it is presented as Ontology Capture and Formalization (started in previously phase). Therefore, in this phase, we will create the axioms and formalize the competence questions and the axioms. In the end of this phase, we will have the formal competence questions, the formal axioms and the reference ontology. To formalize the competence questions, we use SQWRL and to formalize the axioms, we use Description Logic.

4.4.1 Competence questions

We present the identification of some competence question and the question in natural language and in SQWRL. All competence questions is present in Appendix C.

ID CQ01

CQ in Natural Language: What are the sensors in the scope of the system?

CQ in Formal Language: `Sensor(?sen)->sqwrl:select(?sen)`

ID CQ80

CQ in Natural Language: How many mechanisms are there in Change X?

CQ in Formal Language: `hasMechanism(ChangeX, ?mec)->sqwrl:select(ChangeX) ^ sqwrl: count (?mec)`

ID CQ92

CQ in Natural Language: How does the system achieve the goals?

CQ in Formal Language: `Goal(?goa) ^ hasAction(?goa,?aci) ^ makeSet (?s1, ?goa) ^ makeSet (?s1,?aci) ^ sqwrl:union(?s3, ?s1, ?s2) ^ sqwrl:groupBy(?s3,?goa)->sqwrl:select (?s3)`

4.5 Implementation

This phase is present in METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997) and SABiO (FALBO, 2014) with the same name. We also include in this phase, the design process by SABiO. In this phase, we can implement the ontology in an operational language, such as OWL, and implement the competence questions and the axioms with a formal language, such as SQWRL or Description Logic. As a result of this phase, we have the operational ontology.

We have chosen Protegé² version 5.2.0 to develop Onto4SASGORE because Protegé is a free ontology editor and it allows creating, editing and viewing ontologies.

4.6 Evaluation

This phase is present in METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997) as Validation and SABiO (FALBO, 2014) as Testing process and Evaluation Support process.

Given that our goal was to develop an ontology for SAS in order to aid the elicitation and specification activities requirements and make it (re)usable by the requirements engineering community, we will evaluate the ontology by completeness, verification, validation, usability, usefulness, ease of use and correctness. To completeness, we can illustrate the use of the ontology and compare it with related works. For the verification, we can create a table with the competence questions and ontology concepts, relations and axioms as answers. The validation can be achieved by domain experts and instantiation of a real world system with the ontology elements. For the usefulness, ease of use and accordance criteria, we can use a case study and a survey to achieve these criteria. The result of this phase is the documentation of the evaluations performed.

² <http://protege.stanford.edu/>

We present the Onto4SASGORE evaluation in Section 6.

4.7 Conclusion

This Chapter presented an ontology to aid the elicitation and specification goal-oriented requirements activities, the Onto4SASGORE, with its concepts, relationships and axioms. Onto4SASGORE was constructed from scratch by reusing knowledge from different sources. We have used the approach by Uschold and Gruninger (1996), the METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997) and the SABiO (FALBO, 2014) to create the ontology. This Chapter presented what are and how to use the steps from theses methodologies to create the Onto4SASGORE. In the end of the process, we have objective, scope, motivating scenarios, the informal competence questions, the material used as knowledge source, concepts, relations, properties, axioms, OntoUML model, formal competence questions, formal axioms, a reference ontology (described by using OntoUML), an operational ontology (Protegé), the evaluation documentation and a wiki. SABiO, by the way, suggests using a wiki for documentation.

The Onto4SASGORE has several elements and it can impact its use. Therefore, we organize the steps to use the Onto4SASGORE in a process of use. This process is described in the next Chapter.

5 THE ONTO4SASGORE PROCESS

In this Chapter, we explain how to use the ontology proposed in this thesis. Therefore, we propose the Onto4SASGORE process which instance the requirements elicitation and analysis process by Sommerville (2010). The created instance defines the input and output artifacts from each activity and how to proceed in the activities, using Onto4SASGORE ontology. This process aims to aid the requirements engineer by using our ontology in the GORE requirements engineering process for self-adaptive systems. It allows a requirements engineer to elicit and specify GORE requirements for SAS. In Section 5.1, we present a overview of the process. The Onto4SASGORE process has a Goal-oriented Requirements Discovery Sub-process with 5 activities, presented in Section 5.2 and three more activities, namely, Goal-oriented Requirements Classification and Organization presented in Section 5.3, Goal-oriented Requirements Prioritization and Negotiation presented in Section 5.4 and Goal-oriented Requirements Specification presented in Section 5.5. In Section 5.6, we present a conclusion of this Section.

5.1 Overview of the Onto4SASGORE Process

We created the instance of the Sommerville (2010) process defining the input and output artifacts from each activity and how to proceed in the activities, using Onto4SASGORE ontology. The Onto4SASGORE process aims to aid the requirements engineer of using the Onto4SASGORE ontology in the goal-oriented requirements engineering process for self-adaptive systems. It allows a requirements engineer to elicit and specify GORE requirements for self-adaptive systems. The process has a Goal-oriented requirements discovery sub-process (with 5 activities) and three more activities, namely, Goal-oriented Requirements Classification and Organization, Goal-oriented Requirements Prioritization and Negotiation and Goal-oriented Requirements Specification activities.

The inputs of the Onto4SASGORE process are the Onto4SASGORE ontology presented in previously Section (Section 4) and a some SAS description.

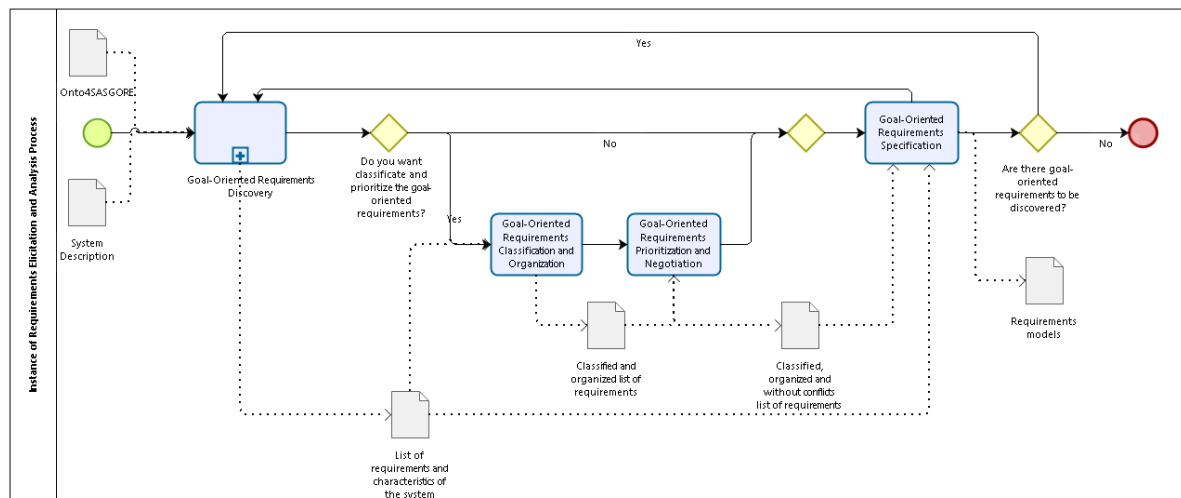
In the requirements discovery activity by Sommerville (2010) occurs the process of gathering information about the required and existing systems and distilling the user and system requirements from this information. There is interaction with system stakeholders. Systems normally have a range of stakeholders. In our instance of this process, the stakeholders are partially substituted by Onto4SASGORE ontology because it is a knowledge base for SAS.

The goal-oriented requirements discovery sub-process has the objective of eliciting the goal-oriented requirements for self-adaptive systems. To achieve this, the sub-process offers a guidance through its activities by using the Onto4SASGORE ontology and the

systems description. The goal-oriented requirements classification and organization activity aims to classify the elements found at goal-oriented requirements discovery sub-process to later, organize them. The goal-oriented requirements prioritization and negotiation allows the requirements engineer to decide how the found elements should be presented on the specification. Lastly, the goal-oriented requirements specification presents the goal-oriented requirements found. Appendix D presents a manual to use the Onto4SASGORE process without the aid of Protégé as tool and Appendix E presents a manual with the aid of Protégé.

Figure 32 shows the instance of the process, we instantiated the requirements discovery activity as sub-process, with five activities. Goal-Oriented Requirements classification and organization, goal-oriented requirements prioritization and negotiation, and goal-oriented requirements specification activities were instantiated as activities in our process. The process instance input is the system description and the concepts of the Onto4SASGORE. The process instance output is a document of the self-adaptive system requirements in text or in models. Following, we describe the goal-oriented requirements discovery sub-process and its five activities.

Figure 32 – The Onto4SASGORE process.



Source: Own.

5.2 Goal-oriented Requirements Discovery Sub-process

The sub-process requirements discovery has five sequence activities to discover the goal-oriented requirements of a self-adaptive systems. We introduce the sub-process as a guidance for elicitation for SAS requirements. There is no goal-oriented requirements elicitation guide for SAS on literature (to the best of our knowledge). The specification approaches proposed to SAS do not explicit how to elicit the goal-oriented requirements,

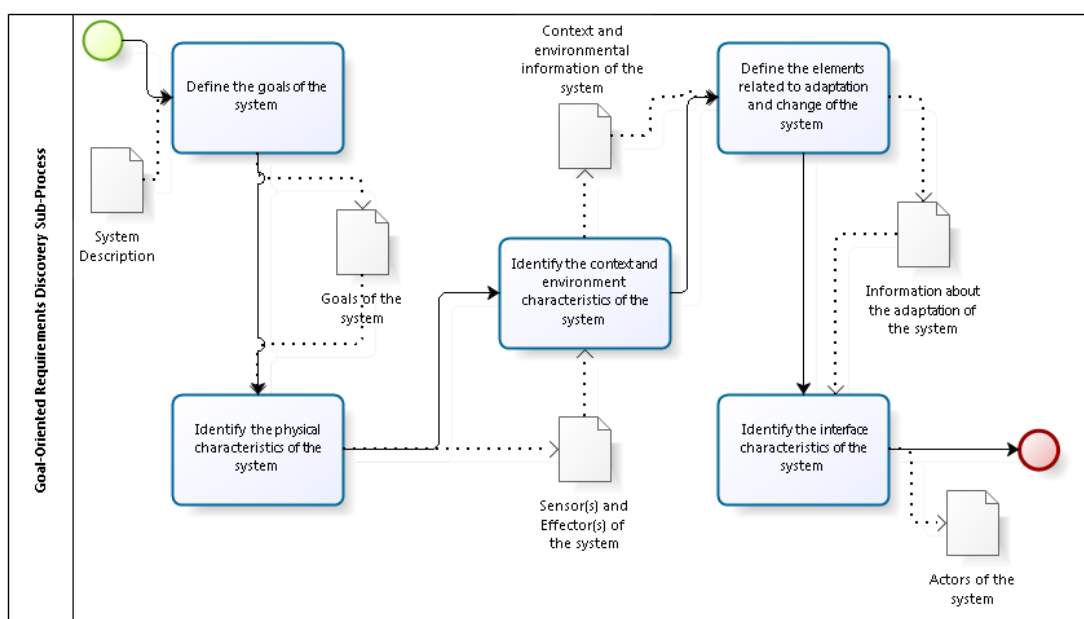
they just allow the specification of SAS on a predefined group of characteristics, without making it clear how the characteristics were found.

It is not our purpose to discover goal-oriented requirements for SAS in an automatic way. We intend to guide the requirements engineer on the goal-oriented requirements discovery activity on this kind of system, facilitating the activities on requirements elicitation for SAS phase, rendering them on practical and organized activities. For this, we propose a guide that can be followed through some activities. The sub-process presents five activities to be followed and performed. These activities present guidelines to requirements engineer. Each sub-process activity presents:

- a name;
- an objective;
- a description;
- expected results;
- a set of questions (in order to facilitate the performing of the activity);
- concepts and competence questions of Onto4SASGORE ontology (that the activity was based on).

Figure 33 presents the activities on the goal-oriented requirement discovery sub-process.

Figure 33 – Requirements Discovery Sub-process.



Source: Own.

The sub-process presents five activities and it has the objective to guide the requirements engineer in the goal-oriented requirements elicitation for SAS. The objective of the sub-process is to elicit and list the requirements of the system. The requirements engineer uses one or more elicitation techniques previously chosen and collect the requirements together to the stakeholders of the system. In our case, the chosen technique is questionnaire, because the competence questions facilitate the creation of questionnaire. The expected results are the instantiating of the concepts of the Onto4SASGORE ontology. The glossary that it is built on this sub-process can also be replaced by the Onto4SASGORE ontology. To replace the system stakeholders, we use as input artifact the Onto4SASGORE ontology as knowledge base and the system description. Each activity on sub-process has a set of questions to be answered by the requirements engineer using the knowledge offered by Onto4SASGORE. All the questions compose a guidance and they were founded on the concepts and competence questions from the Onto4SASGORE. Following, we present each activity with their descriptions and guidelines.

5.2.1 Activity 1

Name: Define the goals of the system.

Objective: With this activity, we aim to find the goals of the system.

Description: The engineer should answer a set of questions to the system description based on some concepts of Onto4SASGORE and some competence questions by using the knowledge offered by information present in Table 16, in order to discovery all goals of the system.

Expected results: In the end of this activity, the requirements engineer will obtain the goals of the system.

Questions:

1. What is the main *goal* of the system?
2. What are the secondary *goals* of the system?
3. What are the characteristics of each *goal*?
4. What are the *functional goals* of the system?
5. What are the *non-functional goals* of the system?
6. What are the *constraints* of the system?

Table 16 presents the concepts and competence questions related to this activity.

Table 16 – Concepts and CQ related to activity 1.

Concept
Goal, Functional Goal, Hardgoal, Role Goal, Duration, Flexibility, Dependency, Evolution, Non-Functional Goal, Softgoal and Quality Constraint.
Competence Questions
CQ04, CQ08, CQ09, CQ14, CQ15, CQ19, CQ20, CQ21, CQ22, CQ23, CQ24, CQ25, CQ26, CQ27, CQ28, CQ29, CQ30, CQ31

Source: Own.

5.2.2 Activity 2

Name: Identify the physical characteristics of the system.

Objective: In this activity, we intend to find the sensors and effectors of the system.

Description: Some systems may have a software or service actuating as a sensor or effector, not necessarily a physical component. Therefore, this activity involves mechanism or resource concepts from the ontology. The requirements engineer should answer the set of questions based on the system description and according to the Onto4SASGORE concepts.

Expected results: The identification of the sensor(s) and effector(s) of the system.

Questions:

1. What is the *sensor* of the system?
2. What is the *effector* of the system?
3. What are the *resources* used by system?

Table 17 presents the concepts and competence questions related to this activity.

Table 17 – Concepts and CQ related to activity 2.

Concept
Sensor, Effector and Resource.
Competence Questions
CQ01, CQ02, CQ03, CQ37, CQ39.

Source: Own.

5.2.3 Activity 3

Name: Identify the context and environment characteristics of the system.

Objective: In this activity, we can find the context to be considered by the system and the characteristics from the environment where the system is operating.

Description: With the Onto4SASGORE ontology and the description of the system, the requirements engineer will answer the set of questions of this activity.

Expected results: In the end of this activity, the requirements engineer will have the context and environmental information of the system.

Questions:

1. What are the contexts considered by the system?
2. What are the users of the system?
3. What is the environment where the system is operating?

Table 18 presents the concepts and competence questions related to this activity.

Table 18 – Concepts and CQ related to activity 3.

Concept
Context, Goal, Location, Time, Action, Role, Status, Actor and Environment
Competence Questions
CQ54, CQ55, CQ56, CQ57, CQ58, CQ59, CQ60.

Source: Own.

5.2.4 Activity 4

Name: Define the goal-oriented requirements related to adaptation and change of the system.

Objective: In this activity, we intend to find all goal-oriented requirements related to adaptation and change of the system.

Description: To accomplish the objective, the requirements engineer will answer the questions of this activity based on the system description and the Onto4SAS ontology. This activity also involves goal, mechanism and effect concepts.

Expected results: After performing this activity, the requirements engineer will have the information about the adaptation of the system.

Questions:

1. What are the actions of the system?
2. What are the effects by action?
3. What are the changes of the system?
4. What is the cause of the change?
5. What is the mechanism used to perform a change?

Table 19 presents the concepts and competence questions related to this activity.

Table 19 – Concepts and CQ related to activity 4.

Concept
Goal, Effects, Action, Predictability, Resilience, Criticality, Overhead, Change Plan, Symptom, Change Request, Goal, Duration, Flexibility, Dependency, Evolution, Change, Frequency, Change Type, Anticipation, Source, Mechanism, MechanismType, Timeliness, Triggering, Autonomy, Organization, Scope, Duration and Interface Mechanism.
Competence Questions
CQ08, CQ09, CQ10, CQ11, CQ12, CQ13, CQ14, CQ15, CQ16, CQ17, CQ18, CQ28, CQ29, CQ30, CQ31, CQ32, CQ33, CQ34, CQ35, CQ36, CQ37, CQ38, CQ39, CQ40, CQ41, CQ42, CQ43, CQ44, CQ45, CQ46, CQ47, CQ48, CQ49, CQ50, CQ51, CQ52, CQ53, CQ61, CQ62, CQ63, CQ64, CQ65, CQ66, CQ67, CQ68, CQ69, CQ70.

Source: Own.

5.2.5 Activity 5

Name: Identify the interface characteristics of the system.

Objective: In this activity, we intend to list the goal-oriented requirements related to interface of the system. The objective of this activity is identifying to whom or to what the system communicates or interacts, as well as the communication interface to realize it.

Description: The requirements engineering will identify all the actors of the system based on the system description and guided by the Onto4SASGORE. He or she will answer the questions for this activity using the Onto4SASGORE concepts and competence questions

present on Table 20.

Expected results: Find the actors of the system.

Questions:

1. Who are the actors of the system?
2. What is the input of each action of the system?
3. What is the output of each action of the system?

Table 20 presents the concepts and competence questions related to this activity.

Table 20 – Concepts and CQ related to activity 5.

Concept
Actor, Role, Input, Monitor, Analyze, Plan, Execute, Effector, Output, Mechanism and Sensor.
Competence Questions
CQ05, CQ06, CQ07, CQ32, CQ33, CQ34, CQ35, CQ36, CQ37, CQ38, CQ39, CQ40, CQ41, CQ42, CQ43.

Source: Own.

The sub-process offers a questionnaire with twenty question that it should be answered with the light of the Onto4SASGORE ontology. We list the twenty questions and highlight the main concept of each question.

1. What is the main *goal* of the system?
<Goal>, *<Functional_Goal>*, *<Hardgoal>*, *<NonFunctional_Goal>*, *<Softgoal>*,
<QualityConstraint>.
2. What are the secondary *goals* of the system?
<Goal>, *<Functional_Goal>*, *<Hardgoal>*, *<NonFunctional_Goal>*, *<Softgoal>*,
<QualityConstraint>.
3. What are the characteristics of each *goal*?
<Duration>, *<Flexibility>*, *<Dependency>*, *<Evolution>*.
4. What are the *functional goals* of the system?
<Functional_Goal>, *<Hardgoal>*, *<Softgoal>*.
5. What are the *non-functional goals* of the system?
<NonFunctional_Goal>, *<QualityConstraint>*.

6. What are the *constraints* of the system?
 <QualityConstraint>.
7. What is the *sensor* of the system?
 <Sensor>.
8. What is the *effector* of the system?
 <Effector>.
9. What are the *resources* used by system?
 <Resource>.
10. What are the *contexts* considered by the system?
 <Context>, <Goal>, <Location>, <Time>, <Action>, <Role>, <Status>, <Actor>.
11. What are the *users* of the system?
 <Actor>.
12. What is the *environment* where the system is operating?
 <Environment>.
13. What are the *actions* of the system?
 <Action>, <Input>, <Output>.
14. What are the *effects* by action?
 <Effects>, <Predictability>, <Resilience>, <Criticality>, <Overhead>.
15. What are the *changes* of the system?
 <Goal>, <Effects>, <Action>, <Input>, <Output>, <Predictability>, <Resilience>, <Criticality>, <Overhead>, <ChangePlan>, <Symptom>, <Change Request>, <Goal>, <Duration>, <Flexibility>, <Dependency>, <Evolution>, <Change>, <Frequency>, <ChangeType>, <Anticipation>, <Source>, <Mechanism>, <MechanismType>, <Timeliness>, <Triggering>, <Autonomy>, <Organization>, <Scope>, <Duration>, <Interface Mechanism>.
16. What is the cause of the *change*?
 <Change>, <Source>.
17. What is the *mechanism* used to perform a *change*?
 <Mechanism>, <MechanismType>, <Timeliness>, <Triggering>, <Autonomy>, <Organization>, <Scope>, <Duration>, <Interface Mechanism>.
18. Who are the *actors* of the system?
 <Actor>, <Role>, <Input>, <Monitor>, <Analyze>, <Plan>, <Execute>, <Effector>, <Output>, <Mechanism>, <Sensor>.

19. What is the *input* of each *action* of the system?

<Action>, *<Input>*.

20. What is the *output* of each *action* of the system?

<Action>, *<Output>*.

After the conclusion of the requirements discovery sub-process, the output artifact is a list of system elements instantiated from Onto4SASGORE concepts portraying the characteristics of the system. This list of elements is the input of the requirements classification and organization activity.

5.3 Goal-oriented requirements classification and organization

This activity aims to group the related requirements and it organizes them into coherent clusters. The input artifact is a list of elements and characteristics of the system created on the requirements discovery sub-process. In this activity, the requirement engineer will classify and organize the list of elements according the following four groups: Goal, Interface and Communication, Context and Environment, and Change and Adaptation.

Every communicated information is part of Goal group. Domain assumption, evolution, task and goal are communicated information. Functional goal, non-functional goal, personal goal, role goal, quality constraint and softgoal are goals. Every goal has mandatory multiplicity and dependence.

Actor, role, sensor, effector and touchpoint are part of the Interface and Communication group. Human and SW_Agent are actors. A sensor can to have a name, input, output and description. A touchpoint may have a type.

The Context and Environment group contains the elements from environment, behavior and context. Location, status and time are context.

The Change and Adaptation group will contain the elements instantiated by action, goal, change, effect, mechanism and resource. Every action has mandatory an input and an output. Analyze, execute, monitor and plan are actions. An action can also to have a precondition. A change can have a source, type, frequency and anticipation. An effect can have a criticality, predictability, overhead and resilience. A mechanism can have a type, autonomy, organization, scope, duration, timeliness and triggering. A resource can have a type, status, topologies, users, hosts, identification, configuration, capacity, throughput and metrics. Change request, plan and symptom are resources.

A goal with dynamic evolution is part of the change and adaptation group. A goal with not rigid flexibility also is part of the of the change and adaptation group and the same happens with a goal with temporary duration.

The output artifact of this activity is a classified and organized list of elements. Following, we list the concepts by group.

Goal

- Communicated Information
 - Domain Assumption -
 - Evaluation -
 - Goal (evolution, flexibility, duration, multiplicity and dependency)
 - * Functional Goal -
 - * Non-functional Goal -
 - Quality Constraint -
 - * Hardgoal -
 - * Softgoal -

Interface and Communication

- Actor -
 - Human -
 - SW_Agent -
- Role -
- Sensor -
- Effector -
- Touchpoint -

Context and Environment

- Environment -
- Behavior -
- Context -
 - Location -
 - Status -
 - Time -

Change and Adaptation

- Action - (Precondition, input and output)
 - Analyze -
 - Execute -
 - Monitor -
 - Plan -
- Change (source, type, frequency and anticipation) -
- Effect (criticality, predictability, overhead and resilience) -
- Mechanism (type, autonomy, organization, scope, duration, timeliness and triggering) -
- Resource -
 - Change Request -
 - Plan -
 - * Change Plan -
 - Symptom -
- Goal - (evolution, flexibility, duration, multiplicity and dependency)
 - Functional Goal -
 - Non-functional Goal -
 - Personal Goal -
 - Role Goal -
 - Quality Constraint -
 - Softgoal -

5.4 Goal-oriented Requirements Prioritization and Negotiation

In the prioritization and negotiation activity, the requirements engineer will prioritize requirements and resolve requirements conflicts. The input artifact is a classified and organized list of requirements created on previous activity, requirements classification and organization. The requirements engineer is free to choose an approach to perform this activity. We can suggest a general approach like as presented in Lai, Mahmood and Liu (2011). The output artifact of this activity is a classified, organized and without conflicts list of possible requirements.

5.5 Goal-oriented requirements specification

In this activity, the goal-oriented requirements are documented and used as input to the next round of the spiral. The input artifact is a classified, organized and without conflicts list of elements, Onto4SASGORE concepts instances. In this activity, the requirements engineer will specify the goal-oriented requirements. The template is specified as following. The concepts and its relations can be replicated how many times are needed. The asterisk (*) means that the concept is mandatory.

Action – *<instance>*

- Analyze – *<instance>*
 - Input* – *<instance>*
 - Output* – *<instance>*
 - Precondition – *<instance>*
- Execute – *<instance>*
 - Input* – *<instance>*
 - Output* – *<instance>*
 - Precondition – *<instance>*
- Monitor – *<instance>*
 - Input* – *<instance>*
 - Output* – *<instance>*
 - Precondition – *<instance>*
- Plan – *<instance>*
 - Input* – *<instance>*
 - Output* – *<instance>*
 - Precondition – *<instance>*

Actor - *<instance>*

- Human - *<instance>*
- SW_Agent - *<instance>*

Communicated Information - *<instance>*

- Domain Assumption - *<instance>*

- Evaluation - *<instance>*
- Goal - *<instance>*
 - Functional Goal - *<instance>*
 - * Multiplicity* - *<instance>*
 - * Dependency* - *<instance>*
 - * Evolution - *<instance>*
 - * Flexibility - *<instance>*
 - * Duration - *<instance>*
 - Non-functional Goal - *<instance>*
 - * Multiplicity* - *<instance>*
 - * Dependency* - *<instance>*
 - * Evolution - *<instance>*
 - * Flexibility - *<instance>*
 - * Duration - *<instance>*
 - Hardgoal - *<instance>*
 - * Multiplicity* - *<instance>*
 - * Dependency* - *<instance>*
 - * Evolution - *<instance>*
 - * Flexibility - *<instance>*
 - * Duration - *<instance>*
 - Quality Constraint - *<instance>*
 - * Multiplicity* - *<instance>*
 - * Dependency* - *<instance>*
 - * Evolution - *<instance>*
 - * Flexibility - *<instance>*
 - * Duration - *<instance>*
 - Softgoal - *<instance>*
 - * Multiplicity* - *<instance>*
 - * Dependency* - *<instance>*
 - * Evolution - *<instance>*
 - * Flexibility - *<instance>*
 - * Duration - *<instance>*

Context - *<instance>*

- Location - *<instance>*

- Status - *<instance>*

- Time - *<instance>*

- Role - *<instance>*

Change - *<instance>*

- Source - *<instance>*

- Type - *<instance>*

- Frequency - *<instance>*

- Anticipation - *<instance>*

Effect - *<instance>*

- Criticality - *<instance>*

- Predictability - *<instance>*

- Overhead - *<instance>*

- Resilience - *<instance>*

Mechanism - *<instance>*

- Type - *<instance>*

- Autonomy - *<instance>*

- Organization - *<instance>*

- Scope - *<instance>*

- Duration - *<instance>*

- Timeliness - *<instance>*

- Triggering - *<instance>*

Resource - *<instance>*

- Change Request - *<instance>*

- Plan - *<instance>*

- Change Plan - *<instance>*

- Symptom - *<instance>*

Sensor* - *<instance>*

Effector* - *<instance>*

Touchpoint - *<instance>*

Environment - *<instance>*

Behavior - *<instance>*

Task - *<instance>*

5.6 Conclusion

This Chapter presented the process to use the ontology proposed in this thesis. The Onto4SASGORE process presents a sub-process to discovery goal-oriented requirements for SAS which present a guide to elicitation activity. The process also relies with three more activities, one of them is for classification and organization of the requirements found, the other is for prioritization and negotiation of the requirements and the last one is for the requirements specification, which it offers a template to instantiate the concepts of the Onto4SASGORE ontology.

6 EVALUATION

The common thing on methodologies to create ontology is that they start from the identification of the purpose of the ontology, explain the need for domain knowledge acquisition, and the need for ontology evaluation. However, evaluation is performed differently in each one. METHONTOLOGY proposes that evaluation be carried out throughout the entire lifetime of the ontology development process (GÓMEZ-PÉREZ, 2001). The work presented in (BRANK; GROBELNIK; MLADENIC, 2005) has a classification of ontology evaluation approaches. It advocates that most evaluation approaches fall into one of the following categories:

- Comparing the ontology to a “golden standard” (which may itself be an ontology);
- Using the ontology in an application and evaluating the results;
- Comparisons with a source of data (e.g. a collection of documents) about the domain to be covered by the ontology;
- Humans who trying to assess how well the ontology meets a set of predefined criteria, standards, requirements, etc.

Based on this, we intend evaluate our ontology using six criteria, *comprehensiveness*, *verification validation*, *usefulness*, *easiness of use* and *accordance*. For the *comprehensiveness* criterion, presented in Section 6.1, we intend to find out whether the Onto4SASGORE has the concepts presented in the works in literature for SAS. So, we have evaluated the Onto4SASGORE by checking whether it presents the concepts of the works selected the SLRs presented at the Section 3. In Section 6.2, we explain the *verification* and *validation* criteria, we intend to prove that our ontology can provide reliable answers to competence questions using its terminology and that it can be instantiated by some real-world systems. To evaluate the *usefulness* and *easiness of use* we have performed a case study with eight requirements engineers, in which they use the Onto4SASGORE process and answer a survey created through the Technology Acceptance Model (TAM) method. This is explained in Section 6.3. Lastly, in Section 6.4, we have the *accordance* evaluation, we have also performed other survey with specialists in SAS. Section 6.5, we present the summary of this Chapter.

6.1 Comprehensiveness

In this Section, we evaluate the comprehensiveness of Onto4SASGORE ontology to verify whether our ontology meets the knowledge present on others works about SAS domain.

We want to prove that Onto4SASGORE is more complete than works on literature about SAS domain. We have used the works found on the SLR presented in Section 3.

We reinforce the importance of the works like the ontology for SAS by Qureshi, Jureta and Perini (2011), the CA5W1HOnto (JEONG-DONG; SON; BAIK, 2012) ontology for context, the modeling dimensions for SAS and the MAPE-K (KEPHART; CHESS, 2003) adaptation cycle. However, our ontology used those works as base of knowledge, then the Onto4SASGORE already contemplate the concepts present in those works. Note the S6 study corresponds to ontology by Qureshi, Jureta and Perini (2011), because this work was selected by the SLR.

Tables from 21 to 33 present an alignment of Onto4SASGORE concepts with the concepts of others works in the literature about SAS. In the left side of those tables are the concepts of the 13 studies and in the right side of the table are the concepts of the Onto4SASGORE that correspond to each concept presented by the other works about ontologies for SAS. The tables are separated by study to offer a better visualization. Analyzing the Tables from 21 to 33, we conclude that in relation with the works about ontology and SAS in the literature, the Onto4SASGORE is complete, considering that it can represent through its concepts all concepts of those works.

We correlate the concepts of the S1 study and the Onto4SASGORE concepts in the following way. For the Goal, Resource, Software, Communication, Time, Action, Behavior, Task and State concepts we have the same (or almost the same) name. For the other concepts, we understand as follow. Situation as Action, Situations Relations as Dependency, Conditions as Quality Constraint, Events as Action, User as Actor, Capacity as Scope, Knowledge as Domain Assumption and Policy as Goal. The correlation of the concepts from S1 study and the concepts from Onto4SASGORE is presented in Table 21.

Table 22 presents the correlation of the concepts from S2 study and the concepts from Onto4SASGORE concepts. In the S2 study, the Behavior, Environment, Interface, State, Time and Agent concepts have the same (or almost same) name that some concept in Onto4SASGORE. The other concepts, we understand as follows. Systems as SW_Agent, Objective as Goal, Function as Functional Goal, Model as Resource, System to build as SW_Agent, Relation as Dependency, Interaction as Dependency, Structure as Mechanism, System element as Task or Goal, Memory as Resource, Attractor as Sensor and Complex Adaptive Systems - CAS (set of agents) as Actor.

For S3 study, the Agent concept is correlated to Onto4SASGORE by SW_Agent concept, the Space concept as Location or Environment and The Device as Mechanism. The correlation between the S3 study concepts and Onto4SASGORE concepts is presented in Table 23.

In the S4 study, the Plan, Behaviour and Context concepts has the same name that concepts of Onto4SASGORE. In the other hand, When is understand as Time, Where as Location and Uncertainty as Non-Functional Goal, as we can note in Table 24.

Table 21 – Study S1 concepts vs Onto4SASGORE concepts

S1 Concepts	Onto4SASGORE Correspondent Concepts
Policies	Goal
Goal	Goal
Situations	Action
Situation relations	Dependency
Conditions	Quality constraint
Events	Action
Actions	Action
Situations	Action
Resource	Resource
User	Actor
Software	SW_Agent
Phenomenon	Communicated Information
Communication	Communicated Information
Time	Time
Capacity	Scope
Knowledge	Domain Assumption
Knowledge Goal	Goal
Action	Action
Event	Action
Behavior	Behavior
Task	Task
Policy	Goal
State	Status
Situation	Action

Source: Own.

For S5 study, the Context, Task, Environment, Domain and Agent concepts have a correspondent on Onto4SASGORE with the same name. But Presentation, System, Service and User are not. Then, we understand Presentation as Communicated Information, System as SW_Agent, Service as Task and User as Actor. The correlation between the S5 study concepts and Onto4SASGORE concepts is presented in Table 25.

The S6 study is a core ontology for requirements engineer for SAS by Qureshi, Jurata and Perini (2011). It was used as knowledge base of Onto4SASGORE, therefore all concepts from S6 is in Onto4SASGORE and with the same name. Table 26 presents the correlation of the concepts from S6 study versus the concepts from Onto4SASGORE.

The S7 study presents Sensor, Context, Analyze concepts with a similar name that

Table 22 – Study S2 concepts vs Onto4SASGORE concepts

S2 Concepts	Onto4SASGORE Correspondent Concepts
Systems	SW_Agent
Behavior	Behavior
Objective	Goal
Function	Functional Goal
Environment	Environment
Model	Resource
System to build	SW_agent
Interface	Interface Mechanism
Relation	Dependency
Interaction	Dependency
Structure	Mechanism
System element	Goal Task
State	Status
Memory	Resource
Time	Time
Attractor	Sensor
Agent	SW_Agent
Complex Adaptive Systems CAS	Actor

Source: Own.

Table 23 – Study S3 concepts vs Onto4SASGORE concepts

S3 Concepts	Onto4SASGORE Correspondent Concepts
Agent	SW_Agent
Space	Location Environment
Device	Mechanism

Source: Own.

Onto4SASGORE, but to the others concepts, we can understand the Actuator as Effector, Simple as Functional Goal, Complex as Softgoal and Sense Symptom as presented in Table 27.

S8 study has Context, Quality and Trigger concepts with the same (or almost same) name that the corresponding on Onto4SASGORE. The ActiveObject (preference, event), Agent, Service, Rule, Person and Role do not have corresponding with a similar name, but we understand ActiveObject (preference, event) as Effect or Mechanism, Agent as

Table 24 – Study S4 concepts vs Onto4SASGORE concepts

S4 Concepts	Onto4SASGORE Correspondent Concepts
When (flexibility)	Time
Where (variability)	Location
plans - How	Plan
Uncertainty	Non-Functional Goal
Context	Context
Behaviours	Behavior

Source: Own.

Table 25 – Study S5 concepts vs Onto4SASGORE concepts

S5 Concepts	Onto4SASGORE Correspondent Concepts
Context	Context
Task	Task
Presentation	Communicated Information
Environment	Environment
System	SW_Agent
Domain	Domain Assumption
Service	Task
Agent	SW_Agent
User	Actor

Source: Own.

SW_Agent, Service as Task, Rule as Precondition, Person as Human and Role Actor. Table 28 presents the concepts from S8 study and Onto4SASGORE.

S9 study presents the Sensors, Monitors, Analyzers, Planner, Change Advisory Board and Change Manager concepts with the same (or almost the same) name of some Onto4SASGORE concepts. About the others concepts, we understand Implementation team as Actor, Approver as Analyze, Scheduler as Plan, Process Owner as Actor and Affected Users as Actor. Table 29 presents the correlation of the concepts from S9 study versus the concepts from Onto4SASGORE.

Monitor & Detect, Analyze & Characterize, Plan & Adapt are the concepts from S10 study that they have almost the same name of some concept on Onto4SASGORE. About the other concepts from S10 study, we have the correlation with Onto4SASGORE as follows. Network as Resource, Host as Resource, App Software as SW_Agent, Services as Task, Abstract Architecture as Resource, Development Time as Time, Runtime

Table 26 – Study S6 concepts vs Onto4SASGORE concepts

S6 Concepts	Onto4SASGORE Correspondent Concepts
Context	Context
Communicated information	Communicated information
Resource	Resource
Domain assumption	Domain assumption
Goal	Goal
Functional goal	Functional goal
Quality constraint	Quality constraint
Softgoal	Softgoal
Task	Task
Evaluation	Evaluation

Source: Own.

Table 27 – Study S7 concepts vs Onto4SASGORE concepts

S7 Concepts	Onto4SASGORE Correspondent Concepts
Device	Resource
Sensor	Sensor
Actuator	Effector
Feature	Goal
User	Actor
Context	Context
Feature Simple	Functional Goal
Feature Complex	Softgoal
Sense	Symptom
Analyse	Analyze

Source: Own.

as Execute, Confidentiality as Non-Functional Goal, Integrity as Non-Functional Goal, Availability as Non-Functional Goal, Local Only as Location, Centralized as Organization centralized, Decentralized as Organization decentralized, Human-Driven as Human, Heuristics-Driven as Precondition, Algorithm-Driven as Precondition, Reactive as Effect, Proactive as Action, System Boundary as SW_Agent, System Internals as SW_Agent, Proxy/Containment as Resource, Redundancy as Goal, Diversity as Goal, Recomposition as Organization, Rejuvenation as Anticipation. The correlation of concepts from S10 study and concepts from Onto4SASGORE is presented in Table 30.

Table 28 – Study S8 concepts vs Onto4SASGORE concepts

S8 Concepts	Onto4SASGORE Correspondent Concepts
Context	Context
ActiveObject (preference, event)	Effect Mechanism
Agent	SW_Agent
Service	Task
Rule	Precondition
Person	Human
Role	Actor
Quality	Quality constraint
Trigger	Triggering

Source: Own.

Table 29 – Study S9 concepts vs Onto4SASGORE concepts

S9 Concepts	Onto4SASGORE Correspondent Concepts
Sensors	Sensor
Monitors	Monitor
Implementation team	Actor
Analyzers	Analyze
Approver	Analyze
Planer	Plan
Change Advisory Board	Change
Approvers	Analyze
Scheduler	Plan
Process Owner	Actor
Change Manager	Change
Affected Users	Actor

Source: Own.

S11 study has the Effector, Sensor Failure and Sensor Noise concepts names like corresponding concept in Onto4SASGORE. The other concepts on S11 study that do not have a similar name of some concept in Onto4SASGORE, we have corresponding in the following way. We understand Missing Requirement as Goal, Ambiguous Requirement as Goal, Falsifiable Assumption as Domain Assumption, Unsatisfiable as Softgoal, Requirements as Goal, Requirements Interactions as Dependency, Doubt as Predictability, Claim as Communicated Information, Changing Requirements as Change Plan, Design as Re-

Table 30 – Study 10 concepts vs Onto4SASGORE concepts

S10 Concepts	Onto4SASGORE Correspondent Concepts
Monitor & Detect	Monitor
Analyze & Characterize	Analyze
Plan & Adapt	Plan
Network	Resource
Host	Resource
App Software	SW_Agent
Services	Task
Abstract Architecture	Resource
Development Time	Time
Runtime	Execute
Confidentiality	Non-Functional goal
Integrity	Non-Functional goal
Availability	Non-Functional goal
Local Only	Location
Centralized	Organization centralized
Decentralized	Organization decentralized
Human-Driven	Human
Heuristics-Driven	Precondition
Algorithm-Driven	Precondition
Reactive	Effect
Proactive	Action
System Boundary	SW_Agent
System Internals	SW_Agent
Proxy/Containment	Resource
Redundancy	Goal
Diversity	Goal
Recomposition	Organization
Rejuvenation	Anticipation

Source: Own.

source, Unexplored Alternatives as Anticipation, Untraceable Design as Resource, Risk as Change, Misinformed Tradeoff Analysis as Analyze, Inadequate Design as Resource, Unverified Design as Anticipation, Inadequate Implementation as Scope, Latent Behavior as Behavior, Imprecision as Softgoal, Inaccuracy as Softgoal, Unpredictable Environment as Environment, Ambiguity as Goal, Non-Specificity as Goal, Inconsistency as Goal and Incomplete Information as Communicated Information. The correlation of concepts from

S11 study and concepts from Onto4SASGORE is presented in Table 31.

Table 31 – Study 11 concepts vs Onto4SASGORE concepts

S11 Concepts	Onto4SASGORE Correspondent Concepts
Requirements	Goal
Missing Requirement	Goal
Ambiguous Requirement	Goal
Falsifiable Assumption Unsatisfiable	Domain Assumption
Requirements	Goal
Requirements Interactions	Dependency
Doubt	Predictability
Claim	Communicated Information
Changing Requirements	Change Plan
Design	Resource
Unexplored Alternatives	Anticipation
Untraceable Design	Resource
Risk	Change
Misinformed Tradeoff Analysis	Analyze
Inadequate Design	Resource
Unverified Design	Anticipation
Inadequate Implementation	Scope
Latent Behavior	Behaviour
Run-Time	Time
Effector	Effector
Sensor Failure	Sensor
Sensor Noise	Sensor
Imprecision	Softgoal
Inaccuracy	Softgoal
Unpredictable Environment	Environment
Ambiguity	Goal
Non-Specificity	Goal
Inconsistency	Goal
Incomplete Information	Communicated Information

Source: Own.

Since S12 study is a first version of Onto4SASGORE, then all concepts on S12 study are present on Onto4SASGORE and they have the same name. The correlation between the concepts from S12 and the concepts from Onto4SASGORE is in Table 32.

Table 32 – Study 12 concepts vs Onto4SASGORE concepts

S12 Concepts	Onto4SASGORE Correspondent Concepts
Context	Context
Role	Role
Action	Action
Location	Location
Status	Status
Time	Time
Communicated information	Communicated information
Resource	Resource
Domain assumption	Domain assumption
Goal	Goal
Functional goal	Functional goal
Quality constraint	Quality constraint
Softgoal	Softgoal
Task	Task
Evaluation	Evaluation
Change	Change
Mechanism	Mechanism
Effect	Effect

Source: Own.

S13 study does not have none concept name that it is equal to some Onto4SASGORE concept name. Then, we have the correlation with S13 study and Onto4SASGORE concepts as follows. We understand the Requirements S13 study concept as a Goal on the Onto4SASGORE. Software Function Universal, Program Software Function and Loaded Program Copy as SW_Agent, Observable State as Status, Program Copy Execution, Requirement Artifact, Runtime Requirement Artifact, Compliance Program Copy Execution, Monitoring Runtime Requirement Artifact and Adaptation Program Copy Execution as Resource and Change Runtime Requirement Artifact as Change. The correlation between the concepts from S13 study and the concepts from Onto4SASGORE is in Table 33.

For the studies from S14 to S23, we do an analysis in the ten most cited concepts, presented in Section 3.2. This Section present a SLR for context-aware system, the concepts are referent to a specifically feature of SAS, the context-aware. The ten concepts are: Location, Time, Activity, Role, Environment, Network, Person, User, Agent and Task.

The Location, Time, Role, Environment, Agent and Task concepts have the same name in our ontology. To other concepts, we understood Activity as Task or Action, Network

Table 33 – Study 13 concepts vs Onto4SASGORE concepts

S13 Concepts	Onto4SASGORE Correspondent Concepts
Requirement	Goal
Software Function Universal	SW_Agent
Program	SW_Agent
Software Function	SW_Agent
Loaded Program Copy	SW_Agent
Program Copy Execution	Resource
Observable State	Status
Requirement Artifact	Resource
Runtime Requirement Artifact	Change
Compliance Program Copy Execution	Resource
Monitoring Runtime Requirement Artifact	Sensor
Adaptation Program Copy Execution	Resource
Change Runtime Requirement Artifact	Change

Source: Own.

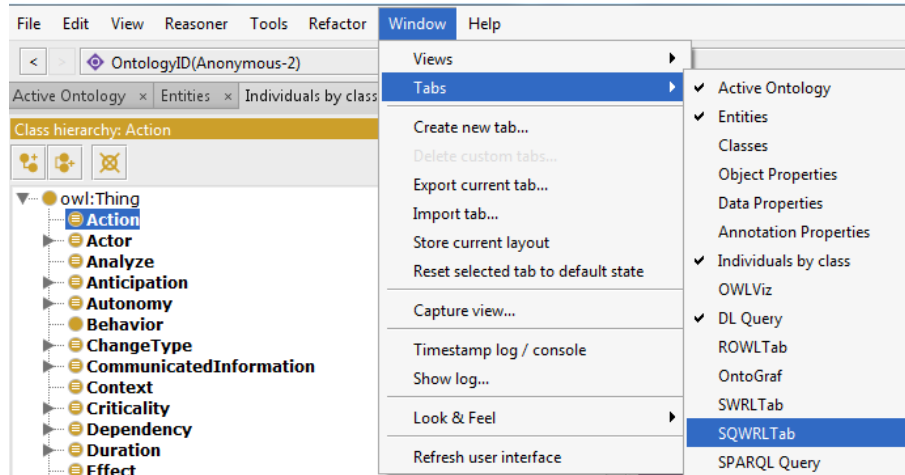
as Resource, Person as Human and User as Actor.

6.2 Verification and Validation

Uschold and Gruninger (1996) advocate that informal and formal questions provide a initial evaluation of an ontology. They also advocate the ontology evaluation by motivating scenarios. The motivating scenarios are story problems or examples. We have used some motivating scenario to create the competency questions. An ontology must be able to represent these questions using its terminology, and can characterize the answers to these questions using the axioms and definitions. SABiO (FALBO, 2014) suggests that verification criteria can be achieved by capability of the ontology elements (concepts, relations and axioms) to answer each competency question. For validation, SABiO suggests the instantiate of real world situations. Then, for verification, we use the concepts, relations and axioms of the Onto4SASGORE to answer the competency questions formulated in Chapter 4, for validation, we instantiated the Onto4SASGORE ontology with individuals extracted from the ZNN.com example, used to analyze the goal oriented-modeling languages in Section 3.4 of this thesis and the cleaner robot example.

The competence questions were implemented in SQWRL in Protégé. Figure 34 presents how to open the SQWRL tab in Protégé to write and run the rules. Figure 35 presents a screen to create or edit a SQWRL rule in Protégé.

Figure 34 – SQWRL Tab in Protégé



Source: Own.

For the verification criteria, we present the informal and formal competence questions and the answers as a concept, relations or axioms. The formal questions are expressed in SQWRL and the answers in Description Logic. For validation criteria, we also present a instance of the Znn.com (together the CQ - natural language and SQWRL - and its answer). After this, we also present a instance of the cleaner robot (still in the validation criteria).

The research questions that these evaluation criteria want answer are: 1) Is it possible

Figure 35 – New SQWRL rule screen

The screenshot shows a standard Java-style dialog box titled 'Edit'. It contains several input fields and a large text area. The 'Name' field is filled with 'CQ01'. The 'Comment' field is filled with 'What are the sensors in the scope of the project?'. The 'Status' field is filled with 'Ok'. Below these fields is a large text area containing the SQWRL rule: 'Sensor(?sen)->sqwrl:select(?sen)'. At the bottom of the dialog are two buttons: 'Cancel' and 'Ok'.

Source: Own.

to represent real-world situations with the Onto4SASGORE? 2) Can this ontology be more expressive than the analyzed goal-oriented languages to specify a self-adaptive system?

We present the ID and the competence question in natural language, after, we present the competence question in SQWRL, and the answer as ontology element, if the answer is an axiom, it will be expressed in Description Logic and, lastly, the instance to represent the Znn.com example. All the questions about number "How many" were answered through the Protégé. Figure 36 presents a Protégé screen for a question with result and Figure 37 presents the log when the questions does not return answer. For lastly, all answer in *italic* have been achieved through the implementation of the rules in Protégé SQWRL Tab.

CQ01: What are the sensors in the scope of the system?

SQWRL: `Sensor(?sen)->sqwrl:select(?sen)`

Answer: A55.

Znn.com Instance: Load balancer.

CQ02: What is the sensor status?

SQWRL: `Sensor(?sen) ^ hasStatus(?sen,?sta)->sqwrl:select(?sta)`

Answer: `Sensor \sqsubseteq \exists hasStatus.Status.`

Znn.com Instance: High load; load drops.

CQ03: What are the effectors in the scope of the system?

SQWRL: `Effector(?eff)->sqwrl:select(?eff)`

Answer: Effector. AC06.

Figure 36 – Result for CQ71

Name	Query	Comment
CQ52	autogen0:Change(?cha) ^ autogen0:hasAnticipation(?cha, ?ant) -> sqwrl:select(?ant)	What is the anticipation of the change?
CQ53	autogen0:Change(?cha) ^ autogen0:hasSource(?cha, ?sou) -> sqwrl:select(?sou)	What is the source of the change?
CQ54	autogen0:Context(?ctx) -> sqwrl:select(?ctx)	What are the context involved on project?
CQ55	autogen0:Context(?ctx) ^ autogen0:hasGoal(?ctx, ?goa) -> sqwrl:select(?goa)	What is the goal of the context?
CQ56	autogen0:Context(?ctx) ^ autogen0:hasLocation(?ctx, ?lct) -> sqwrl:select(?lct)	What is the location of the context?
CQ57	autogen0:Context(?ctx) ^ autogen0:hasTime(?ctx, ?tme) -> sqwrl:select(?tme)	What is the time of the context?
CQ58	autogen0:Context(?ctx) ^ autogen0:hasAction(?ctx, ?aci) -> sqwrl:select(?aci)	What is the action of the context?
CQ59	autogen0:Context(?ctx) ^ autogen0:hasRole(?ctx, ?rle) -> sqwrl:select(?rle)	What is the role of the context?
CQ60	autogen0:Context(?ctx) ^ autogen0:hasStatus(?ctx, ?stt) -> sqwrl:select(?stt)	What is the status of the context?
CQ61	autogen0:Mechanism(?mec) -> sqwrl:select(?mec)	What are the mechanism of the project?
CQ62	autogen0:Mechanism(?mec) ^ autogen0:hasMechanismType(?mec, ?mtp) -> sqwrl:select(?mtp)	What is the type of the mechanism?
CQ63	autogen0:Mechanism(?mec) ^ autogen0:hasTimeliness(?mec, ?tin) -> sqwrl:select(?tin)	What is the timeliness of the mechanism?
CQ64	autogen0:Mechanism(?mec) ^ autogen0:hasTriggering(?mec, ?tgg) -> sqwrl:select(?tgg)	What is the triggering of the mechanism?
CQ65	autogen0:Mechanism(?mec) ^ autogen0:hasAutonomy(?mec, ?atn) -> sqwrl:select(?atn)	What is the autonomy of the mechanism?
CQ66	autogen0:Mechanism(?mec) ^ autogen0:hasOrganization(?mec, ?org) -> sqwrl:select(?org)	What is the organization of the mechanism?
CQ67	autogen0:Mechanism(?mec) ^ autogen0:hasScope(?mec, ?scp) -> sqwrl:select(?scp)	What is the scope of the mechanism?
CQ68	autogen0:Mechanism(?mec) ^ autogen0:hasMechanismDuration(?mec, ?mdr) -> sqwrl:select(?mdr)	What is the duration of the mechanism?
CQ69	autogen0:InterfaceMechanism(?ifm) -> sqwrl:select(?ifm)	What is the interface of the mechanism?
CQ70	autogen0:Plan(?pla) ^ autogen0:achievesGoal(?pla, ?goa) -> sqwrl:select(?goa)	What is the goal of the plan?
CQ71	autogen0:Goal(?goa) -> sqwrl:count(?goa)	How many goals are there on project?
CQ72	autogen0:Functional_Goal(?fgo) -> sqwrl:count(?fgo)	How many functional goals are there on project?
CQ73	autogen0:NonFunctional_Goal(?nfg) -> sqwrl:count(?nfg)	How many non-functional goals are there on project?
CQ74	autogen0:Hardgoal(?pgo) -> sqwrl:count(?pgo)	How many Hardgoals are there on project?
CQ75	autogen0:Hardgoal(?rgo) -> sqwrl:count(?rgo)	How many hardgoals are there on project?
CQ76	autogen0:Softgoal(?sgo) -> sqwrl:count(?sgo)	How many softgoals are there on project?
CQ77	autogen0:Quality_Constraint(?qct) -> sqwrl:count(?qct)	How many quality constraints are there on project?
CQ78	autogen0:Change(?cha) -> sqwrl:count(?cha)	How many changes are there on project?

SQWRL Queries OWL 2 RL CQ71

*8^^xsd:int

count(?goa)

Save as CSV... Rerun Close

Source: Own.

Figure 37 – Result for CQ74

SQWRL Queries OWL 2 RL CQ71

See the 'OWL 2 RL' subtab for more information on this reasoner.

Executing queries in this tab does not modify the ontology.

Using Drools for query execution.

See the CQ73 tab to review results of the SQWRL query.

The query took 8142 milliseconds. 1 row was returned.

'CQ73' tab closed.

SQWRL query CQ74 did not generate any result.

Source: Own.

Znn.com Instance: SW_Agent.

CQ04: What are the effects of the system?

SQWRL: Effect(?efe)->sqwrl:select(?efe)

Answer: A51; A52; A53; A54.

Znn.com Instance: Increment the server pool size; Decrement the server pool size.

CQ05: What are the actors of the system?

SQWRL: Actor(?act)->sqwrl:select(?act)

Answer: Actor $\sqsubseteq \exists$ plays(Role) \sqcap performsAction (Action).

Znn.com Instance: Clients; Sys-admins; SW_Agent.

CQ06: What are the roles played by the actor?

SQWRL: Actor(?act) \wedge plays(?act,?rol)->sqwrl:select(?rol)

Answer: Actor $\sqsubseteq \exists$ plays(Role).

Znn.com Instance: Clients; SW_Agent.

CQ07: What are the actions held by the actor?

SQWRL: Actor(?act) \wedge performsAction(?act, ?aci) \wedge sqwrl: makeSet(?s, ?aci) \wedge sqwrl: groupBy
(?s, ?act) -> sqwrl:select(?s)

Answer: A13.

Znn.com Instance: Requests.

CQ08: What are the actions on the system?

SQWRL: Action(?aci)->sqwrl:select(?aci)

Answer: Actor $\sqsubseteq \exists$ plays(Role).

Znn.com Instance: Monitor server load and bandwidth of server-client connections.

CQ09: What are the Effects of the action?

SQWRL: Action(?aci) \wedge hasEffect(?aci,?efe)-> sqwrl:select(?efe)

Answer: Action $\sqsubseteq \exists$ hasEffect (Effect)

Znn.com Instance: Increment the server pool size; Decrement the server pool size.

CQ10: What is the predictability of the effect?

SQWRL: Effect(?efe) \wedge hasPredictability(?efe,?pre)->sqwrl:select(?pre)

Answer: Effect $\sqsubseteq \exists$ hasPredictability.Predictability

Znn.com Instance: Deterministic.

CQ11: What is the resilience of the effect?

SQWRL: Effect(?efe) \wedge hasResilience(?efe,?rsl)->sqwrl:select(?rsl)

Answer: Effect $\sqsubseteq \exists$ hasResilience.Resilience

Znn.com Instance: Vulnerable.

CQ12: What is the criticality of the effect?

SQWRL: Effect(?efe) \wedge hasCriticality(?efe,?cri)->sqwrl:select(?cri)

Answer: Effect $\sqsubseteq \exists$ hasCriticality.Criticality

Znn.com Instance: Mission-critical.

CQ13: What is the overhead of the effect?

SQWRL: `Effect(?efe) ^ hasOverhead(?efe, ?ovh) -> sqwrl:select(?ovh)`

Answer: `Effect \sqsubseteq \exists hasOverhead.Overhead`

Znn.com Instance: –

CQ14: What are the resources of the system?

SQWRL: `Resource(?res) -> sqwrl:select(?res)`

Answer: A40; A41; A42.

Znn.com Instance: Servers.

CQ15: What are the Plans of the system?

SQWRL: `Plan(?pla) -> sqwrl:select(?pla)`

Answer: A01; AC04.

Znn.com Instance: To balance requests across a pool of replicated servers; Size pool is dynamically adjusted; To balance server utilization; Add more servers to the pool; To change the quality of the content.

CQ16: What are the Change Plans of the system?

SQWRL: `ChangePlan(?chp) -> sqwrl:select(?chp)`

Answer: A42; AC09.

Znn.com Instance: *Add_more_servers_to_the_pool*

To_sever_multimedia_mode

To_change_the_quality_of_the_content

To_reduce_the_pool_size

To_server_minimalist_textual_content

CQ17: What are the Symptoms of the system?

SQWRL: `Symptom(?sym) -> sqwrl:select(?sym)`

Answer: A40; Ac07.

Znn.com Instance: *Server_load*

Response_time

Bandwith_of_server-clients_connections

CQ18: What are the Change Requests of the system?

SQWRL: `ChangeRequest(?chr) -> sqwrl:select(?chr)`

Answer: A41; AC08.

Znn.com Instance: –

CQ19: What are the goals of the system?

SQWRL: `Goal(?goa)->sqwrl:select(?goa)`

Answer: A07; A14; A15; A16; A17; A22; A23.

Znn.com Instance: *Spikes_in_new_requests*

Within_operating_budget

High_performance

High_fidelity

Reasonable_response_time_range

Cost_efficiency

To_server_news_content

Serve_news

CQ20: What are the Functional Goals of the system?

SQWRL: `Functional_Goal(?fgo)->sqwrl:select(?fgo)`

Answer: `Functional_Goal` \sqsubseteq `Goal`.

Znn.com Instance: To serve news content.

CQ21: What are the actions that achieve the functional goal X?

SQWRL: `Action(?aci) ^ achieves(?aci,?fgo)-> sqwrl:select(?aci)`

Answer: `Action` $\sqsubseteq \exists$ `achieves.Functional_Goal`

Znn.com Instance: –

CQ22: What are the Non Functional Goals of the system?

SQWRL: `NonFunctional_Goal(?nfg)->sqwrl:select(?nfg)`

Answer: `NonFunctional_Goal` \sqsubseteq `Goal`.

Znn.com Instance: -Within operating budget; -High performance; -High fidelity;

Spikes_in_new_requests

CQ23: What are the actions that contribute to the Non functional goal?

SQWRL: `Action(?aci) ^ contributes(?aci,?nfg)-> sqwrl:select(?aci)`

Answer: `Action` $\sqsubseteq \exists$ `contributes.NonFunctional_Goal`

Znn.com Instance:–

CQ24: What are the Hardgoals of the system?

SQWRL: `Hardgoal(?hgo)->sqwrl:select(?hgo)`

Answer: `Hardgoal` \sqsubseteq `Goal`.

Znn.com Instance: –

CQ25: What are the Tasks of the system?

SQWRL: Task(?tsk)->sqwrl:select(?tsk)

Answer:

Znn.com Instance:

CQ26: What are the Softgoals of the system?

SQWRL: Softgoal(?sgo)->sqwrl:select(?sgo)

Answer: Softgoal \sqsubseteq Goal.

Znn.com Instance: Reasonable response time range; Cost efficiency.

CQ27: What are the Quality Constraints of the system?

SQWRL: Quality_Constraint(?qct)->sqwrl:count(?qct)

Answer: Quality_Constraint \sqsubseteq NonFunctional_Goal.

Znn.com Instance: Single server; content resolution is high.

CQ28: What is the duration of the goal?

SQWRL: Goal(?goa) \wedge hasDuration(?goa,?gdt)->sqwrl:select(?gdt)

Answer: Goal $\sqsubseteq \exists$ hasDuration.Duration

Znn.com Instance: –

CQ29: What is the flexibility of the goal?

SQWRL: Goal(?goa) \wedge hasFlexibility(?goa,?flx)->sqwrl:select(?flx)

Answer: Goal $\sqsubseteq \exists$ hasFlexibility.Flexibility

Znn.com Instance: –

CQ30: What is the dependency of the goal?

SQWRL: Goal(?goa) \wedge hasDependency(?goa,?dpc)->sqwrl:select(?dpc)

Answer: Goal $\sqsubseteq \exists$ hasDependency.Dependency

Znn.com Instance: –

CQ31: What is the evolution of the goal?

SQWRL: Goal(?goa) \wedge hasEvolution(?goa,?evl)->sqwrl:select(?evl)

Answer: Goal $\sqsubseteq \exists$ hasEvolution.Evolution

Znn.com Instance: –

CQ32: What are the inputs of the system?

SQWRL: Input(?inp)->sqwrl:select(?inp)

Answer: Input. hasInput.

Znn.com Instance: –

CQ33: What is the input of the monitor?

SQWRL: $\text{Monitor}(\text{?mon}) \wedge \text{hasInput}(\text{?mon}, \text{?ipt}) \rightarrow \text{sqwrl:select}(\text{?ipt})$

Answer: $\text{Monitor} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasInput.Input}$

Znn.com Instance: –

CQ34: What is the input of the analyze?

SQWRL: $\text{Analyze}(\text{?anl}) \wedge \text{hasInput}(\text{?anl}, \text{?ipt}) \rightarrow \text{sqwrl:select}(\text{?ipt})$

Answer: $\text{Analyze} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasInput.Input}$

Znn.com Instance: –

CQ35: What is the input of the plan?

SQWRL: $\text{Plan}(\text{?pla}) \wedge \text{hasInput}(\text{?pla}, \text{?ipt}) \rightarrow \text{sqwrl:select}(\text{?ipt})$

Answer: $\text{Plan} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasInput.Input}$

Znn.com Instance: –

CQ36: What is the input of the execute?

SQWRL: $\text{Execute}(\text{?exe}) \wedge \text{hasInput}(\text{?exe}, \text{?ipt}) \rightarrow \text{sqwrl:select}(\text{?ipt})$

Answer: $\text{Execute} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasInput.Input}$

Znn.com Instance: –

CQ37: What is the input of the effector?

SQWRL: $\text{Effector}(\text{?eff}) \wedge \text{hasInput}(\text{?eff}, \text{?ipt}) \rightarrow \text{sqwrl:select}(\text{?ipt})$

Answer: $\text{Effector} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasInput.Input}$

Znn.com Instance: –

CQ38: What are the outputs of the system?

SQWRL: $\text{Output}(\text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

Answer: $\text{Output} . \text{hasOutput}.$

Znn.com Instance: –

CQ39: What is the output of the sensor?

SQWRL: $\text{Sensor}(\text{?sen}) \wedge \text{hasOutput}(\text{?sen}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

Answer: $\text{Sensor} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasOutput.Output}$

Znn.com Instance: –

CQ40: What is the output of the monitor?

SQWRL: $\text{Monitor}(\text{?mon}) \wedge \text{hasOutput}(\text{?mon}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

Answer: $\text{Monitor} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasOutput.Output}$

Znn.com Instance: –

CQ41: What is the output of the analyze?

SQWRL: $\text{Analyze}(\text{?an1}) \wedge \text{hasOutput}(\text{?an1}, \text{? out}) \rightarrow \text{sqwrl:select}(\text{?out})$

Answer: $\text{Analyze} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasOutput.Output}$

Znn.com Instance: –

CQ42: What is the output of the plan?

SQWRL: $\text{Plan}(\text{?pla}) \wedge \text{hasOutput}(\text{?pla}, \text{? out}) \rightarrow \text{sqwrl:select}(\text{?out})$

Answer: $\text{Plan} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasOutput.Output}$

Znn.com Instance: –

CQ43: What is the output of the execute?

SQWRL: $\text{Execute}(\text{?exe}) \wedge \text{hasOutput}(\text{?exe}, \text{? out}) \rightarrow \text{sqwrl:select}(\text{?out})$

Answer: $\text{Execute} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasOutput.Output}$

Znn.com Instance: –

CQ44: What are the preconditions of the Action X?

SQWRL: $\text{Action}(\text{?aci}) \wedge \text{hasPrecondition}(\text{?aci}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

Answer: $\text{Action} \sqsubseteq \exists \text{hasPrecondition.Precondition}$

Znn.com Instance: –

CQ45: What are the preconditions of the monitor?

SQWRL: $\text{Monitor}(\text{?mon}) \wedge \text{hasPrecondition}(\text{?mon}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

Answer: $\text{Monitor} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasPrecondition.Precondition}$

Znn.com Instance: –

CQ46: What are the preconditions of the analyze?

SQWRL: $\text{Analyze}(\text{?an1}) \wedge \text{hasPrecondition}(\text{?an1}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

Answer: $\text{Analyze} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasPrecondition.Precondition}$

Znn.com Instance: –

CQ47: What are the preconditions of the plan?

SQWRL: $\text{Plan}(\text{?pla}) \wedge \text{hasPrecondition}(\text{?pla}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

Answer: $\text{Plan} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasPrecondition.Precondition}$

Znn.com Instance: –

CQ48: What are the preconditions of the execute?

SQWRL: $\text{Execute}(\text{?exe}) \wedge \text{hasPrecondition}(\text{?exe}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

Answer: $\text{Execute} \sqsubseteq \text{Action} . \text{Action} \sqsubseteq \exists \text{hasPrecondition.Precondition}$

Znn.com Instance: –

CQ49: What are the changes of the system?

SQWRL: $\text{Change}(\text{?cha}) \rightarrow \text{sqwrl}:\text{select}(\text{?cha})$

Answer: A24; A25; A26; A27; A28.

Znn.com Instance: –

CQ50: What is the frequency of the change?

SQWRL: $\text{Change}(\text{?cha}) \wedge \text{hasFrequency}(\text{?cha}, \text{?fre}) \rightarrow \text{sqwrl}:\text{select}(\text{?fre})$

Answer: $\text{Change} \sqsubseteq \exists \text{hasFrequency.Frequency}$

Znn.com Instance: –

CQ51: What is the type of the change?

SQWRL: $\text{Change}(\text{?cha}) \wedge \text{hasChangeType}(\text{?cha}, \text{?ctp}) \rightarrow \text{sqwrl}:\text{select}(\text{?ctp})$

Answer: $\text{Change} \sqsubseteq \exists \text{hasType.Type}$

Znn.com Instance: –

CQ52: What is the anticipation of the change?

SQWRL: $\text{Change}(\text{?cha}) \wedge \text{hasAnticipation}(\text{?cha}, \text{?ant}) \rightarrow \text{sqwrl}:\text{select}(\text{?ant})$

Answer: $\text{Change} \sqsubseteq \exists \text{hasAnticipation.Anticipation}$

Znn.com Instance: –

CQ53: What is the source of the change?

SQWRL: $\text{Change}(\text{?cha}) \wedge \text{hasSource}(\text{?cha}, \text{?sou}) \rightarrow \text{sqwrl}:\text{select}(\text{?sou})$

Answer: $\text{Change} \sqsubseteq \exists \text{hasSource.Source}$

Znn.com Instance: –

CQ54: What are the contexts involved on system?

SQWRL: $\text{Context}(\text{?ctx}) \rightarrow \text{sqwrl}:\text{select}(\text{?ctx})$

Answer: A06.

Znn.com Instance: To serve text-only content or to server low resolution content or to serve high resolutions content; To add or remove server.

CQ55: What is the goal of the context?

SQWRL: $\text{Context}(\text{?ctx}) \wedge \text{hasGoal}(\text{?ctx}, \text{?goa}) \rightarrow \text{sqwrl}:\text{select}(\text{?goa})$

Answer: $\text{Context} \sqsubseteq \exists \text{hasGoal.Goal}$

Znn.com Instance: –

CQ56: What is the location of the context?

SQWRL: $\text{Context}(\text{?ctx}) \wedge \text{hasLocation}(\text{?ctx}, \text{?lct}) \rightarrow \text{sqwrl:select}(\text{?lct})$

Answer: $\text{Context} \sqsubseteq \exists \text{hasLocation.Location}$

Znn.com Instance: –

CQ57: What is the time of the context?

SQWRL: $\text{Context}(\text{?ctx}) \wedge \text{hasTime}(\text{?ctx}, \text{?tme}) \rightarrow \text{sqwrl:select}(\text{?tme})$

Answer: $\text{Context} \sqsubseteq \exists \text{hasTime.Time}$

Znn.com Instance: –

CQ58: What is the action of the context?

SQWRL: $\text{Context}(\text{?ctx}) \wedge \text{hasAction}(\text{?ctx}, \text{?aci}) \rightarrow \text{sqwrl:select}(\text{?aci})$

Answer: $\text{Context} \sqsubseteq \exists \text{hasAction.Action}$

Znn.com Instance: –

CQ59: What is the role of the context?

SQWRL: $\text{Context}(\text{?ctx}) \wedge \text{hasRole}(\text{?ctx}, \text{?rle}) \rightarrow \text{sqwrl:select}(\text{?rle})$

Answer: $\text{Context} \sqsubseteq \exists \text{hasRole.Role}$

Znn.com Instance: –

CQ60: What is the status of the context?

SQWRL: $\text{Context}(\text{?ctx}) \wedge \text{hasStatus}(\text{?ctx}, \text{?stt}) \rightarrow \text{sqwrl:select}(\text{?stt})$

Answer: $\text{Context} \sqsubseteq \exists \text{hasStatus.Status}$

Znn.com Instance: –

CQ61: What are the mechanisms of the system?

SQWRL: $\text{Mechanism}(\text{?mec}) \rightarrow \text{sqwrl:select}(\text{?mec})$

Answer: A44; A45; A46; A47; A48; A49; A50.

Znn.com Instance: –

CQ62: What is the type of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasMechanismType}(\text{?mec}, \text{?mtp}) \rightarrow \text{sqwrl:select}(\text{?mtp})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasMechanismType.MechanismType}$

Znn.com Instance: adjusts the server pool size: - to serve minimalist textual content during peak times; - to multimedia mode in combination with reducing the pool size to reduce operating cost.

CQ63: What is the timeliness of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasTimeliness}(\text{?mec}, \text{?tln}) \rightarrow \text{sqwrl:select}(\text{?tln})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasTimeliness.Timeliness}$

Znn.com Instance: –

CQ64: What is the triggering of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasTriggering}(\text{?mec}, \text{?tgg}) \rightarrow \text{sqwrl:select}(\text{?tgg})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasTriggering.Triggering}$

Znn.com Instance: –

CQ65: What is the autonomy of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasAutonomy}(\text{?mec}, \text{?atn}) \rightarrow \text{sqwrl:select}(\text{?atn})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasAutonomy.Autonomy}$

Znn.com Instance: –

CQ66: What is the organization of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasOrganization}(\text{?mec}, \text{?org}) \rightarrow \text{sqwrl:select}(\text{?org})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasOrganization.Organization}$

Znn.com Instance: –

CQ67: What is the scope of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasScope}(\text{?mec}, \text{?scp}) \rightarrow \text{sqwrl:select}(\text{?scp})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasScope.Scope}$

Znn.com Instance: –

CQ68: What is the duration of the mechanism?

SQWRL: $\text{Mechanism}(\text{?mec}) \wedge \text{hasMechanismDuration}(\text{?mec}, \text{?mdr}) \rightarrow \text{sqwrl:select}(\text{?mdr})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasMechanismDuration.MechanismDuration}$

Znn.com Instance: –

CQ69: What is the interface mechanism?

SQWRL: $\text{InterfaceMechanism}(\text{?ifm}) \rightarrow \text{sqwrl:select}(\text{?ifm})$

Answer: $\text{Mechanism} \sqsubseteq \exists \text{hasMechanismInterface.MechanismInterface}$

Znn.com Instance: –

CQ70: What is the goal of the plan?

SQWRL: $\text{Plan}(\text{?pla}) \wedge \text{achievesGoal}(\text{?pla}, \text{?goa}) \rightarrow \text{sqwrl:select}(\text{?goa})$

Answer: $\text{Plan} \sqsubseteq \exists \text{hasGoal.Goal}$

Znn.com Instance: –

CQ71: How many goals are there in the system?

SQWRL: $\text{Goal}(\text{?goa}) \rightarrow \text{sqwrl:count}(\text{?goa})$

Answer: Goal.

Znn.com Instance: 8.

CQ72: How many functional goals are there in the system?

SQWRL: `Functional_Goal(?fgo)->sqwrl:count(?fgo)`

Answer:

Znn.com Instance: 1.

CQ73: How many non-functional goals are there in the system?

SQWRL: `NonFunctional_Goal(?nfg)->sqwrl:count(?nfg)`

Answer: `NonFunctional_Goal`.

Znn.com Instance: 4.

CQ74: How many Hardgoals are there in the system?

SQWRL: `Hardgoal(?pgo)->sqwrl:count(?pgo)`

Answer: `Hardgoal`.

Znn.com Instance: any result.

CQ75: How many tasks are there in the system?

SQWRL: `Task(?tsk)->sqwrl:count(?tsk)`

Answer:

Znn.com Instance: any result.

CQ76: How many softgoals are there in the system?

SQWRL: `Softgoal(?sgo)->sqwrl:count(?sgo)`

Answer: `Softgoal`.

Znn.com Instance: 2.

CQ77: How many quality constraints are there in the system?

SQWRL: `Quality_Constraint(?qct)->sqwrl:count(?qct)`

Answer: `Quality_Constraint`.

Znn.com Instance: any result.

CQ78: How many changes are there in the system?

SQWRL: `Change(?cha)->sqwrl:count(?cha)`

Answer: `Change`.

Znn.com Instance: 1.

CQ79: How many mechanisms are there in the system?

SQWRL: Mechanism(?mec)->sqwrl:count(?mec)

Answer: Mechanism.

Znn.com Instance: any result.

(Change X = Spikes_in_new_requests)

CQ80: How many mechanisms are there in the Change X?

SQWRL: hasMechanism(ChangeX, ?mec)->sqwrl:select(ChangeX)^ sqwrl:count(?mec)

Answer: Change. hasMechanism.

Znn.com Instance: any result.

CQ81: What are the mechanisms of the Change X?

SQWRL: hasMechanism(ChangeX, ?mec)->sqwrl:select(?mec)

Answer: Change $\sqsubseteq \exists$ hasMechanism.Mechanism

Znn.com Instance:

CQ82: How many effects are there in the system?

SQWRL: Effect(?efe)->sqwrl:count(?efe)

Answer: Effect.

Znn.com Instance: 2.

CQ83: How many effects does the Change X have?

SQWRL: hasEffect(ChangeX, ?efe)->sqwrl:select (ChangeX)^ sqwrl:count(?efe)

Answer: Change. hasEffect.

Znn.com Instance: any result.

CQ84: What are the effects produced by the Change X?

SQWRL: hasEffect(ChangeX, ?efe)->sqwrl:select(?efe)

Answer: Change $\sqsubseteq \exists$ hasEffect.Effect

Znn.com Instance:

CQ85: How many actors are there in the system?

SQWRL: Actor(?act)->sqwrl:count(?act)

Answer: Actor.

Znn.com Instance: any result.

CQ86: How many contexts are there in the system?

SQWRL: Context(?ctx)->sqwrl:count(?ctx)

Answer: Context.

Znn.com Instance: any result.

CQ87: How many resources are there in the system?

SQWRL: `Resource(?res)->sqwrl:count(?res)`

Answer: Resource.

Znn.com Instance: 12.

CQ88: How many sensors are there in the system?

SQWRL: `Sensor(?sen)->sqwrl:count(?sen)`

Answer: Sensor.

Znn.com Instance: 1.

CQ89: How many effectors are there in the system?

SQWRL: `Effector(?efe)->sqwrl:count(?efe)`

Answer: Effector.

Znn.com Instance: 1.

CQ90: What are the goals of static evolution?

SQWRL: `Goal(?goa) ^ hasEvolution(Static)->sqwrl:select(?goa)`

Answer: $\text{Goal} \sqsubseteq \exists \text{hasEvolution.Static}$

Znn.com Instance:

CQ91: What are the goals of dynamic evolution?

SQWRL: `Goal(?goa) ^ hasEvolution(Dynamic)->sqwrl:select(?goa)`

Answer: $\text{Goal} \sqsubseteq \exists \text{hasEvolution.Dynamic}$

Znn.com Instance:

CQ92: How does the system achieve the goals?

SQWRL: `Goal(?goa) ^ hasAction(?goa,?aci) ^ sqwrl: makeSet (?s1, ?goa) ^ sqwrl: makeSet (?s1,?aci) ^ sqwrl:union(?s3, ?s1, ?s2) ^ sqwrl:groupBy(?s3,?goa)->sqwrl:select(?s3)`

Answer: A13.

Znn.com Instance:

CQ93: How does the system achieve the goal X?

SQWRL: `Goal(goalX) ^ hasAction(goalX,?aci) ^ sqwrl: makeSet (?s2, ?aci) ^ sqwrl: groupBy (?s2, ?aci)->sqwrl:select(?s2)`

Answer: A07; A13.

Znn.com Instance:

We note that the Onto4SASGORE is able to answer the competency questions with its elements (concepts, relations and axioms). To continue the validation criteria, we instantiate the ontology with individuals extracted from the cleaner robot example (Appendix F). For this, it was used Protegé to create inferences through the reasoner Hermit. The individuals by class are:

Action: Star_in_new_location

Actor: Cleaner_robot

Analyze: Dock (hasOutput: Try_to_dock; before: To_dock);

Way_to_base (hasOutput: Go_to_base; before: Return_to_base).

Behavior: Dirt_detection (dependsOf: Senses_dirt);

Room_crossing (DependsOf: Crisscrosses);

Spiraling (DependsOf: Concentrated_area);

Wall_following (dependsOf: Furniture_and_obstacles).

Change: Clean_at_programed_time (hasSource: External; hasAnticipation: Foreseen; hasMechanism: Time_clock; hasChangeType: FunctionalType; hasFrequency: Rare; hasEffect: Clean_environment.)

DomainAssumption: HomeBase_always_pluggedin

Goal: Calculates_room_size (hasEvolution: Dynamic; hasDuration: Temporary.);

Clean_filter (hasFlexibility: Rigid.);

Empty_robot_bin (hasFlexibility: Rigid.);

Schedule_cleaning.

NonFunctional_Goal: Ensure_full_cleaning_coverage (hasFlexibility: Constrained)

Softgoal: Adjust_cleaning_time_appropriately.

Context: Concentrated_area;

Crisscrosses;

Furniture_and_obstacles;

Room_size (hasGoal: Calculates_room_size);

Senses_dirt;

Time_to_clean (hasGoal: Schedule_cleaning; hasTime: Time_clean).

Effect: Clean_Environment (hasPredictability: Deterministic;

hasOverhead: Sensor_dirty_clean_fail;

hasCriticality: Harmless;

hasResilience: Resilient).

Environment: Carpet; Laminate; Tile; Vinyl; Wood.

Error: Sensor_dirty

Execute: Charge; Get_on_base.

Mechanism: Time_clock (hasTimeliness: Guaranteed; hasScope: Global; hasMechanism-Duration: Short; hasMechanismType: Parametric; hasAutonomy: Autonomous; hasTriggering: Time-Trigger; hasOrganization: Centralized).

Monitor: Find_dock (hasOutput: Dock_location; before: Dock);
Find_infrared_signal (hasOutput: Base_location; before: Way_to_base).
Failure: Sensor_dirty_clean_fail (generatesError: Sensor_dirty).
Planner: Return_to_base (before: Get_on_base; hasOutput: To_get_on_base.);
To_dock (hasOutput: To_get_charge; before: Charge).
Resource: Battery (status: xsd:string "empty_red"; status: xsd:string "charging_amberpulse"; status: xsd:string "fully_gree");
Bin_release_button;
Brush;
Dust_bin;
Filter;
Infrared_signal;
White_light.
ChangeRequest: Go_to_base; Try_to_dock.
ChangePlan: To_get_charge; To_get_on_base.
Symptom: Base_location; Dock_location.
Sensor: Dirt_detection_Sensor; Infrared_Sensor.
Status: Battery_charging_amberpulse;
Battery_complete_green;
Battery_empty_red.
Task: Powerful_vacuum_picks; Side_brush_sweeps; Two_brushs_scoop.
Time: Time_clean.

The Hermit 1.3.8.423 reasoner processed the ontology in 8909360 ms (2,47 hours). After run the reasoner, the inferences created are in the following:

Actor: The **Cleaner_robot** is also a **SW_Agent**.

Analyze: The **Dock** comes *before* **Charge** and *after* **Find_dock**; The **Way_to_base** comes *before* **Get_on_base** and *after* **Find_infrared_signal**.

Change: The **Clean_at_programmed_time** has duration (*hasDuration*) **Temporary** and has evolution (*hasEvolution*) **Dynamic**.

Goal: The **Calculates_room_size** is also a **Change** and a **Quality_Constraint**;

The **Clean_filter** is also a **Hardgoal**;

The **Empty_robot_bin** is also a **Hardgoal**.

NonFunctional_Goal: The **Ensure_full_cleaning_coverage** is also a **Quality_Constraint**.

Softgoal: The **Adjust_cleaning_time_appropriately** has flexibility (*hasFlexibility*) **Unconstrained**.

Execute: The **Charge** comes *after* **To_dock**, **Find_dock** and **Dock**; The **Get_on_base** comes *after* **Return_to_base**, **Way_to_base** and **Find_infrared_signal**.

Monitor: The **Find_dock** comes *before* **To_Dock** and **Charge**; The **Find_infrared_signal** comes *before* **Get_on_base** and **Return_to_base**.

Planner: The **Return_to_base** comes *after* **Way_to_base** and **Find_infrared_signal**; The **To_dock** comes *after* **Find_dock** and **Dock**.

We conclude that the Onto4SASGORE is able to represent real-world situations because it was instantiated with individuals extracted from the ZNN.com. Therefore, with the ontology proposed in this thesis, we can represent the self-adaptive systems domain resulting in a capturing of some elements that the goal modeling languages for SAS, investigated in Chapter 3.4 were unable to capture. The set of concepts shows a guide to generation of requirements. The difference of our ontology is that the set of concepts is wider and can get more details of some SAS. Instances of the concepts related to context, goal, users, change and adaptation were captured by the Onto4SASGORE ontology, which shows a major expressiveness in relation to the studies from SLRs and the analyzed goal-oriented modeling languages.

6.3 Usefulness and Easy of Use

Case studies provide deeper understanding of the phenomena under study in its real context. They are an empirical method aimed at investigating contemporary phenomena in their context. The research methodology is a case study when the researcher is studying the effects of a change, for example, in pre- and post-event studies. One of the key characteristics of a case study is that it is of flexible type, coping with the complex and dynamic characteristics of real world phenomena, like software engineering (WOHLIN et al., 2012).

Wohlin et al. (2012) claim that a case study may contain elements of other research methods, for example, a survey may be conducted within a case study. Because of this affirmative, we perform a survey that it is explained in this section.

To conduct this case study, we used the guidelines by Perry, Sim and Easterbrook (2004) and Wohlin et al. (2012).

This case study has an exploratory objective that is to use the Onto4SASGORE process. The case which is studied is the ease of use of the Onto4SASGORE ontology. The collect data method is a survey.

Considering our objective to perform this case study, we set the research questions:

1. How to elicit and specify requirements for SAS using Onto4SASGORE?
2. How is Onto4SASGORE process in practice?
3. How does Onto4SASGORE ontology aid the requirements engineer in practice?

The propositions for this study are:

1. The requirements for SAS can be elicited and specified through Onto4SASGORE process;
2. An ontology can aid requirements engineers when performing requirements activities for self-adaptive systems.

This case study is a descriptive type, because it describes sequence of events and underlying mechanisms. The units of analysis are the Onto4SASGORE process and the Onto4SASGORE ontology.

Wohlin et al. (2012) advocates that the validity must be addressed during all previous phases of the case study. We consider four aspects of validity, according Wohlin et al. (2012), and we present the threats to validity and how to minimize.

Construct validity reflect to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions. The threat to this validity is the possibility of the questions be interpreted in a different way by the subjects. To minimize this, we use the TAM pattern to create the questions.

Internal validity is of concern when causal relations are examined. The threat of this aspect of validity is the subject to elicit and specify the system in a wrong way in relation at Onto4SASGORE ontology. To minimize this threat, we offer a site with all concepts and descriptions about the concepts.

External validity This aspect of validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case. The threat to this validity is not get relevant findings to extent this study case to others cases with common characteristics. To minimize this, we get a SAS because the onto4SASGORE is proposed to self-adaptive systems.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researches. The threat to this validity is not be clearer how to use the ontology. To minimize this threat, we create a process and expose the process online for the subjects to answer the questions.

We have performed a case study through a form online with requirements engineers, in order to apply the Onto4SASGORE process. Each requirements engineer received one description of the LAS case (SILVASOUZA; MYLOPOULOS, 2015), and elicited and specified requirements using the Onto4SASGORE ontology available in <http://onto4sas.wiki-site.com/> and they have the Onto4SASGORE process as guide. For the purpose of evaluating the elicitation and specification activities, we have just used the requirements discovery sub-process and the requirements specification activity. Although requirements classification and organization and requirements prioritization and negotiation activities are very important to the requirements elicitation and analysis process.

Eight requirements engineers answered an online form and it allows they follow the step-by-step of the Onto4SASGORE process. First, they have seen the process and sub-process images and answered the following question:

Do you have some comments about Onto4SASGORE Process or Requirements Discovery Sub-process?

We have made improvements in the Onto4SASGORE Process, in the Requirements Discovery Sub-process and in their images.

After, they have used the process activities in a LAS example, modeling the LAS using the Onto4SASGORE ontology concepts and they did several considerations about the specifying activity and the presentation of the process. Lastly, they answer the survey whose results are discussed in next section. We have made several changes and improvements in the Onto4SASGORE process after this study case.

With this case study, we concluded that it is possible to elicit and specify a SAS using the Onto4SASGORE ontology through its process.

After performed the case study, the requirements engineer answered a survey available from April 17 to May 17, 2017. The survey was reasoned on TAM (DAVIS, 1989). Davis (1989) advocates that people tend to use or not use an application whether they believe it will help them perform their job better, this is the perceived usefulness. Even if potential users believe that a given application is useful, they may believe that the system is too hard to use and that the performance benefits of usage are outweighed by the effort of using the application. That is, usage can be influenced by perceived ease of use. Perceived ease of use is the degree to which a person believes that using a particular system would be free of effort. Eight people participated in the case study and the survey. The SAS used in case study was taken from Page, Williams and Boyd (1993). They present a computer-aided ambulance dispatch system. Then, the case study and survey participants used the description of the well-known London Ambulance Service Computer-Aided Dispatch (LAS-CAD).

6.3.1 Results and Discussions

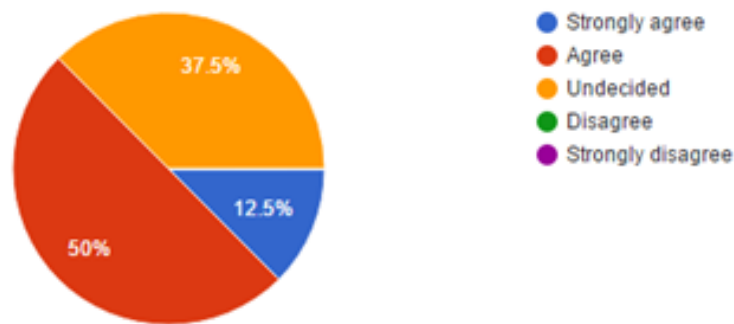
In this sub-section, we present the results and conclusions about the survey. In the general question, we have asked about age, sector of activity, relation with requirements engineering and experience time with requirements engineering. The answers about age vary between 28 and 48 years old. The sectors of activity cited were education, information technology, government and software engineering. The relation with requirements engineering cited were PhD and MsC student, software engineering professor, requirements engineering master, requirements analyst and requirements engineering professor, while the experience time with requirements engineering range between 3 and 20 years. The survey also has five affirmatives about perceived usefulness and five affirmatives about

perceived ease of use. All of them use Likert-scale. We present the results and discussions by affirmative.

About the Onto4SASGORE ontology perceived usefulness

Affirmative 1: The Onto4SASGORE ontology helps in elicitate requirements for SAS domain. Figure 38 shows that 50% of survey participants agree with affirmative 1, 12.5% strongly agree and 37.5% is undecided about the affirmative. With this, we realize that over half (62.5%) of participants agree that Onto4SASGORE ontology helps in elicitate requirements for SAS domain. Although there are a substantial amount of undecided (37.5%), no one disagreed that Onto4SASGORE ontology helps the requirements elicitation for self-adaptive systems.

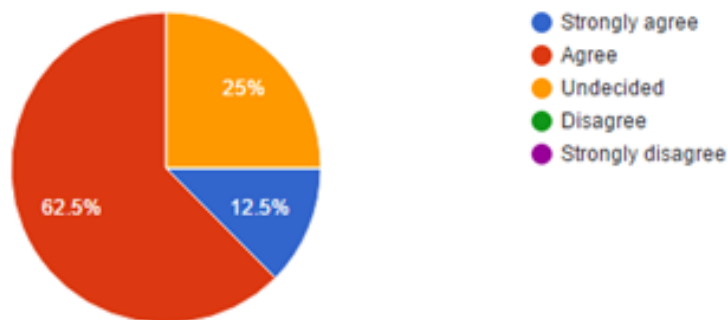
Figure 38 – Graph with answers of affirmative 1.



Source: Own.

Affirmative 2: The Onto4SASGORE is useful to guide the requirements elicitation for SAS. 62.5% agree, 12.5% strongly agree and 25% of the survey participants is undecided about the affirmative 2 as pictured in Figure 39. With this, we realize that a great quantity (75%) of survey participants agree that Onto4SASGORE is useful to guide the requirements elicitation for SAS and no one disagreed with affirmative 2.

Figure 39 – Graph with answers of affirmative 2.

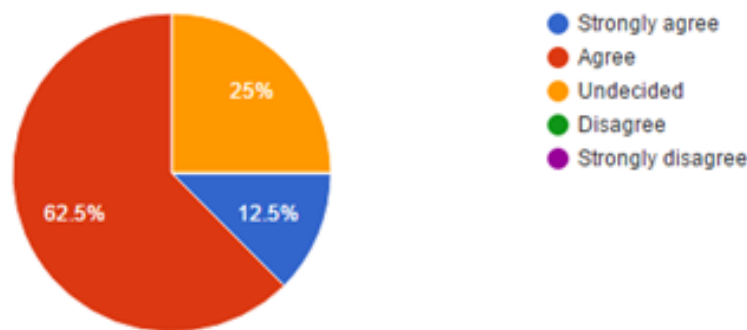


Source: Own.

Affirmative 3: The Onto4SASGORE is useful in specifying requirements for SAS. 62.5% and 12.5% of the survey participants agree and strongly agree (respectively) about

the affirmative 3 and 25% is undecided if the Onto4SASGORE is useful in specifying requirements for SAS, as pictured in Figure 40. With this, we realize that 75 percent of the participants is in accordance that the Onto4SASGORE is useful in specifying requirements for SAS.

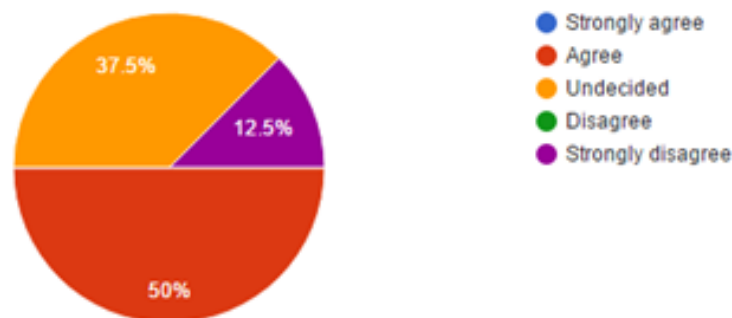
Figure 40 – Graph with answers of affirmative 3.



Source: Own.

Affirmative 4: I have difficulties in eliciting requirements for SAS. Figure 41 shows that 50% of the survey participants agree, 37.5% is undecided and 12.5% strongly disagree with affirmative 4. Thus, we realize that half (50%) of participants has difficulties in eliciting requirements for self-adaptive systems and a great quantity, 37.5% is undecided about this affirmative.

Figure 41 – Graph with answers of affirmative 4.

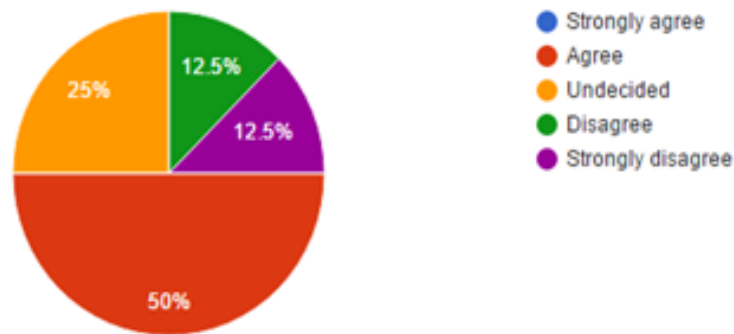


Source: Own.

Affirmative 5: I have difficulties in specifying requirements for SAS. 50% of the participants agree with affirmative 5, as depicted in Figure 42. Although 12.5% disagree and 12.5% strongly disagree about this affirmative, and 25% is undecided, we can realize that half (50%) of survey participants has difficulties in specifying requirements for SAS.

In the end of the form fields about perceived usefulness, we put a comments field for participant comment about the Onto4SASGORE process usefulness. This field was not required, so six of the eight participants have commented. Two participants cited that they have not experience with SAS, and this may have impaired in the execution of the

Figure 42 – Graph with answers of affirmative 5.



Source: Own.

study. One participant wrote that he or she had difficulty understanding the affirmatives, other participant said that he or she believes that the elicitation activity was not used on study, just the specification activity. One participant mentioned that a tool is missing. A last participant had given a suggestion about the application of the study, he or she suggested a simple spreadsheet or document template where each participant could work on thru the questions and to send the file in the end of study.

Half of the participants stated that they had difficulties both to elicit and to specify requirements for the self-adaptive systems domain, and more than half of the participants agreed that Onto4SASGORE ontology helps in eliciting requirements for SAS domain. 75% of participants agreed that Onto4SASGORE is useful to guide the requirements elicitation for SAS and more than half also agreed that the Onto4SASGORE is useful in specifying requirements for SAS. With this, we can conclude that the majority of study participants agree that Onto4SASGORE is useful in eliciting and specifying requirements in the domain of self-adaptive systems.

In Table 34, we show the quantity of answers for each alternative by affirmative related to perceived usefulness. We have excluded the affirmatives 4 and 5, because they are related to difficulties of participants and they are not related to process usefulness. Table 34 columns are labeled according the Likert-scale alternative, for the sake of space, we use the initials of the names, SA for strongly agree, A for agree, U for undecided, D for disagree and SD for strongly disagree. The table also presents the average of the answers by alternative. The higher average is 8, total number of participants. Therefore, this means that average 4 is half of numbers of participants. The higher average of perceived usefulness is on agree with 4.6. Then, we conclude that the Onto4SASGORE process is useful and it can lead benefits for requirements elicitation and specification for SAS.

About the Onto4SASGORE process ease of use

Affirmative 6: The Onto4SASGORE process provides clear and sufficient information for eliciting requirements for SAS. 37.5% of participants agree, 50% is undecided and 12.5% disagree about the affirmative 6, as pictured in Figure 43. The 50% of undecided is

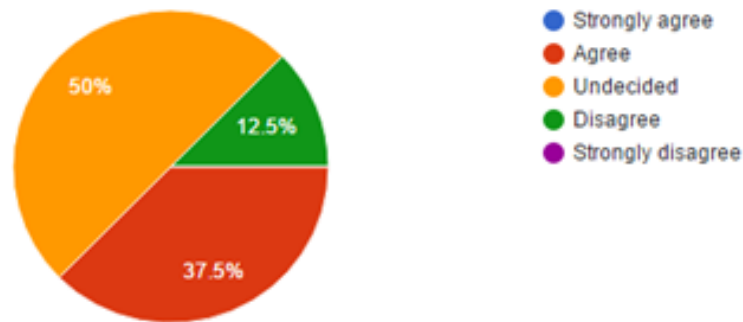
Table 34 – Results for perceived usefulness

Affirmative	SA	A	U	D	SD
1	1	4	3	0	0
2	1	5	2	0	0
3	1	5	2	0	0
Average	1	4.6	2.3	0	0

Source: Own.

worrisome. With this, we realize that 62.5% of the survey participants do not agree that Onto4SASGORE process provides clear and sufficient information for eliciting requirements for SAS.

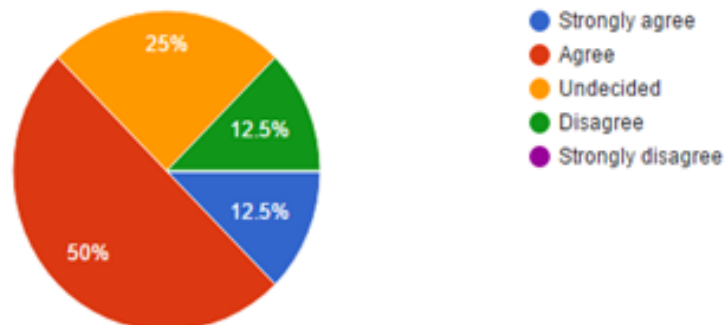
Figure 43 – Graph with answers of affirmative 6.



Source: Own.

Affirmative 7: The Onto4SASGORE process provides clear and sufficient information for specifying requirements for SAS. Figure 44 shows that 50% of the survey participants agree, 12.5% strongly agree, 25% is undecided and 12.5% disagree about the affirmative 7. With this, we realize that 62.5% of the participants agree that the Onto4SASGORE process provides clear and sufficient information for specifying requirements for SAS.

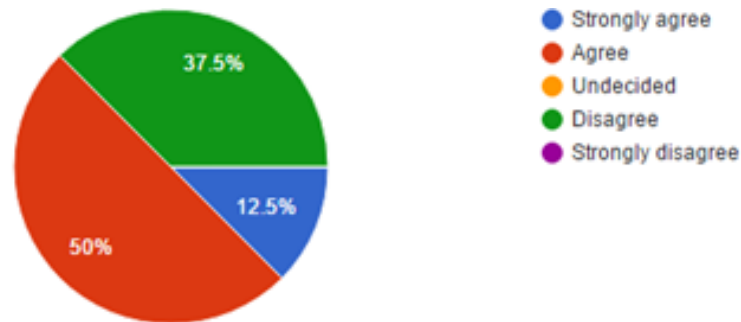
Figure 44 – Graph with answers of affirmative 7.



Source: Own.

Affirmative 8: The description of each activity of the Onto4SASGORE process is clear and easy to understand. In relation to affirmative 8, 50% of survey participants agree, 12.5% strongly agree and 37.5% disagree about the affirmative as pictured in Figure 45. In this affirmative no one was undecided, and although over half (62.5%) of the participants agree that the description of each activity of Onto4SASGORE process is clear and easy to understand, we realize that 37.5% is a considerable percentage.

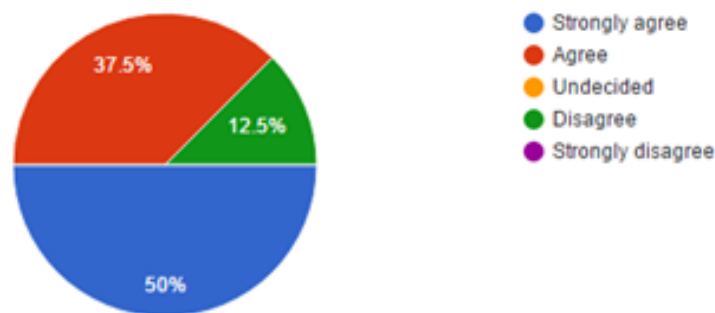
Figure 45 – Graph with answers of affirmative 8.



Source: Own.

Affirmative 9: Training is required to use the Onto4SASGORE process. Figure 46 shows that 37.5% of survey participants agree, 50% strongly agree and 12.5% disagree about the affirmative 9. With this, we realize that 87.5% of the participants agree that training is required to use the Onto4SASGORE process.

Figure 46 – Graph with answers of affirmative 9.

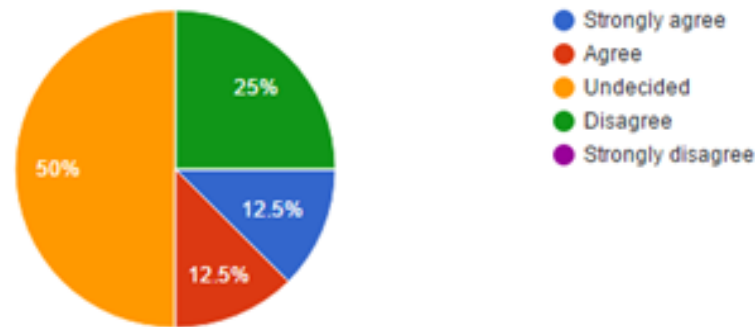


Source: Own.

Affirmative 10: The Onto4SASGORE process is easy to apply. Affirmative 10 deals with the ease of applying the process, 12.5% of the participants agree, 12.5% strongly agree, 50% is undecided and 25% disagree about the affirmative as pictured in Figure 47. Then, we realize that 75% of the participants do not agree that the Onto4SASGORE process is easy to apply.

In the end of the form about perceived ease of use, we put a comments field for the participant to comment about the Onto4SASGORE process ease to use. This field was not required, so just two of the eight participants commented. One participant has suggested

Figure 47 – Graph with answers of affirmative 10.



Source: Own.

a tool, he or she also have commented that would be easier to apply the Onto4SASGORE process with the help of some templates and that some kind of automation is possible in this scenario. The other one said to believe that the eliciting activity was not used in case study and the Onto4SASGORE process was used only to specify the system.

Although more than half of participants agreed that Onto4SASGORE process provides clear and sufficient information for specifying requirements for SAS and they also agree that the description of each activity of the Onto4SASGORE process is clear and easy to understand, we realize that almost all participants (87.5%) agree that training is required to use the Onto4SASGORE process. In addition, we have also seen that 75% of participants do not agree that the process is easy to apply. Then, we conclude that the Onto4SASGORE process is not so easy to use.

In Table 35, we show the quantity of answers for each alternative by affirmative related to ease of use. We have excluded affirmative 9, because it has the opposite direction since needing training means that the process is not easy to use. Table 35 columns are labeled equal to Table 34. Table 35 also present the average of the answers by alternative. The higher average of perceived ease of use agree with 3, less than average that is 4 (even adding 0.75 from SA).

Table 35 – Results for perceived ease of use

Affirmative	SA	A	U	D	SD
6	0	3	4	1	0
7	1	4	2	1	0
8	1	4	0	3	0
10	1	1	4	2	0
Average	0.75	3	2.5	1.75	0

Source: Own.

From the above, we can conclude that the Onto4SASGORE process is not so easy to use and it needs training. Adding the average of undecided participants and those who disagree, we have an average of 4.25, it is higher than half participants. This may have been caused by a failure to present the process, according to comments from participants. The process was submitted online and each activity was presented separately. Another possible reason, also taken from the participants' comments, was the lack of a structured template.

6.4 Accordance

To evaluate the Onto4SASGORE ontology accordance, we performed a survey through an online form with self-adaptive domain specialists. We sent e-mail for authors of the twenty three papers found in the two SLR performed by us and a research group at UFPE which its members work with adaptation controller, so the participants were people who have some experience with the development of self-adaptive systems. Six specialists answered the survey. The Onto4SASGORE was presented by a wiki available at <http://onto4sas.wiki-site.com/> and the specialists answered the online form. The research questions that the accordance criteria intend answer is: *Is the specialist in accordance with the concepts descriptions of Onto4SASGORE?*

6.4.1 Results and Discussions

The specialists have between 22 and 45 years old (22, 29, 31, 32, 36 and 45). Four of them are students, three are PhD student, one is MsC student, one is senior scientist, and one of them is professor. The sector of activities cited by them are: intelligent control system, production research, neural networks, computation view, robotic and research/education (University).

The contact of the students and the professor with self-adaptive systems is through their research. The senior scientist work with research in enterprise interoperability using distributed approaches (ranging from multi agent systems to knowledge management), where Enterprise is a Complex Adaptive System.

The experience time with self-adaptive range from one to twenty years (1, 3, 5, 5, 10 and 20) of experience and just one of the specialists know one ontology for SAS, the ontology by Qureshi, Jureta and Perini (2011).

The survey contains six open questions. We analyze one by one.

Question 01: Did you miss the representation of some hardware feature? What feature(s)?

This question had five responses.

"The limitation of the hardware".

"Sensors and Actors are also resources (e.g. Human Resources for planing). And many Resources have a representation in reality. I missing these links rather than specific/isolated hw features."

"no"

"Controller"

"No. I think the ontology should abstract away from hardware issues and just consider that things get monitored somehow. Is the ontology focused on requirements or architecture? If requirements, then I would strongly suggest you forget anything related to specific technology (such as sensors, for instance)."

To implement these contributions, we put some properties to *Resource*, such as *configuration* and *capacity* to solve the limitation of the hardware. The sensors and actors can be represented by individuals in protégé or instantiated as resource in the ontology template. Lastly, the controller was implemented as feedback loop represented by the MAPE cycle, so the relationships and axioms from the MAPE represent the controller.

Question 02: Did you miss any adaptation information? What information(s)?

This question had four responses.

"Rules of adaptation, Effectiveness of effect"

"Take a look at early works of Holland an the properties identified there wrt. Self-organising feature. More specifically I am missing the concept of 'Self'."

"Link EXECUTE -> MONITOR"

"Your e-mail said that "Time to answer is about 15 minutes". In 15 minutes only a superficial review is possible. That being said, I didn't miss any adaptation information."

The rules of adaptation can be represented by *Goal* (Softgoal, Change, Functional Goal and Non Functional Goal) concept and the effectiveness of effect can be represented by *Overhead* (Insignificant or Failure) concept. How our ontology is to help in the goal-oriented requirements activities for SAS, the *self* concept is represented by the concepts in MAPE cycle (Sensor, Monitor, Analyze, Planner, Execute and Effector). A relationship *updateMonitor* was created to relate the *Execute* and *Monitor*.

Question 03: Did you miss the representation of some context-related information?

What information(s)?

This question had four responses.

"no"

"Actors have sensors to determine the context. Is this possible to specify in the ontology?"

"No"

"Context is a very complex thing to be represented by a single concept in the ontology. I think the ontology should focus on adaptive systems and a separate ontology could be built regarding context-aware systems. They are orthogonal concepts (the former is based on a feedback loop, the latter on a feedforward loop). The two ontologies could be, of course, combined at a later point."

We create a *hasSensor* relationship to link the *SW_Agent* to *Sensor*.

Question 04: Did you miss any concept? What concept(s)?

This question had four responses.

"Controller, System Identifier"

"see above"

"No"

"Your e-mail said that "Time to answer is about 15 minutes". In 15 minutes only a superficial review is possible. That being said, I didn't miss any concept."

The controller can be represented by the MAPE cycle concepts.

Question 05: Do you think the Onto4SASGORE ontology has redundant elements? Which are them?

This question had four responses.

"no"

"nothing obvious (to me)"

"No"

"Plan and Action seem to be redundant. Moreover, you say that a Task is an Action, so the redundancy is on your description, although I think Task and Action are different things."

It was created a Planner concept to differentiate of a plan. So, a Task is a Plan, not an Action.

Question 06: Do you disagree with the description of some concept of the ontology? What concept(s)?

This question had four responses.

"Yes, Context with goal. In the concept of Actor, I think the subkinds are the internal and external agent."

"see above"

"Effector -> Actuator"

"In general the descriptions need to be improved, to make themselves clearer, without incomplete information (e.g. Personal Goal defined as a Goal that expresses some persona feature – what's a persona feature?), redundancy (e.g. Source defined as the source,

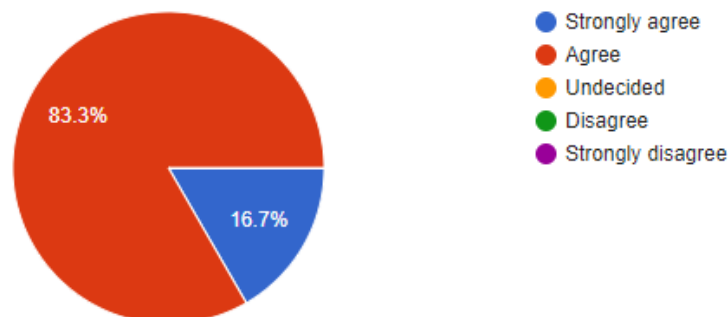
Environment defined as the system plus the environment – how is something itself plus something else?) and contradictions (e.g. Task defined as a plan, but does not specialize Plan). I disagree that Communicated Information is information communicated in the system (according to [1] it's information communicated by stakeholders to requirements engineers). I disagree that Quality Constraint is a specialization of Goal."

The descriptions of the *Context* and *Goal* were improved. The *Personal Goal* was removed. *Quality constraint* was placed as specialization of a non functional goal. To facilitate the use of the software agent concept, we use the *Human* and *SW_Agent* as subkinds of *Actor*.

The survey also had seven affirmatives to be realized by the specialists, the Likert-scale was used to standardize the answer of these affirmatives.

Affirmative 01: The Onto4SASGORE has the main concepts for SAS representation. According to figure 48 all specialists agree with affirmative 01.

Figure 48 – Graph with percentage to affirmative 1.



Source: Own.

Affirmative 02: The Onto4SASGORE ontology helps in finding new requirements.

Figure 49 shows that four of five specialists agree that Onto4SASGORE helps in finding new requirements. Two of them is undecided.

Affirmative 03: The Onto4SASGORE ontology helps in building SAS systems.

Figure 50 shows that 80% of the specialists agree with affirmative 03.

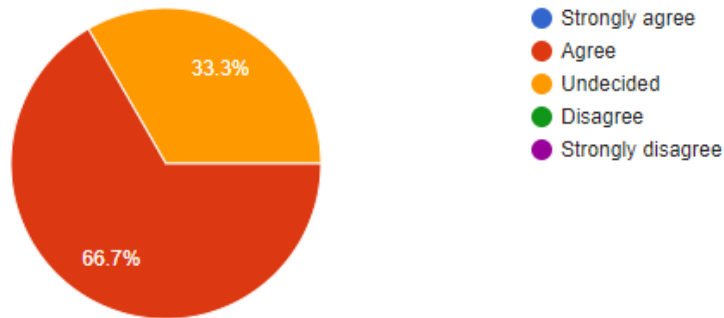
Affirmative 04: It is possible to elicit all the information necessary to specify a SAS by using the concepts presented in this ontology.

All specialists agree that with Onto4SASGORE concepts it is possible to elicit all the information to specify a SAS, according to figure 51

Affirmative 05: The Onto4SASGORE has concepts to capture all information about SAS hardware.

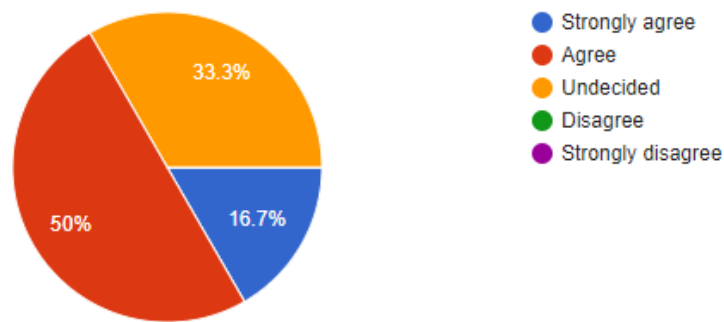
Figure 52 shows that three specialists agree with affirmative 05, two is undecided about it and one of them disagree.

Figure 49 – Graph with percentage to affirmative 2.



Source: Own.

Figure 50 – Graph with percentage to affirmative 3.



Source: Own.

Figure 51 – Graph with percentage to affirmative 4.



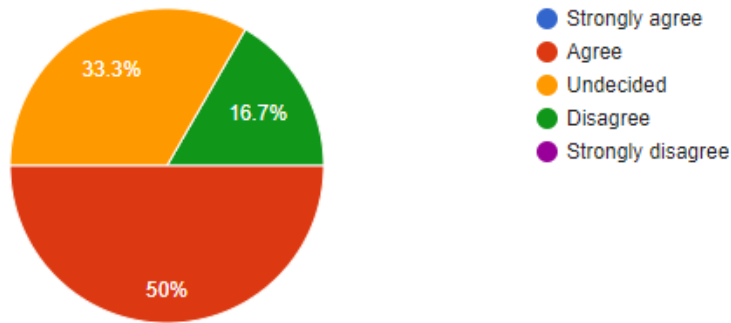
Source: Own.

Affirmative 06: The Onto4SASGORE has concepts to capture all context-related information.

According figure 53, five specialists agree that Onto4SASGORE has concepts to capture all context-related information, and one specialist strongly disagree.

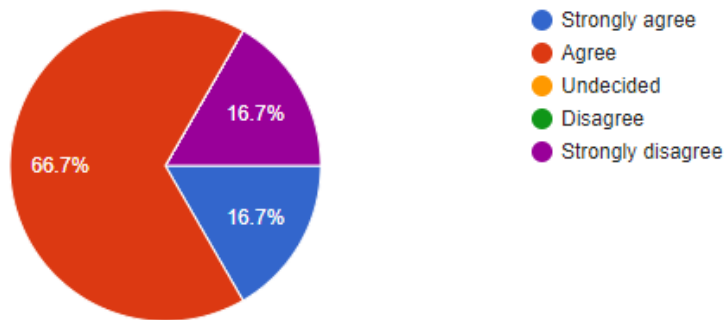
Affirmative 07: The Onto4SASGORE has concepts to capture all information related to adaptation.

Figure 52 – Graph with percentage to affirmative 5.



Source: Own.

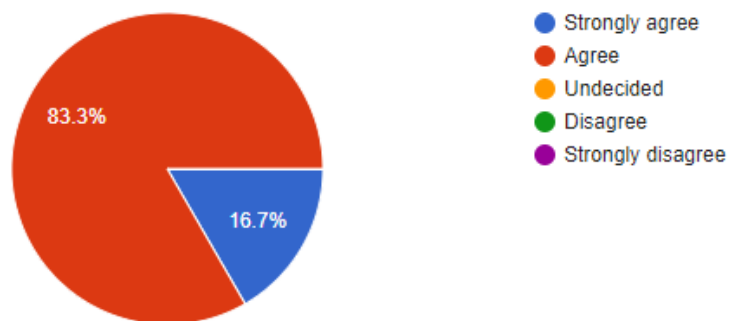
Figure 53 – Graph with percentage to affirmative 6.



Source: Own.

All specialists agree with affirmative 07, according figure 54.

Figure 54 – Graph with percentage to affirmative 7.



Source: Own.

We perceive that affirmatives 02, 03 and 05 let some specialists undecided. Affirmative 02 is about finding new requirements and these specialists did not know the Onto4SASGORE process. We advocate that the Onto4SASGORE process with the elicitation guide and the specification template helps to find new requirements. The affirmative 03 is about building self-adaptive systems, we believe that the knowledge about the axioms

can heal the doubts of the undecided. Lastly, the affirmative 05 is about the concepts to capture all information about SAS hardware and it is related to Question 01, that we solve by adding elements in our ontology. Thus, we claim that the Onto4SASGORE ontology is correct.

The ontology was modified from the specialists finds in open questions. The new version is the presented in Chapter 4 of this thesis.

6.5 Conclusion

In this Chapter, we presented the evaluation of the proposal of this thesis. The contribution of this thesis is an ontology and a process to use it. So, we use six criteria to evaluate the ontology and its process. We evaluated the comprehensiveness, verification, validation, usefulness, ease of use and accordance. We claim that our ontology is embracing, verifiable, valid, according and the process is usable and not so ease to use, considering the amount of people were involved in the experiments and surveys.

7 CONCLUSION

This thesis presents a work to define and create an ontology to aid the GORE requirements elicitation and specification activities when developing self-adaptive systems. The main objective of this thesis was to support the requirements elicitation and specification activities for SAS.

In order to achieve our objective, we analyzed three goal-oriented modeling approaches to specify SAS. It was noticed that these approaches do not have all the concepts necessary to specify the goal-oriented requirements of SAS, then, two SLRs were performed to analyze the works in the literature that deal with knowledge representation, SAS and context. With these works, the Onto4SASGORE, an ontology for goal-oriented requirements elicitation and specification for SAS was created. A process to use the ontology also was created and its main objective is to support requirements engineers to discover GORE requirements during the elicitation activity in a development process of SAS, as well as to specify the elicited requirements.

Following, the specifics objectives defined to facilitate the achievement of the main objective of this thesis:

1. Analyze the existing ontologies related to some characteristics of SAS;
2. Analyze the existing ontologies related to some characteristics of context-awareness;
3. Systematically organize the knowledge of SAS by analyzing some goal-oriented requirements languages proposed to specify self-adaptive systems;
4. Define a core ontology with the main concepts to be considered for the self-adaptive systems domain;
5. Define a domain ontology for self-adaptive systems, with concepts, relationships, axioms and competence questions;
6. Define a process to use the ontology with a GORE elicitation guide and a specification template;
7. Evaluate the ontology and the process.

To achieve the first and second goals two systematic literature reviews were performed to know the works in the literature related to ontologies for self-adaptive systems and context-aware systems. The guidelines proposed by Brereton et al. (2007) were used and the SLRs obtained a total of 23 studies. Then, information from these studies were extracted to answer some research questions. The main objective of this step was to find out

the ontologies proposed in the literature to the self-adaptive systems and context-aware systems domains.

The search string of the systematic literature review about self-adaptive systems returned 2162 papers, of these 13 made part of this research. The search string of the SLR about context-aware systems returned 653 papers, of these 10 made part of this research.

Three approaches proposed for modeling goal-oriented requirements to self-adaptive systems were analyzed. The purpose of this analysis was to understand which elements should be elicited to compose the specification of this type of system. The chosen goal modeling languages for SAS were Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), Design Goal Model (PIMENTEL, 2015) and AdaptiveRML (QURESHI; JURETA; PERINI, 2012). The analysis has shown that not all concepts related to SAS domain can be represented by these languages, such as the modeling dimensions (ANDERSSON et al., 2009) and the concepts of an adaptation cycle such as MAPE (KEPHART; CHESS, 2003) cycle. This analysis was made to achieve the third goal.

To achieve the fourth goal, the core ontology was created and the first version is in Soares et al. (2016).

The results of the two SLRs and the analysis of the three goal modeling languages were essential both to identify the research problem and to create the Onto4SASGORE ontology with concepts related to goal-oriented requirements and self-adaptive systems domains. Three methodologies to create ontologies were used, the methodology by Uschold and Gruninger (1996), the SABiO (FALBO, 2014) and the METHONTOLOGY (FERNANDEZ-LOPEZ; GOMEZ-PEREZ; JURISTO, 1997). In this way the fifth objective was achieved.

To achieve the fifth goal and to make the use of the Onto4SASGORE ontology easier, a process based on the requirements elicitation and analysis process by Sommerville (2010) were defined to guide its use.

To evaluate the Onto4SASGORE ontology and the process, six criteria (comprehensiveness, verification, validity, usefulness, ease of use, and accordance) were used. The results of this evaluation have shown that the proposed ontology is embracing, useful, correct, according as well as it is verified and valid.

7.1 Contributions

This work provided the following contributions:

- Two systematic literature reviews with results of 23 studies about ontologies for SAS and CAS domains. To the best of our knowledge there was no previous work to list the concepts used in self-adaptive systems or context-aware systems. The SLRs can be used in researches about concepts of these systems.

- The analysis of three GORE modeling languages for SAS regarding the coverage of the modeling dimensions (ANDERSSON et al., 2009), MAPE-K (KEPHART; CHESS, 2003) and the core ontology for SAS (QURESHI; JURETA; PERINI, 2012). The languages analyzed were Tropos4AS (MORANDINI; PENSERINI; PERINI, 2008), AdaptiveRML (QURESHI; JURETA; PERINI, 2012) and Design Goal Model (PIMENTEL et al., 2014). The conclusion of the analysis was that the GORE modeling languages do not cover all concepts related to SAS domain. This analysis can be useful for researches about modeling languages for SAS or CAS.
- An ontology to aid a requirements engineer to elicit and specify GORE requirements for SAS. The ontology was created by using the best practices of three methodologies tailored for creating ontologies, to know, SABiO by Falbo (2014), METHONTOL-OGY by Fernandez-Lopez, Gomez-Perez and Juristo (1997) and the methodology by Uschold and Gruninger (1996).
- The ontology was based on the results from the SLRs, the modeling dimensions and the MAPE-K (KEPHART; CHESS, 2003) adaptation cycle. The ontology was created considering the 23 papers selected by the SLRs and the concepts considered by the Monitor, Analyse, Plan and Execute cycle.
- The competence questions were created to facilitate the identification of the concepts and relationships of the ontology and to evaluate the validation criteria by answering the competence questions with a instantiating of a real world system. The competence questions can be used to help in creation of another competence questions about the same or similar domains.
- The axioms were created to allow inferences in the ontology. The axioms were validated in Protégé.
- A process to help the requirements engineer to use the ontology. The process has a guide for elicitation and a template for specification. The process was created based on Sommerville (2010) and it uses the Onto4SASGORE ontology, and a SAS description as input and it produces a GORE requirements specification as output. The Onto4SASGORE process is composed of a sub-process and three activities.
- The evaluation of the Onto4SASGORE through six criteria. The ontology evaluation was performed by a survey with specialists, the creation of individuals in Protégé as well as an illustration example. The evaluation of the process through a survey with specialists in SAS domain. The six criteria are: comprehensiveness, verification, validity, usefulness, ease of use, and accordance. The studies selected in the SLRs were used to evaluate the comprehensiveness criteria. The usefulness and ease of use was evaluated through a case study and a survey with requirements engineers. A

survey also was used to evaluate the accordance criteria, with specialists in adaptive. The verification criteria was evaluated by answering the competence questions of the ontology with its own elements (concepts, relationships and axioms). To evaluate the validity criteria, we instantiate the ontology with a real world system.

The first version of the core ontology presented in thesis was published in Soares et al. (2016).

7.2 Limitations

Some limitations were perceived in the present work. First, some relevant work may have been left out of the SLRs. This can represent a limitation of the Onto4SASGORE because some important concepts could have not been known. Although using the guidelines by Brereton et al. (2007), some relevant paper may have been left out of the selection, this may have happened because of the large amount of paper returned in the string and the manual work of discarding the paper.

Regarding the focus just in GORE modeling languages, may have been constraint the concepts. It has been noticed that many languages are used to model self-adaptive systems, and these languages may have some concept that is not contemplated by the GORE languages, which may happen from some important concept to be left out.

The scope of the survey with specialists can represent a limitation, because few specialists answered the survey. Few specialists answered the survey, so it is difficult to analyze the data to draw any conclusions about the facts. As the present work is quite restrictive, it also creates a restriction on the specialists' profile to answer to the survey.

Through the ease of use criteria of evaluation, we can realize that the Onto4SAGORE is not so easy to use. We can associate with it the fact that the ontology is large. The Protégé and the Menthor tools present problems to support the ontology. Some those tools interface functions did not work in the ontology elements.

Another limitation is the fact that this work do not contemplate the goal oriented requirements prioritization and negotiation, though we know the importance of this activity for a requirements process.

7.3 Future Work

For future work, we intend solve the limitations of this work. We list the intentions.

- To facilitate the use, we plan build a tool to support the ontology use, allowing the specification, through the implementation of the specification template and allowing the reasoning on the specification.

- It is a future work to analyze the modeling languages for SAS that they are not GORE in order to know the elements that theses languages can represent. The SLR by Yang et al. (2014) presents the languages used to model SAS, then this work can be a starting point for analyzing other modeling languages not GORE for SAS.
- It is possible to create a new methodology for creating ontologies for formalizing the best practices of the methodologies used in this thesis. This new methodology should be evaluated in the creation of new ontologies.
- We intend apply the ontology and the process in a specification of others SAS. This thesis presented the application of the Onto4SASGORE ontology in three description of systems, in order to illustrate and evaluate the use of the ontology, but it is necessary to use the ontology in more systems as well as perform a case study with more specialists. Thus, the illustration and evaluation of the Onto4SASGORE get more complete.
- We also intend put effort to research about the goal oriented requirements prioritization to contemplate this activity in the Onto4SASGORE process. The requirements prioritization is an activity very important in a requirements process and it needs to be contemplate in Onto4SASGORE process.

REFERENCES

- ABBURU, S. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, Foundation of Computer Science, v. 57, n. 17, 2012.
- AHMAD, M.; ARAUJO, J.; BELLOIR, N.; BRUEL, J.-M.; GNAHO, C.; LALEAU, R.; SEMMAK, F. Self-adaptive systems requirements modelling: Four related approaches comparison. In: IEEE. *Comparing Requirements Modeling Approaches Workshop (CMA@RE)*, 2013 International. [S.l.], 2013. p. 37–42.
- ANDERSSON, J.; LEMOS, R. D.; MALEK, S.; WEYNS, D. Modeling dimensions of self-adaptive software systems. In: *Software engineering for self-adaptive systems*. [S.l.]: Springer, 2009. p. 27–47.
- BARESI, L.; PASQUALE, L.; SPOLETINI, P. Fuzzy goals for requirements-driven adaptation. In: IEEE. *Requirements Engineering Conference (RE)*, 2010 18th IEEE International. [S.l.], 2010. p. 125–134.
- BRANK, J.; GROBELNIK, M.; MLADENIC, D. A survey of ontology evaluation techniques. In: *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*. [S.l.: s.n.], 2005. p. 166–170.
- BRERETON, P.; KITCHENHAM, B. A.; BUDGEN, D.; TURNER, M.; KHALIL, M. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software*, Elsevier, v. 80, n. 4, p. 571–583, 2007.
- BRESCIANI, P.; PERINI, A.; GIORGINI, P.; GIUNCHIGLIA, F.; MYLOPOULOS, J. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, Springer, v. 8, n. 3, p. 203–236, 2004.
- BRUN, Y.; SERUGENDO, G. D. M.; GACEK, C.; GIESE, H.; KIENLE, H.; LITOIU, M.; MÜLLER, H.; PEZZÈ, M.; SHAW, M. Engineering self-adaptive systems through feedback loops. In: *Software engineering for self-adaptive systems*. [S.l.]: Springer, 2009. p. 48–70.
- CASTAÑEDA, V.; BALLEJOS, L.; CALIUSCO, M. L.; GALLI, M. R. The use of ontologies in requirements engineering. *Global journal of researches in engineering*, v. 10, n. 6, 2010.
- CHENG, B.; SAWYER, P.; BENCOMO, N.; WHITTLE, J. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. *Model Driven Engineering Languages and Systems*, Springer, p. 468–483, 2009.
- CHENG, B. H. C.; LEMOS, R.; GIESE, H.; INVERARDI, P.; MAGEE, J.; ANDERSSON, J.; BECKER, B.; BENCOMO, N.; BRUN, Y.; CUKIC, B.; SERUGENDO, G. M.; DUSTDAR, S.; FINKELSTEIN, A.; GACEK, C.; GEIHS, K.; GRASSI, V.; KARSAI, G.; KIENLE, H. M.; KRAMER, J.; LITOIU, M.; MALEK, S.; MIRANDOLA, R.; MÜLLER, H. A.; PARK, S.; SHAW, M.; TICHY, M.; TIVOLI, M.; WEYNS, D.; WHITTLE, J. Software engineering for self-adaptive systems. In:

_____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. chap. Software Engineering for Self-Adaptive Systems: A Research Roadmap, p. 1–26. ISBN 978-3-642-02161-9. Available at: <http://dx.doi.org/10.1007/978-3-642-02161-9_1>.

COMPUTING, A. et al. An architectural blueprint for autonomic computing. *IBM White Paper*, v. 31, 2006.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, JSTOR, p. 319–340, 1989.

DERMEVAL, D.; VILELA, J.; BITTENCOURT, I. I.; CASTRO, J.; ISOTANI, S.; BRITO, P. A systematic review on the use of ontologies in requirements engineering. In: IEEE. *Software Engineering (SBES), 2014 Brazilian Symposium on*. [S.l.], 2014. p. 1–10.

DERMEVAL, D.; VILELA, J.; BITTENCOURT, I. I.; CASTRO, J.; ISOTANI, S.; BRITO, P.; SILVA, A. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering*, Springer, p. 1–33, 2015.

DUARTE, B. B.; SOUZA, V. E. S.; LEAL, A. L. de C.; GUIZZARDI, R. d. A. F. G.; GUIZZARDI, R. S. Towards an ontology of requirements at runtime. In: IOS PRESS. *Formal Ontology in Information Systems: Proceedings of the 9th International Conference (FOIS 2016)*. [S.l.], 2016. v. 283, p. 255.

FALBO, R. de A. Sabio: Systematic approach for building ontologies. In: *ONTO. COM/ODISE@ FOIS*. [S.l.: s.n.], 2014.

FERNANDEZ-LOPEZ, M.; GOMEZ-PEREZ, A.; JURISTO, N. *Methontology: from ontological art towards ontological engineering*. [S.l.], 1997.

GACEK, C.; GIESE, H.; HADAR, E. Friends or foes?: a conceptual analysis of self-adaptation and its change management. In: ACM. *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*. [S.l.], 2008. p. 121–128.

GOLDSBY, H. J.; SAWYER, P.; BENCOMO, N.; CHENG, B. H.; HUGHES, D. Goal-based modeling of dynamically adaptive system requirements. In: IEEE. *Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the*. [S.l.], 2008. p. 36–45.

GÓMEZ-PÉREZ, A. Evaluation of ontologies. *International Journal of intelligent systems*, Wiley Online Library, v. 16, n. 3, p. 391–409, 2001.

GU, T.; PUNG, H. K.; ZHANG, D. Q. A service-oriented middleware for building context-aware services. *Journal of Network and computer applications*, Elsevier, v. 28, n. 1, p. 1–18, 2005.

GUBER, T. A translational approach to portable ontologies. *Knowledge Acquisition*, v. 5, n. 2, p. 199–229, 1993.

GUIZZARDI, G. *Ontological foundations for structural conceptual models*. [S.l.]: CTIT, Centre for Telematics and Information Technology, 2005.

- GUIZZARDI, G. On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. *Frontiers in artificial intelligence and applications*, IOS Press, v. 155, p. 18, 2007.
- GUIZZARDI, R. S.; LI, F.-L.; BORGIDA, A.; GUIZZARDI, G.; HORKOFF, J.; MYLOPOULOS, J. An ontological interpretation of non-functional requirements. In: *FOIS*. [S.l.: s.n.], 2014. v. 14, p. 344–357.
- HOQUE, M. R.; KABIR, M. H.; THAPA, K.; YANG, S.-H. Ontology-based context modeling to facilitate reasoning in a context-aware system: A case study for the smart home. *International Journal of Smart Home*, v. 9, n. 9, p. 151–156, 2015.
- HORRIDGE, M.; DRUMMOND, N.; GOODWIN, J.; RECTOR, A. L.; STEVENS, R.; WANG, H. The manchester owl syntax. In: *OWLed*. [S.l.: s.n.], 2006. v. 216.
- HU, Y.; LI, X. An ontology based context-aware model for semantic web services. In: IEEE. *Knowledge Acquisition and Modeling, 2009. KAM'09. Second International Symposium on*. [S.l.], 2009. v. 1, p. 426–429.
- JAFFAL, A.; GRAND, B. L.; KIRSCH-PINHEIRO, M. Refinement strategies for correlating context and user behavior in pervasive information systems. *Procedia Computer Science*, Elsevier, v. 52, p. 1040–1046, 2015.
- JEONG-DONG, K.; SON, J.; BAIK, D.-K. Ca5w1h onto: Ontological context-aware model based on 5w1h. *International Journal of Distributed Sensor Networks*, Hindawi Publishing Corporation, 2012.
- JURETA, I. J.; BORGIDA, A.; ERNST, N. A.; MYLOPOULOS, J. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: IEEE. *2010 18th IEEE International Requirements Engineering Conference*. [S.l.], 2010. p. 115–124.
- JURETA, I. J.; BORGIDA, A.; ERNST, N. A.; MYLOPOULOS, J. The requirements problem for adaptive systems. *ACM Transactions on Management Information Systems (TMIS)*, ACM, v. 5, n. 3, p. 17, 2015.
- JURETA, I. J.; 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.
- KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer, IEEE*, v. 36, n. 1, p. 41–50, 2003.
- KEPLER, F. N.; PAZ-TRILLO, C.; RIANI, J.; RIBEIRO, M. M.; DELGADO, K. V.; BARROS, L. N. de; WASSERMANN, R. Classifying ontologies. In: *WONTO*. [S.l.: s.n.], 2006.
- LAI, R.; MAHMOOD, S.; LIU, S. Raap: a requirements analysis and assessment process framework for component-based system. *JSW*, Citeseer, v. 6, n. 6, p. 1050–1066, 2011.
- LAMSWEERDE, A. Kaos tutorial. *Cediti, September*, v. 5, 2003.

- LEMOS, R. D.; GIESE, H.; MÜLLER, H.; SHAW, M.; ANDERSSON, J.; BARESI, L.; BECKER, B.; BENCOMO, N.; BRUN, Y.; CIKIC, B. et al. Software engineering for self-adaptive systems: A second research roadmap. In: SCHLOSS DAGSTUHL-LEIBNIZ-ZENTRUM FUER INFORMATIK. *Dagstuhl Seminar Proceedings*. [S.l.], 2011.
- LEMOS, R. D.; GIESE, H.; MÜLLER, H. A.; SHAW, M.; ANDERSSON, J.; LITOIU, M.; SCHMERL, B.; TAMURA, G.; VILLEGAS, N. M.; VOGEL, T. et al. *Software engineering for self-adaptive systems: A second research roadmap*. [S.l.]: Springer, 2013.
- MACÍAS-ESCRIVÁ, F. D.; HABER, R.; TORO, R. del; HERNANDEZ, V. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications*, Elsevier, v. 40, n. 18, p. 7267–7279, 2013.
- MASOLO, C.; BORGO, S.; GANGEMI, A.; GUARINO, N.; OLTRAMARI, A.; SCHNEIDER, L. Dolce: a descriptive ontology for linguistic and cognitive engineering. *WonderWeb Project, Deliverable D17 v2*, v. 1, p. 75–105, 2003.
- MOORE, P.; HU, B.; WAN, J. Smart-context: A context ontology for pervasive mobile computing. *The Computer Journal*, Oxford University Press, v. 53, n. 2, p. 191–207, 2008.
- MORANDINI, M.; PENSERINI, L.; PERINI, A. Towards goal-oriented development of self-adaptive systems. In: ACM. *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*. [S.l.], 2008. p. 9–16.
- MORANDINI, M.; PERINI, A.; MARCHETTO, A. Empirical evaluation of tropos4as modelling. *iStar*, Citeseer, v. 766, p. 14–19, 2011.
- NUSEIBEH, B.; EASTERBROOK, S. Requirements engineering: a roadmap. In: ACM. *Proceedings of the Conference on the Future of Software Engineering*. [S.l.], 2000. p. 35–46.
- O'CONNOR, M.; DAS, A. Sqwrl: a query language for owl. In: CEUR-WS. *ORG. Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529*. [S.l.], 2009. p. 208–215.
- OLIVEIRA, K.; PIMENTEL, J.; SANTOS, E.; DERMEVAL, D.; GUEDES, G.; SOUZA, C.; SOARES, M.; CASTRO, J.; ALENCAR, F. M.; SILVA, C. 25 years of requirements engineering in brazil: a systematic mapping. In: *WER*. [S.l.: s.n.], 2013.
- OSSADA, J. C.; MARTINS, L. E. G.; RANIERI, B. S.; BELGAMO, A. Gerse: Guia de elicitação de requisitos para sistemas embarcados. 2012.
- PAGE, D.; WILLIAMS, P.; BOYD, D. *Report of the Inquiry Into the London Ambulance Service, February 1993*. [S.l.]: South West Thames Regional Health Authority, 1993.
- PANTSAR-SYVÄNIEMI, S.; PURHONEN, A.; OVASKA, E.; KUUSIJÄRVI, J.; EVESTI, A. Situation-based and self-adaptive applications for the smart environment. *Journal of Ambient Intelligence and Smart Environments*, IOS Press, v. 4, n. 6, p. 491–516, 2012.

- PERRY, D. E.; SIM, S. E.; EASTERBROOK, S. M. Case studies for software engineers. In: IEEE. *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*. [S.l.], 2004. p. 736–738.
- PIMENTEL, J.; CASTRO, J.; MYLOPOULOS, J.; ANGELOPOULOS, K.; SOUZA, V. E. S. From requirements to statecharts via design refinement. In: ACM. *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. [S.l.], 2014. p. 995–1000.
- PIMENTEL, J. H. C. *Systematic design of adaptive systems: control-based framework*. Phd Thesis (PhD Thesis) — Informatics Center, 2015.
- PREUVENEERS, D.; BERGH, J. Van den; WAGELAAR, D.; GEORGES, A.; RIGOLE, P.; CLERCKX, T.; BERBERS, Y.; CONINX, K.; JONCKERS, V.; BOSSCHERE, K. D. Towards an extensible context ontology for ambient intelligence. In: SPRINGER. *European Symposium on Ambient Intelligence*. [S.l.], 2004. p. 148–159.
- QURESHI, N. A.; JURETA, I. J.; PERINI, A. Requirements engineering for self-adaptive systems: Core ontology and problem statement. In: SPRINGER. *Advanced Information Systems Engineering*. [S.l.], 2011. p. 33–47.
- QURESHI, N. A.; JURETA, I. J.; PERINI, A. Towards a requirements modeling language for self-adaptive systems. In: *Requirements Engineering: Foundation for Software Quality*. [S.l.]: Springer, 2012. p. 263–279.
- QURESHI, N. A.; PERINI, A. Engineering adaptive requirements. In: IEEE. *Software Engineering for Adaptive and Self-Managing Systems, 2009. SEAMS'09. ICSE Workshop on*. [S.l.], 2009. p. 126–131.
- RAMIREZ, A. J.; JENSEN, A. C.; CHENG, B. H. A taxonomy of uncertainty for dynamically adaptive systems. In: IEEE. *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*. [S.l.], 2012. p. 99–108.
- SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, ACM, v. 4, n. 2, p. 14, 2009.
- SCHILIT, B.; ADAMS, N.; WANT, R. Context-aware computing applications. In: IEEE. *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*. [S.l.], 1994. p. 85–90.
- SERBEDZIJA, N.; FAIRCLOUGH, S. Reflective pervasive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, ACM, v. 7, n. 1, p. 12, 2012.
- SILVASOUZA, V. E.; MYLOPOULOS, J. Designing an adaptive computer-aided ambulance dispatch system with zanshin: an experience report. *Software: Practice and Experience*, v. 45, n. 5, p. 689–725, 2015. ISSN 1097-024X. Available at: <<http://dx.doi.org/10.1002/spe.2245>>.
- SOARES, M.; VILELA, J.; GUEDES, G.; SILVA, C.; CASTRO, J. New advances in information systems and technologies. In: _____. Cham: Springer International Publishing, 2016. chap. Core Ontology to Aid the Goal Oriented Specification for Self-Adaptive Systems, p. 609–618. ISBN 978-3-319-31232-3. Available at: <http://dx.doi.org/10.1007/978-3-319-31232-3_57>.

- SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2010. ISBN 0137035152, 9780137035151.
- SOUAG, A.; SALINESI, C.; MAZO, R.; COMYN-WATTIAU, I. A security ontology for security requirements elicitation. In: *Engineering Secure Software and Systems*. [S.l.]: Springer, 2015. p. 157–177.
- SOUZA, V. E. S.; LAPOUCHNIAN, A.; ROBINSON, W. N.; MYLOPOULOS, J. Awareness requirements for adaptive systems. In: ACM. *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems*. [S.l.], 2011. p. 60–69.
- STEFANOV, S.; HUANG, V. A semantic web based system for context metadata management. In: SPRINGER. *Research Conference on Metadata and Semantic Research*. [S.l.], 2009. p. 118–129.
- SUDHA, R.; RAJAGOPALAN, M.; SELVANAYAKI, M.; SELVI, S. T. Ubiquitous semantic space: A context-aware and coordination middleware for ubiquitous computing. In: IEEE. *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*. [S.l.], 2007. p. 1–7.
- SWARTOUT, B.; PATIL, R.; KNIGHT, K.; RUSS, T. Toward distributed use of large-scale ontologies. In: *Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*. [S.l.: s.n.], 1996. p. 138–148.
- TRAN, T.; CIMIANO, P.; ANKOLEKAR, A. Rules for an ontology-based approach to adaptation. In: IEEE. *Semantic Media Adaptation and Personalization, 2006. SMAP'06. First International Workshop on*. [S.l.], 2006. p. 49–54.
- USCHOLD, M.; GRUNINGER, M. Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, v. 11, p. 93–136, 6 1996. ISSN 1469-8005. Available at: <http://journals.cambridge.org/article_S0269888900007797>.
- VASSEV, E.; HINCHEY, M. The knowlang approach to self-adaptation. In: *Software, Services, and Systems*. [S.l.]: Springer, 2015. p. 676–692.
- VASSEV, E.; HINCHEY, M. Knowledge representation for adaptive and self-aware systems. In: *Software Engineering for Collective Autonomic Systems*. [S.l.]: Springer, 2015. p. 221–247.
- VIEIRA, V.; TEDESCO, P.; SALGADO, A. C. Towards an ontology for context representation in groupware. In: SPRINGER. *International Conference on Collaboration and Technology*. [S.l.], 2005. p. 367–375.
- WEICHHART, G.; NAUDET, Y. Ontology of enterprise interoperability extended for complex adaptive systems. In: SPRINGER. *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*. [S.l.], 2014. p. 219–228.
- WHITTLE, J.; SAWYER, P.; BENCOMO, N.; CHENG, B. H.; BRUEL, J.-M. Relax: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering*, Springer, v. 15, n. 2, p. 177–196, 2010.

- WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering*. [S.l.]: Springer Science & Business Media, 2012.
- YANG, Z.; LI, Z.; JIN, Z.; CHEN, Y. A systematic literature review of requirements modeling and analysis for self-adaptive systems. In: *Requirements Engineering: Foundation for Software Quality*. [S.l.]: Springer, 2014. p. 55–71.
- YILMAZ, Ö.; ERDUR, R. C. iconawa—an intelligent context-aware system. *Expert Systems with Applications*, Elsevier, v. 39, n. 3, p. 2907–2918, 2012.
- YU, E. Social modeling and i*. In: *Conceptual Modeling: Foundations and Applications*. [S.l.]: Springer, 2009. p. 99–121.
- YUAN, E.; MALEK, S. A taxonomy and survey of self-protecting software systems. In: IEEE PRESS. *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. [S.l.], 2012. p. 109–118.
- ZAVE, P. Classification of research efforts in requirements engineering. *ACM Computing Surveys (CSUR)*, ACM, v. 29, n. 4, p. 315–321, 1997.

APPENDIX A – ONTOLOGY FOR SAS:

SLR PROTOCOL

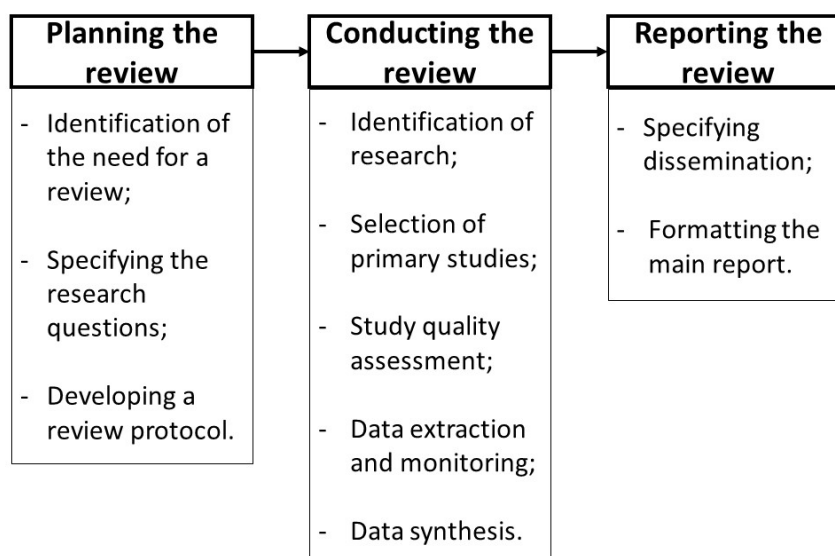
This appendix contains the protocol of the Systematic Literature Review performed in this research, as described in Section 3.1.

A.1 Introduction

The systematic literature review was performed following the guidelines proposed by Brereton et al. (2007). The conduction of a SLR allows identify gaps in the current research in order to suggest new investigations areas.

This SLR investigates the results of the selected studies and the research methods used. The objective is perform a synthesis of literature, identify the questions, and propose solutions for the problems. All selected studies are used and grouped according to their similarities, in order to respond the research questions. The hearing of the results is for the specialized researchers and professionals who are requirements engineers or developers interested in self-adaptive systems. A SLR involves several discrete activities. The stages in a SLR are summarized into three main (BRERETON et al., 2007) as demonstrated in Figure 55. The need of conducting the review was presented in the Chapter 1.

Figure 55 – Systematic Literature Review Phases.



Source: Adapted from Brereton et al. (2007).

A.2 Research Questions

It is intended to investigate and learn about the ontologies approaches used to develop self-adaptive systems over the years. The search starting date was unlimited, whereas the ending date was set to January 2016 (inclusive), attempting to answers the following questions:

RQ01: Which groups and organizations are most active in ontology for SAS research?

RQ02: Which languages are used to describe the ontology?

RQ03: What are the main problems addressed by the ontology?

RQ04: Which requirements engineering activities are considered by the ontology?

RQ05: What application domains are involved?

RQ06: Does the analyzed ontology present just core concepts? (Core ontology is an ontology that presents just a minimal set group of concepts.)

RQ07: Does the ontology have tool support?

RQ08: What are the concepts presented in the ontology?

A.3 Search Process

In the search process phase, we define the search source and the search string. This study is based on automated and manual search. The automated search was done considering six search engines and covering the time period until January 2016. The databases used were: ACM, IEEE, SpringerLink, Science Direct, ISI Web of Science and Scopus.

The search process for this study is also interested in the conferences and events in requirements engineering and self-adaptive systems areas. The manual search was conducted analyzing every issues of major conferences and events in the requirements engineering and self-adaptive systems areas: RE, ICSE, SEAMS, CAiSE, REFSQ, SAC, REJ and JSS. The automated search was performed in the search engines considering the Computer Science area in January 2016. The general string used is presented in Table 36.

Table 36 – Search String.

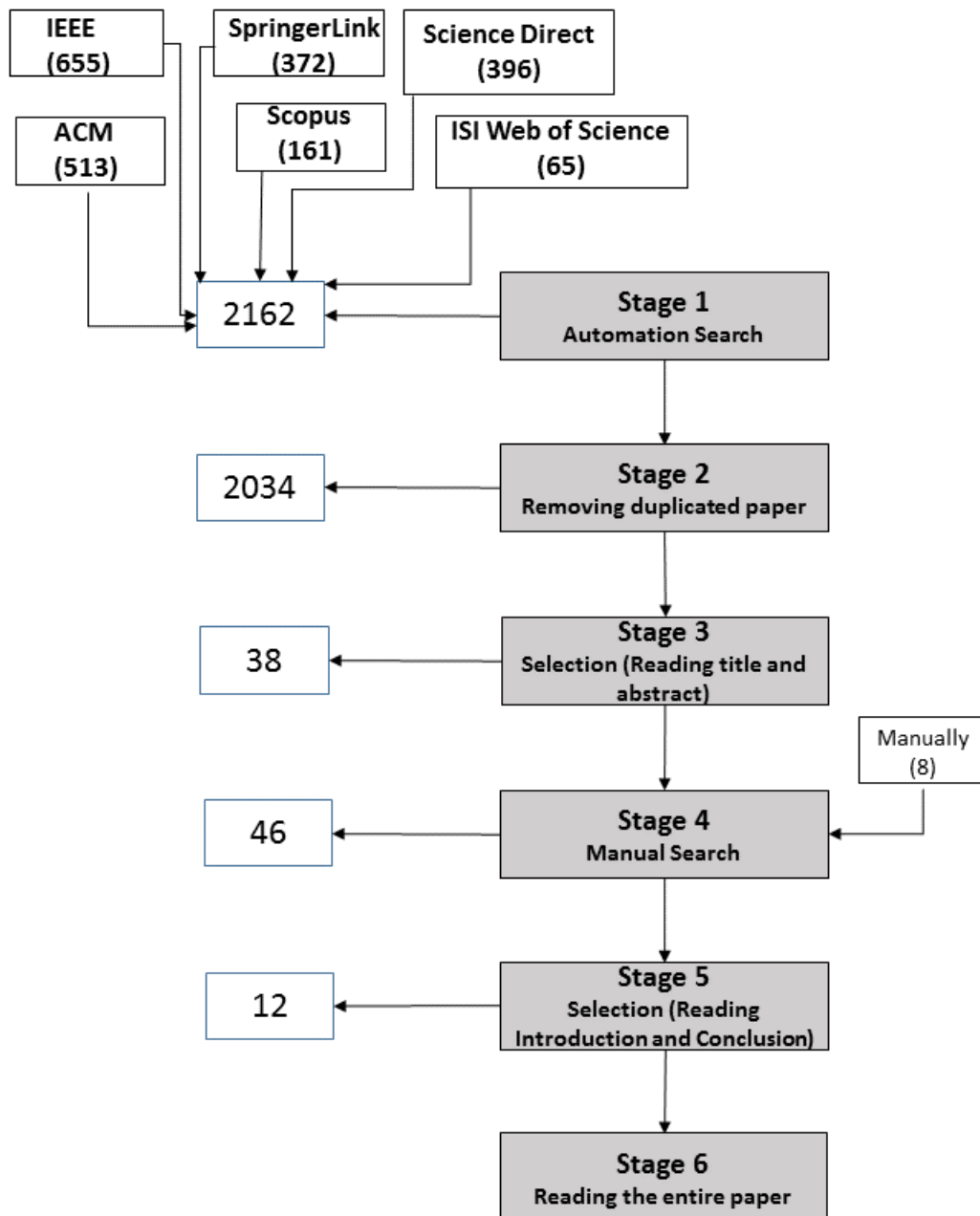
```
(
(ontology OR "knowledge representation" OR ontological) AND
("self-adaptive system" OR "adaptive system" OR "dynamic system")
)
```

Source: Own.

The Figure 56 shows the selection process phases and the quantity of paper returned by search engine.

In the Stage 1 (automation search), we execute the string search in the six engines. It is important to highlight that the string search was adapted to each engine. This stage

Figure 56 – Selection process stages.



Source: Own.

returned 2162 papers. The IEEE returned 655 papers, the SpringerLink returned 372 papers, the Science returned 396, ACM returned 513, Scopus returned 161 and ISI Web of Science returned 65 papers.

In the Stage 2 (removing duplicated paper), we used StArt¹ tool to help removing the duplicated papers. In this stage, 128 papers were classified as duplicated.

In the Stage 3 (selection by reading title and abstract) we performed the selection of

¹ http://lapes.dc.ufscar.br/tools/start_tool

the primary studies, with the support of the StArt tool. In this stage, we read the title and abstract of each paper in order to check the inclusion and exclusion criteria, and 1996 papers were excluded.

The Stage 4 (manual search) started with 38 papers (arising from the Stage 3) and we added 8 papers resulted from the manual research, the RE, ICSE, REJ and CAiSE returned 1 paper each one, the SEAMS returned 4 papers and the SAC, JSS and REFSQ did not return any paper. In this stage, we read the title and abstract of each paper.

The Stage 5 (selection by reading introduction and conclusion) is also a selection stage and we read the introduction and conclusion of the 46 papers in order to check the inclusion and exclusion criteria and realizing not relevant studies for the investigated research questions and exclude them. 34 studies were excluded. In the Stage 6 (reading the entire paper), the remaining 12 papers were read completely and we extract the necessary data and realized the assessment quality of these papers.

A.4 Inclusion and Exclusion Criteria

In the selection 3 and 4 stages, we read the title and abstract of the papers and in the 5 stage, we read the introduction and conclusion and used the inclusion and exclusion criteria. The inclusion and exclusion criteria are presented in Table 37.

Table 37 – Inclusion and Exclusion Criteria.

Inclusion Criteria	Exclusion Criteria
Papers published until January 2016.	It is not clear a proposal of an ontology for self-adaptive systems.
Papers that present an ontology.	Secondary or tertiary paper
Papers proposed for the self-adaptive systems domain.	Gray literature paper
	Not English paper
	Not available paper

Source: Own.

We considered the more complete work when we detected duplicated works and the papers that cannot be accessed from the UFPE's network were obtained by contacting authors.

A.5 Quality Assessment

Some evaluation criteria are defined in the works from Oliveira et al. (2013) and Dermeval et al. (2014) to evaluate the quality of the studies. In this SLR, we use a questionnaire form with nine questions that can be answered with “Yes”, “No” or “Partially” (P). In this

work, the quality of the included studies was evaluated according to the criteria presented in the Table 38.

Table 38 – Quality Assessment.

#	Question	Answer
1	Is the study's goal mentioned clearly? (OLIVEIRA et al., 2013)	Yes = 1; No = 0; P=0.5.
2	Is there a rationale for why the study was undertaken? (DERMEVAL et al., 2014)	Yes = 1; No = 0; P=0.5.
3	Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)? (DERMEVAL et al., 2014)	Yes = 1; No = 0.
4	Is the proposed technique clearly described? (DERMEVAL et al., 2014)	Yes = 1; No = 0; P=0.5.
5	Are the results of the study reported in a clear and unambiguous way? (OLIVEIRA et al., 2013)	Yes = 1; No = 0; P=0.5.
6	Were the results reported based on evidence? (OLIVEIRA et al., 2013)	Yes, they are in the paper = 1; Yes, but they are not in paper = 0.5; No = 0.
7	Are the threats to validity/limitations of the study are discussed? (OLIVEIRA et al., 2013)	Yes = 1; Possible threats are cited, but its effect is not discussed = 0.5; No = 0.
8	Does the research also add value to the industrial community? (DERMEVAL et al., 2014)	Yes = 1; No = 0; P=0.5.
9	Was the study empirically evaluated? (DERMEVAL et al., 2014)	Yes = 1; No = 0.

Source: Own.

The 3 and 9 questions can be answered with "Yes" (the paper meets the criteria) or "No" (the paper does not meet the criteria), the study receive 1 point per question answered with "Yes" and 0 point if it was answered with "No". The 1, 2, 4, 5 and 8 questions add the "Partially" answer (when the paper partially meets the criteria), and the study receives 0.5 point if answered with "Partially". In the 6 question, the study receives 1 point if answered with "Yes, they are in the paper" and it receives 0.5 point when answered with "Yes, but they are not in paper". In the 7 question, the study receives 0.5 point when answered with "Possible threats are cited, but its effect is not discussed".

The quality assessment of the selected studies was performed through the questionnaire. In Table 39, we present the score per question of each study. Only one study (S8)

(7.7% of the total) reached the highest score, that is 9 points, and no study obtained the lowest score, that is 0 point. Just one paper, S4, got a score smaller than 4.5 (medium score), the remaining studies got a score higher than 4.5, which means that almost all (92.3%) studies scored higher than the medium score.

Table 39 – Studies by quality score.

ID	#1	#2	#3	#4	#5	#6	#7	#8	#9	Total
S1	0.5	1	1	0.5	1	1	0.5	1	1	7.5
S2	1	1	1	1	0.5	1	0.5	1	0	7
S3	1	1	1	1	1	1	1	1	0	8
S4	0.5	1	1	0.5	0.5	0.5	0	0	0	4
S5	1	1	1	1	1	1	1	1	0	8
S6	1	1	1	1	1	1	0.5	0	0	6.5
S7	1	1	1	0.5	1	1	0.5	1	1	8
S8	1	1	1	1	1	1	1	1	1	9
S9	0.5	0.5	1	1	0.5	0.5	0.5	1	0	5.5
S10	1	1	1	1	1	1	0.5	0	1	7.5
S11	1	1	1	1	1	0	0	0	0	5
S12	1	1	1	0.5	0.5	1	0	0	0	5
S13	1	1	1	1	0.5	1	0	0	0	5.5

Source: Own.

A.6 Data Extraction

In the data extraction phase, the researchers extracted the following information for each paper:

- Year of publication;
- Authors;
- Which is the organization that the work belongs? (related to RQ01)
- Validation Methods (Case study, simulation, empirical study, survey, experiment, not applicable).
- Type of paper (conference paper, journal paper).
- What is the requirements activities involved in the ontology? (Eliciting Requirements, Modelling and analyzing requirements, Communicating requirements, Agreeing requirements or Evolving requirements according Nuseibeh and Easterbrook (2000)). (related to RQ04)

- Ontology's Language (Ontolingua/KIF, CycL, Loom, Flogic, RDF, SHOE, XOL, OIL, OWL, OML, Lingo, CML, other). (related to RQ02)
- What are the main problems addressed by the ontology? (related to RQ03)
- What application domains are involved? (related to RQ05)
- Is it a core ontology? (Yes; No) (related to RQ06)
- Is there a tool support? (Yes; No) (related to RQ07)

A.7 Threats to Validity

We used the Wohlin et al. (2012) threats classification.

Construct validity: This aspect of validity reflect to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions (WOHLIN et al., 2012). It is related to generalizing the results beyond the study. It concerns establishing correct operational measures for the concepts being studied. The main constructs in this review are the concept of "Ontology" and "Self-Adaptive Systems". To mitigate this threat, we have used for the first, the "Ontology" as term, and for the second, we have used "Self-adaptive System", "Dynamic System", "Self-adaptive Software" and "Adaptive System" as terms. We also realizing a complementary manual search in the SLR in order to search for studies about ontologies at conferences and journals that deal with requirements engineering or self-adaptive systems.

Internal validity is concerned when causal relations are examined. When the researcher is investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor (WOHLIN et al., 2012). The protocol of the review was validate by experienced researchers.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case (WOHLIN et al., 2012). (It concerns establishing the domain to which a study's findings can be generalized. Some studies cannot be captured by this review.) To minimizing this threat, we are integrating manual and automated search at ACM, IEEE, Springer, Science Direct, ISI Web of Science and Scopus.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducted the same study, the result should be the same (WOHLIN et al., 2012).

APPENDIX B – ONTOLOGY FOR CAS: SLR PROTOCOL

This appendix contains the protocol of the Systematic Literature Review performed in this research, as described in Section 3.2.

B.1 Introduction

The systematic literature review was performed following the guidelines proposed by (BRERETON et al., 2007). The conduction of a SLR allows identify gaps in the current research in order to suggest new investigations areas.

B.2 Research Questions

It's intended to investigate and know the Ontologies proposed to Context-aware over the years. The search starting date was unlimited, whereas the ending date was set to May 2016 (inclusive), attempting to answers these questions:

- RQ01: Which are the concepts present on ontology?
- RQ02: Which languages are used to describe the ontology?
- RQ03: What application domains are involved in the paper?

B.2.1 Search Strategy and Selection Criteria

The search process for this study will be based on an automated search of the following digital libraries covering the time period until May 2016. We have used the following databases in the automated search. The number between the parentheses shows the amount of paper returned by the database when the execution of the search string:

- ACM (99)
- IEEE (87)
- Springer (96)
- ISI Web of Science (37)
- Scopus (122)

- Science Direct (212)

The general string used was: (“ontolog*” AND “context-aware system”) Figure 57 shows the selection process phases and the quantity of paper returned by search engine. In the Stage 1 (automation search), we executed the string search in the five engines. It is important to highlight that the string search was adapted to each engine. This stage returned 653 papers. The IEEE returned 87 papers, the SpringerLink returned 96 papers, ACM returned 99, Scopus returned 122, Science Direct returned 212 and ISI Web of Science returned 37 papers.

In the Stage 2 (removing duplicated paper), we used StArt tool to help removing the duplicated papers. In this stage, 33 papers were classified as duplicated. In the Stage 3 (selection by reading title and abstract) we performed the selection of the primary studies, with the support of the StArt tool. In this stage, we read the title and abstract of each paper in order to check the inclusion and exclusion criteria, and 584 papers were excluded.

The inclusion criteria were the following:

- Papers published until May 2016 with presentation of ontology for context with defined research questions, search process, data extraction and data presentation.

And the exclusion criteria were:

- Papers those are not clear to present ontology for context are excluded.
- Secondary and tertiary papers are excluded.
- Gray literature papers are excluded.
- Duplicated works are considered the more complete work.

The Stage 4 (manual inclusion) we include one paper it was found on individual research.

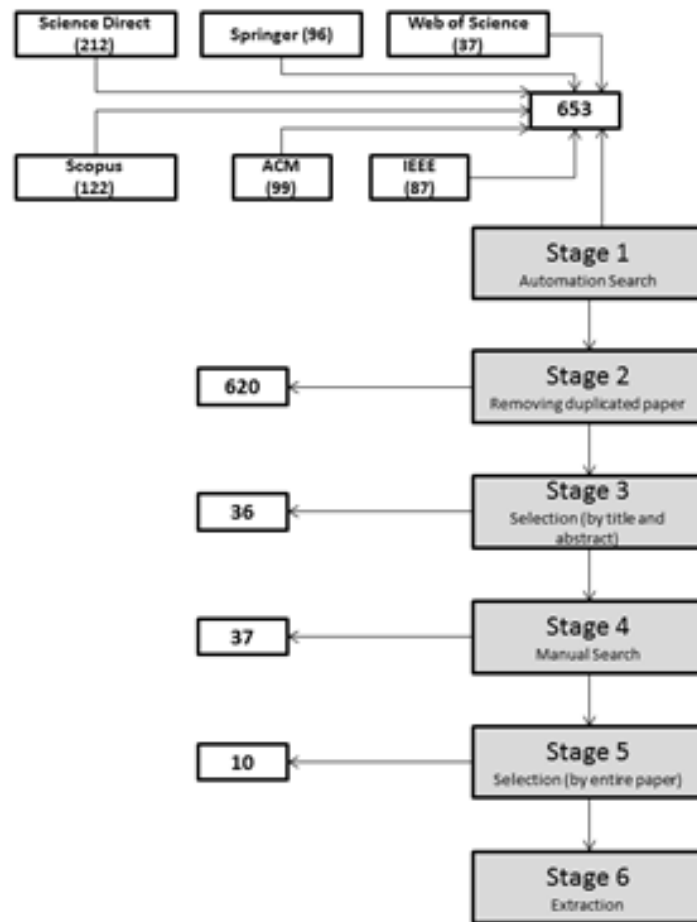
The Stage 5 (selection by reading entire paper) is also a selection stage and we read the 10 entire papers in order to check the inclusion and exclusion criteria and excluding non-relevant studies for the research questions. 27 studies were excluded.

In the Stage 6 (extraction), we extracted the necessary data and realized the quality assessment of the remaining 10 papers.

B.2.2 Quality Assessment

Some evaluation criteria are defined in the Oliveira et al. (2013) and Dermeval et al. (2014) works to evaluate the quality of the studies. In this SLR, the quality of the selected studies will be evaluated according to the following criteria:

Figure 57 – Selection Process Stages.



Source: Own.

1. Is the study's goal mentioned clearly?
Yes = 1; No = 0; P=0.5.
2. The results of the study are reported in a clear and unambiguous?
Yes = 1; No = 0; P=0.5.
3. The results were reported based on evidence?
Yes, they are in the article = 1; Yes, but they are not in Article = 0.5; No = 0.
4. Threats to validity/limitations of the study are discussed?
Yes = 1; Possible threats are cited, but its effect is not discussed = 0.5; No = 0.
5. Is there a rationale for why the study was undertaken?
Yes = 1; No = 0; P = 0.5.
6. Is the paper based on research (or is it merely a "lessons learned" report based on expert opinion)?
Yes = 1; No = 0.

7. Is the proposed technique clearly described?

Yes = 1; No = 0; P = 0.5.

8. Does the research also add value to the industrial community?

Yes = 1; No = 0; P = 0.5.

B.2.3 Data Extraction

The data extraction form present in Table 40 records the information derived from the primary studies. The extraction process is conducted by a PhD student.

Table 40 – Data extraction form.

Nº	Study Data	Description	Relevant RQ
1	Study ID		Study overview
2	Authors, Year, Title		Study overview
3	Validation Methods	Case study, simulation, empirical study, survey, experiment, not applicable.	Study overview
4	Type of study	Ontology for context proposed?	Study overview
5	Concepts	Which are the concepts present on ontology?	RQ01
6	Description Language	Which languages are used to describe the ontology? (Ontolingua/KIF, CycL, Loom, Flogic, RDF, SHOE, XOL, OIL, OWL, OML, Lingo, CML)	RQ02

Source: Own.

APPENDIX C – COMPETENCE QUESTIONS

This Appendix is cited in Section 4.4.1.

ID CQ01

CQ in Natural Language: What are the sensors in the scope of the project?

CQ in Formal Language: `Sensor(?sen)->sqwrl:select(?sen)`

ID CQ02

CQ in Natural Language: What is the sensor status?

CQ in Formal Language: `Sensor(?sen) ^ hasStatus(?sen,?sta)->sqwrl:select(?sta)`

ID CQ03

CQ in Natural Language: What are the effectors in the scope of the project?

CQ in Formal Language: `Effector(?eff)->sqwrl:select(?eff)`

ID CQ04

CQ in Natural Language: What are the effects of the project?

CQ in Formal Language: `Effect(?efe)->sqwrl:select(?efe)`

ID CQ05

CQ in Natural Language: What are the actors of the project?

CQ in Formal Language: `Actor(?act)->sqwrl:select(?act)`

ID CQ06

CQ in Natural Language: What are the roles played by actor?

CQ in Formal Language: `Actor(?act) ^ plays(?act,rol)->sqwrl:select(?rol)`

ID CQ07

CQ in Natural Language: What are the actions held by actor?

CQ in Formal Language: `Actor(?act) ^ performsAction(?act, ?aci) ^ makeSet(?s, ?aci) ^ sqwrl:groupBy(?s, ?act) -> sqwrl:select(?s)`

ID CQ08

CQ in Natural Language: What are the actions on the project?

CQ in Formal Language: `Action(?aci)->sqwrl:select(?aci)`

ID CQ09

CQ in Natural Language: What are the Effects of the action?

CQ in Formal Language: $\text{Action}(\text{?aci}) \wedge \text{hasEffect}(\text{?aci}, \text{?efe}) \rightarrow \text{sqwrl:select}(\text{?efe})$

ID CQ10

CQ in Natural Language: What is the predictability of the effect?

CQ in Formal Language: $\text{Effect}(\text{?efe}) \wedge \text{hasPredictability}(\text{?efe}, \text{?pre}) \rightarrow \text{sqwrl:select}(\text{?pre})$

ID CQ11

CQ in Natural Language: What is the resilience of the effect?

CQ in Formal Language: $\text{Effect}(\text{?efe}) \wedge \text{hasResilience}(\text{?efe}, \text{?rsl}) \rightarrow \text{sqwrl:select}(\text{?rsl})$

ID CQ12

CQ in Natural Language: What is the criticality of the effect?

CQ in Formal Language: $\text{Effect}(\text{?efe}) \wedge \text{hasCriticality}(\text{?efe}, \text{?cri}) \rightarrow \text{sqwrl:select}(\text{?cri})$

ID CQ13

CQ in Natural Language: What is the overhead of the effect?

CQ in Formal Language: $\text{Effect}(\text{?efe}) \wedge \text{hasOverhead}(\text{?efe}, \text{?ovh}) \rightarrow \text{sqwrl:select}(\text{?ovh})$

ID CQ14

CQ in Natural Language: What are the resources of the project?

CQ in Formal Language: $\text{Resource}(\text{?res}) \rightarrow \text{sqwrl:select}(\text{?res})$

ID CQ15

CQ in Natural Language: What are the Plan of the project?

CQ in Formal Language: $\text{Plan}(\text{?pla}) \rightarrow \text{sqwrl:select}(\text{?pla})$

ID CQ16

CQ in Natural Language: What are the Change Plan of the project?

CQ in Formal Language: $\text{ChangePlan}(\text{?chp}) \rightarrow \text{sqwrl:select}(\text{?chp})$

ID CQ17

CQ in Natural Language: What are the Symptoms of the project?

CQ in Formal Language: $\text{Symptom}(\text{?sym}) \rightarrow \text{sqwrl:select}(\text{?sym})$

ID CQ18

CQ in Natural Language: What are the Change Request of the project?

CQ in Formal Language: $\text{ChangeRequest}(\text{?chr}) \rightarrow \text{sqwrl:select}(\text{?chr})$

ID CQ19

CQ in Natural Language: What are the goals of the project?

CQ in Formal Language: $\text{Goal}(\text{?goa}) \rightarrow \text{sqwrl:select}(\text{?goa})$

ID CQ20

CQ in Natural Language: What are the Functional Goals of the project?

CQ in Formal Language: $\text{Functional_Goal}(\text{?fgo}) \rightarrow \text{sqwrl:select}(\text{?fgo})$

ID CQ21

CQ in Natural Language: What are the actions that achieve the functional goal X?

CQ in Formal Language: $\text{Action}(\text{?aci}) \wedge \text{achieves}(\text{?aci}, \text{?fgo}) \rightarrow \text{sqwrl:select}(\text{?aci})$

ID CQ22

CQ in Natural Language: What are the Non Functional Goals of the project?

CQ in Formal Language: $\text{NonFunctional_Goal}(\text{?nfg}) \rightarrow \text{sqwrl:select}(\text{?nfg})$

ID CQ23

CQ in Natural Language: What are the actions that contribute to Non functional goal?

CQ in Formal Language: $\text{Action}(\text{?aci}) \wedge \text{contributes}(\text{?aci}, \text{?nfg}) \rightarrow \text{sqwrl:select}(\text{?aci})$

ID CQ24

CQ in Natural Language: What are the Hardgoals of the project?

CQ in Formal Language: $\text{Hardgoal}(\text{?hgo}) \rightarrow \text{sqwrl:select}(\text{?hgo})$

ID CQ25

CQ in Natural Language: What are the Role Goals of the project?

CQ in Formal Language: $\text{Role_Goal}(\text{?rgo}) \rightarrow \text{sqwrl:select}(\text{?rgo})$

ID CQ26

CQ in Natural Language: What are the Softgoals of the project?

CQ in Formal Language: $\text{Softgoal}(\text{?sgo}) \rightarrow \text{sqwrl:count}(\text{?sgo})$

ID CQ27

CQ in Natural Language: What are the Quality Constraints of the project?

CQ in Formal Language: $\text{Quality_Constraint}(\text{?qct}) \rightarrow \text{sqwrl:count}(\text{?qct})$

ID CQ28

CQ in Natural Language: What is the duration of the goal?

CQ in Formal Language: $\text{Goal}(\text{?goa}) \wedge \text{hasDuration}(\text{?goa}, \text{?gdt}) \rightarrow \text{sqwrl}:\text{select}(\text{?gdt})$

ID CQ29

CQ in Natural Language: What is the flexibility of the goal?

CQ in Formal Language: $\text{Goal}(\text{?goa}) \wedge \text{hasFlexibility}(\text{?goa}, \text{?flx}) \rightarrow \text{sqwrl}:\text{select}(\text{?flx})$

ID CQ30

CQ in Natural Language: What is the dependency of the goal?

CQ in Formal Language: $\text{Goal}(\text{?goa}) \wedge \text{hasDependency}(\text{?goa}, \text{?dpc}) \rightarrow \text{sqwrl}:\text{select}(\text{?dpc})$

ID CQ31

CQ in Natural Language: What is the evolution of the goal?

CQ in Formal Language: $\text{Goal}(\text{?goa}) \wedge \text{hasEvolution}(\text{?goa}, \text{?evl}) \rightarrow \text{sqwrl}:\text{select}(\text{?evl})$

ID CQ32

CQ in Natural Language: What are the inputs of the project?

CQ in Formal Language: $\text{Input}(\text{?inp}) \rightarrow \text{sqwrl}:\text{select}(\text{?inp})$

ID CQ33

CQ in Natural Language: What is the input of the monitor?

CQ in Formal Language: $\text{Monitor}(\text{?mon}) \wedge \text{hasInput}(\text{?mon}, \text{?ipt}) \rightarrow \text{sqwrl}:\text{select}(\text{?ipt})$

ID CQ34

CQ in Natural Language: What is the input of the analyze?

CQ in Formal Language: $\text{Analyze}(\text{?anl}) \wedge \text{hasInput}(\text{?anl}, \text{?ipt}) \rightarrow \text{sqwrl}:\text{select}(\text{?ipt})$

ID CQ35

CQ in Natural Language: What is the input of the plan?

CQ in Formal Language: $\text{Plan}(\text{?pla}) \wedge \text{hasInput}(\text{?pla}, \text{?ipt}) \rightarrow \text{sqwrl}:\text{select}(\text{?ipt})$

ID CQ36

CQ in Natural Language: What is the input of the execute?

CQ in Formal Language: $\text{Execute}(\text{?exe}) \wedge \text{hasInput}(\text{?exe}, \text{?ipt}) \rightarrow \text{sqwrl}:\text{select}(\text{?ipt})$

ID CQ37

CQ in Natural Language: What is the input of the effector?

CQ in Formal Language: $\text{Effector}(\text{?eff}) \wedge \text{hasInput}(\text{?eff}, \text{?ipt}) \rightarrow \text{sqwrl}:\text{select}(\text{?ipt})$

ID CQ38

CQ in Natural Language: What are the outputs of the project?

CQ in Formal Language: $\text{Output}(\text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

ID CQ39

CQ in Natural Language: What is the output of the sensor?

CQ in Formal Language: $\text{Sensor}(\text{?sen}) \wedge \text{hasOutput}(\text{?sen}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

ID CQ40

CQ in Natural Language: What is the output of the monitor?

CQ in Formal Language: $\text{Monitor}(\text{?mon}) \wedge \text{hasOutput}(\text{?mon}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

ID CQ41

CQ in Natural Language: What is the output of the analyze?

CQ in Formal Language: $\text{Analyze}(\text{?anl}) \wedge \text{hasOutput}(\text{?anl}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

ID CQ42

CQ in Natural Language: What is the output of the plan?

CQ in Formal Language: $\text{Plan}(\text{?pla}) \wedge \text{hasOutput}(\text{?pla}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

ID CQ43

CQ in Natural Language: What is the output of the execute?

CQ in Formal Language: $\text{Execute}(\text{?exe}) \wedge \text{hasOutput}(\text{?exe}, \text{?out}) \rightarrow \text{sqwrl:select}(\text{?out})$

ID CQ44

CQ in Natural Language: What are the precondition of the Action X?

CQ in Formal Language: $\text{Action}(\text{?aci}) \wedge \text{hasPrecondition}(\text{?aci}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

ID CQ45

CQ in Natural Language: What is the precondition of the monitor?

CQ in Formal Language: $\text{Monitor}(\text{?mon}) \wedge \text{hasPrecondition}(\text{?mon}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

ID CQ46

CQ in Natural Language: What is the precondition of the analyze?

CQ in Formal Language: $\text{Analyze}(\text{?anl}) \wedge \text{hasPrecondition}(\text{?anl}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

ID CQ47

CQ in Natural Language: What is the precondition of the plan?

CQ in Formal Language: $\text{Plan}(\text{?pla}) \wedge \text{hasPrecondition}(\text{?pla}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

ID CQ48

CQ in Natural Language: What is the precondition of the execute?

CQ in Formal Language: $\text{Execute}(\text{?exe}) \wedge \text{hasPrecondition}(\text{?exe}, \text{?pcd}) \rightarrow \text{sqwrl:select}(\text{?pcd})$

ID CQ49

CQ in Natural Language: What are the changes of the project?

CQ in Formal Language: $\text{Change}(\text{?cha}) \rightarrow \text{sqwrl:select}(\text{?cha})$

ID CQ50

CQ in Natural Language: What is the frequency of the change?

CQ in Formal Language: $\text{Change}(\text{?cha}) \wedge \text{hasFrequency}(\text{?cha}, \text{?fre}) \rightarrow \text{sqwrl:select}(\text{?fre})$

ID CQ51

CQ in Natural Language: What is the type of the change?

CQ in Formal Language: $\text{Change}(\text{?cha}) \wedge \text{hasType}(\text{?cha}, \text{?ctp}) \rightarrow \text{sqwrl:select}(\text{?ctp})$

ID CQ52

CQ in Natural Language: What is the anticipation of the change?

CQ in Formal Language: $\text{Change}(\text{?cha}) \wedge \text{hasAnticipation}(\text{?cha}, \text{?ant}) \rightarrow \text{sqwrl:select}(\text{?ant})$

ID CQ53

CQ in Natural Language: What is the source of the change?

CQ in Formal Language: $\text{Change}(\text{?cha}) \wedge \text{hasSource}(\text{?cha}, \text{?sou}) \rightarrow \text{sqwrl:select}(\text{?sou})$

ID CQ54

CQ in Natural Language: What are the context involved on project?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \rightarrow \text{sqwrl:select}(\text{?ctx})$

ID CQ55

CQ in Natural Language: What is the goal of the context?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \wedge \text{hasGoal}(\text{?ctx}, \text{?goa}) \rightarrow \text{sqwrl:select}(\text{?goa})$

ID CQ56

CQ in Natural Language: What is the location of the context?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \wedge \text{hasLocation}(\text{?ctx}, \text{?lct}) \rightarrow \text{sqwrl:select}(\text{?lct})$

ID CQ57

CQ in Natural Language: What is the time of the context?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \wedge \text{hasTime}(\text{?ctx}, \text{?tme}) \rightarrow \text{sqwrl:select}(\text{?tme})$

ID CQ58

CQ in Natural Language: What is the action of the context?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \wedge \text{hasAction}(\text{?ctx}, \text{?aci}) \rightarrow \text{sqwrl:select}(\text{?aci})$

ID CQ59

CQ in Natural Language: What is the role of the context?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \wedge \text{hasRole}(\text{?ctx}, \text{?rle}) \rightarrow \text{sqwrl:select}(\text{?rle})$

ID CQ60

CQ in Natural Language: What is the status of the context?

CQ in Formal Language: $\text{Context}(\text{?ctx}) \wedge \text{hasStatus}(\text{?ctx}, \text{?stt}) \rightarrow \text{sqwrl:select}(\text{?stt})$

ID CQ61

CQ in Natural Language: What are the mechanism of the project?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \rightarrow \text{sqwrl:select}(\text{?mec})$

ID CQ62

CQ in Natural Language: What is the type of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasMechanismType}(\text{?mec}, \text{?mtp}) \rightarrow \text{sqwrl: select}(\text{?mtp})$

ID CQ63

CQ in Natural Language: What is the timeliness of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasTimeliness}(\text{?mec}, \text{?tln}) \rightarrow \text{sqwrl: select}(\text{?tln})$

ID CQ64

CQ in Natural Language: What is the triggering of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasTriggering}(\text{?mec}, \text{?tgg}) \rightarrow \text{sqwrl: select}(\text{?tgg})$

ID CQ65

CQ in Natural Language: What is the autonomy of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasAutonomy}(\text{?mec}, \text{?atn}) \rightarrow \text{sqwrl: select}(\text{?atn})$

ID CQ66

CQ in Natural Language: What is the organization of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasOrganization}(\text{?mec}, \text{?org}) \rightarrow \text{sqwrl: select}(\text{?org})$

ID CQ67

CQ in Natural Language: What is the scope of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasScope}(\text{?mec}, \text{?scp}) \rightarrow \text{sqwrl:select}(\text{?scp})$

ID CQ68

CQ in Natural Language: What is the duration of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasMechanismDuration}(\text{?mec}, \text{?mdr}) \rightarrow \text{sqwrl:select}(\text{?mdr})$

ID CQ69

CQ in Natural Language: What is the interface of the mechanism?

CQ in Formal Language: $\text{Mechanism}(\text{?mec}) \wedge \text{hasMechanismInterface}(\text{?mec}, \text{?mif}) \rightarrow \text{sqwrl:select}(\text{?mif})$

ID CQ70

CQ in Natural Language: What is the goal of the plan?

CQ in Formal Language: $\text{Plan}(\text{?pla}) \wedge \text{achievesGoal}(\text{?pla}, \text{?goa}) \rightarrow \text{sqwrl:select}(\text{?goa})$

ID CQ71

CQ in Natural Language: How many goals are there on project?

CQ in Formal Language: $\text{Goal}(\text{?goa}) \rightarrow \text{sqwrl:count}(\text{?goa})$

ID CQ72

CQ in Natural Language: How many functional goals are there on project?

CQ in Formal Language: $\text{Functional_Goal}(\text{?fgo}) \rightarrow \text{sqwrl:count}(\text{?fgo})$

ID CQ73

CQ in Natural Language: How many non-functional goals are there on project?

CQ in Formal Language: $\text{NonFunctional_Goal}(\text{?nfg}) \rightarrow \text{sqwrl:count}(\text{?nfg})$

ID CQ74

CQ in Natural Language: How many Hardgoals are there on project?

CQ in Formal Language: $\text{Hardgoal}(\text{?pgo}) \rightarrow \text{sqwrl:count}(\text{?pgo})$

ID CQ75

CQ in Natural Language: How many role goals are there on project?

CQ in Formal Language: $\text{Role_Goal}(\text{?rgo}) \rightarrow \text{sqwrl:count}(\text{?rgo})$

ID CQ76

CQ in Natural Language: How many softgoals are there on project?

CQ in Formal Language: $\text{Softgoal}(\text{?sgo}) \rightarrow \text{sqwrl:count}(\text{?sgo})$

ID CQ77

CQ in Natural Language: How many quality constraints are there on project?

CQ in Formal Language: `Quality_Constraint(?qct)->sqwrl:count(?qct)`

ID CQ78

CQ in Natural Language: How many changes are there on project?

CQ in Formal Language: `Change(?cha)->sqwrl:count(?cha)`

ID CQ79

CQ in Natural Language: How many mechanisms are there on project?

CQ in Formal Language: `Mechanism(?mec)->sqwrl:count(?mec)`

ID CQ80

CQ in Natural Language: How many mechanisms are there in Change X?

CQ in Formal Language: `hasMechanism(ChangeX, ?mec)->sqwrl:select(ChangeX) ^ sqwrl: count (?mec)`

ID CQ81

CQ in Natural Language: What are the mechanisms of Change X?

CQ in Formal Language: `hasMechanism(ChangeX, ?mec)->sqwrl:select(mec)`

ID CQ82

CQ in Natural Language: How many effects are there on project?

CQ in Formal Language: `Effect(?efe)->sqwrl:count(?efe)`

ID CQ83

CQ in Natural Language: How many effects does the Change X have?

CQ in Formal Language: `hasEffect(ChangeX, ?efe)->sqwrl:select ChangeX) ^ sqwrl: count (?efe)`

ID CQ84

CQ in Natural Language: What are the effects produced by Change X?

CQ in Formal Language: `hasEffect(ChangeX, ?efe)->sqwrl:select(?efe)`

ID CQ85

CQ in Natural Language: How many actors are there on project?

CQ in Formal Language: `Actor(?act)->sqwrl:count(?act)`

ID CQ86

CQ in Natural Language: How many contexts are there on project?

CQ in Formal Language: `Context(?ctx)->sqwrl:count(?ctx)`

ID CQ87

CQ in Natural Language: How many resources are there on project?

CQ in Formal Language: `Resource(?res)->sqwrl:count(?res)`

ID CQ88

CQ in Natural Language: How many sensors are there on project?

CQ in Formal Language: `Sensor(?sen)->sqwrl:count(?sen)`

ID CQ89

CQ in Natural Language: How many effectors are there on project?

CQ in Formal Language: `Effector(?efe)->sqwrl:count(?efe)`

ID CQ90

CQ in Natural Language: What are the goals of static evolution?

CQ in Formal Language: `Goal(?goa) ^ hasEvolution(Static)->sqwrl:select(?goa)`

ID CQ91

CQ in Natural Language: What are the goals of dynamic evolution?

CQ in Formal Language: `Goal(?goa) ^ hasEvolution(Dynamic)->sqwrl:select(?goa)`

ID CQ92

CQ in Natural Language: How does the system achieve the goals?

CQ in Formal Language: `Goal(?goa) ^ hasAction(?goa,?aci) ^ makeSet (?s1, ?goa) ^ makeSet (?s1,?aci) ^ sqwrl:union(?s3, ?s1, ?s2) ^ sqwrl:groupBy(?s3,?goa)->sqwrl:select (?s3)`

ID CQ93

CQ in Natural Language: How does the system achieve the goal X?

CQ in Formal Language: `Goal(goalX) ^ hasAction(goalX,?aci) ^ makeSet(?s2, ?aci) ^ sqwrl:groupBy(?s2, ?aci)->sqwrl:select(?s2)`

APPENDIX D – ONTO4SASGORE PROCESS WITHOUT PROTÉGÉ

This Appendix is cited in Section 5.

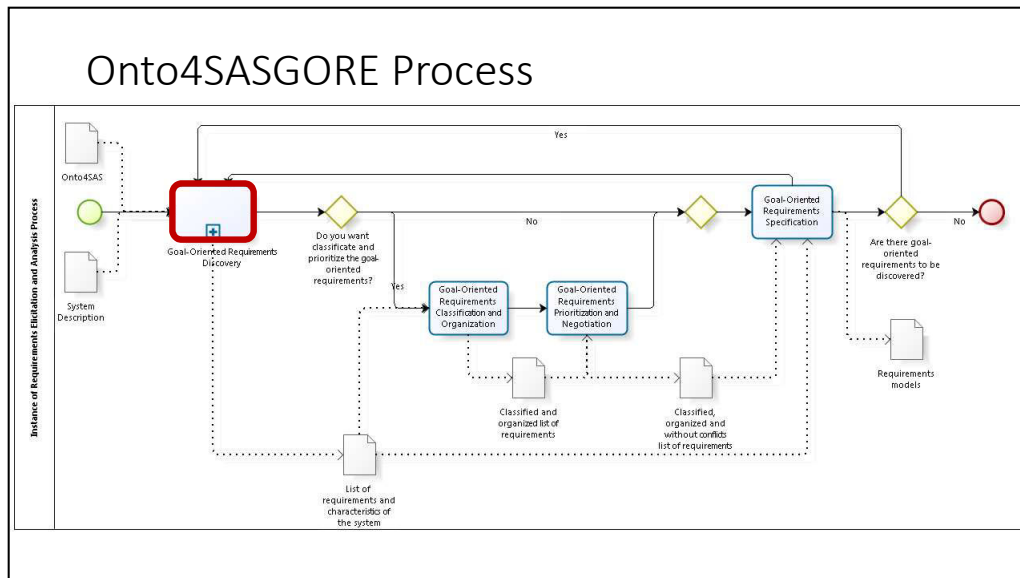
03/07/2017

Onto4SASGORE Process

Purpose

- The purpose of this document is to guide a requirements engineer in the goal-oriented requirements elicitation and specification for self-adaptive systems through the Onto4SASGORE process, without the use of Protégé as tool.
- The requirements engineer should answer all (twenty) questions of the Goal-Oriented Requirements Discovery Sub-Process, using the Onto4SASGORE ontology as glossary and a description of a SAS. The Onto4SASGORE is on <http://onto4sas.wiki-site.com>.
- With the answers of the questions, the requirements engineer should use the GORE specification template to specify the GORE requirements.

03/07/2017

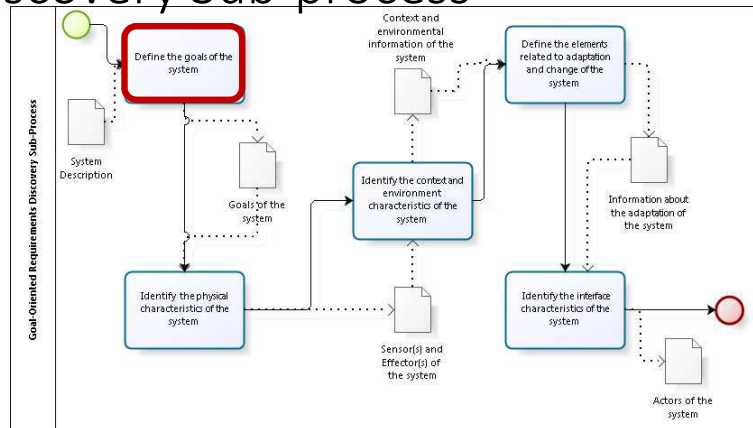


Goal-Oriented Requirements Discovery Sub-process

- It has 5 activities. Each activity has:
 - a name;
 - an objective;
 - a description;
 - expected results;
 - a set of questions;
 - concepts and competence questions of Onto4SASGORE.

03/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 1:** Define the goals of the system
- **Objective:** Finding the goals of the system.
- **Description:** The engineer should answer a set of questions to the system description based on some concepts of Onto4SASGORE and some competence questions, in order to discovery all goals of the system.
- **Expected results:** The goals of the system.
- **Questions:**
 1. What is the main goal of the system?
 2. What are the secondary goals of the system?
 3. What are the characteristics of each goal?
 4. What are the functional goals of the system?
 5. What are the non-functional goals of the system?
 6. What are the constraints of the system?

03/07/2017

- Goals
 - Communicated Information –
 - Domain Assumption - *<instance>*
 - Evaluation - *<instance>*
 - Goal – *<instance>* (evolution, flexibility, duration, multiplicity and dependency)
 - Functional Goal - *<instance>*
 - Non-functional Goal - *<instance>*
 - Quality Constraint - *<instance>*
 - Hardgoal - *<instance>*
 - Softgoal - *<instance>*

Goal-Oriented Requirements Discovery Sub-process

- **Activity 1:** Define the goals of the system

Concepts

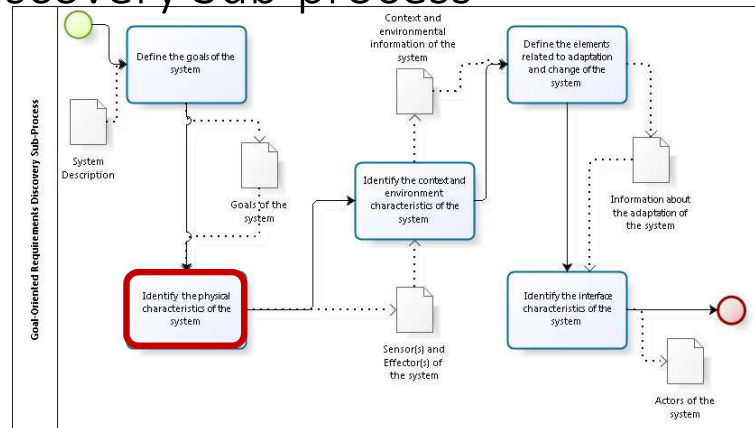
Effects, Action, Resource, Plan, Goal, Functional Goal, Hardgoal, Duration, Flexibility, Dependency and Evolution, Non-Functional Goal, Action and Quality Constraint

Competence Questions

CQ04, CQ08, CQ09, CQ14, CQ15, CQ19, CQ20, CQ21, CQ22, CQ23, CQ24, CQ25, CQ26, CQ27, CQ28, CQ29, CQ30, CQ31

03/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 2:** Identify the physical characteristics of the system
- **Objective:** Finding the sensors and effectors of the system.
- **Description:** Some systems can have the sensor or effector actuating by software or device not physical as well. Therefore, this activity involves mechanism or resource concepts. The requirements engineer should ask the set of questions to the description system and answer based on the Onto4SASGORE concepts.
- **Expected results:** The identification of the sensor and the effector of the system.
- **Questions:**
 1. What is the sensor of the system?
 2. What is the effector of the system?
 3. What are the resources used by system?

03/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 2:** Identify the physical characteristics of the system

Concepts

Sensor, Effector and Resource

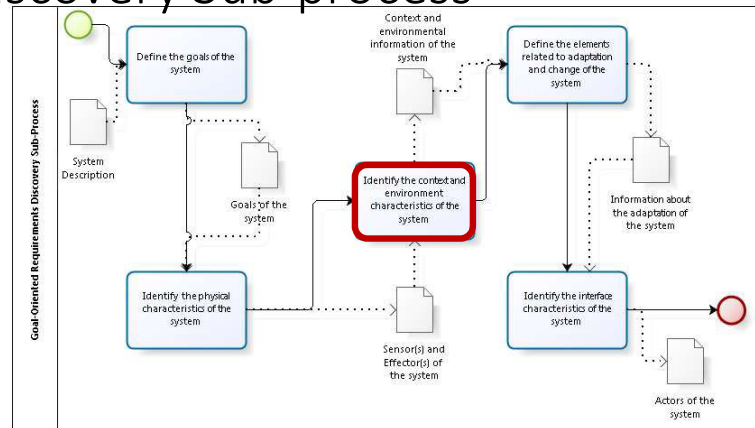
Competence Questions

CQ01, CQ02, CQ03, CQ37, CQ39

- Interface and Communication
 - Sensor – *<instance>*
 - Effector - *<instance>*
 - Resource - *<instance>*

03/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 3:** Identify the context and environment characteristics of the system
- **Objective:** Finding the context and environmental information about the system.
- **Description:** With the Onto4SASGORE ontology and the description of the system, the requirements engineer will answer the set of questions of this activity.
- **Expected results:** The context and environmental information about the system.
- **Questions:**
 1. What are the contexts considered by the system?
 2. What are the users of the system?
 3. What is the environment where the system is operating?

03/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 3:** Identify the context and environment characteristics of the system

Concepts

Context, Goal, Location, Time, Action, Role,
Status, Actor and Environment

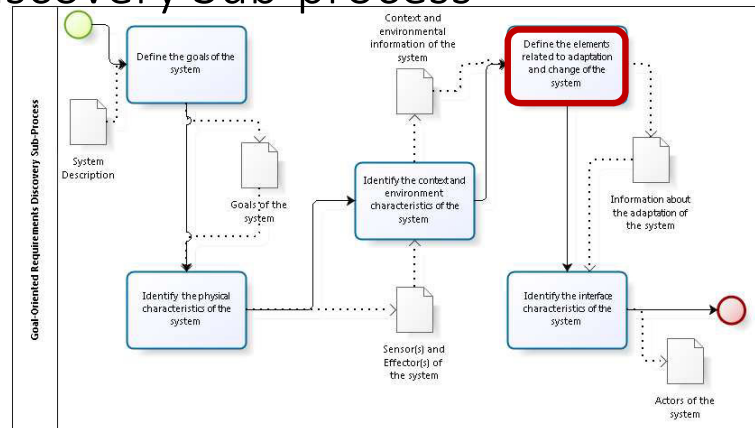
Competence Questions

CQ54, CQ55, CQ56, CQ57, CQ58, CQ59,
CQ60

- Context and environment
 - Environment - *<instance>*
 - Behavior - *<instance>*
 - Context – *<instance>*
 - Location - *<instance>*
 - Status - *<instance>*
 - Time - *<instance>*
- Interface and Communication
 - Actor – *<instance>*
 - Human - *<instance>*
 - SW_Agent - *<instance>*
 - Role - *<instance>*

03/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 4:** Define the requirements related to adaptation and change of the system.
- **Objective:** Finding all goal-oriented requirements related to adaptation and change of the system.
- **Description:** To aim the objective, the requirements engineer will answer the questions of this activity for the system description with the aid of the Onto4SAS ontology. This activity also involves goal, mechanism and effect concepts.
- **Expected results:** The information about the adaptation of the system.
- **Questions:**
 1. What are the actions of the system?
 2. What are the effects by action?
 3. What are the changes of the system?
 4. What is the cause of the change?
 5. What is the mechanism used to perform a change?

03/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 4:** Define the requirements related to adaptation and change of the system.

Concepts

Goal, Effects, Action, Predictability, Resilience, Criticality, Overhead, Change Plan, Symptom, Change Request, Goal, Duration, Flexibility, Dependency, Evolution, Change, Frequency, Change Type, Anticipation, Source, Mechanism, MechanismType, Timeliness, Triggering, Autonomy, Organization, Scope, Duration and Interface Mechanism

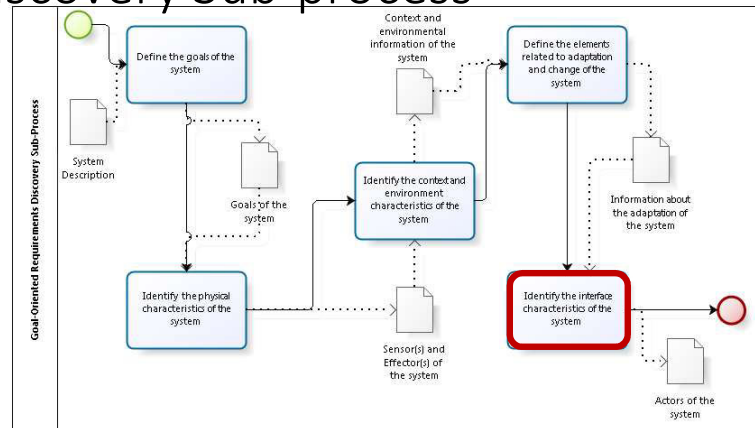
Competence Questions

CQ08, CQ09, CQ10, CQ11, CQ12, CQ13, CQ14, CQ15, CQ16, CQ17, CQ18, CQ28, CQ29, CQ30, CQ31, CQ32, CQ33, CQ34, CQ35, CQ36, CQ37, CQ38, CQ39, CQ40, CQ41, CQ42, CQ43, CQ44, CQ45, CQ46, CQ47, CQ48, CQ49, CQ50, CQ51, CQ52, CQ53, CQ61, CQ62, CQ63, CQ64, CQ65, CQ66, CQ67, CQ68, CQ69, CQ70

- Change and adaptation
 - Action – (Precondition, input and output)
 - Analyze – <instance>
 - Execute – <instance>
 - Monitor – <instance>
 - Plan – <instance>
 - Change – <instance> (source, type, frequency and anticipation)
 - Effect - <instance> (criticality, predictability, overhead and resilience)
 - Mechanism - <instance> (type, autonomy, organization, scope, duration, timeliness and triggering)
 - Resource - <instance>
 - Change Request - <instance>
 - Plan – <instance>
 - Change Plan - <instance>
 - Symptom – <instance>

03/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 5:** Identify the interface characteristics of the system.
- **Objective:** Identifying who or what the system communicates and interacts and the communication interface to realize it.
- **Description:** The requirements engineering will identify all the actors of the system present on system description through the Onto4SASGORE. He or she will answer the questions for this activity using the Onto4SASGORE concepts and competence questions.
- **Expected results:** Find the actors of the system.
- **Questions:**
 1. Who are the actors of the system?
 2. What is the input of each action of the system?
 3. What is the output of each action of the system?

03/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 5:** Identify the interface characteristics of the system.

Concepts

Actor, Role, Input, Monitor, Analyze, Plan, Execute, Effector, Output, Mechanism and Sensor

Competence Questions

CQ05, CQ06, CQ07, CQ32, CQ33, CQ34, CQ35, CQ36, CQ37, CQ38, CQ39, CQ40, CQ41, CQ42, CQ43

- Interface and Communication
 - Actor – <instance>
 - Human - <instance>
 - SW_Agent - <instance>
 - Role - <instance>
- Change and adaptation
 - Action – <instance> (Precondition, input and output)
 - Analyze – <instance>
 - Execute – <instance>
 - Monitor – <instance>
 - Plan – <instance>

03/07/2017

Questionnaire with 20 questions

1. What is the main goal of the system?
<Goal>, <Functional_Goal>, <Hardgoal>, <NonFunctional_Goal>, <Softgoal>,<QualityConstraint>.
2. What are the secondary goals of the system?
<Goal>, <Functional_Goal>, <Hardgoal>, <NonFunctional_Goal>, <Softgoal>,<QualityConstraint>.
3. What are the characteristics of each goal?
<Duration>, <Flexibility>, <Dependency>, <Evolution>.
4. What are the functional goals of the system?
<Functional_Goal>, <Hardgoal>, <Softgoal>.
5. What are the non-functional goals of the system?
<NonFunctional_Goal>, <QualityConstraint>

Questionnaire with 20 questions

6. What are the constraints of the system?
<QualityConstraint>.
7. What is the sensor of the system?
<Sensor>.
8. What is the effector of the system?
<Effector>.
9. What are the resources used by system?
<Resource>.
10. What are the contexts considered by the system?
<Context>, <Goal>, <Location>, <Time>, <Action>, <Role>, <Status>, <Actor>

03/07/2017

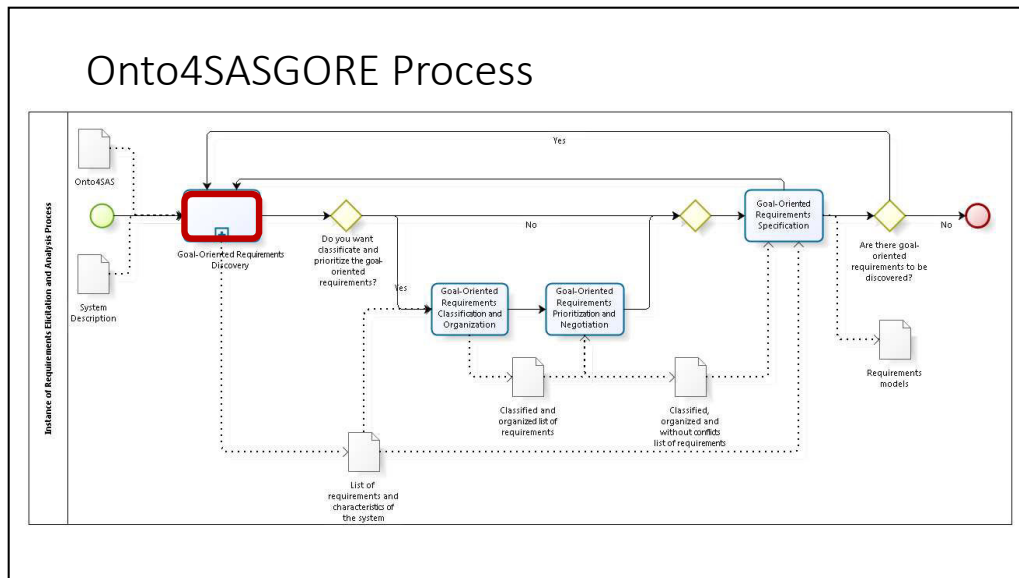
Questionnaire with 20 questions

11. What are the users of the system?
<Actor>.
12. What is the environment where the system is operating?
<Environment>.
13. What are the actions of the system?
<Action>, <Input>, <Output>.
14. What are the effects by action?
<Effects>, <Predictability>, <Resilience>, <Criticality>, <Overhead>.
15. What are the changes of the system?
<Goal>, <Effects>, <Action>, <Input>, <Output>, <Predictability>, <Resilience>, <Criticality>, <Overhead>, <ChangePlan>, <Symptom>, <Change Request>, <Goal>, <Duration>, <Flexibility>, <Dependency>, <Evolution>, <Change>, <Frequency>, <ChangeType>, <Anticipation>, <Source>, <Mechanism>, <MechanismType>, <Timeliness>, <Triggering>, <Autonomy>, <Organization>, <Scope>, <Duration>, <Interface Mechanism>

Questionnaire with 20 questions

16. What is the cause of the change?
<Change>, <Source>.
17. What is the mechanism used to perform a change?
<Mechanism>, <MechanismType>, <Timeliness>, <Triggering>, <Autonomy>, <Organization>, <Scope>, <Duration>, <Interface Mechanism>.
18. Who are the actors of the system?
<Actor>, <Role>, <Input>, <Monitor>, <Analyze>, <Plan>, <Execute>, <Effector>, <Output>, <Mechanism>, <Sensor>.
19. What is the input of each action of the system?
<Action>, <Input>.
20. What is the output of each action of the system?
<Action>, <Output>.

03/07/2017



Goal-Oriented Requirements Specification

• Template – Part 1 (Goal)

- Communicated Information –
 - Domain Assumption,
 - Evaluation,
 - Goal – (evolution, flexibility, duration, multiplicity and dependency)
 - Functional Goal -
 - Non-functional Goal -
 - Quality Constraint -
 - Hardgoal -
 - Softgoal -

03/07/2017

Goal-Oriented Requirements Specification

- Template – Part 2 (**Interface and Communication**)

- Actor –
 - Human -
 - SW_Agent -
- Role -
- Sensor –
- Effector -
- Touchpoint -

Requirements Specification

- Template – Part 3 (**Context and Environment**)

- Environment -
- Behavior -
- Context –
 - Location -
 - Status -
 - Time -
 - Role -

03/07/2017

Requirements Specification

• Template – Part 4 (**Change and Adaptation**)

- Action – (Precondition, input and output)
 - Analyze –
 - Execute –
 - Monitor –
 - Plan –
- Change –
(source, type, frequency and anticipation)
- Effect -
(criticality, predictability, overhead and resilience)
- Mechanism -
(type, autonomy, organization, scope, duration, timeliness and triggering)
- Resource -
 - Change Request -
 - Plan –
 - Change Plan -
 - Symptom –
- Goal – (evolution, flexibility, duration, multiplicity and dependency)
 - Functional Goal -
 - Non-functional Goal -
 - Quality Constraint -
 - Hardgoal -
 - Softgoal -

Finishing

- Once this is done, the requirements engineer can choose a goal-oriented requirements modeling language and model the self-adaptive system with the data found and specified through the Onto4SASGORE process.

APPENDIX E – ONTO4SASGORE PROCESS WITH PROTÉGÉ

This Appendix is cited in Section 5 and in Appendix ??.

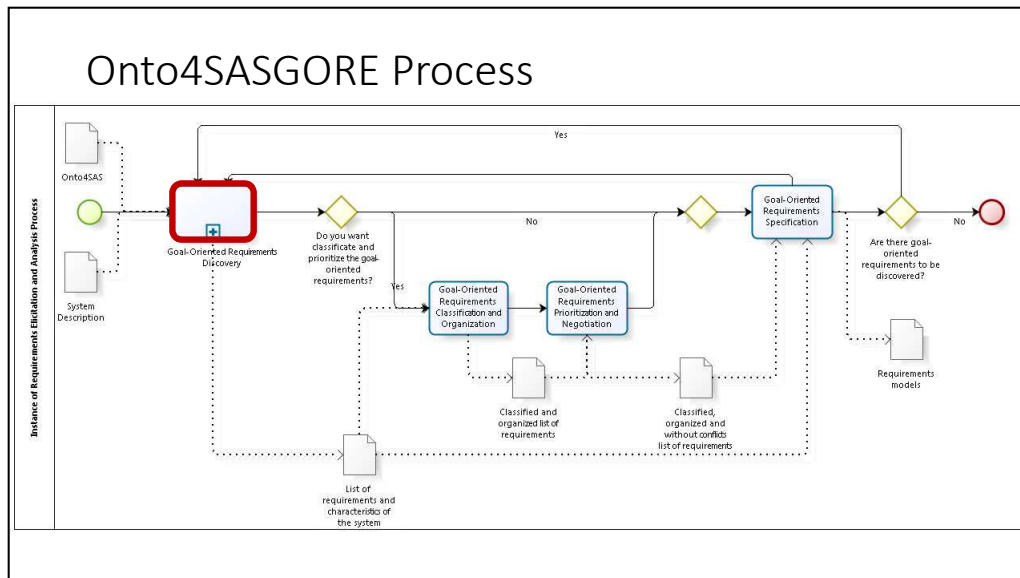
06/07/2017

Onto4SASGORE Process with Protege

Purpose

- The purpose of this document is to guide a requirements engineer in the goal-oriented requirements elicitation and specification for self-adaptive systems through the Onto4SASGORE process, with the use of Protégé as tool. In Protégé a concept is a class and a relationship is a property.
- The requirements engineer should answer all (twenty) questions of the Goal-Oriented Requirements Discovery Sub-Process, using the Onto4SASGORE ontology as glossary and a description of a SAS. The Onto4SASGORE is on <http://onto4sas.wiki-site.com>.
- With the answers of the questions, the requirements engineer should use the GORE specification template to specify the GORE requirements.

06/07/2017

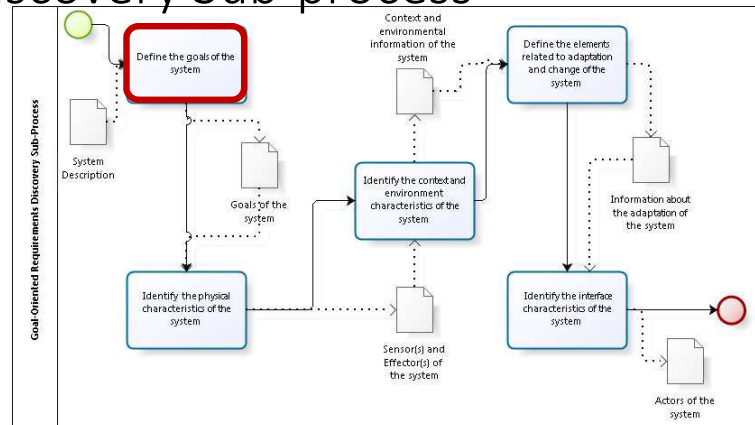


Goal-Oriented Requirements Discovery Sub-process

- It has 5 activities. Each activity has:
 - a name;
 - an objective;
 - a description;
 - expected results;
 - a set of questions;
 - concepts and competence questions of Onto4SASGORE.

06/07/2017

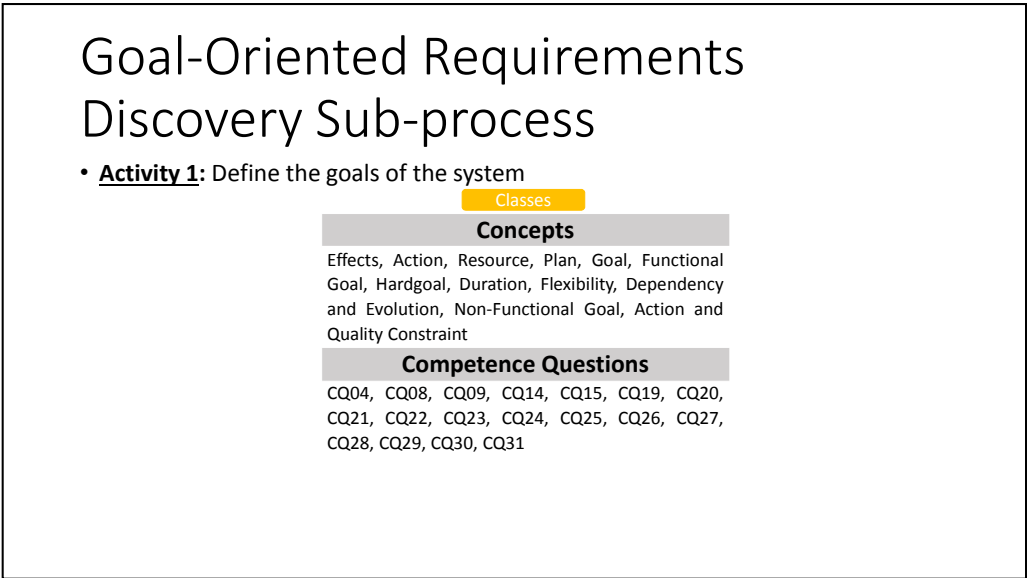
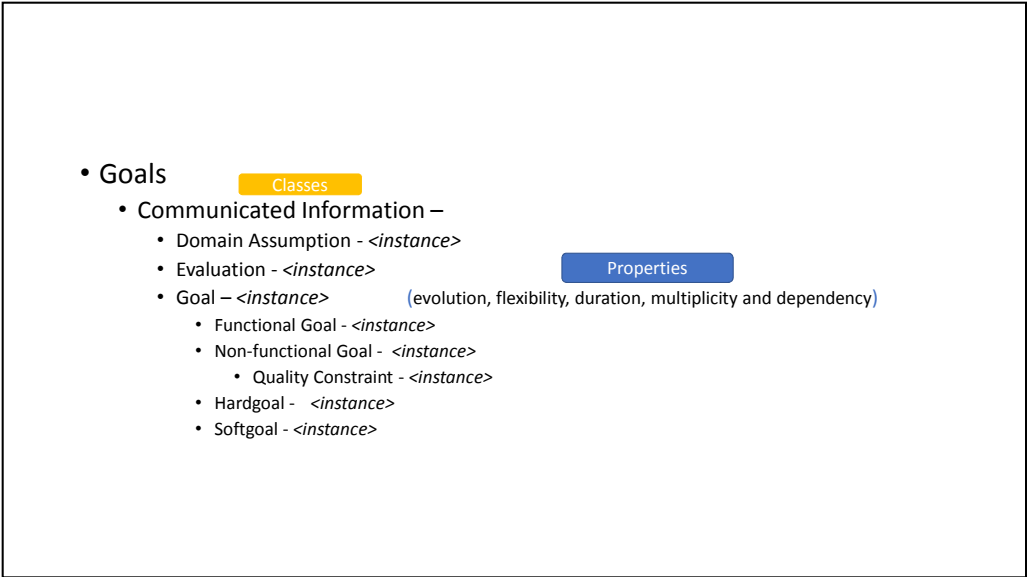
Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

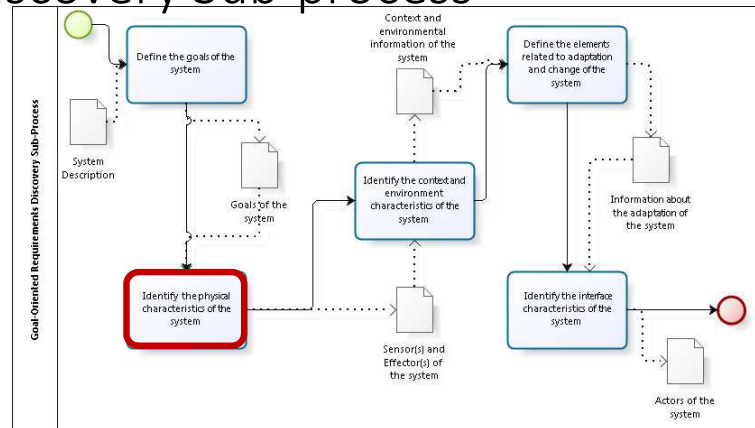
- **Activity 1:** Define the goals of the system
- **Objective:** Finding the goals of the system.
- **Description:** The engineer should answer a set of questions to the system description based on some concepts of Onto4SASGORE and some competence questions, in order to discovery all goals of the system.
- **Expected results:** The goals of the system.
- **Questions:**
 1. What is the main goal of the system?
 2. What are the secondary goals of the system?
 3. What are the characteristics of each goal?
 4. What are the functional goals of the system?
 5. What are the non-functional goals of the system?
 6. What are the constraints of the system?

06/07/2017



06/07/2017

Goal-Oriented Requirements Discovery Sub-process



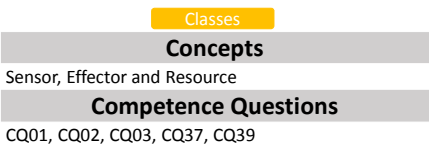
Goal-Oriented Requirements Discovery Sub-process

- **Activity 2:** Identify the physical characteristics of the system
- **Objective:** Finding the sensors and effectors of the system.
- **Description:** Some systems can have the sensor or effector actuating by software or device not physical as well. Therefore, this activity involves mechanism or resource concepts. The requirements engineer should ask the set of questions to the description system and answer based on the Onto4SASGORE concepts.
- **Expected results:** The identification of the sensor and the effector of the system.
- **Questions:**
 1. What is the sensor of the system?
 2. What is the effector of the system?
 3. What are the resources used by system?

06/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 2:** Identify the physical characteristics of the system

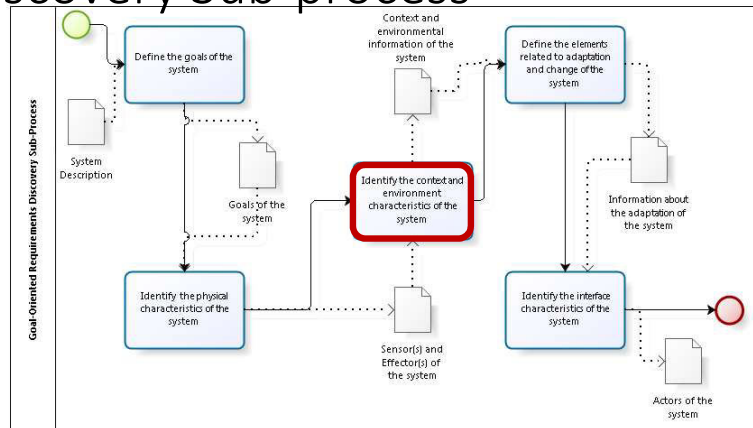


- Interface and Communication
 - Sensor – <instance>
 - Effector - <instance>
 - Resource - <instance>

Classes

06/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 3:** Identify the context and environment characteristics of the system
- **Objective:** Finding the context and environmental information about the system.
- **Description:** With the Onto4SASGORE ontology and the description of the system, the requirements engineer will answer the set of questions of this activity.
- **Expected results:** The context and environmental information about the system.
- **Questions:**
 1. What are the contexts considered by the system?
 2. What are the users of the system?
 3. What is the environment where the system is operating?

06/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 3:** Identify the context and environment characteristics of the system

Classes

Concepts

Context, Goal, Location, Time, Action, Role, Status, Actor and Environment

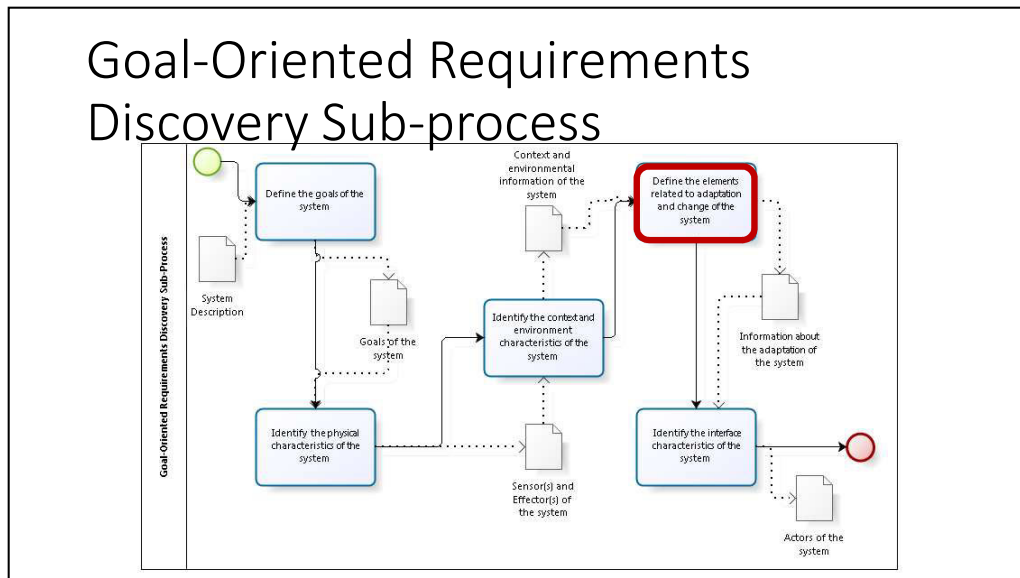
Competence Questions

CQ54, CQ55, CQ56, CQ57, CQ58, CQ59, CQ60

- Context and environment
 - Environment - <instance>
 - Behavior - <instance>
 - Context – <instance>
 - Location - <instance>
 - Status - <instance>
 - Time - <instance>
- Interface and Communication
 - Actor – <instance>
 - Human - <instance>
 - SW_Agent - <instance>
 - Role - <instance>

Classes

06/07/2017



Goal-Oriented Requirements Discovery Sub-process

- **Activity 4:** Define the requirements related to adaptation and change of the system.
- **Objective:** Finding all goal-oriented requirements related to adaptation and change of the system.
- **Description:** To aim the objective, the requirements engineer will answer the questions of this activity for the system description with the aid of the Onto4SAS ontology. This activity also involves goal, mechanism and effect concepts.
- **Expected results:** The information about the adaptation of the system.
- **Questions:**
 1. What are the actions of the system?
 2. What are the effects by action?
 3. What are the changes of the system?
 4. What is the cause of the change?
 5. What is the mechanism used to perform a change?

06/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 4:** Define the requirements related to adaptation and change of the system.

Classes

Concepts

Goal, Effects, Action, Predictability, Resilience, Criticality, Overhead, Change Plan, Symptom, Change Request, Goal, Duration, Flexibility, Dependency, Evolution, Change, Frequency, Change Type, Anticipation, Source, Mechanism, MechanismType, Timeliness, Triggering, Autonomy, Organization, Scope, Duration and Interface Mechanism

Competence Questions

CQ08, CQ09, CQ10, CQ11, CQ12, CQ13, CQ14, CQ15, CQ16, CQ17, CQ18, CQ28, CQ29, CQ30, CQ31, CQ32, CQ33, CQ34, CQ35, CQ36, CQ37, CQ38, CQ39, CQ40, CQ41, CQ42, CQ43, CQ44, CQ45, CQ46, CQ47, CQ48, CQ49, CQ50, CQ51, CQ52, CQ53, CQ61, CQ62, CQ63, CQ64, CQ65, CQ66, CQ67, CQ68, CQ69, CQ70

- Change and adaptation

Properties

- Action – (Precondition, input and output)
 - Analyze – <instance>
 - Execute – <instance>
 - Monitor – <instance>
 - Plan – <instance>

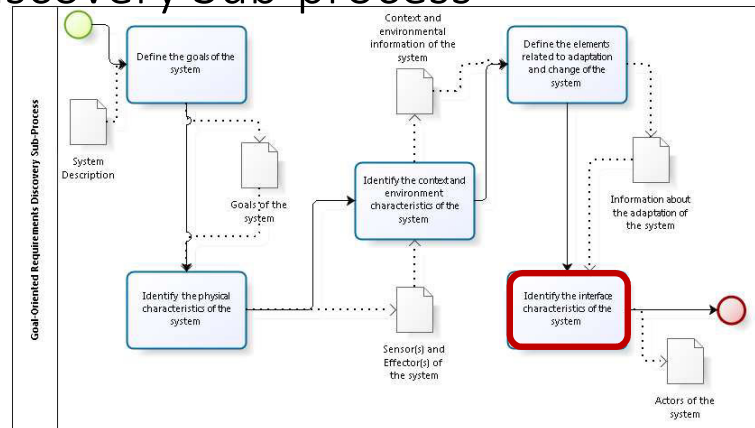
Properties

- Change – <instance> (source, type, frequency and anticipation)
- Effect - <instance> (criticality, predictability, overhead and resilience)
- Mechanism - <instance> (type, autonomy, organization, scope, duration, timeliness and triggering)
- Resource - <instance>
 - Change Request - <instance>
 - Plan – <instance>
 - Change Plan - <instance>
 - Symptom – <instance>

Classes

06/07/2017

Goal-Oriented Requirements Discovery Sub-process



Goal-Oriented Requirements Discovery Sub-process

- **Activity 5:** Identify the interface characteristics of the system.
- **Objective:** Identifying who or what the system communicates and interacts and the communication interface to realize it.
- **Description:** The requirements engineering will identify all the actors of the system present on system description through the Onto4SASGORE. He or she will answer the questions for this activity using the Onto4SASGORE concepts and competence questions.
- **Expected results:** Find the actors of the system.
- **Questions:**
 1. Who are the actors of the system?
 2. What is the input of each action of the system?
 3. What is the output of each action of the system?

06/07/2017

Goal-Oriented Requirements Discovery Sub-process

- **Activity 5:** Identify the interface characteristics of the system.

Concepts

Actor, Role, Input, Monitor, Analyze, Plan, Execute, Effector, Output, Mechanism and Sensor

Competence Questions

CQ05, CQ06, CQ07, CQ32, CQ33, CQ34, CQ35, CQ36, CQ37, CQ38, CQ39, CQ40, CQ41, CQ42, CQ43

- Interface and Communication

- Actor – <instance>
 - Human - <instance>
 - SW_Agent - <instance>
- Role - <instance>

- Change and adaptation

- Action – <instance> (Precondition, input and output)
 - Analyze – <instance>
 - Execute – <instance>
 - Monitor – <instance>
 - Plan – <instance>

Properties

06/07/2017

Questionnaire with 20 questions

1. What is the main goal of the system?
<Goal>, <Functional_Goal>, <Hardgoal>, <NonFunctional_Goal>,
<Softgoal>,<QualityConstraint>.
2. What are the secondary goals of the system?
<Goal>, <Functional_Goal>, <Hardgoal>, <NonFunctional_Goal>,
<Softgoal>,<QualityConstraint>.
3. What are the characteristics of each goal?
<Duration>, <Flexibility>, <Dependency>, <Evolution>.
4. What are the functional goals of the system?
<Functional_Goal>, <Hardgoal>, <Softgoal>.
5. What are the non-functional goals of the system?
<NonFunctional_Goal>, <QualityConstraint>

Questionnaire with 20 questions

6. What are the constraints of the system?
<QualityConstraint>.
7. What is the sensor of the system?
<Sensor>.
8. What is the effector of the system?
<Effector>.
9. What are the resources used by system?
<Resource>.
10. What are the contexts considered by the system?
<Context>, <Goal>, <Location>, <Time>, <Action>, <Role>, <Status>, <Actor>

06/07/2017

Questionnaire with 20 questions

11. What are the users of the system?
<Actor>.
12. What is the environment where the system is operating?
<Environment>.
13. What are the actions of the system?
<Action>, <Input>, <Output>.
14. What are the effects by action?
<Effects>, <Predictability>, <Resilience>, <Criticality>, <Overhead>.
15. What are the changes of the system?
<Goal>, <Effects>, <Action>, <Input>, <Output>, <Predictability>, <Resilience>, <Criticality>, <Overhead>, <ChangePlan>, <Symptom>, <Change Request>, <Goal>, <Duration>, <Flexibility>, <Dependency>, <Evolution>, <Change>, <Frequency>, <ChangeType>, <Anticipation>, <Source>, <Mechanism>, <MechanismType>, <Timeliness>, <Triggering>, <Autonomy>, <Organization>, <Scope>, <Duration>, <Interface Mechanism>

Questionnaire with 20 questions

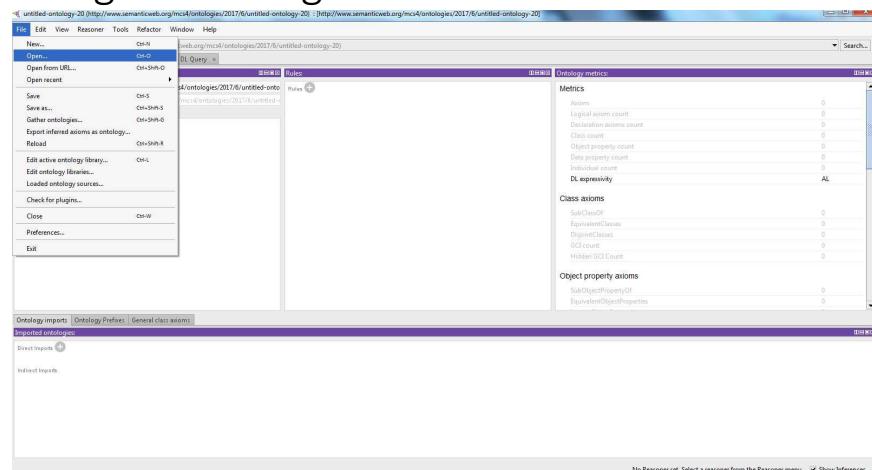
16. What is the cause of the change?
<Change>, <Source>.
17. What is the mechanism used to perform a change?
<Mechanism>, <MechanismType>, <Timeliness>, <Triggering>, <Autonomy>, <Organization>, <Scope>, <Duration>, <Interface Mechanism>.
18. Who are the actors of the system?
<Actor>, <Role>, <Input>, <Monitor>, <Analyze>, <Plan>, <Execute>, <Effector>, <Output>, <Mechanism>, <Sensor>.
19. What is the input of each action of the system?
<Action>, <Input>.
20. What is the output of each action of the system?
<Action>, <Output>.

06/07/2017

Using the Protégé


- Open the Protégé;
- Open the Onto4SASGORE in OWL.

Using the Protégé

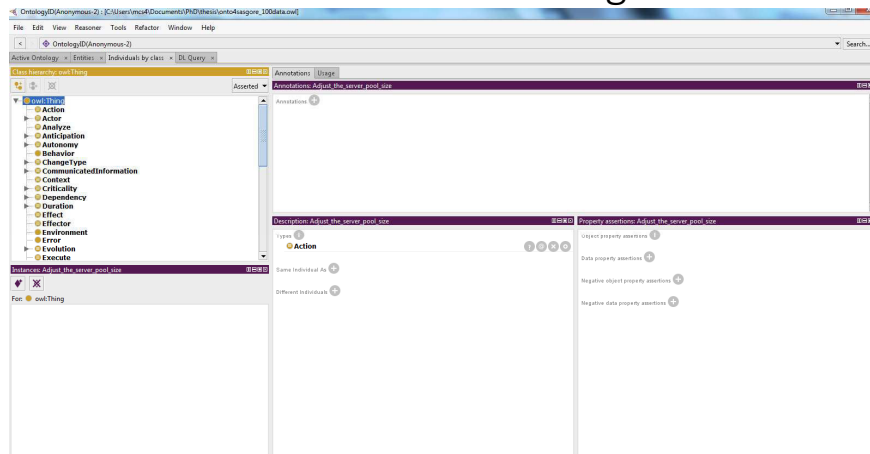


06/07/2017

Insert a Class Individual in Protégé

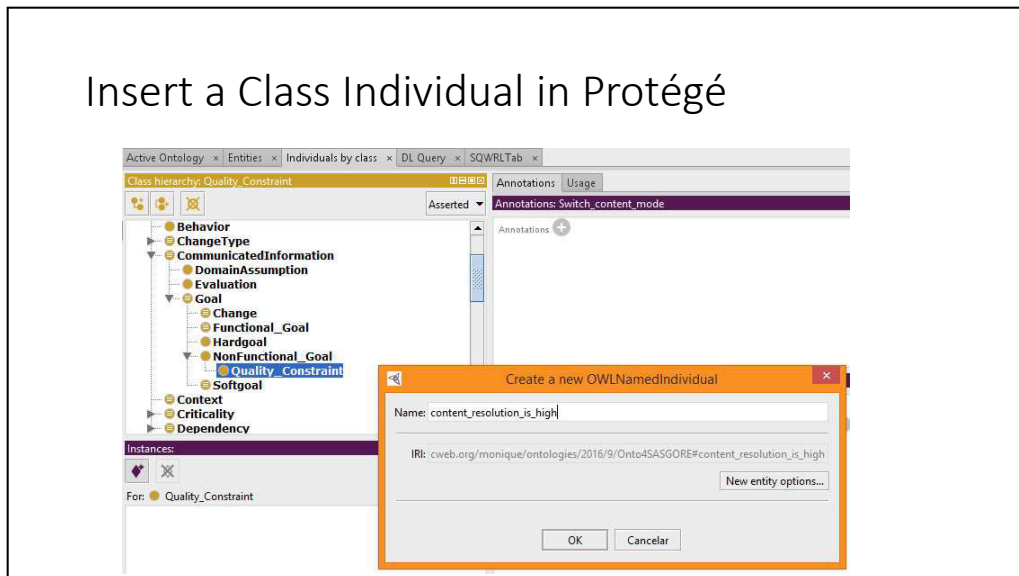
- 'Individual by class' tab
- Click on wished Class to create the individual
- In 'instances' area click on 
- Write the name of the individual and click on 'ok' button.

Insert a Class Individual in Protégé



06/07/2017

Insert a Class Individual in Protégé

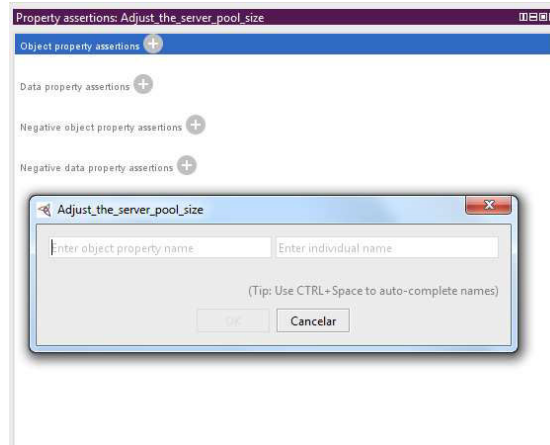


Insert a Property Individual in Protégé

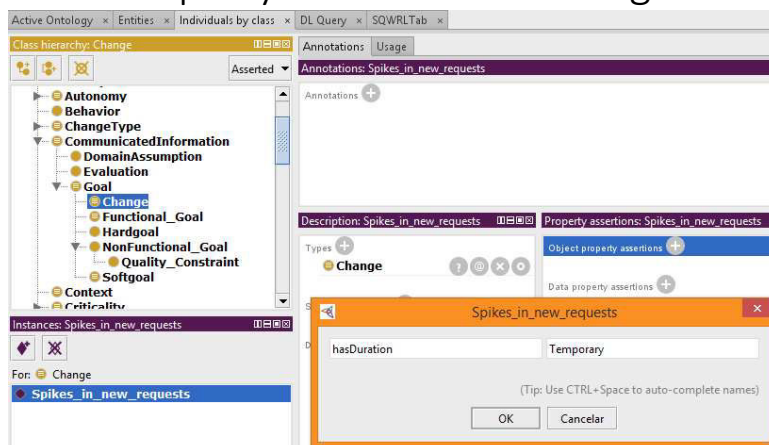
- 'Individual by class' tab
- Click on wished Individual to create the property
- In 'property assertions' area click on 'plus' symbol in Object property assertions +
- Write the property's name and the individual's name of the property and click on 'ok' button.

06/07/2017

Insert a Property Individual in Protégé




Insert a Property Individual in Protégé

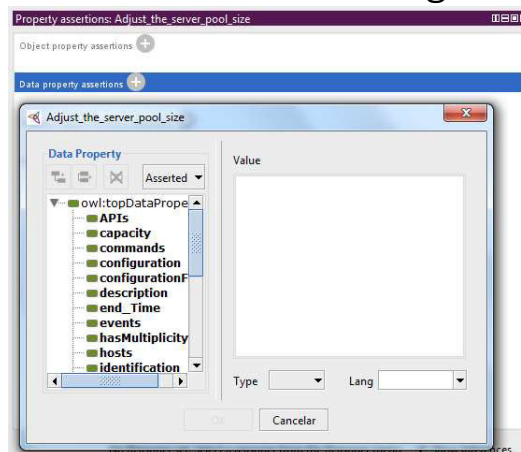


06/07/2017

Insert a Data Individual in Protégé

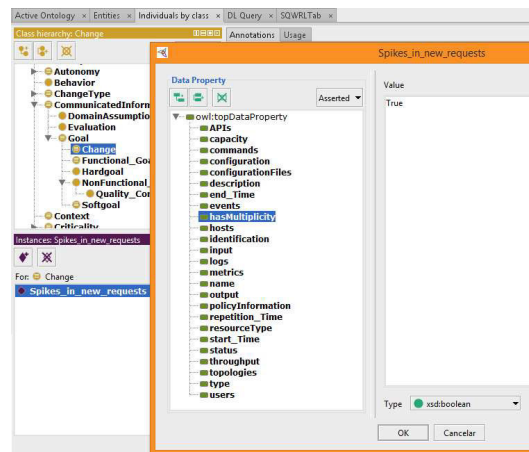
- 'Individual by class' tab
- Click on wished Individual to create the data property
- In 'property assertions' area click on 'plus' symbol in 
- Click on wished data property, write the value and click on 'ok' button.

Insert a Data Individual in Protégé

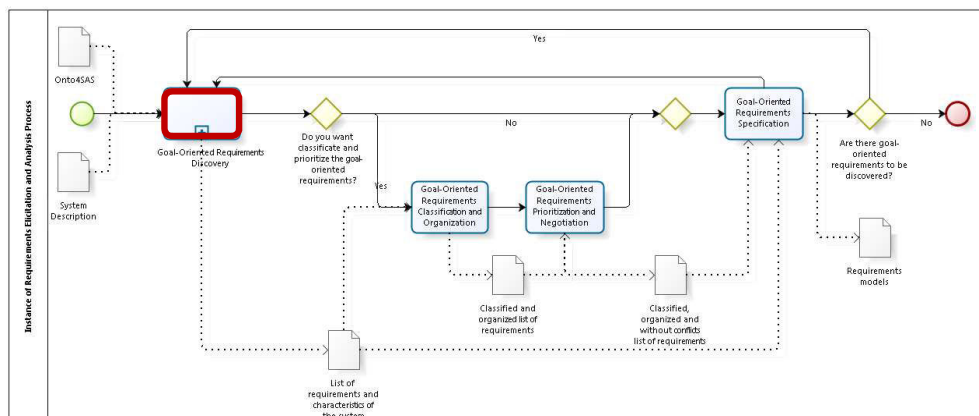


06/07/2017

Insert a Data Individual in Protégé



Onto4SASGORE Process



06/07/2017

Goal-Oriented Requirements Specification

• Template – Part 1 (Goal)

- Communicated Information –
 - Domain Assumption,
 - Evaluation,
 - Goal – (evolution, flexibility, duration, multiplicity and dependency)
 - Functional Goal -
 - Non-functional Goal -
 - Quality Constraint -
 - Hardgoal -
 - Softgoal -

Goal-Oriented Requirements Specification

• Template – Part 2 (Interface and Communication)

- Actor –
 - Human -
 - SW_Agent -
- Role -
- Sensor –
- Effector -
- Touchpoint -

06/07/2017

Requirements Specification

• Template – Part 3 (Context and Environment)

- Environment -
- Behavior -
- Context –
 - Location -
 - Status -
 - Time -
 - Role -

Requirements Specification

• Template – Part 4 (Change and Adaptation)

- | | |
|--|--|
| <ul style="list-style-type: none"> • Action – (Precondition, input and output) <ul style="list-style-type: none"> • Analyze – • Execute – • Monitor – • Plan – • Change – (source, type, frequency and anticipation) • Effect - (criticality, predictability, overhead and resilience) • Mechanism - (type, autonomy, organization, scope, duration, timeliness and triggering) | <ul style="list-style-type: none"> • Resource - <ul style="list-style-type: none"> • Change Request - • Plan – <ul style="list-style-type: none"> • Change Plan - • Symptom – • Goal – (evolution, flexibility, duration, multiplicity and dependency) <ul style="list-style-type: none"> • Functional Goal - • Non-functional Goal - <ul style="list-style-type: none"> • Quality Constraint - • Hardgoal - • Softgoal - |
|--|--|

06/07/2017

Finishing

- Once all individuals were created in Protégé, the requirements engineer can start a reasoner and find new characteristic of the systems.

APPENDIX F – CLEANER ROBOT DESCRIPTION

This Appendix is cited in Section 4.1.3 and Section 6.2.

F.1 Cleaning robot cycle

Cleaning robot cycle

At the end of a cleaning cycle or when cleaning robot battery is running low, it returns to the Home Base to charge. Cleaning robot needs to find the infrared signal of the Home Base in order to return. Considering that Home Base always plugged in.

If cleaning robot returns to the Home Base and is unable to dock, it will try again until it docks successfully.

The cleaning robot has the following functions: Clean environment; Pause the cleaning; Resume; Turn off; Empty robot's bin; clean filter.

The three-stage cleaning system: 1 - Side brush sweeps along edge of walls and into corners.

2 - Two counter-rotating brushes scoop up dirt, hair, and debris into the bin.

3 - Powerful vacuum picks up the remaining fine particles, dirt, and hair. The filter traps dust and small particles.

Cleaning robot works on wood, carpet, tile, vinyl and laminate, and adjusts automatically to different floor types. It automatically senses stairs and others cliffs.

The cleaning robot has a anti-tangle system. It won't get stuck on cords, carpet fringe or tassels. When cleaning robot senses it has picked up a cord or tassel, it will automatically stop its main brushes or side brush and try to escape.

Cleaning robot has several cleaning behaviours: Spiraling - It uses a spiral motion to clean a concentrated area.

Wall following - It cleans the full perimeter of the room and navigates around furniture and obstacles.

Room crossing - It crisscrosses the room to ensure full cleaning coverage.

Dirt Detection - When cleaning robot senses dirt, the white dirt detection light lights up and it cleans more intensely in that area.

Three cleaning modes: Clean mode - Cleaning robot automatically calculates the room size and adjusts its cleaning time appropriately.

Spot mode - It spirals approximately three feet (one meter) in diameter and then spirals

back to where it started, intensely cleaning a localized area.

Scheduled Cleaning Mode - When a future cleaning time is programmed, Cleaning robot enters this mode. At the specified time, it leaves its Home Base, cleans and then returns to the Home Base to recharge when it's done.

The cleaning robot has: Infrared sensor; Battery charger socket; Bin Release Button; Dust bin; Edge-Cleaning side brush; Dirt detect sensor.

The schedule sets: day, hour, minute, schedule, clock.

A virtual wall creates an invisible barrier that cleaning robot won't cross. If the battery is low, the virtual wall turns off.

Battery status: red - empty; green - fully charged; amber pulse - charging.

After each use to wipe/empty: Bin; Filter; Bristle brush.

Once per week: Sensor; Front wheel.

Errors: Sensor dirt - clean and start in a new location.

APPENDIX G – WHERE DID THIS COME FROM?

Table 41 presents the Onto4SASGORE concepts and the reference of some works that present the concept. This Appendix is cited in Section 4.3.

Table 41 – Onto4SASGORE Concepts and References

Concept	References
Action	(SALEHIE; TAHVILDARI, 2009), (JEONG-DONG; SON; BAIK, 2012), (VASSEV; HINCHEY, 2015a)
Actor	(MORANDINI; PENSERINI; PERINI, 2008), (JEONG-DONG; SON; BAIK, 2012)
Agent	(GUIZZARDI et al., 2014), (WEICHHART; NAUDET, 2014), (SUDHA et al., 2007), (TRAN; CIMIANO; ANKOLEKAR, 2006), (PANTSAR-SYVÄNIEMI et al., 2012), (STEFANOV; HUANG, 2009), (YILMAZ; ERDUR, 2012), (GU; PUNG; ZHANG, 2005), (VIEIRA; TEDESCO; SALGADO, 2005).
Analyze	(KEPHART; CHESS, 2003), (COMPUTING et al., 2006), (YUAN; MALEK, 2012), (SERBEDZIJA; FAIRCLOUGH, 2012), (GACEK; GIESE; HADAR, 2008).
Behavior	(KEPHART; CHESS, 2003), (VASSEV; HINCHEY, 2015a), (WEICHHART; NAUDET, 2014), (PANTSAR-SYVÄNIEMI et al., 2012), (RAMIREZ; JENSEN; CHENG, 2012), (QURESHI; PERINI, 2009).
Communicated Information	(QURESHI; JURETA; PERINI, 2011).
Domain Assumption	(JURETA; MYLOPOULOS; FAULKNER, 2008), (QURESHI; JURETA; PERINI, 2011) (QURESHI; JURETA; PERINI, 2012), (PIMENTEL et al., 2014).

Table 41 – Onto4SASGORE Concepts and References

Concept	References
Evaluation	(JURETA; MYLOPOULOS; FAULKNER, 2008), (QURESHI; JURETA; PERINI, 2011).
Task	(JURETA et al., 2015), (QURESHI; JURETA; PERINI, 2011) (QURESHI; JURETA; PERINI, 2012), (PIMENTEL, 2015), (VASSEV; HINCHEY, 2015a), (TRAN; CIMIANO; ANKOLEKAR, 2006), (MOORE; HU; WAN, 2008), (PREUVENEERS et al., 2004), (VIEIRA; TEDESCO; SALGADO, 2005).
Goal	(JURETA et al., 2015) (MORANDINI; PENSERINI; PERINI, 2008), (QURESHI; JURETA; PERINI, 2011), (QURESHI; JURETA; PERINI, 2012), (PIMENTEL et al., 2014) (VASSEV; HINCHEY, 2015a), (JEONG-DONG; SON; BAIK, 2012).
Functional Goal	(JURETA; MYLOPOULOS; FAULKNER, 2008). (GUIZZARDI et al., 2014), (QURESHI; JURETA; PERINI, 2011), (JEONG-DONG; SON; BAIK, 2012).
Non-Functional Goal	(JURETA; MYLOPOULOS; FAULKNER, 2008), (JEONG-DONG; SON; BAIK, 2012) (GUIZZARDI et al., 2014).
Quality Constraint	(JURETA; MYLOPOULOS; FAULKNER, 2008), (QURESHI; JURETA; PERINI, 2011), (QURESHI; JURETA; PERINI, 2012), (JURETA et al., 2015), (GUIZZARDI et al., 2014), (PIMENTEL et al., 2014).
Hardgoal	(GUIZZARDI et al., 2014) (MORANDINI; PENSERINI; PERINI, 2008).
Softgoal	(JURETA; MYLOPOULOS; FAULKNER, 2008), (JURETA et al., 2015), (MORANDINI; PENSERINI; PERINI, 2008), (QURESHI; JURETA; PERINI, 2011) (QURESHI; JURETA; PERINI, 2012) (GUIZZARDI et al., 2014).

Table 41 – Onto4SASGORE Concepts and References

Concept	References
Error	(MORANDINI; PENSERINI; PERINI, 2008)
Evolution	(ANDERSSON et al., 2009).
Flexibility	(ANDERSSON et al., 2009).
Duration	(ANDERSSON et al., 2009).
Multiplicity	(ANDERSSON et al., 2009).
Dependency	(ANDERSSON et al., 2009).
Conflicting	(ANDERSSON et al., 2009), (QURESHI; JURETA; PERINI, 2012).
Context	(QURESHI; PERINI, 2009) (QURESHI; JURETA; PERINI, 2011), (QURESHI; JURETA; PERINI, 2012) (JEONG-DONG; SON; BAIK, 2012), (SERBEDZ-LJA; FAIRCLOUGH, 2012), (TRAN; CIMIANO; ANKOLEKAR, 2006), (PANTSAR-SYVÄNIEMI et al., 2012), (HU; LI, 2009), (STEFANOV; HUANG, 2009), (YILMAZ; ERDUR, 2012), (GU; PUNG; ZHANG, 2005), (JAFFAL; GRAND; KIRSCH-PINHEIRO, 2015), (HOQUE et al., 2015), (MOORE; HU; WAN, 2008), (PREUVENEERS et al., 2004), (VIEIRA; TEDESCO; SALGADO, 2005).
Change	(ANDERSSON et al., 2009).
Source	(ANDERSSON et al., 2009).
Type	(ANDERSSON et al., 2009).
Frequency	(ANDERSSON et al., 2009).
Anticipation	(ANDERSSON et al., 2009).
Effect	(ANDERSSON et al., 2009), (JEONG-DONG; SON; BAIK, 2012).
Criticality	(ANDERSSON et al., 2009).

Table 41 – Onto4SASGORE Concepts and References

Concept	References
Predictability	(ANDERSSON et al., 2009).
Overhead	(ANDERSSON et al., 2009).
Failure	(ANDERSSON et al., 2009), (MORANDINI; PENSERINI; PERINI, 2008), (JEONG-DONG; SON; BAIK, 2012).
Resilience	(ANDERSSON et al., 2009).
Effector	(COMPUTING et al., 2006), (RAMIREZ; JENSEN; CHENG, 2012).
Environment	(WEICHHART; NAUDET, 2014), (TRAN; CIMIANO; ANKOLEKAR, 2006), (HU; LI, 2009), (JAFFAL; GRAND; KIRSCH-PINHEIRO, 2015), (HOQUE et al., 2015) (PREUVENEERS et al., 2004).
Execute	(COMPUTING et al., 2006).
Input	(JEONG-DONG; SON; BAIK, 2012).
Location	(SALEHIE; TAHVILDARI, 2009), (HU; LI, 2009), (STEFANOV; HUANG, 2009), (YILMAZ; ERDUR, 2012), (GU; PUNG; ZHANG, 2005) (JAFFAL; GRAND; KIRSCH-PINHEIRO, 2015), (PREUVENEERS et al., 2004), (VIEIRA; TEDESCO; SALGADO, 2005), (JEONG-DONG; SON; BAIK, 2012).
Mechanism	(ANDERSSON et al., 2009).
Type	(ANDERSSON et al., 2009).
Autonomy	(ANDERSSON et al., 2009).
Organization	(ANDERSSON et al., 2009), (JEONG-DONG; SON; BAIK, 2012).
Centralized	(YUAN; MALEK, 2012).
Decentralized	(YUAN; MALEK, 2012).

Table 41 – Onto4SASGORE Concepts and References

Concept	References
Scope	(ANDERSSON et al., 2009).
Duration	(ANDERSSON et al., 2009).
Timeliness	(ANDERSSON et al., 2009).
Triggering	(ANDERSSON et al., 2009), (PANTSAR-SYVÄNIEMI et al., 2012).
Monitor	(KEPHART; CHESS, 2003). (COMPUTING et al., 2006), (GACEK; GIESE; HADAR, 2008), (YUAN; MALEK, 2012).
Output	(JEONG-DONG; SON; BAIK, 2012).
Precondition	(JEONG-DONG; SON; BAIK, 2012)
Resource	(MORANDINI; PENSERINI; PERINI, 2008), (QURESHI; JURETA; PERINI, 2011) (QURESHI; JURETA; PERINI, 2012), (VASSEV; HINCHEY, 2015a), (MOORE; HU; WAN, 2008), (PREUVENEERS et al., 2004).
Change Request	(COMPUTING et al., 2006).
Plan	(COMPUTING et al., 2006), (KEPHART; CHESS, 2003), (MORANDINI; PENSERINI; PERINI, 2008), (QURESHI; PERINI, 2009).
Change Plan	(COMPUTING et al., 2006).
Symptom	(COMPUTING et al., 2006).
Planner	(GACEK; GIESE; HADAR, 2008), (YUAN; MALEK, 2012).
Role	(PANTSAR-SYVÄNIEMI et al., 2012), (JAFFAL; GRAND; KIRSCH-PINHEIRO, 2015), (MOORE; HU; WAN, 2008), (PREUVENEERS et al., 2004), (VIEIRA; TEDESCO; SALGADO, 2005), (JEONG-DONG; SON; BAIK, 2012).

Table 41 – Onto4SASGORE Concepts and References

Concept	References
Sensor	(KEPHART; CHESS, 2003). (COMPUTING et al., 2006), (SERBEDZIJA; FAIRCLOUGH, 2012), (GACEK; GIESE; HADAR, 2008), (RAMIREZ; JENSEN; CHENG, 2012), (STEFANOV; HUANG, 2009), (HOQUE et al., 2015).
Status	(JEONG-DONG; SON; BAIK, 2012).
Time	(SALEHIE; TAHVILDARI, 2009), (VASSEV; HINCHEY, 2015a), (WEICHHART; NAUDET, 2014), (HU; LI, 2009), (YILMAZ; ERDUR, 2012), (GU; PUNG; ZHANG, 2005), (JAFFAL; GRAND; KIRSCH-PINHEIRO, 2015), (HOQUE et al., 2015), (PREUVENEERS et al., 2004), (VIEIRA; TEDESCO; SALGADO, 2005), (JEONG-DONG; SON; BAIK, 2012).
Touchpoint	(COMPUTING et al., 2006).

Source: Own.