



HAL
open science

A context manager for solving conflicts between designer's viewpoints

The Can Do

► **To cite this version:**

The Can Do. A context manager for solving conflicts between designer's viewpoints. Ubiquitous Computing. COMUE Université Côte d'Azur (2015 - 2019), 2019. English. NNT : 2019AZUR4106 . tel-02884122

HAL Id: tel-02884122

<https://theses.hal.science/tel-02884122>

Submitted on 29 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Un manager de contextes résolvant les conflits entre les points de vue des concepteurs

The Can DO

Université Côte d'Azur, CNRS, I3S

**Présentée en vue de l'obtention
du grade de docteur en
informatique d'Université Côte
d'Azur**

Dirigée par : Nhan Le-Thanh

Co-encadrée par : Gaëtan Rey

Soutenue le : 16/12/2019

Devant le jury, composé de :

Patrick Brézillon, Professeur, Sorbonne Université

Parisa Ghodous, Professeur, Université Lyon 1

Thanh Binh Nguyen, Assoc. Prof, Danang Université

Peter Sander, Professeur, Université Côte d'Azur

Gaëtan Rey, Maître de conférences, Université Côte d'Azur

Jean-Yves Tigli, Maître de conférences, Université Côte
d'Azur

Un manager de contextes résolvant les conflits entre les points de vue des concepteurs

Jury :

Président du jury*

Peter Sander, Professeur,
Université Côte d'Azur, CNRS
Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis (I3S)

Rapporteurs

Patrick Brézillon, Professeur,
Sorbonne Université, Paris, France,
Laboratoire d'Informatique de Paris 6 (LIP6)

Thanh Binh Nguyen, Associate Professor,
Université de Danang, Vietnam
Faculté d'informatique

Examineurs

Parisa Ghodous, Professeur,
Université Claude Bernard Lyon 1, France
Laboratoire d'InfoRmatique en Image et Systèmes d'information

Gaëtan Rey, Maître de conférences,
Université Côte d'Azur, CNRS
Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis (I3S)

Jean-Yves Tigli, Maître de conférences,
Université Côte d'Azur, CNRS
Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis (I3S)

Un manager de contextes résolvant les conflits entre les points de vue des concepteurs

Résumé

De nos jours, les systèmes auto-adaptatifs (AAS) sont devenus des éléments essentiels pour piloter les applications dans le domaine de l'informatique ambiante. Ils offrent des capacités d'autoadaptation dynamique de leurs comportements en réponse aux situations contextuelles et aux besoins actuels de l'utilisateur.

Cette thèse vise à proposer des solutions pour surmonter les défis de développement des systèmes auto-adaptatifs tels que la modélisation et la gestion du contexte pour ces systèmes et l'amélioration de leur capacité d'adaptation. L'étude se concentre sur deux aspects : la séparation des préoccupations contextuelles dans la modélisation du contexte et l'amélioration de la réactivité du système auto-adaptatif. Pour séparer les préoccupations contextuelles, nous introduisons la notion de points de vue indépendants et fournissons un mécanisme pour les utiliser dans le processus de modélisation du contexte. Ce mécanisme simplifie également la tâche du concepteur car il lui permet de se concentrer uniquement sur son domaine d'expertise pour modéliser sa préoccupation contextuelle (son point de vue). Pour améliorer la réactivité de SAS, nous fournissons un nouveau modèle d'architecture utilisant un manager de contextes (MC) pour prendre en charge le système auto-adaptatif dans le traitement du contexte. Dans cette architecture, le MC se concentre sur l'identification de la situation contextuelle actuelle, en fonction des points de vue spécifiques dont il a la charge, dans le but de déployer les règles d'adaptation adapter à cette situation. Le SAS n'a alors plus qu'à gérer un ensemble de règles limité, déjà adapté à la situation courante.

Dans notre approche, chaque point de vue spécifique a été construit indépendamment des autres par des concepteurs différents. Chaque point de vue correspond à une préoccupation particulière, et il convient d'utiliser plusieurs points de vue en même temps pour répondre à tous les scénarios usages. Par conséquent, les possibilités de conflit entre les points de vue, à un moment donné, existent. Néanmoins, il est impossible de résoudre les conflits entre les points de vue au moment de la conception parce que nous ne sommes pas en mesure de prévoir tous les scénarios d'usages, les combinaisons de points de vue dépendant de l'emplacement, de l'activité, de la réglementation et des choix de l'utilisateur. C'est pourquoi nous proposons plusieurs solutions pour détecter et résoudre les conflits entre les points de vue dans le manager de contextes.

Mots clés : Informatique ambiante, système auto-adaptatifs, modélisation du contexte gestion du contexte, résolution de conflits

A context manager for solving conflicts between designer's viewpoints

Abstract

Presently, self-adaptive systems (SAS) have become an essential feature in ubiquitous computing applications. They provide a capable of dynamically self-adapting their behavior in response to the user's current situation and needs.

This thesis aims at providing several solutions for overcoming challenges on the development of self-adaptive system such as context modeling, context handling and improving adaptation ability. Our study focuses on two aspects: separating contextual concerns in context modeling and improving the responsiveness of self-adaptive system. To separate contextual concerns, we introduce the notion of independent viewpoints and provide a mechanism to use them for the context modeling process. This mechanism also simplifies the designers' task because it allows the designers to focus only on their expertise domain to modeling their contextual concern. To improve the responsiveness of SAS, we provide a new architectural pattern using context-aware management (CAM) to support self-adaptive system in handling context. In this architecture, the CAM focuses on the current context identification according to the specific viewpoints which it is in charge of, having the goal to deploy the adaptation rules to this situation. The SAS only has to manage a limited set of rules, already adapted to the current situation.

In our approach, each specific viewpoint was built independently by different designers. Each viewpoint is related to a different scenario setup, but they might operate in one system at the same time. Therefore, the possibilities of conflict between viewpoints at one time always exist. Nevertheless, it is impossible to solve conflicts between viewpoints at design time because we are not able to predict all the users' scenarios and the combinations of viewpoints because they depend on the location, activity, regulation and user's choice. Therefore, we propose several solutions to detect and solve the conflicts between viewpoints in the context-aware management layer.

Keywords: Ubiquitous computing, self-adaptive system, context modeling, context manager, conflict resolution.

Acknowledgments

I would really like to express my profound gratitude to all those who have helped me throughout the Ph.D. period, both directly and indirectly.

First and foremost, I would like to express my sincere appreciation to my supervisor Prof. Nhan Le-Thanh, who guided me throughout my Ph.D. study. I thank him for his invaluable guidance, supervision, and continuous support during the past four years.

I would like to thank my principal advisor, Assoc. prof. Gaëtan Rey, who provided scientific guidance, tireless support, encouragement, and inspiration throughout my Ph.D. study. His guidance helped me in overcoming obstacles I have been facing during my research and writing of this thesis. Without his assistance in every step throughout my research process, my thesis would have never been accomplished.

My sincere thanks also to Assoc. prof. Jean-Yves Tigli. The door to his office was always open whenever I had any questions about my research or thesis writing. So, I want to say to him: “thank you for your unconditional support, encouragement, shared knowledge, and thoughtful guidance.”

I wish to extend my deep appreciation to the reviewers, Prof. Patrick Brézillon and Assoc. prof. Thanh Binh Nguyen, who gave their time and attention to reviewing my manuscript and evaluating my work. Their invaluable comments and advice helped me in improving the quality of my manuscript and overcoming the research limitations. I am deeply honored to have Prof. Parisa Ghodous and Prof. Sander Peter as members of my Ph.D. committee. I would like to thank them for accepting the invitation.

Many thanks to the Vietnam Ministry of Education and Training (MOET) and Campus France for granting me financial support for the Ph.D. studies for four years, and provided me an opportunity to live and study in France.

I am indebted to all my fellow doctoral students and friends in Sophia Antipolis for their continuous support, cooperation, and of course friendship. In addition, I would like to thank my best friend, Huu Duyen Nguyen, for her enthusiastic support.

Finally, I must express my very profound gratitude to my family for being my biggest supporters, especially during many years of study. Most importantly, huge thank you to my wife, Thanh Tra Phan, for her unconditional support and for always believing in me. To her, I would like to dedicate not only this thesis but also my heart.

The Can DO

Nice, December 2019

Contents

List of Figures	ix
List of Tables.....	xi
CHAPTER 1: Introduction.....	1
1.1 Ubiquitous computing	1
1.2 Context-aware and mobility of the user.....	3
1.2.1 What is Context?.....	3
1.2.2 Context-aware system.....	4
1.2.3 The mobility of user and changes of contextual information	5
1.3 Overview of challenges.....	6
1.4 Outline	8
CHAPTER 2: Background and State of the Art	9
2.1 Context modeling	9
2.1.1 Context metamodel	10
2.1.2 Context ontology model.....	11
2.1.3 Graphical model.....	13
2.1.4 Discussion	15
2.2 Adaptation system.....	17
2.2.1 Context toolkit	18
2.2.2 Adaptation middleware.....	18
2.2.3 Context-aware system.....	20
2.2.4 Self-adaptive system	24
2.2.5 Discussion	25
2.3 Chapter conclusion	27
CHAPTER 3: Context intermediate model and Architecture pattern	29
3.1 Context-aware management (CAM)	29
3.2 Context model solution.	31
3.2.1 Problem with designing a single big context model	31
3.2.2 Separate concerns: the solution to solve the problems related to designing a single big context model	32
3.3 Special viewpoints in context domain	33
3.4 Using special viewpoints to separate concerns in context modeling.	36

3.5 Activity design: Architecture of self-adaptive system and context intermediate model.....	37
3.5.1 The model architecture of self-adaptive system	37
3.5.2 Context intermediate model	38
3.6 The methods of generating the context model from the models of concerns	42
3.6.1 Generating the context model with the Business process viewpoint	42
3.6.2 Generating the context model with Tasks model viewpoint	46
3.7 Valid and Invalid state.....	50
3.7.1 Problems related to observation	50
3.7.2 The state identification	50
3.7.3 The proposed solution to filter states and react to invalid states.....	52
3.8 Chapter conclusion	53
CHAPTER 4: Conflict detection and solution.....	55
4.1 Conflicts between adaptation rules of viewpoints in self-adaptive system.....	55
4.2 State of the art	57
4.3 Objectives of state definition	60
4.4 Detect the conflicts between the objective of states in different viewpoints	64
4.4.1 State of the art	64
4.4.2 Conflict detection algorithms	66
4.5 Chapter conclusion	73
CHAPTER 5: Solving conflicts between objective of states	75
5.1 Introduction	75
5.2 State of the art	76
5.3 Using priority to classify the objective of states	77
5.4 Conflict resolution	80
5.4.1 Solving conflict between viewpoints based on priority level.....	81
5.4.2 Solving conflict based on getting maximum of services	86
5.4.3 Solving conflict based on the combination of priority and maximum supported services.....	90
5.5 Chapter conclusion	92
CHAPTER 6: Adaptation rules deployment and Implementations.....	94
6.1 Introduction	94

6.2. Objective designer database	97
6.3 Adaptation rules deployment	97
6.3.1 First solution	98
6.3.2 Second solution.....	99
6.3.3 Third solution.....	100
6.4 Implementations	101
6.4.1 Tools for designer.....	102
6.4.2 Context-aware management (CAM).....	103
6.4.3 The simulator of self-adaptive system and observation system.....	106
6.5 Chapter conclusion	107
CHAPTER 7: Conclusion and Future works.....	109
7.1 General considerations	109
7.2 Research contributions	110
7.2.1 Simplifying the designer’s task	110
7.2.2 Detecting and solving the conflicts between viewpoints	111
7.2.3 Improve reactivity of SAS	112
7.3 Future works	112
Bibliography	114
Appendix	125

List of Figures

Figure 1.1: User applications in ubiquitous computing environments	2
Figure 1.2: The adaptation to the environment using self-adaptive system.....	2
Figure 1.3: User mobility in dynamic environments.....	5
Figure 1.4: The classical model of a self-adaptive system	6
Figure 2.1: Context meta-model [Shishkov-2018].....	10
Figure 2.2: CAMN: Context metamodel on TriPlet.....	11
Figure 2.3: The CONON upper ontology for smart home application.....	12
Figure 2.4: Hierarchical design of a Context ontology [Aguilar-2018].....	13
Figure 2.5: The graphical context modeling follows fact type [Henricksen-2003]	14
Figure 2.6: Architecture of an adaptation system [Keling-2011]	17
Figure 2.7: Architecture diagram of Context Toolkit	18
Figure 2.8: The SOCAM architecture	19
Figure 2.9: Context and user driven self-adaptation [Tigli-2009]	20
Figure 2.10: Context Broker architecture CoBra	22
Figure 2.11: Context-aware system [Mahmud-2018]	23
Figure 2.12: Taxonomy of Self-adaptive system architecture [Krupitzer-2015].....	25
Figure 3.1: Generic context manager main functionalities [Vieira-2007]	30
Figure 3.2: Context-aware management structure [Kim-2008].....	31
Figure 3.3: A single big context model in self-adaptive system	32
Figure 3.4: Using CIM as standard data for all viewpoints.....	33
Figure 3.5: The procedure for managing architectural contests [Marcinkowski-2012].	34
Figure 3.6: Use of task models in the design cycle [Paternò-2002]	35
Figure 3.7: Using special viewpoints in context modeling.....	36
Figure 3.8: Model architecture of system.....	37
Figure 3.9: The data flow of viewpoint in CAM.....	38
Figure 3.10: Each viewpoint is managed independently in CAM.....	39
Figure 3.11: The XML schema of context Intermediate model.....	41
Figure 3.12: Using BPMN to describe specific viewpoint in context modeling	43
Figure 3.13: The E-Order process that is described by graphical notation.....	44
Figure 3.14: Using CTT to describe specific viewpoint in context modeling	46
Figure 3.15: The UML notation for activity diagram [Nóbrega-2005].....	47
Figure 3.16: The Hotel reservation process that is described by ConcurTaskTrees.....	48
Figure 3.17: The updated version of CIM with adding Invalid state	51
Figure 3.18: The filter state in CAM.....	52
Figure 3.19: Detection Invalid states algorithm	53
Figure 4.1: The conflict management in Context-aware management (CAM).....	56
Figure 4.2: Ontological context-aware using 5W&1H question [Kim-2012].....	58
Figure 4.3: The 5W1H top-level classes [Rathi-2018].....	59

Figure 4.4: The 2W1H description of the objective of state	60
Figure 4.5: The 2W1H contextual ontology based on 5W1H-STPO [Yang-2011].....	61
Figure 4.6: Contextual ontology of “where” class [Rey-2005]	62
Figure 4.7: The modified of class “what” and “how” of CA _{5W1H} Onto.....	62
Figure 4.8: Adding the Objectives of state in the schema of CIM	63
Figure 4.9: Conflict analysis between two actions [Maternaghan-2013].....	65
Figure 4.10: Conflict ability checking process	66
Figure 4.11: The relationship between “what” elements of two OOSs (O ₁ (a), O ₂ (a)) ...	67
Figure 4.12: The relationship between “where” elements of two OOSs (O ₁ (b), O ₂ (b)) .	68
Figure 4.13: The relationship between “how” elements of two OOSs (O ₁ (c), O ₂ (c))	68
Figure 4.14: Relation properties between “How” elements in 2W1H ontology.....	69
Figure 4.15: The conflict situations when space O ₁ (b) cover a part or all space O ₂ (b) ..	71
Figure 5.1: The conflict relation of two OOSs	75
Figure 5.2: The objective of state description with 2W1H&1P	78
Figure 5.3: The updated version of Contextual ontology database 2W1H&1P	79
Figure 5.4: Classifying and solving conflict between multiple OOSs	80
Figure 5.5: The input and output of the conflict solving process.....	93
Figure 6.1: The output of context-aware management.	94
Figure 6.2: The final version of CIM schema	96
Figure 6.3: Objective designer database.....	97
Figure 6.4: Requesting AR for OOS in CAM	98
Figure 6.5: Mapping OOS to AR by using objective designer database.....	99
Figure 6.6: Using the local database to save time in compute order for SAS.....	100
Figure 6.7: The CAM associate AR for all OOSs automatically	101
Figure 6.8: The specific tool to convert BPMN viewpoint into CIM format.....	102
Figure 6.9: The specific tool to convert CTT viewpoint into CIM format.....	102
Figure 6.10: The XML validator tool.....	103
Figure 6.11: The architecture of CAM for implementation	103
Figure 6.12: The CAM life cycle	104
Figure 6.13: The viewpoint life cycle	105
Figure 6.14: The observation simulator interface	106
Figure 6.15: The adaptation simulator interface.....	107
Figure 7.1: The diagram of context description in our approach.....	110
Figure 7.2: Two independent cycles of observation of SAS.	112

List of Tables

Table 2.1: Evaluating current context modeling initiatives	16
Table 2.2: Evaluating current adaptation system approaches.....	27
Table 3.1: The mapping from BPMN to CIM standard data type.....	43
Table 4.1: Objectives conflict analysis results	70
Table 4.2: Objectives conflict analysis of O_1 and O_2	70
Table 4.3: An example of the conflict situation between eight OOS in one system	73
Table 5.1: Sample Priority matrix [Glaser-2015]	76
Table 5.2: Default priority and role schema.....	77
Table 5.3: The four levels of priority in the description of the OOS.....	78
Table 5.4: The priority level of OOS at the current time.	80
Table 5.5: The classifying OOS based on priority level.	80
Table 5.6: The result of checking “same OOS” in each priority level	82
Table 5.7: Conflict situation and the result of checking process in step 2	83
Table 5.8: The updated table of OOSs in four priority levels.....	84
Table 5.9: The number of conflicts in class C_{P2}	84
Table 5.10: The conflict situation of O_4 and O_8 with other OOSs in the lower level	84
Table 5.11: The result of the conflict checking and solving in class C_{P2}	85
Table 5.12: The result of the conflict checking and choice OOS for adaptation.....	85
Table 5.13: The priority levels and conflict situations of eight OOS.	87
Table 5.14: The number of the conflicts of each OOS with others.	87
Table 5.15: Conflict situations without O_5	88
Table 5.16: The conflict solving without O_5 and O_6	88
Table 5.17: The conflict solving without O_7	89
Table 5.18: The result of solving conflict between OOSs follow solution two.	89
Table 5.19: The conflict situations and priority levels of OOSs.....	91
Table 5.20: The conflict number of each OOS with the other OOSs	91
Table 5.21: The conflict situation without O_1	92
Table 5.22: Checking conflict for lower priority levels	92

CHAPTER 1: Introduction

1.1 Ubiquitous computing

During the past four decades, the rapid development of semiconductors and communication technology has created a revolution in embedded and mobile devices. It promotes a growing trend of embedding computational capacity into everyday objects to make them effectively communicate and perform a useful task in a way that minimizes the interaction required between the end-user and computers. Moreover, the advances in sensor technology, wireless communication, and information infrastructures such as GPS, Wi-Fi, and mobile device technology expose users to an enormous amount of mobile services available anywhere and anytime. The rapid development of embedded devices, mobile services, and communication technology make computation seem to appear everywhere.

The term “Ubiquitous Computing” was first introduced by Weiser [Weiser-1993], he defined that “*ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.*” According to Weiser, we are able to understand that ubiquitous computing is the integration of microcomputers into physical objects of any shape such as embedded systems, mobile devices, *etc.* Ubiquitous computing becomes useful tools but an invisible force to the user because they provide a list of services at the center of interactions between application layer and devices. Moreover, the main power of ubiquitous computing comes from the interconnection between many different kinds of individual computing devices. They communicate with each other through wireless or cable networks and use local information to adapt to the requirement of the user. On the other hand, ubiquitous computing is depicted by the omnipresent and mobile accessibility of services themselves. These services do not depend on the target platform; they will be automatically tailored to the physical capacity of a target device, whether it is a smartphone, embedded system, or other mobile devices.

The power of ubiquitous computing has changed the way we interact with other people or digital devices in physical spaces [Sen-2010]. The user can use ubiquitous devices for many different purposes such as entertainment, business, group activity, *etc.* (as shown in figure 1.1) at anytime and anyplace.

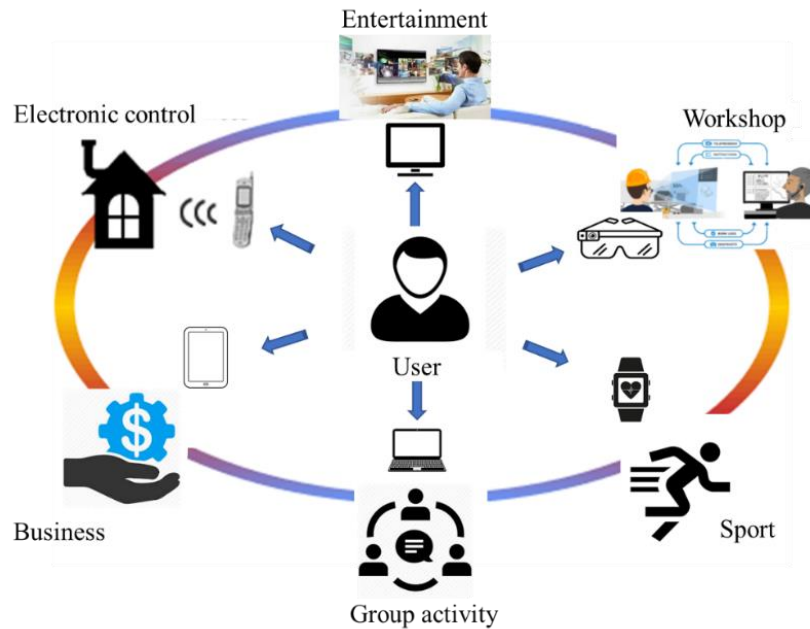


Figure 1.1: User applications in ubiquitous computing environments

Moreover, ubiquitous computing has also changed the ways we use the computer in our daily life. The users do not need to supply explicit instructions or make decisions because it has a variety of self-adaptive systems (SAS) operates automatically in the background and interacting on behalf of the user [Baldauf-2007]-[Miraoui-2008], as shown in figure 1.2.

We might see that ubiquitous computing is not only a special field of technology but also an application of information and communications technology. It is integrated into our daily lives more than ever before. The main goal of this integration is to adapt the requirements of users everywhere and at any time. Ubiquitous devices use sensors and communication protocols to collect information of both their users and the environment and then adjust their behavior accordingly.

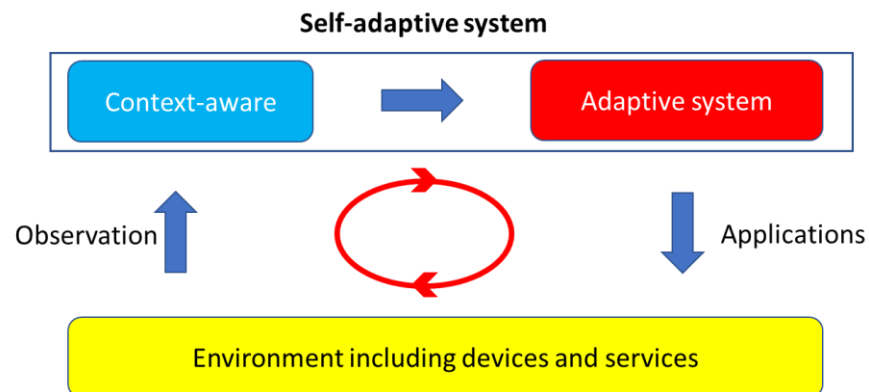


Figure 1.2: The adaptation to the environment using self-adaptive system

1.2 Context-aware and mobility of the user

1.2.1 What is Context?

Currently, it is impossible to expect a static execution environment for computer applications because the world around an application is constantly changing. In each situation, the information on the ubiquitous environment is different and changes over time. In such a dynamic environment, the interaction and management of all various devices that a user may be using will be a difficult task. Moreover, when we talk about the user's mobility, we need to take into account the changing of the factors affecting user applications such as environment, connection, situation, devices, etc. Many works have been done to define the term "Context" in various domains of sciences such as economy, computer science, philosophy, etc. Bazire et al. provided an analysis of a database with 150 context definitions to identify the main elements of the context [Bazire-2005]. Their work might help us understand the context before using it.

In this thesis, we only focus on the definitions of context in the domain of computer science. It is not difficult to find many different definitions of context from Web sources. However, we only point out some typical definitions accepted by the majority of researchers. Hull believed that the context simply is *"the aspects of a current situation"* [Hull-1997]. Schilit indicated context is *"the set of location, identities of nearby people and objects and changes to those objects"* [Schilit-1995]. Concurrent view with Schilit, Brown said that context is *"location, identities of the people around the user, the time of day"* [Brown-1997]. While Brézillon defined that *"Context is what constrains a problem solving without intervening in it explicitly"* [Brézillon-1999]. On the other hand, Henricksen suggested that the context as *"the set of circumstances surrounding it are of relevance to its completion"* [Henricksen-2002]. In addition, Chen et al. defined context is *"extending to model the activities and tasks that are taking place in a location"* [Chen H.-2004]. Also, we highlight that the context definition of Dey is the most accepted one, he considered context as *"any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves"* [Dey A.-2001].

We have many definitions of context because each researcher has their definition own way. Nevertheless, we also see that most of these definitions lack standardization and generality. They focus on enumerating the situation of entities for context and pieces of information related to interactions between users and applications. Within this thesis, context is considered in relation to the designer's concern. We do not consider the context definition according to the type of attributes or entities observed. Recently, Li

et al. also offered the following definition, context *“is any piece of information that can represent changes of the circumstance (either static or dynamic). Further, it could be useful for understanding the current situation and predicting potential changes”* [Li-2015]. We almost agree with the definition of Li because it provides a great help for designers to identify and classify the type of a given context before modeling it. Based on this definition, we can divide the general context into many context elements corresponding to each designer’s concern. The context of each designer’s concern is established based on the set of circumstances surrounding it. Within this thesis, we describe the context through a set of situations of the multiple-element components of the environment. In that environment, the application designers have the freedom to observe anything necessary and interest them depending on their viewpoint. Each designer writes their viewpoints that correspond to their application purpose and contextual concerns. And, each viewpoint decomposes into a set of states, and each state is defined by a set of specific predicates for each viewpoint. Therefore, we can say that context is the set of possible situations for a set of specific viewpoints.

1.2.2 Context-aware system

Context-awareness is an essential part of the pervasive computing system, which was introduced since the early 1990s. However, the term context-aware computing was introduced by Mark Weiser in 1991 [Weiser-1991], and the term “context-aware” was first used by Schilit et al. in 1995 [Schilit-1995]. Since then, many researchers have proposed definitions of context-aware such as the definition by Ryan: *“if an application has the ability to monitor input from sensing devices and choose the suitable context according to user’s need or interests, then it can be labeled as a context-aware application”* [Ryan-1997]. While Pascoe described context-aware is *“the ability of computing devices to detect and sense, interpret and respond to aspects of a user’s local environment and the computing devices themselves”* [Pascoe-1998].

It really has many different definitions of context-aware, but the definition of Dey is widely accepted by the research community. He defined that *“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task”* [Dey-2001]. Besides, the context-aware definition of Weiser [Weiser-1991] is also used by many researchers today. According to Weiser, we can understand that context-aware may be any hardware or software that use to collect necessary context information of both user and environment to build user application. Within this thesis, we are in line with Weiser’s definition. We use multiple devices and observation services to collect context information based on the requirements of the user’s application.

1.2.3 The mobility of user and changes of contextual information

Today with the development of communication technology, users can activate and use ubiquitous applications everywhere. However, when the user's location changes, it required that the system must update the changes of context to keep the continuity of user application. For example, when the user uses digital applications to manage electric devices in their home through the smartphone. If the user moves from their home to the office, then there are many things change around them such as communication, connection, device, *etc.* These changes may change application behavior and application functionality of adaptation system. In the world of ubicomp, communication between devices may be provided by wireless transmission, physically cabled connections or 4G network, *etc.* When the user's location changes, the network protocol changes, and different protocols may need to be employed.

Location information is also an important part of contextual information because it stored information about the social situation and physical conditions. When a user moves, their relationship to stationary objects and another mobile user change also. And the physical conditions such as light, humidity, temperature, *etc.* in destination places are also different. That is the reason why we need an adaptation system which can not only manage context information but also automatically adapts to its changing environment and requirements of user and devices [Gutowski-2017].

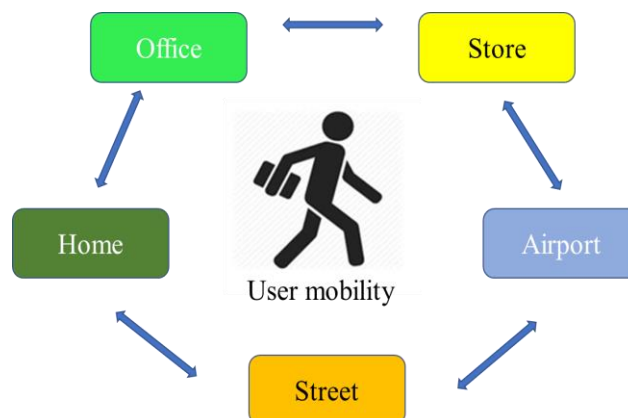


Figure 1.3: User mobility in dynamic environments

Many works had been completed to build an adaptation system (context-aware system/self-adaptive system) that is used to manage necessary context information and to keep the continuity of user application on the ubiquitous environment. However, while developing the adaptation system, the developer faces challenges related to different research domains such as context modeling and improving the adaptation ability of adaptation systems, *etc.*

1.3 Overview of challenges

Today, we might see that software applications are able to adapt to their environment. The works of Weiser, Dey, Schilit, *etc.* on ubiquitous computing, context definition, and context-aware system paved the way for a multitude of solutions to this problem. Most of these solutions were based on self-adaptive system (SAS) as shown in Figure 1.4. It operates according to the cycle of observation-decision and action. Depending on the status of context observation, the SAS makes decisions based on the set of adaptation rules which define the adaptation capacities of the system. This solution can provide an automatic adaptation of software application to dynamic context.

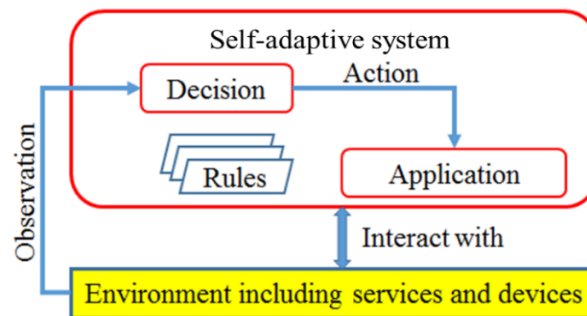


Figure 1.4: The classical model of a self-adaptive system

The main goal of using the automatic adaptation of software applications is to keep continuity of services on whatever the context. Therefore, with user mobility, we must account for adapting applications to all situations of life. That means we must find a solution to improve the adaptation capacities of the SAS. But with regards to figure 1.4, when we want to improve the adaptation capacities of the system, we have to increase the number of user situations that we need modeling. However, that work also create more challenges for designers [Siadat-2012].

Challenge A. Simplifying the designer's task on context modeling

Given the array of mobility of the user, the number of user situations will increase significantly. However, when we want to significantly increase the context situation that we wish to support, the context modeling will become more complicated. It requires that the designers to build a single large model of context and combine many concepts in the context modeling process. To do that, the designers must be experts because they have to work on many different domains of context. This work makes the task of the designer more difficult, tedious, and even impossible.

With challenge A, we pose the following two research questions to determine the requirements on context modeling and to indicate how to support designers in context modeling process.

- 1) *What are the advantages and limitations of the current context modeling techniques?*
- 2) *What is the solution to simplify the designer's task in the context modeling process?*

By answering question one, we get an overview of the requirements for context modeling. Research question two aims to find out a mechanism to make simpler the designer's task in the context modeling process.

Challenge B. Improve the adaptation capacities of self-adaptive systems

With the classical self-adaptive system, if we want to improve the adaptation capacities, we must add more adaptation rules to drive the decision process. However, the increase of adaptation rules in SAS can lead to two other problems: the conflict between adaptation rules and the lack of reactivity of the SAS.

B1. Solving conflicts between adaptations

When we improve the adaptation capacities of SAS by adding more adaptation rules, it is more likely we face potential problems that may increase the conflict ability between adaptation rules on the SAS. In that case, we need a solution to solve conflicts between adaptation rules, but solving these conflicts at design time is impossible because it means that we have to predict all concerns that will be related to the context of the ubiquitous environment. Moreover, we must calculate all cases of conflict ability that could never take place at the real operation of the system. To solve the problem relating to the increasing of AR, we must develop an automatic method of detection and conflict resolution.

With challenge B1, we pose the research questions three and four to evaluate the current solutions in development adaptation system and to propose another way to solve the problems related to adaptation conflicts in SAS.

- 3) *What are the main requirements that development of adaptation systems should meet?*
- 4) *How to detect and solve the conflicts between adaptations at runtime?*

B2. Improving reactivity of self-adaptive system

With the significant increase in adaptation rules at runtime, it makes the decision step on SAS become more complex and more time consuming. In this case, the contextual information may have changed when the system was idle at the decision time. It makes the adaptations of SAS in a situation where the performed adjustment of the system does not make sense and it must begin a new cycle of adaptation with other context observation cycles. This problem has the effect of significantly reducing the responsiveness of self-adaptive system.

With challenge B2, we pose the research question five to indicate the main challenges in improving the reactivity of SAS and figures out the solution for these challenges.

5) How to improve the responsiveness of self-adaptive systems?

1.4 Outline

Within this thesis, each chapter focus on solving the different problems of each research domain related to self-adaptive system. Therefore, the state of the art has been divided according to the different issues studied in the different chapters.

In the following, chapter 2 gives a background for the used concepts in this thesis and state of the art in adaptation system. In this chapter, we present a discussion that focuses on two main branches of research, namely context modeling and adaptation system development.

Chapter 3 presents state of the art related to using context-aware management in self-adaptive system and shows away for using independent viewpoints to model context based on using the new architecture of self-adaptive system. We describe the Context intermediate model and implementation with two special viewpoints: BPMN and CTT. We also propose a solution to filter and validate the state of the applied viewpoints.

Chapter 4 presents state of the art in the conflict detection and proposes to use the objective of state in detecting the conflicts between viewpoints.

Chapter 5 presents state of the art in adaptation conflict resolution and proposes the method to solve the conflicts between viewpoints in the CAM.

Chapter 6 describes the solutions to deploy adaptation rules and implementations of our framework.

Chapter 7 concludes the thesis by recalling the context definition and presenting the main contributions of the thesis. Finally, we identify the main topics that require further investigation for future work.

CHAPTER 2: Background and State of the Art

This chapter presents the state of the art in context modeling and adaptation system. We analyze the related works to indicate the advantages and limitations of the current solutions in context modeling and adaptation system development. This analysis can also figure out the requirements which we must solve to overcome challenges presented in chapter 1. It is also the answering for the first and the third research question.

What are the advantages and limitations of the current context modeling techniques?

What are the main requirements that development of an adaptation system should meet?

Chapter 2 is organized as follows: Section 2.1 presents the current related context modeling techniques, then we present a discussion about the criteria and requirements for context modeling. In section 2.2, we analyze the advantages and disadvantages of the current architecture of adaptation systems. Section 2.3 presents the conclusions related to the works mentioned in the state of the art and proposes a research orientation of our approach.

2.1 Context modeling

Currently, due to an increase of ubiquitous computing devices integrated into our surroundings, compounded with the requirement about the mobility of both users and devices, it will be very important for ubiquitous systems and software applications to be context-aware. Many studies have already been done on the use of context-awareness as a solution for developing ubiquitous computing applications. And most of these studies focus on the approaches to modeling context information and context reasoning techniques. Context information is captured using sensors or gathered from heterogeneous sources [Dey-2001], and the context modeling formulates all the data from these sources into an understandable manner.

There are many works in context modeling presented in the researches of [Bettini-2010], [Baldauf-2007], [Chen G.-2000], [Strang-2004], *etc.* Most of them choose to model the physical environment, and some others model the current user's situation. However, it is not easy to find a solution to model context, which can solve all the basic issues of the context modeling. This section includes an overview of the context modeling on the ubiquitous computing domain based on supporting the requirements

that context modeling should meet such as domain-independent, reuse ability, separation of concerns, imperfection, context interpretation, *etc.*

2.1.1 Context metamodel

A context metamodel defines a language and the semantics of the key concepts that can be used to define the context. Many authors used metamodel as a solution to solve problems of context modeling. They defined a metamodel for guiding a context modeling in different applications [Vieira-2010], [Jaouadi-2015], proposing a graphical modeling notation [Henricksen-2002], using the specification meta-object facility to allow a precise syntax and an abstract representation common to all the models [Leite-2007]. On the other hand, Fuchs et al. [Fuchs-2005] proposed a metamodel to define contextual information and its association. They also defined some additional requirements of context modeling such as evolutionary development, interoperability, reasoning and ease of use. Their work can provide a way to restrict the interoperability of service to those that use information models. In addition, their context modeling technique can also adopt a widely used conceptual metamodelling architecture.

Shishkov et al. derived a meta-model using the notation of UML [Shishkov-2018] as shown in figure 2.1. This meta-model describes a system and its environment based on the composing of numerous entities that can be components or agents. Each entity can enact different roles categorized into four roles: user, processor, actuator, and sensor. These entities are subject of regulations and they are restricted by corresponding rules. This solution uses general context modeling for all applications, so it cannot support the ability to separate concerns on context modeling process. However, this meta-model can easy to modify and reconfigure to use in different adaptation systems.

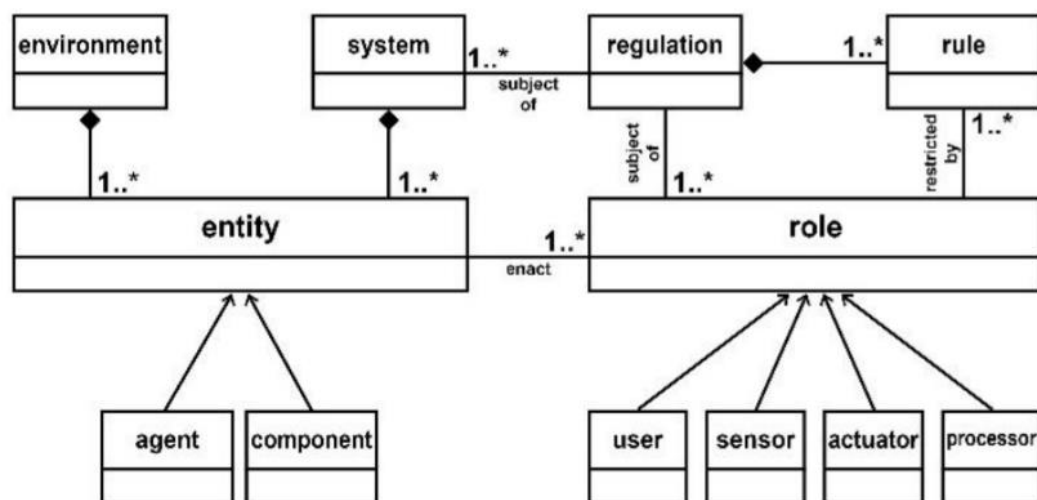


Figure 2.1: Context meta-model [Shishkov-2018]

TriPlet [Motti-2013] is structured in three core components: a metamodel is called Context-aware metamodel, a reference framework called Context-aware reference framework, and a Context-aware design space that supports stakeholders in the analysis and evaluation the adaptable application for context-aware system.

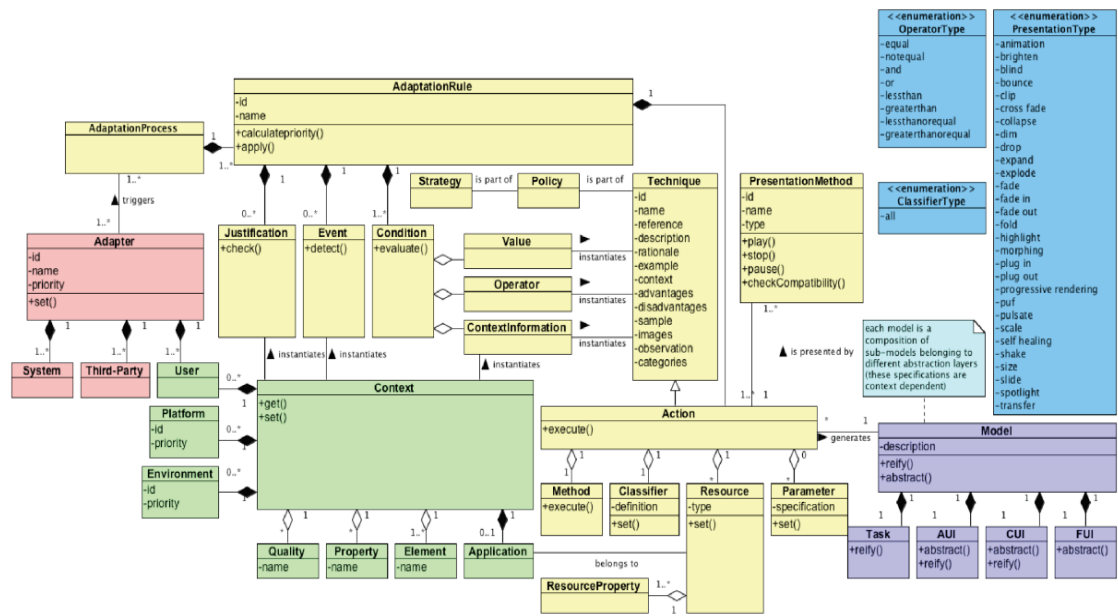


Figure 2.2: CAMN: Context metamodel on TriPlet

The context-aware metamodel in the TriPlet approach named CAMN that is used to abstract concepts that is necessary for model context, to establish the relationships between these concepts, and to define their properties. The main goal of this study is that it offers the methodology for building context-aware applications. However, the TriPlet establishes mappings between adaptation and context information based on covering all adaptation needs without being specific to a particular application scenario or domain. Therefore, this study does not support capabilities of reuse the context model in different adaptation systems. Moreover, it also makes the designer's task more complicated.

2.1.2 Context ontology model

Context ontology is a solution using Ontology to represent semantics, concepts, and relationships in the context data [Ejigu-2007]. We might see that it is formed by the combination of several ontologies that deals with questions concerning what entities exist in the pervasive environment and how such entities may be grouped or classified, related within a hierarchy, and subdivided according to similarities and differences. The use of ontologies on developing context models brings us several benefits and additional

functionalities such as formal knowledge representation, logic reasoning, knowledge sharing and reuse.

Some researchers used Ontology techniques to model context such as works of [Chaari-2006], [Wang-2004], *etc.* These works focus on defining generic context storage and processing model to create the intelligence to analyze the context information and deduce the meaning out of it. They use ontology to make their model independent of programming and application environments. Moreover, the standardization of the structure of the context representation provides semantic descriptions and relationships of entities.

Wang et al. proposed an encoded context ontology named CONON for modeling context information. It is based on OWL for reasoning and representation of contexts in ubiquitous computing environments [Wang-2004], as shown in Figure 2.3. Their approach provides a reusable infrastructure for context-aware mechanisms and solutions for sharing knowledge and learning for context interpretation. In CONON, Wang et al. propose the solution to separate application domains, encouraging to improve the reuse of general concepts.

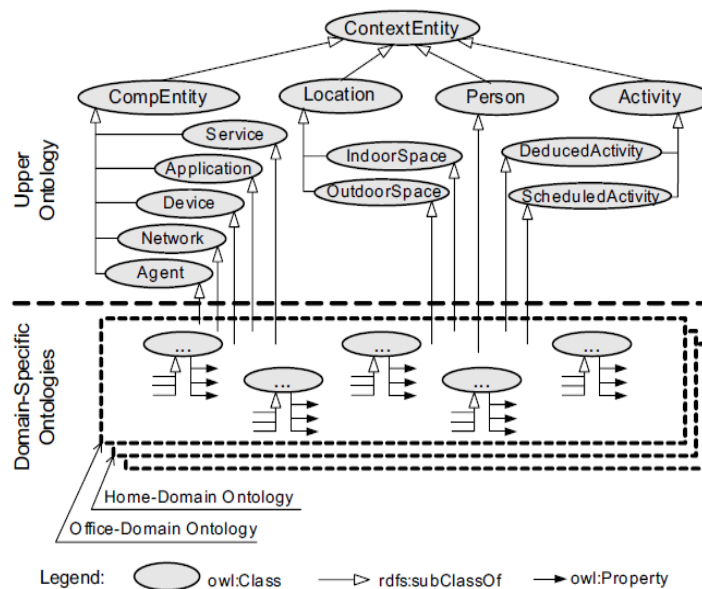


Figure 2.3: The CONON upper ontology for smart home application

This solution also provides a flexible interface for defining specific application knowledge. They divide their context model into upper ontology and specific ontology. The main task of upper ontology at a high-level is capturing general features of basic contextual entities. In lower-level, specific ontology used to collect ontology set, which use to define the details of general concepts and their features in all sub-domains. Nevertheless, this solution requires the designers must be experts because they have to use the knowledge of the different domains in context modeling.

Xu et al. introduced an ontology-based general and extensible context model (CACOnt) for modeling context information and providing inference mechanisms [Xu-2013]. CACOnt structured in two ontologies components: the generic context ontologies for capturing basic concepts about context and context extensibility ontologies for adding domain-specific ontologies in a hierarchical manner. This separation can reduce the scale of context knowledge and improve the reusability of general concepts in context modeling. In addition, CACOnt provides a hybrid approach of context reasoning and a semantic similarity-based rule matching algorithm. However, this study does not offer a solution to separate contextual concerns in different tasks. Therefore, it exists a limitation on the reusability of the context ontology among several independent applications.

Aguilar et al. used a CAMEnto meta-ontology for context modeling [Aguilar-2018]. This ontology is based on the 5W questions, and it is characterized by two levels: general ontology and specific domain context ontology, as shown in figure 2.4. In this study, the context is categorized into three types: boundary context, external context and internal context. And they describe context through six contextual classes include user, activity, time, device, services, and location. This approach allows reuse context model in development applications without too much modification. However, this study does not support a mechanism to separate concerns in context modeling.

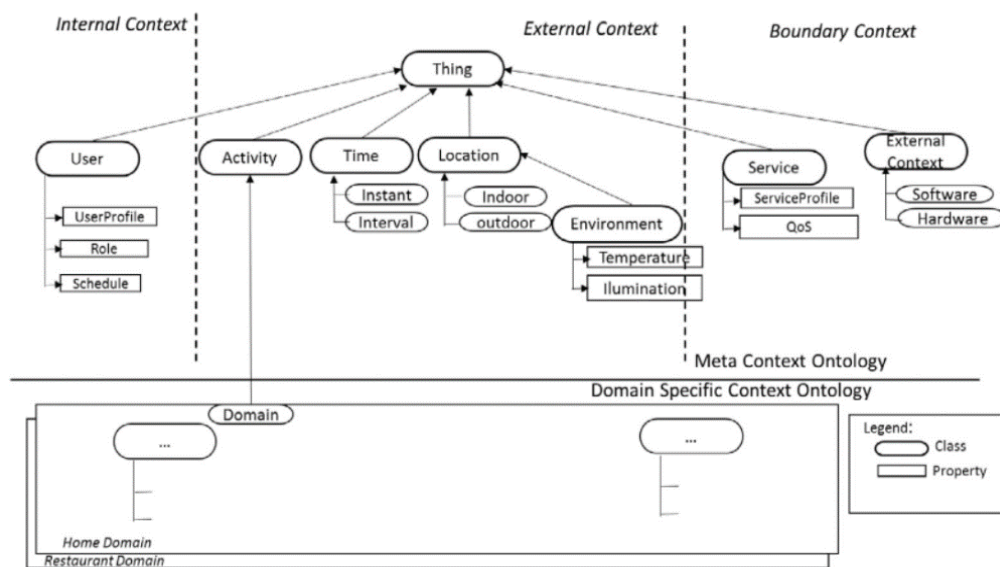


Figure 2.4: Hierarchical design of a Context ontology [Aguilar-2018]

2.1.3 Graphical model

“Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied

mathematics and engineering – uncertainty and complexity – and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms” [Jordan-1998]. The graphical models are commonly used in probability theory, statistics—particularly and machine learning for context modeling. One of the graphical models very well known on the general-purpose modeling instrument is Unified Modeling Language (UML), which is also appropriate to model the context. This is shown in the approach of Bauer [Bauer-2001], where contextual aspects are modeled as UML extensions.

Henricksen et al. introduced a nicely designed graphics-oriented context model [Henricksen-2003], as shown in Figure 2.5. It is a context extension to the Object role modeling (ORM) approach [Halpin-1998] according to some contextual classification and description properties. In the approach of Halpin, ORM is a conceptual modeling approach that comprised both a graphical model notation and accompanying design methodology. With ORM, the basic modeling concept is the fact in the pervasive environment, and the modeling of each domain using ORM such a tool to identify appropriate fact types. Henricksen extended ORM to allow fact types can be categorized into static, profiled, sensed or derived that can support a variety of context management tasks.

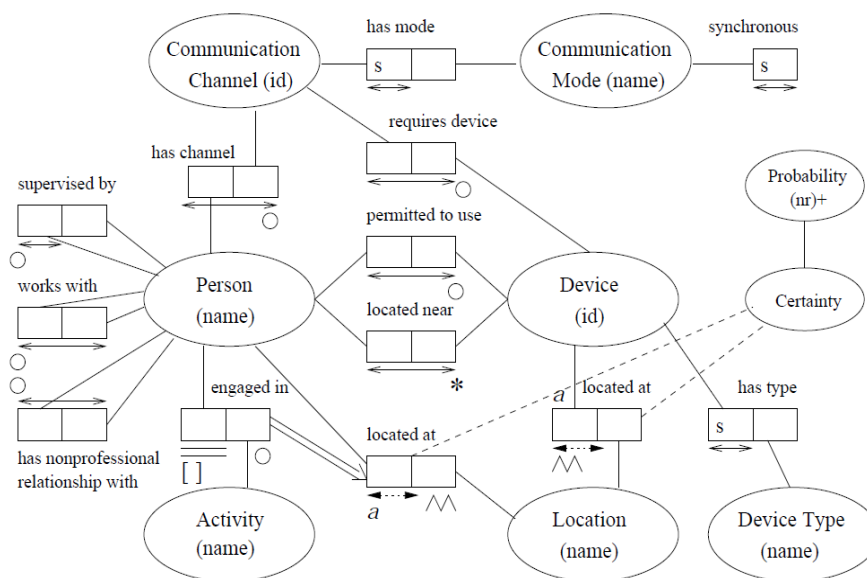


Figure 2.5: The graphical context modeling follows fact type [Henricksen-2003]

Moreover, the modeling solution of Henricksen can provide the ability to update context information in both static and dynamic situations of the facts in the pervasive environment. Nevertheless, this study does not mention any solution to simplify the designer’s task and to reuse the context model of designer’s concerns in different applications.

2.1.4 Discussion

In order to have a good understanding of relevant works and characterize the contributions of each approach, we propose some criteria and requirements for the context modeling process that use as an important aspect to compare these approaches. Some of these criteria and requirements are inspired by the works reviewed in the above section, and some other requirements have been identified based on our problematic that presented in chapter 1:

Separate concerns: Is a design principle for separating a model of each application into the distinct model. It can provide a higher degree of freedom for some aspects of the model design and deployment. The designer can modify or improve each concern's section of code without having to know the other concerns of application.

Reusability of concerns: It is the ability to reuse special designer's concerns to model context in different applications. Each single context concern can be reused by designers depending on the application's requirement and user situation.

Context interpreting: It is the ability of context modeling technique to transform contextual information collected from the ubiquitous environment into a significant format easier to handle and use by context-aware management layer at runtime [Jaouadi I.-2015].

Simplifying expert task: The context modeling process does not require that experts must understand the structure and semantics of the context modeling framework. The experts need only knowledge in their expertise domain and focus on it to build their applications.

Moving forward, we will discuss each of these requirements to clarify the advantages and limitations of the current approaches in literature.

- Separating concerns: The use of independent models in context modeling for each application is necessary and useful in order to simplify the designer's task. The works of Wang, Fuck, and Henricksen provide a solution to build a complex context model by using ontology or a special type of metamodel, where they combine many different concepts on one big model to represent many situations of the user. They also focus on evaluating the feasibility of logic-based context reasoning for non-time-critical applications in a ubiquitous environment. It requires that they must deal with the limitation of computational resources and increasing of conflict ability between adaptation rules at runtime.

- Reusability of concerns: if we do not have a separation between application semantics and low-level details of context acquisition, it leads to a loss of generality and reduces

the reusability of contextual concerns in different applications and difficult to use simultaneously in multiple ubiquitous applications. The metamodel on the study of Motti and Fuchs is used to abstracts necessary concepts, establish their relationships, and define their properties in context modeling. They used a context model on the instance layer, so these works require a combination between application semantics and low-level details of context acquisition. This solution may limit the reusability of context concerns on other applications.

- Context Interpretation: In many current context-aware frameworks such as [Chaari-2006], [Vieira-2010], *etc.* we see that context data must go through multiple layers before reaching an application, due to the need for additional abstraction. This can be much more difficult for the adaptation layer to read and analyze complex information from context modeling. In order to support this problem, the context must often be interpreted before it can be used by the adaptation layer.

- Simplifying expert task: With some approaches (as the works of Henricksen, Chaari, *etc.*), we might see that the driver for sensor or other hardware and software that are used for detecting context are directly hardwired into their applications. This work required the designer to code and use whatever protocol that deals with the sensor detail when they build an application. Moreover, the designer must also develop context-aware management founded upon the modeling constructs. It makes the task of the designer become very complicated and burdensome.

In Table 2.1, we present a comparative table evaluating the aforementioned context modeling initiatives using the dimensions are requirements just presented above. We annotate the symbol “++” or “+”, which indicates that the approach addresses and partially address the requirement with context modeling process, and the symbol “-” does not address the given dimension.

Table 2.1: Evaluating current context modeling initiatives

Approach \ Requirements	Context interpreting	Separating concerns	Simplifying expert task	Reusability of concerns
[Henricksen-2003]	++	+	-	-
[Wang-2004]	+	+	-	+
[Chaari-2006]	-	+	-	+
[Motti-2013]	+	-	+	-
[Xu-2013]	+	+	-	+
[Jaouadi-2015]	++	-	-	++
[Shishkov-2018]	+	-	+	+
[Aguilar-2018]	++	-	+	+

We might see that all of the approaches towards context modeling presented above do not investigate the benefits of conceptual modeling in the design process. Many approaches support a modeling solution focus on providing a solution for context domain-independent, rich and dynamics on present all aspects of context. However, most of them do not indicate the solution to separate concerns and simplify the expert's task in context modeling.

2.2 Adaptation system

An adaptation system is a system that uses middleware and an adaptation platform for providing user application according to the pervasive environment. Many works related to adaptation system have been done within the last two decades as initiatives about self-adaptive system, context-aware system, context toolkit, adaptation middleware such as [Dey-2001], [Bolchini-2013], [Keling-2011], [Cheung-Foo-Wo-2007], [Brun-2009], *etc.*

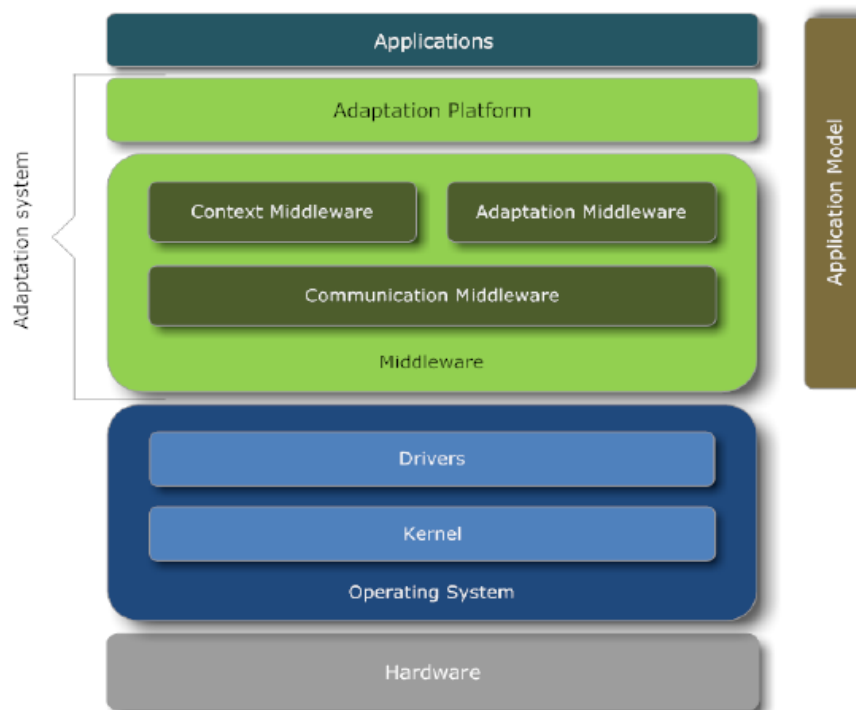


Figure 2.6: Architecture of an adaptation system [Keling-2011]

Both do sense changes in their environment and respond by changing their behavior. However, the context toolkit and context-aware system focuses on modeling and reasoning relevant environmental and respond to an anticipated change of application or context while self-adaptive system focuses on how the system responds to unanticipated environmental changes.

2.2.1 Context toolkit

Dey et al. introduced a context toolkit based on combining of the sensor to support the rapid prototype of context [Dey-2001]. This approach provides a solution for the reusability of model components, development of applications, acquisition, and use of complex context. Moreover, it provides the methods to access such information, translating the contextual information into high-level formats that are easier to handle by the adaptation system. And it allows the separation of the acquisition process and the context representation of the adaptation system. The architecture of the Context toolkit can be used to build a context-aware application that allows the reusability of components in different applications. Moreover, it supports the evolution of applications and the use of complex context information. However, this study does not mention any solution to separate context management and adaptation layer. This work can lead to reducing the responsiveness of the adaptation system when we increase the number of user's scenarios or applications.

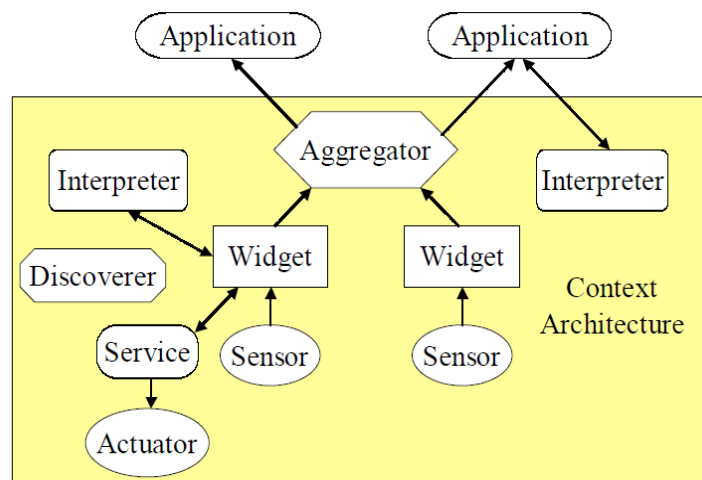


Figure 2.7: Architecture diagram of Context Toolkit

2.2.2 Adaptation middleware

An adaptation middleware is a middleware that provides mechanisms or services to achieve adaptation tasks. It allows the user application to support many concurrent users, connect to many different data sources, and interoperate with another middleware service.

In recent years, many context-aware middleware architectures such as [Li-2015], [Yu J.-2010],[Sain-2010],[Fahy-2004] etc. have been proposed to provide adaptive behavior to the continually changing environment proactively. Most of these architectures developed were based on adapting the requirement of the adaptation system such as

context collecting, context controlling, adapting of application, separation context management and application, *etc.* In this section, we present some middleware architecture that would partially respond or have most the requirements align with an adaptation system.

SOCAM [Gu-2005] is a service-oriented context-aware middleware which aims to provide efficient infrastructure support for building a context-aware application in a ubiquitous environment. The main power of SOCAM is the ability to convert contexts information from various physical space of context into a semantic space where contextual information can be easily used and shared by context-aware application. The SOCAM architecture provided an intelligent system of reasoning about the context based on OWL for context modeling. This approach allows the developer to define rules and specify method to be invoked when a condition of the current situation becomes true. It also allows the recording of adaptation rules and updates the contextual information when the system detects context changes. This architecture provides a context database to store both the current context and the past contexts which can help improving the responsiveness of context-aware system. However, this study does not mention any solution to reduce the conflicts between adaptation rules at runtime.

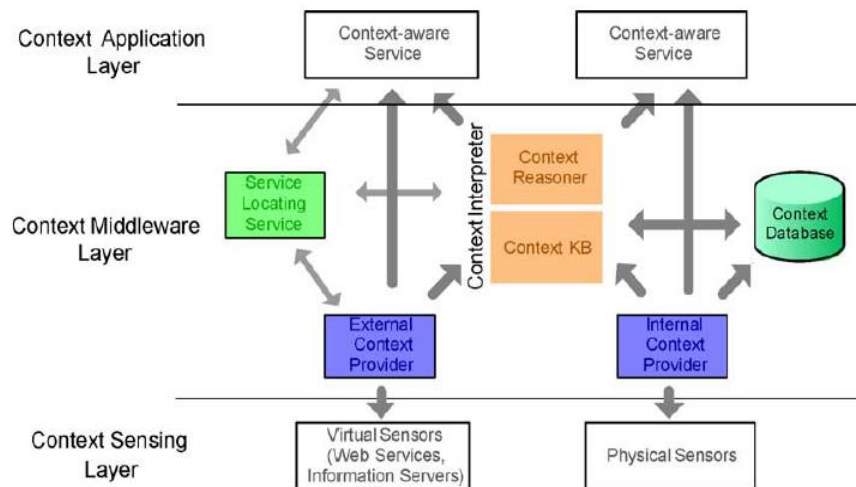


Figure 2.8: The SOCAM architecture

WComp [Tigli-2009] is a typical AOP-based adaptive middleware; it uses a concept named Aspect of Assembly for adaptation, as shown in Figure 2.9. The WComp middleware model is structured among three main paradigms: event-based Web services, a lightweight component-based approach to design dynamic composite services, and an adaptation approach using the original concept called Aspect of Assembly. These paradigms provide two solutions to design ubiquitous computing applications dynamically. With applications in a common and usual context, a

component-based approach paradigm can support a solution to design higher-level composite Webservices and increment the graph of cooperating services for the ubiquitous computing applications. The second solution uses a compositional approach for adaptation system using the aspect of assembly.

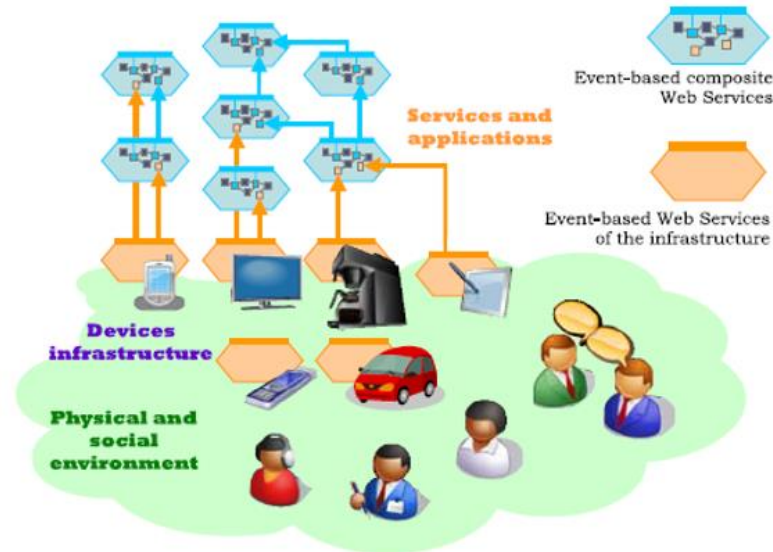


Figure 2.9: Context and user driven self-adaptation [Tigli-2009]

This paradigm suited to drive a set of composite services in reaction to changing preferences of users or a particular changing of context information. This approach provides a user-driven approach as a solution to make adaptation middleware adapt to the application to its environment. It allows the user to intervene on the base assembly and operate directly on the assembly as to erase or integrate the behaviors and functionalities of the adaptation system. In this study, the conflicts between adaptations are solved in the adaptation rules level. It leads to a limitation of responsiveness when the number of adaptation rules increases significantly.

2.2.3 Context-aware system

The context-aware system is a system or system component to gather information about its environment at any given time and adapt behaviors accordingly. It uses software and hardware to collect and analyze data to guide responses automatically. It focuses on modeling and reasoning about relevant environments by using an ontology, metamodel, *etc.* to respond to an anticipated change of application or configuration setting.

We can find many works that have been done in the area of the context-aware framework with many major categories as device communication, self-learning, application-oriented [David-2005], [Dey-1998], [Nandyala-2016], [Perera-2013]. In this

part, we present some context-aware frameworks based on context-aware application and interaction with context view.

PersonisAD [Assad-2007] is a context-aware framework that builds upon the homogenous modeling of all the entities relevant such as people, sensors, devices and places to supporting ubiquitous applications. This approach provides a context model technique that organized as a tree of the context model, which contains components of the model. Conversely, the PersonisAD framework also provides a powerful and consistent response to the changing of context information. The main power of this framework is supporting ability personalized ubiquitous services, integrating the collection of diverse types of evidence about the ubiquitous environment, and interpreting it flexibly. The limitation of this study is the lack of context information storage. The authors also do not provide any mechanism to solve the conflicts between adaptations generated by the PersonisAD framework.

Costa [Costa-2007] proposed an integrated solution for the development of context-aware system, focusing on context modeling and context handling platform. Their context modeling abstractions support application designers with proper conceptual foundations that are used to adapt to the requirements of a specific application. This approach also proposes a model-driven for the specification of a situation in the context-aware application, and context-aware application behaviors can be presented as the logic rules, which are called event-control and action rules. They use context processor component to handle context concerns, gather context information from the user's environment, and generate context and situation events. In this approach, they use controller Components to monitor conditions rules and observe events through context processors. When the condition is satisfied, the controller component will trigger actions on the action performer. This solution can simplify the designer's task on application development and improve the ability to integrate multiple applications in the same system. However, this study does not indicate the problem of the conflicts between adaptations of applications and solution.

Achilleo et al.[Achilleos-2010] proposed a model-driven methodology that supports the creation of a context modeling framework. It can simplify the task of the design and implementation of pervasive services. Their model-driven methodology also provides a higher level of automation in software generation. They built a Context-aware pervasive service creation framework (CA-PSCF), which enables the designer to define a context model at an abstract level. The CA-PSCF supports a validation of context models to determine valid and invalid definitions, and then it rectifies these invalid definitions before the implementation phase. This solution allows designers to define independent context models following their concerns and reuse them for different applications.

However, the CA-PSCF does not mention to the conflict ability between independent models when the system merges them at runtime.

Context Broker architecture [Chen H.-2003] is an architecture for supporting context-aware applications, as shown in figure 2.10.

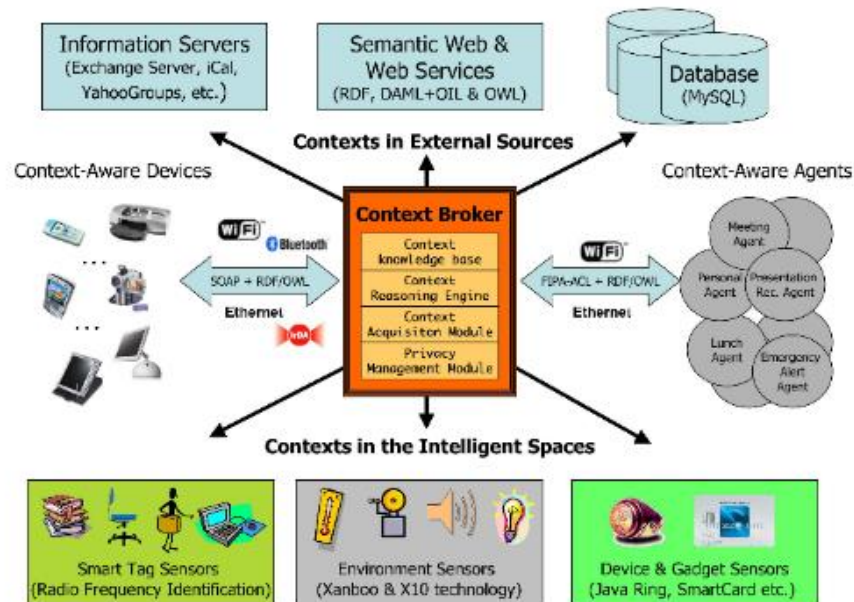


Figure 2.10: Context Broker architecture CoBra

The central theme of this approach is a broker agent that maintains a shared model of context based on Web ontology language (OWL) for all computing entities in the space on behalf of a set of collaborating agents, services, users and devices. The CoBra architecture acquires contextual information from different sources such as smart sensors, devices and gadgets. It consists of four functional components which includes context knowledge base, context reasoning engine, context acquisition module and privacy management module. CoBra provides a policy language that allows users to control distribution and notification of the user context information. Nevertheless, the CoBra architecture still has the limitation on separation between context-aware management and adaptation system. Moreover, this architecture requires that the developer use too much expert knowledge to configure the system.

CAISDA [Jaouadi I.-2015] provides the architecture of context-aware framework, which simplifies the designer task and the implementation of context-aware systems. At design time, the developer must use a special model tool to build a context model of its application that is consistent with the metamodel provided by CAISDA. At runtime, this approach using Context providers to collect information from different heterogeneous sources and convert it to an understandable structure by the system. Moreover, they use a Context controller to detect the change of context and notify the system for

reconfiguration. This solution can improve the adaptation ability of context-aware system because it allows integrate and extend many context models in one system to cover user's scenarios and applications. However, it is better if this study provides a mechanism to detect and solve the conflicts between applications in the context-aware management layer.

Mahmud et al. developed a context-aware system where the context process is done through five modules which includes context aggregator, context representation, context history, context inference, and service adaptation [Mahmud-2018], as shown in figure 2.12. The context information is collected from the sensors system through context aggregator module. After that, the contextual information data is stored as per the data organization of context-aware system in the context representation module.

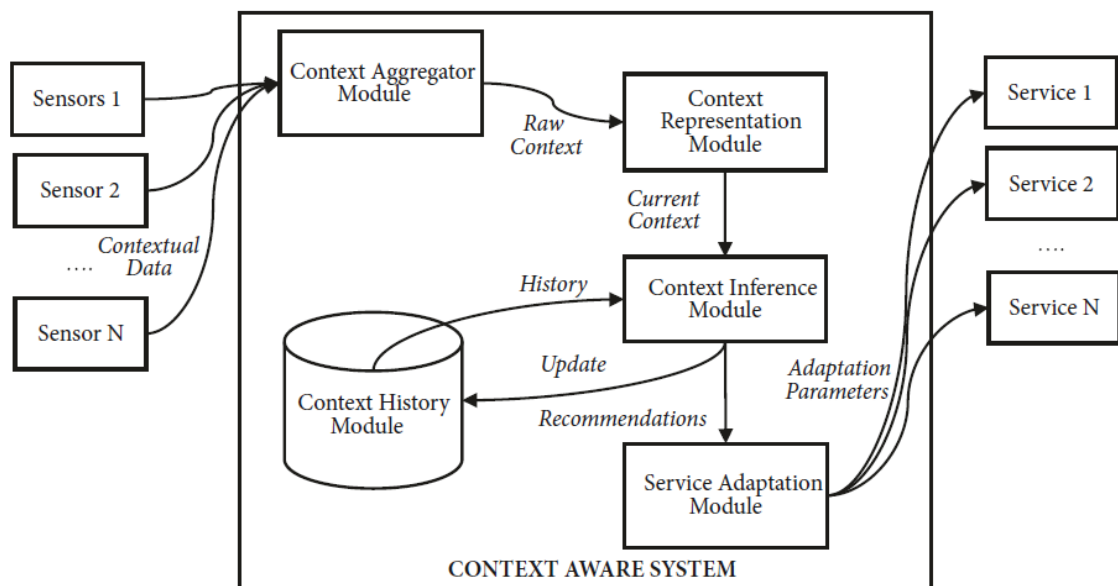


Figure 2.11: Context-aware system [Mahmud-2018]

In this context-aware system, the system used the context history module to maintain the history of contexts, supporting the context inference module on detecting the changing of context situations. Based on the information from the context representation module and the context history module, the context inference module uses a mechanism to classify the activity and situation of the current context. In the service adaptation module, the services are requested and adapts to the following recommendations from the inference module. Nevertheless, this study used the service adaptation module to manage and analyze the user's scenarios or application requirements. If we want to improve the adaptation ability of the system by increasing the number of the user's scenarios then the system must face problems related to system consumption and the lack of reactivity of the system.

2.2.4 Self-adaptive system

A self-adaptive system (SAS) is a system that is able to automatically modify itself in response to changes of context in its pervasive operating environment. The operating environment includes anything observable, such as software or hardware, end-user input, surrounding context, *etc.* The modification of the system is done by adjusting attributes or artifacts of the adaptive system in response to changes in its operating environment or the system itself. In recent years, we can find many definitions of self-adaptive system, such as the definition of Brun et al.: *“Self-adaptivity is the capability of the system to adjust its behavior in response to the environment. The “self” prefix indicates that the systems autonomously decide how to adapt or to organize themselves so that they can accommodate changes in their contexts and environments. While some self-adaptive systems may be able to function without any human intervention, guidance in the form of higher-level objectives is useful and realizable in many systems”*[Brun-2009]. Cheng et al. defined a self-adaptive system as a system that *“is able to adjust its behavior in response to their perception of the environment and the system itself”* [Cheng-2009]. Danny Weyns introduced the external principal of a self-adaptive system: *“A self-adaptive system is a system that can handle changes and uncertainties in its environment, the system itself and its goals autonomously”* [Weyns-2017].

Abeywickrama et al. introduced a new general model for self-adaptive system called SOTA that was used for modeling the adaptation[Abeywickrama-2012]. The SOTA provides a systematic method to model the real-world goals of self-adaptive system. It can be used to early assess self-adaptation requirements via model checking techniques and identification of knowledge requirements for self-adaptation. This study provides a framework method to design and develop a complex self-adaptive software system. However, the developers have to use expert knowledge to develop an application with the SOTA model. It is a limitation with the ability of rapid application development.

A survey on the Engineering approach of self-adaptive system of Krupitzer et al. [Krupitzer-2015] showed that self-adaptive systems have seen an increasing level of interest in different research areas such as Pervasive Computing, Autonomic Computing, and Nature-Inspired Computing. Moreover, this approach mentioned to some new terms on self-adaptive system, such as self-configuration, self-healing in the presence of failures, self-optimization, and self-protection against threats. To achieve adaptive behavior, the requirement with system properties are self-awareness and context-awareness [Salehie-2009]. Self-awareness describes the ability of self-adaptive system on aware of itself or its operating environment to be able to monitor its resources, state, and behavior. However, the studies in this survey also does not mention to any solution to reduce the conflicts between adaptations in context-aware management layer.

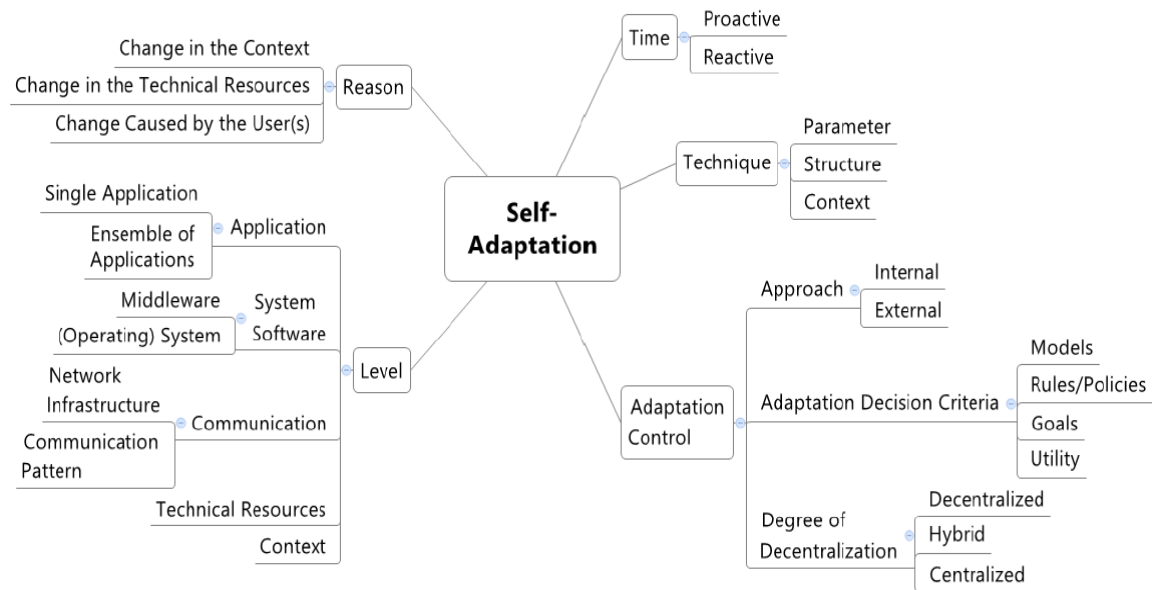


Figure 2.12: Taxonomy of Self-adaptive system architecture [Krupitzer-2015]

2.2.5 Discussion

Most of the middleware and platforms that support context-aware application development focuses on providing a solution for transparent context sharing, context controlling, and separation between adaptation and context management. We have enumerated several self-adaptive system characteristics and compared works completed to clarify the challenges in development of self-adaptive system:

Context information storage: In some cases, self-adaptive system can use both the current context and the past contexts to adapt their behavior, detect system problems, or improve adapting ability.

Separation context management and adaptation: It allows a self-adaptive system to focus on the adaptation of the application without sacrificing the responsiveness of the system, whereas, at the same time, the context-aware management focuses on observing the current context.

Reducing the conflicts on adaptations: it can decrease the conflict ability between adaptations of different services in one system.

The rapid development of application: The architecture of self-adaptive system should offer mechanisms for the deployment and the configuration of system applications, which are simple to use [Jaouadi I.-2015]. It should not require much programming and configuration efforts. Moreover, the developers must not be forced to become experts in the field.

Now we will discuss each of these characteristics to clarify the contributions of the current studies on improving the adaptation ability of self-adaptive system.

Contextual information storage: The difference between self-adaptive system and context-aware system is how the system responds to unanticipated environmental change. If the system has a contextual information storage function then it can detect the repetition of the previous state that can help the system reuse the adaptation rules. Rather, sometimes the system can detect one invalid state that exists in many adaptation cycles, which means the hardware relates to input that may contain a problem. The work in PersonisAD does not provide any solution to the system which can use the past contexts to adapt their behavior or detect the problem of the sensor system.

Separation context management and adaptation: This requirement with the adaptation system is very important because it can help adaptation system on improving adaptation ability, reusability and extensibility of approach. Most of the studies mentioned above are interested in separation between context management and adaptation. However, the works of Dey, Chen and Achilleo focus only on supporting a rich context model and a framework to share the model of context. They do not mention using context-aware management to manage contextual information.

- **Reducing the conflicts on adaptations:** The conflict between adaptation rules is the main reason for reducing the responsiveness of the adaptation system. Therefore, reducing the number of conflicts at runtime is essential to solve the problem related to improving the responsiveness of the system. Most of the aforementioned studies provide an adaptation system that allows the developer to define the rules and specify the methods to be invoked when a condition of the current situation becomes true. However, they do not mention the solution to reduce the conflicts between adaptation rules at runtime.

- **The rapid development of application:** It means that the mechanism of the system does not require much programming and configuration efforts. Therefore, the deployment time should be reduced significantly with respect to the time required to develop an application. The studies of Chen and Abeywickrama do not support any mechanism or solution to reduce the time of application development. Their approach requires that the developers must be experts because they must take the complicated reconfigures during application development.

In Table 2.2, we present a comparison evaluating the aforementioned context-aware framework initiatives using the dimensions required that were presented previously. We annotate the symbol “++” and “+” indicating that the approach addresses and

partially address the requirements for an adaptation system, and the symbol “-” does not address the given dimension.

Table 2.2: Evaluating current adaptation system approaches

Requirement \ Approach	Contextual information storage	Separation CAM and adaptation	Reducing conflicts on adaptations	Rapid development application
[Dey-2001]	+	-	-	++
[Chen H.-2003]	+	-	+	-
[Gu-2005]	+	++	-	+
[Costa-2007]	+	+	-	+
[Assad-2007]	-	++	-	+
[Tigli-2009]	+	+	-	++
[Achilleos-2010]	+	-	-	++
[Abeywickrama-2012]	+	+	+	-
[Jaouadi I.-2015]	++	++	-	+
[Mahmud-2018]	++	+	-	+

We might see that all of the approaches towards the adaptation system presented above do not support all requirements of the adaptation system. Most of the approaches support a framework that focuses on providing the solution for context controlling, context management, and application adaptation. However, these approaches do not consider the existing problem of adaptation systems such as conflicts between applications, improving adaptation ability of the system, *etc.*

2.3 Chapter conclusion

From our state-of-the-art research, we can conclude that none of the initiatives presented here address the distribution of using specific contextual concerns on context modeling. Moreover, we might see that many approaches offer support to rich and dynamic context modeling. However, most of them do not support a solution to simplify the expert task and improve the reusability of context concerns in different applications. The simplifying expert’s task and increasing the reusability of context concerns in context modeling should be seen as the important requirements of context modeling development. Therefore, we should consider these aspects when building a context handling platform.

With the adaptation system, the most of researchers used CAM as a key solution to support adaptation system on manage and handle contextual information. However, none mention to solving conflicts between adaptations in CAM. If we can solve the

conflicts between adaptations in CAM then it can reduce the number of the conflicts which the adaptation system must solve before providing services to the user. Hence, solving the conflict between adaptations in CAM becomes an important aspect that we want to consider when designing a context-aware management.

The discussion section in this chapter has answered the first and the third research question by pointed out the advantages and limitations of the current approaches on adapting the requirements of context modeling and adaptation system development. The next chapter will indicate the solutions to solve the research limitations of the current approaches.

CHAPTER 3: Context intermediate model and Architecture pattern

This chapter presents the new architecture pattern that we propose to support the development of self-adaptive system. This architecture is designed based on solving the limitations of current approaches as mentioned in chapter two. In this architecture, we propose to use independent viewpoints as a solution to simplify the expert's task in context modeling, and it is also the answering to the second research question:

- *What is the solution to simplify the designer's task in the context modeling process?*

We also suggest using context-aware management to manage both context information and self-adaptive system. This solution can meet requirements related to improving the reactivity of self-adaptive system that was mentioned in the fifth research question:

- *How to improve the responsiveness of self-adaptive systems?*

This chapter is structured as follows: Section 3.1 presents the Context-aware management; Section 3.2 presents the limitation on context modeling with a single big model and solution for this problem by using independent viewpoints; Section 3.3 introduces the special viewpoints in context domain; Section 3.4 presents using independent viewpoints in context modeling to separate concerns; Section 3.5 describes our activity design, and context intermediate model; Section 3.6 presents two real implementations with BPMN and CTT viewpoints; Section 3.7 introduces a solution to filter valid and invalid states of viewpoints in the CAM; Section 3.8 presents the chapter conclusion.

3.1 Context-aware management (CAM)

Presently, many researchers used context-aware management as a solution to separate the execution cycles of the context-aware and adaptation on self-adaptive system (SAS). This solution allows the self-adaptive system to have more time to take care of the changes in the current context information. The role of context-aware management system is defined as follow in [Henricksen-2005]:

“Context management includes the distributed context maintenance, access, update and synchronization, as well as the provision of a transparent interface to context handling, support of ad-hoc context exchange, real-time and non-real-time context handling.”

We might understand that the main purpose of the context management layer is to design and implement a mechanism of self-adaptive system, which allows the context information to be updated and distributed independently with adaptation. The work of Vieira et al. [Vieira-2007] showed that context-sensitive systems are those that understand and use contextual elements to provide relevant services or information to the users or to other applications during the execution of some task. In this system, context-aware management has an important role in the definition of model and systems to assist the acquisition, manipulation, and maintenance of a shared repository of contextual elements between different context-sensitive systems. The main goal of using context management is to reduce the complexity of building a context-aware system. It allows transfer tasks related to contextual elements manipulation to an intermediate layer. In this way, the context-aware management focuses on the mechanism to model context and share context information, infrastructure to detect and update context information, mechanisms to process, and infer new contextual elements sets from existing ones.

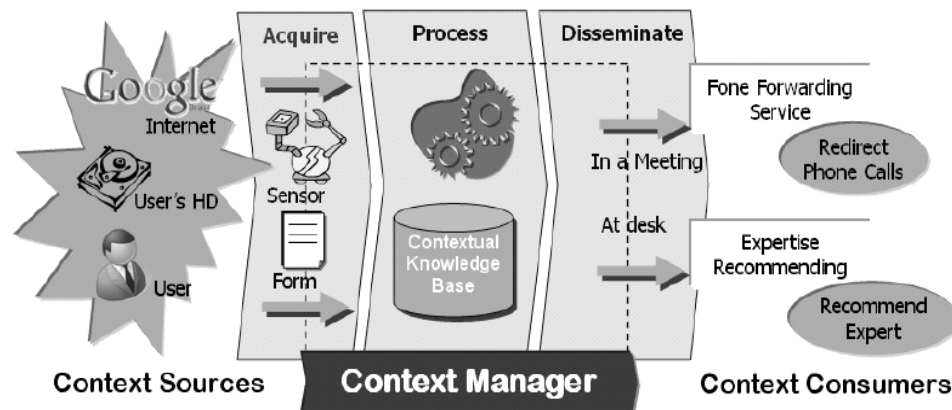


Figure 3.1: Generic context manager main functionalities [Vieira-2007]

Kim et al. introduced a context management system that manages context information using context metadata [Kim-2008]. This context-aware management was divided into six context components: Context aggregator service, Context discovery service, Context provider services, Context observer service, Context ontology reasoner service, and Context query service, as shown in Figure 3.2. Following Kim's approach, the context-aware management is used to collect and manage context information from heterogeneous devices including sensors, cameras, embedded systems, traditional computing devices, and communication devices. It also uses a service-oriented architecture to solve the problem of heterogeneity in a ubiquitous computing environment. As a result, this solution supports consistent interfaces and uniform protocol with standardized manners of the system.

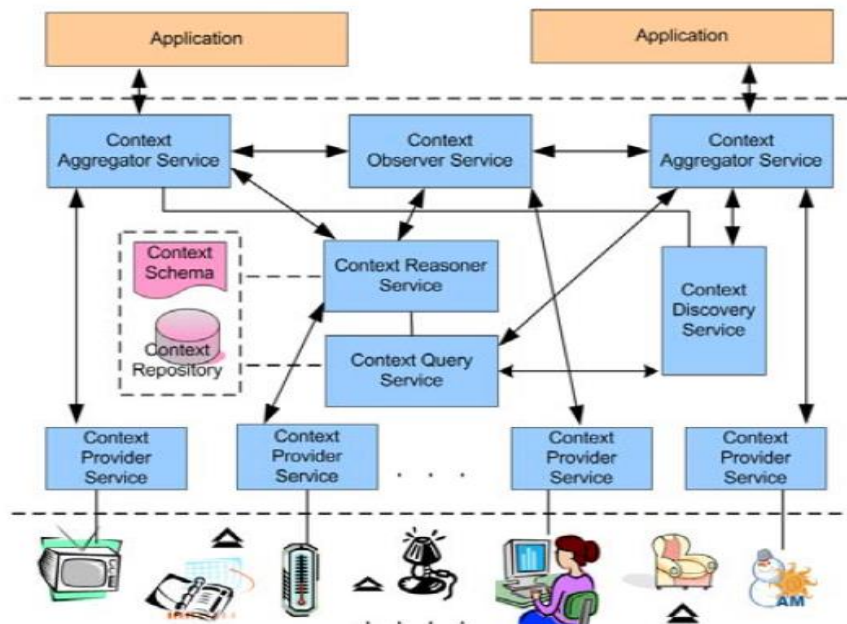


Figure 3.2: Context-aware management structure [Kim-2008]

From the literature reviewed above, we might see that the most recent studies usually use Context-aware management (CAM) to manage the context information. It can help self-adaptive system (SAS) focus only on adaptation function. Moreover, the separation between CAM and SAS can reduce the complexity of building self-adaptive system. However, in order for CAM to collect and process contextual information, it also requires a suitable technique of the context modeling process, which still has many limitations.

3.2 Context model solution.

3.2.1 Problem with designing a single big context model

As we mentioned in Chapter 1, when the user changes their location, the context surrounds them also changes. If we want to keep the continuity of the user services, we must increase the adaptation capacities of SAS. The SAS must provide mobile applications that can intelligently adapt to different environments and to different users. This means the SAS has the ability to cover all user situations and user scenarios. However, the modeling user context base on prediction all user situations can increase the number of concepts that need to be defined during the context modeling process. It also requires the designer to design a single big context model and combine bits of knowledge of different expert domains in context modeling process (as shown in Figure 3.3). It makes the tasks of the designer on context modeling very complicated.

Creating a single big model of context still has other limitations on development and reconfigure context modeling process. It makes detecting and solving problems about architecture and algorithm of the model become very difficult. This work requires that the designer be an expert in many different domains. Moreover, with a huge number of situations modeled in context modeling, it also takes a lot of time for context analytics in CAM. Furthermore, we might see that it limits the reuse of the model because it is very specific to the case at study and therefore not generalizable to other cases. That is the reason why we need a solution to solve the limitations of the single big context model.

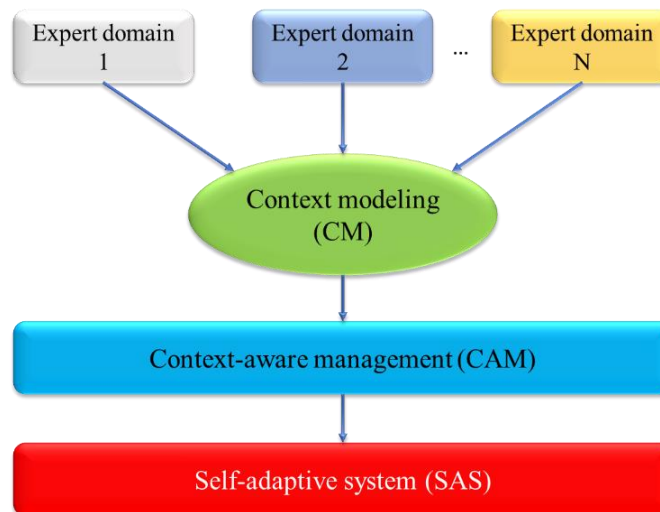


Figure 3.3: A single big context model in self-adaptive system

3.2.2 Separate concerns: the solution to solve the problems related to designing a single big context model

Nešković et al. introduced a solution for context modeling in complex SASs consisting of many independent context-aware applications [Nešković-2015]. They use an independent local context model (which is defined as an independent view) to the specific needs of a particular application. This solution paves the way for using a multitude of independent context models in context modeling. Each independent context model is used to present one user's scenario or one designer's viewpoint. In this thesis, we propose to split the current context model into many independent context models. In each model, the experts use knowledge of their expertise domain to describe the application scenario following their concerns (viewpoint). That solution can help the task of the designer become simpler. Each expert needs to focus only on their expertise domain. It makes the development and reconfiguration of context modeling become simpler, thus minimizing the impact on the operation of the system.

However, each expert uses different techniques or modeling tools to describe their viewpoint. So, the data type and structure of each model after the modeling process are different. We must convert these data into one standard format before adding to CAM. For this, we need to build a Context intermediate model (CIM) as a standard format for all viewpoints. The viewpoints must convert into standard data of CIM before the user can add it to CAM, as shown in figure 3.4. The usage of independent viewpoints to model each context domain at design time will reduce the impact on the operation of the system. This solution provides the reusability of each designer's viewpoint in building the application.

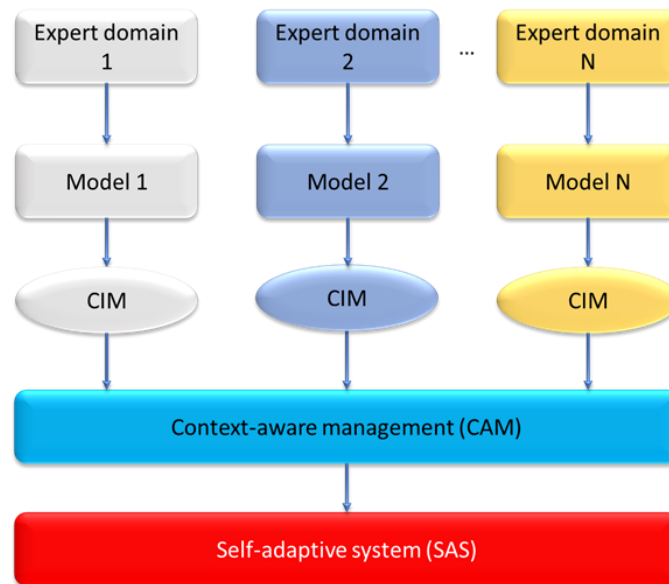


Figure 3.4: Using CIM as standard data for all viewpoints.

3.3 Special viewpoints in context domain

Special viewpoints in the context domain are a collection of specialized methods that have been defined by many different experts. We can use them to describe how context data are structured and maintained such as Scenario context, Business process modeling notation (BPMN), ConcurTasktree (CTT), Security and Task model, *etc.*

- **Scenario context:** Describes the operation of the user or situation of the user in the story, interacting with the device and user task. A design scenario builds must clarify the interaction points in the story and technology would help the user complete their task. The design scenario should describe information related to the user in the story and the interaction between user and device on the dynamic environment. Many works had been done in learning scenarios on context modeling such as [Chaabouni-2016], [Uchitel-2004], [Greiner-2014]. It shows that it is possible to describe a dynamic environment using the story.

- **Business process modeling notation (BPMN):** Is one of the most widely used processes modeling notations developed by the Object Management Group since 2005, which offers a range of notational detail to capture quite complex business processes. The main objective of BPMN is to provide the notation intuitive to the business users. Also, it represents the complicated process semantics process for both technical users and business users [Harris-2016]. One of its strongest features is its simplicity in supporting a notation language easy to understand and usable by people with different training fields [Marcinkowski-2012].

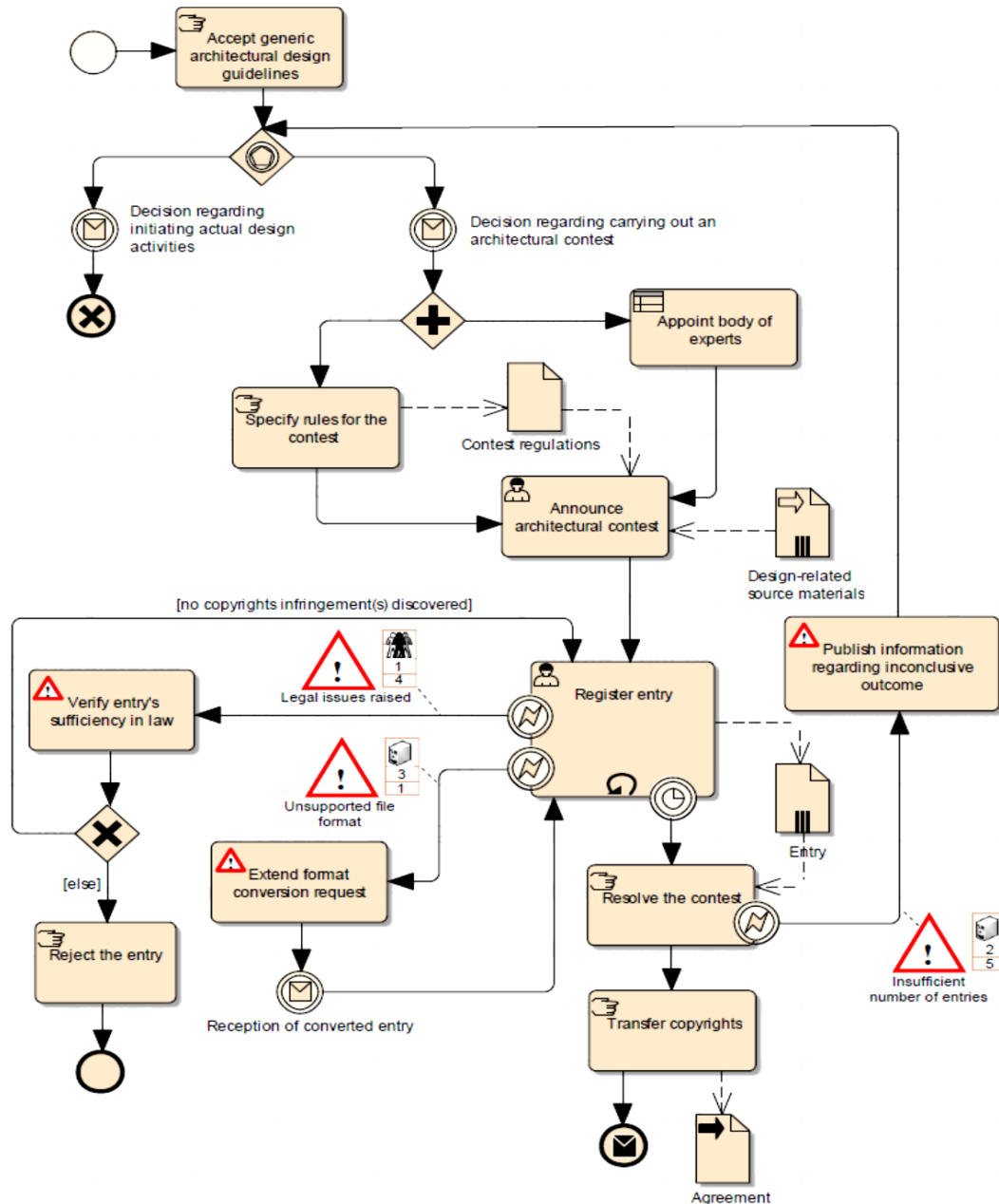


Figure 3.5: The procedure for managing architectural contests [Marcinkowski-2012]

David Schumm et al. show the methods and approaches using the graphical aspect of BPMN to model the BPEL process and facilitate the modeling of executable processes using BPMN without model transformations [Schumm-2009]. This approach used graphical representations to replace transformation techniques for building the BPEL processes. The result of this research shows that it is possible to create the BPEL process using icons, connecting components, and the semantics defined in BPMN.

Ruiz et al., [Ruiz-2012] describe BPMN as the industrial standard for modeling business processes that enable modeling problems on an abstract level and facilitate execution and reuse. They also emphasize that the BPMN tool can be used to support communication between domain experts and computer scientists. They show that BPMN supports a communication bridge between business users and technical people, who are responsible for the process application implementation, both using their own terminology.

- **Task model:** Is a specific model that represents the intersection between user interface design and more engineering approaches [Paternò-2002]. It provides for the designer a means of representing and manipulating a formal abstraction of activities which should be performed to reach the user objective.

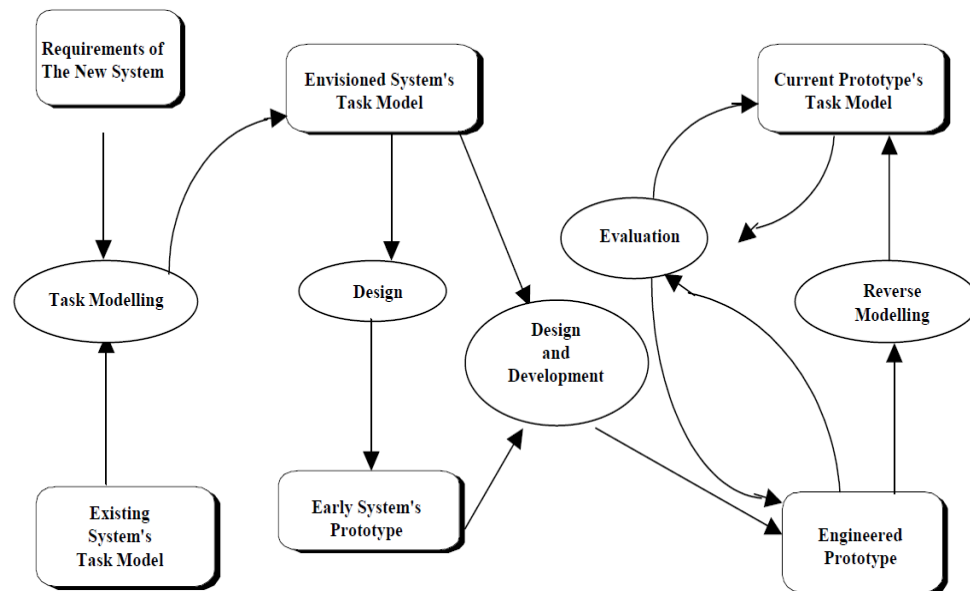


Figure 3.6: Use of task models in the design cycle [Paternò-2002]

ConcurTaskTrees (CTT) has been developed by Paternò is a notation for task model specifications. It has been developed to overcome the limitations of other notations previously used to build interactive applications. The main purpose of CTT is to be an easy-to-use notation with developer and user. It also supports the design of medium dimension applications in real industrial.

We may see that many contextual concerns have its specific model that we can use to describe a special viewpoint in the context modeling process. Therefore, we can use a specific model of each contextual concern to help experts on building special viewpoints. That is one of the reasons why we want to propose using special viewpoints in context modeling.

3.4 Using special viewpoints to separate concerns in context modeling.

As we analyzed in section 3.2.2, we can separate context concerns by using an independent Context model for each domain of context. And in our approach, we proposed a specific solution, as shown in figure 3.7, where we use different special viewpoints to model context. The designer in each special domain will use knowledge in their domain to design user scenario applications. When the designer uses an independent viewpoint to describe the context for each application, it can decrease the effect of relevant concerns on different applications.

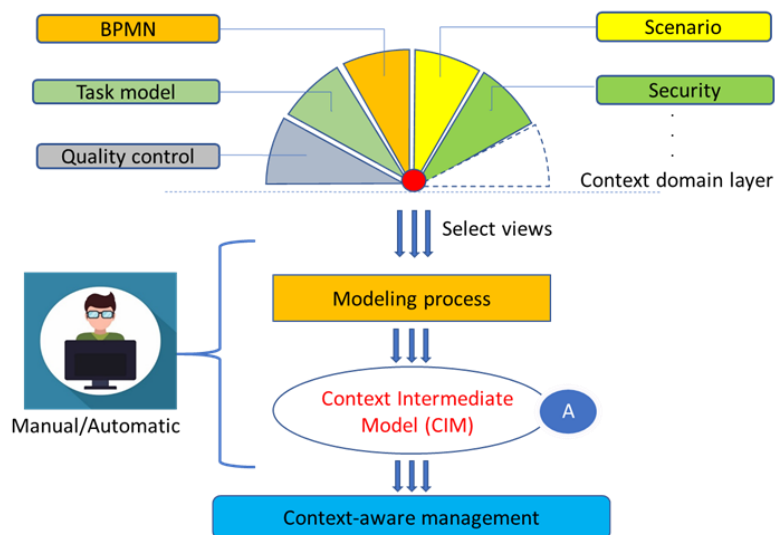


Figure 3.7: Using special viewpoints in context modeling

In our approach, the Context-aware management (CAM) is used to collect information from the context and manage the list of special viewpoints of applications. However, each designer uses a special viewpoint to model their user scenario application. Furthermore, each special viewpoint has a different structure, property, and language, so each viewpoint is described in an entirely independent way compared to other viewpoint. Therefore, the result of this independent design exists of multiple data types after the modeling process. To solve problems about convert and synchronize context information, we currently introduce a Context intermediate model (CIM), as shown in Figure 3.8, between expert specific models and the CAM. All special viewpoints (BPMN, CTT, etc.) must be converted into the format of CIM. The transforming into CIM model

process can be done manually by a designer or automatically by converting tools. In this approach, we developed some tools to convert data from specific viewpoints automatically (this is described more in detail in the implementation section).

3.5 Activity design: Architecture of self-adaptive system and context intermediate model

3.5.1 The model architecture of self-adaptive system

In our approach, we propose a new architecture of self-adaptive system (as shown in Figure 3.8). At design time, we use specific tools to convert each special viewpoint into the XML file, which follows the format of CIM. At runtime, all special viewpoints will be merged in context-aware management. The CAM is used to manage all special viewpoints and detect changing of the context [Rey-2017]. It also detects and solves conflicts between viewpoints before the CAM executes adaptation rules to SAS. In this approach, we propose to use two independent observation cycles of CAM and SAS. The observation of CAM was used to handles the context information. When the CAM finds a change in context information or adding and deleting special viewpoint from the user, it requires self-adaptive system reconfiguration. This solution can provide more time to the CAM to analyze the current context and preserve the responsiveness of SAS.

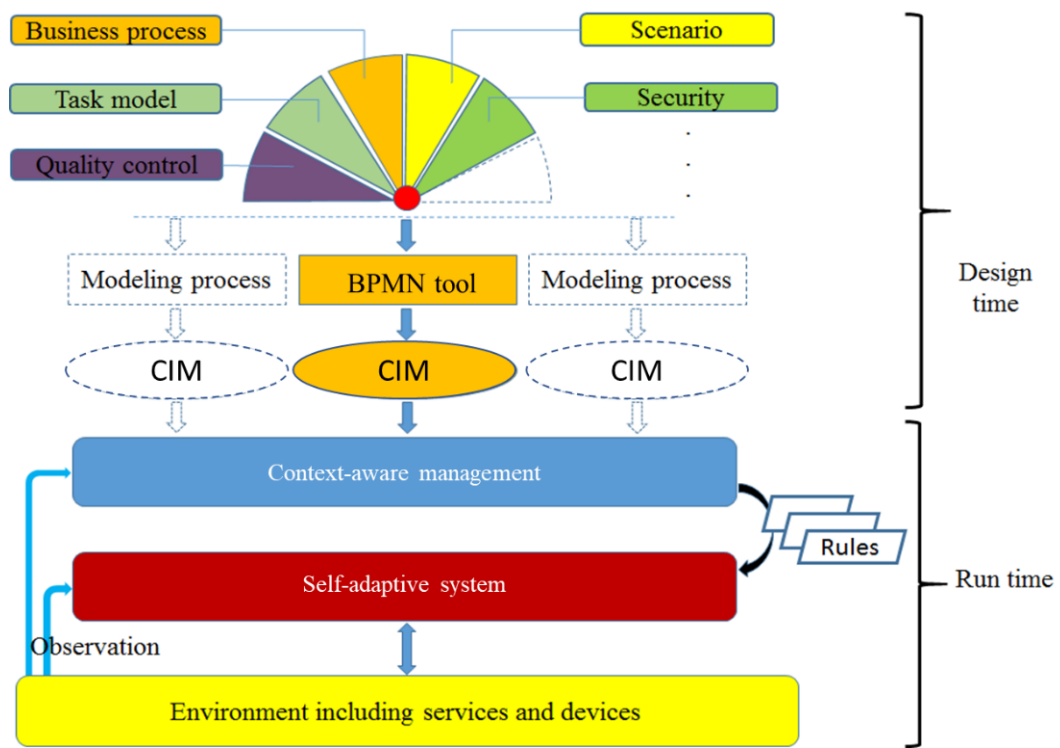


Figure 3.8: Model architecture of system

With the independence of the execution cycle of the SAS and CAM, the operations of the application layer are independent with the adaptation process. Thus, the system application can continue during the decision stage of CAM, and it is interrupted only during the implementation of the adaptation plan when the CAM detects the context switching. That means the SAS can reduce adaptation time and part of the problem about the consumption system. The CAM evaluates each predicate of the viewpoint to identify the current situation. Depending on the current situation, the list of adaptation rules will be applied, and SAS will give suitable action or decision.

3.5.2 Context intermediate model

In our approach, the Context Intermediate Model (CIM) is an XML description of the specific type of the Moore automaton described by the following 6-tuples $(S, s_0, \Sigma, \Lambda, \delta, g)$:

- S is a finite set of viewpoint states.
- s_0 ($s_0 \in S$) is a start state at t_0 .
- Σ is a set of input predicate vector that is defined as a finite set of predicates.
- $\Lambda: \Lambda \rightarrow (\Omega)$ is an output alphabet defined as a finite set of Objectives of State
- $\delta: F(\Sigma_i) \rightarrow s_i$ is a Surjective function mapping an input alphabet Σ to the state.
- $g: S \rightarrow \Lambda$ is an output function mapping each state to the output alphabet Λ .

The data flow of viewpoint is described in Figure 3.9. With viewpoint G_1 , the set of n predicate is an input alphabet defined as a finite set of predicates that are used to identify the states in S . With n predicates leads a maximum of 2^n possible vector of states. In this approach, we use a specific type of Moore automaton. We use a Surjective function to replace the transition function. With one state of viewpoint at one time, we care only about the situation of observation predicate. This is the reason why we use the Surjective function that was introduced by Nicolas Bourbaki (1968) to replace for transition function in Moore automaton.

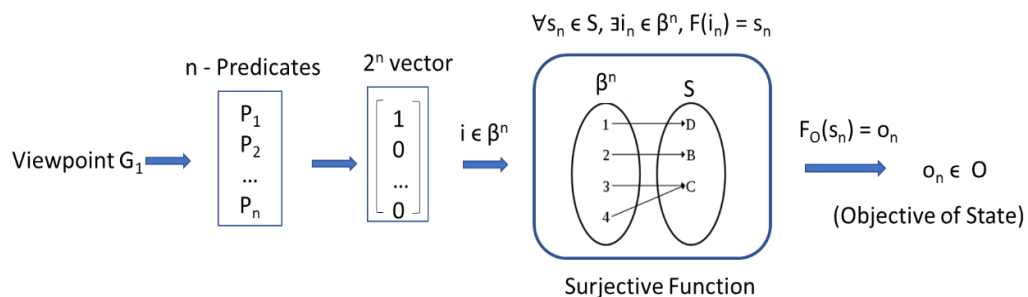


Figure 3.9: The data flow of viewpoint in CAM

Following Wikipedia, “a function f from a set X to a set Y is surjective, or a surjection, if for every element y in the codomain Y off there is at least one element x in the domain X off such that $f(x) = y$. It is not required that x be unique; the function f may map one or more elements of X to the same element of Y ” (wikipedia.org/wiki/Surjective_function).

In the case we apply K different viewpoints in one system, we need K different tools to convert information from special viewpoints into CIM format. Both use the Surjective function to replace the transition function of normal Moore automaton. However, each tool uses different algorithms to convert information for each special viewpoint due to different structures, formats.

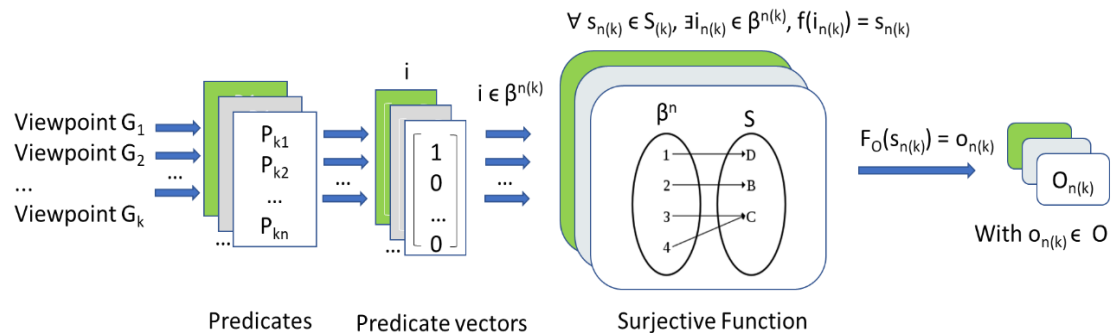


Figure 3.10: Each viewpoint is managed independently in CAM

To understand the application of Moore automata theory for the Context intermediate model, we might see in the following example.

Example 1:

The designer needs to model a user’s scenario where user comes to the laboratory to take some tasks. The user must provide an ID card to open the door. When the door opened, the system activates the Tools for the user to be able to use.

We have three Predicates:

- Predicate 1 (P1): Check ID (*The system will check the ID card provided: “true or wrong ID”*).
- Predicate 2 (P2): Check door opened (*The system will check the Door situation: “open or close”*).
- Predicate 3 (P3): Tool activate? (*The system will check the Tool activate situation: “activated or deactivated”*).

And it has a total of 2^3 states. Normally in this example, it will have eight states since it has three predicates. However, we use a specific notation to merge some states. For example, when we write $P1 \& \overline{P2}$, we describe two states defined by $P1 \& \overline{P2} \& P3$ and $P1 \& \overline{P2} \& \overline{P3}$.

- $s_1: \overline{P1}$ (not P1, this state happens when the system detects ID supported wrong or not valid).

- $s_2: P1 \& \overline{P2}$ (P1 and not P2, this state happens when the system detects ID true but the Door not opened).

- $s_3: P1 \& P2 \& \overline{P3}$ (P1, P2 and not P3, this state happens when the system detects ID true, the Door opened, but the Tool still deactivated).

- $s_4: P1 \& P2 \& P3$ (P1, P2 and P3, this state happens when the system detects ID true, the Door opened, and the Tool activated).

- The user's scenario above can be captured by Moor automaton following our description, then we have:

$$M = (S, s_1, \Sigma, \Lambda, \delta, \lambda)$$

Where:

$$- S = \{s_1, s_2, s_3, s_4\}$$

$$- \Sigma = \{P1, P2, P3\}$$

- $\Lambda \rightarrow$ (observation rules, adaptation rules)

$$- \delta: s_n * \begin{pmatrix} P1(t) \\ P2(t) \\ P3(t) \end{pmatrix} = s_{n+1} \text{ (where } P_i(t) \text{ is the value of predicate at the time } t)$$

- $\lambda = \{s_i, \text{output function}\}$ (i: the number of states in the set of state)

Figure 3.11 represents a diagram of CIM schema that is used as an output standard data of the converting tool in our previous work [Rey-2017]. In this diagram, each special viewpoint named "Viewpoint" which will be structured by three parts: "Description", "Predicate", and "State". The "Description" part stores programmers' comments and description information that is non-structured information in natural language for the designer (can be used for the name of the view, version, etc.). The "Predicate" part stores all identified predicates (through predicatedID unique for a view) and "ObservationRulesSet" which included the rules for describing how to build the observation mechanism that will return the value of the predicate. The "State" element is structured by two parts "Evaluation" and "AdaptationRulesSet". The "Evaluation" stores the information of the vector of predicates values, which is used to identify the state. And the "AdaptationRulesSet" stores adaptation rules that will be deployed within the SAS when the state will be identified and executed by it. The "Attributes" of viewpoint, Predicate, and State stores only the ID of them.

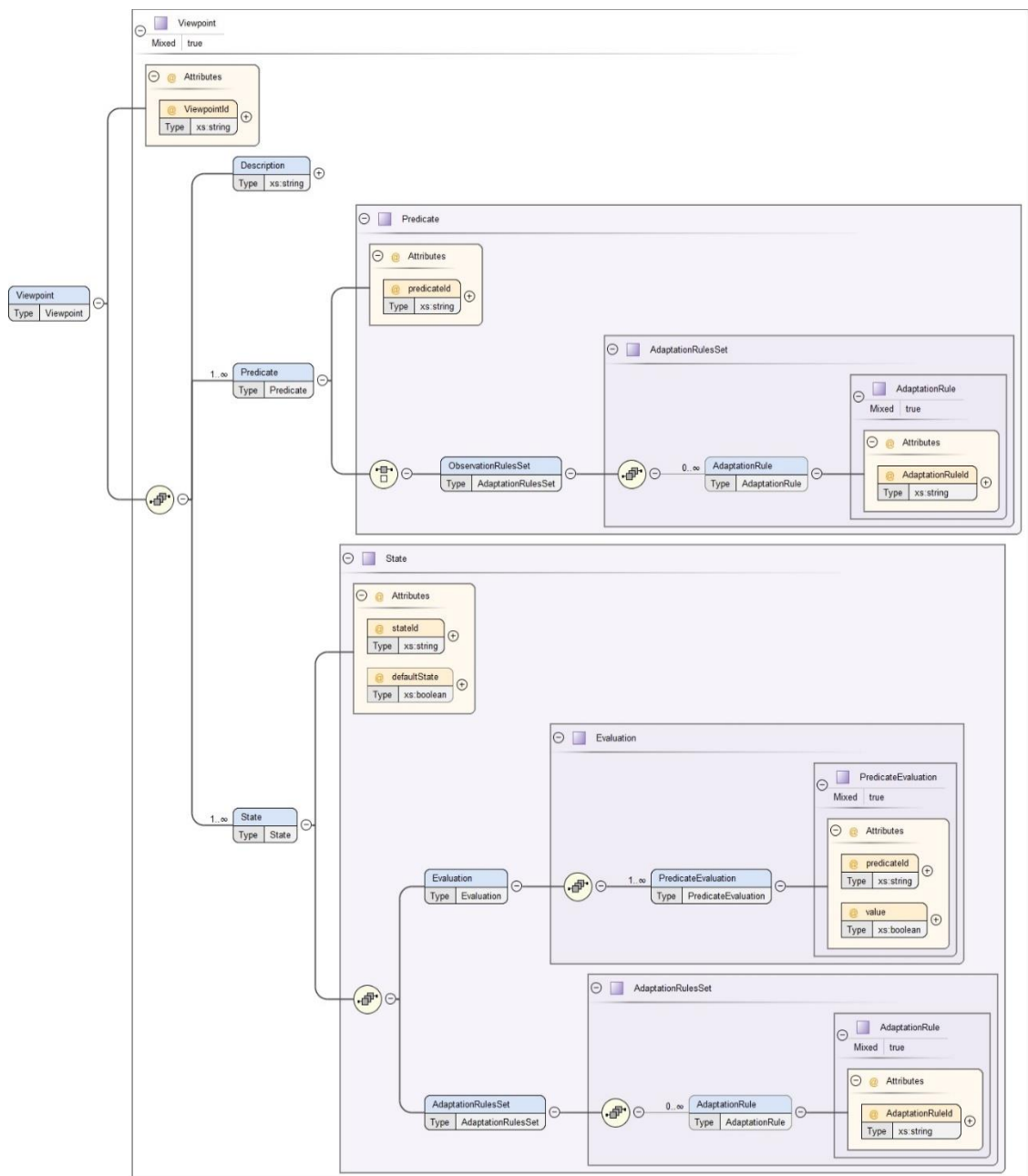


Figure 3.11: The XML schema of context Intermediate model

Each AR will store a set of rules to ensure that the SAS system can adapt the function of services corresponding to each situation of the current state. The data of adaptation rules on each state depend on each designer and the hardware of each application, as shown in the below example. This example presents an AR in our project [Continuum-2012], [Ferry-2012]. This AR is structured into two parts: Firstly, the system will search all devices or services that have names that mention “switch” and “lamp” (*switch = switch**; *lamp = lamp**) in all running devices or services available for the system.

Secondly, if there are at least one switch and one lamp is discovered, the system will make a connection between the switch and the lamp which was detected in the previous step. For this, the system will link the lamp method (*lamp.SetTarget*), allowing to illuminate it, with the event (*Status_Event*) emitted by the switch during its use (which corresponds to a change of the state).

```
<AR ARId="switch_lamp_checking">
  <![CDATA[
    switch = switch*
    lamp = lamp*
    advice lamp_switch (switch, lamp):
      switch.^Status_Event -> (lamp.SetTarget)
  ]]>
</AR>
```

3.6 The methods of generating the context model from the models of concerns

Many special viewpoints in the context domain have their specific language to model context. In this section, we present two concrete examples illustrating the transformation of a domain-specific model into context intermediate model. The two specific models used are Business Process based on Business Process Modeling Notation (BPMN) and Task Model based on ConcurTaskTree (CTT) notation.

3.6.1 Generating the context model with the Business process viewpoint

The Business Process Modeling Notation (BPMN) is a popular model that provides a graphical notation for expressing a business process in the Business process diagram. The main purpose of BPMN is providing a process management tool for both business users and technical users. The work of A. White [White-2004] provides a solution to define and execute business processes through XML Process Definition Language (XPDL). It is a standard mechanism to describe an analysis of a business process.

In our approach, the BPMN is used as a specific tool to describe a specific viewpoint which can be an application scenario or user scenario [Do-2019]. However, the BPMN tool is only simple and suitable for the businessman on a daily basis. The XPDL data exported from the BPMN tool still has much unnecessary information such as location, size of the graphical object, *etc.* Therefore, the execution of information from BPMN to CAM is a complicated process. In this approach, we developed a specific tool by C# to convert the XPDL file from BPMN into an XML file following CIM schema. We do not map all graphical objects of BPMN; we focus only on popular elements of BPMN. With a big scenario of business process, we propose to use Sub-process objects to limit the number of objects on one business process.

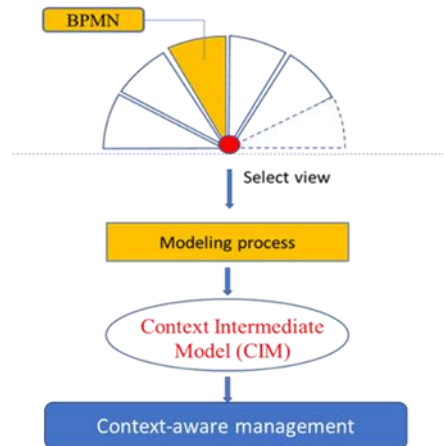
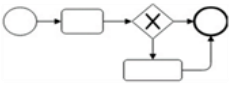







Figure 3.12: Using BPMN to describe specific viewpoint in context modeling

In this approach, we propose using CIM as standard data for all viewpoints. Therefore, we need to convert XPDL data from the BPMN viewpoint into an XML file following standard data of CIM. Table 3.1 shows the contribution of A. White on mapping from BPMN to XPDL, and our work is mapping XPDL to standard data of CIM.

Table 3.1: The mapping from BPMN to CIM standard data type

BPMN graphical object	XPDL mapping	CIM standard
 BPMN process	<code><WorkflowProcess></code>	<code><View xmlns=""></code>
Start event 	<code><Activity></code> <code><Route/></code> <code></Activity></code>	<code><State " " ></code> <code><Evaluation/></code> <code><AdaptationRuleSet/></code> <code></Predicate></code>
Task 	<code><Activity></code> <code><Implementation></code> <code></Implementation></code> <code></Activities></code>	<code><Predicate " " ></code> <code><ObsRulesSet></code> <code></ObsRulesSet></code> <code></Predicate></code>
Decision 	<code><Activity></code> <code><TransitionRestriction></code> <code><Split Type="XOR"/></code> <code></TransitionRestriction></code> <code></Activities></code>	<code><Predicate " " ></code> <code><ObsRulesSet></code> <code></ObsRulesSet></code> <code></Predicate></code>
Transition 	<code><Transition/></code>	<code><State/> to <State/></code>
Sub-Process	<code><Activity></code> <code><Implementation></code> <code><SubFlow/></code> <code></Implementation></code> <code></Activities></code>	<code><State></code> <code><Evaluation/></code> <code><AdaptationRuleSet/></code> <code><State/></code>
Stop event 	<code><Activity></code> <code><Route/></code> <code></Activity></code>	<code><State " " ></code> <code><Evaluation/></code> <code><AdaptationRuleSet/></code> <code></Predicate></code>

To clarify the relationships between BPMN, XPD and CIM, we might see in the below example of a business process modeled by BPMN tool. It will be analyzed and mapped to CIM standard.

Example 1:

The BPMN process starts with an order being received (as shown in Figure 3.13). The order data is then sent through a “Check availability” Task. The order data is passed to the “Ship article” Task if it was available on the system. Then the order data continue sending through an application “Financial settlement”. After that, the process is finished in “Payment received”.

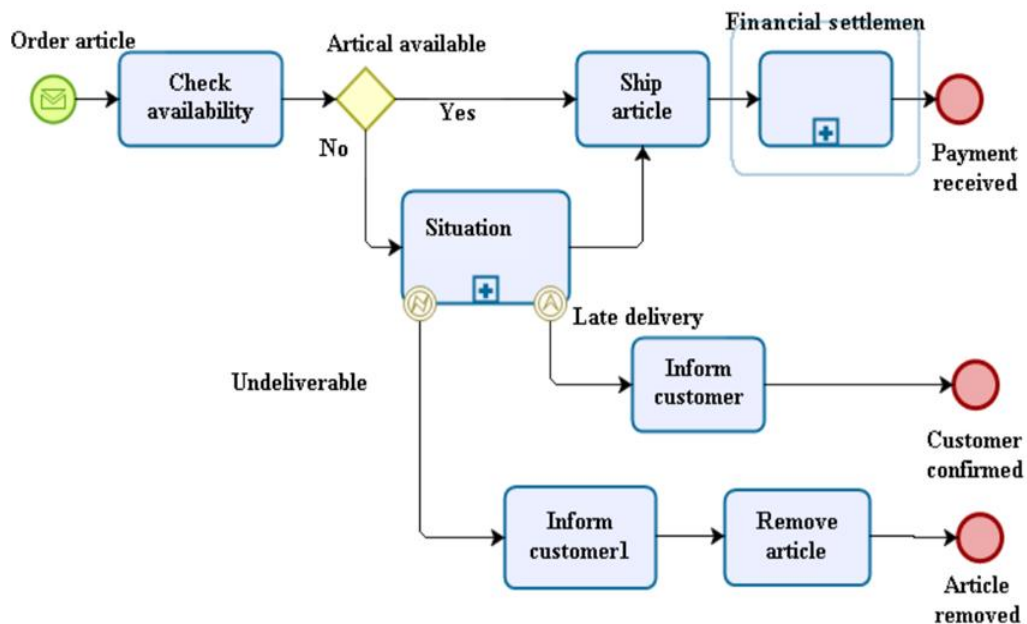


Figure 3.13: The E-Order process that is described by graphical notation.

If the order is not available, the order data enters through a “Situation” task. This task may generate a Process Error or Escalation, as shown by the Intermediate Events concatenated to the boundary of the “Situation” task. A task to “Inform customer” follows the “Process error” or “Escalation intermediate event”. If the system detected a problem with the order process in this step, then a Task to “Remove article” is performed. Order information is passed to action “Article removed” to complete the process. The details of the mapping process will be shown in the next section.

- After the designer uses any Business process modeler tool to build a business process application, they can use export data function in the modeling tool to convert the business process graphical into a text file as XPD. With the above example, the E-Order process is exported into XPD file by Business process modeler tool, as shown in the below XPD file:


```

<?xml version="1.0" encoding="utf-8" ?>
<Package xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
  <WorkflowProcesses>
    <Activities>
      <Activity Id="8d711814" Name="Order received" />
      <Activity Id="6b6a8fb3" Name="Check availability" />
      <Activity Id="7535eb5c" Name="Artical available" />
      <Activity Id="14284d81" Name="Ship article" />
      <Activity Id="1a47fa66" Name="Payment received" />
      <Activity Id="ad7ae05e" Name="Inform customer1" />
      <Activity Id="642b2822" Name="Inform customer" />
      <Activity Id="c414055e" Name="Customer confirmed" />
      <Activity Id="a6426903" Name="Remove article from catalogue" />
      <Activity Id="c6c8d035" Name="Article removed" />
      <Activity Id="bdf0288b" Name="Procurement" />
      <Activity Id="08e51851" Name="Error" />
      <Activity Id="8303bdb7" Name="Escalation" />
      <Activity Id="9d9110f0" Name="Financial settlement" />
    </Activities>
    <Transitions>
      <Transition Id="dcfc751e" From="8d711814" To="6b6a8fb3" />
      <Transition Id="5b535525" From="6b6a8fb3" To="7535eb5c" />
      <Transition Id="7021af00" From="7535eb5c" To="14284d81" Name="Yes" />
    </Transitions>
  </WorkflowProcesses>
</Package>

```

After we have the XPDL file from the Business process modeler tool, we continue to convert the XPDL file into XML format following the standard data type of context intermediate model. This process can be done manually by designer or automatically by using an automatic converting tool that will be described in chapter 6. The result of converting process is presented in below XML file.

```

<?xml version="1.0" encoding="UTF-8" ?>
<Viewpoint xmlns="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
  <Description> Example viewpoint </Description>
  <Predicate predicateId = "Order received" EvaluationFrequency="0">
  <Predicate predicateId = "Check availability" EvaluationFrequency="0">
  <Predicate predicateId = "Artical available" EvaluationFrequency="0">
  <Predicate predicateId = "Ship article" EvaluationFrequency="0">
  <Predicate predicateId = "Payment received" EvaluationFrequency="0">
  <Predicate predicateId = "Inform customer1" EvaluationFrequency="0">
  <Predicate predicateId = "Inform customer" EvaluationFrequency="0">
  <Predicate predicateId = "Customer confirmed" EvaluationFrequency="0">
  <Predicate predicateId = "Remove artivale from catalogue" EvaluationFrequency="0">
  <Predicate predicateId = "Article removed" EvaluationFrequency="0">
  <Predicate predicateId = "Procurement" EvaluationFrequency="0">
  <Predicate predicateId = "Esc Procureme" EvaluationFrequency="0">
  <Predicate predicateId = "Err Procureme" EvaluationFrequency="0">
  <Predicate predicateId = "Financial settlement" EvaluationFrequency="0">
  <State stateId="non Order received" defaultState="true">
  <State stateId="non Check availability" defaultState="true">
  <State stateId="non Ship article" defaultState="true">
  <State stateId="non Financial settlement" defaultState="true">
  <State stateId="non Payment received" defaultState="true">
  <State stateId="Payment received" defaultState="true">

```

In this section, we present the work of BPMN as only a simple example of graphical notation that model business processes and that identical works should be able to be made with other models as long as they provide a manipulate model (XML, XPDL, JSON, etc.).

3.6.2 Generating the context model with Tasks model viewpoint

Currently, many Task models are used to design interactive applications; Among them ConcurTaskTrees (CTT) notation [Paternò-2002] is the most popular tool. It is used to design interactive applications specifically tailored for user interface model-based design.

We might see that ConcurTaskTrees has been developed based on taking into account the previous experience in task modeling and adding several new features in order to obtain an easy-to-use and powerful notation to describe the dialogue in interactive systems [Nobrega-2005]. Moreover, it also supports methods that indicate how to derive user interfaces for different platforms from ConcurTaskTrees specifications. The user can easily build an application with CTT because it provides the hierarchical structured of tasks, which can be interrelated through a powerful set of operators that describe the temporal relationships between subtasks. In addition, when designers use the CTT tool to develop an application, they can indicate a wide set of optional task attributes, such as the type, category, manipulation of objects, and time requested for performance. The ConcurTaskTrees structured in three parts: Hierarchical structure; Graphical syntax; Concurrent notation

We use the ConcurTaskTrees tool to describe a special viewpoint that is related to using graphical notation in building the application, as shown in figure 3.14.

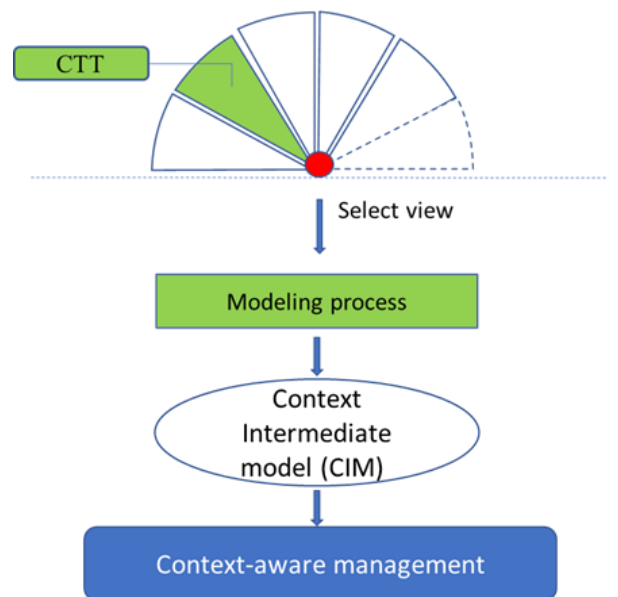


Figure 3.14: Using CTT to describe specific viewpoint in context modeling

And, we also establish a specific tool which was developed by C# to convert information from CTT into an XML file following CIM schema. In our approach, we use the mapping

solution of Nobrega et al. to find the relationship between tasks in a CTT process as shown in figure 3.15.

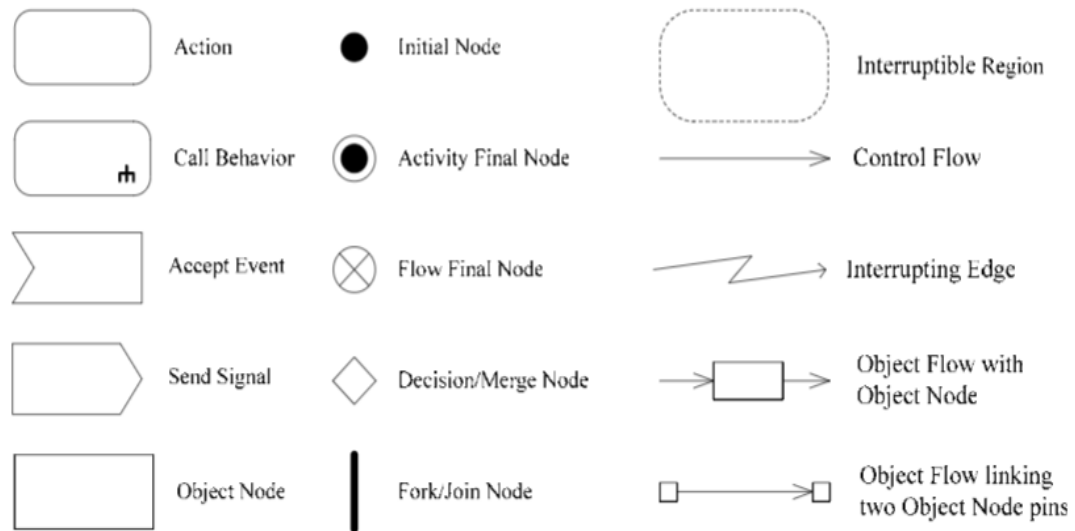


Figure 3.15: The UML notation for activity diagram [Nóbrega-2005]

In our mapping from CTT data to CIM standard, each task element in the CTT model is considered as a Predicate (P) in CIM schema. The composition of Predicates must take in account the precedence of temporal operators in CTT include: >> (Enabling), [> (Deactivation), |> (Suspend-Resume), [] (Choice), |[|] (Concurrency with information exchange), |=| (Order Independence), ||| (Independent Concurrency). With the description of temporal operators following:

- Independent Concurrency (P1 ||| P2): Actions corresponding to the situation of two Predicates can be performed in any order without any specific constraint.
- Choice (P1 [] P2): It is possible to choose from a set of Predicates. When once the choice has been made, the action corresponding to the predicate chosen can be performed. The other actions are not available at least until action corresponding the predicate chosen has been terminated.
- Concurrency with information exchange (P1 |[|] P2): Two actions corresponding to two predicates can be executed concurrently, but they have to synchronize in order to exchange information.
- Order Independence (P1 |=| P2): Both actions corresponding to two predicates have to be performed, but when one is started then it has to be finished before starting the second one.
- Deactivation (P1 [> P2): The action corresponding to P1 is definitively deactivated once the second action has been performed.

- Enabling (P1 >> P2): In this case, one action corresponding to P1 enables a second one when it terminates.

- Enabling with information passing (P1 []>> P2): In this case, the action corresponding to P1 provides some information to action in P2 other than enabling it.

- Suspend-Resume (P1 |> P2): This operator gives action corresponding to P2 the possibility of interrupting action corresponding to P1, and when action in P2 is terminated, the action in P1 can be reactivated from the state reached before the interruption.

To make clearly on using CTT to model context and how to convert into standard data of CIM, we might see in the below example of a business process modeled by the CTT tool.

Example 2:

The Hotel reservation process starts with action “Select Room Type” of the client (as shown in figure 3.16). Have two options for the client to choose that are “Single room” and “Double room”. The system will show the situation of the room in “Show availability”. If the room available on the system is selected, the client will confirm “Select room” to complete the process.

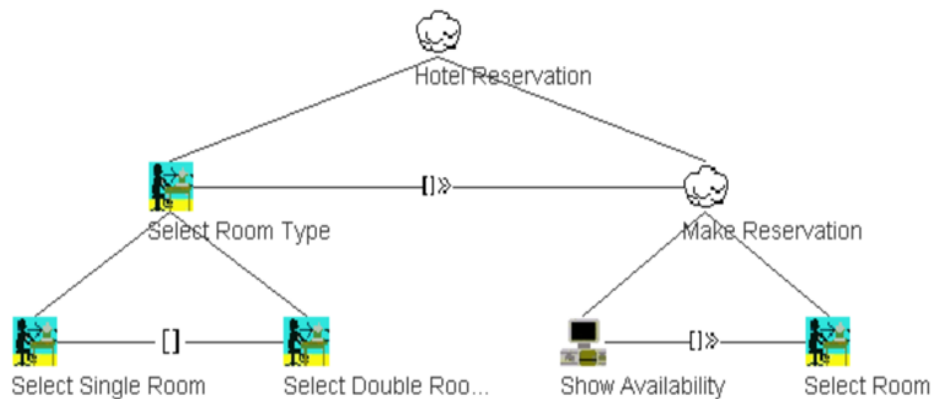


Figure 3.16: The Hotel reservation process that is described by ConcurTaskTrees

The CTT tool also supports an exporting data function to help the designer to choose a suitable data type format. In our approach, we choice export CTT data into an XML file. The XML file from CTT is structured into <Task> and <SubTask> section with detail information as shown in below XML file. After we have the XML file from the CTT tool, we continue to convert this file into other XML file following the standard data type of the context intermediate model. This process can be done manually by designer or automatically by using an automatic converting tool that will be described in chapter 6 (this tool is different from the converting tool using for BPMN viewpoint).

The XML file of the Hotel reservation process that converted by CTT tool:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<TaskModel xmlns="http://giove.isti.cnr.it/ctt" xmlns:coop="http://giove.isti.cnr.it/ctt-coop"
  <Task Identifier="Hotel Reservation" Category="abstraction" Iterative="false"
    <Name>name</Name>
    <SubTask>
      <Task Identifier="Select Room Type" Category="interaction" Iterative="false"
        <Name>name</Name>
        <TemporalOperator name="SequentialEnablingInfo"/>
        <Parent name="Hotel Reservation"/>
        <SiblingRight name="Make Reservation"/>
        <SubTask>
          <Task Identifier="Select Single Room" Category="interaction"
            <Name>name</Name>
            <Parent name="Select Room Type"/>
            <SiblingRight name="Select Double Room"/>
          </SubTask>
        </SubTask>
      </Task>
      <Task Identifier="Make Reservation" Category="abstraction" Iterative="false"
        <Name>name</Name>
        <Parent name="Hotel Reservation"/>
        <SiblingLeft name="Select Room Type"/>
        <SubTask>

```

We get the result of conversion into the CIM format of the Hotel reservation process by using the automatic converting tool as shown in the below XML file.

```

<?xml version="1.0" encoding="UTF-8" ?>
<Viewpoint xmlns="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <Description> Example viewpoint </Description>
  <Predicate predicateId = "Select Single Room" EvaluationFrequency="0">
  <Predicate predicateId = "Select Double Room" EvaluationFrequency="0">
  <Predicate predicateId = "Show Availability" EvaluationFrequency="0">
  <Predicate predicateId = "Select Room" EvaluationFrequency="0">
  <State stateId="non_Select Single Room" defaultState="true">
  <State stateId="non_Select Double Room" defaultState="true">
  <State stateId="non_Show Availability" defaultState="true">
  <Evaluation>
    <PredicateEvaluation predicateId = "Select Double Room" value = "true"/>
    <PredicateEvaluation predicateId = "Show Availability" value = "false"/>
  </Evaluation>
  <AdaptationRulesSet></AdaptationRulesSet>
</State>
  <State stateId="Select Room" defaultState="true">
  <Evaluation>
    <PredicateEvaluation predicateId = "Select Double Room" value = "true"/>
    <PredicateEvaluation predicateId = "Show Availability" value = "true"/>
    <PredicateEvaluation predicateId = "Select Room" value = "true"/>
  </Evaluation>
  <AdaptationRulesSet></AdaptationRulesSet>

```

We provided two examples of BPMN (in section 3.6.1) and CTT to illustrate the fact that we can help designers by partially automating the transformation between their specific models and the CIM. And, that of course, it should not only be able to extend to other languages describing business processes or task models but also to other areas of concern such as security, energy management, etc.

3.7 Valid and Invalid state

3.7.1 Problems related to observation

As described in section 3.5, at one moment each viewpoint has one state identified based on the vector of predicate values. We get these values from the evaluation of all the predicates of the viewpoint (described in the CIM). However, we might see that all combinations related to the predicate value vectors do not necessarily correspond to all valuable states of the viewpoint. Indeed, it is very likely that certain combinations have no meaning to see either theoretically impossible.

In our previous project [Continuum-2012], the viewpoints with n predicates have exactly 2^n states, or if they had less of them, it was that grouped states. And, the 2^n combination was valid. The advantage of this approach is that it can detect the cases where the designer of the viewpoint had forgotten or not a state. However, the disadvantage of this solution is that it makes it necessary to consider all states as valid states. However, as we have seen in our work on the conversion of specific business models, many states defined by vector predicates. They have no real meaning or even correspond to an impossible situation. The trick, used by designers applying this model, was to associate it with any rules that did not make sense. It works globally well since, normally, one should never end up in one of its states. But this forces the designer to identify all these states, and nothing happens if the system is found in one of these states.

In this approach, we separate states of viewpoint into two types. Normal states that correspond to possible situations that the designer wishes to detect in order to deploy adaptation rules, we call they are valid states. And all other states are invalid states. These invalid states are unnatural states by nature or non-conforming states in the natural order of execution.

The use of these invalid states allows us to identify some of the errors of the observation mechanism. Indeed, the observation mechanism is based on physical sensors, it is likely that erected values are raised (due to noise signal, sensor error or other factors outside the expected scenario) and thus this leads to errors and then the observation falls on a state that has not been described as the normal state.

3.7.2 The state identification

The CAM needs to manage both valid and invalid state to provide exactly a reaction for self-adaptive system. For this reason, we must modify our Context intermediate model with adding the invalid state, as shown in figure 3.17. However, when we introduce the invalid states, we offer two possibilities to designers:

1) They identify all the invalid states as they did for the states left empty in our previous model [Continuum-2012]. This does not save time for the design, but it allows us to detect the states that have been forgotten (and also allows managing invalid states).

2) They indicate that all other states (different from the valid states) are invalid states (InvalidstateDefault). And in this case, they do not need to identify all invalid states. These states correspond to all unlisted states. This saves time for the design and greatly simplifies the designer's task. On the other hand, although the management of invalid states remains assured, it will no longer be possible to check if the designer has forgotten a state.

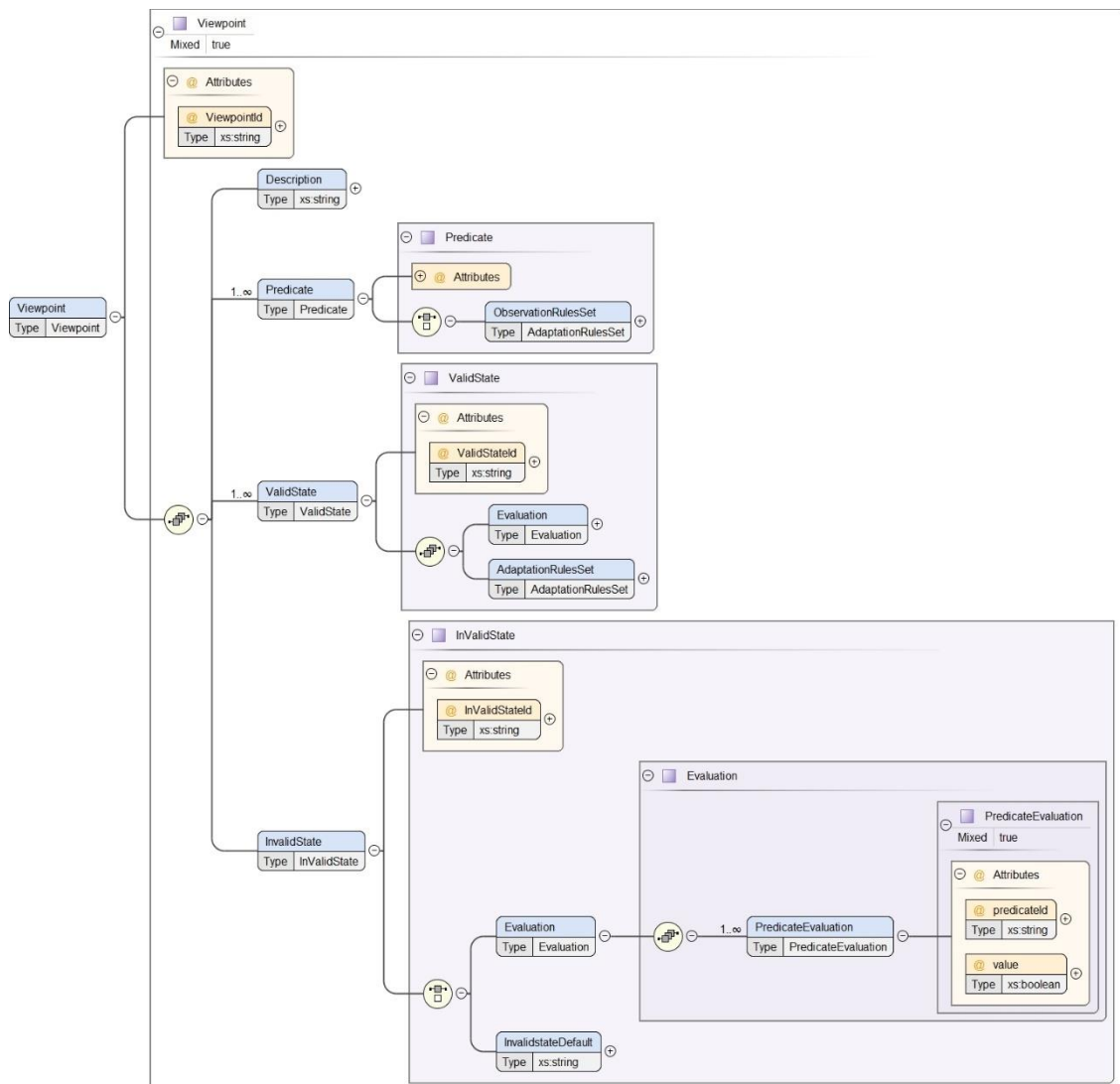


Figure 3.17: The updated version of CIM with adding Invalid state

It is necessary to react to the detection of invalid states, such as log off the error of the system and inform for admin or user about the problem of the system. The CIM provide state ID (as shown in example 1 and 2 in section 3.6) for every state (both valid and

invalid state) based on analyzing application scenarios (viewpoints). And, when the viewpoint is added into CAM, the CAM can detect the valid and invalid states through state ID.

There are several cases with the appearance of invalid states that can occur:

- The same invalid states appear continuously during several cycles of observation. In this case, the problem is probably an observation system error related to the predicate of viewpoint. The CAM can send error situations to admin or user to help them in detecting and solving the problem of the observation system.

- Some invalid states occur randomly during several cycles of observation. In this case, the problem is probably resulting from a noise signal. The CAM can send the noise to inform the user or admin; It can also set an emergency case and send a checking system requirement to admin.

3.7.3 The proposed solution to filter states and react to invalid states.

After the “Identify state” step to clarify the current state is valid or invalid state. We need a “Filter state” step (as shown in figure 3.18) to collect the valid state for adaptation and invalid state to inform the system problem to user and admin, as we mentioned above.

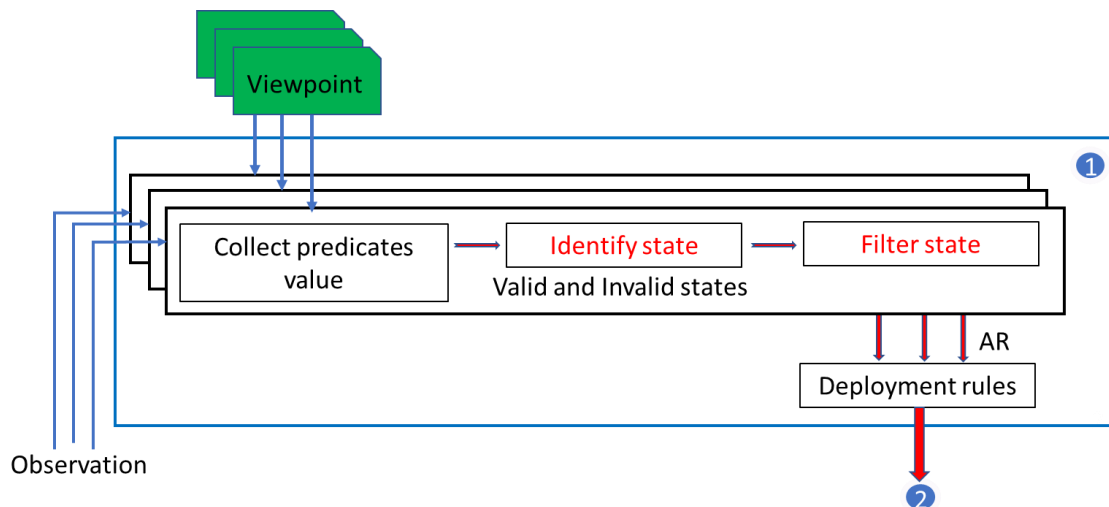


Figure 3.18: The filter state in CAM

The details about our proposal for filtering the valid and invalid state process at runtime described in the below example.

Example 3:

- With one viewpoint store n predicate (nP), we have 2^n States: S_i ($i = 1$ to 2^n), with “ S_i may be valid state or invalid state”. We suppose to use two counter C and C_d to detect

the changing of invalid state. The main goal of this detecting is keeping or changing the previous state to have suitable Adaptation rules, as shown in Figure 3.19.

- In case of invalid state repeat many times:

+ If only one invalid state appearance on many observation cycles: Add a Counter (C) of each invalid state, if situation of state does not change in next cycle of observation then $C = C + 1$; Check Counter: if $C > K$ (K is constants proposed by designer depend on application) \rightarrow Send command Error to user \rightarrow Check "Set of predicate in this state" \rightarrow Detect problem of system.

+ If many Invalid states appear random: Set a Counter of random invalid state (C_d) and increase C_d after each Invalid state. If $C_d > Q$ (Q is constants proposed by designer depend on application) \rightarrow Send Error and set Emergency state: wait for checking system.

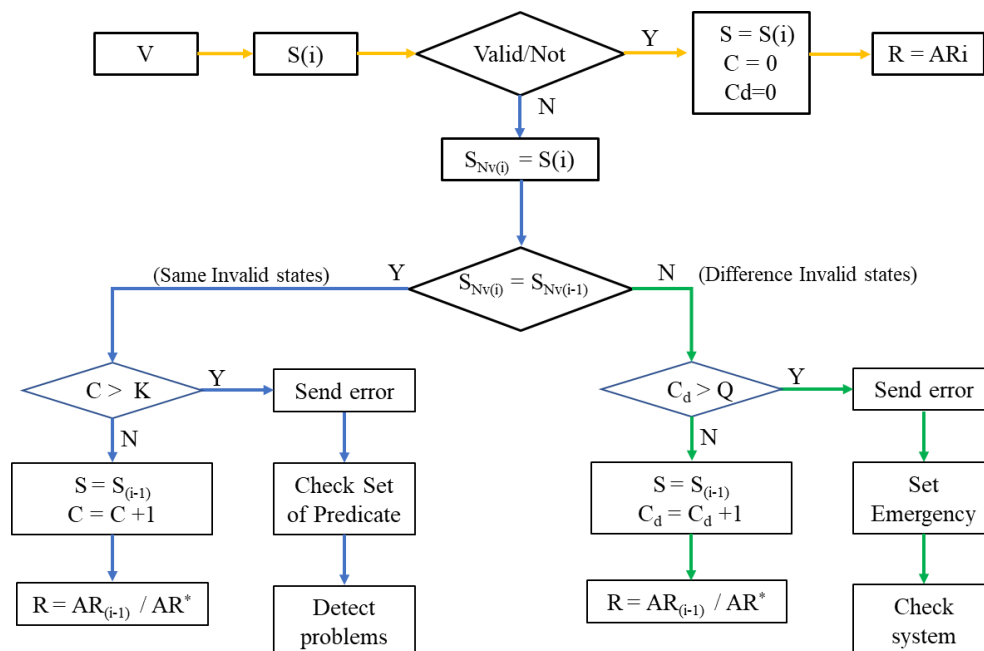


Figure 3.19: Detection Invalid states algorithm

3.8 Chapter conclusion

Using independent viewpoints in context modeling can solve the problems mentioned in the research question two that related to simplifying the expert's task. Each expert focuses only on their knowledge domain to model the context of their application. Moreover, using independent viewpoints in context modeling can make the context model development and reconfigure to becoming more straightforward.

Within this thesis, the data of a specific viewpoint is converted into an XML file following the Context intermediate model schema automatically. It is a critical step to support

CAM in handling contextual information from different specific viewpoints. After the conversion and validation process, the viewpoint will be added into CAM depending on the application requirements and user needs. Next, the CAM takes a filter state process to detect all valid and invalid states. Using CAM to support SAS on manage and handle contextual information is a solution that can help SAS focusing on adaptation. Therefore, it can help improve the responsiveness of SAS that mentioned in the fifth research question.

In the CAM, we have a combination of many independent viewpoints and each viewpoint designed by an independent designer. Therefore, the possibilities of conflict between viewpoints applied in one system at one time are always exists. Detecting and solving the conflicts between viewpoints will be presented in the next chapters.

CHAPTER 4: Conflict detection and solution

In chapter 3, we have illustrated the using independent viewpoints on context modeling to separate concerns and simplified the expert's task. At the end of the data conversion process, we used a filter state algorithm to get the valid state of each viewpoint. Each valid state stores a set of adaptation rules corresponding to the situation of predicates that are observed to take the goal of the state. The viewpoints are designed by different designers, and we use multiple viewpoints into one system, so the possibilities of conflict between viewpoints at one time are always exists. In this chapter, we introduce the solution to detect the conflicts between viewpoints in CAM. This solution is also a part of the answering to the fourth research question.

How to detect and solve the conflicts between adaptations at runtime?

This chapter is organized as follow: Section 4.1 presents the conflict abilities of the viewpoints in self-adaptive system; Section 4.2 presents the state of the art related to the description of adaptation's goal; Section 4.3 introduces the definition of objective of state; Section 4.4 presents the conflict detection algorithm; Section 4.5 presents the chapter conclusion.

4.1 Conflicts between adaptation rules of viewpoints in self-adaptive system

In our everyday life, when we build a self-adaptive system that works in a dynamic environment, we have to consider all of the users' different situation in that environment to provide them the exact services that they need. It requires us to use a big context model to cover all of the users' different scenarios. However, building a single big context model still has many limitations as we analyzed in chapter 3. Therefore, we proposed using multiple special viewpoints in context modeling to replace the single big context model. In our approach, each special viewpoint is built independently by a different designer. Each viewpoint relates to a different scenario setup, but they might have to operate in one system at the same time to provide services to the users, and the viewpoints are managed independently in CAM. Typically, after the "Filter state" process, we acquire the set of adaptation rules of each viewpoint's state as mentioned in chapter three. However, at one moment, the adaptation rules of one viewpoint can conflict with the other viewpoints because they could be designed based on the different users' scenarios by different designers. Therefore, the action performed

through adaptation rules of the state in this viewpoint could be impact or conflicted the action of other viewpoints. In that case, we might say that there is a conflict between the adaptation rules from the different viewpoints in self-adaptive system. And we need a solution to detect and to solve these conflicts. However, we might see that it is impossible to solve the inconsistency between adaptation rules (AR) at the design time because we cannot predict all users' scenarios and concerns that associate with context. In addition, we are required to calculate all of the variances that could never take place at the execution or at the real operation of the system. And we do not know which the viewpoints are being used together in one system because adding new viewpoints depend on the location, activity, the regulation, and the users' choices.

Currently, we can confirm that self-adaptive systems can detect and solve conflicts between adaptation rules at runtime as presented in [Continuum-2012], [Sun-2014], [Trollmann-2015], [Zhang-2017], *etc.* However, the number of adaptation rules increase significantly if the system uses multiple viewpoints at one time. Hence, detecting conflicts between viewpoints at runtime has become complex and consumed a lot of time in the adaptation process. This is the reason why we need one mechanism to detect and solve the conflicts of the semantics of viewpoint on CAM before supporting adaptation rules to self-adaptive system.

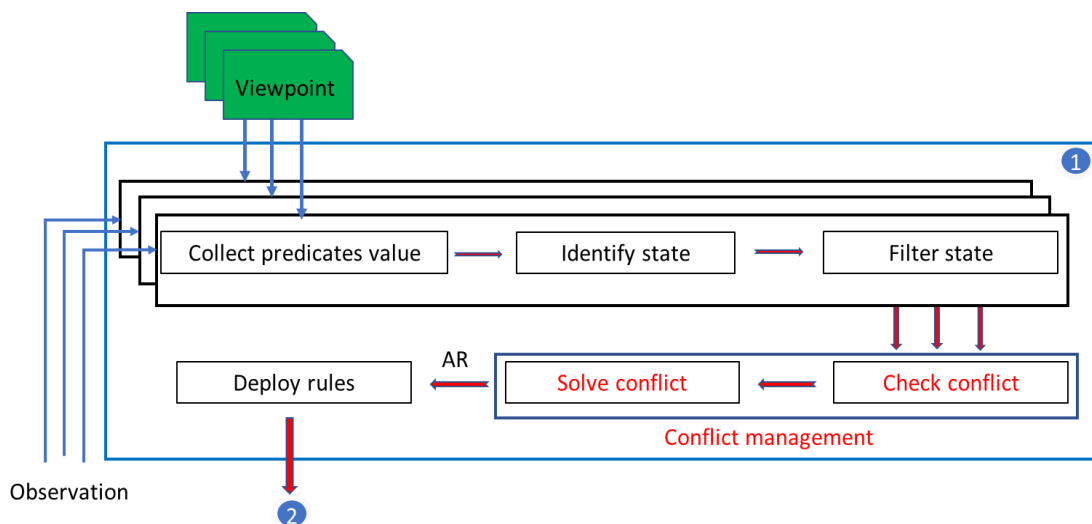


Figure 4.1: The conflict management in Context-aware management (CAM)

To solve that problem, we propose to add a “Conflict management” step into CAM. We separate the Conflict management step into two sub-steps are: “Check conflict” and “Solve conflict” as shown in Figure 4.1. The “Check conflict” step focuses on the identification of conflict that will be described in section 4.4 and the “Solve conflict” step supports the conflict resolution that will be presented in Chapter 5.

Besides, the CAM is independent with the ASS, and we can anticipate actions in CAM to unload the unnecessary rules in the SAS. In addition, the SAS manages the rules without considering the semantics behind them. That means the SAS can only resolve the conflicts of access to devices; it cannot deal with the differences related to the semantics of the viewpoint state. And in CAM, we cannot detect the variations in adaptation rules level because we do not have the information about the instance of components that are impacted by the adaptation rules. And, the conflict in the adaptation is not easy to define, because it occurs in many different aspects of responses. W. Wang et al. described that conflict is a natural disagreement between different attitudes, beliefs, values, or needs [Wang-2011]. We might see that the essence of the conflict between the adaptations seems to be disagreement, contradiction, or incompatibility in the main goal of adaptation. In our approach, each viewpoint at one observation cycle has one state is defined by a vector of predicates, and each state has its objective. The objective of state (OOS) defines implementation steps to attain the identified goals. Unlike goals, OOS is specific, measurable, and has a defined action. It is more specific and outlines the “what, where, and how” of reaching the goal. Thus, CONFLICT refers to any situations in which there are incompatible between the two OOSs. The incompatible between the two OOSs is a situation in which the desired end states or preferred outcomes appeared to be incompatible. From that, we want to detect conflicts between states on CAM base on analyzing and comparing the OOS of all states that existing in CAM at the same time. To use the OOS for detecting the conflict process, we need a standard definition of OOS for all viewpoints.

4.2 State of the art

Many works have been done in self-adaptive system that using 5W&1H questions for eliciting adaptation requirements such as [Kim-2012], [Salehie-2009], *etc.* Salehie and Tahvildari used the 5W&1H questions for emphasizing adaptation information such as: When does an adaptation need to be applied? Why do we have to adapt (identify the goals of the adaptation)? Where do we must implement change? What kind of adaptation is needed? Who has to perform the adaptation? How is the adaptation performed?

Krupitzer et al. proposed a taxonomy that answers the questions for adaptation. They used 5W&1H question as question dimension of their taxonomy. Each question provides different detail information of adaptation: “*When? Time (Reactive vs. Proactive). Why? Reason (Context, Technical Resources, User). Where? Level (Application, System Software, Communication, Technical Resources, Context). What? Technique (Parameter, Structure, Context). Who? N/A (Nature of a self-adaptive system leads to an automatic*

type of adaptation). How? Adaptation Control (Approach, Adaptation Decision Criteria, Degree of Decentralization)” [Krupitzer-2015].

The work of the researchers above has shown that we can use six elements on the 5W1H approach to describe the adaptation information. In our work, we use OOS to outline the object, place, and action of implementation steps in each state to reach the main goal of the states. Therefore, we believe that we might use the 5W1H approach to describe the OOS in our approach.

In other ways, many works have been used 5W&1H or a part of it to model context such as [Rathi-2018], [Rey-2005], [Kim-2012], etc. In these studies, Kim et al. used 5W&1H questions to define models for context information, as shown in Figure 4.2. They proposed an ontology-based context-aware modeling technique to enable efficient specification of context information.

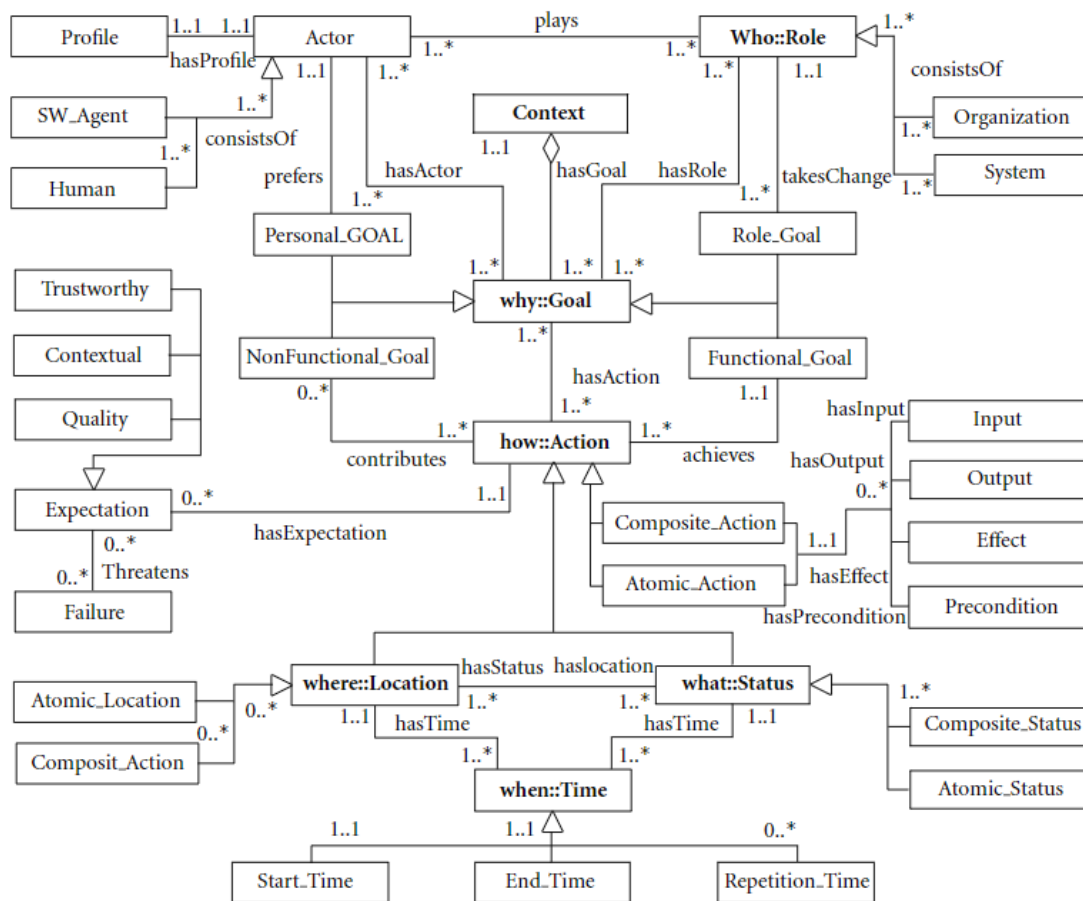


Figure 4.2: Ontological context-aware Action using 5W&1H question [Kim-2012]

This model is named $CA_{5W1H}Onto$ that defined in the unit of $\langle Concept, Instance, Context \rangle$. The $CA_{5W1H}Onto$ model independently separates ontology and context in the form of modules; it provides the high levels of expandability and recyclability of the context model. Moreover, the $CA_{5W1H}Onto$ also provides some formalism as contextual

information via the web ontology language-description logic OWL-DL. This formalism allows modeling a special domain by defining classes, instances, data type properties. In the CA_{5W1H}Onto, the context is defined by six elements: role, goal, action, status, location and time that corresponding to six elements of context modeling ontology (why, who, where, what, how).

Rathi et al. [Rathi-2018] proposed using the ESO-5W1H framework to adjudge the human roles and machines in their respective interactions. The ESO-5W1H framework used the 5W1H classes (Who, When, Where, Why, What, How) to structure the underlying decision process follow four operation stages include: The Problematisation stage, the Interesement stage, the Enrolment stage, and the Mobilisation stage. With “- *“Who” is used to describe Role (Car, Road, etc.).*

- *“What” is used to describe Status (Car in Motion True, Power break is Active true, Pedestrian in Motion False, etc.).*

- *“Where” is used to describe Location (Car at Place 4th Street, Pedestrian at Place 4th, etc.).*

- *“Why” is used to describe Goal (Policy for right of way),*

- *“How” is used to describe Action (Stop Car, Slow Car, Switch to manual, etc.),*

- *“When” is used to describe Causal Chains.”*

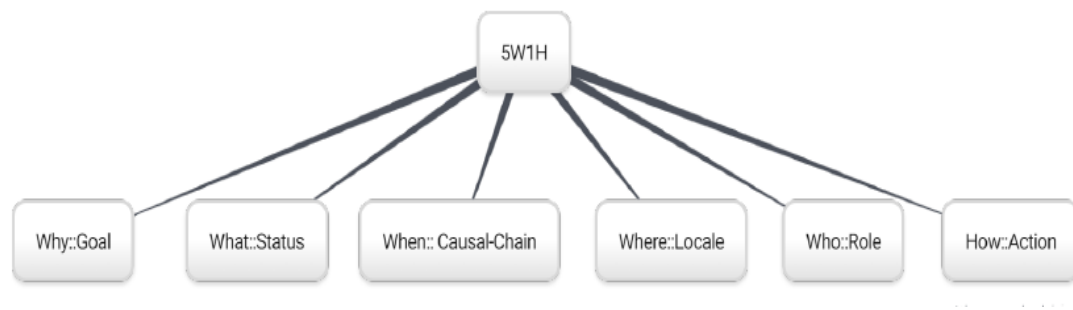


Figure 4.3: The 5W1H top-level classes [Rathi-2018]

From the ontology based on 5W1H question for context modeling such as [Rathi-2018], [Kim-2012], etc., we might see that the 5W1H approach was used to develop domain ontology for software requirement elicitation. The ontology specifies concepts (classes) that extracted from six interrogative elements of 5W1H, which include: Who, When, Where, What, Why and How, and relations between the concepts in software requirement elicitation domain. The 5W1H approach supports the ability to extracting and analyzing concepts and relations within the domain of discourse. It is a motivation to us uses materials from the 5W1H approach to describe OOS. And, we need a standard definition of the objective of state to support the designer in describing it.

4.3 Objectives of state definition

From the studies above, we see that we can use a part of the 5W1H approach to building our Contextual ontology database (COD) that is used as the standard format to describe the objective of state. We used three elements of the 5W1H approach included: “what, where and how” to define OOS. We do not use “When” for OOS because the observation predicates allow the identification of the current state, and the OOS is associated with the current state. We also do not use the element “Why” because it corresponds to the choices of the designer who wrote the viewpoints. The designers explain their choices either in a formal language (usable by the system) or in human language (which could be used to explain to the user). The element “who” is used to indicate the priority of adaptation [Kim-2012]. We use OOS to detect the conflict between viewpoints, so we do not use the element “Who” to describe OOS. However, the element “Who” can be used as an important index to choose adaptation in case of existing conflict between adaptations which will be described in details in chapter 5.

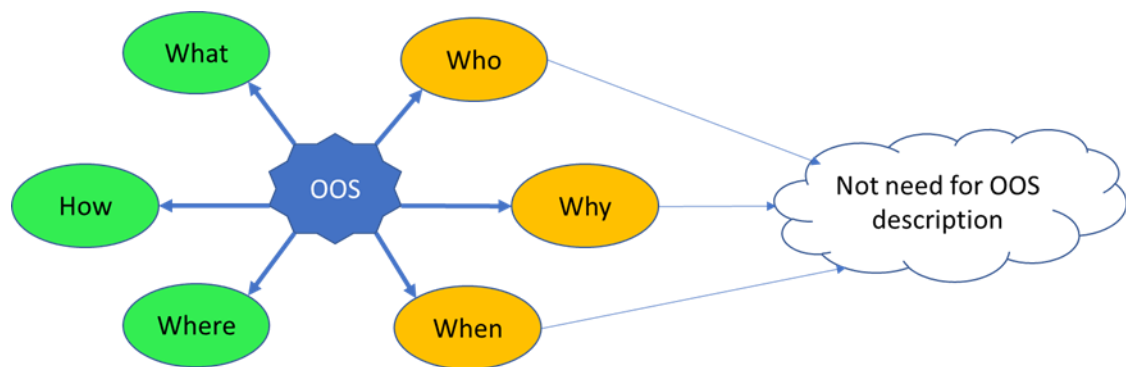


Figure 4.4: The 2W1H description of the objective of state

From the three necessary elements to describe OOS, we also inherited the works of [Yang-2011], [Benslimane-2006], [Bonino-2008] and [Rathi-2018] to build our contextual ontology database using 2W1H (what-where-how) approach as shown in Figure 4.5. The 2W1H domain ontology is modified from 5W1H ontology [Yang-2011], it can help analyzing and extracting the concepts and the relations within domain ontology.

Domain ontology definition: A domain contextual ontology is a 4-tuple $COD = (C, AC, R, A_0)$, where:

- C is a set of contextual concepts. Each concept is considered as a class in the domain ontology.
- AC is a set of contextual concepts features and properties.
- R is a set of relations properties between contextual concepts.

- A_0 is an axiom that is used to define constructs and rules among concepts and attributes in the domain ontology.

Domain concept definition: Domain concept is a union of 3 union sets, and each one is analyzed and extracted from each of three aspects of 2W1H:

$$C = C_{\text{what}} \cup C_{\text{where}} \cup C_{\text{how}}$$

$$\text{where } C_j \cap C_i = \emptyset \ (j \neq i \wedge C_i, C_j \in \{C_{\text{what}}, C_{\text{where}}, C_{\text{how}}\})$$

- C_{what} is the set of concepts from the object that was impacted by the action of service (Light intensive, humidity, temperature, *etc.*)

- C_{where} is the set of concepts from the location of impacted objects (Office, Parking, House, *etc.*)

- C_{how} is the set of concepts from the action of OOS (Increase, decrease, activate, *etc.*)

A domain relation is a subset of related properties of any two concept sets from the three concepts above. In the 2W1H ontology, there are two kinds of relationships, including hierarchical relation and associate relation:

$$R = R_h \cup R_a$$

- Relation R_h : represents the hierarchical relation between concepts (class-subclass).

- Relation R_a : represents the associate relation that indicates semantic relationships between two non-hierarchical relation concepts [Yang-2011].

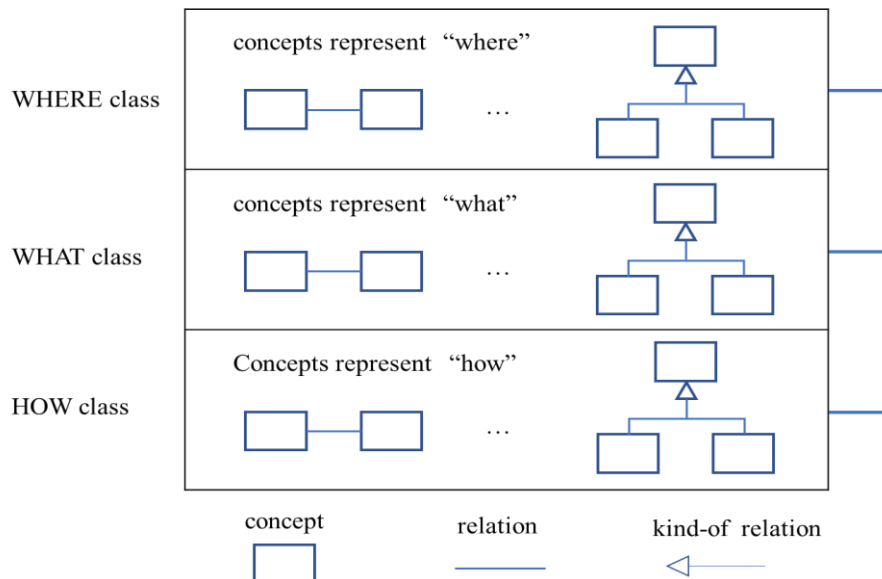


Figure 4.5: The 2W1H contextual ontology based on 5W1H-STPO [Yang-2011]

We used the contextual ontology "where" in the works of [Rey-2005] to apply for class WHERE in 2W1H ontology.

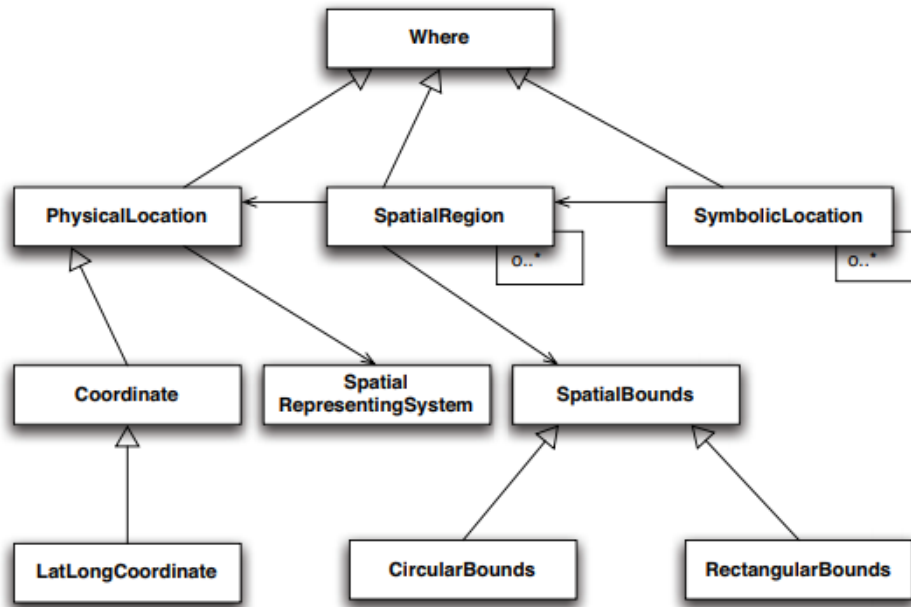


Figure 4.6: Contextual ontology of “where” class [Rey-2005]

And we modified two class “WHAT” and “HOW” on CA_{5W1H}Onto [Kim-2012] to use for 2W1H ontology.

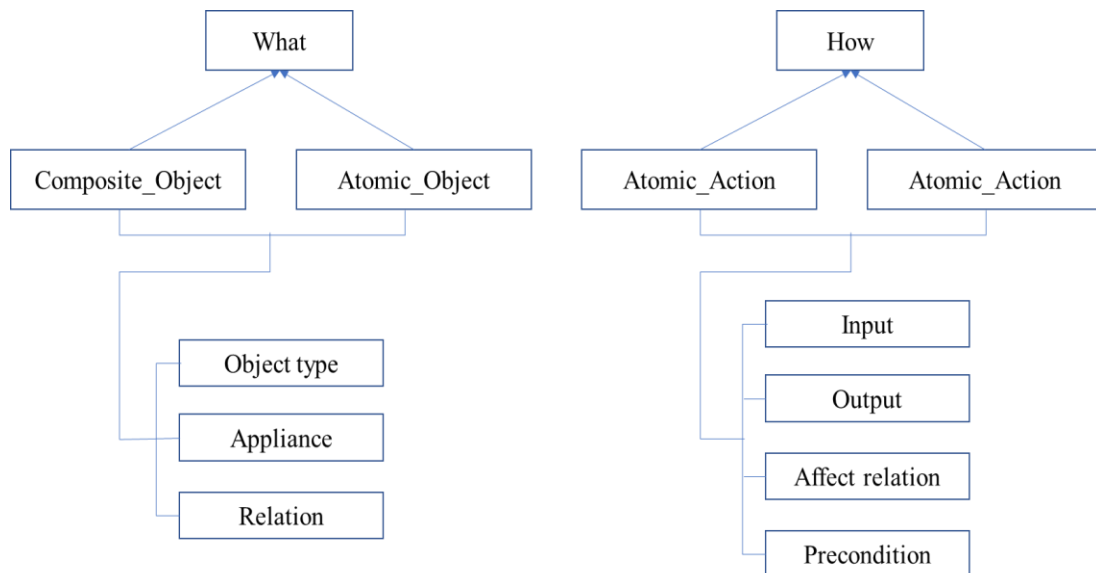


Figure 4.7: The modified of class “what” and “how” of CA_{5W1H}Onto

We use data from 2W1H ontology to define the objective of state following: Let $\{W, P, H\}$ be a set of concepts names of OOS. The OOS elements concept terms O can be formed according to the following syntax: $O[w, p, h]$.

With:

- $w \in W$ with W is a list of objects on class What (i.e., Light, Humidity, etc.)

- $p \in P$ with P is a list of location on class Where (i.e., Office/Department, etc.)
- $h \in H$ with H is a list of actions on class How (i.e., Increase/Decrease, etc.)

The designer can use information from the 2W1H database to describe the objective of state. Each OOS is described by three elements, example: $O_i = \{w, p, h\} = \{Temperature, Office, Increase\}$

And now, we have to update the CIM schema mentioned in chapter 3, as shown in figure 4.8.

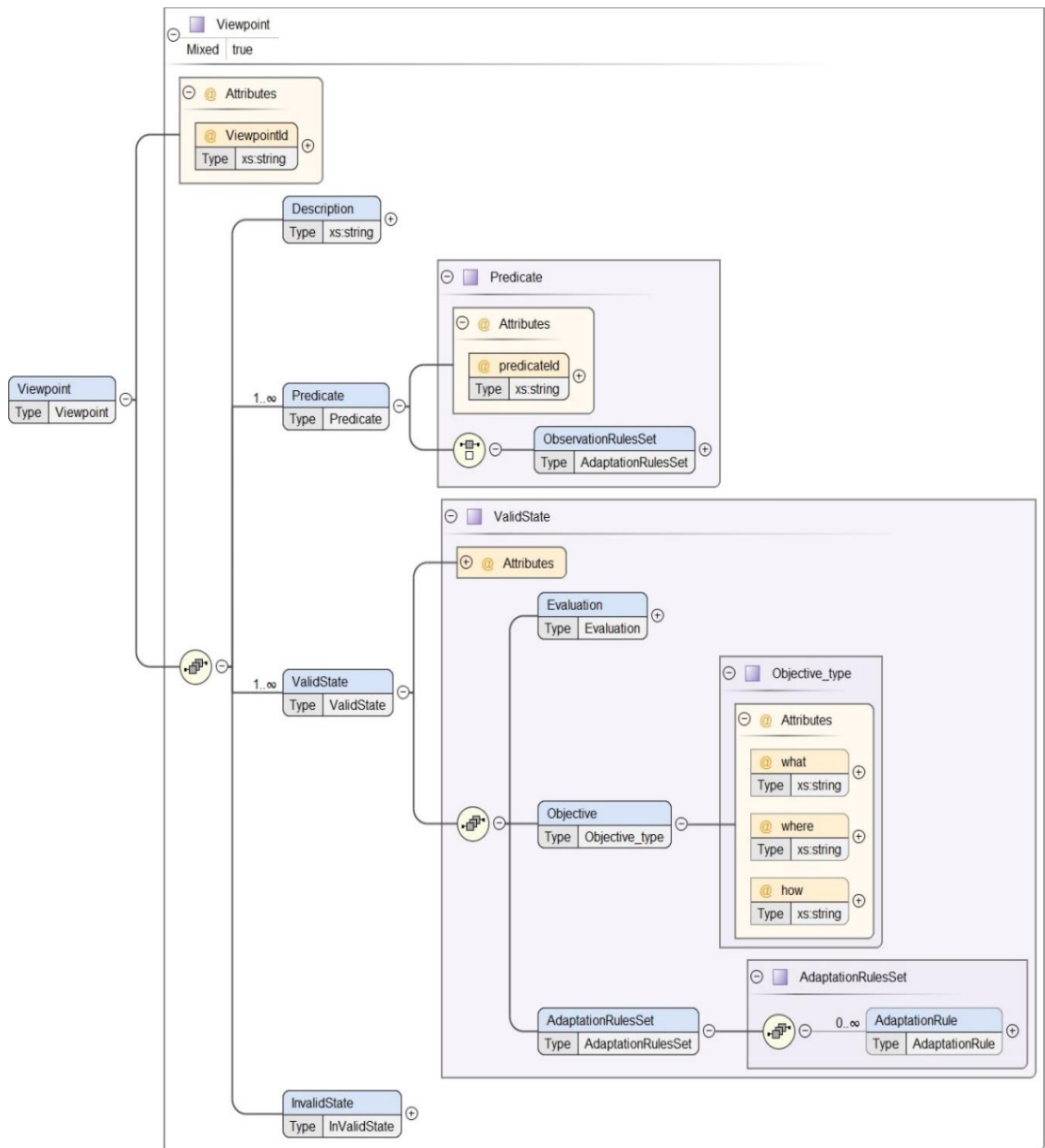


Figure 4.8: Adding the Objectives of state in the schema of CIM

We add the “Objective” element as one new part of the valid states in CIM schema to store information about the objective of state. The data transformed from special viewpoints stores an empty space of objective description and would manually complete by the designers at design time based on using three elements of the 2W1H contextual ontology.

4.4 Detect the conflicts between the objective of states in different viewpoints

As we introduced in section 4.1, the CAM can manage many different states of multiple viewpoints at one time. And we know that at one point, each viewpoint provides one state and each state has one objective. Therefore, it is existed conflict ability between any two of the objectives of states in different viewpoints. Detecting and solving conflict between the objectives of states on CAM are very important to help SAS on improving the adaptation ability of self-adaptive system. In the rest of this chapter, we mainly focus on detecting conflicts between the objectives of states in the different viewpoints. The conflict resolution shall be presented in chapter 5.

4.4.1 State of the art

Detection conflict on self-adaptive system is an important work that decides the responsibility of the system in the pervasive environment. Many works have been done on conflict detection in auto-adapted and context-awareness system such as [Dunlop-2002], [Edwards-1997], [Sun-2014], [Carreira-2014], [Resendes-2014]. They choose to detect and solve the conflicts between adaptations at runtime. We also choose to detect conflict at runtime, but we avoid using adaptation rules with the reason mentioned in section 4.1. Therefore, in this section, we mainly focus on some works that compared the action of adaptations to detect conflicts in self-adaptive system.

Maternaghan and Turner [Maternaghan-2013] proposed a technique for conflict analysis among policies. In this approach, they used the policy system that independent of particular components to apply for a wide variety of devices. The conflict situation depended on whether the actions affect the same device or different devices (as shown in Figure. 4.9). For the same device, they used three types of action to analyze the conflicts between adaptations were the same action (e.g., both ‘turn on’), were opposing action (e.g., ‘turn on’ and ‘turn off’), or were otherwise different action (e.g., ‘turn on’ and ‘dim’). When the system detected the conflicts of a pair of actions on the same devices, the user is allowed to decide what is important and what should be ignored. On the other hand, the user could edit the new or existing policy, chose to ignore classes of conflict, and disabled or deleted the existing policy.

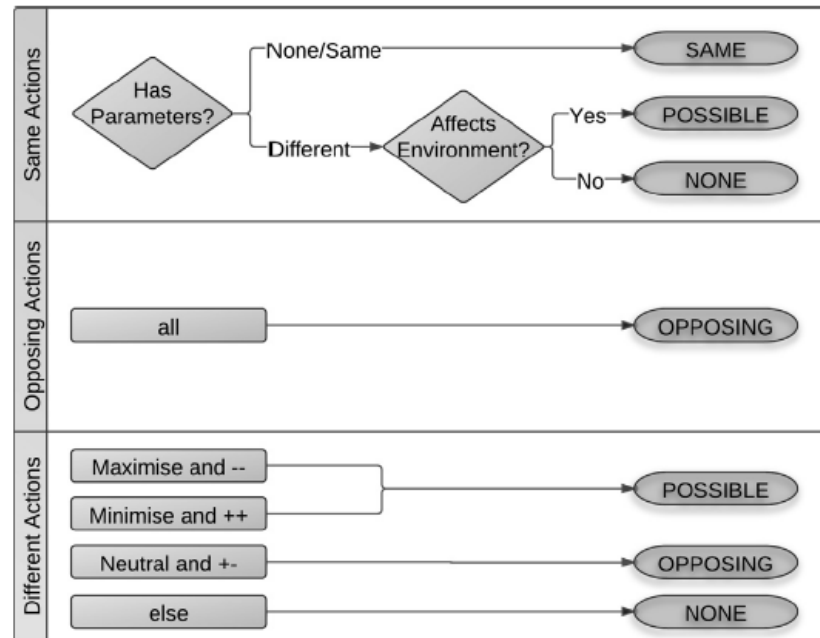


Figure 4.9: Conflict analysis between two actions [Maternaghan-2013]

Trollmann introduced a framework for the runtime adaptation conflict analysis called *models@run.time* [Trollmann-2015]. Trollmann's approach assumes model-driven engineering to detect adaptation conflicts and derived information about reasons for detected adaptation conflicts. In *model@run.time*, the adaptation conflicts are separated into two types: adaptation-adaptation conflicts and adaptation-consistency conflicts. An adaptation-adaptation conflict occurred when different orders of the same adaptations lead to different results. This situation could happen when adaptations overlap in the aspects of the program they adapt. It could change if one adaptation was overwritten by another adaptation. In this situation of adaptation-adaptation conflict, the running software system needs a way to decide in which order to execute these adaptations. Adaptation-consistency conflicts happened when some of the states in the adaptation sequence were not considered a consistent state. In this situation, the running software system has to detect all inconsistent states to preserve or re-establish consistency.

Kephart and Walsh [Kephart-2004] defined the three policies for autonomic systems following: Action, Goal and Utility-function policies. Then Hussein et al. [Hussein-2010] applied these three policies to mechanisms that are selected for adaptations in a self-adaptive system. In Hussein's approach, the Goal-based mechanisms are defined by the goal of the adaptation and are determined by the required adaptation actions to fulfill this goal.

From the related works above, we might see that comparing a pair of actions from adaptations is a solution to detect the conflicts between adaptations. In our approach,

the element “How” in OOS also stores information about the action of a self-adaptive system corresponding with each observation situation. The element “What” and “Where” will give us information about the object impacted by the action of the adaptation process. Therefore, we can use three elements of the OOS to support necessary information for conflict detection process on CAM.

4.4.2 Conflict detection algorithms

As we presented in section 4.3, at one Observation time, if we apply N viewpoint on CAM, the system has N state at that time. That means we could also have N OOS corresponding with N State. And, we need an OOS merging process to detect conflicts between them.

A. Analyzing the conflicts between two OOSs

After the Validation State process (described in chapter 3), we got a list of the OOS as shown in Figure 4.10.

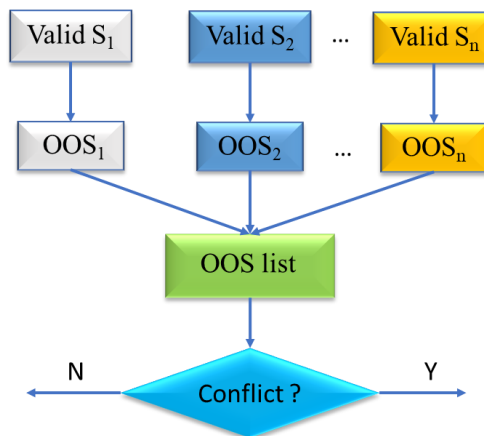


Figure 4.10: Conflict ability checking process

In section 4.1, we mentioned that the CONFLICT refers to any situation in which there are incompatibility between the objectives of states. In this approach, we use three elements “what, where, how” of the objectives of states to analyze conflict situations between them.

In one application, if we use n ($n > 1$) viewpoints to model n different user concerns, we have n states at the moment. That means we could also have n objectives of states corresponds n states at one time. From the Contextual ontology database that mentioned in section 4.1, if we call **W** is a set of all elements belong to the class “What”; **P** is a set of all elements belong to the class “Where”; **H** is a set of all elements belong to the class “How”. And $\{a, b, c\}$ are the three concepts corresponding with OOS

description information (what, where, how) of two objectives of states O_1 and O_2 . We have shown the correlation between elements of these OOS following:

With $O_i(a) = w_i$ ($i=1$ to 2 ; $w_i \in W$), we have 4 correlation situations of $O_1(a)$ and $O_2(a)$ corresponding with for case (a), (b), (c) and (d) in figure 4.11:

- In (a): O_1 and O_2 have same Object (Ex: $O_1(a) = O_2(a) = \text{Temperature}$).
- In (b): O_1 and O_2 have totally different Objects (Ex: $O_1(a) = \text{Temperature}$, $O_2(a) = \text{Light intensive}$).
- In (c): $O_1(a)$ and $O_2(a)$ exist intersection correlation (Ex: $O_1(a) = (10^\circ\text{C to } 30^\circ\text{C})$ and $O_2(a) = (20^\circ\text{C to } 40^\circ\text{C})$).
- In (d): $O_1(a)$ is a subset of $O_2(a)$ or $O_2(a)$ is a subset of $O_1(a)$ (Ex: $O_1(a) = (10^\circ\text{C to } 40^\circ\text{C})$ and $O_2(a) = (20^\circ\text{C to } 30^\circ\text{C})$).

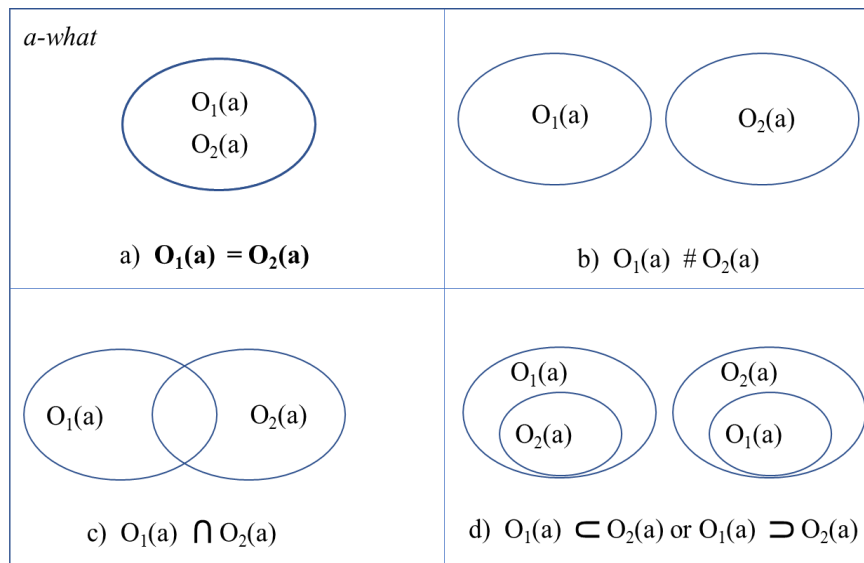


Figure 4.11: The relationship between “what” elements of two OOSs ($O_1(a)$, $O_2(a)$)

With $O_i(b) = p_i$ ($i=1$ to 2 ; $p_i \in P$), we have 4 correspondence situations of $O_1(b)$ and $O_2(b)$ corresponding with for case (e), (f), (g) and (h) in figure 4.12.

- In (e): O_1 and O_2 have same Place (Ex: $O_1(b) = O_2(b) = \text{Office A}$).
- In (f): O_1 and O_2 have totally different Place (Ex: $O_1(b) = \text{Office}$, $O_2(b) = \text{In car}$).
- In (g): $O_1(b)$ and $O_2(b)$ exist intersection relation (Ex: $O_1(b) = \text{First floor of building A}$ and $O_2(b) = \text{Corridor of building A}$).
- In (h): $O_1(b)$ is a subset of $O_2(b)$ or $O_2(b)$ is a subset of $O_1(b)$ (Ex: $O_1(b) = \text{First floor of building A}$ and $O_2(b) = \text{Building A}$).

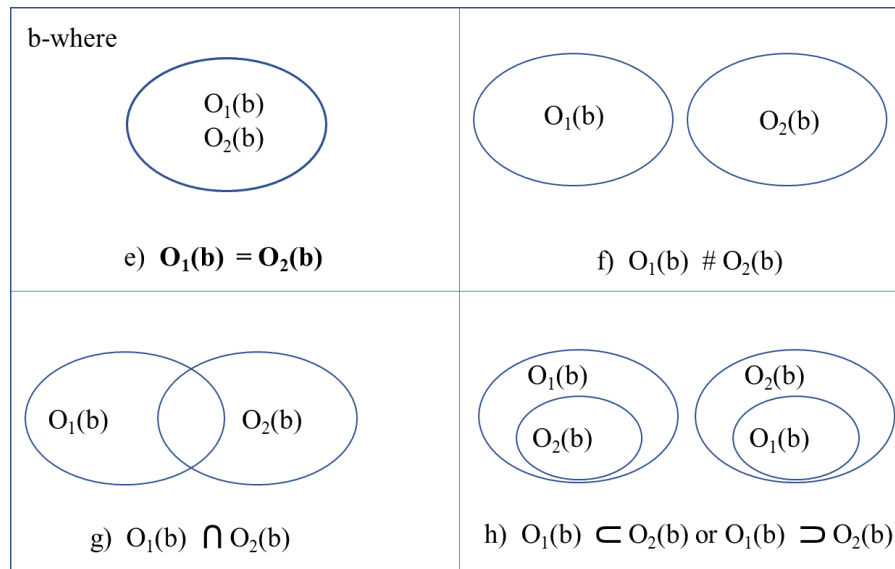


Figure 4.12: The relationship between “where” elements of two OOSs ($O_1(b)$, $O_2(b)$)

- With $O_i(c) = h_i$ ($i=1$ to 2 ; $h_i \in H$), we have four relation situations of $O_1(c)$ and $O_2(c)$ corresponding with for case (i), (j), (k), (l) in figure 4.13.

In (i): O_1 and O_2 have same Action (Ex: $O_1(c) = O_2(c) = \text{Activate}$).

In (j): O_1 and O_2 have totally different Action (Ex: $O_1(c) = \text{Activate}$, $O_2(c) = \text{Turn Off}$).

In (k): $O_1(c)$ and $O_2(c)$ exist intersection relation.

In (l): $O_1(c)$ is a subset of $O_2(c)$ or $O_2(c)$ is a subset of $O_1(c)$.

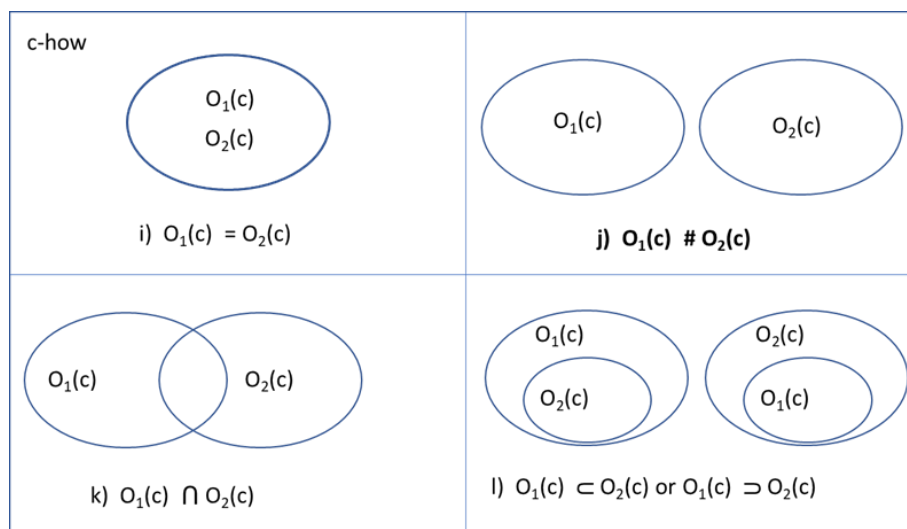


Figure 4.13: The relationship between “how” elements of two OOSs ($O_1(c)$, $O_2(c)$)

If two OOSs have the same action as the case (i), we can confirm that they never have a conflict. On the other hand, if two OOSs have different action as the case (j), (k) and (l),

we need to check the relation properties between these actions to know if they are conflict or not. Based on the Contextual ontology database 2W1H, we add the relational properties of elements “O(c)” in class “How” as shown in Figure 4.14. In “How” class, the relationship between two element x and y exist only two value h and \bar{h} . The relation (R) between x and y: $R(x,y) = h$ when the action x does not affect action y (example: Increase and Activate). $R(x,y) = \bar{h}$ when the action x incompatible or opposite with action y (example: Activate and Deactivate).

The relation properties of all element of “How” class and Subclass database base on the rules:

If X impact on Y: $(R(X,Y)=\bar{h}) \rightarrow R((\forall x \in X), (\forall y \in Y)) = \bar{h}$ and $R(x,Y) = \bar{h}$ and $R(y,X) = \bar{h}$. If X does not affect to Y: $(R(X,Y)=h) \rightarrow R((\forall x \in X), (\forall y \in Y)) = h$ and $R(x,Y) = h$ and $R(y,X) = h$.

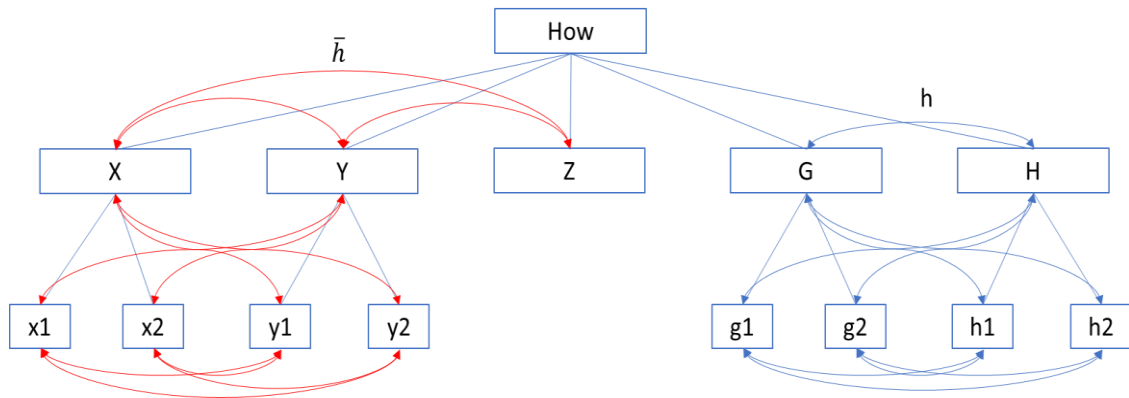


Figure 4.14: Relation properties between “How” elements in 2W1H ontology

Objectives conflict definition: At one time, two OOSs (O_1 and O_2) are called CONFLICT when “action” (how) of O_1 incompatible or opposite with “action” of O_2 and these “action” impact to one “object” (what) in same “place” (where).

Check the conflicts between two OOSs: $O_1(a_1, b_1, c_1)$ and $O_2(a_2, b_2, c_2)$ with $a_i \in W$ (class “what”), $b_i \in P$ (class “where”), $c_i \in H$ (class “how”), $i=1$ to 2. With $R_{1,2}(a, b, c)$ is the relation situation of O_1 and O_2 . We have:

If $O_1(a_1) = O_2(a_2) \rightarrow \text{set } R_{1,2}(a) = w$ (same object)

else $\rightarrow \text{set } R_{1,2}(a) = \bar{w}$ (different object)

If $O_1(b_1) = O_2(b_2) \rightarrow \text{set } R_{1,2}(b) = p$ (same place)

else $\rightarrow \text{set } R_{1,2}(b) = \bar{p}$ (different place)

If $O_1(c) \neq O_2(c)$ (different action) \rightarrow we have two case:

$R_{1,2}(c) = R(O_1(c_1), O_2(c_2)) = \bar{h}$ (incompatible or opposite)

or $R_{1,2}(c) = R(O_1(c_1), O_2(c_2)) = h$ (not affect)

We can get result of the conflict analyzing between two OOSs as below table.

Table 4.1: Objectives conflict analysis results

N_0	$R_{1,2}(a,b,c)$	Conflict analysis "What, Where"	Conflict analysis "Relation of $O_1(c_1)$ and $O_2(c_2)$ "	Obj conflict status
1	wph	Same object, same place	Not affect	Not conflict
2	$wp\bar{h}$	Same object, same place	Incompatible-opposite	Conflict
3	$w\bar{p}h$	Same object, different place		Not Conflict
4	$w\bar{p}\bar{h}$	Same object, different place		Not Conflict
5	$\bar{w}ph$	Different object, same place		Not Conflict
6	$\bar{w}p\bar{h}$	Different object, same place		Not Conflict
7	$\bar{w}\bar{p}h$	Different object, different place		Not Conflict
8	$\bar{w}\bar{p}\bar{h}$	Different object, different place		Not Conflict

Example 1: At the current time (11:30 AM) we have two services required by two users:

User 1: → Service 1: Office: Light intensity not enough → Increase light intensity

User 2: → Service 2: Office: During 11:00 AM to 12:00 AM every day → Decrease light intensity

We have:

Table 4.2: Objectives conflict analysis of O_1 and O_2

	what	where	how
Service 1: O_1	Light	Office	Increase
Service 2: O_2	Light	Office	Decrease
Check	w	p	\bar{h}

$$\text{With: } \left. \begin{array}{l} O_{1(a)} = O_{2(a)} \\ O_{1(b)} = O_{2(b)} \\ R(O_{1(c)}, O_{2(c)}) = \bar{h} \end{array} \right\} \Rightarrow R_{1,2} = wp\bar{h} \Rightarrow O_1 \text{ conflict } O_2$$

In the case of (c)&(d) in figure 4.11 and (g)&(h) in figure 4.12, we cannot be sure about the conflict situation between OOS if we do not have the instance information of the device that impacted by the action of OOS.

For instance, in figure 4.12(g) and (h), if we have many devices (what) in the space of $O_1(b)$ as shown in figure 4.15 (m and n), we might see that the action $O_2(c)$ do not impact all of the devices that existed on $O_1(b)$ because $O_2(b)$ does not cover all space $O_1(b)$. In that case, we need the device instance information to know exactly which device on $O_1(b)$ is impacted by the action of $O_2(c)$. It also similar for the case of (o) and (p) in figure

4.15, we need the device instance information to know exactly if there are conflicts between states are existed or not.

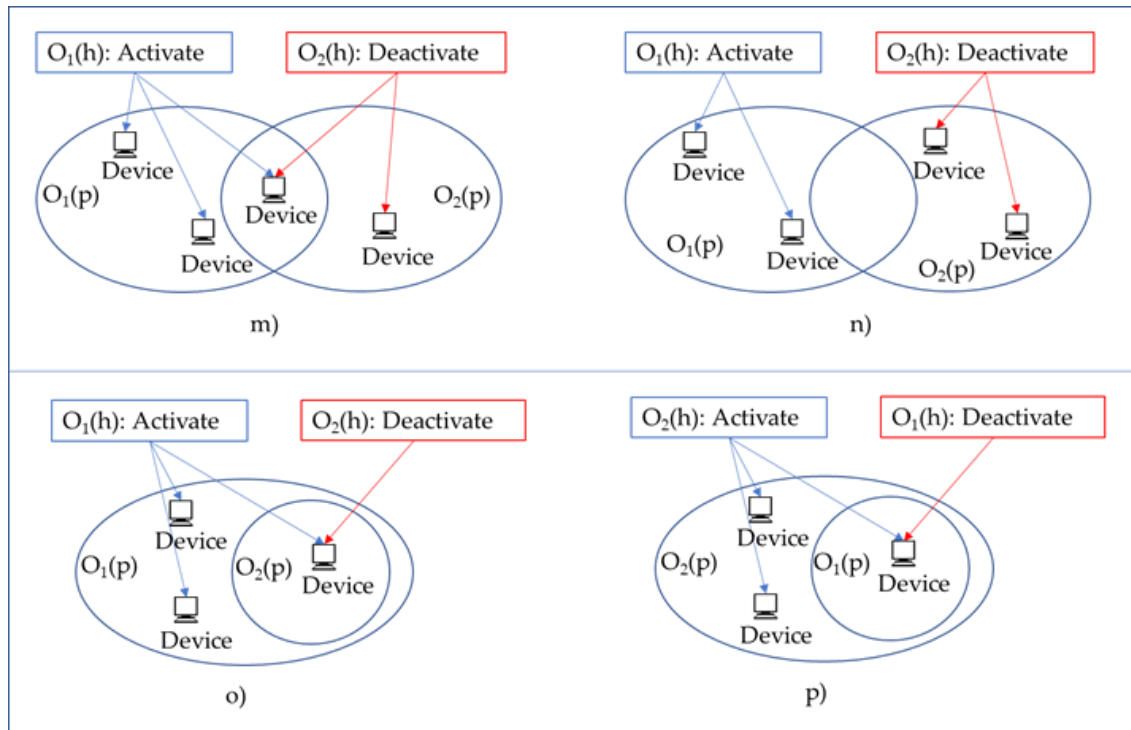


Figure 4.15: The conflict situations when space $O_1(b)$ cover a part or all space $O_2(b)$

In this approach, we solve the conflict in CAM without any information about device instance. Therefore, we can only provide a warning about conflict ability between viewpoints in case (c)-(d) in figure 4.11, (g)-(h) in figure 4.12, and (k)-(l) in figure 4.13 to the user or conflicts will be solved in SAS.

B. Analyzing the conflicts between multiple OOSs

As we mentioned in section 4.4, if we use more than two viewpoints in our application, this means at every moment, the CAM can manage more than two states. And, each state has one OOS, so the system must take more than two OOSs at one time. In this case, the conflict detection process will be compared one by one of all OOSs. That means the number of OOS pairs that needed to check the conflict will be followed by the combination, which is a selection from a collection of all OOSs, with the order of selection does not matter. We have a k -combination of a set S is a subset of k distinct elements of S . If the set S that has n elements, we have the number of k -combinations is equal to the binomial coefficient.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

For instance, given three objects: Temperature, Humidity and Light, there are three combinations of two that can be drawn from this set: Temperature and Humidity, Humidity and Light, or Temperature and Light.

When we have multiple OOSs need to check, we compare one by one for each pair of OOS, so we have $k=2$, the number of OOS pair that need to check is:

$$C_2^n = \binom{n}{2} = \frac{n!}{2!(n-2)!} \quad (1)$$

Example 2: If we have 3 OOS (O_1, O_2, O_3), follow (1) we have $C_2^3 = \frac{3!}{2!(3-2)!} = 3$

they are (O_1 and O_2), (O_1 and O_3), (O_2 and O_3).

However, with multiple OOSs detected at one time, it may exist some OOS are similar to others. To save the time of checking the conflict process, we should use the relation properties between OOS to detect the pair of similar OOS that can help reduce the number of pairs of OOS that is necessary to check the conflict.

B1. Relation properties of two OOSs

As we analyzed in section A, two Objectives O_1 and O_2 called conflict when they have the same object, same place, different action, and the O_1 's action can affect to result of O_2 's action or the O_2 's action can affect to result of O_1 's action.

- With the conflict definition above, we might see that if O_1 conflict O_2 :

$$\Rightarrow O_1(a) = O_2(a) \text{ and } O_1(b) = O_2(b) \text{ and } R(O_1(c), O_2(c)) = \bar{h}$$

$$\Leftrightarrow O_2(a) = O_1(a) \text{ and } O_2(b) = O_1(b) \text{ and } R(O_2(c), O_1(c)) = \bar{h}$$

This means that the correlation between O_1 and O_2 is "**Symmetry relation**".

Therefore, we can conclude that \Rightarrow If O_1 conflict $O_2 \Rightarrow O_2$ conflict O_1

In the case of three or more than three OOSs have the same situations:

$$\text{If we have } O_1 = O_2 \text{ and } O_2 = O_3 \Rightarrow \begin{cases} O_1(a) = O_3(a) \\ O_1(b) = O_3(b) \end{cases} \Rightarrow O_1 = O_3$$

This means that relationship between O_1 and O_3 is "**Transitive relation**".

So, we can conclude that if $O_1 = O_2$ and $O_2 = O_3 \Rightarrow O_1 = O_3$

B2. Conflict analysis between multiple OOSs

Base on the properties of OOS relation mentioned in B1, the system can check the conflicts between multiple OOSs following two steps:

- First, check all OOSs to find union similar OOS. Minimize the number of OOS that need checking by keeping one OOS of union similar OOS. Example: we have three OOS (O_1 , O_2 , O_3) at one time, and two of them are similar (Ex: $O_1 = O_2$). We can choice keep O_1 to check conflict with O_3 . And now, we have only two OOSs need checking (O_1 and O_3). If O_1 conflict O_3 , we can conclude that O_3 also conflict O_2 .

- Second, using the solution in section A to check the conflict one by one with all OOSs appears after step 1.

At the end of the checking the conflict step, we will have the list of OOS where each OOS conflicted with at least one other OOS. Table 4.3 shows a simple example of the result of the conflict detection on one application that has at least 8 viewpoints was applied. The conflict situations between OOS will be presented on a symmetric matrix.

Table 4.3: An example of the conflict situation between eight OOS in one system

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
O_1								
O_2								
O_3	x							
O_4		x						
O_5	x			x				
O_6			x	x	x			
O_7		x						
O_8	x				x		x	

4.5 Chapter conclusion

When we build an application that uses multiple viewpoints to model context, we have to face the problem of the conflicts between adaptations of viewpoints at runtime. Detecting and solving the conflicts in CAM are very important and useful to improve the adaptation ability of self-adaptive system. However, in CAM, we do not have the information at the level of service or device instances, so to detect the conflicts between viewpoints through the adaptation rules are very difficult. In this chapter, we proposed to use the objective of state to describe the main goal of action in each state and used it as a dimension to evaluate the conflict ability between viewpoints. And the first thing we need was a standard definition of OOS for all designers. We used materials and methods from 5W1H approaches to build a contextual ontology database with three classes in term of what, where, and how. The designer will use concepts in the contextual ontology database to describe the objective of each state.

This chapter provided the solutions to detect conflicts between objective of states base on analyzing and comparing three elements of them. This solution is also the answer to the problem related to the research question four. At the end of the conflict detecting process, if the system finds the conflict between OOSs, then it needs a special mechanism to solve the conflicts before supporting a new set of OOS to the next step. The conflict solving process will be described in chapter 5.

CHAPTER 5: Solving conflicts between objective of states

In chapter 4, we have described the solution for detecting the conflicts between the objective of states. In this chapter, we will examine several solutions to solve these conflicts which are dependent on the situation of application and services priority. It is also the answering to the fourth research question.

How to detect and solve the conflicts between adaptations at runtime?

In addition, we introduce the implemented version of objective definition and contextual ontology database that use for solving the conflicts between the OOSs.

This chapter is composed of following: Section 5.1 presents the introduction about using priority of the objective of state (OOS) for solving conflicts between states; section 5.2 presents state of the art on using priorities to solve conflicts between adaptations; 5.3 describes the algorithm to classify the OOS based on the priority level of OOS and our three solutions to solve conflicts; and chapter conclusion in section 5.4.

5.1 Introduction

As mentioned in chapter 4, if there are at least two viewpoints applied in one self-adaptive system (SAS) then at every moment the context-aware management (CAM) can manage at least two states. In addition, each state has one OOS so the system must check the conflict situation between any two OOSs by comparing one by one of all OOSs at every moment. Two OOSs will result in conflict when their three elements have a relationship as shown in figure 5.1.

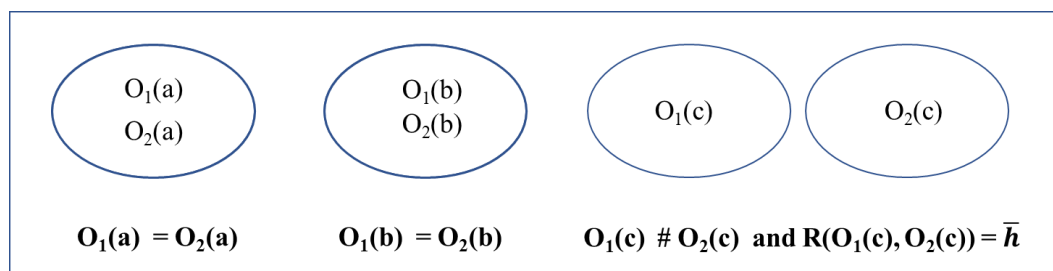


Figure 5.1: The conflict relation of two OOSs

When the OOS of two states has conflict, the action of the first state can make a negative influence on the result of the second state. And the action in each state will be done through adaptation rules that were built by the designer at design time. However, using our approach, the system checks the conflict between the objective of states at CAM and it does not work in rules level [Yagita-2015]. Normally, the system cannot change action in each state; it can only select one best course of action at one point. But the

question here is: *“How to choose the best action that user needs?”* To answer this question, it requires the designer to provide the additional information when they build the viewpoints. One of the commonly used solutions is using the priority level. The CAM can use priority information as an important index to choose more valuable adaptation when existing conflicts between states.

5.2 State of the art

We can find many examples that used “Priority” as an important index to evaluate and to choose services on many domains such as health care, computer science, security, etc. One of the useful initial illustrations of priority level might be the general statements reviewed by Ham [Ham-1997]. Another work of Glaser supports a methodology, named “BMC Remedyforce” that produces an actionable priority based on two measures: impact and urgency of service [Glaser-2015]. An impact and urgency record include three values: High, Medium, and Low. When one assigns an impact and urgency to a task or request, the BMC Remedyforce uses the values to calculate the priority follow the formula: *“Impact” + “Urgency” = “Priority”*. The total number of priority values for the service delivery environment will be calculated following the matrix in Table 5.1

Table 5.1: Sample Priority matrix [Glaser-2015]

Priority				
		Urgency		
		High	Medium	Low
Impact	High	1	2	3
	Medium	2	3	4
	Low	3	4	5

The work of Spicker [Spicker-2009] on priority definition shows that it has five types of priority: Priority as importance; Relative value; Precedence; Priority as special status and Lexical ordering. With the opinion that Priority as importance, Spicker agreed that the most basic meaning of a “priority” is as something which is more important than something else.

Thyagaraju et al. [Thyagaraju-2008] proposed a conflict manager that is built using an array of algorithms that works on the principles of preemption, non-preemption, and it uses methodologies such as Role, Priority, Time slice. This conflict manager uses to resolve conflicts in Context-aware applications. In this approach, they apply the proposed conflict resolution method to Context-aware TV, where the conflicts arise when multiple users try to access an application. Their solving conflict solution based on the combination index of the role and the priority (as shown in Table 5.2) to support the

application or recommendation for the user. The conflict manager sums up preferences of the user, which conflict with each other. And then, the system will provide the recommends specific contents depending on different criterions based on the role of the user and the priority assigned with the user.

Table 5.2: Default priority and role schema

Role	Priority	Role	Role Factor
Father	1	Father	1.45
Mother	2	Mother	1.30
Son	3 & 4 (Based on age)	Son	1.15
Daughter		Daughter	1.15
		Grandfather	1.00
		Grandmother	1.00

We might see many techniques have been used priority level to help the system on providing well-suited services to cater to the versatile needs of the user. In these approaches, application developer or end-user specify conflicts situations, and they used computing middleware to detect and solve the conflict between applications. They used priority methodology as a special index to help the system on choosing applications or services when one of the conflict situations arises. From these approaches, we see that we can use the “Priority level” to describe the important level of each OOS in viewpoints. And it is a good solution to help the system solve conflicts between OOS on CAM at run time.

5.3 Using priority to classify the objective of states

In our approach, we also agree that the most basic meaning of a “Priority” is something that is more important than other things [Spicker-2009]. When the system detects the conflict between two different OOS, it needs to know what to keep for adaptation and what to ignore. In this case, we propose to use the priority element in each OOS to compare the important level of each OOS. The system will then select the best solution for adaptation based on the priority of two OOSs.

In the state-of-the-art section, some researchers use three levels of priority, while others may use four. The number of priority levels depends on the criteria that they may need to consider when they find the conflict between services in their applications. With one self-adaptive system, it must provide adaptations that may relate to security

functional, admin's requirement, user's requirement, applications and devices. Therefore, we inherit the jobs of Spicker, Glaser on using the "Priority level" of service to build our Objective priority with four levels: Emergency, Important, Normal, and Low, as shown in table 5.3. The priority of the OOS will show the important level of the state's goal. From that, the system can choose the best result for the adaptation of the SAS.

Table 5.3: The four levels of priority in the description of the OOS

Priority	Name	Using for
P ₁	Emergency	States related to security functional
P ₂	Important	States related to admin's requirement
P ₃	Normal	States related to user's requirement
P ₄	Low	States related to application and devices

In chapter 4, we have introduced a description of OOS based on the 2W1H approach. With three elements "What-Where-How" on the 2W1H ontology, we can detect the conflicts between OOSs. To solve those conflicts, we need to add the element "Priority" to help the system in evaluating and choosing the best OOS for adaptation. We might see that the element "Priority" indicates the important level of adaptation as the meaning of the element "Who" in the 5W1H approach (mentioned in chapter 4). The CAM can use the "Priority" element of each OOS to evaluate the important level of states that can help CAM can choose what service needs support. It is the reason why we use "Priority" to combine with "What-Where-How" to describe the OOS. That means each OOS will be described by four elements, as shown in figure 5.1.

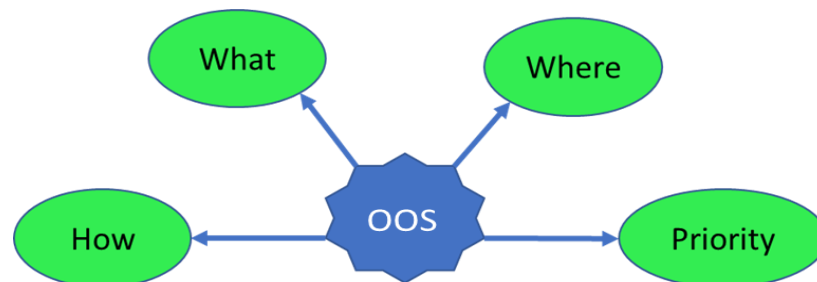


Figure 5.2: The objective of state description with 2W1H&1P

After we have the changing in the objective of state description, we must update the 2W1H contextual ontology database. Figure 5.3 shows a description of the Contextual ontology database with four elements: what-where-how-priority. From the Contextual ontology 2W1H&1P we can describe one objective of state following:

Do "How" to "What" in "Where" with "Priority" → Action – Object – Place – Important level. We have a new structure of OOS = {2W1H&1P} = {What, How, Where, Priority}.

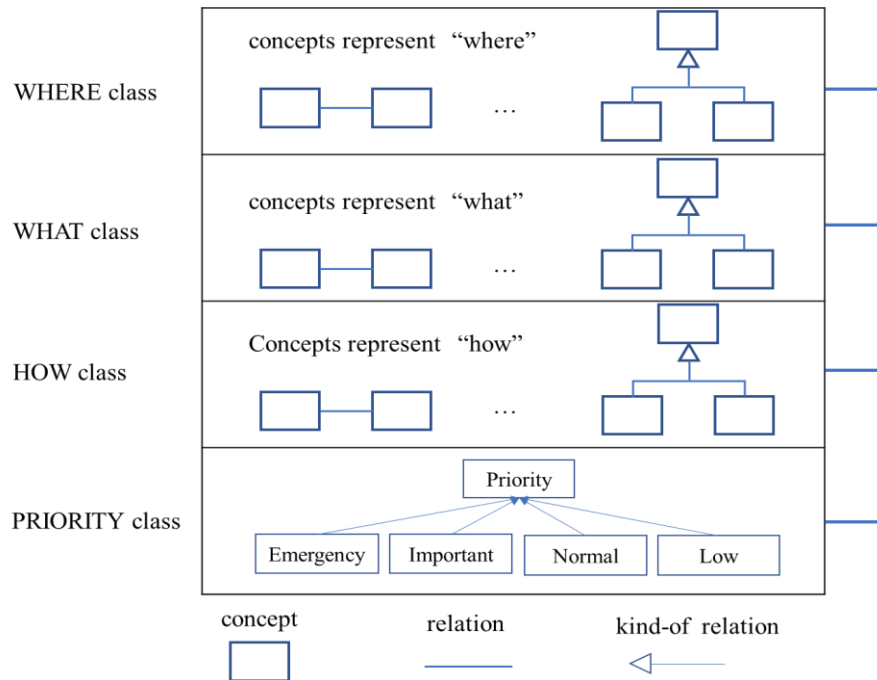


Figure 5.3: The updated version of Contextual ontology database 2W1H&1P

In CAM, the system uses priority information of each OOS to classify OOS into four levels of Priority (P_1 to P_4). If we call $\{a, b, c, d\}$ is four elements corresponding with $\{\text{what, where, how, priority}\}$ of each OOS.

- In time t_0 , with N objective of states which are detected by observation predicate cycle, we can classify them following:

```

int n = N; // N is number of OOS at current time
int j = 0, k = 0, l = 0, m = 0; d = 1;
string[] CP1 = { }; // CPi is an array of OOS (Oi[d] = Pi)
string[] CP2 = { };
string[] CP3 = { };
string[] CP4 = { };
for (int i = 1; i <= n; i++) {
    string nittem;
    string nittem = Oi[d]; // get priority element of Oi
    switch (nittem) {
        case "Emergency": CP1[j] = "0" + i; j = j + 1; break;
        case "Important": CP2[k] = "0" + i; k = k + 1; break;
        case "Normal": CP3[l] = "0" + i; l = l + 1; break;
        case "Low": CP4[m] = "0" + i; m = m + 1; break;
    }
}

```

Example 1: At the current time, we have 11 different OOS, and they have a priority level as shown in Table 5.4. The symbol “x” indicates the priority level of each OOS corresponding with four priority levels including: Emergency, Important, Normal, and Low.

Table 5.4: The priority level of OOS at the current time.

Priority	O _{1[d]}	O _{2[d]}	O _{3[d]}	O _{4[d]}	O _{5[d]}	O _{6[d]}	O _{7[d]}	O _{8[d]}	O _{9[d]}	O _{10[d]}	O _{11[d]}
P ₁	x		x			x				x	
P ₂				x				x			
P ₃					x		x				
P ₄		x							x		x

We use the classification OOS algorithm above to classify the priority level of current objectives; we will have the result of classification OOS process as shown in table 5.5:

Table 5.5: The classifying OOS based on priority level.

Priority class	Objective of states
Emergency (C _{P1})	O ₁ , O ₃ , O ₆ , O ₁₀
Important (C _{P2})	O ₄ , O ₈
Normal (C _{P3})	O ₅ , O ₇
Low (C _{P4})	O ₂ , O ₉ , O ₁₁

5.4 Conflict resolution

After the filter states, we have a set of valid states that need to merge and check conflicts between them before support to adaptation rules step. The conflicts between States at one time also are conflicts between objective of states. Before solving these conflicts, we must classify all OOSs follow the Priority level mentioned in section 5.2. The main goal of this classification step is to support for solving conflict step.

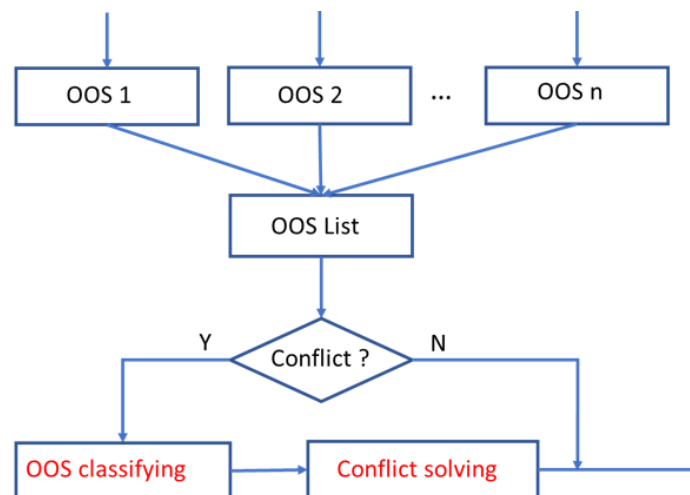


Figure 5.4: Classifying and solving conflict between multiple OOSs

In the next step, we need a solving conflict mechanism to solve the conflicts between viewpoints in the OOS list that we have detected (as presented in chapter 4). As

mentioned in section 5.1, when the system finds a conflict between two OOSs, it must select one of the two. In section 5.2, we suggested to use four Priority levels: Emergency, Important, Normal, and Low to point out the level of important in each state. In the real application of the SAS, some systems use a special viewpoint related to security requirement. It also has the system that uses many viewpoints on the same priority level. Thence, we have three ways to choose the OOS following:

- Choosing the OOS based on Priority level;
- Choosing the OOS based on a maximum number of services can support;
- Combination of both “Priority level” and “A maximum number of services can support” to find the best group of the OOSs without any conflict.

Therefore, we propose three solutions to solve the conflicts between the OOSs corresponding with three aspects to choose the OOS.

5.4.1 Solving conflict between viewpoints based on priority level

5.4.1.1 Description

In this solution, the priority level of state is considered an important aspect to choose the OOS. If there is conflict between the OOSs then the one with higher priority level will be chosen to respond, and the system will ignore the other. If they have the same priority level, the system uses other aspects such as the number of conflicts with other OOSs in lower priority levels or the number of viewpoints that have the same OOS to select the best one for adaptation. This solution can be used in the special application where the priority level is the most important index to choose the state for adaptation such as security applications, production lines, *etc.*

5.4.1.2 The conflicts solving algorithm

Step 1: Merging the same OOSs

Two OOSs (O_i and O_j) are called identical OOSs when their three elements are same ($O_i(a) = O_j(a)$; $O_i(b) = O_j(b)$; $O_i(c) = O_j(c)$). The CAM compares all OOSs at the same priority level (C_{Pi}) to find the same OOSs. If two or more OOSs are exactly the same then one of them is retained, and the CAM saves the number of viewpoints that have same OOS and concatenates this number to the retained OOS.

Suppose we have class C_{Pi} containing N_{Pi} OOS that has the same priority level (as mentioned in example 1), we can find the number of the viewpoints that have same OOS in each class (C_{Pi}) by using the below algorithm:

```

int j = 0;
int [] k = new int[NP1]; // NP1 is number of OOS that has same priority level
int n = 1;
// This code is applied for class CP1, similar for other classes)
for (n = 1; n < NP1; n++) {
    for (int m = n + 1; m <= NP1; m++) {
        if (CP1[n] == CP1[m]) {
            //(On = Om and (On, Om) ∈ CP1)
            k[n] = k[n] + 1; // Save number of viewpoints
            for (int h = m; h < NP1; h++){
                CP1[m] = CP1[m + 1]; // Ignore Om
            }
        }
    }
}
}

```

Example: If we have eleven OOSs (which has conflict situation as example 1) and $O_3=O_{10}$, and N_{P_i} ($i=1$ to 4) is the number of OOS in each class of priority.

We use the above algorithm to check the number of viewpoints that have the same OOS. The result of checking process is presented in table 5.6.

Table 5.6: The result of checking “same OOS” in each priority level

	Decision	After check “same”
C_{P1}	- Ignore O_{10} because $O_3 == O_{10}$ - Save $N_0 2$ with O_3	$O_1, O_3^{(2)}, O_6$
C_{P2}	$O_4 \# O_8$	O_4, O_8
C_{P3}	--	O_5, O_7
C_{P4}	--	O_2, O_9, O_{11}

Step 2: Checking conflicts between the OOSs at each priority level.

The CAM checks the conflicts between OOSs at the same priority level from high to low (C_{P1} to C_{P4}). Then it compares the number of conflicts between each OOS with the others in the same class (C_{P_i}) and ignore the OOS that has most conflict with others (highest number of conflicts). It checks again without ignored OOS and does similar to previous step (ignore the OOS that most conflict with others) until not any conflict detected in each class. If two OOSs in one priority level (class) have the same number of conflicts with others, the CAM must compare the number of viewpoints related to each OOS ($k[n]$). It keeps the OOS that has a higher $k[n]$ (with $k[n]$ is the number of viewpoints that have the same OOS) and ignore the other. In the case, two OOSs have the same priority level and same $k[n]$, the CAM must choose one of them randomly to keep for adapting and ignore the other following below algorithm:

```


int [] h = new int[N]; //h[N] is the number of conflicts, N is number of OOS in CP1
int i, l, j;
//This code is applied for class CP1, similar for other classes
for (int i = 1; i <= NP1; i++) {
    // NP1 is number of OOS that has same priority level in class CP1
    // Check each pair of OOSs in each level of Priority.
    for (l = i+1; l <= NP1; l++) {
        checkconflict(O[i],O[l]); //call function check conflict situation
        if (cfl == "true") { // Oi conflict Ol
            h[l] = h[l] + 1;
        }
    }
    for (j = 1; j <= NP1; j++) {
        checkhighest(h[Oj]); //call function compare number of conflicts
        if (highest[Oj] == "true") { // if h[Oj] is highest
            for (int h = j; h < NP1; h++) {
                CP1[h] = CP1[h + 1]; // Ignore Oj
            }
        }
        if (h[Oj] == h[Og]) { //if existing h[Of] = h[Og] with Of,Og ∈ CPi
            if (k[j] > k[g]) { // k[] is the number of viewpoints have same OOS
                for (int h = g; h < NP1; h++) {
                    CP1[h] = CP1[h + 1]; // Ignore Og
                }
            }
            else {
                if (k[j] < k[g]) {
                    for (int h = j; h < NP1; h++) {
                        CP1[j] = CP1[j + 1]; // Ignore Oj
                    }
                }
            }
        }
    }
}
}

```

In the above example, with the C_{P1} level, suppose $O_1 \gg O_3$ (symbol “ \gg ” be used to REPLACE the conflict situation between two OOSs). The CAM compares the number of viewpoints related to O_1 and O_3 ($k[1]$ and $k[3]$), the OOS with smaller number of viewpoints related to it will be ignored. In this case, O_3 and O_1 have the same number of conflicts with other OOS in priority level C_{P1} . However, O_3 is related two viewpoints, and O_1 is related to one viewpoint. So the CAM keeps O_3 , and ignores O_1 as shown in table 5.7.

Table 5.7: Conflict situation and the result of checking process in step 2

	O_1	O_3	O_6	
O_1				O_1
O_3	X			O_3
O_6				O_6



OOS	Conflict number	Ignore
O_1	1	O_1 ($k[2] < k[3]$)
O_3	1	
O_6	0	

After the CAM has ignored O_1 out of the checking step, it must update the checking table without O_1 as shown in table 5.8.

Table 5.8: The updated table of OOSs in four priority levels


	Objectives	N_{pi}
C_{P1}	$O^{(2)}_3, O_6$	2
C_{P2}	O_4, O_8	2
C_{P3}	O_5, O_7	2
C_{P4}	O_2, O_9, O_{11}	3

In the case of two OOSs have the same number of viewpoints related to them and they are in conflict, the CAM needs to check the conflicts of these two OOSs with other OOSs in the lower level and ignore the OOS has most conflict with others. At the end of the checking process, if there are two OOSs with the same number of the conflict with other OOSs in lower level then the CAM must choose one randomly and ignore the other.

With above example: If we have the conflict situations between OOS following: ($O_4 \gg O_8$) and ($O_4 \gg O_5, O_4 \gg O_7, O_8 \gg O_5$). The CAM continues to check the number of conflicts between OOS in C_{P2} as shown in table 5.9.

Table 5.9: The number of conflicts in class C_{P2}

	O_4	O_8
O_4		
O_8	x	



OOS	Conflict number
O_4	1
O_8	1

Because the number of the conflict of $O_4 = O_8 = 1$, the CAM need to check the conflicts between O_4 and O_8 with OOS in the lower level as shown in table 5.10:

Table 5.10: The conflict situation of O_4 and O_8 with other OOSs in the lower level

	O_4	O_8	O_5	O_7
O_4				
O_8				
O_5	x	x		
O_7	x			

We get the result as shown in table 5.11:

Table 5.11: The result of the conflict checking and solving in class C_{P2}

OOS	Conflict number	Ignore
O_4	2	O_4
O_8	1	

The CAM ignores O_4 because it has more conflict with others than O_8 .

Step 3: Checking conflict with OOSs in lower priority level

If the CAM does not detect any conflict between OOSs in each priority level (class), then the CAM checks the conflicts of OOSs in the higher priority level with each OOS in the lower level:

- * If there is conflict, then the CAM ignores the OOS at the lower priority level.
- * If there is no conflict, then the CAM checks conflict in the next lower priority level.

Continue until no conflict appears; we will get a set of OOS without conflict.

To clarify this step, we continue with the example above, if ($O_3 \succ O_5$), the CAM will ignore O_5 because the priority of O_5 is lower than O_3 . Next, it checks the conflicts of OOS in the high level with other OOSs in the lower level. We get the result as shown in table 5.12:

Table 5.12: The result of the conflict checking and choice OOS for adaptation

	After check conflict with lower level	Reason
C_{P1}	$O_3^{(2)}, O_6$	Not conflict
C_{P2}	O_8	Not conflict
C_{P3}	O_7	O_3 conflict $O_5 \rightarrow$ Ignore O_5
C_{P4}	O_2, O_9, O_{11}	Not conflict

After the CAM ignores O_5 , we get the result: $O^* = O_2 + O_3 + O_6 + O_7 + O_8 + O_9 + O_{11}$

5.4.1.3 Discussion

In the security applications and production line, the task of each state usually associates with the priority level to ensure that the higher priority tasks are performed first. When the CAM use priority level to solve conflicts between viewpoints, it will support adaptations based on the evaluation of priority level. Therefore, this solution is a good choice with security applications and production lines. However, this solution has also its limitations. When the CAM detects the conflicts between two OOSs have the same priority level (ex: C_{P1}) and the same number of conflicts with other OOSs in the lower

level, it must choose the random one of them to respond and ignore the other. This randomly selection can lead to the loss of the best result of adaptation. Because in that case, we do not make sure what is the best choice in them (first or second OOS). After the CAM chooses one of them, it must ignore OOS at the lower level that conflict with the selected one. At this moment, we cannot make sure about the conflict situation of the OOS at the lower level. Hence, choosing a random of CAM can make it miss a good union of OOS in lower level. The checking conflict of OOS at the lower level at this time is very complex because the CAM must check the conflicts between these OOS with some other OOS that may also be ignored in the next step. This work takes more time for the system that can affect to adaptation ability of SAS.

5.4.2 Solving conflict based on getting maximum of services

5.4.2.1 Description

In the second solution, the maximum number of services can be supported that is considered an important aspect to choose the OOS. This solution can be used in the normal application environment as applications for the smart house, mobile devices, *etc.* where self-adaptive system must support the maximum of services without any conflict among services. The number of services is the most important index on choosing OOS in conflict cases. The priority level is used as an additional index to choose the best group of services for adaptation.

5.4.2.2 The conflict solving algorithm

Step1: Detecting the OOS that most conflict with others

The CAM checks the conflict of each OOS with all OOSs detected by the observation predicate cycle at the current time and ignore the OOS has most conflict with others. Update the list of OOS and repeat checking and ignoring process until no conflict is detected. The algorithm to check the number of conflicts of each OOS with others following:

```
int [] Numcfl = new int [N]; //N is total number of OOS at current time
int n, i;
for (n = 1; n <= N; n++){
    for(i = 1; i <= N; i++) {
        checkconflict(O[n],O[i]); //call function check conflict situation
        if(cfl == "true") { // O(n) conflict O(i)
            Numcfl[n] = Numcfl[n] + 1;
        }
    }
}
```

Step 2: Using the priority level to choose the OOS

If there are more than one OOS that have the same highest number of conflicts with the others then the CAM checks the priority level of these OOSs and ignores the OOS has the lowest priority level. Update the OOS list and go back to the first step.

If we have more than one OOS have the same highest number of conflicts with the others and the same lowest priority level, we must choose one of them randomly to ignore. Update the OOS list and go back to the first step.

To clarify this solution, we might see in the below example.

Example 3: We have eight OOS with conflict situations and priority levels as shown in the below table:

Table 5.13: The priority levels and conflict situations of eight OOS.

OOS	Priority level
O ₁	Emergency
O ₂	Important
O ₃	Normal
O ₄	Normal
O ₅	Normal
O ₆	Low
O ₇	Normal
O ₈	Important

	O ₁	O ₂	O ₃	O ₄	O ₅	O ₆	O ₇	O ₈
O ₁								
O ₂								
O ₃	x							
O ₄		x						
O ₅	x			x				
O ₆			x	x	x			
O ₇		x						
O ₈	x				x		x	

Checking to follow the rule of step 1, we have the result as shown in table 5.14.


Table 5.14: The number of the conflicts of each OOS with others.

OOS	Conflict number	Ignore
O ₁	3	
O ₂	2	
O ₃	2	
O ₄	3	
O ₅	4	x
O ₆	3	
O ₇	2	
O ₈	3	

Because O_5 have the highest conflict number with other OOSs (N_0 conflict $O_5 = 4$), so the CAM ignores O_5 , and checks the conflict situation again without O_5 , as shown in table 5.15.

Table 5.15: Conflict situations without O_5 .

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
O_1								
O_2								
O_3	x							
O_4		x						
O_5								
O_6			x	x				
O_7		x						
O_8	x						x	




OOS	Conflict number	Ignore
O_1	2	
O_2	2	
O_3	2	
O_4	2	
O_5		
O_6	2	
O_7	2	
O_8	2	

Because all OOSs have the same number of conflicts with others, so the CAM needs to check the priority, and O_6 has the priority is "Low" \rightarrow the CAM ignores O_6 . After it ignored O_5 and O_6 , the CAM repeats the conflict checking process with the updated table of OOS; we get the result as shown in table 5.16.

Table 5.16: The conflict solving without O_5 and O_6

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
O_1								
O_2								
O_3	x							
O_4		x						
O_5								
O_6								
O_7		x						
O_8	x						x	



OOS	Conflict number	Priority level	Ignore
O_1	2	Emergency	
O_2	2	Important	
O_3	1	Normal	
O_4	1	Normal	
O_5			
O_6			
O_7	2	Normal	x
O_8	2	Important	

We see that O_1, O_2, O_7, O_8 have the same number of the conflict with others ($N = 2$), but only O_7 has lower priority (Normal) with others, so that the CAM ignores O_7 . Now, the CAM updates the table of conflict situation without O_7 and continue with conflict solving step as shown in table 5.17

Table 5.17: The conflict solving without O_7

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
O_1								
O_2								
O_3	x							
O_4		x						
O_5								
O_6								
O_7								
O_8	x							

OOS	Conflict number	Ignore
O_1	2	x
O_2	1	
O_3	1	
O_4	1	
O_5		
O_6		
O_7		
O_8	1	

The OOS O_1 will be ignored because it has the most conflict with others. Continue until not any conflict detected as shown in table 5.18.

Table 5.18: The result of solving conflict between OOSs follow solution two.

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
O_1								
O_2								
O_3								
O_4		x						
O_5								
O_6								
O_7								
O_8								

OOS	Conflict number	Priority level	Ignore
O_1			
O_2	1	Important	
O_3	0	Normal	
O_4	1	Normal	x
O_5			
O_6			
O_7			
O_8	0	Important	

After the CAM ignored O_4 , we get the result of solving conflict process with a new set of OOSs without conflict: $O^* = O_2 + O_3 + O_8$

5.4.2.3 Discussion

This solution can help self-adaptive system has a quick and maximum of services without conflict at one time. However, it has also the same limitation as the first solution in the same case (choosing random OOS in special cases). For example, if the priority level of O_4 in the above example is "Important", we have to choose randomly between O_2 and O_4 to ignore one. And, if we keep O_4 and ignore O_2 , that means we lost O_7 for adaptation because we have ignored O_7 with reason conflict with O_2 in previous step.

Moreover, this solution can apply only in normal applications without security requirements because the CAM can ignore any OOS if it has the highest number of the conflicts with others without taking care of priority of this OOS.

5.4.3 Solving conflict based on the combination of priority and maximum supported services

5.4.3.1 Description

In this solution, the priority level of OOS is the first aspect that is considered in choosing OOS. The maximum number of supported services is the second aspect that is used to optimize the result of the choosing OOS process in an existing conflict case. Different from the first solution, if it has more than one OOS that have the same number of conflicts with the lower level, the system will check bottom-up to find the OOS has most conflict with other OOSs at a higher level. It will be ignored to minimize mistakes in choosing a good union of OOS at the lower level as mentioned in section 5.4.1.3.

5.4.3.2 The conflict solving solution

The CAM checks the conflict situation between the OOSs in highest priority level, it establishes a table of the number of conflicts between the OOSs at this level.

If it exists an OOS that does not conflict with any other OOS at the same level, the CAM keeps this OOS, and it checks the conflict of this OOS with other OOSs in the lower priority level. If there are conflict then the CAM will ignore the OOS at the lower priority level.

If there are more than one OOS has the minimum number of conflicts (#0) then the CAM checks the conflict of them with all OOS in lower priority level. The CAM keeps the OOS have the lowest the number of conflicts with others and ignore all OOSs conflict with it. The CAM updates the conflict checking table again without OOS that ignored.

Repeat the above process with OOS at the highest level until no conflict detected.

If there are two OOSs in the lowest priority level that have the same number of conflicts, the CAM checks bottom-up to find the OOS in lowest level has the most conflict with the other OOSs in the higher level, delete that OOS and update the conflict checking table to continue the process. If the result of checking bottom-up also exists two OOSs that have the same number of conflicts. The CAM must choose random to ignore one of them, update the table and do again. Do similar for the lower level until not any conflict detected, we will have the result with the best union of OOS that can combine in adaptation layer without any conflict. To understand this solution, we might see in the below example.

Example 4: When we have nine OOS with the conflict situations and priority levels as shown in the table 5.19:

Table 5.19: The conflict situations and priority levels of OOSs

	O ₁	O ₂	O ₃	O ₄	O ₅	O ₆	O ₇	O ₈	O ₉
O ₁									
O ₂									
O ₃	x								
O ₄		x							
O ₅	x			x					
O ₆				x					
O ₇		x							
O ₈					x				
O ₉	x				x		x		

C _{P1}	O ₁	O ₂	O ₃	O ₄	O ₅
C _{P2}	O ₆	O ₇			
C _{P3}	O ₈				
C _{P4}	O ₉				

The CAM checks the conflict of OOS in level P₁ we have the number of conflict of O₂ = O₃ = 1 (less). It continue check the conflicts of O₂ and O₃ with the other OOSs in the lower level. Because O₃ not conflict with anyone, the CAM choose keep O₃. The CAM must ignore O₁ because O₁ >> O₃. The result of checking is presented in the below table (with Cf-R is the number of conflicts with others in each priority level, and Cf-C is the number of conflicts with others in lower priority level).

Table 5.20: The conflict number of each OOS with the other OOSs

C _{P1}	Cf-R	Cf-C
O ₁	2	1
O ₂	1	1
O ₃	1	0
O ₄	2	1
O ₅	2	2

We might see that O₃ and O₂ have the same number of conflicts in class C_{P1}, but O₃ does not conflict with other OOSs at the lower level. Therefore, the CAM chooses to keep O₃. However, O₁ >> O₃, so the CAM must ignore O₁.

The CAM updates the OOSs table without O₁ as shown in table 5.21 and the checking process is repeat again.

Table 5.21: The conflict situation without O_1

C_{P1}	Cf-R	Cf-C
O_1		
O_2	1	1
O_3	0	0
O_4	2	1
O_5	1	2

We see that O_2 has the same number of conflicts with O_5 in class C_{P1} and less conflict with other OOSs at the lower level. Therefore, the CAM keeps O_2 , and it must ignore O_4 and O_7 because they conflict with O_2 . And now we do not have any conflict in level C_{P1} :
 \rightarrow In P_1 we keep O_2, O_3, O_5 . However, we must ignore O_8 & O_9 (O_8 & $O_9 > O_5$).

Continue with the lower priority level in the same way until no conflict detected, as shown in table 5.22.

Table 5.22: Checking conflict for lower priority levels

C_{P2}	Cf-R	Cf-C
O_6	0	0

In level C_{P2} , it has not any conflict, so the CAM keeps O_6

In the end of conflict solving process, we have the result: $O^* = O_2 + O_3 + O_5 + O_6$

5.4.3.3 Discussion

Solution 3 is the combination of the solutions 1 and 2 with using priority level as the most important index to choose the OOS and combining with a special algorithm to choose the OOS that less conflict with others. This solution can solve the limitations of solutions 1 and 2. However, this solution takes more time of system for checking and updating the situation of OOSs during the conflict solving process.

5.5 Chapter conclusion

This chapter described the conflict solving process where the input of this process is a list of OOSs, and the output is the other OOS list without any conflict. We proposed three solutions to solve the conflicts between viewpoints corresponding with three application situations. Depend on application situations, the designer can choose a suitable conflict resolution for their SAS. In this approach, we use the "Priority" level of

OOS to classify all OOSs into four levels (C_{P1} to C_{P4}). These four priority levels corresponding with four important levels of application that adapts for security functional, admin's requirement, user's requirement, applications and devices. The priority level of each OOS is considered as an important index to choose the OOS in conflict case. After the solving conflict step, we got the OOS list without any conflict, as shown in figure 5.5. The result of the conflict solving process in this chapter is also the answering for research question four.

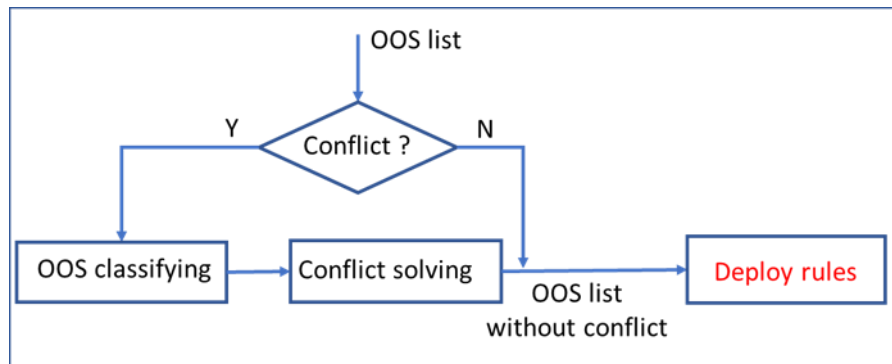


Figure 5.5: The input and output of the conflict solving process

After conflict solving step, the CAM must use the rules deployment step to provide adaptation rules to self-adaptive system. This step will be described in chapter 6.

CHAPTER 6: Adaptation rules deployment and Implementations

In chapters 4 and 5, we have described our framework for detecting and solving the conflicts between viewpoints based on comparing four elements of the objective of states. In this chapter, we focus on describing the rules deployment process, and then we take the implementation to check the operation of the CAM.

This chapter is organized as follows: Section 6.1 introduces the main problems of Adaptation rules deployment; Section 6.2 presents the objective designer database; Section 6.3 exploits our mapping adaptation rules and rules deployment schema; Section 6.4 describes the implementation of the approach, and section 6.5 presents the chapter conclusion.

6.1 Introduction

In chapter 3, we proposed the architecture of self-adaptive system where CAM is used to manage context and independent viewpoints. Each viewpoint identifies a state using vector of predicates, and each state is associated with a goal and rules deployed on the SAS. At runtime, the viewpoints will be added into CAM depend on applications. After the conflicts solving step, it remains the last step of gathering the rules and deploying them on the SAS, as shown in Figure 6.1.

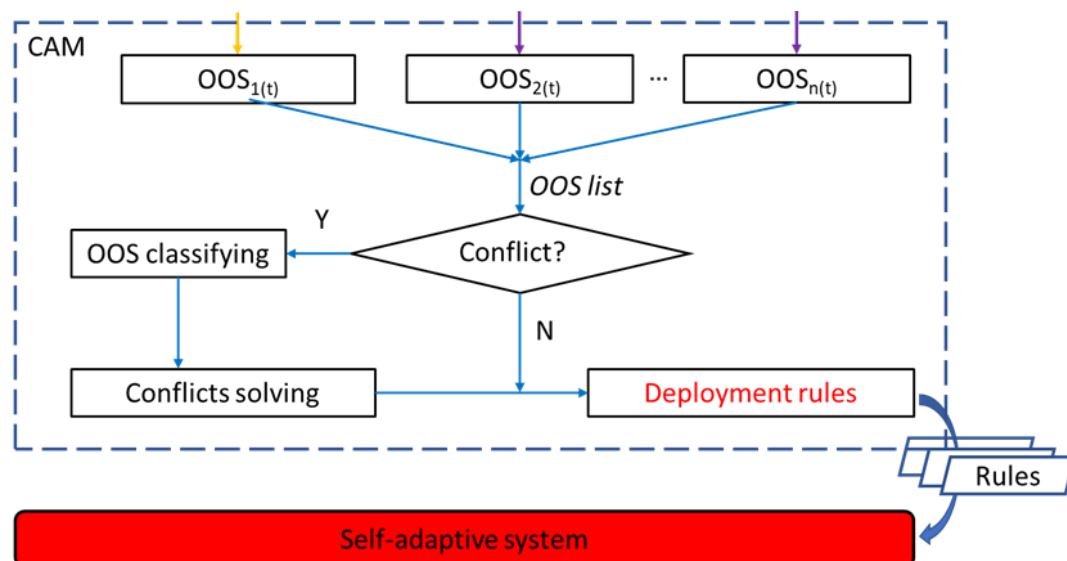


Figure 6.1: The output of context-aware management.

At the end of the deployment rules step, the CAM calculates the difference between the rules deployed previously and the rules to be deployed during the current observation predicate cycle, and only the difference is sent to the SAS. At the implementation level, for performance issues, all rules are deployed as soon as a viewpoint is added, and they are disabled. And in the last step, the CAM does not actually deploy the rules, but it activates or deactivates the rules.

If this solution is viable and works well, it poses a problem in the design. Indeed, if we introduced the independent viewpoint is to facilitate the work of designers and simplify their tasks in writing views. But writing adaptation rules does not match the expertise of our designers. For that, we propose to separate the rules of the viewpoints. Viewpoint states would only be associated with objectives. And on the other hand, we would associate these objectives with the rules. This solution has two main advantages following:

- The first advantage of this solution is simplifying the work of writing viewpoints by designers. It makes the viewpoints independent of the SAS. Since the code of the rules executed in the SAS is no longer in the viewpoints, it is possible (for the CAM) to retrieve the rules associated with an OOS, not only according to the OOS but also according to the SAS. (i.e., depending on the format or language of the rules supported by the SAS).
- The second advantage of this solution is to facilitate the reuse of viewpoints since they will no longer depend on SAS, and it will also be possible to update the rules without the need to modify viewpoints (and vice versa).

However, if we separate the adaptation rules and viewpoint at design time, we must answer two questions that arise then to implement this solution are:

- How to associate adaptation rules to each OOS?
- When is the best moment to associate the adaptation rules to the objective of the states in the CAM?

These questions will be answered in section 6.3. With some requirements for modifications of the CIM, as we discuss above, we must update the CIM for the final version. In the final version of the CIM, we updated the description of the Valid states without adaptation rules; it stores only two sections: "Evaluation" and "Objective". The Objective stores four elements correspond to four elements collected from contextual ontology included "*what; where; how; priority*". The designer must use the information from contextual ontology to fill the description of the OOS in the XML file, which is converted from special viewpoints. This work must be done before adding a viewpoint into CAM.

We get the final CIM schema as shown in figure 6.2

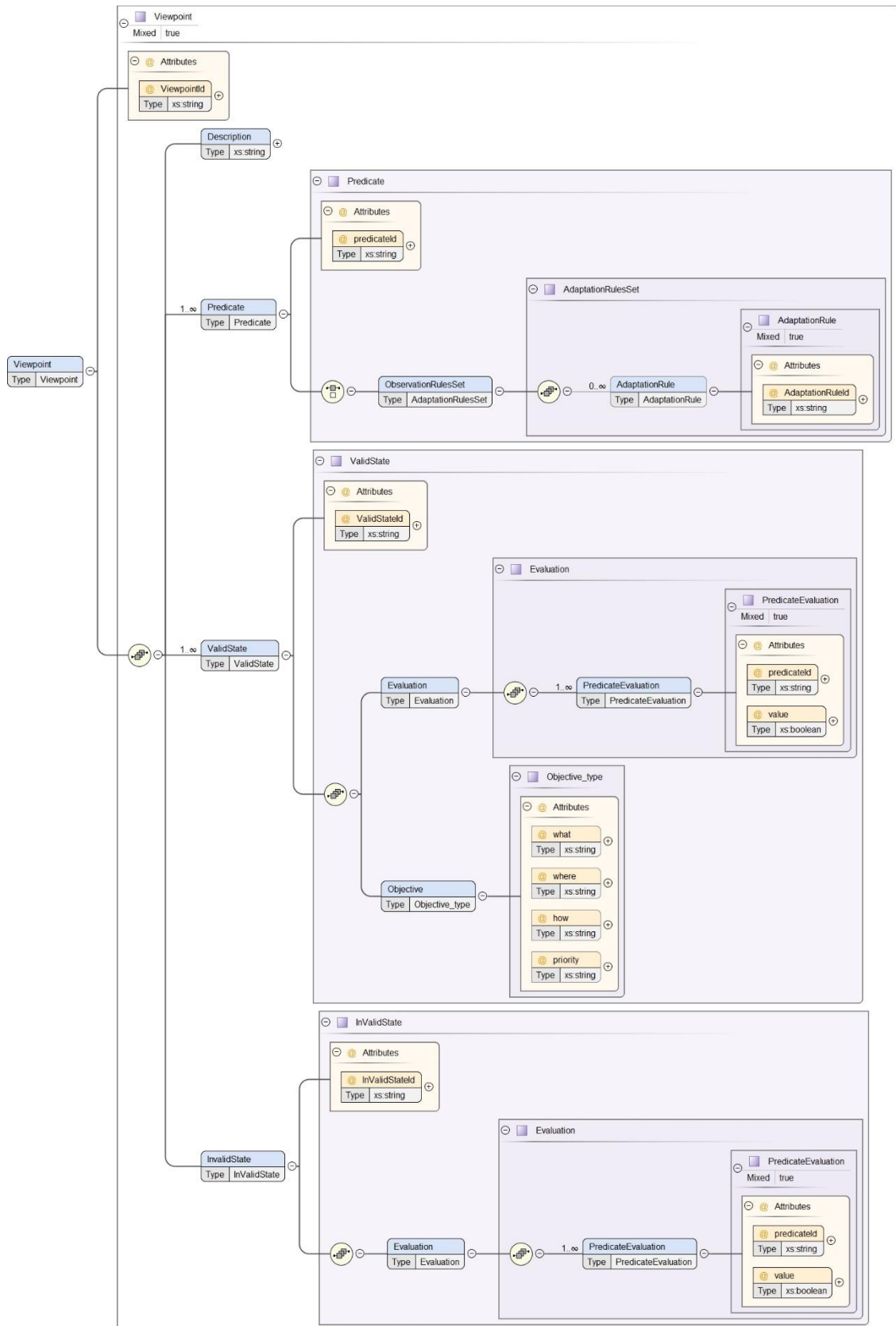


Figure 6.2: The final version of CIM schema

6.2. Objective designer database

Today we might see that getting data from a relational database through a query to the database becomes common. The separation of the relational database with the runtime system can simplify the designing system and building a relational database. Many in the science of data integration use ontologies as ways to unify the description and retrieval of data in relational databases as [Astrova-2007], [Astrova-2009], [Shen-2006], [An-2005]. The using ontologies for data integration can provide the ability to recognize inconsistency and redundancy of data in relational databases [Li-2005]. Ontology also provides a shared and reusable piece of knowledge about a specific domain and mapping of data in the relational database.

In our approach, we used an objective designer database (as shown in Figure 6.2) to support information for the adaptation rules mapping process.

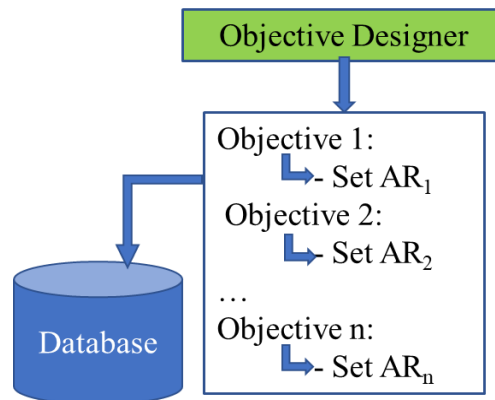


Figure 6.3: Objective designer database

The objective designer database store multiple sets of adaptation rules corresponding to multiple objectives. It can build by independent designers at design time. The list of objectives in the objective designer database will follow the format of objective description in the contextual ontology mentioned in chapter four. The designer will build the set of adaptation rules for each OOS that suits their application environment.

This objective designer database is an open database. Each designer can build the set of adaptation rules for their OOS base on the format of OOS on Contextual ontology. Each objective will store one set of adaptation rules that used to take the main goal of the state through the SAS.

6.3 Adaptation rules deployment

Today almost every industry in the world possesses databases. Using an external database can solve many problems in computer science, such as the size of data, ease

of updating data, accuracy, security, redundancy, incomplete data. In this approach, we propose to use database web service as a tool to get AR for OOS from the objective designer database. The database web services technology is an access data solution that provides access to access data and Metadata through the web services interface. We also use a relational database management system (RDBMS) (as shown in figure 6.3) that allows the designer to create, update, and administer a relational database. The relational database management systems use the SQL language to access the database.

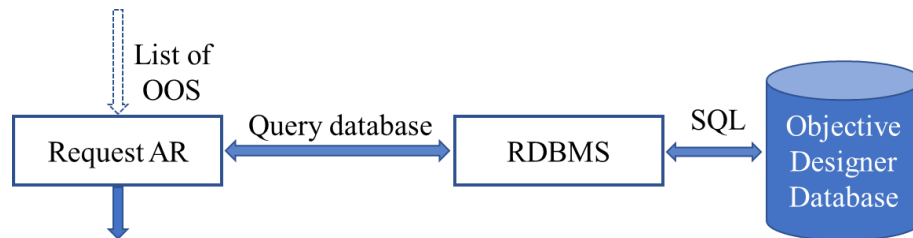


Figure 6.4: Requesting AR for OOS in CAM

We can use the database web service to access database sources like web service and add AR for each OOS based on information from the relational database. But, when we add AR for OOS on CAM is an important question because it relates to problems such as time consumption, optimal error handling in the system, *etc.* We might see more detail of that problem on three solutions for rules deployment process below:

6.3.1 First solution

In each cycle of the observation predicate in SAS (mentioned in chapter 3), self-adaptive system will detect the situation changing of the predicates in all viewpoints that integrated on the CAM. And then, the CAM will continue with “Identify state” and “Filter State” step to provide a list of objectives of all states at the moment (mentioned in chapter 4). All these objectives will be checked to detect and solve conflict (mentioned in chapter 4, 5).

After the system has the list of objectives without conflict, it sends a request to the objective designer database to get the set of AR for each OOS, as shown in Figure 6.4. And then, with all AR provided from the objective designer database, the system will make a “Compute order for SAS” process to detect the difference between the set of adaptation rules associate at the current cycle of adaptation and the set of adaptation rules deployed during the previous cycle of adaptation.

- If an AR, present in the set of adaptation rules of the current adaptation cycle, was already present in the set of adaptation rules of the previous cycle, there is nothing to do.

- If an AR is present in the set of adaptation rules of the current adaptation cycle, was not present in the previous cycle, it must be activated.
- If an AR in the previous cycle is no longer present in the current cycle, it must be disabled.

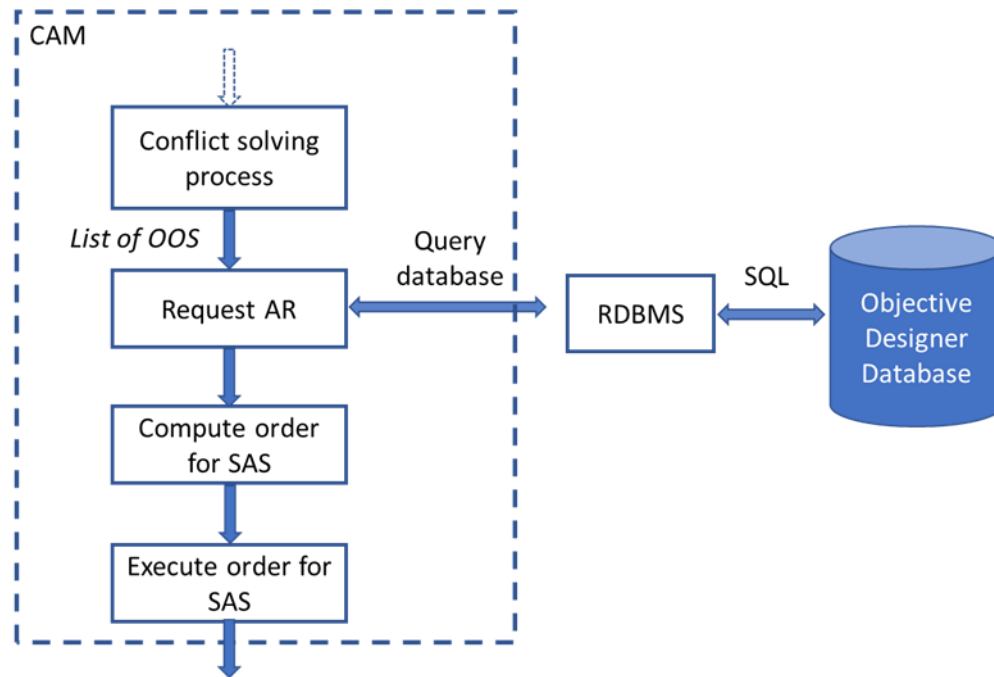


Figure 6.5: Mapping OOS to AR by using objective designer database

The “Execute order for SAS” will enable and disable the rules according to the computed orders in the previous step.

If this solution is very simple to implement, it is relatively inefficient and poses certain problems. The main one of them is a performance problem. On the one hand, we are interested in a database potentially distant, and we are asking for information that has already been asked previously. It is indeed not necessary to ask again the rules associated with an objective if one already made this request not long ago. It would be better if we have the mechanism to detect the new OOS on each time of the observation cycle and query only AR for the new OOS.

6.3.2 Second solution

To solve the limitations of the first solution on request AR for all OOSs of the new viewpoint, we used a checking algorithm to detect changing of OOS list compare with the OOS list in the previous observation cycle. If the system does not find any changing in the list of OOS compare with the previous observation, it will use data in a local database that stores AR of the previous state for the next step. In case the system

detects existing of new OOS in the OOS list in current time, the system will send only request AR for new OOS and then update information in a local database, as shown in figure 6.5.

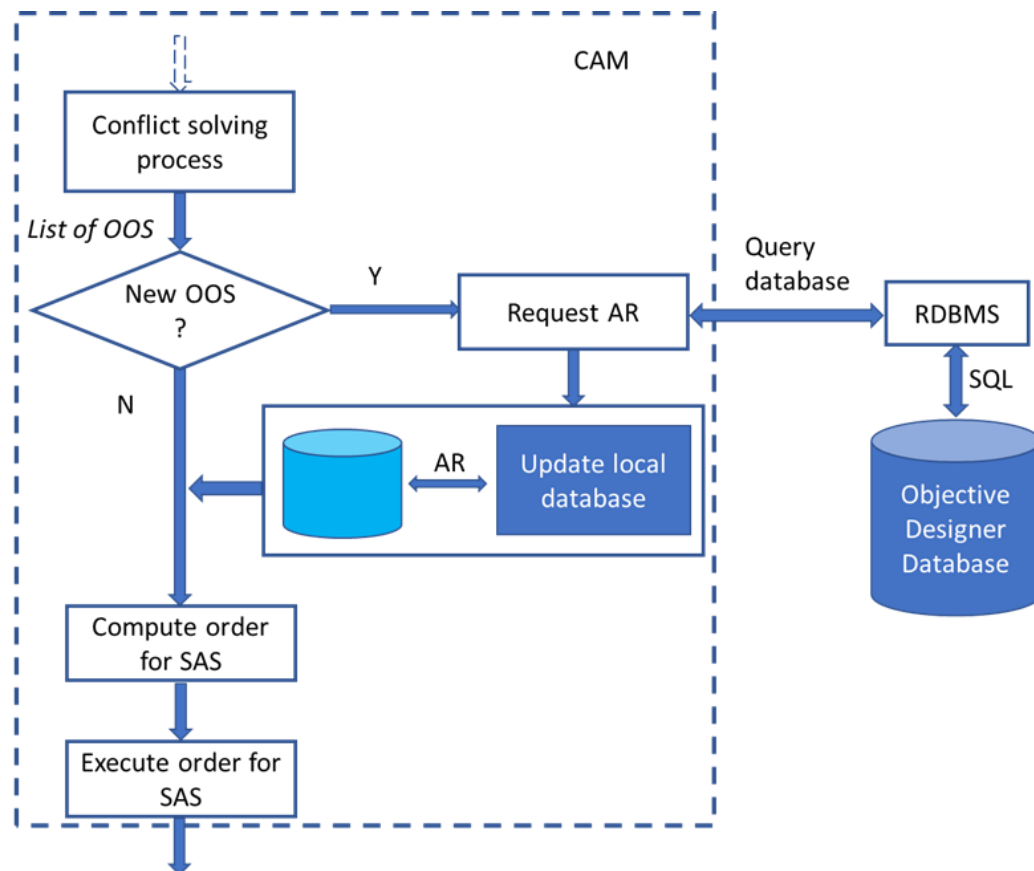


Figure 6.6: Using the local database to save time in compute order for SAS

After the local database is updated with new AR, the rules deployment process will continue with “Compute order for SAS” and “Execute order for SAS” process (as mentioned in section 6.3.1) to support adaptation rules for SAS.

However, in case, the CAM request AR for one objective that is not present on objective designer database, the result for request AR is null, and the CAM will be in an error situation or inaccurate adaptation with the state of the observation process.

6.3.3 Third solution

To solve the limitations in the first and second solutions, we propose thirist solution with a requirement about adding AR for each OOS immediately when a new viewpoint is added into CAM. When any new viewpoint is added into CAM by the user, administrator, or an external process, the CAM will automatically send the request AR to the objective designer database as shown in figure 6.7.

If the CAM detects any OOS that is not present on the objective designer database, it will inform the user and disable that new viewpoint. With this solution, we might see that the problem mentioned in the second solution will be solved. The warning of CAM to the user can help them check or add the necessary OOS for their viewpoints in the objective designer database.

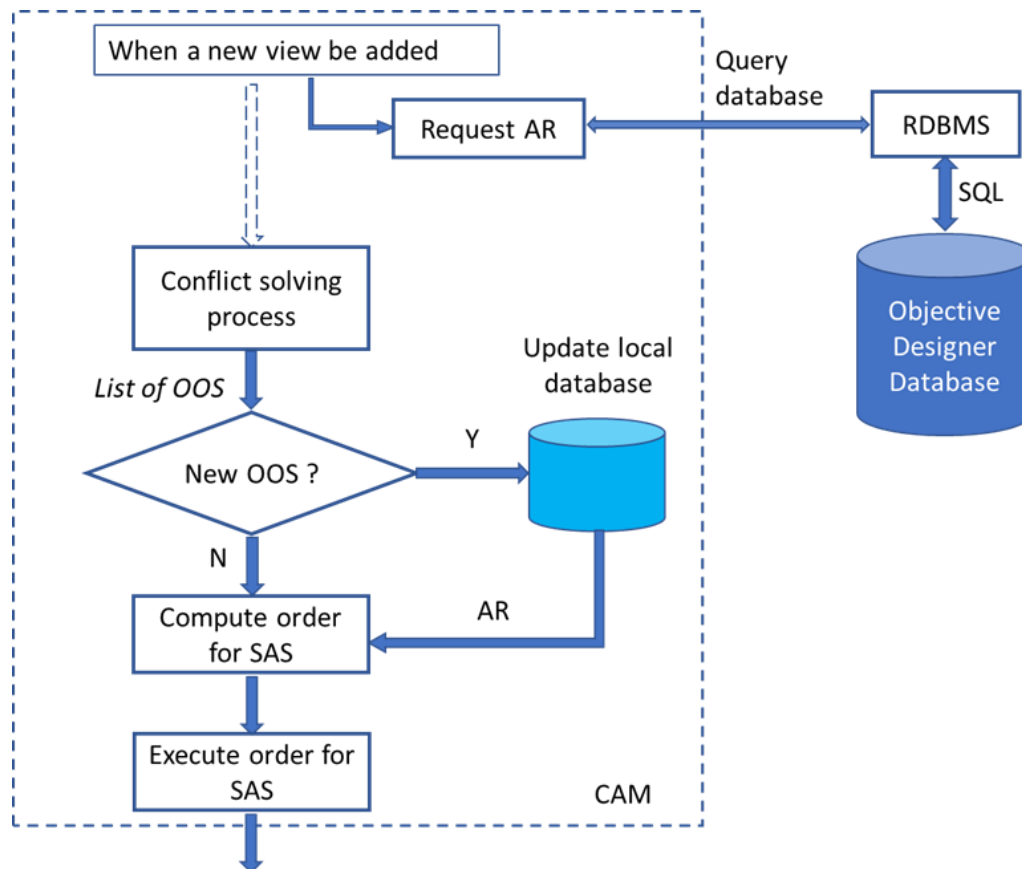


Figure 6.7: The CAM associate AR for all OOSs automatically

After the conflict solving process, we have the OOS list included AR in each OOS. The process will continue with detecting the “New OOS” step; if the list of OOS at the moment different from the OOS list at the previous cycle of observation, the local database will be updated with the new OOS. And, the rules deployment process will continue with two steps “Compute order for SAS” and “Execute order for SAS”, as presented in the previous section.

6.4 Implementations

In this thesis, several tools have been developed to validate the proposals described in the previous chapters experimentally.

6.4.1 Tools for designer

In the first step, we use C# to develop the specific tool to help the designer on converting a specific viewpoint (which is described by BPMN and CTT models) into our context model (CIM). We developed a simple interface for each special viewpoint, as shown in figure 6.8. The structure of special viewpoints is different, so the algorithm to convert data from a special viewpoint into the CIM standard of each viewpoint is different from others. The users only need to select the file of special viewpoints from browser function on interface. They can select “Read file” to check the file and select the “Convert file” function to transform the file of special viewpoint into standard data of CIM.

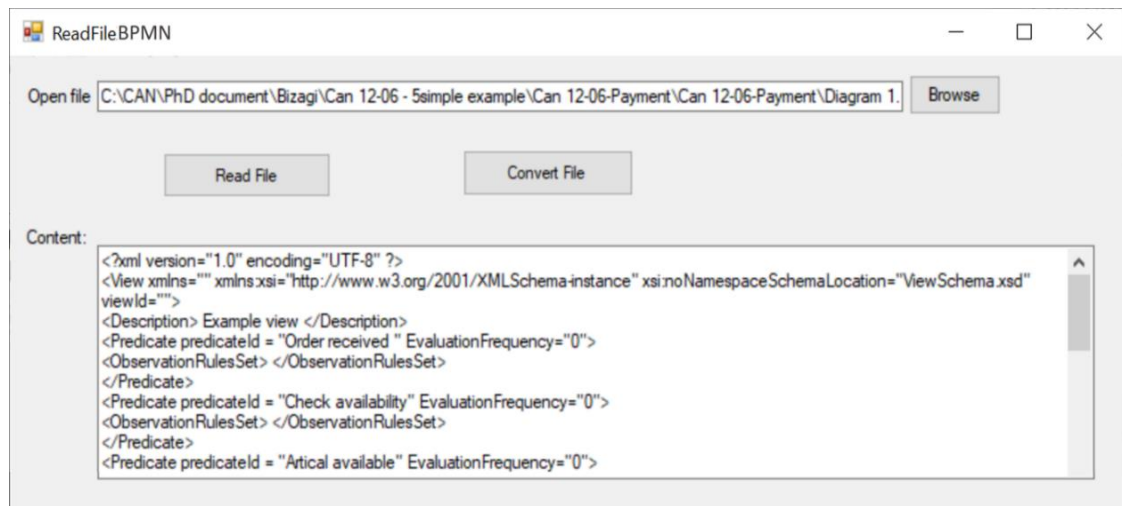


Figure 6.8: The specific tool to convert BPMN viewpoint into CIM format

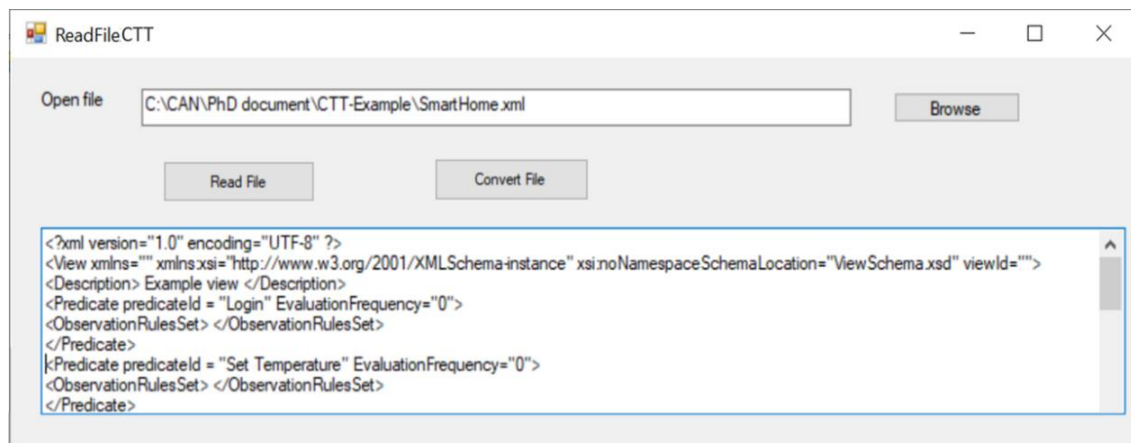


Figure 6.9: The specific tool to convert CTT viewpoint into CIM format

Another tool is available to the designer; It is a simple XML validator that verifies the file produced to describe the viewpoint is in XML format, and it conforms to our context model. Note, it is possible to use other validating and conforming tools that handle XML schemas.

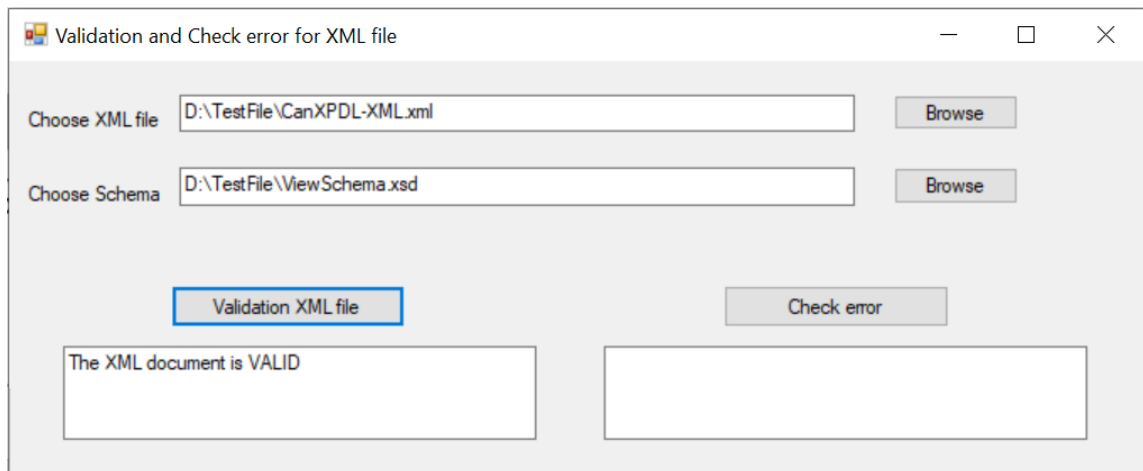


Figure 6.10: The XML validator tool

6.4.2 Context-aware management (CAM)

Our implementation of the CAM has been done in C # and is based on the Core.Net platform (version 2.1 or higher). The structure of the CAM is detail described in figure 6.11.

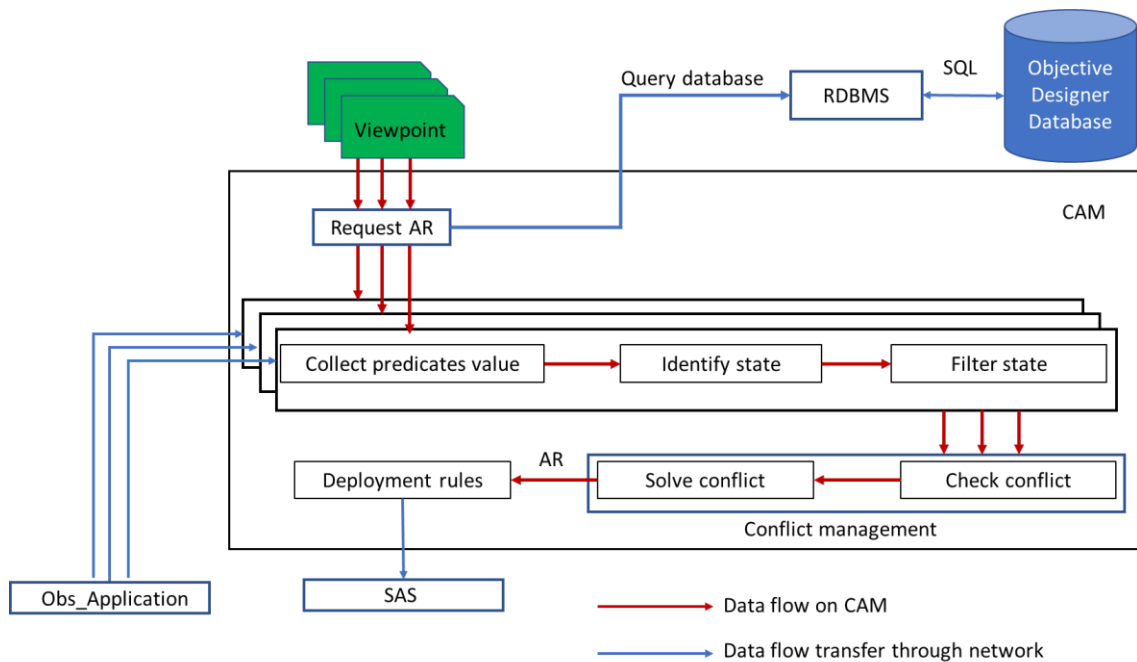


Figure 6.11: The architecture of CAM for implementation

The current implementation of the CAM is a command-line version whose life cycle is shown in Figure 6.12. This one is modular that it is possible to choose the instance of each element that composes it to evaluate different algorithms such as the resolution of conflicts for example.

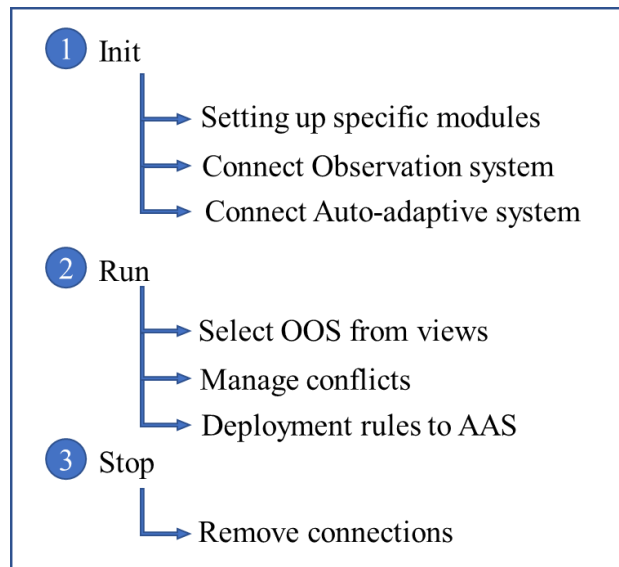


Figure 6.12: The CAM life cycle

When it is initialized, the CAM starts by setting up the different modules for managing conflicts and deploying rules. Then it connects to the SAS and observation system.

Then it shows three streams of activities:

CAM Management: This is an administration flow for users to control the CAM. It allows to add or delete viewpoints, but also to control all the settings of the CAM via the command line or an external GUI.

Viewpoint Management: This stream supports viewpoints from their addition via the administration part until deleted. The life cycle of a viewpoint is detailed later.

Mainstream: This is the part that retrieves the objectives selected by the viewpoints and ensures the deployment of the rules to the SAS. Of course, it is also here that conflicts are managed. This stream manages the adaptation cycle.

The heart of the CAM is the part related to the identification of the context and thus to the management of the viewpoints. We recall that the viewpoints are independent of each other and are therefore managed as such in our implementation of the CAM. So, look now at the life cycle of a viewpoint in figure 6.13.

The initialization of a viewpoint starts when it is added to the CAM. The first step is to check the validity and conformity of the viewpoint with our context model. Then, if everything is normal, the rules associated with the set of objectives present in the viewpoint are retrieved as described in section 6.3.3. And these are immediately deployed on the SAS, but they are in a disabled state. This early deployment step will save time and limit exchanges with the SAS during the various adaptation cycles.

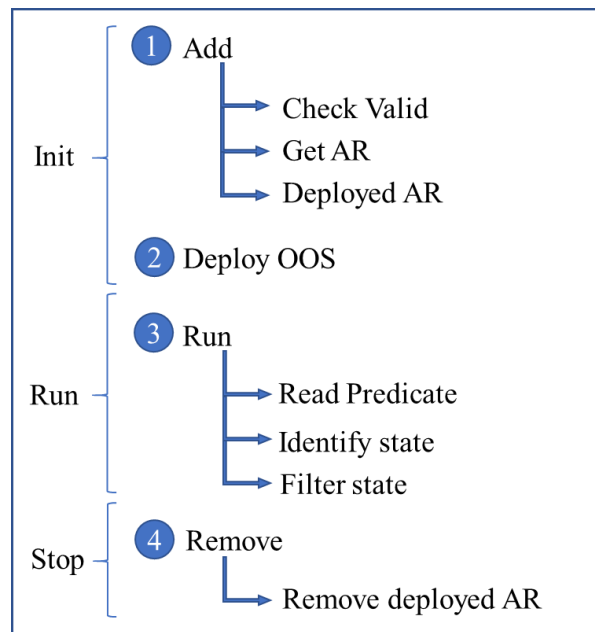


Figure 6.13: The viewpoint life cycle

Once initialized, the viewpoint instantiates the three modules that compose it and then starts its execution cycle, which consists of reading the predicate values, identifying the current state, and if necessary, managing the case of invalid states. And then, the cycle continues with the filter state to select the validation state for adaptation. The first of these three modules have a special place since it provides the link with the observation system to collect the value of predicates. This module begins to deploy the observation rules in the observation system and then configure the data collection mode. It has two collection modes are available:

On request: the predicate values are requested by the observation module of the CAM according to its rhythm.

On change: the values of the predicates are transmitted by the observation system each time they change

Regardless of the mode of data collection, these will be either whenever the value of one of the predicates changes (this mode is very responsive but changes too frequent can cause problems) or according to a configured frequency.

Finally, when deleting the viewpoint, the deployed rules will be removed from the SAS. Note that there is a phase of the life cycle called "suspended" that blocks the execution of the viewpoint without removing the observation rules of the observation system and the rules of adaptation of the SAS. This phase is triggered by a too important succession of invalid states for this viewpoint.

6.4.3 The simulator of self-adaptive system and observation system

The preliminary tests of our CAM were carried out using the WComp software [Ferry-2012] and its adaptation layer based on the aspect of the assembly to serve as SAS but also as an observation system. For that, we used two different instances of WComp, one for each role.

The use of sensors, their deployment in more or less important test environments, the observation of the modifications made to the final application once the adaptation mechanism of the CAM and the SAS works well.

However, the implementation of large-scale examples, based on WComp or other SAS and observation systems, is very complex. And it is very difficult to evaluate the choices made during our work. To overcome this problem, we have developed two simulators simplifying the implementation of tests and allowing us to focus on the CAM and its operation. Developed simulators behave like the real platforms they simulate so that the CAM cannot know that it uses simulators.

6.4.3.1 Observation simulator

The observing system simulator has been implemented on Node.js (version 10.9.0 or later) and with the help of the Angular CLI tool (version 7.1.3 or later). It comes in the form of a simple web interface that lists all the predicates whose CAM wants the evaluation. It is then possible to indicate the value of a predicate by ticking (true) or not (false) the checkbox associated with each predicate.

An automatic mode ticking a predicate random combination and changing it every 3 seconds has been added to provide a bit of dynamic to the observation. Figure 6.13 shows the interface of the observation simulator. In this implementation, we add three independently viewpoints into CAM.

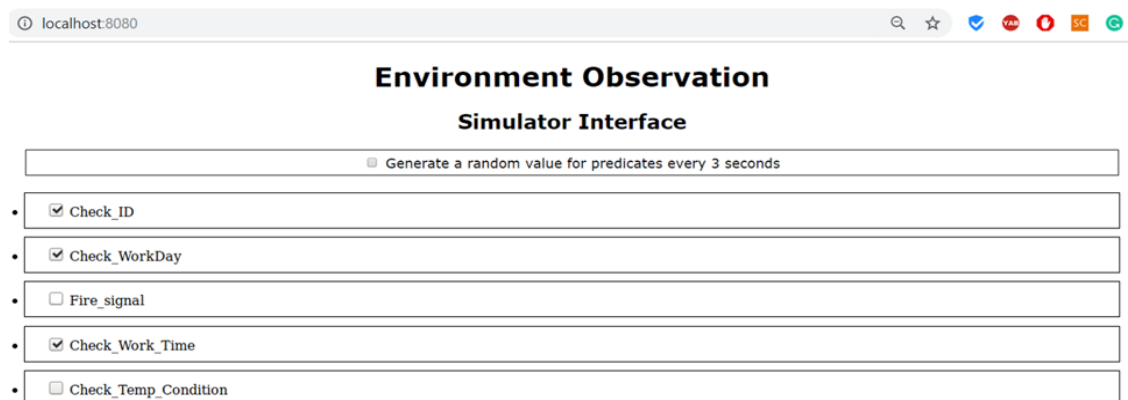


Figure 6.14: The observation simulator interface

We might see that the predicates "*Check_ID; Check_WorkDay; Fire_Signal; Check_Work_Time; Check_Temp_Condition*" have been deployed. Three predicates are checked and evaluated as true with the CAM, while the two others unchecked will be evaluated as false.

6.4.3.2 Self-adaptive system simulator

Like our observation simulator, we have developed a SAS simulator. It also comes in the form of a web interface and is based on the same technology as our previous simulator.

It will then be very easy to viewpoint the list of deployed adaptation rules and their status within the SAS: red for disabled and green for activated (as shown in Figure 6.14). In the SAS simulator interface, we only show the label of each set of adaptation rules (as mentioned in chapter 3). Each label corresponds to a set of adaptation rules which is integrated inside of it. This simulator clarifies that the adaptation rules are deployed and executed to SAS.

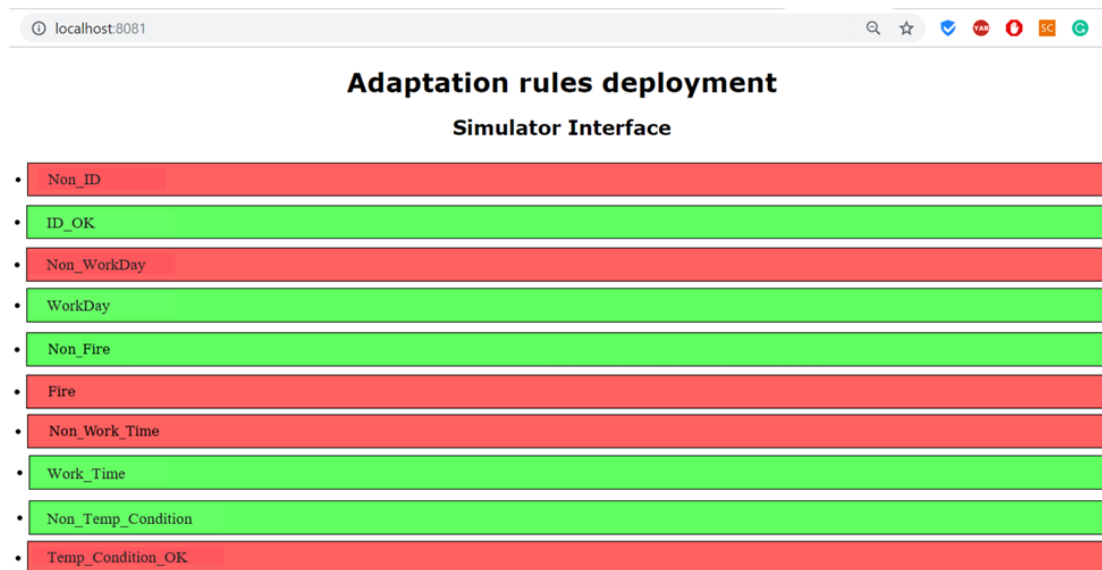


Figure 6.15: The adaptation simulator interface

6.5 Chapter conclusion

In this chapter, we proposed the separation of relational database (objective designer database) with a runtime system that can simplify the designing system. The designer can build the objective designer database at design time. One of the most significant advantages of using a relational database is that it lets end users and designer access and use the same data while managing data integrity. We also present the solution to associate adaptation rules to each OOS by using information from the objective designer database. And then, we proposed three solutions to answer the question: "when is the

best moment to associate the objective of the states to the adaptation rules in the CAM?”. We might see that the first and second solutions will impact to runtime system if it has a problem with the relational database. While the third solution will minimize the impact of the problem in the relational database to the runtime system because the viewpoints will be evaluated before CAM use them on detecting the conflict process. The operation of CAM and SAS is not interrupted if we have any reconfiguration action with the objective designer database.

Our implementation with CAM can show that we can use independent viewpoints to model contextual concerns. The adding or delete the viewpoint can be done easily by users or administrators. However, we do not support the tool to evaluate improving the responsiveness of self-adaptive system. In addition, it is also difficult to find all the problems of CAM without testing in a real system of observation mechanism and SAS.

CHAPTER 7: Conclusion and Future works

In this chapter, we step back on the general considerations of the work presented in this thesis and present the contributions of this study. We also identify the research areas that we recommend for future work. This chapter is structured as follows: Section 7.1 presents the general considerations; section 7.2 presents the contributions of our approach, and finally, section 7.3 proposes the future works.

7.1 General considerations

Today, self-adaptive system has emerged as an important part of ubiquitous applications. It allows the applications to use the contextual information in order to adapt behaviors accordingly to the user's situation and needs. As mentioned in chapter 1, many researchers have defined the context according to the type of attributes or entities observed such as [Dey-2001], [Brown-1997]. Following their opinion, context is all useful information on the dynamic environment that related to application and designer's concern. From that, we might see that the situation of context at one time is a combination of values of variables observed, and context is the set of possible situations. However, if the designer wants to model the context as a whole in a unitary way then the task of the designers on context modeling will become complex because they must develop a single big context model with the combination of many concepts on the different context domains. In this thesis, we simplify the designer's task by splitting this context model into many small models. Each small model of context corresponds to one designer's concern. And, each model of the designer's concern is called a specific viewpoint. The specific viewpoint is described independently of any system and corresponds to the description of one designer's concern. Each specific viewpoint is structured by a set of states (with the state of a viewpoint is identifiable using real world observation).

We might see that at one time of observation, the system can observe many values of predicates from viewpoints and identify a set of states based on vectors of predicates (one state for each viewpoint). The situation of context at one time corresponds perfectly to all the states of different viewpoints observed at that time. From that, we can conclude that context is formed by total situations observed that existing during all observation times. Therefore, we can describe the context through a series of photographs (set of situations) of the multiple-element components (states) on the environment during the observation time, as shown in figure 7.1.

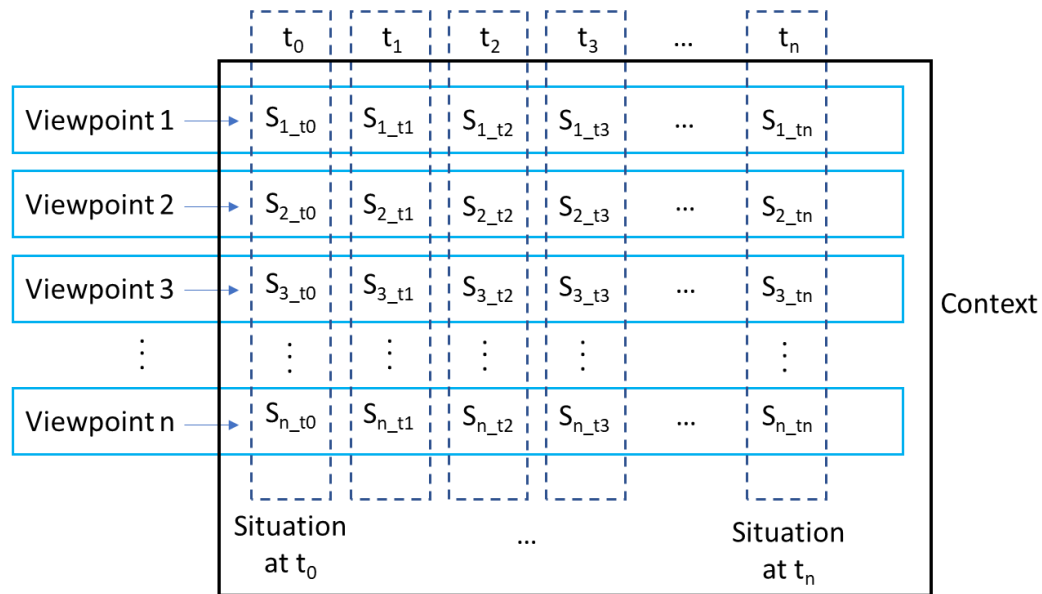


Figure 7.1: The diagram of context description in our approach

With:

- Context is defined based on a system and a set of concerns. It corresponds to all possible situations over time in relation to the couple (system, concerns).
- A situation is formed by the states characterizing the couple (system, concerns) at one time.
- A viewpoint is defined independently of any system. It corresponds to the description of one concern in the form of a state set.
- A state of viewpoint is identified through the values of the real-world observation (values of the predicate of viewpoints, as mentioned in chapter 3).

7.2 Research contributions

The main contributions of this thesis are reflected in the works of addressing the challenges of research that we have presented in chapter 1. These contributions of our approach are discussed in the sequel.

7.2.1 Simplifying the designer's task

As mentioned in chapter 1, the first one of the challenges in this thesis is simplifying the designer's task on context modeling. This challenge was resolved by using independent viewpoints in context modeling. The contributions of our approach in this first part are shown in three aspects:

Modeling independent viewpoint: Using independent viewpoints allows designers to focus on their domain of expertise. Instead, each expert can specify its own viewpoint without having to know the viewpoints specified by the other experts. Each viewpoint related to one designer's concern, and it is built independently. Therefore, each viewpoint can be reused in different applications.

Context intermediate model: We have argued throughout this thesis that it is not easy for the CAM to handle the multiple data types from viewpoints. Instead, we introduced the context intermediate model to use as a standard data for all viewpoints. We remove the adaptation rules (AR) out of CIM schema because designers do not work on adaptation level; they work on the concept level. The adding AR for each objective of states will be done automatically by CAM when a new viewpoint is added into CAM. This work can help the designer do not take much time on modeling their concerns. The task of the designer becomes simpler because they can model their concerns without caring about adaptation rules.

Develop tools to support designer: As described in chapter 3, we developed two specific transforming tools to convert specific viewpoints (BPMN, CTT) into standard data following the CIM schema. The transforming process can be done automatically by these tools; this work can help the designer save time on transforming data from a specific viewpoint. And, we also support a validation tool which used to validate the XML file following CIM schema. These tools can reduce the application development effort and time of the experts. It allows the experts to better focus on their core expertise, instead of being bothered with the details related to application realization.

7.2.2 Detecting and solving the conflicts between viewpoints

Using independent viewpoints has not only advantages. One limitation is facing a new obstacle when we merge viewpoints at runtime for adaptation. This problem is the conflict ability between viewpoints at runtime. This problem occurs because each viewpoint is designed by different designers, and their viewpoint description focuses on the purpose of their application. Therefore, when the viewpoints are merged at runtime, the adaptation situation of this viewpoint can conflict with the others. However, we have argued throughout this thesis that it is impossible to solve the conflicts between viewpoints as design time. We want to detect and solve these conflicts at CAM before executing rules for SAS. For this, we have introduced the concept of the objective of state in the CIM. The conflict between viewpoints at one moment is also a conflict between OOS of viewpoints. From that, we have introduced solutions to detect and solve conflicts between OOS, as mentioned in chapter 4 and 5. This work reduces the

number of conflicts that must be solved in SAS. Therefore, it can help SAS save time on the decision step.

7.2.3 Improve reactivity of SAS

To improve the reactivity of SAS, we have introduced a new architecture of SAS with multiple layers of adaptation. The contributions of our approach in this second part are shown in the advantages of the SAS architecture pattern.

Self-adaptive system architecture patterns: At runtime, we propose to use two independent cycles of observation, as shown in Figure 7.2. The first cycle of observation predicate uses to handles the current context and the information of each viewpoint managed by context-aware management (CAM). The second cycle handles the adaptation of the application according to the rules granted by the previous level. We might see that this solution can provide more time to the CAM to analyze the current context and preserve the responsiveness of SAS.

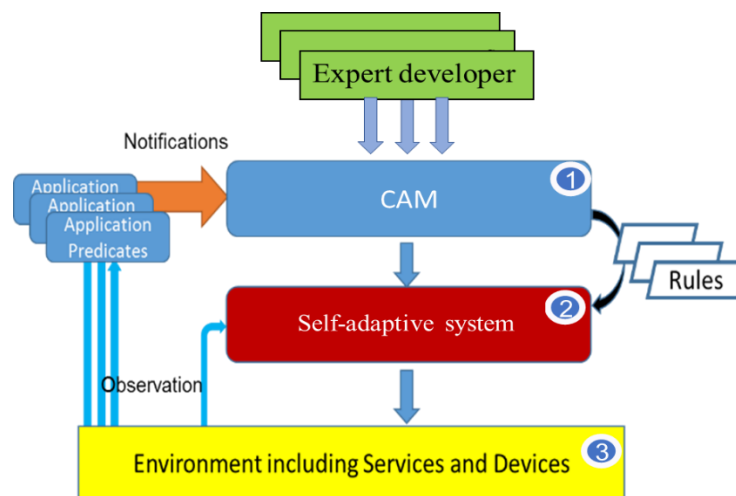


Figure 7.2: Two independent cycles of observation of SAS.

With the independence of the execution cycle of the SAS and CAM, the operations of the application layer do not depend on the operation of the adaptation process. It is an advantage that the application can continue to operate during the decision stage of SAS. In case of having a context switching that finds by CAM, the SAS be required an interrupted to reconfigure the system. That means the SAS can focus only on adaptation that can help to reduce adaptation time and a part of the problem about system consumption.

7.3 Future works

We have identified our future works in the following two areas:

Observation mechanism: In this thesis, we focus on the solutions to help designers on context modeling at design time, and context handling platform in CAM. In the future, we want to continue with the research on observation mechanism to support system on handling uncertainty in the value of observation, filtering, and redundancy with noise. It will be better if we can integrate a filtering problem with uncertainty in the observation mechanism. This work can reduce the problem related to Invalid states, as mentioned in chapter 3. In addition, we want to provide an observation service that can operate independently with CAM. It observes context and provides useful contextual information for CAM automatically (do not depend on trigger observation from CAM). This solution can open the way to use one observation mechanism for more than one CAM at one time.

Supporting the-end user on managing the system: We want to develop a monitoring service to help the end-user on interaction and management all layers of the system from viewpoint to SAS. This service provides a simple interface to support the end-user on selecting and adding viewpoint, managing SAS without needs about computer science knowledge. Moreover, we also want to develop a smart observation system that can observe both SAS and user automatically. It can support the end-user on designing or modifying the specific viewpoint without attending from the expert. From that, the end-user can easy to use specific viewpoints from experts and modify them following their viewpoint concerns. This work can create flexibility in building specific viewpoints and reuse the viewpoint in other applications.

Bibliography

My own bibliography

[Do-2019]

Do, T. C., Lavirotte, S., Rey, G., Le Thanh, N., & Tigli, J. Y. From BPMN to Live Application: How the Context Can Drive an Auto-Adapted System. In 2019 IEEE-RIVF International Conference on Computing and Communication Technologies 2019, pp. 1-6, IEEE.

[Rey-2017]

Rey, G., Do, T. C., Tigli, J. Y., Lavirotte, S., & Le Thanh, N. (2017). Intermediate Common Model-The Solution to Separate Concerns and Responsiveness in Dynamic Context-Aware System. *Journal of Communications and Computer Engineering*, pp 44-60.

General bibliography

[Abeywickrama-2012]

Abeywickrama, D. B., Bicocchi, N., & Zambonelli, F. (2012). SOTA: Towards a general model for self-adaptive systems. In 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 48-53, IEEE.

[Achilleos-2010]

Achilleos, A., Yang, K., & Georgalas, N. (2010). Context modelling and a context-aware framework for pervasive service creation: A model-driven approach. *Pervasive and Mobile Computing*, pp 281-296.

[Aguilar-2018]

Aguilar, J., Jerez, M., & Rodríguez, T. (2018). CAMEonto: Context awareness meta ontology modeling. *Applied computing and informatics*, 14(2), pp 202-213.

[An-2005]

An, Y., Borgida, A., & Mylopoulos, J. (2005). Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 1152-1169. Springer, Berlin, Heidelberg.

[Assad-2007]

Assad, M., Carmichael, D. J., Kay, J., & Kummerfeld, B. (2007). PersonisAD: Distributed, active, scrutable model framework for context-aware services. In *International Conference on Pervasive Computing*, pp. 55-72, Springer, Berlin, Heidelberg.

[Astrova-2007]

Astrova, I., Korda, N., & Kalja, A. (2007). Rule-based transformation of SQL relational databases to OWL ontologies. In *Proceedings of the 2nd International Conference on Metadata & Semantics Research*, pp. 415-424.

[Astrova-2009]

Astrova, I. (2009). Rules for mapping SQL relational databases to OWL ontologies. In *Metadata and semantics*, pp. 415-424, Springer, Boston, MA.

[Baldauf-2007]

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, pp 263-277.

[Bauer-2001]

Bauer, B., Müller, J. P., & Odell, J. (2001). Agent UML: A formalism for specifying multiagent software systems. *International journal of software engineering and knowledge engineering*, 11(03), pp 207-230.

[Bazire-2005]

Bazire, M., & Brézillon, P. (2005). Understanding context before using it. In *International and Interdisciplinary Conference on Modeling and Using Context*, pp. 29-40, Springer, Berlin, Heidelberg.

[Benslimane-2006]

Benslimane, D., Arara, A., Falquet, G., Maamar, Z., Thiran, P., & Gargouri, F. (2006). Contextual ontologies. In *International Conference on Advances in Information Systems*, pp. 168-176, Springer, Berlin, Heidelberg.

[Bettini-2010]

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2), pp 161-180.

[Bolchini-2013]

Bolchini, C., Carminati, M., Miele, A., & Quintarelli, E. (2013). A framework to model self-adaptive computing systems. In *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, pp. 71-78, IEEE.

[Bonino-2008]

Bonino, D., & Corno, F. (2008). Dogont-ontology modeling for intelligent domotic environments. In *International Semantic Web Conference*, pp. 790-803, Springer, Berlin, Heidelberg.

[Brézillon-1999]

Brézillon, P. (1999). Context in Human-machine problem solving: a survey. *The Knowledge Engineering Review*, 14(1), pp 47-80.

[Brown-1997]

Brown, P. J., Bovey, J. D., & Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE personal communications*, 4(5), pp 58-64.

[Brun-2009]

Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., ... & Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems*, pp. 48-70, Springer, Berlin, Heidelberg.

[Carreira-2014]

Carreira, P., Resendes, S., & Santos, A. C. (2014). Towards automatic conflict detection in home and building automation systems. *Pervasive and Mobile Computing*, 12, pp 37-57.

[Chaabouni-2016]

Chaabouni, M., Laroussi, M., Piau-Toffolon, C., Choquet, C., & Ghezala, H. B. (2016). A context modeling approach and a tool for reusing learning scenarios. 24th International Conference on Computers in Education - ICCE'2016, Nov 2016, Bombay, India, pp 288-293.

[Chaari-2006]

Chaari, T., Ejigu, D., Laforest, F., & Scuturici, V. M. (2006). Modeling and Using Context in Adapting Applications to Pervasive Environments. In *ICPS*, pp. 111-120.

[Chen G.-2000]

Chen, G., & Kotz, D. (2000). A survey of context-aware mobile computing research. Dartmouth Computer Science Technical Report TR2000-381, pp 1-16

[Chen H.-2003]

Chen, H., Finin, T., & Joshi, A. (2003). An intelligent broker for context-aware systems. In *Adjunct Proceedings of Ubicomp 2003*, pp. 183-184.

[Chen H.-2004]

Chen, H., Perich, F., Finin, T., & Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. pp. 258-267.

[Cheng-2009]

Cheng, B. H., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., ... & Serugendo, G. D. M. (2009). Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, pp. 1-26. Springer, Berlin, Heidelberg.

[Cheung-Foo-Wo-2007]

Cheung-Foo-Wo, D., Tigli, J. Y., Lavirotte, S., & Riveill, M. (2007). Self-adaptation of event-driven component-oriented middleware using aspects of assembly. In *Proceedings of the 5th international workshop on Middleware for pervasive and ad-hoc computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference*, pp. 31-36, ACM.

[Continuum-2012]

Continuum project, Continuum ANR, Programme VERTHEREFORE, Continuum ANR-08-VERS-005, 12-2008/09-2012.

[Costa-2007]

Costa, P. D. (2007). Architectural Support for Context-Aware Applications: From Context Models to Services Platforms. CTIT Ph.D.-Thesis Series, No. 07-108

[David-2005]

David, P. C., & Ledoux, T. (2005). WildCAT: a generic framework for context-aware applications. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pp. 1-7, ACM.

[Dey-1998]

Dey, A. K. (1998). Context-aware computing: The CyberDesk project. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*, pp. 51-54.

[Dey-2001]

Dey, A. K., Abowd, G. D., & Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2-4), pp 97-166.

[Dey A.-2001]

Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1), pp 4-7.

[Dunlop-2002]

Dunlop, N., Indulska, J., & Raymond, K. (2002). Dynamic conflict detection in policy-based management systems. In *Proceedings. Sixth International Enterprise Distributed Object Computing*, pp. 15-26, IEEE.

[Edwards-1997]

Edwards, W. K. (1997). Flexible conflict detection and management in collaborative applications. In *ACM Symposium on User Interface Software and Technology*, pp. 139-148.

[Ejigu-2007]

Ejigu, D., Scuturici, M., & Brunie, L. (2007). An ontology-based approach to context modeling and reasoning in pervasive computing. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07)*, pp. 14-19, IEEE.

[Fahy-2004]

Fahy, P., & Clarke, S. (2004). CASS—a middleware for mobile context-aware applications. In *Workshop on context awareness, MobiSys*. pp 1-6.

[Ferry-2012]

Ferry, N., Hourdin, V., Lavirotte, S., Rey, G., Riveill, M., & Tigli, J. Y. (2012). Wcomp, middleware for ubiquitous computing and system focused adaptation. hal-01330474

[Fuchs-2005]

Fuchs, F., Hochstatter, I., & Krause, M. (2005). A metamodel approach to context information. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 8-14, IEEE.

[Glaser-2015]

Glaser, N., (2015), Impact, Urgency and Priority, Getting Started with BMC Remedyforce, BMC.

[Greiner-2014]

Greiner, R., Puig, J., Huchery, C., Collier, N., & Garnett, S. T. (2014). Scenario modelling to support industry strategic planning and decision making. *Environmental modelling & software*, 55, pp 120-131.

[Gu-2005]

Gu, T., Pung, H. K., & Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and computer applications*, 28(1), pp 1-18.

[Gutowski-2017]

Gutowski, N., Amghar, T., Camp, O., & Hammoudi, S. (2017). A framework for context-aware service recommendation for mobile users: A focus on mobility in smart cities. *From Data To Decision*. pp 1-17

[Ham-1997]

Ham, C. (1997). Priority setting in health care: learning from international experience. *Health policy*, 42(1), pp 49-66.

[Halpin-1998]

Halpin, T. (1998). Object-role modeling (ORM/NIAM). In *Handbook on architectures of information systems*, pp 81-103. Springer, Berlin, Heidelberg.

[Harris-2016]

Harris, A. (2016). Integrating Business Process Management To Model Context In Healthcare: A Case Study Using Perioperative Processes (Doctoral dissertation, Université d'Ottawa/University of Ottawa).

[Henricksen-2002]

Henricksen, K., Indulska, J., & Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. In *International Conference on Pervasive Computing*, pp 167-180. Springer, Berlin, Heidelberg.

[Henricksen-2003]

Henricksen, K. (2003). *A framework for context-aware pervasive computing applications*. Ph.D Thesis, University of Queensland, Queensland.

[Henricksen-2005]

Henricksen, K., Indulska, J., McFadden, T., & Balasubramaniam, S. (2005). Middleware for distributed context-aware systems. In *OTM Confederated*

International Conferences" On the Move to Meaningful Internet Systems", pp 846-863. Springer, Berlin, Heidelberg.

[Hull-1997]

Hull, R., Neaves, P., & Bedford-Roberts, J. (1997). *Towards situated computing*, pp. 146-153. Hewlett Packard Laboratories.

[Hussein-2010]

Hussein, M., Han, J., & Colman, A. (2010). Specifying and verifying the context-aware adaptive behavior of software systems. *Technical Report# C3-516_03*, Swinburne University of Technology.

[Jaouadi I.-2015]

Jaouadi, I., Djemaa, R. B., & Abdallah, H. B. (2015). Approach to Model-Based Development of Context-Aware Application. *Journal of Computer and Communications*, 3(05), pp 212-219.

[Jaouadi-2015]

Jaouadi, I., Djemaa, R. B., & Abdallah, H. B. (2015). A generic metamodel for context-aware applications. In *Progress in Systems Engineering*, pp. 587-594. Springer, Cham.

[Jordan-1998]

Jordan, M. I. (Ed.). (1998). *Learning in graphical models* (Vol. 89). Springer Science & Business Media.

[Keling-2011]

Keling D., Marc D., Philippe R., A Survey of adaptation systems. *International Journal on Internet and Distributed Computing Systems*, 2011, 2 (1), pp.1-18.

[Kephart-2004]

Kephart, J. O., & Walsh, W. E. (2004). An artificial intelligence perspective on autonomic computing policies. In *Proceedings. Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004. POLICY 2004*. pp. 3-12. IEEE.

[Kim-2008]

Kim, E., & Choi, J. (2008). A context management system for supporting context-aware applications. In *2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Vol. 2, pp. 577-582. IEEE.

[Kim-2012]

Kim J. D., Son J., and Baik D. K., (2012). CA5W1HOnto: Ontological Context-AwareModel Based on 5W1H. *International Journal of Distributed Sensor Networks*. doi:10.1155/2012/247346, pp 1-11.

[Krupitzer-2015]

Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., & Becker, C. (2015). A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17, pp 184-206.

[Leite-2007]

Leite, M. M., Calvi, C. Z., Pessoa, R. M., Pereira Filho, J. G., & Pereira Filho, J. (2007). A MOF Metamodel for the Development of Context-Aware Mobile Applications. In *22nd Annual ACM Symposium on Applied Computing, SAC 2007*. pp 1-6

[Li-2005]

Li, M., Du, X. Y., & Wang, S. (2005). Learning ontology from relational database. In *2005 International Conference on Machine Learning and Cybernetics*, Vol. 6, pp. 3410-3415. IEEE.

[Li-2015]

Li, X., Eckert, M., Martinez, J. F., & Rubio, G. (2015). Context aware middleware architectures: survey and challenges. *Sensors*, *15*(8), pp 20570-20607.

[Mahmud-2018]

Mahmud, U., Hussain, S., & Yang, S. (2018). Power Profiling of Context Aware Systems: A Contemporary Analysis and Framework for Power Conservation. *Wireless Communications and Mobile Computing*, 2018. pp 1-15.

[Marcinkowski-2012]

Marcinkowski, B., & Kuciapski, M. (2012). A business process modeling notation extension for risk handling. In *IFIP International Conference on Computer Information Systems and Industrial Management*, pp. 374-381. Springer, Berlin, Heidelberg.

[Maternaghan-2013]

Maternaghan, C., & Turner, K. J. (2013). Policy conflicts in home automation. *Computer Networks*, pp 2429-2441.

[Miraoui-2008]

Miraoui, M., Tadj, C., & Amar, C. B. (2008). Architectural survey of context-aware systems in pervasive computing environment. *Ubiquitous Computing and Communication Journal*, pp 68-76.

[Motti-2013]

Motti, V. G., & Vanderdonckt, J. (2013). A computational framework for context-aware adaptation of user interfaces. In *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*, pp. 1-12. IEEE.

[Nandyala-2016]

Nandyala, C. S., & Kim, H. K. (2016). Crop Production Context-Aware Enterprise Application Using IoT. *International Journal of Software Engineering and Its Applications*, *10*(4), pp 189-200.

[Nešković-2015]

Nešković, S., & Matić, R. (2015). Context modeling based on feature models expressed as views on ontologies via mappings. *Computer Science and Information Systems*, *12*(3), pp 961-977.

[Nóbrega-2005]

Nóbrega, L., Nunes, N. J., & Coelho, H. (2005). Mapping concurrent task trees into UML 2.0. In *International Workshop on Design, Specification, and Verification of Interactive Systems*, pp. 237-248. Springer, Berlin, Heidelberg.

[Pascoe-1998]

Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In *2nd international symposium on wearable computers*, pp. 92-99. IEEE Computer Soc.

[Paternò-2002]

Paternò, F. (2002). ConcurTaskTrees: an engineered approach to model-based design of interactive systems. *The handbook of analysis for human-computer interaction*, pp 483-500.

[Perera-2013]

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1), pp 414-454.

[Rathi-2018]

Rathi, S., & Alam, A. (2018). ESO-5W1H Framework: Ontological model for SITL paradigm. In *HumL@ ISWC*, pp. 51-63.

[Ryan-1997]

Ryan, N. (1997). Mobile Computing in a Fieldwork Environment: Metadata Elements. Project working document, version 0.2.

[Resendes-2014]

Resendes, S., Carreira, P., & Santos, A. C. (2014). Conflict detection and resolution in home and building automation systems: a literature review. *Journal of Ambient Intelligence and Humanized Computing*, 5(5), pp 699-715.

[Rey-2005]

Rey, G., (2005) Contexte en Interaction Homme-Machine: le contexteur, PhD Thesis, Université Joseph-Fourier-Grenoble I.

[Ruiz-2012]

Ruiz, F., Garcia, F., Calahorra, L., Llorente, C., Gonçalves, L., Daniel, C., & Blobel, B. (2012). Business process modeling in healthcare. *Stud Health Technol Inform*, 179, pp 75-87.

[Sain-2010]

Sain, M., Lee, H., & Chung, W. Y. (2010). Designing context awareness middleware architecture for personal healthcare information system. In 2010 The 12th International Conference on Advanced Communication Technology (ICACT), Vol. 2, pp. 1650-1654. IEEE.

[Salehie-2009]

Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TSAS)*, 4(2), pp 1-14.

[Schilit-1995]

Schilit, W. N. (1995). *A system architecture for context-aware mobile computing* PhD Thesis, Doctoral dissertation, Columbia University.

[Schumm-2009]

Schumm, D., Karastoyanova, D., Leymann, F., & Nitzsche, J. (2009). On visualizing and modelling BPEL with BPMN. In *2009 Workshops at the Grid and Pervasive Computing Conference*, pp. 80-87. IEEE.

[Sen-2010]

Sen, J. (2010). Ubiquitous computing: potentials and challenges. *Proceedings of the International Conference on Trends & Advances in Computation & Engineering*. pp 1323-1346.

[Shen-2006]

Shen, G., Huang, Z., Zhu, X., & Zhao, X. (2006). Research on the Rules of Mapping from Relational Model to OWL. In *OWLED*.

[Shishkov-2018]

Shishkov, B., Larsen, J. B., Warnier, M., & Janssen, M. (2018, July). Three categories of context-aware systems. In *International Symposium on Business Modeling and Software Design*, pp. 185-202. Springer, Cham.

[Siadat-2012]

Siadat, S. H., & Song, M. (2012). Understanding requirement engineering for context-aware service-based applications. *Journal of Software Engineering and Applications*, 5(08), pp 536-544.

[Spicker-2009]

Spicker, P. (2009), What is a priority? *Journal of Health Services Research & Policy*, pp 112-116

[Strang-2004]

Strang, T., & Linnhoff-Popien, C. (2004). A context modeling survey. In *Workshop Proceedings*. pp 1-8

[Sun-2014]

Sun, Y., Wang, X., Luo, H., & Li, X. (2014). Conflict detection scheme based on formal rule model for smart building systems. *IEEE Transactions on Human-Machine Systems*, 45(2), pp 215-227.

[Thyagaraju-2008]

Thyagaraju, G. S., Joshi, S. M., Kulkarni, U. P., NarasimhaMurthy, S. K., & Yardi, A. R. (2008). Conflict resolution in multiuser context-aware environments. In *2008*

International Conference on Computational Intelligence for Modelling Control & Automation, pp. 332-338. IEEE.

[Tigli-2009]

Tigli, J. Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., & Riveill, M. (2009). WComp middleware for ubiquitous computing: Aspects and composite event-based Web services. *annals of telecommunications-Annales des télécommunications*, 64(3-4), pp 197-214.

[Trollmann-2015]

Trollmann, F. (2015). Detecting adaptation conflicts at run time using models@run. time. PhD Thesis, Technischen Universität Berlin.

[Uchitel-2004]

Uchitel, S., Chatley, R., Kramer, J., & Magee, J. (2004). System architecture: the context for scenario-based model synthesis. In *ACM SIGSOFT Software Engineering Notes*, Vol. 29, No. 6, pp. 33-42. ACM.

[Vieira-2007]

Vieira, V., Tedesco, P., Salgado, A. C., & Brézillon, P. (2007). Investigating the specifics of contextual elements management: the cematika approach. In *International and Interdisciplinary Conference on Modeling and Using Context*, pp. 493-506. Springer, Berlin, Heidelberg.

[Vieira-2010]

Vieira, V., Tedesco, P., & Salgado, A. C. (2010). Using a metamodel to design structural and behavioral aspects in context-sensitive groupware. *The 14th International Conference on Computer Supported Cooperative Work in Design*, pp 59-64. IEEE.

[Wang-2004]

Wang, X., Zhang, D., Gu, T., & Pung, H. K. (2004). Ontology Based Context Modeling and Reasoning using OWL. In *Percom workshops*, Vol. 18, pp. 1-22.

[Wang-2011]

Wang, W. M., & Ting, S. L. (2011). Development of a computational simulation model for conflict management in team building. *International Journal of Engineering Business Management*, 3, pp 1-14.

[Weiser-1993]

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7), pp 75-84.

[Weiser-1991]

Weiser, M. (1991). The computer for the 21st century. *Scientific American* Vol 265, No 3, pp 94-104.

[Weyns-2017]

Weyns, D. (2017). Software engineering of self-adaptive systems: an organised tour and future challenges. *Chapter in Handbook of Software Engineering*.

[White-2004]

White, A. S. (2004) Business Process Modeling Notation, Business Process Management Initiative (BPMI)

[Xu-2013]

Xu, N., Zhang, W. S., Yang, H. D., Zhang, X. G., & Xing, X. (2013). CACOnt: A ontology-based model for context modeling and reasoning. In Applied Mechanics and Materials, Vol. 347, pp. 2304-2310. Trans Tech Publications.

[Yagita-2015]

Yagita, M., Ishikawa, F., & Honiden, S. (2015). An application conflict detection and resolution system for smart homes. In Proceedings of the First International Workshop on Software Engineering for Smart Cyber-Physical Systems, pp. 33-39. IEEE Press.

[Yang-2011]

Yang, L., Hu, Z., Long, J., & Guo, T. (2011). 5W1H-based conceptual modeling framework for domain ontology and its application on STPO. In 2011 Seventh International Conference on Semantics, Knowledge and Grids, pp. 203-206. IEEE.

[Yu J.-2010]

Yu, J., Huang, Y., Cao, J., & Tao, X. (2010). Middleware support for context-awareness in asynchronous pervasive computing environments. In 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pp. 136-143. IEEE.

[Zhang-2017]

Zhang, A., Ji, C., Bao, Y., & Li, X. (2017). Conflict analysis and detection based on model checking for spatial access control policy. Tsinghua Science and Technology, 22(5), pp 478-488.

Appendix

In this thesis, we studied and developed several tools, which we used for the simulator of observation and self-adaptive system. In this appendix, we give some more details for each of these tools and support our commands to set up and control each simulator.

Observation and Deployment rules simulator

Packages require

* [Node.js] (<https://nodejs.org/en/>)

* [Angular Cli] (<https://angular.io/cli>)

* [dotnet core] (<https://dotnet.microsoft.com/download>)

The version used

Angular

Angular CLI: 7.1.3

Node: 10.13.0

Angular:

Package	Version
---------	---------

@angular-devkit/architect	0.11.3
---------------------------	--------

@angular-devkit/core	7.1.3
----------------------	-------

@angular-devkit/schematics	7.1.3
----------------------------	-------

@schematics/angular	7.1.3
---------------------	-------

@schematics/update	0.11.3
--------------------	--------

rxjs	6.3.3
------	-------

typescript	3.1.6
------------	-------

Dotnet: dotnet --version: 2.2.101

Start the observation interface

Start the server (in the background):

```
cd observation-interface
```

```
npm install
```

```
npm start
```

Start the deployment interface

Start the server (in the background):

```
cd deployment-interface
npm install
npm start
```

Start the user interface

Start the Angular application:

```
cd gui/
npm install
ng serve
```

* The interface is tested on Mozilla FireFox Quantum 65.0 (64-bit)

* Open GUI in a browser (<http://localhost:4200/>)

Start the context manager

Start the Dotnet core application:

```
``shell
cd src
dotnet run
```

Launching applications:

* These scripts allow you to install the npm packages from the package.json and launch the applications

* ##### Linux

```
``shell
sudo sh install.sh
sudo sh run.sh
```

* ##### Windows

```
``bash
run.bat
```

Context Manager in Console Mode:

* add

Usage: add [arguments] [options]

Arguments:

FILE_PATH The path of the file of the view that should be use!

Options:

-f|--freq <frequency> Refresh frequency of the new view chain to create!

-h|--help Show help information

* freq

Usage: freq [arguments] [options]

Arguments:

FREQUENCY The refresh rate (in ms) of given modules!

Options:

-d|--deploy To select the deployment modules!

-v|--view <view> To select the view with the given index modules!

-h|--help Show help information

* ls

Usage: ls [options]

Options:

-l|--long Show all the information about the current used views!

-h|--help Show help information

* remove

Usage: remove [arguments] [options]

Options:

-h|--help Show help information

* restart

Usage: restart [options]

Options:

-h|--help Show help information

* stop

Usage: stop [options]

Options:

-h|--help Show help information

* verbose

Usage: verbose [options]

Options:

-h|--help Show help information

Deployment rules interface coding:

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Deployment rules interface</title>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script src="http://localhost:8080/socket.io/socket.io.js"></script>
  <style>
    body {font-family: verdana; padding: 0 150px 0 150px;}
    h1, h2 {text-align: center;}
    ul li {font-size: 100%; border: 1px solid black; padding: 10px 25px 10px 25px;
      margin: 10px;}
    ul li p {font-size: 50%; border: 0; padding: 0; margin: 0;}
    li.deployed {background-color: #60FF60;}
    li.undeployed {background-color: #FF6060;}
  </style>
</head>
<body>
  <h1>Adaptation rules deployment</h1>
  <h2>Simulator Interface</h2>
  <div id="rulelist">
    <ul>
      <rule-item v-for="r in rules" v-bind:rule="r" v-bind:key="r.rule_id">
      </rule-item>
    </ul>
  </div>
  <script>
    Vue.component('rule-item', { props: { 'rule': Object }, template: '<li v-bind:class="{
    deployed: rule.deployed, undeployed: !rule.deployed }"> {{ rule.name }} <p> {{ rule.text
    }} </p> </li>' });
    let rulelist = new Vue({el: '#rulelist',data: {rules: []} });
```

```

    fetch('http://localhost:8081/rules')
      .then((res) => res.json())
      .then((res) => { rulelist.rules = res});
    var socket = io.connect('http://localhost:8081');
    socket.on('new_rule', (rule) => {rule.deployed = false; rulelist.rules.push(rule)});
    socket.on('rule_deployment_updated', (rule) => {
      for (var i = 0; i < rulelist.rules.length; i++) {
        if (rulelist.rules[i].rule_id === rule.rule_id) {
          rulelist.rules[i].deployed = rule.deployed;
          return;}
      }
    });
    socket.on('rule_removed', (removed_rule) => {
      for (var i = 0; i < rulelist.rules.length; i++) {
        if (rulelist.rules[i].rule_id == removed_rule.rule_id) {
          rulelist.rules.splice(i, 1); return; }
      }
    });
  </script>
</body>
</html>

```

Observation interface coding:

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Interface d'observation</title>
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
  <script src="http://localhost:8080/socket.io/socket.io.js"></script>
  <style>
    body {font-family: verdana; padding: 0 150px 0 150px;}
    h1, h2 { text-align: center;}

```

```

    ul li { font-size: 100%; border: 1px solid black; padding: 10px 25px 10px 25px;
            margin: 10px; }
    ul li p { font-size: 50%; border: 0; padding: 0; margin: 0; }
  </style>
</head>
<body>
  <h1>Environment Observation</h1>
  <h2>Simulator Interface</h2>
  <div style="text-align: center; border: 1px solid black; padding: 5px">
    <input id="randomizePredicates" type="checkbox" />
    Generate a random value for predicates every 3 seconds
  </div>
  <div id="predicatelists">
    <ul>
      <predicate-item v-for="p in predicates" v-bind:predicate="p" v-bind:key="
p.predicate_id">
        </predicate-item>
    </ul>
  </div>
  <script>
    Vue.component('predicate-item', {
      props: { 'predicate': Object },
      template: '<li> <input type="checkbox" checked="predicate.state" v-
model="predicate.state" v-on:change="check($event)"/> {{ predicate.name }} <p> {{
predicate.text }} </p> </li>',
      methods: {check: function (event) {
        fetch('http://localhost:8080/setPredicate?predicate_id=' +
this.predicate.predicate_id + '&state=' + this.predicate.state);
      }
    }
  });
  let predicatelists = new Vue({el: '#predicatelists', data: { predicates: [] }});
  setInterval(() => { if (document.getElementById("randomizePredicates").checked) {

```

```
        for (var i = 0; i < predicatelist.predicates.length; i++) {
            predicatelist.predicates[i].state = Math.random() >= 0.5;
            fetch('http://localhost:8080/setPredicate?predicate_id=' +
predicatelist.predicates[i].predicate_id + '&state=' + predicatelist.predicates[i].state);
        }
    }
}, 3000);
fetch('http://localhost:8080/predicates')
    .then((res) => res.json())
    .then((res) => {
        predicatelist.predicates = res
    });
var socket = io.connect('http://localhost:8080');
socket.on('new_predicate', (predicate) => {predicate.state = false;
    predicatelist.predicates.push(predicate); });
socket.on('predicate_removed', (removed_predicate) => {
    for (var i = 0; i < predicatelist.predicates.length; i++) {
        if (predicatelist.predicates[i].predicate_id == removed_predicate.predicate_id)
            { predicatelist.predicates.splice(i, 1); return; }
    }
});
</script>
</body>
</html>
```