UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JOÃO PABLO SILVA DA SILVA

SaSML: A UML-Based Domain-Specific Modeling Language for Self-Adaptive Systems Conceptual Modeling

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Marcelo S. Pimenta

Porto Alegre
December 05, 2018

CIP — CATALOGING-IN-PUBLICATION

Silva da Silva, João Pablo

SaSML: A UML-Based Domain-Specific Modeling Language for Self-Adaptive Systems Conceptual Modeling / João Pablo Silva da Silva. – Porto Alegre: PPGC da UFRGS, 2018.

110 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2018. Advisor: Marcelo S. Pimenta.

1. Self-adaptive Systems. 2. Conceptual Modeling. 3. UML Extensions. I. S. Pimenta, Marcelo. II. Título.

I dedicate this work to
my wife Larissa B. L. da Silva
and my son Joaquim L. da Silva.

ACKNOWLEDGMENT

I would like to thank my wife Larissa B. L. da Silva for supporting me at home and making possible the work accomplishment.

I would like to thank my advisor Prof. Marcelo S. Pimenta for accepting the task of guiding me during the doctoral course. I would like to express my gratitude to my fellow students Ana C. T. Klock and João G. Faccin for presenting the papers in the conferences.

I would like to thank my coworkers Prof. Miguel da S. Ecar and Prof. Gilleanes T. A. Guedes for engaging in writing and proofreading. I would like to express my gratitude to my coworkers Prof. Carlos M. Betemps, Prof. Fabio N. Kepler, Prof. Elder de M. Rodrigues, and Prof. Sam da S. Devincenzi and my students Gabriel B. Moro, Eduardo F. Amaral, Luiz P. Franz, and Luciano A. M. de Paula for contributing to the research. I am also grateful to my coworker Prof. Alice F. Finger for reviewing the formal models produced in the research.

I would like to thank the Postgraduate Program in Computing (PPGC) of the Federal University of Rio Grande do Sul (UFRGS) for accepting me as a student. I would like to express my gratitude to the Federal University of Pampa (Unipampa) for supporting the research project. Finally, I am also grateful to all volunteers for participating of the empirical studies.

ABSTRACT

Self-adaptive Systems (SaSs) can autonomously decide how to adapt their behavior at runtime in response to contextual changes. They operate under uncertainty conditions and have intrinsic properties that have posed some challenges for software engineering. In this work, we approach issues related to SaS conceptual modeling, which is challenging because it is needed to deal with requirements uncertainty, contextual changes, and behavioral possibilities. Conceptual modeling is the act of creating models that describe problems independently of the solutions for purposes of understanding and communication. Unified Modeling Language (UML) is a graphical General-Purpose Modeling Language (GPML) that supports conceptual modeling through class diagrams. Once UML is a GPML, it does not have primitives customized to model SaSs, therefore, the modeling quality becomes dependent on the software engineers skills and abilities, which is not a good software engineering practice. This complexity can be minimized by using Domain-Specific Modeling Languages (DSMLs), which may be created by extending UML. Our literature investigation revealed that UML has been extended to SaSs domain, however, the SaSs higher-level abstractions have not been fully covered. We propose a UML-based DSML for SaSs conceptual modeling called SaSML that introduces a new modeling element called Adaptive Behavior. Adaptive Behavior is a wrapper that encapsulates a SaSs modeling schema, exposing only what needs to be defined at modeling time. SaSs modeling schema applies a set of design patterns to capture the higher-level abstractions related to SaSs domain. We carried out this work by establishing a conceptual and technological background, investigating the state-of-the-art of UML-based DSML for SaSs, defining a SaSs modeling schema, specifying the Adaptive Behavior syntax, semantics, and pragmatics, and evaluating SaSML through empirical techniques. The experiment results provide statistical evidence that the Adaptive Behavior modeling element provides an effective support for SaSs conceptual modeling, and it is able to synthesize the SaSs modeling schema without losing expressiveness. Thus, we can conclude that SaSML contributes to the SaSs conceptual modeling quality.

Keywords: Self-adaptive Systems. Conceptual Modeling. UML Extensions.

SaSML: Uma Linguagem de Modelagem de Domínio Específico Baseada em UML para Modelagem Conceitual de Sistemas Autadaptativos

RESUMO

Sistemas Autoadaptativos (SAs) podem decidir autonomamente como adaptar seu comportamento em tempo de execução em resposta às mudanças contextuais. Eles operam em condições de incerteza e possuem propriedades intrínsecas que impõem alguns desafios para a engenharia de software. Neste trabalho, abordamos questões relacionadas à modelagem conceitual de SAs, a qual é desafiadora porque é preciso lidar com incerteza de requisitos, mudanças contextuais e possibilidades comportamentais. Modelagem conceitual é o ato de criar modelos que descrevem problemas independentemente da solução para fins de entendimento e comunicação. *Unified Modeling Language* (UML) é uma Linguagem de Modelagem de Propósito Geral (LMPG) gráfica que suporta a modelagem conceitual através dos diagramas de classe. Uma vez que a UML é uma LMPG, ela não tem primitivas customizadas para modelar SAs, logo, a qualidade do modelo se torna dependente das competências e habilidades dos engenheiros de software, o que não é uma boa prática de engenharia de software. Essa complexidade pode ser minimizada com Linguagens de Modelagem de Domínio Específico (LMDEs), as quais podem ser criadas estendendo a UML. Nossa investigação da literatura revelou que a UML tem sido estendida para SAs, no entanto, as abstrações de alto nível relacionadas aos SAs não tem sido plenamente cobertas. Nós propomos neste trabalho uma LMDE baseada em UML chamada SaSML que introduz um novo elemento de modelagem chamado Adaptive Bahavior. O Adaptive Behavior é um empacotador que encapsula um esquema de modelagem, expondo somente o que precisa ser definido em tempo de modelagem. O esquema de modelagem aplica um conjunto de padrões de projeto para captura as abstrações de alto nível relacionadas ao domínio de SAs. Este trabalho foi realizado estabelecendo a fundamentação teórica e tecnológica, investigando o estado da arte de LMDE baseadas em UML para SAs, definindo o esquema de modelagem para SAs, especificando a sintaxe, semântica e pragmática do Adaptive Behavior e avaliando a SaSML através de técnicas empíricas. Os resultados evidenciaram que o Adaptive Behavior suporta a modelagem conceitual de SAs e sintetiza o esquema de modelagem de SAs sem perder expressividade. Portanto, concluímos que a SaSML contribui para a qualidade de modelos conceituais de SAs.

Palavras-chave: Sistemas Autoadaptativos, Modelagem Conceitual, Extensões UML.

LIST OF ABBREVIATIONS AND ACRONYMS

AAL	Ambient Assisted Living
CASE	Computer-Aided Software Engineering
DSML	Domain-Specific Modeling Language
EJB	Enterprise JavaBeans
ER	Entity-Relationship
ERP	Enterprise Resource Planning
FBTL	Fuzzy Branching Temporal Logic
GPML	General-Purpose Modeling Language
GQM	Goal/Question/Metric
LO	Learning Objects
MAPE	Monitor-Analyze-Plan-Execute
MDD	Model-Driven Development
MOF	Meta Object Facility
MVC	Model-View-Controller
OMG	Object Management Group
PBL	Problem-Based Learning
RE	Requirements Engineering
SaS	Sef-adaptive System
SLR	Systematic Literature Review
UML	Unified Modeling Language
Unipampa	Federal University of Pampa
UFRGS	Federal University of Rio Grande do Sul
WBS	Work Breakdown Structure

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Self-adaptive Systems (SaSs) can autonomously decide how to adapt their behavior at runtime in response to contextual changes (ANDERSSON et al., 2009; BRUN et al., 2009; CHENG et al., 2009). SaSs operate under uncertainty conditions (KRUPITZER et al., 2015) and have intrinsic properties (SALEHIE; TAHVILDARI, 2009), such as self-awareness, context-awareness, autonomic properties, and self-adaptiveness. These characteristics have posed some challenges for SaSs developing and maintaining. In the last years, software engineering research agendas have been formulated to address these challenges (MACÍAS-ESCRIVÁ et al., 2013). We have concerned with issues related to SaSs conceptual modeling (SILVA et al., 2018a; SILVA et al., 2018b; SILVA et al., 2018c).

SaSs conceptual modeling is challenging because it is needed to deal with requirements uncertainty, contextual changes, and behavioral possibilities. Conceptual modeling is the act of creating models that describe problems independently of the solutions for purposes of understanding and communication (BATINI; CERI; NAVATHE, 1992; MYLOPOULOS, 1992; CHEN; THALHEIM; WONG, 1999). Conceptual models are useful for requirements analysis because they aid in understanding the situation in which a problem occurs (BOURQUE; FAIRLEY, 2014). Unified Modeling Language (UML) is a graphical General-Purpose Modeling Language (GPML) used by industry and academy (BOOCH; RUMBAUGH; JACOBSON, 2005), which supports conceptual modeling through class diagrams (GOGOLLA, 2011).

1.1 Motivation

Models are represented by languages and they are restricted by the expressiveness power of these languages (THALHEIM, 2011). The model quality is directly related to its capability in providing to all stakeholders the same understanding of what it represents (HULL; JACKSON; DICK, 2011). In other words, a low expressiveness language makes harder to represent a domain in a clear and objective way, impacting in the resulting conceptual model comprehensibility.

UML is used by the software industry as modeling language to specify conceptual models (STÖRRLE, 2017). Once it is a GPML, it does not have primitives customized to express the abstraction related to SaSs domain. Therefore, SaSs conceptual model quality heavily depends on the software engineers skills and abilities. This might result in model

misinterpretations, unnecessary complexity, bad concerns separation, and cohesion and coupling problems. When it comes to SaSs, the exposure to quality risks increases because of intrinsic characteristics in this class of system.

In a software engineering[1] perspective, it would be useful to have a modeling language able to provide customized support for SaSs conceptual modeling. Domain-Specific Modeling Languages (DSMLs) provides primitives to express higher-level abstractions and constraints of a targeted domain (FRANK, 2011). A way for creating a DSML is extending UML by metamodel modifying or by profile creating, customizing it in order to meet the modeling needs (STAHL et al., 2006).

In face of foregoing, we propose the following main question: **How the UML could be customized for SaSs domain in order to contribute to the SaSs conceptual modeling quality?** In this work, the model quality refers to the capability to express SaS higher-level abstractions and constraints in a comprehensible way for all stakeholders.

1.2 Objectives

Our main goal is developing a UML-based DSML for SaSs conceptual modeling. Hence, we propose SaSML, a UML extension that introduces a new modeling element called Adaptive Behavior. Adaptive Behavior is a wrapper that encapsulates a SaSs modeling schema, exposing only what needs to be defined at modeling time. SaSs modeling schema applies a set of design patterns to capture the higher-level abstractions related to SaSs domain. Our main goal is decomposed in the following specific goals:

- establishing the related conceptual and technological background;
- investigating the state-of-the-art of UML-based DSMLs for SaSs modeling;
- defining a generic schema that supports SaSs conceptual modeling;
- extending UML to provide proper support for SaSs conceptual modeling;
- evaluating the proposed SaSs modeling schema and UML extension.

1.3 Hypothesis

We advocate that SaSML contributes to the modeling quality by capturing the higher-level abstractions related to SaSs domain. This hypothesis is supported by the

[1] Software engineering is “the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software” (BOURQUE; FAIRLEY, 2014, p. xxxi).

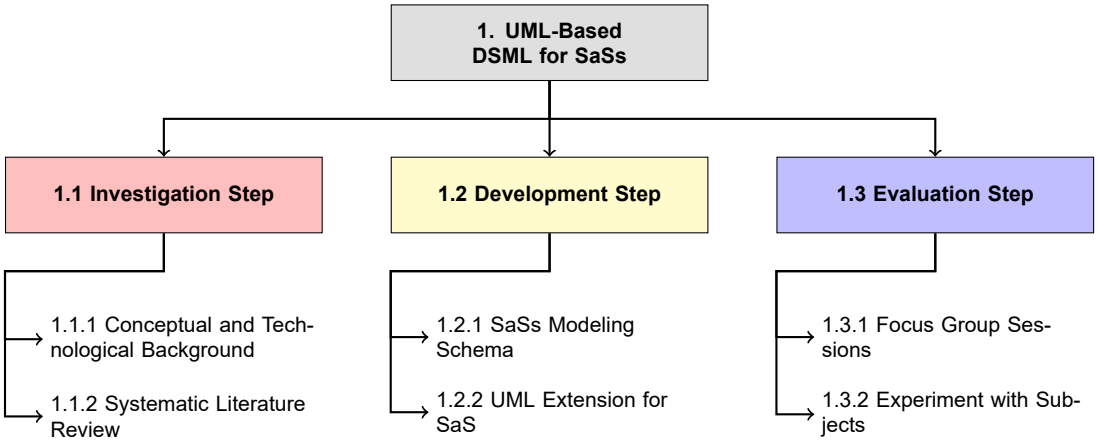
following arguments:

- the model quality is related to its communication capability (HULL; JACKSON; DICK, 2011);
- models are restricted by the languages expressiveness power (THALHEIM, 2011);
- DSMLs provide higher-level abstractions for mapping a targeted domain (FRANK, 2011).

1.4 Methodology

The Work Breakdown Structure (WBS)[2] presented in Figure 1.1 describes our work methodology. To achieve our objectives, we organized the research work in three distinct steps: investigation step, development step, and evaluation step. In the **investigation step**, we established the research conceptual and technological background, and we investigated the state-of-the-art by running a Systematic Literature Review (SLR). In the **development step**, we defined a modeling schema to capture the SaSs higher-level abstractions, and we extended UML introducing a new modeling element to synthesize schema in SaSs conceptual models. In the **evaluation step**, we evaluated the SaSs modeling schema by running focus group sessions, and we evaluated the proposed UML extension by performing an experiment with subjects.

Figure 1.1: Research work methodology.



Source: The authors.

[2] A WBS is a work scope hierarchical decomposition to be performed by the project team to achieve the project goals (PMI, 2013).

1.5 Contributions

The evaluation process results provide evidence that SaSML supports SaSs conceptual modeling. The Adaptive Behavior modeling element is able to synthesize the SaSs modeling schema without losing expressiveness. The design patterns usage in the SaSs modeling schema development contributed to a more flexible and reusable model. Moreover, we highlight the following complementary contributions:

- a metamodel and a process to support SaSs conceptual modeling published in 33rd ACM/SIGAPP Symposium On Applied Computing (SILVA et al., 2018b);
- a literature investigation about UML-based DSMLs for SaSs modeling published in 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SILVA et al., 2018a);
- a UML profile for SaSs conceptual modeling published in XXXII Brazilian Symposium on Software Engineering (SILVA et al., 2018c).

Additionally, we present in Appendix A the list of all academic works done during the doctoral course.

1.6 Organization

The remainder of this thesis is organized as follows. We present in **Chapter 2** the conceptual and technological overview of SaSs, conceptual modeling, and UML-based DSMLs, which provide the background required to understand the work motivation, objective, and methodology. We report in **Chapter 3** the results of a SLR that aimed to investigate how UML has been customized to create DSMLs that provide proper support for SaSs modeling. We propose in **Chapter 4** SaSML, a UML-based DSML that extends UML introducing a new modeling element called Adaptive Behavior, which is able to synthesize a SaSs modeling schema without lose expressiveness. We report in **Chapter 5** the empirical techniques used to evaluate the SaSs modeling schema correctness and completeness, and the Adaptive Behavior modeling element expressiveness and effectiveness. Finally, we summarize in **Chapter 6** the work results and conclusions, moreover, we discuss some work limitations and propose future work.

2 CONCEPTUAL AND TECHNOLOGICAL BACKGROUND

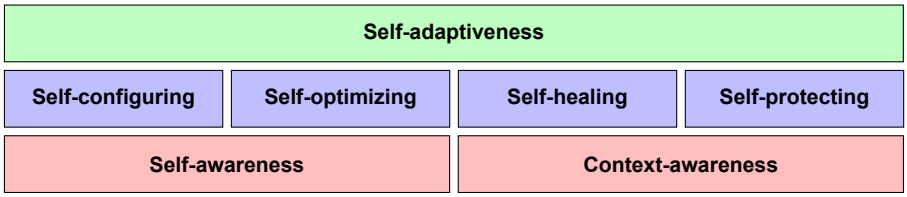
In this chapter, we present the conceptual and technological background required to understand the work motivation, objective, and methodology. We introduce in Section 2.1 the SaSs definition and characteristics, and a language for SaSs requirements specification. We present in Section 2.2 a conceptual modeling general review. We describe in Section 2.3 the DSMLs and how creating them from UML customization. We explain in Section 2.4 some design patterns relevant to this work. Finally, we summarize in Section 2.5 the chapter lessons.

2.1 Self-adaptive Systems

The specialized literature presents some definitions for SaSs (MACÍAS-ESCRIVÁ et al., 2013). However, the term "self-adaptive" is usually employed to denote that a system can autonomously decide how to adapt its behavior at runtime (OREIZY et al., 1999; ANDERSSON et al., 2009). This type of system is able to perceive contextual changes and (re)organize its features or services in response to these changes (BRUN et al., 2009; CHENG et al., 2009). SaSs have been proposed for several domains, such as, automotive, embedded, industrial, mobile, telecommunication, etc. The Software Engineering for SaSs community keeps a repository[1] of examples, problems, and solutions related to SaSs.

As can be seen in Figure 2.1, SaSs have a set of intrinsic characteristics known as **Self-*** properties, which are organized in a three levels structure (SALEHIE; TAHVILDARI, 2009).

Figure 2.1: SaSs intrinsic properties.



Source: Adapted from Salehie and Tahvildari (2009).

The bottom row is the **primitive level**, which contains the self-awareness and context-awareness properties (SALEHIE; TAHVILDARI, 2009). Both properties are aware-

[1] This exemplars repository is available at: <https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/>.

ness classes, where self-awareness refers to internal world and context-awareness refers to external world (VASSEV; HINCHEY, 2015). A self-aware system must have information about its internal state and it must also know its environment to determine how it is externally perceived (LEWIS et al., 2011). A context-aware system uses context to provide users with relevant information or services. Context is any information that can be used to define entity situation. Entity is anything considered relevant to a system, including the users and the system itself (DEY, 2001).

The middle row is the **major level**, which contains the four autonomic properties: self-configuring, self-healing, self-optimizing, and self-protecting (SALEHIE; TAHVILDARI, 2009). Self-configuring is the capability to automatically configure the system based on policies. Self-healing is the capability to automatically detect, diagnose, and repair problems. Self-optimizing is the capability to continuously improve their performance and efficiency. Self-protecting is the capability to automatically defends against attacks and failures (KEPHART; CHESS, 2003).

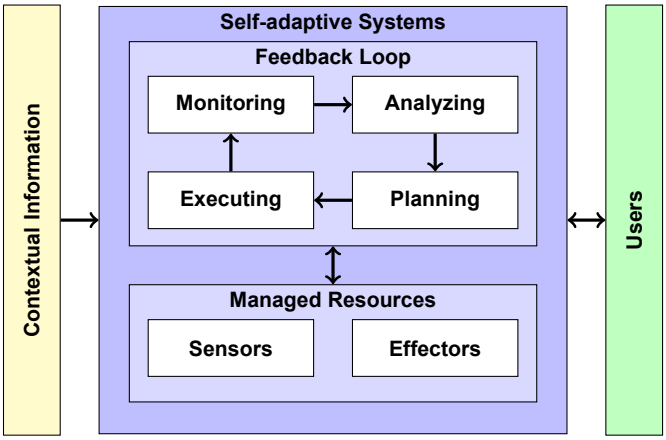
The top row is the **general level**, which contains the self-adaptiveness global property. Self-adaptiveness brings together all SaSs intrinsic characteristics (primitive and major level properties) (SALEHIE; TAHVILDARI, 2009). This property is directly related to SaSs definition itself.

A SaS makes decisions at runtime to control the dynamic behavior and it reasons about its state and environment (BRUN et al., 2009). Feedback control is an essential concept for self-adaptation because it guarantees that the observable output tracks the desired behavior (MACÍAS-ESCRIVÁ et al., 2013). We illustrate in Figure 2.2 a SaSs architecture basic layout, where an adaptation mechanism based on Monitor-Analyze-Plan-Execute (MAPE) architecture can be seen. The mechanism *monitors* the contextual information, *analyzes* data for changes, *plans* the adaptation, and *executes* the adaptation (KRUPITZER et al., 2015).

2.1.1 SaSs Requirements Specification Language

Requirements Engineering (RE) for SaSs have attracted the researchers interest because traditional approaches have not properly met the needs of this type of system (BENCOMO et al., 2010; SAWYER et al., 2010). Regarding SaSs requirements specification languages, there is in literature a set of works with relevant contributions, such as, Baresi and Pasquale (2010), Luckey et al. (2011), Luckey and Engels (2013), Souza et

Figure 2.2: SaSs architecture basic layout.



Source: Adapted from Krupitzer et al. (2015).

al. (2011), Souza et al. (2013), and Whittle et al. (2010). Among alternatives, we chose the RELAX language to specify the requirements in this work. Despite some limitation, RELAX meets our needs in terms of SaSs requirements specification.

RELAX is a language defined by Whittle et al. (2010) to express the uncertainty in SaSs requirements. Requirements written in RELAX take the form of structured natural language sentences enriched with operators and uncertainty factors. The RELAX vocabulary is composed of modal, temporal, and ordinal operators, and a set of uncertainty factors. The language has the syntax defined by a grammar and the semantics defined in terms of Fuzzy Branching Temporal Logic (FBTL).

We present in Table 2.1 an overview of the RELAX vocabulary according to Whittle et al. (2010). To illustrate RELAX usage, we present in Table 2.2 an Ambient Assisted Living (AAL) system requirement according to Whittle et al. (2010) definitions.

2.2 Conceptual Modeling

Conceptual modeling is the act of creating models that describe problem structures independently of the solution strategy (BATINI; CERI; NAVATHE, 1992; CHEN; THALHEIM; WONG, 1999). It forms a common basis for developers and stakeholders, integrating the domain expertise into software development (ROUSSOPOULOS; KARAGIANNIS, 2009). Conceptual models formally describe world aspects for purposes of understanding and communication (MYLOPOULOS, 1992). In other words, conceptual models map domain concepts and relations, reflecting real-world relationships and dependencies (BOURQUE; FAIRLEY, 2014).

Table 2.1: RELAX vocabulary overview.

Modal Operators	
SHALL	A requirement must hold.
MAY..OR	A requirement specifies one or more alternatives.
Temporal Operators	
EVENTUALLY	A requirement must hold eventually.
UNTIL	A requirement must hold until a future position.
BEFORE | AFTER	A requirement must hold before or after a particular event.
IN	A requirement must hold during a particular time interval.
AS EARLY | LATE AS POSSIBLE	A requirement specifies something that should hold as soon as possible or should be delayed as long as possible.
AS CLOSE AS POSSIBLE TO [f]	A requirement specifies something that happens repeatedly but the frequency may be relaxed.
Ordinal Operators	
AS CLOSE AS POSSIBLE TO [q]	A requirement specifies a countable quantity but the exact count may be relaxed.
AS MANY | FEW AS POSSIBLE	A requirement specifies a countable quantity but the exact count may be relaxed.
Uncertainty Factors	
ENV	Defines a set of properties that specify the system environment.
MON	Defines a set of properties that can be monitored by the system.
REL	Defines the relationship between the ENV and MON properties.
DEP	Identifies the dependencies between the requirements.

Source: Whittle et al. (2010).

Table 2.2: AAL system requirement written in RELAX.

Req-1	The fridge SHALL detect and communicate information with AS MANY food packages AS POSSIBLE.	
	ENV:	Food locations, Food item information, and Food state.
	MON:	RFID readers, Cameras, Weight sensors.
	REL:	RFID tags provide Food locations, Food information, and Food state; Cameras provide Food locations; and Weight sensors provide Food information.
	DEP:	None.

Source: Whittle et al. (2010).

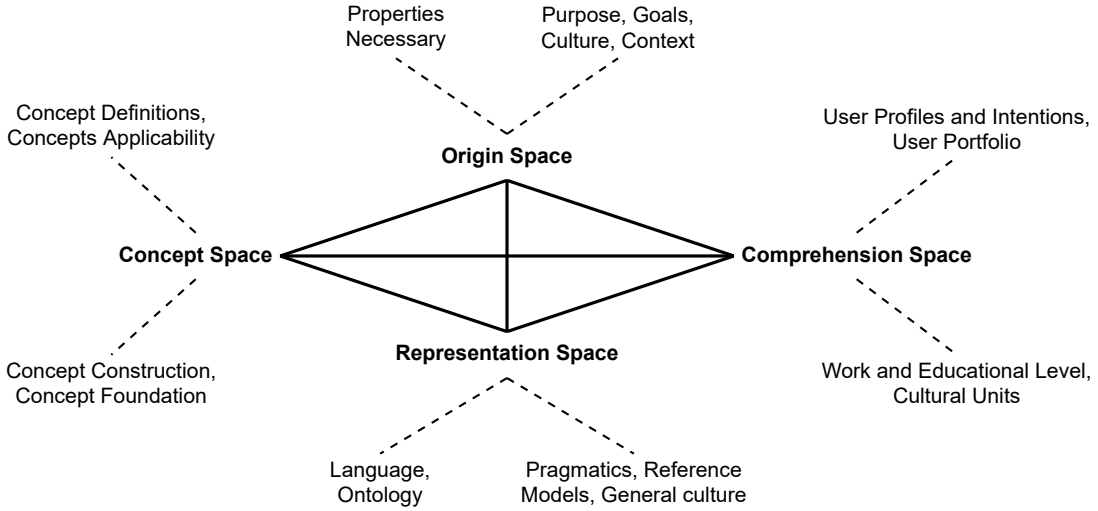
Conceptual models are abstract representations of a situation under investigation, which have the following commonalities:

- they preserve the purpose, which is invariant and governs the modeling process;
- they are a mapping of an origin, reflecting the properties observed and envisioned;
- they are represented by languages and restricted by the languages expressiveness;
- they provide a worth based on their utility, capability, or quality properties (THALHEIM, 2011).

The conceptual model space may be characterized through four different aspects: its origins, concepts, representations, and comprehensions by related stakeholders. The

model elements should not be considered in isolation, but rather as a suite of abstractions for theses aspects, as illustrated in Figure 2.3. Origins are defined through properties, purpose, goals, and context. Concepts are described in terms of their definitions, applicability, construction, and foundation. Representations are established through languages, ontologies, pragmatics, reference models, and general culture. Comprehensions are obtained from user profiles, intentions and portfolio, work and educational level, and cultural units (THALHEIM, 2011).

Figure 2.3: Conceptual model space characterization.



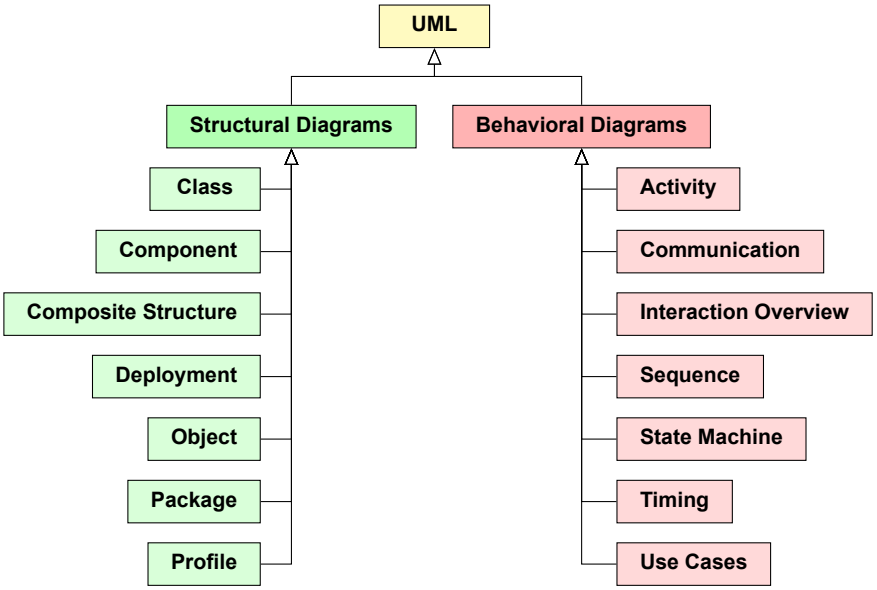
Source: Thalheim (2011).

2.2.1 UML as Conceptual Modeling Language

UML is a graphical GPML broadly used by industry and academy (BOOCH; RUMBAUGH; JACOBSON, 2005; STÖRRLE, 2017). As depicted in Figure 2.4, UML proposes a set of diagrams organized in two groups. Structural diagrams allow to represent the static system structure. Behavioral diagrams allow to represent the dynamic object behavior in a system (OMG, 2017).

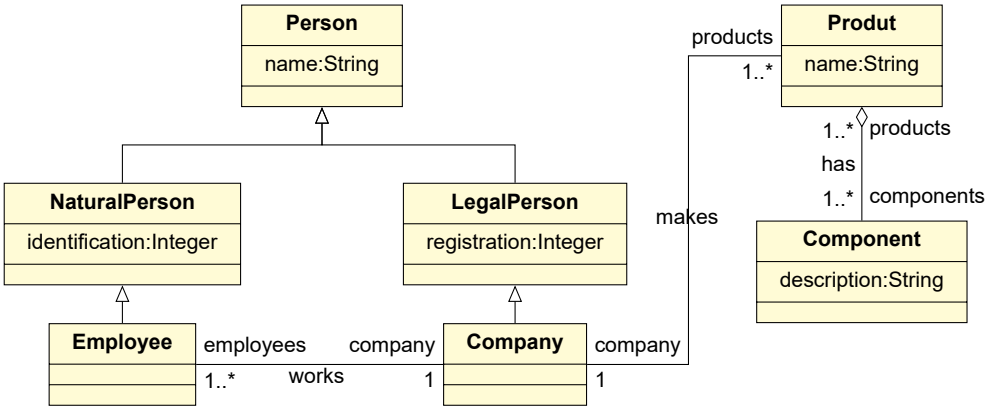
UML supports conceptual modeling through class diagrams (GOGOLLA, 2011), which capture the system structure as classes, constraints, and relationships (OMG, 2017). Classes are used to specify the domain entities. Attributes are used to specify the entities properties. Associations, Aggregations, and Generalizations are used to specify relationships among entities. Multiplicities are used to specify the relationships cardinality. We present in Figure 2.5 a conceptual model example to illustrate the UML usage.

Figure 2.4: UML diagrams hierarchy.



Source: Adapted from OMG (2017).

Figure 2.5: Conceptual model example specified in UML.



Source: The authors.

Employee class is a subclass of NaturalPerson class, which in turn is a subclass of Person class. Company class is a subclass of LegalPerson class, which also is a subclass of Person class. Employee class has a many-to-many association with Company class. Company class has a one-to-many association with Product class. Product class has a one-to-many aggregation association with Component class.

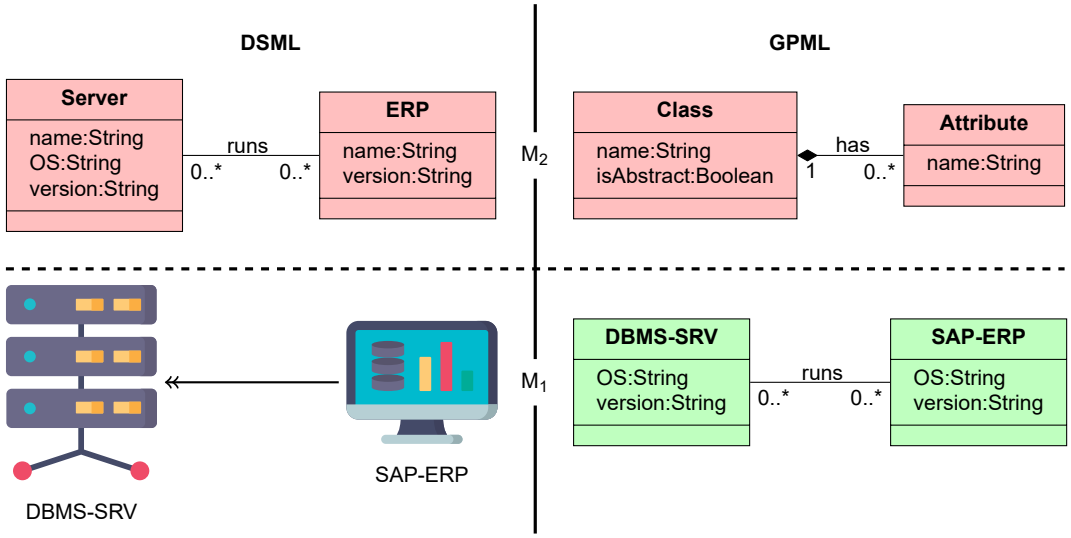
2.3 Domain-Specific Modeling Languages

DSMLs provide primitives to express higher-level abstractions and constraints of a targeted domain. They promote modeling productivity because software engineers may reuse concepts instead of specify them. Moreover, they contribute to quality because

concepts include implicit integrity constraints, which prevent construction of nonsensical models (FRANK, 2011). DSMLs[2] have been used for expressing concepts of several domains, such as, automotive, embedded, enterprise applications, industrial, mobile, telecommunication, user interface design, testing, etc.

We illustrate in Figure 2.6 some benefits of DSMLs compared to GPMLs. GPMLs provide software engineers with basic primitives, such as, class, attribute, etc. DSMLs, in turn, provide them with customized primitives that capture domain expertise. Furthermore, DSMLs usually have a specific graphical notation that maximize the models comprehensibility (FRANK, 2011).

Figure 2.6: DSML in comparison to GPML.



Source: Adapted from Frank (2011). Icons made by Freepik from www.flaticon.com is licensed by CC 3.0 BY.

2.3.1 UML-Based DSML

A way for creating a DSML is by extending the UML metamodel, modifying or introducing the modeling elements in order to meet the modeling needs (STAHL et al., 2006). There are two extension approaches that allow creating UML-based DSMLs:

- **metamodel-based** extends the UML metamodel by applying a language of the higher-metalevel, for example, the Meta Object Facility (MOF);
- **profile-based** extends the UML metamodel by applying mechanisms defined by UML itself, such as, stereotypes, tagged values, and constraints (STAHL et al., 2006).

[2]DSMLs real examples are available at: <http://www.metacase.com/cases/dsm_examples.html>.

Metamodel-based extension approach allows to modify the UML syntax and semantics (OMG, 2017), for example, it is possible to define a new syntax to represent views in a Model-View-Controller (MVC) architecture or modify the attributes semantics to represent primary and foreign keys in Entity–Relationship (ER) model.

Profile-based extension approach allows to extend UML with concepts coming from a particular platform or domain (MALAVOLTA; MUCCINI; SEBASTIANI, 2015), such as, embedded systems domain or Enterprise JavaBeans (EJB) platform, but it is not allowed modifying existing metamodels (OMG, 2017).

We present in Table 2.3 a profile and metamodel customizing options comparison according to Bruck and Hussey (2008).

Table 2.3: Profile and metamodel customizing options comparison.

Customizing Option	**Profile-based**	**Metamodel-based**
Keyword support	Yes	Yes
Icon support	Yes	Yes
Restrict or constrain existing types	Yes	Yes
Extend existing types	Yes	Yes
Add new types that do not extend an existing type	No	Yes
Remove existing type	No	Yes
Add new properties	Yes	Yes
Remove existing properties	No	No
Restrict/constrain existing properties	Yes	Yes
Add new operations	No	Yes
Remove existing operations	No	Yes
Restrict/constrain existing operations	Yes	Yes
Reuse UML concepts	Yes	Yes
Restrict multiplicity	Yes	Yes
Remove existing constraints	No	Yes
Add new constraints	Yes	Yes
First class extensibility	No	Yes
Validation	No	Yes
Programmatic usage	Awkward	Easy
Efficiency of running code	Not optimal	Very efficient
Cost of development	Medium	Highest
Ability to evolve	Easy	Difficult
Deploy so end users can work with	Easy	Easy
Dependency on UML implementation	No	No

Source: Adapted from Bruck and Hussey (2008).

2.4 Design Patterns

Design patterns are tested and proven reusable solutions to common problems in software design (GAMMA et al., 1994; FREEMAN et al., 2004; SHALLOWAY; TROTT, 2004). A design pattern is not a finished design, but it is a schema for how to solve a problem, which can be applied in different situations (SHVETS, 2018). A schema can be defined as an understandable representation of an idea in the form of a model (CAMBRIDGE, 2018; OXFORD, 2018). Each design pattern can be classified according to its purpose:

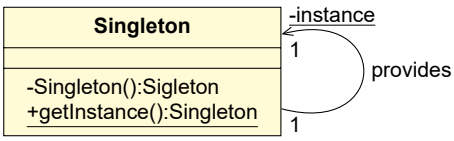
- **creational design patterns**, which deal with object creation process;
- **structural design patterns**, which deal with class and object relationships;
- **behavioral design patterns**, which deal with objects communication (GAMMA et al., 1994).

There is a wide variety of design patterns for the most varied problems. The choice of a design patterns requires a analysis process that allow understanding the problem to be solved. In addition, it is also necessary understand the design pattern purpose and how it can solved the problem under analysis (GAMMA et al., 1994).

2.4.1 Singleton Design Pattern

Singleton is a creational design pattern, which ensures that a class has only one instance, providing a global access point to it. We illustrate in Figure 2.7 the Singleton structure. A singleton class has a private constructor to avoid external instantiation. The single instance is created on the first access and stored in a private static attribute. The instance is accessed through a public static method (SHVETS, 2018).

Figure 2.7: Singleton structure.

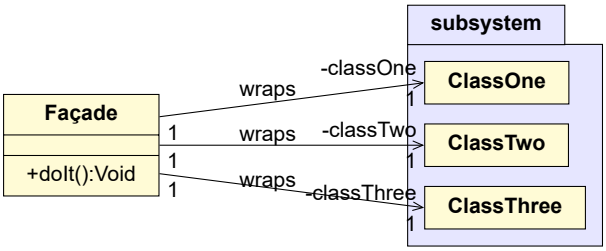


Source: Adapted from Shvets (2018).

2.4.2 Façade Design Pattern

Façade is a structural design pattern, which provides a higher-level interface to a complex subsystem, making it easier to use. We illustrate in Figure 2.8 the Façade structure. A façade class is a wrapper that encapsulates the subsystem, capturing the component complexity and collaborations and delegating to the appropriate methods (SHVETS, 2018).

Figure 2.8: Façade structure.

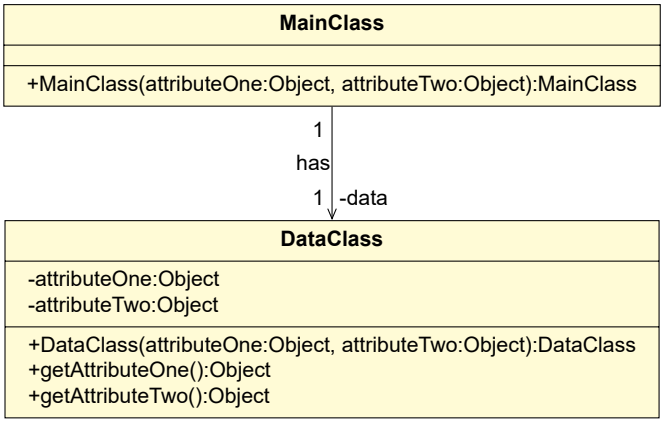


Source: Adapted from Shvets (2018).

2.4.3 Private Class Data Design Pattern

Private Class Data is a structural design pattern, which controls the access to class attributes by separating data from methods that use it. We illustrate in Figure 2.9 the Private Class Data structure. The data class contains all attributes that need hiding. The main class instantiates data class, initializing data class through its constructor (SHVETS, 2018).

Figure 2.9: Private Class Data structure.

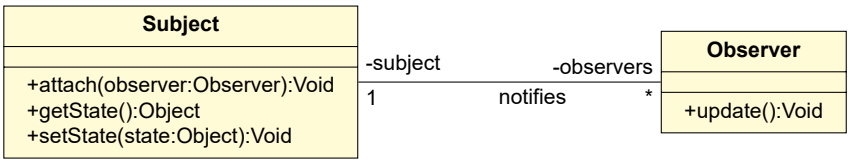


Source: Shvets (2018).

2.4.4 Observer Design Pattern

Observer is a behavioral design pattern, which defines a relationship between objects in such a way when one object changes state, all related objects are notified automatically. We illustrate in Figure 2.10 the Observer structure. The subject class represents the core abstraction, whereas the observer class represents the variable abstraction. A subject notifies the known observers every time that its state changes. A observer can access the subject when needed (SHVETS, 2018).

Figure 2.10: Observer structure.

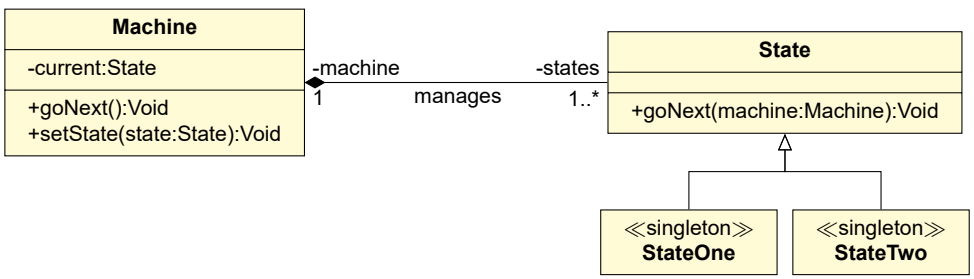


Source: Shvets (2018).

2.4.5 State Design Pattern

State is a behavioral design pattern, which uses an object-oriented state machine to alter the object behavior when its internal state changes. We illustrate in Figure 2.11 the State structure. The machine class is a wrapper that encapsulates a state machine. It knows the possible states and it implements the transitions rules. The state class defines the default behavior and each state subclass overrides the methods needed to provide the specific behavior. All requests to the machine class are delegated to the current state object (SHVETS, 2018).

Figure 2.11: State structure.



Source: Shvets (2018).

2.5 Chapter Lessons

In this chapter, we present the conceptual and technological overview of SaSs, conceptual modeling, and UML-based DSMLs, which provide the background required to understand the work motivation, objective, and methodology. The chapter main points are:

- SaSs can autonomously decide how to adapt their behavior at runtime in response to contextual changes;
- RELAX is a language to express the uncertainty in SaSs requirements through a set of operators and uncertainty factors;
- conceptual models map domain concepts and relations, reflecting real-world relationships and dependencies;
- DSMLs provide primitives to express higher-level abstractions and constraints of a targeted domain;
- one of the ways to create DSMLs is by extending the UML metamodel, modifying or introducing the modeling elements;
- design patterns are tested and proven reusable solutions to common problems in software design.

In next chapter, we report the results of a SLR that aimed to investigate how UML has been customized to create DSMLs for SaSs modeling.

3 UML-BASED DSML FOR SAS LITERATURE REVIEW

In this chapter, we report the results of a SLR that aimed to investigate how UML has been customized to create DSMLs that provide proper support for SaSs modeling. We present in Section 3.1 some studies that report related SLRs. We describe in Section 3.2 the protocol that guided our work. We report in Section 3.3 the results obtained with the protocol execution. We discuss in Section 3.4 threats to the SLR validity. Finally, we summarize in Section 3.5 the chapter lessons.

3.1 Related Work

We searched in scientific literature secondary studies[1] that investigated UML-based DSMLs for SaSs. As a result, we found 10 secondary studies published between 2008 and 2016 that somehow approached the theme.

Van Velsen et al. (2008) carried out a literature review to discover how the user-centered evaluations of personalized systems have been conducted and how they can be improved. Weyns et al. (2012) performed a study to understand and characterize the use of formal methods in SaSs. Patikirikorala et al. (2012) investigated the design of SaSs using control engineering approaches. Becker, Luckey and Becker (2012) provided a literature review of the state-of-the-art in performance engineering for SaSs. Mohabbati et al. (2013) aimed to identify and characterize the existing research on service-orientation and software product line engineering.

Weyns and Ahmad (2013) provided a comprehensive study that identifies claims and evidence for architecture-based self-adaptation. Yang et al. (2014) carried out a literature review of requirements modeling and analysis for SaSs. Yuan, Esfahani and Malek (2014) performed a systematic survey of the state-of-the-art of self-protecting software systems using a classifying taxonomy. Brings, Salmon and Saritas (2015) proposed a search strategy for a systematic review on uncertainty. Siqueira et al. (2016) performed a literature review to characterize the challenges for testing adaptive systems.

Analyzing these secondary studies, we noticed that most of them have relevant contributions. However, we also perceived that they did not answer our main question.

[1] According to Jalali and Wohlin (2012), research literature may be divided into primary studies, which are new studies on a specific topic, or secondary studies, which summarize the research current state on a specific topic.

Hence, we carried out a new SLR to investigate how UML has been customized to create DSMLs for SaSs modeling.

3.2 Review Protocol

We performed this SLR according to guidelines presented by Kitchenham and Charters (2007). As a first step, we established the research questions that guided all the review process. Firstly, we would like to understand the challenges related to SaS modeling and how they have been addressed. Secondly, we would like to understand the customization processes and what UML primitives have been extended. Thus, we defined the following research questions:

- **RQ-1:** What modeling issues have motivated UML customization?
- **RQ-2:** How UML has been customized to support SaSs modeling?

3.2.1 Search Process

We applied in this SLR the snowballing technique[2] to discover studies. According to the technique, we started by establishing an initial studies list, which was used as starting point for snowballing. To achieve it, we retrieved all papers published in International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). We chose SEAMS as the source because it is one of the main scientific events of the Software Engineering for Self-Adaptive Systems Community. After retrieving studies from SEAMS, we defined the initial studies list by applying the inclusion criteria.

Since the initial studies list has been defined, we performed the backward and the forward steps according to the snowballing technique. For each entry in initial studies list, we retrieved all papers that it cited (backward snowballing) and all papers that cite it (forward snowballing). The retrieved studies were verified to eliminate redundancies and ensure information completeness. We reapplied inclusion criteria in the retrieved papers to select new studies for the list. These steps (backward and forward snowballing) were iterated until no more new studies were found. Hence, in each iteration, we selected new relevant studies from the retrieved studies.

[2]Snowballing refers to using study references and citations to identify additional studies (WOHLIN, 2014).

3.2.2 Selection Process

We selected the relevant studies by applying the inclusion and exclusion criteria in the retrieved studies list.

The **inclusion criteria** acted as the first filter, accepting studies directly related to the research objective. As stated previously, the inclusion criteria was applied in each snowballing round by a peer review process. We accepted studies that received "Yes" in all of the following questions:

- Is there in the abstract any mention to UML extensions?
- Is there in the abstract any mention to SaSs properties[3]?

The **exclusion criteria** acted as the second filter, excluding studies that did not meet some general requirements. Differently from inclusion criteria, the exclusion criteria were applied in the backward and the forward output, also by a peer review process. We rejected studies that received "No" in at least one of the following criteria:

- Is the study a full primary study?
- Is the study written in English?
- Does the study have at least six pages?
- Is not the study a repeated publication?

We noted that, in some cases, a study has more than one version (conference and journal, for example). In these cases, we considered the study as a repeated publication and selected the more complete version.

3.2.3 Data Extraction Strategy

The data extraction was performed by a peer review process, in which the reviewers read the selected studies and collected the following data:

- addressed SaSs properties;
- approached engineering models;
- customized UML diagrams;
- applied extension mechanisms.

After extraction, we tabulated the data to summarize the basic information about each study, ordering by publication year. The data were analyzed to answer the research questions based on the following simple strategy:

[3] According to the set of intrinsic characteristics presented in Figure 2.1.

- to answer the RQ-1, we related SaSs properties with engineering models;
- to answer the RQ-2, we related UML diagrams with extension mechanisms.

3.3 Review Results

We ran the snowballing technique on January 8-10, 2018. We present in Table 3.1 the snowballing execution history. **Iteration 0** refers to the initial studies list definition, which is composed of all papers published in SEAMS between 2006 and 2017. **Iterations 1 to 6** refer to the execution of backward and forward snowballing. As an output, we retrieved 786 studies, of which 26 were accepted by applying the inclusion criteria.

Table 3.1: Snowballing execution history.

Iteration	Retrieved	Selected
Iteration 0	222	2
Iteration 1	59	4
Iteration 2	69	2
Iteration 3	31	6
Iteration 4	254	9
Iteration 5	100	3
Iteration 6	51	0
Total	**786**	**26**

Source: The authors.

After finishing the backward and the forward execution, we concluded the selection process by applying the exclusion criteria. As an output, we rejected 10 from 26 studies previously accepted, therefore, we **accepted 16 primary studies**[4] for this SLR, which are presented in Table 3.2. Since the relevant studies list has been defined, we performed data extraction and we analyzed the results to answer the research questions.

Sheng and Benatallah (2005) propose a UML profile for the context-aware services modeling. The profile extends the class diagram to model context, context-aware objects, and context-aware mechanisms. Ayed, Delanote and Berbers (2007) present a UML profile to specify adaptation in context-aware systems. The profile allows creating analysis and design models based on Model-Driven Development (MDD) principles.

Simons and Wirtz (2007) propose a UML profile to specify context models in mobile distributed systems. The profile allows modeling types of context items, the association among them, and the contextual situations related to the system. Fuentes, Gamez and Sanchez (2008) present a UML profile to design pervasive applications. The profile

[4] All retrieved studies, with their respective status, are available at <https://goo.gl/xkVjTi>.

Table 3.2: SLR selected studies.

Year	Title	Authors
2005	ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development	SHENG; BENATALLAH
2007	MDD Approach for the Development of Context-Aware Applications	AYED; DELANOTE; BERBERS
2007	Modeling context in mobile distributed systems with the UML	SIMONS; WIRTZ
2008	Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications	FUENTES; GAMEZ; SANCHEZ
2008	PCP: Privacy-aware Context Profile towards Context- aware Application Development	KAPITSAKI; VENIERIS
2010	Making control loops explicit when architecting self-adaptive systems	HEBIG; GIESE; BECKER
2011	Adapt Cases: Extending Use Cases for Adaptive Systems	LUCKEY et al.
2012	Specifying security requirements of context aware system using UML	ALMUTAIRI; BELLA; ABU-SAMAHA
2012	Extended UML for the Development of Context-Aware Applications	BENSELIM; SERIDI-BOUCHELAGHEM
2012	Extending UML to model Web 2.0-based context-aware applications	HSU
2013	High-quality specification of self-adaptive software systems	LUCKEY; ENGELS
2013	Modelling context-aware RBAC models for mobile business processes	WENZL; STREMBECK
2015	An Extension of UML Activity Diagram to Model the Behaviour of Context-Aware Systems	AL-ALSHUHAI; SIEWE
2016	FAME: A UML-based framework for modeling fuzzy self-adaptive software	HAN et al.
2017	Towards A UML Profile for Context-Awareness Domain	BENSELIM; SERIDI-BOUCHELAGHEM
2017	Heavyweight extension to the UML class diagram metamodel for modeling context aware systems in ubiquitous computing	BOUDJEMLINE et al.

Source: The authors.

supports the aspect-oriented modeling, which allows encapsulating crosscutting concerns, such as context-awareness, into well-localized modules.

Kapitsaki and Venieris (2008) propose a UML profile to specify privacy issues in context-aware systems. The profile enables modeling context items and specifying users information access constraints. Hebig, Giese and Becker (2010) present a UML profile to make explicit the control loops in SaSs architectures. The profile enables designing control loops and their interactions at architectural level.

Luckey et al. (2011) propose a UML profile to formalize and to explicit the adap-

tivity in use case models. The profile, besides the use case diagram, also extends the class and sequence diagrams. Almutairi, Bella and Abu-Samaha (2012) present a UML extension to model the context-aware systems security requirements. The extension customizes the use case diagram to make UML more applicable for the mobility and security functions modeling.

Benselim and Seridi-Bouchelaghem (2012) present a UML profile to specify contextual elements of context-aware applications. The profile allows creating analysis and design models, which separate the context elements from general aspects of the system. Hsu (2012) proposes a UML profile to specify in a conceptual level the context-aware application structural properties. The profile allows creating analysis and design models that consider context-aware aspects in a Web 2.0 architecture.

Luckey and Engels (2013) propose a UML extension to specify the SaSs structural and behavioral concerns. The extension customizes the activity and component diagrams to allow separating the adaptation logic from software features. Wenzl and Strembeck (2013) present a UML extension to model context constraints. The extension customizes the activity diagram to address the context-awareness in the process modeling.

Al-alshuhai and Siewe (2015) propose a UML extension to represent context aspects in the system behavior. The extension customizes the activity diagram to enable the context-aware process modeling. Han et al. (2016) present a UML profile to model static and dynamic aspects in SaSs. The profile extends the use case, class, and sequence diagrams to capture the fuzzy behavior in requirements, analysis, and design models.

Benselim and Seridi-Bouchelaghem (2017) present a UML profile to specify contextual elements in context-aware applications. The profile extends the use case, activity, and sequence diagrams to capture contextual information in ubiquitous environments. Boudjemline et al. (2017) propose a UML extension to model context-aware systems in ubiquitous computing. The extension customizes the class diagram to provide a universal model capable of dealing with the context-awareness concerns.

3.3.1 What modeling issues have motivated UML customization?

SaSs have intrinsic properties organized in a three levels hierarchy, as presented in Figure 2.1. The primitive level has self-awareness and context-awareness properties. The major level has the four autonomic properties: self-configuring, self-healing, self-optimizing, and self-protecting. The general level has self-adaptiveness property (SALEHIE;

TAHVILDARI, 2009). We present in Table 3.3 the SaSs properties addressed in each selected study. We observe that 9 studies address context-awareness property, 1 study address self-configuring property, 1 study address self-protecting property, and 6 studies address self-adaptiveness property.

Table 3.3: SaS properties addressed in the studies.

	Context-awareness	Self-configuring	Self-protecting	Self-adaptiveness
Sheng and Benatallah (2005)	✓			
Ayed, Delanote and Berbers (2007)	✓			
Simons and Wirtz (2007)	✓			
Fuentes, Gamez and Sanchez (2008)		✓		
Kapitsaki and Venieris (2008)			✓	
Hebig, Giese and Becker (2010)				✓
Luckey et al. (2011)				✓
Almutairi, Bella and Abu-Samaha (2012)	✓			
Benselim and Seridi-Bouchelaghem (2012)	✓			
Hsu (2012)	✓			
Luckey and Engels (2013)				✓
Wenzl and Strembeck (2013)	✓			
Al-alshuhai and Siewe (2015)				✓
Han et al. (2016)				✓
Benselim and Seridi-Bouchelaghem (2017)	✓			
Boudjemline et al. (2017)	✓			

Source: The authors.

Models are abstractions of reality (BOOCH; RUMBAUGH; JACOBSON, 2005). They help to cope with complexity, focusing on a determined perspective and hiding non relevant details. During development, several models are created to express aspects of the software in different perspectives (SOMMERVILLE, 2010). We highlight the following models to develop a software: requirements, analysis, design, coding, testing, and deployment. We present in Table 3.4 the models engineering addressed in each selected study. We note that 4 studies address requirements models, 12 studies address analysis models, 13 studies address design models, and 2 studies address coding models.

We present in Figure 3.1 a cross analysis that relates the SaSs properties with engineering models. We emphasize that each study can count more than once. From SaSs properties angle, we notice that the most addressed properties are context-awareness and self-adaptiveness. From engineering models perspective, we observe that the most addressed engineering models are analysis and design models. The cross analysis shows that

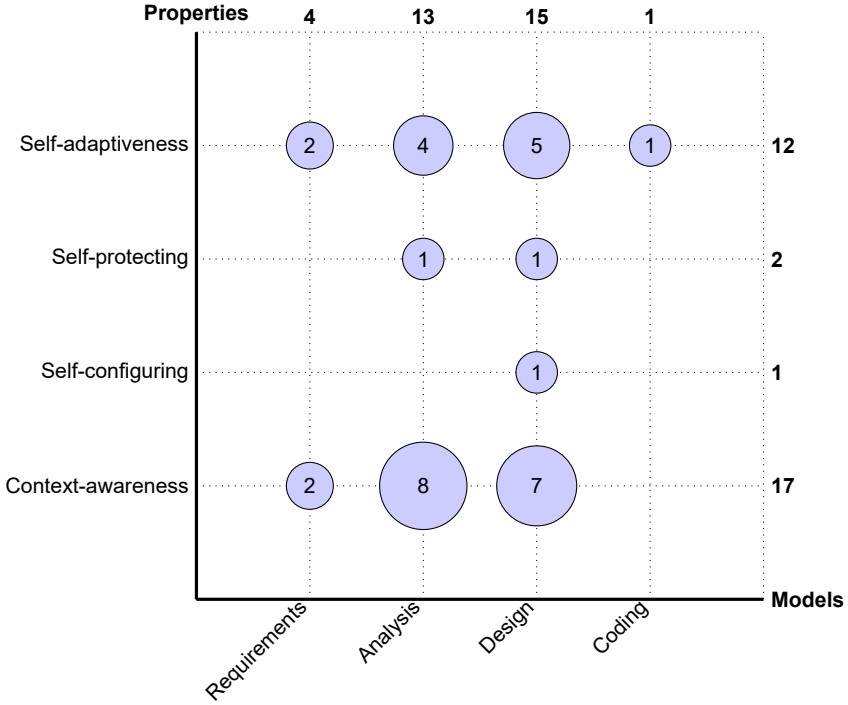
Table 3.4: Engineering models addressed in the studies.

	Requirements	Analysis	Design	Coding
Sheng and Benatallah (2005)		✓	✓	
Ayed, Delanote and Berbers (2007)		✓	✓	
Simons and Wirtz (2007)		✓	✓	
Fuentes, Gamez and Sanchez (2008)			✓	✓
Kapitsaki and Venieris (2008)		✓	✓	
Hebig, Giese and Becker (2010)			✓	
Luckey et al. (2011)	✓	✓	✓	
Almutairi, Bella and Abu-Samaha (2012)	✓			
Benselim and Seridi-Bouchelaghem (2012)		✓	✓	
Hsu (2012)		✓	✓	
Luckey and Engels (2013)			✓	✓
Wenzl and Strembeck (2013)		✓		
Al-alshuhai and Siewe (2015)		✓		
Han et al. (2016)	✓	✓	✓	
Benselim and Seridi-Bouchelaghem (2017)	✓	✓	✓	
Boudjemline et al. (2017)		✓	✓	

Source: The authors.

context-awareness analysis and design, followed by self-adaptiveness design and analysis are widely exploited in the studies.

Figure 3.1: SaSs properties *versus* engineering models.



Source: The authors.

Therefore, according to these SLR results, we notice that issues related to context-awareness and self-adaptiveness analysis and design have motivated the creation of UML-based DSMLs for SaSs. We also highlight that there are gaps in relations between SaSs properties and engineering models. We did not find studies that address issues related to self-healing or self-protecting (from SaSs properties perspective) and testing or deployment (from engineering models perspective). Moreover, there are gaps for self-configuring and self-protecting requirements, self-configuring analysis, and context-awareness, self-configuring, and self-protecting coding.

3.3.2 How UML has been customized to support SaSs modeling?

Diagrams are graphical presentations of a set of elements (BOOCH; RUMBAUGH; JACOBSON, 2005). There is in UML 2.5, a set of fourteen diagrams organized into two groups, as presented in Figure 2.4. Structural diagrams (class, component, composite structure, deployment, object, package, and profile) allow representing the static system structure. Behavioral diagrams (activity, communication, interaction overview, sequence, state machine, timing, and use cases) allow representing the dynamic object behavior in a system (OMG, 2017). We present in Table 3.5 the UML diagrams customized in each selected study. We observe that 10 studies customize class diagram, 3 studies customize component diagram, 4 studies customize use case diagram, 5 studies customize activity diagram, and 5 studies customize sequence diagram.

We explain in Section 2.3 the mechanisms to extend UML: metamodel-based extends the UML metamodel by applying a language of the higher-metalevel; profile-based extends the UML metamodel by applying mechanisms defined by UML itself (STAHL et al., 2006). We present in Table 3.6 the mechanisms used to extend UML in each study. We note that 5 studies report metamodel-based extensions and 11 studies report profile-based extensions.

We present in Figure 3.2 a cross analysis that relates the customized diagrams with extension mechanisms. We note that each study can count more than once. From UML diagrams perspective, we notice that the most customized structural diagram is class diagram and the most customized behavioral diagrams are activity and sequence diagrams. From extension mechanisms angle, we observe that the most used mechanism is profile-based. The cross analysis shows that the class diagram customization through the profile-based mechanism has been widely used.

Table 3.5: UML diagrams customized in the studies.

	Class	Component	Use Case	Activity	Sequence
Sheng and Benatallah (2005)	✓				
Ayed, Delanote and Berbers (2007)	✓				✓
Simons and Wirtz (2007)	✓				
Fuentes, Gamez and Sanchez (2008)	✓	✓		✓	✓
Kapitsaki and Venieris (2008)	✓				
Hebig, Giese and Becker (2010)		✓			
Luckey et al. (2011)	✓		✓		✓
Almutairi, Bella and Abu-Samaha (2012)			✓		
Benselim and Seridi-Bouchelaghem (2012)	✓				
Hsu (2012)	✓				
Luckey and Engels (2013)		✓		✓	
Wenzl and Strembeck (2013)				✓	
Al-alshuhai and Siewe (2015)				✓	
Han et al. (2016)	✓		✓		✓
Benselim and Seridi-Bouchelaghem (2017)			✓	✓	✓
Boudjemline et al. (2017)	✓				

Source: The authors.

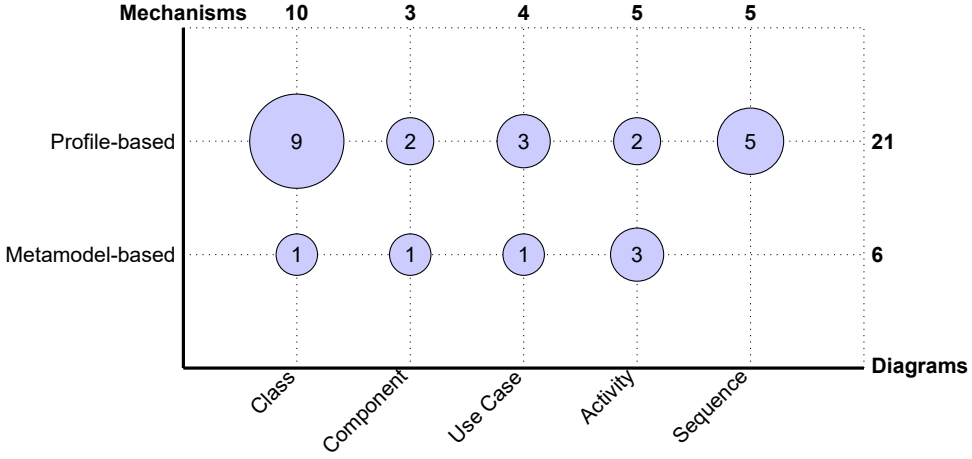
Table 3.6: Mechanisms used to extend UML in the studies.

	Metamodel-based	Profile-based
*Sheng and Benatallah (2005)		✓
Ayed, Delanote and Berbers (2007)		✓
Simons and Wirtz (2007)		✓
Fuentes, Gamez and Sanchez (2008)		✓
Kapitsaki and Venieris (2008)		✓
Hebig, Giese and Becker (2010)		✓
Luckey et al. (2011)		✓
Almutairi, Bella and Abu-Samaha (2012)	✓	
Benselim and Seridi-Bouchelaghem (2012)		✓
Hsu (2012)		✓
Luckey and Engels (2013)	✓	
Wenzl and Strembeck (2013)	✓	
Al-alshuhai and Siewe (2015)	✓	
Han et al. (2016)		✓
Benselim and Seridi-Bouchelaghem (2017)		✓
Boudjemline et al. (2017)	✓	

* Defined according to UML 1.x.

Source: The authors.

Figure 3.2: Customized diagrams *versus* extension mechanisms.



Source: The authors.

Therefore, according to these SLR results, we notice that the profile-based mechanism has been used to customize the class diagram, providing proper support to SaSs modeling. We point out that none of the studies propose customizations for composite structure, deployment, object, or package structural diagrams and communication, interaction overview, state machine, or timing behavioral diagrams.

3.4 Threats to Validity

A threat to SLR validity is that the searching process may not find some relevant studies. We mitigated this by applying the snowballing technique, which had its effectiveness demonstrated by Jalali and Wohlin (2012). Other threat is that the selection process may produce false-positives and false-negatives. We mitigated this by ensuring that each paper was screened by at least two researchers. Another threat is that conceptual background and interpretation capability may affect researchers decisions in data extraction step. We mitigate this by applying Delphi method (LINSTONE; TUROFF, 1975) to obtain consensus about the extracted data.

3.5 Chapter Lessons

In this chapter, we report the results of a SLR that aimed to investigate how UML has been customized to create DSMLs that provide proper support for SaSs modeling. The chapter main points are:

- the most addressed SaSs properties are context-awareness and self-adaptiveness;
- the most addressed engineering models are analysis and design models;
- the most customized structural diagram is the class diagram;
- the most customized behavioral diagrams are activity and sequence diagrams;
- the most used extension mechanism is profile-based;
- context-awareness and self-adaptiveness analysis and design are widely exploited;
- profile-based class diagram extensions has been widely used;
- there are SaSs properties not covered by any of the selected studies;
- none of the studies integrate the context-awareness and self-adaptiveness modeling.

In next chapter, we propose a UML-based DSML for SaSs conceptual modeling called SaSML that introduces a new modeling element called Adaptive Behavior. Adaptive Behavior integrates the context-awareness and self-adaptiveness modeling, synthesizing a SaSs modeling schema without lose expressiveness.

4 UML-BASED DSML FOR SAS CONCEPTUAL MODELING

In this chapter, we propose SaSML, a UML extension that introduces a new modeling element called Adaptive Behavior, which is able to synthesize a SaSs modeling schema without losing expressiveness. We define in Section 4.1 a SaSs modeling schema based on design patterns. We specify in Section 4.2 a UML extension for SaSs conceptual modeling. We illustrate in Section 4.3 the UML extension usage in a real scenario. Finally, we summarize in Section 4.4 the chapter lessons.

4.1 SaSs Conceptual Modeling Schema

We advocate that SaSs modeling complexity can be minimized by using a DSML, therefore, we propose SaSML, a UML-based DSML for SaSs conceptual modeling. Besides capturing higher-level abstractions and constraints, we would also like to capture the essence of a modeling strategy for SaSs conceptual modeling. Thus, before specifying SaSML, we proposed a SaSs conceptual modeling schema, i.e., an understandable representation of a SaSs conceptual modeling idea in the form of a model. Differently from the studies presented in Chapter 3, our intention was to identify the essential concepts related to SaSs domains, establishing the necessary communication to support self-adaptiveness.

For scope definition purposes, we took as basis for our work the SaSs definition presented in Section 2.1). In other words, our SaSs modeling schema supports adaptation based on behavior (re)organization in response to contextual changes. Thus, we started the schema modeling by analyzing the SaSs definition to identify the essential abstractions to be expressed in a conceptual model. A SaS can autonomously decide how to adapt its behavior at runtime in response to contextual changes (ANDERSSON et al., 2009; BRUN et al., 2009; CHENG et al., 2009). As a result, we extracted two main abstractions:

- **context**, which is needed to monitor the environment situations and to trigger behavior adaptations;
- **behavior**, which is needed to define the system behaviors and to implement adaptation mechanisms.

Next, we analyzed the context and behavior definitions to identify the requirements for SaSs conceptual models. Context is any information that can be used to define the situation of an entity (DEY, 2001). Situation is a specific state of affairs (COSTA et al., 2006). Entity is anything considered relevant to a system, including the users and the

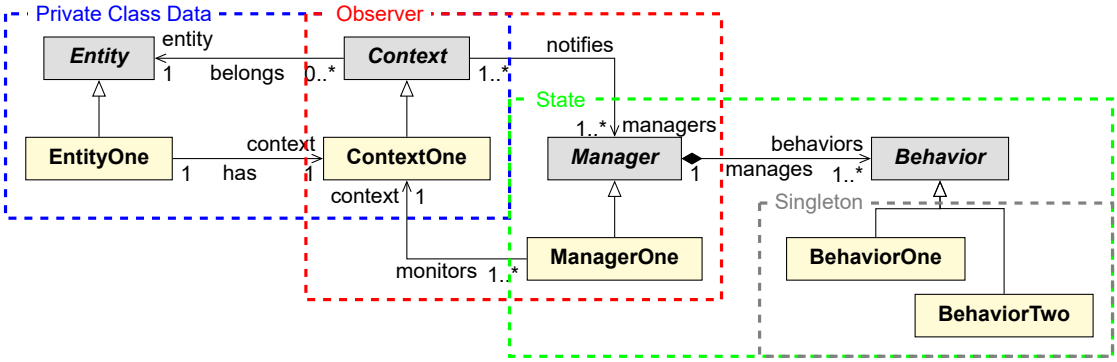
system itself (DEY, 2001). Behavior is a set of actions of an individual in response to stimuli provided by the environment in which one lives (MINTON; KHALE, 2014).

From this analysis process, we enunciated the following requirements for SaSs conceptual models:

1. a model should define the domain entities relevant to the system;
2. a model should define the information that defines the entities situation;
3. a model should define the mechanisms to manage the behavior adaptation process;
4. a model should define the behavior that the system may have at runtime;
5. a model should define the monitoring mechanisms to trigger the behavior adaptations.

The analysis these requirements revealed to us the possibility to based our SaSs modeling schema on design patterns. As seen in Section 2.4, design patterns are tested and proven reusable solutions to common problems. Their usage contribute to a more flexible and reusable SaSs modeling schema. Thus, we searched for proper design patterns to model each one of the proposed requirements. We took as reference the catalog presented by Shvets (2018), which extends the catalog presented by Gamma et al. (1994). We selected design patterns according to their capabilities to solve the modeling problems related to requirements. As an outcome, we applied four design patters to create the modeling schema presented in Figure 4.1.

Figure 4.1: SaSs conceptual modeling schema based on design patterns.



Source: The authors.

We applied the **Private Class Data** pattern to address the first and second requirements. It allowed us to encapsulate contextual information in a data class, associating it to its respective entity class. Hence, it is possible to keep the entity cohesion and to define sets of related contextual information. The Private Class Data pattern usage can be seen in the left side of Figure 4.1. The relevant domain entities to the system should be modeled as main classes without contextual information. The contextual information should

be modeled as data classes. Each entity may have one or many contexts, but a context belongs to only one entity.

We applied the **State** pattern to address the third and fourth requirements. It allowed us to define the system behaviors as a set of states and to create an adaptation mechanism based on a state machine. The State pattern usage can be seen in the right side of Figure 4.1. The mechanism to manage the adaptation process might be modeled as a wrapper class, which knows all possible system behaviors. The behaviors that the system may have at runtime might be modeled as a polymorphic hierarchy. The **Singleton** pattern ensures a single instance of each possible system behavior.

We applied the **Observer** pattern to address the fifth requirement. It allowed us to define a dynamic context monitoring mechanism to trigger the adaptation process at runtime. The Observer pattern usage can be seen in the center of Figure 4.1. Each Context class might be modeled as a subject class, registering at least one Manager class. Manager classes might be modeled as observer classes, keeping a reference to all relevant contexts.

Next, we illustrate the modeling schema usage by modeling a blood pressure monitoring requirement for health care system. The requirement presented in Table 4.1 is written in RELAX and it specifies the system desired behavior for some possible situations.

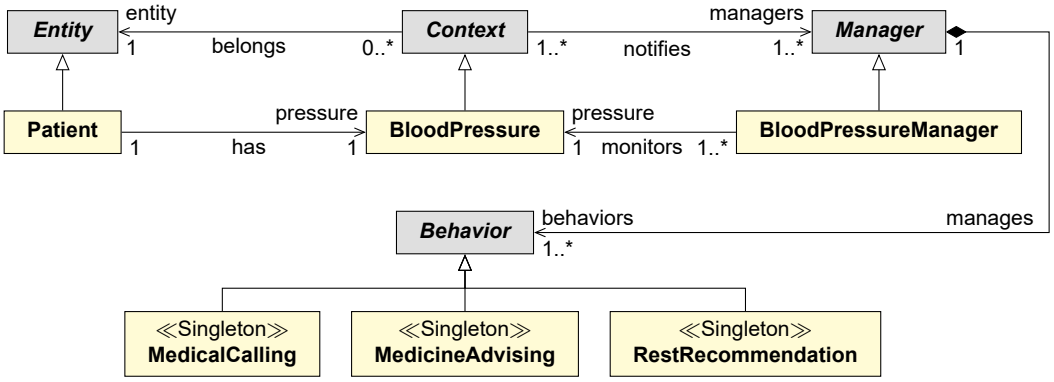
Table 4.1: Blood pressure monitoring requirement for health care.

Req-2	The system SHALL monitor the blood pressure IN every hour to ensure the patient wellness. It MAY call the medical support whenever the blood pressure is very low or very high, OR it MAY advise the medicine intake whenever the blood pressure is high, OR it MAY recommend the rest whenever the blood pressure is low.
	ENV: Patient blood pressure. MON: Sphygmomanometer. REL: Sphygmomanometer provides Patient blood pressure. DEP: None.

Source: The authors.

The analysis process of this requirement resulted in the conceptual model presented in Figure 4.2, which was made based on the modeling schema. Patient is an entity that has BloodPressure as contextual information. BloodPressureManager is the adaptation mechanism wrapper and it is registered in BloodPressure context as an observer to be notified whenever the situation changes. Additionally, BloodPressureManager knows BloodPressure situation and the set of system behaviors. MedicalCalling, MedicineAdvising, and RestRecommendation are system possible behaviors.

Figure 4.2: Blood pressure monitoring requirement conceptual model.



Source: The authors.

Now, we illustrate the modeling schema usage by modeling a thermal comfort manager requirement for a smart office system, which is specified in Table 4.2.

Table 4.2: Thermal comfort manager requirement for smart offices.

Req-3	The system SHALL manage the office temperature AS EARLY AS POSSIBLE to ensure the environment thermal comfort. It MAY turn on the heating whenever the office inside and outside temperatures are low, OR it MAY open the ventilation whenever the office outside temperature is mild, OR it MAY turn on the refrigeration whenever the office inside and outside temperatures are high.
	ENV: Office inside and outside temperature. MON: Thermostat. REL: Thermostat provides Office inside and outside temperature. DEP: None.

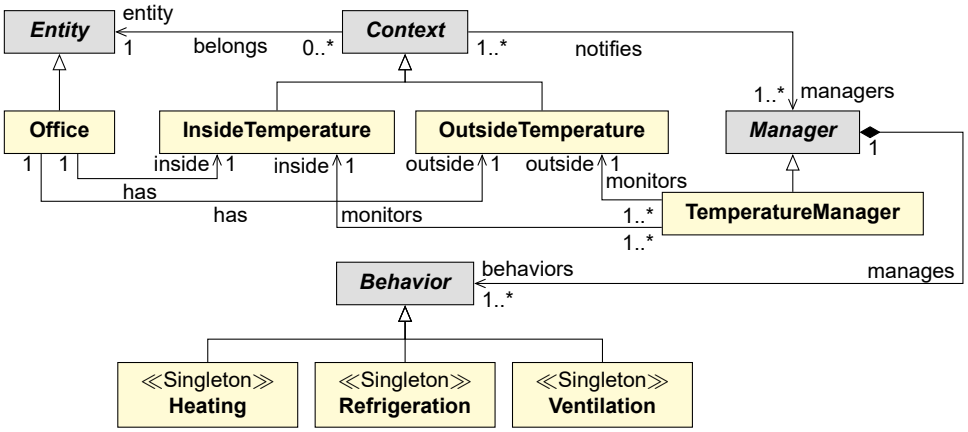
Source: The authors.

We present in Figure 4.3 the conceptual model obtained from modeling schema application. Office is an entity that has InsideTemperature and OutsideTemperature as contextual information. TemperatureManager is the adaptation mechanism wrapper and it is registered in InsideTemperature and OutsideTemperature contexts as an observer to be notified whenever the situation changes. Additionally, TemperatureManager knows InsideTemperature and OutsideTemperature situation and the set of system behaviors. Heating, Refrigeration, and Ventilation are system possible behaviors.

4.2 UML Extension for SaSs Conceptual Modeling

The modeling schema presented previously captures the SaSs higher-level abstractions and constraints, and the necessary communication to support self-adaptiveness. The design patterns used to design it contribute to schema reusability and flexibility. To apply the schema in other scenarios, we need to identify the related entities, contexts, managers,

Figure 4.3: Thermal comfort manager requirement conceptual model.



Source: The authors.

and behaviors, and to specify them as the concrete classes presented in Figure 4.1. However, we notice that complex scenarios may produce complex models, which may impact in conceptual model comprehensibility.

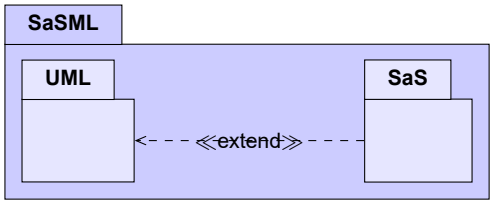
The model quality is related to its capability in providing for all stakeholders the same understanding of what it represents (HULL; JACKSON; DICK, 2011). Therefore, in a software engineering perspective, besides expressing the real-world abstractions, a model should do this in an understandable way. Hence, we propose a UML-based DSML called **SaSML** that extends UML introducing a new modeling element for SaSs conceptual modeling. It is important to notice that the SaSML first version focuses on conceptual modeling, which does not limit its potential in terms of SaSs modeling support. Further, we point out how SaSML may be evolved to support SaSs behavioral modeling.

UML may be extended by modifying the metamodel, which allows modifying the UML syntax and semantics, or by creating a profile, which allows customizing UML to a particular domain (STAHL et al., 2006). Once we intended to introduce a new modeling element into UML, we extended it through modifying the metamodel, i.e., we worked directly over its metamodel. We extended the version 2.5.1 of the UML Abstract Syntax Metamodel[1]. As can be seen in Figure 4.4, SaSML is composed of two packages. UML package contains the UML metamodel without modifications, i.e., the UML elements syntax and semantics are preserved. Hence, all UML valid constructions also are SaSML valid constructions. SaS package contains the UML metamodel extensions needed to specify modeling elements oriented to SaSs modeling.

We proposed in the SaSML first version a new modeling element called **Adaptive Behavior**. Adaptive Behavior is a wrapper that encapsulates the SaSs modeling schema,

[1]Available at: <https://www.omg.org/spec/UML/20161101/UML.xmi>.

Figure 4.4: SaSML packages overview.



Source: The authors.

exposing only what needs to be defined at modeling time. We specified Adaptive Behavior by defining its syntax, semantics, and pragmatics. Syntax is the form of a language, semantics is the meaning of a language, and pragmatics is the implementation of a language (TURBAK; GIFFORD; SHELDON, 2008).

4.2.1 Adaptive Behavior Syntax

We started the Adaptive Behavior syntax specification by defining its abstract syntax. Abstract syntax establishes the syntax structures without indicating how it appears (TURBAK; GIFFORD; SHELDON, 2008). In a UML extension, the abstract syntax can be specified through a metamodel (SHENG; BENATALLAH, 2005). During the element metamodeling, we attended to some important constraints. The Adaptive Behavior modeling element should:

- not overwrite the UML elements to ensure a full UML reuse;
- be used together with the UML elements related to class diagram.

A challenge to customize UML by metamodel modifying is to identify which metaclasses should be extended. To solve this, we applied a top-down analysis strategy, where we investigated all metaclasses specializations starting from the root metaclass. When investigating each metaclass, we analyzed its specification to decide whether it applies or does not apply to Adaptive Behavior. We took as analysis criteria the element purpose, i.e, to be a wrapper for the SaSs modeling schema. In positive case, we selected the metaclass, otherwise, we discarded the metaclass and its subsequent specializations. Our intention was to select the deepest metaclasses in the hierarchy, generic enough to describe Adaptive Behavior.

We started the UML metamodel analysis from the hierarchy root. Element metaclass generalizes all model constituent elements (OMG, 2017). For this reason, **Element** metaclass was selected.

Next, we analyzed all Element metaclass specializations. Element metaclass has 15 specializations: Comment, MultiplicityElement, NamedElement, ParameterableElement, Relationship, TemplateableElement, TemplateParameter, TemplateParameterSubstitution, TemplateSignature, ExceptionHandler, Image, Slot, Clause, LinkEndData, and QualifierValue (OMG, 2017). We selected **NamedElement**, **ParameterableElement**, and **TemplateableElement** metaclasses because Adaptive Behavior has a name, it may be a formal template parameter, and it can be defined as a template.

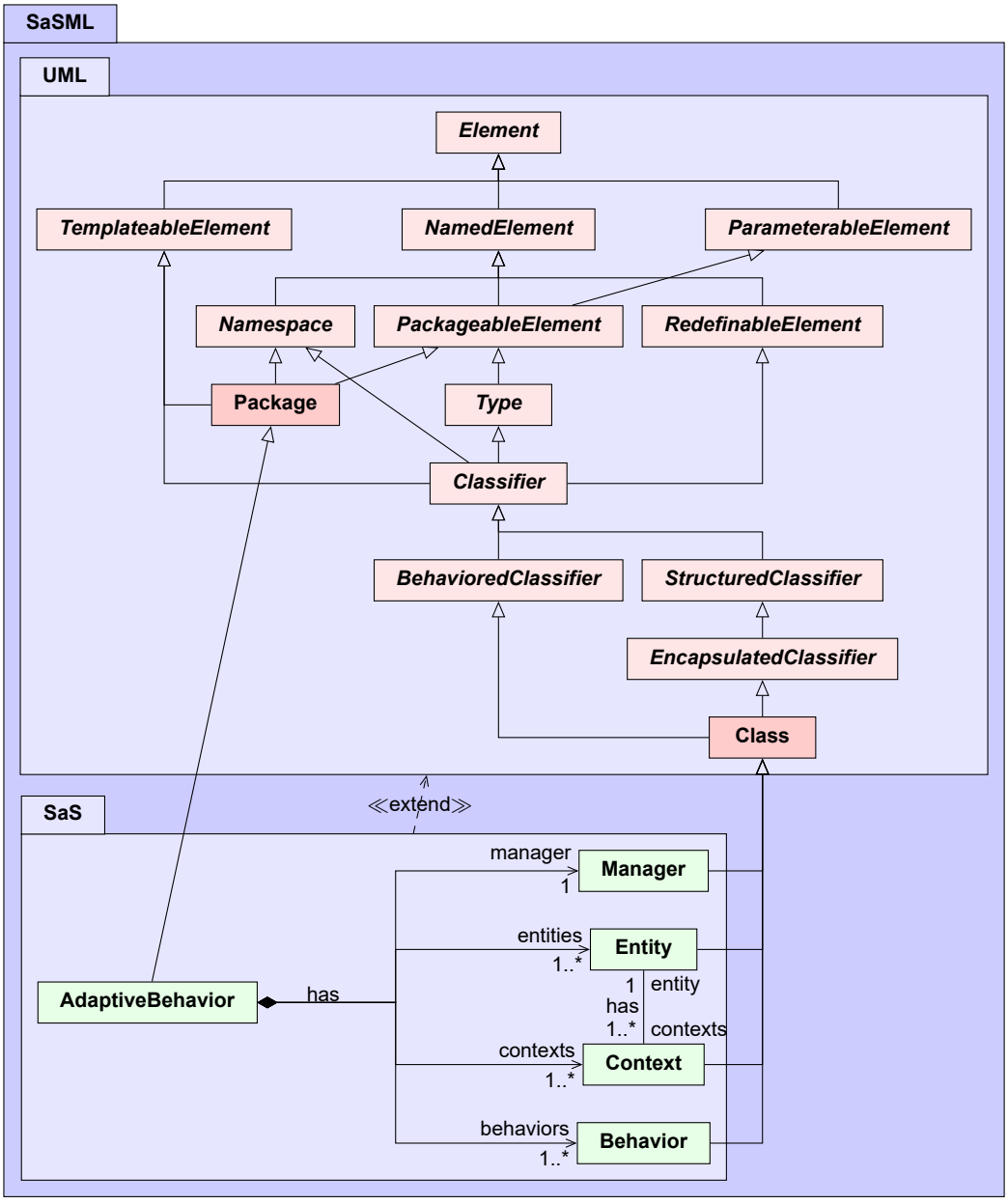
After, we analyzed all NamedElement, ParameterableElement, and TemplateableElement metaclasses specializations. NamedElement metaclass has 18 specializations: Namespace, PackageableElement, TypedElement, ActivityGroup, Trigger, Extend, Include, CollaborationUse, Vertex, GeneralOrdering, InteractionFragment, Lifeline, Message, MessageEnd, DeployedArtifact, DeploymentTarget, ParameterSet, and RedefinableElement. ParameterableElement metaclass has 3 specializations: PackageableElement, ConnectableElement, and Operation. TemplateableElement metaclass has 4 specializations: StringExpression, Package, Classifier, and Operation (OMG, 2017). We selected **Namespace**, **PackageableElement**, and **Package** metaclasses because Adaptive Behavior may be used to group elements, it may be owned by a package, and it provides a namespace for the grouped elements.

Lastly, we analyzed all Namespace, PackageableElement, and Package metaclasses specializations. Namespace metaclass has 8 specializations: Region, State, Transition, Package, InteractionOperand, BehavioralFeature, Classifier, and StructuredActivityNode. PackageableElement metaclass has 10 specializations: Constraint, Dependency, Type, Event, Observation, ValueSpecification, Package, InformationFlow, GeneralizationSet, and InstanceSpecification. Package metaclass has 2 specializations: Model and Profile (OMG, 2017). We selected **Package** metaclass again. As the metaclass has already been analyzed and selected, we finished the UML metamodel analysis. Thus, we considered that Package is the deepest metaclass in the hierarchy, generic enough to describe Adaptive Behavior.

We present in Figure 4.5 the Adaptive Behavior metamodel, where all the metaclasses hierarchy can be seen. UML package contains the UML metaclasses extended to define Adaptive Behavior. Besides the Package metaclass, we also make explicit the Class metaclass. We extended the Class metaclass to define the Adaptive Behavior structures. SaS package contains the Adaptive Behavior definition. AdaptiveBehavior is a concrete metaclass that represents the modeling element itself and it extends the Package

concrete metaclass. Manager, Entity, Context, and Behavior are concrete metaclasses that represent the SaSs modeling schema and they extend the Class concrete metaclass. The AdaptiveBehavior metaclass has an one-to-one composite association with Entity and it has one-to-many composite associations with Entity, Context, and Behavior metaclasses. Additionally, the Entity metaclass has an one-to-many association with Context metaclass.

Figure 4.5: Adaptive Behavior metamodel.



Source: The authors.

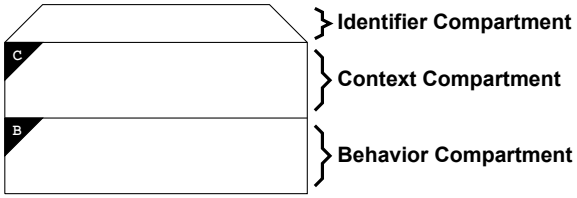
Once the abstract syntax was defined, we specified the Adaptive Behavior concrete syntax. Concrete syntax establishes the forms that can be used to express the abstract syntax (TURBAK; GIFFORD; SHELDON, 2008). In a UML extension the concrete syntax can be specified through a notation (SHENG; BENATALLAH, 2005). During the notation

designing, we attended to some important constraints. The Adaptive Behavior modeling element should have:

- a different notation to not be confused with the UML notation;
- a simple notation to allow being drawn by freehand.

We propose the notation presented in Figure 4.6 for Adaptive Behavior. It is composed of three distinct compartments. The first compartment (trapezoid at the top) is to define the element identifier. The second compartment (rectangle at the middle with "C" tag) is to define the contextual information. The third compartment (rectangle at the bottom with "B" tag) is to define the behavioral information.

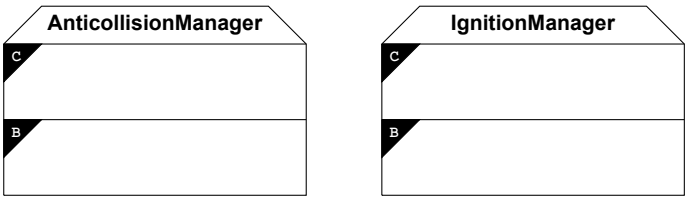
Figure 4.6: Adaptive Behavior notation.



Source: The authors.

The element identifier (Identifier Compartment) must represent the adaptive behavior general purpose. It can be simple (one word) or compound (more than one word), but it has to be written clearly and objectively. We present in Figure 4.7 some identifiers examples for an autonomous car. AnticollisionManager represents the behaviors adaptations required to avoid any collisions. IgnitionManager represents the behaviors adaptations required to manage engine operation.

Figure 4.7: Adaptive Behavior identifiers examples.

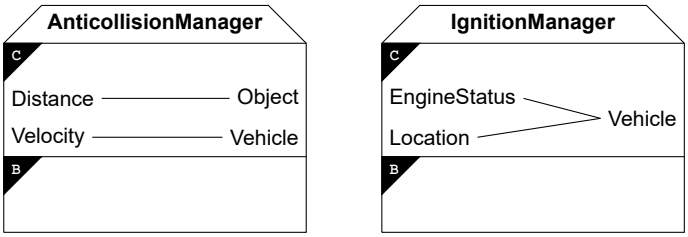


Source: The authors.

The contextual information (Context Compartment) is a pair composed of a Context and an Entity. A Context must be linked to only one Entity. An Entity must be linked to at least one Context. Context and Entity identifiers must be a simple or compound representative expression. We present in Figure 4.8 some contexts examples for an autonomous car. In AnticollisionManager, Velocity is linked to Vehicle and Distance is linked to Object. This contextual information must be monitored to avoid collisions. In IgnitionManager, Location, and EngineStatus are linked to Vehicle. This contextual

information must be monitored to manage the engine.

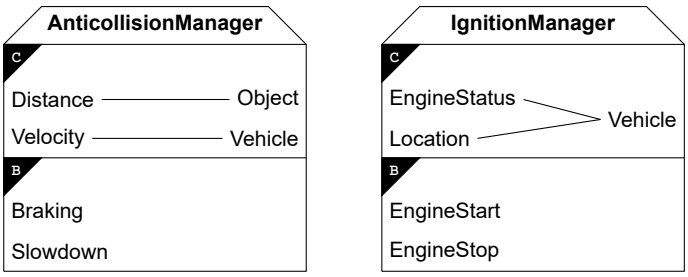
Figure 4.8: Adaptive Behavior contexts examples.



Source: The authors.

The behavioral information (Behavior Compartment) defines the alternative behaviors that a system may have. Each item represents a possible behavior related to adaptation purpose, i.e., one of the behaviors that may be taken at runtime in response to entities context changes. We present in Figure 4.9 some behaviors examples for an autonomous car. In AnticollisionManager, the system may slow down (Slowdown) or brake (Braking) to avoid collisions. In IgnitionManager, the system may start (EngineStart) or stop (EngineStop) the engine.

Figure 4.9: Adaptive Behavior behaviors examples.



Source: The authors.

4.2.2 Adaptive Behavior Semantics

As opposed to Object Management Group (OMG), which presents the UML semantics informally (OMG, 2017), we used a math notation to explain the Adaptive Behavior semantics. Hence, we ensured a correct interpretation of the Adaptive Behavior structures.

Let U be an nonempty finite set of classes, where $U = \{cls \mid cls$ is a SaSs conceptual model class$\}$. Let E, C, M, and B be disjoint nonempty subsets of U, where:

- $E = \{e \mid e$ is an Entity subclass$\}$;
- $C = \{c \mid c$ is a Context subclass$\}$;
- $M = \{m \mid m$ is a Manager subclass$\}$;

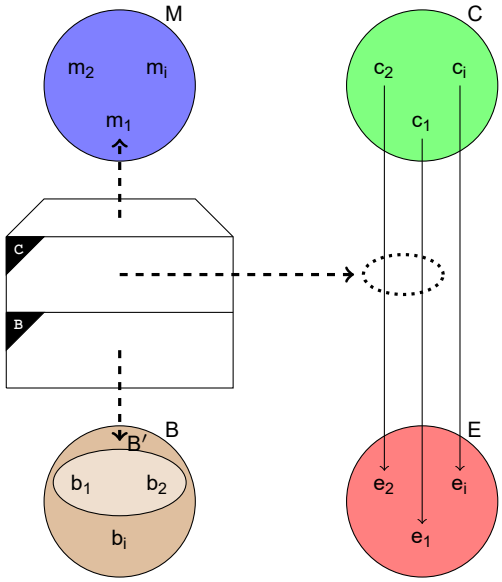
- $B = \{b \mid b \text{ is a Behavior subclass}\}$.

Thus, we defined the Adaptive Behavior modeling element as a triple (m_i, f, B'), where:

- $m_i \in M$ is a Manager subclass;
- f: $C \rightarrow E$ maps a Context subclass to an Entity subclass;
- $B' \subseteq B$ is a nonempty finite subset of Behavior.

We illustrate in Figure 4.10 the relation between the Adaptive Behavior semantics and notation. The Adaptive Behavior identifier compartment displays the Manager class m_i, the context compartment displays the surjective function f, and the behavior compartment displays the subset of Behavior B'. The Venn diagrams present abstract examples for the sets E, C, M, and B.

Figure 4.10: Relation between Adaptive Behavior notation and semantics.



Source: The authors.

4.2.3 Adaptive Behavior Pragmatics

We applied the Façade pattern to define the Adaptive Behavior pragmatics. Each Adaptive Behavior instance encapsulates the subsystem responsible for modeling a specific adaptive behavior, which is modeled according to the SaSs modeling schema. The transformation process must be performed according to the following rules:

1. the identifier compartment entry becomes a concrete subclass of Manager abstract class;
2. for each context compartment entry:

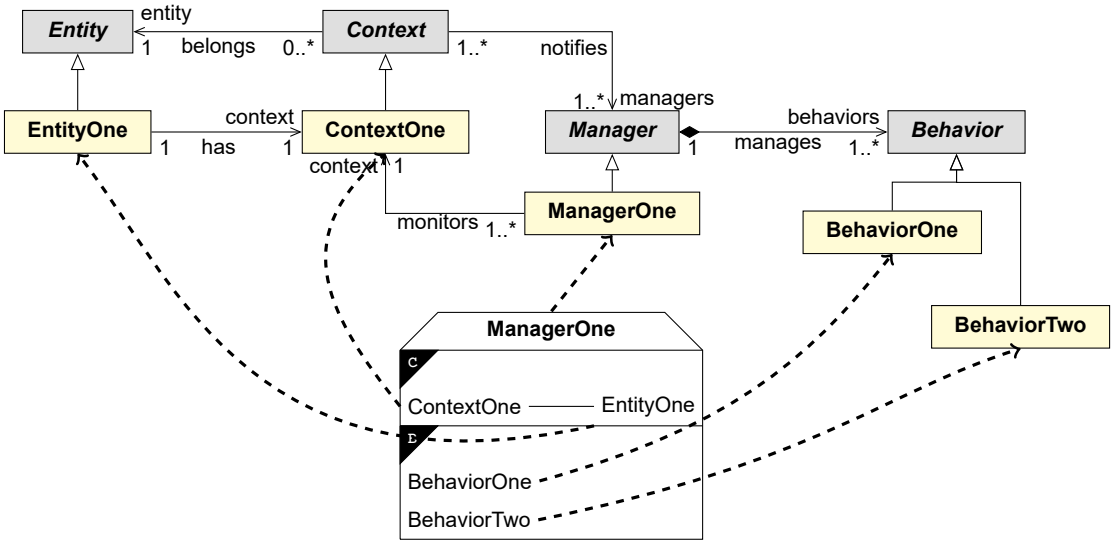
(a) the first entry becomes a concrete subclass of Context abstract class;

(b) the second entry becomes a concrete subclass of Entity abstract;

3. each behavior compartment entry becomes a Singleton concrete subclass of Behavior abstract class;
4. the Manager concrete subclass must be linked to each Context concrete subclass by one-to-many simple association;
5. the Manager concrete subclass is defined as the subsystem interface.

We illustrate in Figure 4.11 the application of the Adaptive Behavior implementation rules. Each Adaptive Behavior entry becomes a concrete class in the SaSs conceptual model. Each concrete class extends a specific abstract class of the SaSs modeling schema.

Figure 4.11: Adaptive Behavior mapping into SaSs modeling schema.



Source: The authors.

4.3 Real Scenario Application

Devincenzi et al. (2017) proposed a SaS called TutorApp to motivate the students from courses based on Problem-Based Learning (PBL) to develop their skills and abilities. TutorApp is composed of mechanisms called triggers, which are able to recommend specific Learning Objects (LO) according to the students needs. Each trigger uses a specific set of contextual information to monitor the students situation and provide proper LOs. We present in Table 4.3 a requirements specification according to features proposed by Devincenzi et al. (2017).

Table 4.3: TutorApp requirements specification.

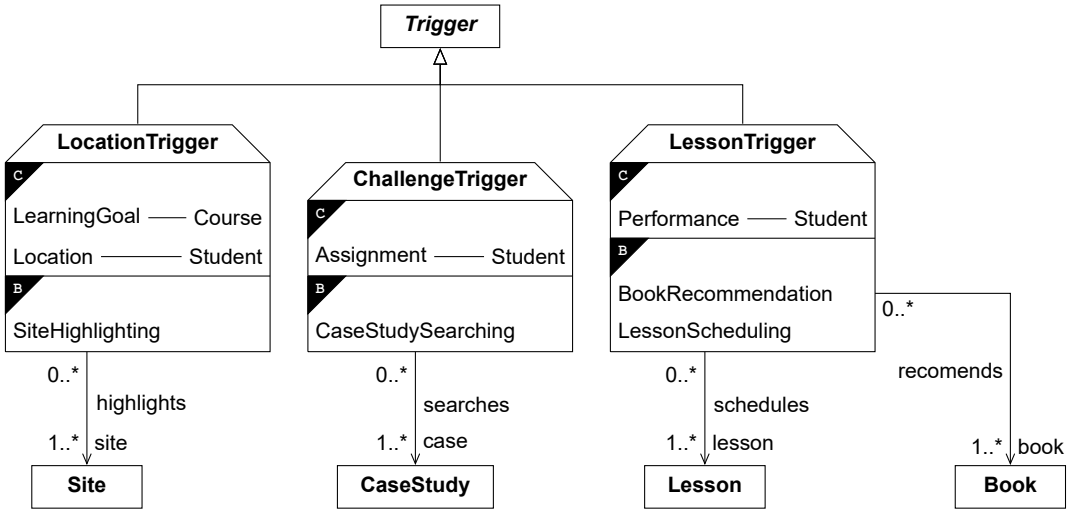
Req-4	The location-based trigger SHALL notify the students AS EARLY AS POSSIBLE about available sites around them related to course learning goals. It MAY provide site information whenever they are at a distance of AS CLOSE AS POSSIBLE TO 1km.
	ENV: Course learning goals and Students location. MON: Courses tracker and Students tracker. REL: Courses tracker provides Course learning goals and Students tracker provides Students location. DEP: None.
Req-5	The challenge-based trigger SHALL provide the students AS EARLY AS POSSIBLE specialized support to solve issues proposed to them. It MAY search for related case studies whenever a new issue is proposed.
	ENV: Students assignments. MON: Students tracker. REL: Students tracker provides Students assignments. DEP: None.
Req-6	The lesson-based trigger SHALL provide the students AS EARLY AS POSSIBLE additional courseware to eliminate some conceptual gaps. It MAY reserve books in the library whenever the students have regular performance, OR it MAY schedule extra lessons whenever the students have low performance.
	ENV: Students performance. MON: Student tracker. REL: Students tracker provides Students performance. DEP: None.

Source: Adapted from Devincenzi et al. (2017).

We present in Figure 4.12 the TutorApp conceptual model specified in SaSML. We create an Adaptive Behavior for each type of trigger proposed by TutorApp. LocationTrigger models the Req-4 behavior, ChallengeTrigger models the Req-5 behavior, and LessonTrigger models the Req-6 behavior.

LocationTrigger has two contextual information entries (LearningGoal linked to Course and Location linked to Student) and one behavioral possibility entry (SiteHighlighting). ChallengeTrigger has one contextual information entry (Assignment linked to Student) and one behavioral possibility entry (CaseStudySearching). LessonTrigger has one contextual information entry (Performance linked to Student) and two behavioral possibilities entries (BookRecomendation and LessonScheduling). In addition, we specified four supporting classes (Site, CaseStudy, Lesson, and Book), which are associated to a specific behavioral possibility (SiteHighlighting, CaseStudySearching, LessonScheduling, and BookRecomendation, respectively).

Figure 4.12: TutorApp conceptual model specified in SaSML.

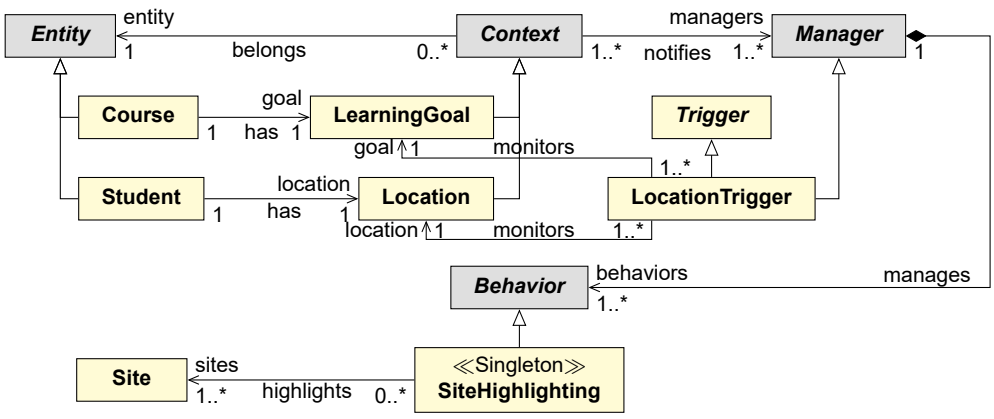


Source: The authors.

We present next the TutorApp conceptual model specified in UML. These models were built by applying the transformation rules proposed in Section 4.2.3. For each Adaptive Behavior modeling element in Figure 4.12, we create an equivalent conceptual model. We present in Figure 4.13 the LocationTrigger transformation, in Figure 4.14 the ChallengeTrigger transformation, and in Figure 4.15 the LessonTrigger transformation.

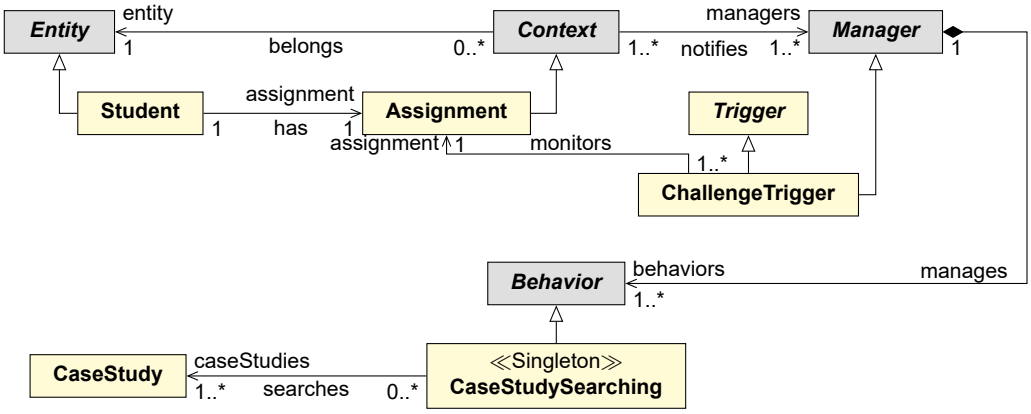
When comparing the two models versions, SaSML and UML, we noted that each occurrence of Adaptive Behavior is equivalent to a set of UML elements. Besides this, Adaptive Behavior encapsulates the SaSs modeling schema presented in Figure 4.1. Therefore, SaSML models are visually more synthetic than UML models, contributing with the model comprehensibility.

Figure 4.13: TutorApp Req-4 conceptual model specified in UML.



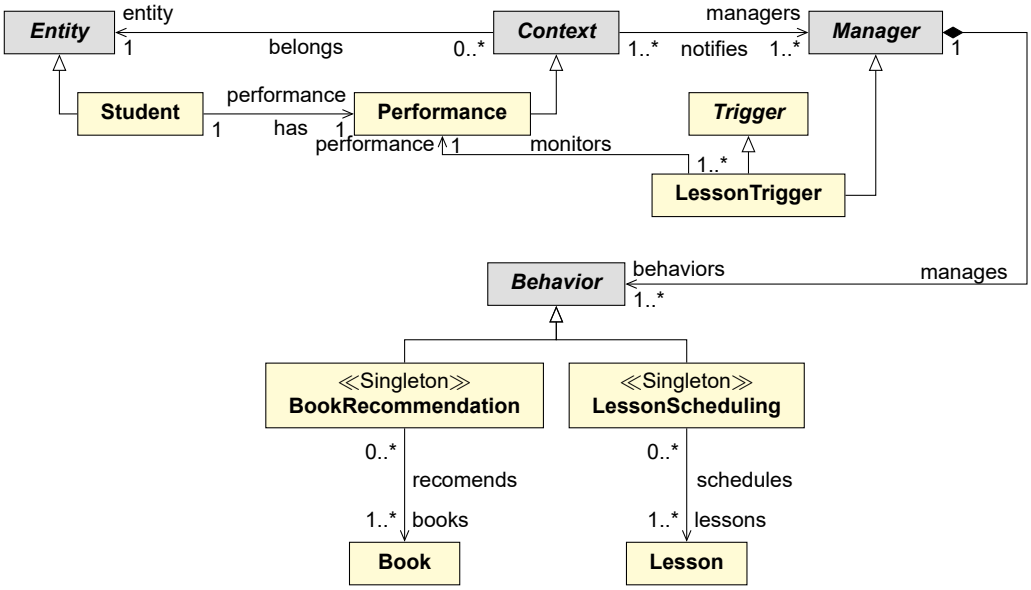
Source: The authors.

Figure 4.14: TutorApp Req-5 conceptual model specified in UML.



Source: The authors.

Figure 4.15: TutorApp Req-6 conceptual model specified in UML.



Source: The authors.

4.4 Chapter Lessons

In this chapter, we propose a UML-based DSML called SaSML that introduces a new modeling element called Adaptive Behavior able to synthesize a SaSs modeling schema based on design patterns. The chapter main points are:

- SaSs modeling schema applies a set of design patterns to capture the SaSs higher-level abstractions;
- the design patterns usage contribute to a more flexible and reusable SaSs modeling schema;
- SaSML was created by modifying the UML metamodel, ensuring the compatibility with UML standard elements;

- Adaptive Behavior synthesizes a SaSs modeling schema, exposing only what needs to be defined at modeling time;
- SaSML models are visually more synthetic than UML models, contributing with the model comprehensibility.

In next chapter, we report the results of empirical techniques used to evaluate the SaSs modeling schema and the Adaptive Behavior modeling element. The SaSs modeling schema correctness and completeness was evaluated by running focus group sessions. Adaptive Behavior modeling element expressiveness and effectiveness was evaluated by running an experiment with subjects.

5 SASML EMPIRICAL EVALUATION

In this chapter, we report the empirical techniques used to evaluate the SaSs modeling schema correctness and completeness, and the Adaptive Behavior modeling element expressiveness and effectiveness. We report in Section 5.1 the SaSs modeling schema evaluation by running focus group sessions. We report in Section 5.2 the Adaptive Behavior evaluation by running an experiment with subjects. Finally, we summarize in Section 5.3 the chapter lessons.

5.1 Focus Group Sessions

We applied the focus group technique (BLOOR et al., 2001) to perform a SaSML preliminary evaluation. The evaluation goal was to verify the SaSs modeling schema correctness and completeness. The feedback obtained from focus group sessions guided our decisions and aided us to produce the SaS modeling schema presented in Figure 4.1.

We involved in this evaluation process a set of subjects, which were responsible for analyzing, discussing, and reporting the collective findings about the proposed models. To facilitate good discussions and to obtain useful feedback, we invited researchers with experience in modeling, metamodeling, UML, or SaSs to compose the set of subjects.

The focus group sessions were driven by a moderator, which was responsible for instigating and limiting the subjects discussions. In each focus group session, the moderator ran the following steps:

1. **opening**, the moderator introduced general information about the session;
2. **background**, the moderator provided a conceptual foundation about SaSs;
3. **discussion**, the subjects evaluated the models and provided feedback;
4. **closure**, the moderator collected the final opinion about the models.

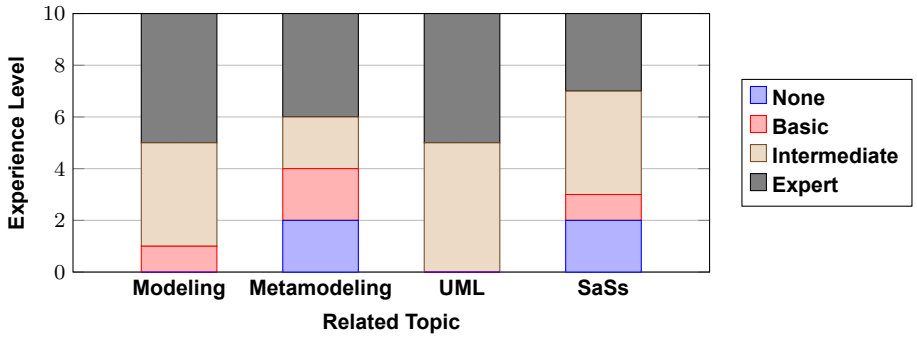
The main data source was a set of questions collectively answered by the subjects in the discussion step. In addition, we used alternative ways to collect data during focus group sessions, such as: audio, video, photos, annotations, and forms. All collected data was analyzed to identify the proposed models improvement opportunities.

5.1.1 Focus Group Execution

We performed 3 focus group sessions between December 2017 and April 2018. The sessions ran in a reserved room and lasted approximately 3 hours. Each session produced a set of findings about models correctness and completeness. After each session, we discussed the findings and promoted improvements in the SaSs modeling schema. The resulting model was the input to the next focus group session.

We note that each session engaged a different set of subjects, potentiating the discussions and feedback diversity. We involved a total of 10 subjects, being 7 professors from 2 universities, 2 graduate students, and 1 practitioner. We summarized in Figure 5.1 the subjects profiles, which was obtained through a questionnaire that asked about their experience in each related topic.

Figure 5.1: Focus group subjects profile.



Source: The authors.

Regarding modeling experience, 1 subject answered Basic level, 4 subjects answered Intermediate level, and 5 subjects answered Expert level. About metamodeling experience, 2 subjects stated no experience, 2 subjects answered Basic level, 2 subjects answered Intermediate level, and 4 subjects answered Expert level. Concerning UML experience, 5 subjects answered Intermediate level and 5 subjects answered Expert level. Lastly, about SaSs experience, 2 subjects stated no experience, 1 subject answered Basic level, 4 subjects answered Intermediate level, and 3 subjects answered Expert level.

5.1.1.1 First Session Execution

The first focus group session was performed on December 14, 2017, at 09:00 AM to 12:00 PM, in the Federal University of Pampa (Unipampa) facilities. This session engaged 4 professors, which discussed about the first SaSs modeling schema version correctness and completeness.

The moderator started the opening step welcoming and thanking the subjects, providing them with instructions about how the focus group would be performed. The subjects were instructed to try working by themselves, noting that the moderator could intervene to keep the work flowing, but the idea was that they worked without interference. The moderator also highlighted that every member was free to ask questions and to express their opinions during the evaluation. This step ended with the signing of the consent document by the subjects. In the background step, the subjects read support material to contextualize and motivate the SaSs modeling schema. The moderator answered some questions to solve conceptual gaps.

The moderator started the discussion step by introducing the evaluation instrument, which is available in Appendix B. This step was composed of 3 iterations, in each one, the moderator provided a SaSs modeling schema increment. In the first iteration, the moderator presented the context model proposal. In the second iteration, he incremented the proposal with the behavior model. Lastly, in the third iteration, he completed the proposal with the adaptivity model. The subjects discussed to build a collective perception about the SaSs modeling schema correctness and completeness. As an outcome, they produced the following findings list:

- it is not clear the relationship between context and property;
- complex contexts could be modeled by a composite structure;
- the context model do not express all required structure.

The moderator started the closure step asking to the subjects to express their final opinions about the models under evaluation. Based on all discussions, they concluded that the SaSs modeling schema **partially meets** the required structures for SaSs conceptual models.

After this focus group session, we analyzed the findings list and the subjects final opinions. We agreed with their opinions, we accepted all improvement suggestions and we promoted changes in the context model. We applied the composite pattern in entity and context modeling and we decomposed the context concept in a more expressive hierarchy. The refactored SaSs modeling schema was submitted to evaluation in the second focus group session.

5.1.1.2 Second Session Execution

The second focus group session was performed on January 12, 2018, at 13:00 PM to 16:00 PM, in the Federal University of Rio Grande do Sul (UFRGS) facilities.

This session involved 2 professors and 2 graduate students, which discussed about the second SaSs modeling schema version correctness and completeness. The moderator ran the opening and background step in the same way as in the previous focus group session.

The moderator started the discussion step by introducing the evaluation instrument, which is available in Appendix B. This step was composed of 4 iterations, in each one, the moderator provided a SaSs modeling schema increment. The iterations quantity was modified because we observed some difficulties on the subjects understanding the full context model. Thus, in this session execution, we first introduced the entity model and then we proposed the context model. The other iterations followed the same strategy of the first session execution. The subjects discussed to build a collective perception about the SaSs modeling schema correctness and completeness. As an outcome, they produced the following findings list:

- the entity model could allow association between entities;
- the context concept could be associated to property concept;
- the behavior model could generalize the action concept;
- the manager concept could be associated to context concept.

The moderator started the closure step asking to the subjects to express their final opinions about the SaSs modeling schema. As well as in the first session, they concluded that the SaSs modeling schema **partially meets** the required structures for SaSs conceptual models.

After this focus group session, we analyzed the findings list and the subjects final opinions. We also agreed with their opinions, accepting all improvement suggestions and promoting changes in the context model. To ensure the models generality, we decided to base our modeling strategy on a set of design patterns. The refactored SaSs modeling schema was submitted to evaluation in the third group session.

5.1.1.3 Third Session Execution

The third focus group session was performed on April 20, 2018, at 09:00 AM to 12:00 PM, in the Unipampa facilities. This session engaged 1 professor and 1 practitioner, which discussed about the third SaSs modeling schema version correctness and completeness. The moderator also ran the opening and background step in the same way as in the previous focus group sessions.

The moderator started the discussion step by introducing the evaluation instrument, which is available in Appendix B. This discussion step was composed of only 1 iteration.

We note that models presented in previous sessions were better understood when they were presented in the full model. Hence, we decided to present to the subjects the full SaSs modeling schema, which is now based on design patterns. The subjects discussed to build a collective perception about the model correctness and completeness. As an outcome, they reported that they did not find relevant improvement opportunities for the SaSs modeling schema.

The moderator started the closure step asking to the subjects to express their final opinions about the models under evaluation. At this time, they concluded that the SaSs modeling schema **fully meets** the required structures for SaSs conceptual models.

5.1.2 Threats to Validity

We adopted the classification scheme proposed by Cook and Campbell (1979) to identify threats to the focus group validity.

5.1.2.1 Conclusion Validity

A threat to the focus group conclusion validity is that the researchers may influence the result searching for a specific feedback. We mitigated this by carrying out a peer review process to analyze the data source and to extract the sessions findings. Other threat is that the moderator may explain the SaSs modeling schema differently in each focus group session, providing a worse or better explanation. We minimized this by providing textual documentation to subjects and monitoring the moderator interventions during sessions executions. Another threat is that some outside element may disturb the subjects during session executions. We mitigated this by performing the focus group in a reserved room and asking to the subjects do not use smart phones during the session execution.

5.1.2.2 Internal Validity

A threat to the focus group internal validity is that the history may affect the session results because they were performed in different times. We minimized this by negotiating with the subjects the best day to run the session, so that there was no event or situation that took away their concentration. Other threat is that the subjects may react negatively or positively as session time passes. We mitigated this by ensuring a maximum duration for each session and monitoring the time spent in each discussion iteration. Another threat

is that the evaluation instrument may be badly designed. We minimized this by carrying out a peer review process to verify all material used in sessions.

5.1.2.3 Construct Validity

A threat to the focus group construct validity is that the subjects may do not know the correctness and completeness meaning. We mitigated this by explaining and exemplifying the correctness and completeness meaning at the session opening step. Other threat is that the subjects may try to guess the expected results by researchers. We minimized this by masking the focus group goal to the subjects. Another threat is that the researchers may bias the results based on what they expect from each focus group session. We mitigated this by carrying out a peer review process to ensure that there are evidence in the data source that support the findings.

5.1.2.4 External Validity

A threat to the focus group external validity is that the researchers may select the wrong subjects to participate in the sessions. We minimized this by inviting to compose the set of subjects researchers with experience in modeling, metamodeling, UML, or SaSs. Other threat is that the researchers may provide the subjects with models specified in some notation unknown for them. We mitigated this by using UML as modeling language.

5.2 Experiment with Subjects

We carried out a controlled experiment with subjects (WOHLIN et al., 2012) to evaluate the SaSML. The evaluation goal was to verify whether the Adaptive Behavior modeling element would make explicit and enable the modeling of contextual information and behavioral possibilities in SaSs conceptual models. The experiment scope[1] was:

Analyze	the Adaptive Behavior modeling element
for the purpose of	evaluation
with respect to	expressiveness and effectiveness
from the point view of	the researcher
in the context of	undergraduate students working with conceptual models.

[1]The experiment scope was formated according the Goal/Question/Metric (GQM) measurement goal template (SOLINGEN; BERGHOUT, 1999).

We used information retrieval metrics as measure to evaluate Adaptive Behavior, such as, De Lucia et al. (De Lucia et al., 2010). Hence, we have:

- $\textit{F-score} = 2 * \frac{\textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}}$;
- $\textit{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$;
- $\textit{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$;
- *True Positive* is the number of relevant items answered by the subject;
- *False Positive* is the number of non-relevant items answered by the subject;
- *False Negative* is the number of relevant items not answered by the subject.

5.2.1 Experiment Planning

To evaluate Adaptive Behavior, we ran a specific purpose off-line experiment with software engineering undergraduate students. During the experiment, they had to interpret and to build conceptual models from real SaSs scenarios. The experiment was organized according to the "One Factor with Two Treatments and Paired Comparison" standard design type. The factor was the modeling language and the treatments were the UML standard elements and the Adaptive Behavior modeling element. We enabled the paired comparison by collecting the subjects F-Scores before and after introducing Adaptive Behavior. Thus, in this experiment, the UML standard elements and the Adaptive Behavior modeling element were the independent variables whereas F-Scores was the dependent variable.

Our first goal was to evaluate the Adaptive Behavior expressiveness, i.e., its ability to make explicit the relevant information in SaSs conceptual models. We advocated that the subjects would be able to identify contextual information and behavioral possibilities in Adaptive Behavior. We verified this by comparing the F-scores before (Baseline) and after (Experimental) introducing Adaptive Behavior. Thus, we formalized our expressiveness hypotheses as follows:

- $H0_{Exp} : \textit{F-score}_{Baseline} = \textit{F-score}_{Experimental}$
- $H1_{Exp} : \textit{F-score}_{Baseline} \neq \textit{F-score}_{Experimental}$

Our second goal was to evaluate the Adaptive Behavior effectiveness, i.e., its ability to produce the relevant information in SaSs conceptual models. We advocated that the subjects would be able to model contextual information and behavioral possibilities

with Adaptive Behavior. We also verified this by comparing the F-scores before (Baseline) and after (Experimental) introducing Adaptive Behavior. Thus, we formalized our effectiveness hypotheses as follows:

- $H0_{Eff}$: $F\text{-}score_{Baseline} = F\text{-}score_{Experimental}$
- $H1_{Eff}$: $F\text{-}score_{Baseline} \neq F\text{-}score_{Experimental}$

The subjects were selected from a population of software engineering undergraduate students. To ensure a sample with similar skills and abilities, we invited only third-year and fourth-year students. Third-year students already attended software analysis and design classes, and they are studying software engineering advanced topics. Fourth-year students already attended all curricular content, and they are doing the supervised internship and the term paper. We used this classification as blocking criteria for this experiment.

We provided to the subjects the material required to carry out the experiment. Besides a supporting documentation, we supplied them with a set of instruments to collect the experiment data. As can be seen in Appendix C, each instrument proposed a specific activity to be performed by the subjects. We ran the experiment according to the following roadmap:

1. reviewing of SaSs, requirements specification, and conceptual modeling;
2. interpreting of a SaSs conceptual model specified in UML;
3. modeling of a SaSs conceptual model with UML;
4. learning of the Adaptive Behavior modeling element;
5. interpreting of a SaSs conceptual model specified with Adaptive Behavior;
6. modeling of a SaSs conceptual model with Adaptive Behavior.

We note that the Baseline data were collected in the steps 2 and 3 whereas the Experimental data were collected in the steps 5 and 6. The set of activities were elaborated by the researchers based on SaSs scenarios presented in literature.

5.2.2 Experiment Execution

First of all, we established the experiment potential sample as planning before. We identified 66 students able to participate in the experiment. From this, 39 students had signed up to participate and 36 effectively participated. We present in Table 5.1 the quantity of subjects according to the blocking criteria.

As a matter of schedule, we divided each execution session in 2 meetings, being the first with 2 hours and the second with 1 hour of duration. We ran the first session on

Table 5.1: Subjects quantity according to blocking criteria.

	Quantity
Third-year Students	18
Fourth-year Students	18
Total	**36**

Source: The authors.

May 8 and 10, 2018, and the second session on May 15 and 16, 2018. Each session took place in an isolated room and was attended by 18 subjects. The researchers guided the sessions without interfering or providing any support to the subjects.

After all sessions, subjects models were corrected in parallel by two researchers. The correction process was supported by a reference conceptual model and a set of guidelines to reviewers, as can bee seen in Appendix D. The results were analyzed to solve corrections differences and the data were tabulated to calculate the metrics.

5.2.3 Results and Analysis

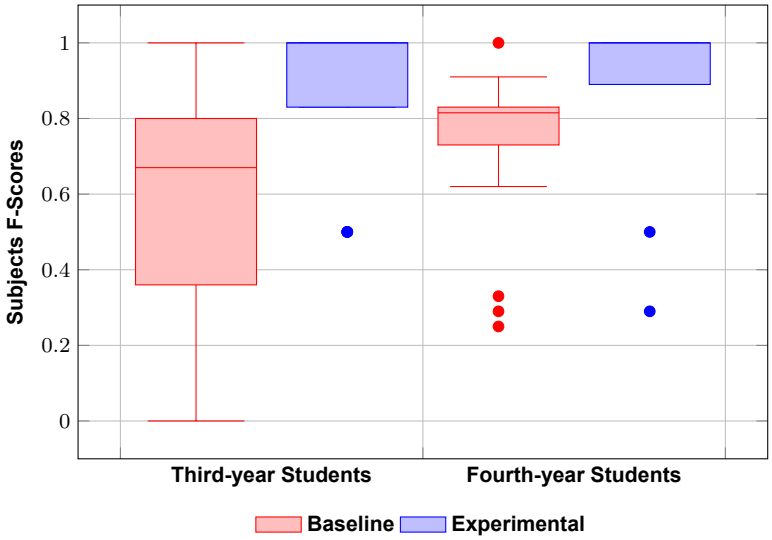
The experiment results data are available in Appendix E. Next, we summarize the data and analyze the results against to the research goals.

5.2.3.1 Expressiveness Analysis

We analyze the Adaptive Behavior expressiveness evaluation result, which is presented in Figure 5.2. The chart is organized in two data groups, where the left presents the Third-year Students results and the right presents the Fourth-year Students results. For each group, it shows the results before (Baseline) and after (Experimental) introducing Adaptive Behavior to the subjects.

Firstly, we analyze the Third-year Students data behavior. In Baseline, the subjects median is 0.67, most the subjects have a F-score between 0.36 and 0.80, but some have F-scores that are as low as 0.00 and as high as 1.00. In Experimental, the subjects median is 1.00, most the subjects have a F-score between 0.83 and 1.00, and there are three outliers with F-score 0.50. The data centers (medians) clearly show that the Third-year Students had a better performance after introducing Adaptive Behavior. This is corroborated by the quartiles behavior: Experimental lower quartile is greater than Baseline upper quartile. Analyzing the data spreads (inter-quartiles), both Baseline and Experimental have a positive asymmetric distribution, but Baseline has a less homogeneous distribution than

Figure 5.2: Adaptive Behavior expressiveness evaluation result.



Source: The authors.

Experimental.

Secondly, we analyze the Fourth-year Students data behavior. In Baseline, the subjects median is 0.82, most the subjects have a F-score between 0.73 and 0.83, but some have F-scores that are as low as 0.62 and as high as 0.91, and there are four outliers with F-scores 0.25, 0.29, 0.33, and 1.00 respectively. In Experimental, the subjects median is 1.00, most the subjects have a F-score between 0.89 and 1.00, and there are two outliers with F-score 0.29 and 0.50 respectively. The data centers (medians) clearly show that the Fourth-year Students had a better performance after introducing Adaptive Behavior. This is corroborated by the quartiles behavior: Experimental lower quartile is greater than Baseline upper quartile. Analyzing the data spreads (inter-quartiles), both Baseline and Experimental have a positive asymmetric distribution and there is no significant difference between distributions homogeneity.

It is noticeable in Figure 5.2 a visual difference between Baseline and Experimental results. We performed a hypothesis test to determine whether this difference is statistically significant. To choose a proper test method, we first verified if the sample comes from a normal population. According to the Shapiro-Wilk test, we have $W_{\textit{F-score}} = 0.898 < W_{(0.05,36)} = 0.935$ and $\textit{P-value}_{\textit{F-Score}} = 0.003 < \alpha = 0.050$, meaning that the sample does not come from a normal population with a 5% significance level. For this reason, we applied a paired Wilcoxon test to compare the dependent samples. As a result, we obtained $\textit{P-value} = 0.00008 < \alpha = 0.05$, therefore the difference between Baseline and Experimental results is considered statistically significant.

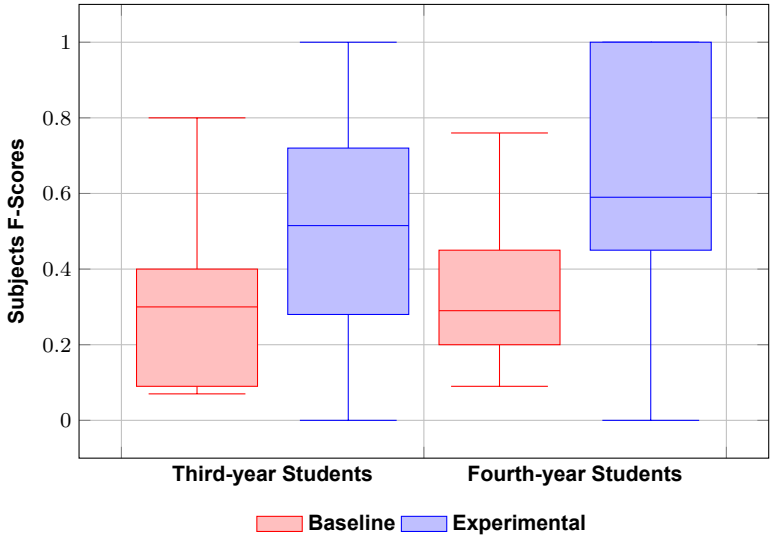
Based on the hypothesis test result, we can refute the null hypothesis $H0_{Exp}$ and

accept the alternative hypothesis $H1_{Exp}$. Based on the F-Scores behavior, we can say that the Adaptive Behavior is able to make explicit contextual information and behavioral possibilities in SaSs conceptual model. We note that these conclusions are restricted by this experiment scope, not being possible to generalize to other populations.

5.2.3.2 Effectiveness Analysis

We analyze the Adaptive Behavior effectiveness evaluation result, which is presented in Figure 5.3. The chart is organized in two data groups, where the left presents the Third-year Students results and the right presents the Fourth-year Students results. For each group, it shows the results before (Baseline) and after (Experimental) introducing Adaptive Behavior to the subjects.

Figure 5.3: Adaptive Behavior effectiveness evaluation result.



Source: The authors.

Firstly, we analyze the Third-year Students data behavior. In Baseline, the subjects median is 0.30, most the subjects have a F-score between 0.09 and 0.40, but some have F-scores that are as low as 0.40 and as high as 0.80. In Experimental, the subjects median is 0.52, most the subjects have a F-score between 0.28 and 0.72, but some have F-scores that are as low as 0.00 and as high as 1.00. The data centers (medians) clearly show that the Third-year Students had a better performance after introducing Adaptive Behavior. This is corroborated by the quartiles behavior: Experimental lower quartile is relatively close to Baseline upper quartile. Analyzing the data spreads (inter-quartiles), both Baseline and Experimental have a positive asymmetric distribution, but Experimental has a little less homogeneous distribution than Baseline.

Secondly, we analyze the Fourth-year Students data behavior. In Baseline, the subjects median is 0.29, most the subjects have a F-score between 0.20 and 0.45, but some have F-scores that are as low as 0.09 and as high as 0.76. In Experimental, the subjects median is 0.59, most the subjects have a F-score between 0.45 and 1.00, but some have F-scores that are as low as 0.00 and as high as 1.00. The data centers (medians) clearly show that the Fourth-year Students had a better performance after introducing Adaptive Behavior. This is corroborated by the quartiles behavior: Experimental lower quartile is equal to Baseline upper quartile. Analyzing the data spreads (inter-quartiles), both Baseline and Experimental have a negative asymmetric distribution, but Experimental has a less homogeneous distribution than Baseline.

It is also noticeable in Figure 5.3 a visual difference between Baseline and Experimental results. We performed other hypothesis test to determine whether this difference is statistically significant. Again, we started by verifying if the sample comes from a normal population. According to the Shapiro-Wilk test, we have $W_{\textit{F-score}} = 0.986 > W_{(0.05,36)} = 0.935$ and $\textit{P-value}_{\textit{F-Score}} = 0.920 > \alpha = 0.050$, meaning that the sample comes from a normal population with a 5% significance level. For this reason, we applied a paired Student's t-test to compare the dependent samples. As a result, we obtained $\textit{P-value} = 0.00006 < \alpha = 0.05$, therefore the difference between Baseline and Experimental results is considered statistically significant.

Based on the hypothesis test result, we can refute the null hypothesis $H0_{Eff}$ and accept the alternative hypothesis $H1_{Eff}$. Based on F-Scores behavior, we can say that Adaptive Behavior is able to model contextual information and behavioral possibilities in SaSs conceptual models. We also note that these conclusions are restricted by this experiment scope, not being possible to generalize to other populations.

5.2.4 Threats to Validity

We also adopted the classification scheme proposed by Cook and Campbell (1979) to identify threats to the experiment validity.

5.2.4.1 Conclusion Validity

A threat to the experiment conclusion validity is that the statistical test may not reveal a true pattern in the data. We mitigated this by maximizing the sample size ($\cong 25\%$

of the population) and applying statistical conventional criteria ($\alpha = 0.05$). Other threat is that the researcher may have violated some statistical test assumptions. We mitigated this by choosing a suitable statistical test for each data group (Wilcoxon test for dependent or independent samples that do not come from a normal population and Student's t-test for dependent or independent samples that come from a normal population). Another threat is that the researchers may search for a specific result. We mitigated this by defining reference models and guidelines to drive the correction of the models built by the subjects.

Another threat is that the researchers may use measures with low reliability. We mitigated this by using information retrieve metrics (Precision, Recall, and F-score), which are well-defined and widely used. Another threat is that the implementation may be different among the subjects. We mitigated this by providing textual documentation to subjects and monitoring the implementation during sessions executions. Another threat is that outside elements may disturb the experiment results. We mitigated this by performing the sessions in a reserved room and asking to the subjects do not use smart phones during the session execution. Another threat is that the subjects heterogeneity may affect the results interpretation. We mitigated this by selecting third-year and fourth-year software engineering undergraduate students and using this classification as blocking criteria.

5.2.4.2 Internal Validity

A threat to the experiment internal validity is that the history may affect the experiment results. We mitigated this by negotiating with the subjects the best day to run the session so that there was no event or situation that took away their concentration. Other threat is that the subjects may react positively (learning) or negatively (tired or bored) as time passes. We mitigated this by ensuring a maximum duration for each session (two hours) and monitoring the time spent in each activity during each session. Another threat is that the subjects may respond differently at different times once they know how the test is conducted. We mitigated this by introducing the activities for all subjects at the same time and in a specific order, and avoiding any feedback about the activities during the session execution.

Another threat is that the artifacts used for the experiment execution may be badly designed. We mitigated this by carrying out a peer review process to verify all material used in each session. Another threat is that the subjects performance may be very heterogeneous. We mitigated this by selecting the subjects from a population with similar skills and abilities, and applying a blocking strategy to distinguish the subjects maturity.

Another threat is that the subjects may leave the experiment execution. We mitigated this by negotiating with some professors a slot in their class schedule to perform the activities. Another threat is that the interactions with selection may lead to different behavior in different groups. We mitigated this by determining a time-box for each activity and allocating subjects with the same background.

5.2.4.3 Construct Validity

A threat to the experiment construct validity is that the constructs may be insufficiently defined. We mitigated this using a set of well-defined quantitative measures (information retrieve metrics). Other threat is that the experiment may under-represent the construct. We mitigated this by proposing activities based on a set of different but equivalent scenarios. Another threat is that the treatment application may become the subjects more receptive to the treatment. We mitigated this by hiding from the subjects the technique used to analyze the models produced by them.

Another threat is that the treatment may affect positively or negatively the studied construct. We mitigated this by mapping the relevant attributes to treatment evaluation (expressiveness and effectiveness). Another threat is that the subjects may try to discover the experiment purpose. We mitigated this by hiding from the subjects the evaluation goals. Another threat is that the subjects may bias the results based on their expectations. We mitigated this by carrying out peer reviews to ensure that the models built by the subjects were correctly analyzed.

5.2.4.4 External Validity

A threat to the experiment external validity is that the sample may not represent the population. We mitigated this by selecting subjects according to a required profile (Third-year and Fourth-year students). Other threat is that the material may not be suitable to the experiment activities. We mitigated this by using paper and pencil to perform the modeling activity.

5.3 Chapter Lessons

In this chapter, we report the empirical techniques used to evaluate the SaSs modeling schema correctness and completeness, and the Adaptive Behavior modeling element

expressiveness and effectiveness. The chapter main points are:

- the feedback obtained from focus group sessions guided our decisions and aided us to produce the SaSs modeling schema;
- Adaptive Behavior is able to make explicit contextual information and behavioral possibilities in SaSs conceptual model;
- Adaptive Behavior is able to model contextual information and behavioral possibilities in SaSs conceptual models;
- the experiment conclusions are restricted by this experiment scope, not being possible to generalize to other populations.

In next chapter, we summarize the work results and conclusions, moreover, we discuss some work limitations and propose future work.

6 CONCLUSIONS

SaSs can autonomously decide how to adapt their behavior at runtime in response to contextual changes (ANDERSSON et al., 2009; BRUN et al., 2009; CHENG et al., 2009). Conceptual modeling is the act of creating models that describe problems independently of the solutions for purposes of understanding and communication (BATINI; CERI; NAVATHE, 1992; MYLOPOULOS, 1992; CHEN; THALHEIM; WONG, 1999). SaSs conceptual modeling is challenging because it is needed to deal with requirements uncertainty, contextual changes, and behavioral possibilities. This complexity can be minimized by using DSMLs (FRANK, 2011), which may be created by extending UML (STAHL et al., 2006).

Our literature investigation revealed that UML has been extended to provide proper support for SaSs conceptual modeling. The class diagram has been widely customized to support the context-awareness and self-adaptiveness analysis models (SILVA et al., 2018a). However, we also discovered that none of the studies integrate the context-awareness and self-adaptiveness modeling. Thus, we propose a UML-based DSML called SaSML that introduces a new modeling element called Adaptive Behavior able to synthesize a SaSs modeling schema based on design patterns.

SaSs modeling schema applies a set of design patterns to capture the higher-level abstractions related to SaSs domain. The design patterns usage in the SaSs modeling schema development contributed to a more flexible and reusable model. SaSML was created by modifying the UML metamodel, ensuring the compatibility with UML standard elements, therefore, all UML valid constructions also are SaSML valid constructions. Adaptive Behavior synthesizes a SaSs modeling schema, exposing only what needs to be defined at modeling time. SaSML models are visually more synthetic than UML models, contributing with the model comprehensibility.

We evaluated the SaSs modeling schema correctness and completeness by running focus group sessions. The feedback obtained from the subjects guided our decisions and aided us to capture the relevant SaSs higher-level abstractions. We evaluated the Adaptive Behavior modeling element expressiveness and effectiveness by running an experiment with subjects. The experiment results provide statistical evidence that the proposed UML-based DSML effectively supports the SaSs conceptual modeling. Adaptive Behavior is able to make explicit contextual information and behavioral possibilities in SaSs conceptual model. Adaptive Behavior is able to model contextual information and behavioral

possibilities in SaSs conceptual models.

Our hypothesis is that SaSML contributes to the modeling quality by capturing the SaSs higher-level abstractions. Whereas the models quality is related to theirs communications capability (HULL; JACKSON; DICK, 2011), models are restricted by the languages expressiveness power (THALHEIM, 2011), and DSMLs provide higher-level abstractions for mapping a targeted domain (FRANK, 2011). Whereas SaSML is a UML-based DSML that extends UML introducing a new modeling element, Adaptive Behavior synthesizes the SaSs modeling schema without losing expressiveness, and the SaSs modeling schema captures the SaSs higher-level abstractions by applying a set of design patterns. We can conclude that SaSML contributes to the SaSs conceptual modeling quality, therefore, we can accept the hypothesis.

6.1 Future Work

The Adaptive Behavior modeling element introducing by SaSML first version supports only the structural modeling. Adaptive Behavior specifies the contextual information and behavioral possibilities related to the adaptation process, but it does not specify which situation (contextual information instance) triggers each system behavior. We propose extending UML behavioral modeling elements, state machines for example, to support the SaSs adaptation rules modeling.

Conceptual models are useful for requirements analysis because they aid in understanding the situation in which a problem occurs (BOURQUE; FAIRLEY, 2014). The SaSs modeling schema aids in the modeling process by indicating the main concepts to be considered during SaSs conceptual modeling. However, higher-level abstractions do not ensure that the requirements was properly analyzed during the modeling process. We propose creating a process that drives the work of software engineers during the requirements analysis process. The first SaSs modeling process version can be seen in Silva et al. (2018b) and a draft of the next version can be seen in Appendix F.

SaSML is not yet supported by Computer-Aided Software Engineering (CASE) tools. A good CASE tool is essential to make SaSML feasible to practitioners and industry. We have worked towards developing CASE tools for requirements specification (MORO, 2015) and conceptual modeling (FRANZ, 2017) oriented to SaSs. We propose developing a CASE tool that supports the Adaptive Behavior modeling element, as well as transformation of a SaSML model to a UML model.

We evaluated SaSML by applying two distinct empirical techniques (Focus Group and Experiment with Subjects). As in every empirical study, the conclusions are restricted by the evaluation scope, not being possible to generalize to other populations. We propose reapplying the experiment in other scenarios and population to maximize the conclusions generality. We also propose carrying out Case Studies[1] to evaluate the Adaptive Behavior modeling element in a real-life context.

[1] An empirical technique that uses several evidence sources to investigate one or few software engineering phenomenon instance within its real-life context (WOHLIN et al., 2012).

REFERENCES

AL-ALSHUHAI, A.; SIEWE, F. An Extension of UML Activity Diagram to Model the Behaviour of Context-Aware Systems. In: **2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing**. Liverpool: IEEE, 2015. p. 431–437. ISBN 978-1-5090-0154-5.

ALMUTAIRI, S.; BELLA, G.; ABU-SAMAHA, A. Specifying security requirements of context aware system using UML. In: **Seventh International Conference on Digital Information Management (ICDIM 2012)**. Macau: IEEE, 2012. p. 259–265. ISBN 978-1-4673-2430-4.

ANDERSSON, J. et al. Modeling Dimensions of Self-Adaptive Software Systems. In: **Software Engineering for Self-Adaptive Systems**. Berlin: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5525). p. 27–47. ISBN 3642021603.

AYED, D.; DELANOTE, D.; BERBERS, Y. MDD Approach for the Development of Context-Aware Applications. In: **Modeling and Using Context**. Berlin: Springer Berlin Heidelberg, 2007, (Lecture Notes in Computer Science). p. 15–28.

BARESI, L.; PASQUALE, L. Live goals for adaptive service compositions. In: **Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10**. New York: ACM Press, 2010. p. 114–123. ISBN 9781605589718. ISSN 02705257.

BATINI, C.; CERI, S.; NAVATHE, S. B. **Conceptual Database Design: An Entity-Relationship Approach**. Boston: Addison-Wesley, 1992. 470 p. ISBN 9780805302448.

BECKER, M.; LUCKEY, M.; BECKER, S. Model-driven performance engineering of self-adaptive systems. In: **Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures - QoSA '12**. New York: ACM Press, 2012. p. 117. ISBN 9781450313469.

BENCOMO, N. et al. Requirements reflection. In: **Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10**. New York: ACM Press, 2010. v. 2, p. 199. ISBN 9781605587196. ISSN 0270-5257.

BENSELIM, M. S.; SERIDI-BOUCHELAGHEM, H. Extended UML for the Development of Context-Aware Applications. In: **Networked Digital Technologies**. Berlin: Springer, 2012, (Communications in Computer and Information Science, v. 293). p. 33–43.

BENSELIM, M.-S.; SERIDI-BOUCHELAGHEM, H. Towards A UML Profile for Context-Awareness Domain. **International Arab Journal of Information Technology**, v. 14, n. June, p. 195–207, 2017.

BLOOR, M. et al. **Focus Groups in Social Research**. London: SAGE Publications, 2001. ISBN 0761957421.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language User Guide**. 2. ed. Boston: Addison-Wesley, 2005. 496 p. ISBN 0321267974.

BOUDJEMLINE, H. et al. Heavyweight extension to the UML class diagram metamodel for modeling context aware systems in ubiquitous computing. **International Journal of Pervasive Computing and Communications**, v. 13, n. 3, p. 238–251, 2017. ISSN 1742-7371.

BOURQUE, P.; FAIRLEY, R. E. (Ed.). **Guide to the Software Engineering Body of Knowledge**. 3. ed. Piscataway: IEEE Press, 2014. 335 p. ISBN 9780769551661.

BRINGS, J.; SALMON, A.; SARITAS, S. Context uncertainty in requirements engineering: Definition of a search strategy for a systematic review and preliminary results. In: **CEUR Workshop Proceedings**. Essen: CEUR, 2015. v. 1342, p. 171–178.

BRUCK, J.; HUSSEY, K. **Customizing UML: Which Technique is Right for You?** 2008. Available at: <https://www.eclipse.org/modeling/mdt/uml2/docs/articles/Customizing_UML2_Which_Technique_is_Right_For_You/article.html>, Accessed on: 08/30/18.

BRUN, Y. et al. Engineering Self-Adaptive Systems through Feedback Loops. In: **Software Engineering for Self-Adaptive Systems**. Berlin: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5525). p. 48–70. ISBN 3642021603.

CAMBRIDGE. **Cambridge Dictionary**. Cambridge: Cambridge University Press, 2018. Available at: <https://dictionary.cambridge.org/>, Accessed on: 08/30/18.

CHEN, P. P.; THALHEIM, B.; WONG, L. Y. Future Directions of Conceptual Modeling. In: **Conceptual Modeling: Current Issues and Future Directions**. Berlin: Springer Berlin Heidelberg, 1999. p. 287–301. ISBN 978-3-540-65926-6.

CHENG, B. H. C. et al. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In: **Software Engineering for Self-Adaptive Systems**. Berlin: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5525). p. 1–26. ISBN 978-3-642-02160-2, 978-3-642-02161-9.

COOK, T. D.; CAMPBELL, D. T. **Quasi-Experimentation: Design & Analysis Issues for Field Settings**. Boston: Houghton Mifflin, 1979. 420 p. ISBN 9780395307908.

COSTA, P. et al. Situations in Conceptual Modeling of Context. In: **2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06)**. Hong Kong: IEEE, 2006. p. 6–6. ISBN 0-7695-2743-4.

De Lucia, A. et al. An experimental comparison of ER and UML class diagrams for data modelling. **Empirical Software Engineering**, v. 15, n. 5, p. 455–492, 2010. ISSN 1382-3256.

DEVINCENZI, S. et al. O uso de tecnologias persuasivas para potencializar o processo de aprendizagem baseado em problemas. **Revista Spacios**, v. 38, n. 60, p. 13, 2017. ISSN 07981015.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing Journal**, v. 1, n. 5, p. 4–7, 2001.

FRANK, U. Some guidelines for the conception of domain-specific modelling languages. In: **Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures**. Bonn: Gesellschaft für Informatik, 2011. p. 93–106. ISBN 9783885792840.

FRANZ, L. P. **Extração Automática de Modelos Conceituais a Partir de Requisitos Para Sistemas Autoadaptativos**. 2017. Available at: <http://dspace.unipampa.edu.br/bitstream/riu/1941/1/Luiz%20Paulo%20Franz%20-%202017.pdf>, Accessed on: 08/30/18.

FREEMAN, E. et al. **Head First Design Patterns: A Brain-Friendly Guide**. Sebastopol: O'Reilly Media, 2004. 694 p. ISBN 978-0596007126.

FUENTES, L.; GAMEZ, N.; SANCHEZ, P. Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications. In: **2008 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software**. [S.l.]: IEEE, 2008. p. 34–43. ISBN 978-0-7695-3104-5.

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Boston: Addison-Wesley Professional, 1994. 395 p. ISBN 978-0201633610.

GOGOLLA, M. UML and OCL in Conceptual Modeling. In: **Handbook of Conceptual Modeling**. Berlin: Springer Berlin Heidelberg, 2011. p. 85–122. ISBN 978-3-642-15864-3.

HAN, D. et al. FAME: A UML-based framework for modeling fuzzy self-adaptive software. **Information and Software Technology**, Elsevier, v. 76, p. 118–134, 2016. ISSN 09505849.

HEBIG, R.; GIESE, H.; BECKER, B. Making control loops explicit when architecting self-adaptive systems. In: **Proceeding of the second international workshop on Self-organizing architectures - SOAR '10**. New York: ACM Press, 2010. p. 21. ISBN 9781450300872.

HSU, I.-C. Extending UML to model Web 2.0-based context-aware applications. **Software: Practice and Experience**, v. 42, n. 10, p. 1211–1227, 2012. ISSN 00380644.

HULL, E.; JACKSON, K.; DICK, J. **Requirements Engineering**. 3. ed. London: Springer, 2011. ISBN 1849964041.

JALALI, S.; WOHLIN, C. Systematic Literature Studies: Database Searches vs. Backward Snowballing. In: **Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '12**. New York: ACM Press, 2012. p. 29. ISBN 9781450310567. ISSN 19493770.

KAPITSAKI, G. M.; VENIERIS, I. S. PCP: Privacy-aware Context Profile towards Context- aware Application Development. In: **Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services - iiWAS '08**. New York: ACM Press, 2008. p. 104. ISBN 9781605583495.

KEPHART, J.; CHESS, D. The vision of autonomic computing. **Computer**, v. 36, n. 1, p. 41–50, 2003. ISSN 0018-9162.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature reviews in Software Engineering (Version 2.3)**. 2007. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=EC339FBB73AFA1A0175176EBD97F6EFF?doi=10.1.1.117.471&rep=rep1&type=pdf>, Accessed on: 08/30/18.

KRUPITZER, C. et al. A survey on engineering approaches for self-adaptive systems. **Pervasive and Mobile Computing**, v. 17, n. Part B, p. 184–206, 2015. ISSN 15741192.

LEWIS, P. R. et al. A Survey of Self-Awareness and Its Application in Computing Systems. In: **2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops**. Ann Arbor: IEEE, 2011. p. 102–107. ISBN 978-1-4577-2029-1.

LINSTONE, H. A.; TUROFF, M. **The Delphi Method: Techniques and Applications**. Boston: Addison-Wesley, 1975. 621 p. ISBN 978-0-201-04294-8.

LUCKEY, M.; ENGELS, G. High-quality specification of self-adaptive software systems. In: **2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. Los Alamitos: IEEE, 2013. p. 143–152. ISBN 978-1-4673-4401-2.

LUCKEY, M. et al. Adapt Cases: Extending Use Cases for Adaptive Systems. In: **Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11**. New York: ACM Press, 2011. p. 30. ISBN 9781450305754.

MACÍAS-ESCRIVÁ, F. D. et al. Self-adaptive systems: A survey of current approaches, research challenges and applications. **Expert Systems with Applications**, v. 40, n. 18, p. 7267–7279, 2013. ISSN 09574174.

MALAVOLTA, I.; MUCCINI, H.; SEBASTIANI, M. Automatically Bridging UML Profiles to MOF Metamodels. In: **2015 41st Euromicro Conference on Software Engineering and Advanced Applications**. Funchal: IEEE, 2015. p. 259 – 266. ISBN 9781467375856.

MINTON, E. A.; KHALE, L. R. **Belief Systems, Religion, and Behavioral Economics**. New York: Business Expert Press, 2014. ISBN 978-1-60649-704-3.

MOHABBATI, B. et al. Combining service-orientation and software product line engineering: A systematic mapping study. **Information and Software Technology**, v. 55, n. 11, p. 1845–1859, 2013. ISSN 09505849.

MORO, G. B. **Uma Ferramenta de Apoio à Especificação de Requisitos para Sistemas Autoadaptativos**. [S.l.], 2015. Available at: <http://dspace.unipampa.edu.br/bitstream/riu/875/1/Uma%20ferramenta%20de%20apoio%20%C3%A0%20especifica%C3%A7%C3%A3o%20de%20requisitos%20para%20sistemas%20autoadaptativos.pdf>, Accessed on: 08/15/18.

MYLOPOULOS, J. Conceptual modelling and Telos. In: LOUCOPOULOS, P.; ZICARI, R. (Ed.). **Conceptual Modeling, Databases, and Case An integrated view of information systems development.** New York: John Wiley and Sons, 1992. p. 49–68. ISBN 0471554626.

OMG, O. M. G. **OMG Unified Modeling Language - Version 2.5.1**. 2017. Available at: <https://www.omg.org/spec/UML/2.5.1/>, Accessed on: 08/30/18.

OREIZY, P. et al. An architecture-based approach to self-adaptive software. **IEEE Intelligent Systems**, v. 14, n. 3, p. 54–62, 1999. ISSN 1094-7167.

OXFORD. **Oxford Dictionaries**. 2018. Available at: <https://en.oxforddictionaries.com/>, Accessed on: 08/30/18.

PATIKIRIKORALA, T. et al. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In: **2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. Zurich: IEEE, 2012. p. 33–42. ISBN 978-1-4673-1787-0.

PMI. **A Guide to the Project Management Body of Knowledge**. 5. ed. Newtown Square: Project Management Institute, 2013. 580 p. ISBN 978-1-935589-67-9.

ROUSSOPOULOS, N.; KARAGIANNIS, D. Conceptual Modeling: Past, Present and the Continuum of the Future. In: BORGIDA, A. T. et al. (Ed.). **Conceptual Modeling: Foundations and Applications**. Berlin: Springer-Verlag, 2009, (Lecture Notes in Computer Science). p. 139–152. ISBN 978-3-642-02462-7.

SALEHIE, M.; TAHVILDARI, L. Self-Adaptive Software: Landscape and Research Challenges. **ACM Transactions on Autonomous and Adaptive Systems**, v. 4, n. 2, p. 1–42, 2009. ISSN 15564665.

SAWYER, P. et al. Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems. In: **2010 18th IEEE International Requirements Engineering Conference**. Sydney: IEEE, 2010. p. 95–103. ISBN 978-1-4244-8022-7. ISSN 1090-705X.

SHALLOWAY, A.; TROTT, J. R. **Design Patterns Explained: A New Perspective on Object Oriented Design**. 2. ed. Boston: Addison-Wesley, 2004. 480 p. ISBN 978-0321247148.

SHENG, Q.; BENATALLAH, B. ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. In: **International Conference on Mobile Business (ICMB'05)**. Sydney: IEEE, 2005. p. 206–212. ISBN 0-7695-2367-6.

SHVETS, A. **Design Patterns Explained Simply**. Kyiv: Source Making, 2018. 119 p.

SILVA, J. P. S. da et al. A systematic literature review of UML-based domain-specific modeling languages for self-adaptive systems. In: **Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '18**. New York: ACM Press, 2018. p. 87–93. ISBN 9781450357159.

SILVA, J. P. S. da et al. Improving self-adaptive systems conceptual modeling. In: **Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC '18**. New York: ACM Press, 2018. p. 1292–1299. ISBN 9781450351911.

SILVA, J. P. S. da et al. Towards a Domain-Specific Modeling Language for Self-adaptive Systems Conceptual Modeling. In: **XXXII Brazilian Symposium on Software Engineering (SBES 2018)**. New York: ACM Press, 2018. p. 6. ISBN 978-1-4503-6503-1/18/09.

SIMONS, C.; WIRTZ, G. Modeling context in mobile distributed systems with the UML. **Journal of Visual Languages & Computing**, v. 18, n. 4, p. 420–439, 2007. ISSN 1045926X.

SIQUEIRA, B. R. et al. Characterisation of Challenges for Testing of Adaptive Systems. In: **Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST**. New York: ACM Press, 2016. p. 1–10. ISBN 9781450347662.

SOLINGEN, R.; BERGHOUT, E. **The goal/question/metric method**. London: McGraw-Hill, 1999. 216 p. ISBN 0077095537.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Boston: Addison Wesley, 2010. ISBN 0137035152.

SOUZA, V. E. S. et al. Requirements-driven software evolution. **Computer Science - Research and Development**, v. 28, n. 4, p. 311–329, 2013. ISSN 1865-2034.

SOUZA, V. E. S. et al. Awareness requirements for adaptive systems. In: **Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11**. New York: ACM Press, 2011. p. 60. ISBN 9781450305754.

STAHL, T. et al. **Model-Driven Software Development: Techonolgy, Engineering, Management**. Chichester: John Wiley and Sons, 2006. 428 p. ISBN 9780470025703.

STÖRRLE, H. How are Conceptual Models used in Industrial Software Development? In: **Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering**. New York: ACM Press, 2017. p. 160–169.

THALHEIM, B. The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling. In: EMBLEY, D. W.; THALHEIM, B. (Ed.). **Handbook of Conceptual Modeling**. Berlin: Springer Berlin Heidelberg, 2011. p. 543–577. ISBN 978-3-642-15864-3.

TURBAK, F.; GIFFORD, D.; SHELDON, M. A. **Design Concepts in Programming Languages**. Cambridge: MIT Press, 2008. 1322 p. ISBN 9780262201759.

Van Velsen, L. et al. User-centered evaluation of adaptive and adaptable systems: a literature review. **The Knowledge Engineering Review**, v. 23, n. 03, p. 261–281, 2008. ISSN 0269-8889.

VASSEV, E.; HINCHEY, M. Knowledge Representation for Adaptive and Self-aware Systems. In: WIRSING, M. et al. (Ed.). **Software Engineering for Collective Autonomic Systems**. Cham: Springer, 2015, (Lecture Notes in Computer Science, v. 8998). p. 221–247. ISBN 978-3-319-16309-3, 978-3-319-16310-9.

WENZL, S. S.; STREMBECK, M. Modelling context-aware RBAC models for mobile business processes. **International Journal of Wireless and Mobile Computing**, v. 6, n. 5, p. 448, 2013. ISSN 1741-1084.

WEYNS, D.; AHMAD, T. Claims and Evidence for Architecture-Based Self-adaptation: A Systematic Literature Review. In: **Software Architecture**. Berlin: Springer Berlin Heidelberg, 2013, (Lecture Notes in Computer Science, v. 7957). p. 249–265. ISBN 978-3-642-39030-2.

WEYNS, D. et al. A survey of formal methods in self-adaptive systems. In: **Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering - C3S2E '12**. New York: ACM Press, 2012. p. 67–79. ISBN 9781450310840.

WHITTLE, J. et al. RELAX: a language to address uncertainty in self-adaptive systems requirement. **Requirements Engineering**, v. 15, n. 2, p. 177–196, 2010. ISSN 0947-3602.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14**. New York: ACM Press, 2014. p. 1–10. ISBN 9781450324762. ISSN 09505849.

WOHLIN, C. et al. **Experimentation in Software Engineering**. Berlin: Springer-Verlag, 2012. 249 p. ISBN 9783642290435.

YANG, Z. et al. A Systematic Literature Review of Requirements Modeling and Analysis for Self-adaptive Systems. In: **Requirements Engineering: Foundation for Software Quality**. Cham: Springer International Publishing, 2014, (Lecture Notes in Computer Science, v. 8396). p. 55–71. ISBN 978-3-319-05842-9.

YUAN, E.; ESFAHANI, N.; MALEK, S. A Systematic Survey of Self-Protecting Software Systems. **ACM Transactions on Autonomous and Adaptive Systems**, v. 8, n. 4, p. 1–41, 2014. ISSN 15564665.

APPENDIX A — ACADEMIC WORKS

Related to the Thesis

Works published in proceedings:

1. SILVA, João Pablo Silva da *et al*. Improving self-adaptive systems conceptual modeling. In: 33rd ACM SYMPOSIUM ON APPLIED COMPUTING, 2018, Pau. **Proceedings of the 33rd Annual ACM Symposium on Applied Computing.** New York: ACM Press, 2018. p. 1292 – 1299.
2. SILVA, João Pablo Silva da *et al*. A systematic literature review of UML-based domain-specific modeling languages for self-adaptive systems. In: 13th INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING FOR ADAPTIVE AND SELF-MANAGING SYSTEMS, 2018, Gothenburg. **Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems.** New York: ACM Press, 2018. p. 87 – 93.
3. SILVA, João Pablo Silva da *et al*. Towards a domain-specific modeling language for self-adaptive systems conceptual modeling. In:XXXII BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2018, São Carlos. **Proceedings of the XXXII Brazilian Symposium on Software Engineering.** New York: ACM Press, 2018. p. 87 – 93.
4. TORRES, Rafael; FRANZ, Luiz Paulo; SILVA, João Pablo Silva da. Um Editor para a Linguagem de Especificação de Requisitos RELAX. In: 1ª ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE, 2017, Alegrete. **Anais da 1ª Escola Regional de Engenharia de Software.** Alegrete: Unipampa, 2017. p. 33 – 40.

Term paper advisory:

1. AMARAL, Eduardo Florindo. **Uma Abordagem para Especificação de Requisitos Sensíveis ao contexto Baseado em Casos de Uso para Sistemas Autoadaptativos.** 2017. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2017.
2. FRANZ, Luiz Paulo. **Extração Automática de Modelos Conceituais a Partir de Requisitos para Sistemas Autoadaptativos.** 2017. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2017.
3. LIMA, Mário Alan de Oliveira. **Um Projeto de Interface para Ferramentas de**

Modelagem UML Baseadas em Eclipse. 2017. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2017.

4. MORO, Gabriel Bronzatti. **Uma Ferramenta de Apoio à Especificação de Requisitos para Sistemas Autoadaptativos.** 2015. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2015.

Not Related to the Thesis

Book chapters published:

1. TOLFO, Cristiano; SILVA, João Pablo Silva da. Ensino de Gestão de Projetos de Software Mediado pela Aprendizagem Baseada em Problemas. In: TOLFO, Cristiano (Org.). **Aprendizagem Baseada em Problemas na Engenharia de Software: Relatos de Experiência.** Bagé: Ediurcamp, 2018. p. 115 – 133.
2. SILVA, João Pablo Silva da *et al*. Aprendizagem Baseada em Problemas Aplicada no Ensino de Abordagens Ágeis. In: TOLFO, Cristiano (Org.). **Aprendizagem Baseada em Problemas na Engenharia de Software: Relatos de Experiência.** Bagé: Ediurcamp, 2018. p. 93 – 113.

Works published in proceedings:

1. ECAR, Miguel da Silva; Kepler, Fabio Natanael; SILVA, João Pablo Silva da. Cosmic User Story Standard. In: INTERNATIONAL CONFERENCE ON AGILE SOFTWARE DEVELOPMENT, 2018, Porto. **XP 2018: Agile Processes in Software Engineering and Extreme Programming.** Berlin: Springer, 2018.
2. ECAR, Miguel da Silva; SILVA, João Pablo Silva da; Kepler, Fabio Natanael. AutoCosmic: COSMIC Automated Estimation and Management Tool. In: XIV BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS, 2018, Caxias do Sul. **Proceedings of the XIV Brazilian Symposium on Information Systems.** New York: ACM Press, 2018.
3. MENEZES, Stefane *et al*. Empirical Evaluation of Formal Method for Requirements Specification in Agile Approaches. In: XIV BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS, 2018, Caxias do Sul. **Proceedings of the XIV Brazilian Symposium on Information Systems.** New York: ACM Press, 2018.
4. CHEIRAN, Jean Felipe Patikowski *et al*. Problem-Based Learning to Align Theory and Practice in Software Testing Teaching. In: 31st BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2017, Fortaleza. **Proceedings of the**

31st Brazilian Symposium on Software Engineering. New York: ACM Press, 2017. p. 328 – 337.

5. RODRIGUES, Peterson Luiz da Rosa *et al*. Coding Dojo as a transforming practice in collaborative learning of programming. In: 31st BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2017, Fortaleza. **Proceedings of the 31st Brazilian Symposium on Software Engineering.** New York: ACM Press, 2017. p. 348 – 357.
6. ECAR, Miguel da Silva; SILVA, João Pablo Silva da. Multivocal Literature Review on User Story Models for COSMIC Sizing. In: 1ª ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE, 2017, Alegrete. **Anais da 1ª Escola Regional de Engenharia de Software.** Alegrete: Unipampa, 2017. p. 41 – 48.
7. RODRIGUES, Peterson Luiz da Rosa *et al*. DevOps adoption in junior enterprise: an experience report of software development. In: 1ª ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE, 2017, Alegrete. **Anais da 1ª Escola Regional de Engenharia de Software.** Alegrete: Unipampa, 2017. p. 89 – 96.
8. KRUG, Thiago Cassio *et al*. SOFIA: Um Sistema de Suporte para as Inspeções de Software. In: 1ª ESCOLA REGIONAL DE ENGENHARIA DE SOFTWARE, 2017, Alegrete. **Anais da 1ª Escola Regional de Engenharia de Software.** Alegrete: Unipampa, 2017. p. 143 – 150.
9. RODRIGUES, Peterson Luiz da Rosa *et al*. Métodos Formais como Condutores para Especificação de Requisitos Aplicados em Metodologias Ágeis de Desenvolvimento. In: ESCOLA REGIONAL DE INFORMÁTICA, 2016, Rondonópolis. **Anais da Escola Regional de Informática.** Rondonópolis: UFMT, 2016.
10. SILVA, João Pablo Silva da et al. OntoQAI: An Ontology to Support Quality Assurance Inspections. In: 29st BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 2015, Belo Horizonte. **Proceedings of the 29st Brazilian Symposium on Software Engineering.** Washington: IEEE, 2015.

Term paper advisory:

1. ECAR, Miguel da Silva. **AutoCosmic: Platform for COSMIC Automated Estimation and Management.** 2017. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2017.
2. RODRIGUES, Peterson Luiz da Rosa. **Estudos Empíricos sobre a Aplicabilidade de Especificação Formal de Requisitos em Projetos Ágeis.** 2017. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa,

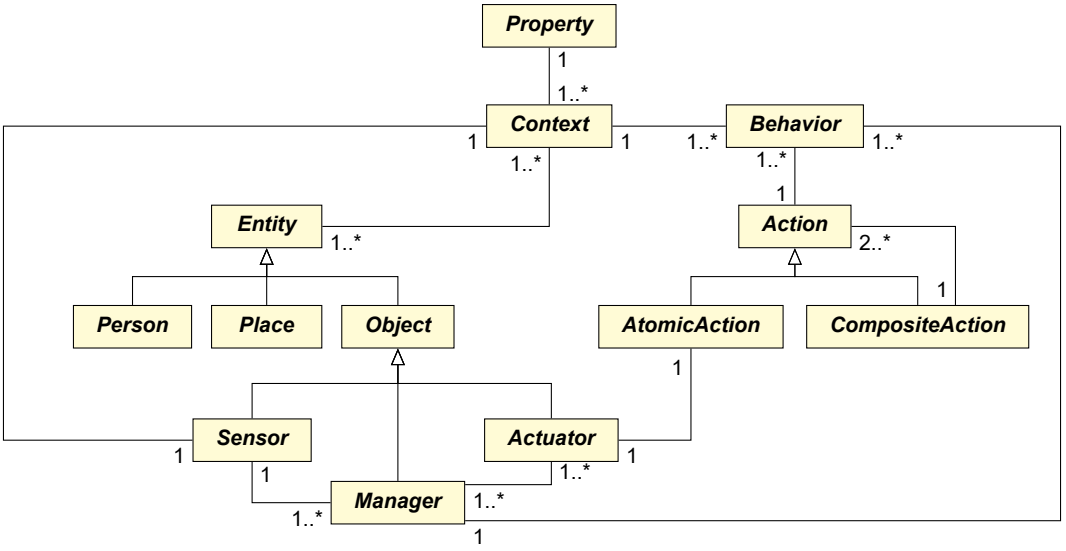
Alegrete, 2017.

3. CHAGAS, Jonas Maria. **Um Plano de Medição para as Disciplinas de Resolução de Problemas do Curso de Engenharia de Software.** 2016. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2016.
4. GIORDANO, Douglas Montanha. **UML Sketch Recognizer: Um aplicativo para reconhecimento de esboços de diagramas de classe em fotos.** 2015. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2015.
5. MACHADO, Ricardo Burg. **Uma Arquitetura Web para Sistemas de Informações Geográficas aplicada a Geotecnia.** 2015. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2015.
6. BRUNING, Eduardo. **Uma Ferramenta de Modelagem Colaborativa de Diagramas de Classes.** 2015. Term Paper – Software Engineering Undergraduate Program, Federal University of Pampa, Alegrete, 2015.

APPENDIX B — FOCUS GROUP INSTRUMENTS

First Session Instrument

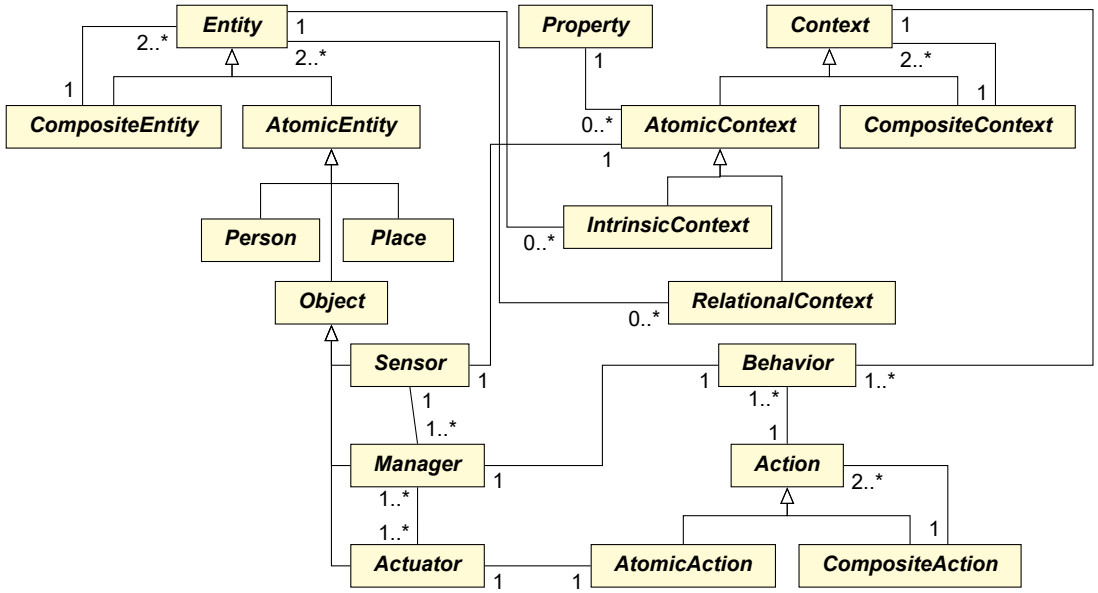
We present below a first self-adaptive systems conceptual modeling schema version. The group should evaluate and discuss the model correctness and completeness, using as criteria its own abilities and skills.



Group Findings

Second Session Instrument

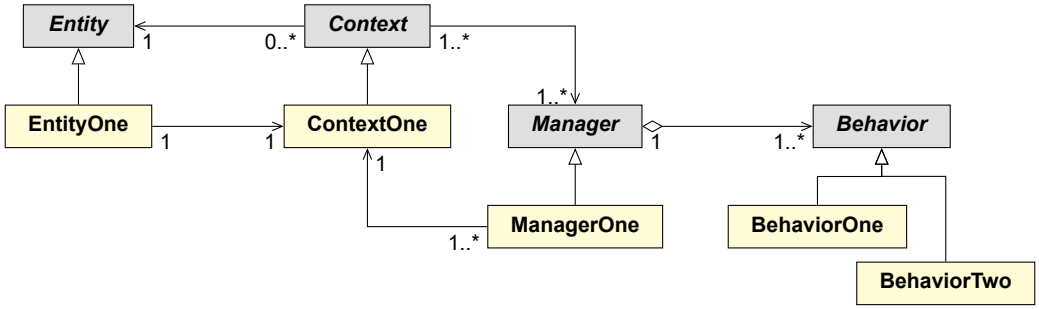
We present below a second self-adaptive systems conceptual modeling schema version. The group should evaluate and discuss the model correctness and completeness, using as criteria its own abilities and skills.



Group Findings

Third Session Instrument

We present below a third self-adaptive systems conceptual modeling schema version. The group should evaluate and discuss the model correctness and completeness, using as criteria its own abilities and skills.

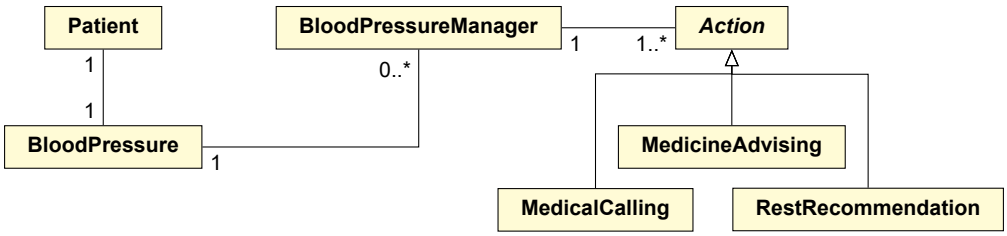


Group Findings

APPENDIX C — EXPERIMENT WITH SUBJECTS INSTRUMENTS

First Activity

The conceptual model below specifies the entities and relationship of a blood pressure monitoring requirement for a health care system. Analyze the model and answer the questions.



1. What items refers to contextual information?
   (a) BloodPressureManager
   (b) MedicineAdvising
   (c) Patient
   (d) BloodPressure
   (e) MedicalCalling
   (f) RestRecommendation
   (g) Action
2. What items refers to adaptive behavior?
   (a) RestRecommendation
   (b) Action
   (c) MedicalCalling
   (d) MedicineAdvising
   (e) BloodPressure
   (f) Patient
   (g) BloodPressureManager

Second Activity

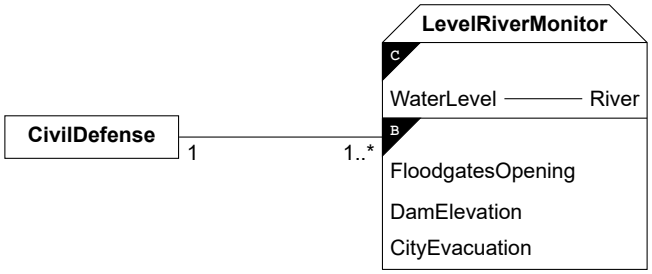
The requirement below specifies a learning support requirement for a smart tutor system. Analyze the requirement and formulate a conceptual model in UML notation.

The system SHALL provide support material to students according to their learning needs. It MAY provide courseware whenever identify student conceptual gaps, OR it MAY provide examples whenever are required student development activities.
ENV: Course teaching plan and Students performance.
MON: Courses tracker and Students tracker.
REL: Courses tracker provides Course teaching plan and Students tracker provides Students performance.
DEP: None.

Subject Answer

Third Activity

The conceptual model below specifies the entities and relationships of a river water level monitoring requirement for a smart city system. Analyze the model and answer the questions.



1. What items refers to contextual information?
    (a) CivilDefense
    (b) LevelRiverMonitor
    (c) WaterLevel
    (d) FloodgatesOpening
    (e) DamElevation
    (f) River
    (g) CityEvacuation
2. What items refers to adaptive behavior?
    (a) DamElevation
    (b) LevelRiverMonitor
    (c) CivilDefense
    (d) FloodgatesOpening
    (e) River
    (f) CityEvacuation
    (g) WaterLevel

Fourth Activity

The requirement below specifies an energy supply management for a company. Analyze the requirement and formulate a conceptual model in SaSML notation.

The system SHALL manage the energy supply to minimize the energy cost. It MAY turn on the power generator whenever consumption is low and tariff is high, OR it MAY buy electricity in any other situation.
ENV: Company energy consumption and Distributor energy tariff.
MON: Consumption tracker and Cost tracker.
REL: Consumption tracker provides Company energy consumption and Cost tracker provides Distributor energy tariff.
DEP: None.

Subject Answer

APPENDIX D — EXPERIMENT WITH SUBJECTS GUIDELINES

First Activity

Activity answers:

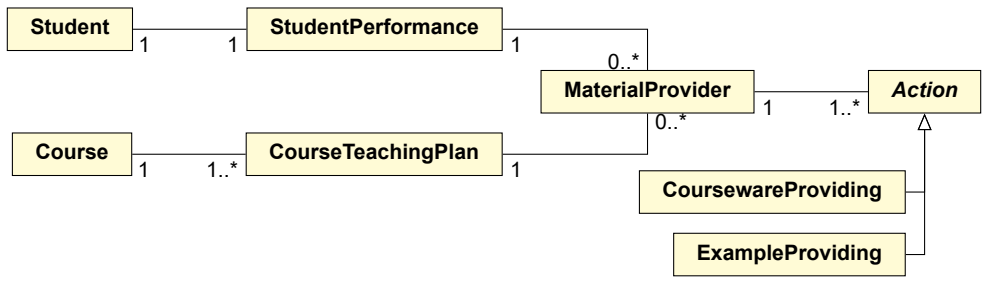
1. What items refers to contextual information?
    - BloodPressureManager
    - Patient
2. What items refers to adaptive behavior?
    - RestRecommendation
    - MedicalCalling
    - MedicineAdvising

Correction criteria:

Measure	Definition
True-positive	Items marked by the subject that exist in the answers.
False-positive	Items marked by the subject that do not exist in the answers.
False-negative	Items not marked by the subject that exist in the answers.

Second Activity

Activity answer:



Correction criteria:

Measure	Definition
True-positive	Items modeled by the subject that exist in the answer.
False-positive	Items modeled by the subject that do not exist in the answer.
False-negative	Items not modeled by the subject that exist in the answer.

Third Activity

Activity answers:

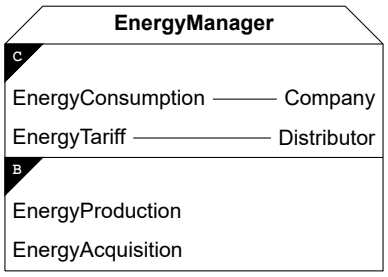
1. What items refers to contextual information?
    - WaterLevel
    - River
2. What items refers to adaptive behavior?
    - DamElevation
    - FloodgatesOpening
    - CityEvacuation

Correction criteria:

Measure	Definition
True-positive	Items marked by the subject that exist in the answers.
False-positive	Items marked by the subject that do not exist in the answers.
False-negative	Items not marked by the subject that exist in the answers.

Fourth Activity

Activity answers:



Correction criteria:

Measure	Definition
True-positive	Items modeled by the subject that exist in the answer.
False-positive	Items modeled by the subject that do not exist in the answer.
False-negative	Items not modeled by the subject that exist in the answer.

APPENDIX E — EXPERIMENT WITH SUBJECTS RESULTS DATA

First Activity

Subject	Blocking	TP	FP	FN	Precision	Recall	F-measure
S1	Third-year	5	2	0	0.71	1	0.83
S2	Fourth-year	1	2	4	0.33	0.2	0.25
S3	Fourth-year	5	3	0	0.63	1	0.77
S4	Third-year	4	1	1	0.8	0.8	0.8
S5	Fourth-year	5	2	0	0.71	1	0.83
S6	Fourth-year	5	2	0	0.71	1	0.83
S7	Third-year	4	3	1	0.57	0.8	0.67
S8	Fourth-year	4	2	1	0.67	0.8	0.73
S9	Third-year	1	1	4	0.5	0.2	0.29
S10	Third-year	5	5	0	0.5	1	0.67
S11	Third-year	3	4	2	0.43	0.6	0.5
S14	Third-year	0	5	5	0	0	0
S15	Third-year	0	4	5	0	0	0
S16	Fourth-year	4	1	1	0.8	0.8	0.8
S17	Fourth-year	5	2	0	0.71	1	0.83
S18	Third-year	2	4	3	0.33	0.4	0.36
S19	Fourth-year	5	1	0	0.83	1	0.91
S20	Fourth-year	5	2	0	0.71	1	0.83
S21	Fourth-year	5	2	0	0.71	1	0.83
S22	Fourth-year	5	2	0	0.71	1	0.83
S23	Fourth-year	5	2	0	0.71	1	0.83
S25	Fourth-year	4	1	1	0.8	0.8	0.8
S26	Third-year	5	0	0	1	1	1
S27	Fourth-year	4	4	1	0.5	0.8	0.62
S28	Third-year	2	2	3	0.5	0.4	0.44
S29	Third-year	5	3	0	0.63	1	0.77
S30	Third-year	4	2	1	0.67	0.8	0.73
S31	Fourth-year	2	5	3	0.29	0.4	0.34
S32	Third-year	1	1	4	0.5	0.2	0.29
S34	Third-year	4	3	1	0.57	0.8	0.67
S35	Third-year	5	1	0	0.83	1	0.91
S36	Fourth-year	5	0	0	1	1	1
S38	Third-year	5	5	0	0.5	1	0.67
S39	Fourth-year	4	1	1	0.8	0.8	0.8
S40	Third-year	5	2	0	0.71	1	0.83
S41	Fourth-year	1	1	4	0.5	0.2	0.29

True-positive (TP), False-positive (FP), and False-negative (FN).

Second Activity

Subject	Blocking	TP	FP	FN	Precision	Recall	F-measure
S1	Third-year	12	3	3	0.8	0.8	0.8
S2	Fourth-year	8	6	7	0.57	0.53	0.55
S3	Fourth-year	5	6	10	0.45	0.33	0.38
S4	Third-year	6	12	9	0.33	0.4	0.36
S5	Fourth-year	4	12	11	0.25	0.27	0.26
S6	Fourth-year	1	8	14	0.11	0.07	0.09
S7	Third-year	1	11	14	0.08	0.07	0.07
S8	Fourth-year	3	10	12	0.23	0.2	0.21
S9	Third-year	3	6	12	0.33	0.2	0.25
S10	Third-year	1	12	14	0.08	0.07	0.07
S11	Third-year	3	8	12	0.27	0.2	0.23
S14	Third-year	6	13	9	0.32	0.4	0.36
S15	Third-year	6	11	9	0.35	0.4	0.37
S16	Fourth-year	3	4	12	0.43	0.2	0.27
S17	Fourth-year	2	12	13	0.14	0.13	0.13
S18	Third-year	8	3	7	0.73	0.53	0.61
S19	Fourth-year	4	7	11	0.36	0.27	0.31
S20	Fourth-year	5	6	10	0.45	0.33	0.38
S21	Fourth-year	7	9	8	0.44	0.47	0.45
S22	Fourth-year	9	7	6	0.56	0.6	0.58
S23	Fourth-year	11	3	4	0.79	0.73	0.76
S25	Fourth-year	9	7	6	0.56	0.6	0.58
S26	Third-year	1	4	14	0.2	0.07	0.1
S27	Fourth-year	3	13	12	0.19	0.2	0.19
S28	Third-year	2	10	13	0.17	0.13	0.15
S29	Third-year	5	7	10	0.42	0.33	0.37
S30	Third-year	1	11	14	0.08	0.07	0.07
S31	Fourth-year	7	9	8	0.44	0.47	0.45
S32	Third-year	1	6	14	0.14	0.07	0.09
S34	Third-year	8	10	7	0.44	0.53	0.48
S35	Third-year	6	9	9	0.4	0.4	0.4
S36	Fourth-year	3	12	12	0.2	0.2	0.2
S38	Third-year	1	8	14	0.11	0.07	0.09
S39	Fourth-year	4	11	11	0.27	0.27	0.27
S40	Third-year	9	6	6	0.6	0.6	0.6
S41	Fourth-year	2	9	13	0.18	0.13	0.15

True-positive (TP), False-positive (FP), and False-negative (FN).

Third Activity

Subject	Blocking	TP	FP	FN	Precision	Recall	F-measure
S1	Third-year	5	1	0	0.83	1	0.91
S2	Fourth-year	5	0	0	1	1	1
S3	Fourth-year	5	0	0	1	1	1
S4	Third-year	5	0	0	1	1	1
S5	Fourth-year	5	0	0	1	1	1
S6	Fourth-year	3	4	2	0.43	0.6	0.5
S7	Third-year	2	1	3	0.67	0.4	0.5
S8	Fourth-year	5	0	0	1	1	1
S9	Third-year	5	0	0	1	1	1
S10	Third-year	5	2	0	0.71	1	0.83
S11	Third-year	5	0	0	1	1	1
S14	Third-year	5	0	0	1	1	1
S15	Third-year	5	0	0	1	1	1
S16	Fourth-year	5	1	0	0.83	1	0.91
S17	Fourth-year	4	0	1	1	0.8	0.89
S18	Third-year	5	1	0	0.83	1	0.91
S19	Fourth-year	5	0	0	1	1	1
S20	Fourth-year	5	0	0	1	1	1
S21	Fourth-year	5	0	0	1	1	1
S22	Fourth-year	5	0	0	1	1	1
S23	Fourth-year	5	0	0	1	1	1
S25	Fourth-year	4	0	1	1	0.8	0.89
S26	Third-year	5	0	0	1	1	1
S27	Fourth-year	4	0	1	1	0.8	0.89
S28	Third-year	5	0	0	1	1	1
S29	Third-year	2	1	3	0.67	0.4	0.5
S30	Third-year	5	0	0	1	1	1
S31	Fourth-year	4	0	1	1	0.8	0.89
S32	Third-year	2	1	3	0.67	0.4	0.5
S34	Third-year	4	0	1	1	0.8	0.89
S35	Third-year	5	0	0	1	1	1
S36	Fourth-year	5	0	0	1	1	1
S38	Third-year	5	2	0	0.71	1	0.83
S39	Fourth-year	5	0	0	1	1	1
S40	Third-year	5	0	0	1	1	1
S41	Fourth-year	1	1	4	0.5	0.2	0.29

True-positive (TP), False-positive (FP), and False-negative (FN).

Fourth Activity

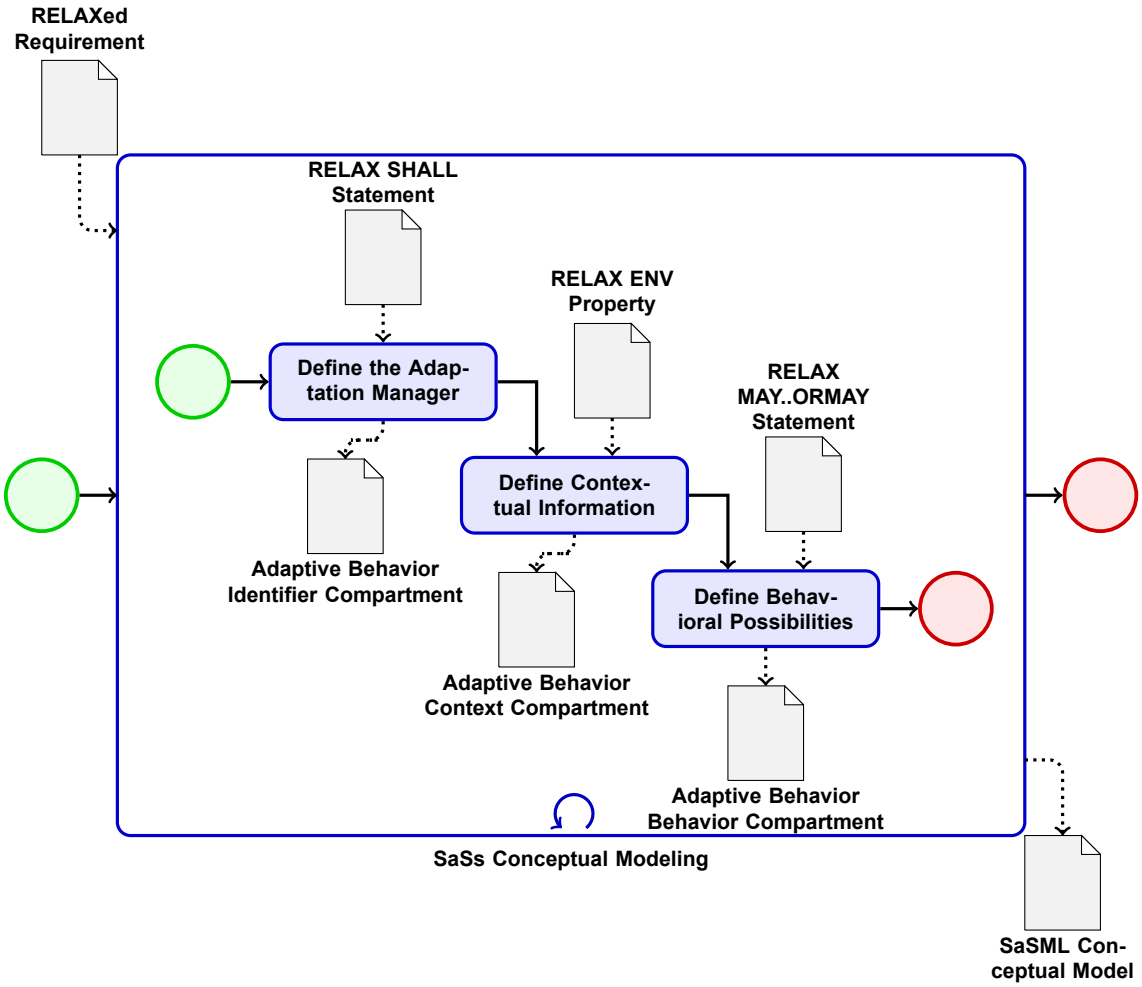
Subject	Blocking	TP	FP	FN	Precision	Recall	F-measure
S1	Third-year	3	13	6	0.19	0.33	0.24
S2	Fourth-year	9	0	0	1	1	1
S3	Fourth-year	5	4	4	0.56	0.56	0.56
S4	Third-year	3	9	6	0.25	0.33	0.28
S5	Fourth-year	5	4	4	0.56	0.56	0.56
S6	Fourth-year	0	11	9	0	0	0
S7	Third-year	5	3	4	0.63	0.56	0.59
S8	Fourth-year	9	0	0	1	1	1
S9	Third-year	9	0	0	1	1	1
S10	Third-year	3	14	6	0.18	0.33	0.23
S11	Third-year	5	6	4	0.45	0.56	0.5
S14	Third-year	6	6	3	0.5	0.67	0.57
S15	Third-year	0	13	9	0	0	0
S16	Fourth-year	3	7	6	0.3	0.33	0.31
S17	Fourth-year	3	6	6	0.33	0.33	0.33
S18	Third-year	5	0	4	1	0.56	0.72
S19	Fourth-year	9	4	0	0.69	1	0.82
S20	Fourth-year	9	0	0	1	1	1
S21	Fourth-year	7	6	2	0.54	0.78	0.64
S22	Fourth-year	5	8	4	0.38	0.56	0.45
S23	Fourth-year	5	3	4	0.63	0.56	0.59
S25	Fourth-year	5	8	4	0.38	0.56	0.45
S26	Third-year	9	0	0	1	1	1
S27	Fourth-year	5	3	4	0.63	0.56	0.59
S28	Third-year	5	3	4	0.63	0.56	0.59
S29	Third-year	3	3	6	0.5	0.33	0.4
S30	Third-year	4	8	5	0.33	0.44	0.38
S31	Fourth-year	9	0	0	1	1	1
S32	Third-year	1	9	8	0.1	0.11	0.1
S34	Third-year	4	7	5	0.36	0.44	0.4
S35	Third-year	9	0	0	1	1	1
S36	Fourth-year	9	6	0	0.6	1	0.75
S38	Third-year	4	2	5	0.67	0.44	0.53
S39	Fourth-year	9	0	0	1	1	1
S40	Third-year	9	0	0	1	1	1
S41	Fourth-year	3	6	6	0.33	0.33	0.33

True-positive (TP), False-positive (FP), and False-negative (FN).

APPENDIX F — ADAPTIVE BEHAVIOR MODELING PROCESS

We propose a process that drives the software engineers work during the requirements analysis. It establishes an iterative and incremental cycle, which receives a requirement written in RELAX and produces a conceptual model specified in SaSML.

The process tasks define the procedures to extract from a requirement specification the information required to specify an Adaptive Behavior instance. Each task receives as input a requirement specification fragment and produces as output an Adaptive Behavior compartment.



Task 1: Define the Adaptation Manager

The first task aims to extract the adaptation manager from the requirement. It receives as input a RELAX SHALL Statement and produces as output the Adaptive Behavior identifier compartment. The task steps are:

1. read the RELAX SHALL statement text;
2. identify the requirement main goal;
3. create an Adaptive Behavior element;
4. define a identifier that expresses the requirement goal;
5. write the identifier in Adaptive Behavior identifier compartment.

Task 2: Define Contextual Information

The second task aims to extract the contextual information from the requirement. It receives as input a RELAX ENV Property and produces as output the Adaptive Behavior context compartment. The task steps are:

1. read the RELAX ENV property text;
2. identify the contexts that define entities situation;
3. identify the entities related to the contexts;
4. write the context and entities in Adaptive Behavior context compartment;
5. link each context to its respective entity.

Task 3: Define Behavioral Possibilities

The third task aims to extract the behavioral possibilities from the requirement. It receives as input a RELAX MAY..ORMAY Statement and produces as output the Adaptive Behavior behavior compartment. The task steps are:

1. read the RELAX MAY..ORMAY statement texts;
2. identify the behavioral possibilities;
3. write a behavior for each possibility in Adaptive Behavior behavior compartment.