REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Farhat Abbas Sétif

Faculté des Science

Département d'Informatique

Approche sémantique de génération automatique d'applications agiles en environnement pervasif

Thèse présentée par :

Abderrahim LAKEHAL

En vue de l'obtention du diplôme de

Doctorat 3ème cycle LMD en Informatique

Spécialité : **Systèmes Informatiques Intelligents et Communicants**

Soutenue devant le jury composé de :

Nadjet	KAMEL	Prof. Université Ferhat Abbas Sétif 1	Président
Abdelaziz	LAKHFIF	MCA. Université Ferhat Abbas Sétif 1	Examinateur
Farid	NOUIOUA	MCA. Université de B.B.A	Examinateur
Adel	ALTI	MCA. Université Ferhat Abbas Sétif 1	Directeur
Philippe	ROOSE	HDR. Université de Pau France	Co-directeur

2020

PEOPLE'S DEMOCRATIC REPUBLIC of ALGERIA

Ministry of Higher Education and Scientific Research

Ferhat Abbas University of Sétif

Faculty of Sciences

Department of Computer Science

Semantic approach of automatic generation of agile applications in a pervasive environment

by

Abderrahim LAKEHAL

A thesis submitted in fulfillment of the requirement

For the degree of PhD

Specialty: **Intelligent and Communicating Computer Systems**

Board of examiners:

Nadjet	KAMEL	Prof. Ferhat Abbas Sétif 1 University	President
Abdelaziz	LAKHFIF	MCA. Ferhat Abbas Sétif 1 University	Examiner
Farid	NOUIOUA	MCA. University of B.B.A	Examiner
Adel	ALTI	MCA. Ferhat Abbas Sétif 1 University	Supervisor
Philippe	ROOSE	HDR. Pau France University	Co-supervisor

2020

Acknowledgments

In the name of ALLAH, the most gracious and the most merciful. Foremost, I thank ALLAH for the strength and blessing he has given me to achieve this work.

I would like to express my sincere gratitude to my supervisor **Dr. Adel ALTI** and my co-supervisor **Dr. Philippe ROOSE**, for the continuous support and insightful instructions during the development of this thesis.

My most sincere gratitude goes to my parents for supporting and encouraging me to reach my dreams and aspirations. May ALLAH reward them well.

I also thank each of the members of my dissertation committee for honoring me and accepting to judge my work.

I would like to extend my sincere thanks to all members of the department of computer science, and all my teachers who have contributed directly or indirectly to enrich my knowledge.

I thank the college of the LIUPPA laboratory of University of Pau for the hosting and exchanging of experiences during my internships.

Abstract

With the enormous number of connected objects in pervasive environments, users tend to retrieve contextual information directly from physical environments to manage easily their needs and daily situations. However, with the rapid growth in the size of data and numbers of devices in distributed pervasive environments, the identification of a large number of user's situations becomes a challenging task. In fact, the identification of situations in pervasive environments involves a semantic interpretation of context raw data provided by various heterogeneous objects, which requires new agile solutions in managing and identifying composite situations. None of the existing approaches are able to efficiently manage and identify composite user-centric urgent situations. They lack the flexibility to evolve and adapt according to the user's current context and environment's requirements. Current systems lose performance facing up the complexity of managing a huge number of situations or looking for suitable services while respecting time requirements. Our goal is to provide an agile semantic approach in the context of smart- * (home, health, cities, vehicle, etc.). This agile approach emphasizes the efficient management of daily and urgent situations via mobile applications and the automatic generation of reconfiguration actions according to the user's profile, his needs and current situations. First, we have proposed a multi-layered ontology for composite situation model called Multi-OCSM to formally model smart services, user's domains, context, situation and priority-based situation. It also allows automatic reconfiguration of distributed pervasive applications with better autonomy. Afterward, we have proposed a new architecture to specify composition operators such as parallel, sequence, alternative, and recurrence among component-based situations. From the specification of component-based situations, we generate the orchestration of the services via semantic rules and evolved moving scenarios. Then, we have offered a new semantic approach based on microservices which allows the exploitation of the Multi-OCSM ontology for the management and identification of composite situations in real time. A context-aware ontology-based recommendation system for situation rules enrichment and adaptation is also proposed. Finally, to validate the efficiency of our approach, we have proposed a prototype for situation management with several experiments. The experimental results show that the proposed approach is efficient in terms of managing and identifying composite situations where the use of factored microservices improves the execution time of the system.

Keywords: ontologies, heterogeneous connected objects, pervasive environments, contextual and urgent situations, Multi-OCSM, automatic reconfiguration.

Résumé

Avec l'apparition abondante d'objets connectés dans les environnements pervasif, les utilisateurs ont tendance à récupérer les informations contextuelles directement, depuis l'environnement physique, pour mieux répondre à leurs besoins et gérer idéalement leurs situations. Cependant, avec la croissance rapide de la taille des données et du nombre d'appareils dans les environnements pervasif, l'identification d'un grand nombre de situations d'utilisateurs devient une tâche difficile. En effet, l'identification de situations dans des environnements pervasif implique une interprétation sémantique des données brutes contextuelles fournies par divers objets hétérogènes, ce qui nécessite de nouvelles solutions agiles dans la gestion des situations composites. Aucune des approches existantes n'est en mesure de gérer et d'identifier efficacement les situations d'urgence composites centrées sur l'utilisateur. Elles manquent de flexibilité pour évoluer et s'adapter en fonction du contexte actuel de l'utilisateur et des exigences de l'environnement. Les systèmes actuels perdent en performance face à la complexité de la gestion d'un grand nombre de situations ou de la recherche de services adaptés tout en respectant les délais. Notre objectif est de fournir une approche sémantique agile dans le contexte du smart-* (maison, santé, etc.). Cette approche agile met l'accent sur la gestion efficace des situations quotidiennes et urgentes via des applications mobiles et la génération automatiquement des actions de reconfigurations en fonction du profil de l'utilisateur, de ses besoins et situations actuels. Tout d'abord, nous avons proposé un modèle d'ontologie multicouche appelé Multi-OCSM pour modéliser formellement des services intelligents des domaines des utilisateurs, des situations et des priorités entre situations. Il permet également une reconfiguration automatique des applications pervasive avec une meilleure autonomie. Ensuite, nous avons proposé une nouvelle architecture pour spécifier des opérateurs de composition tels que parallèle, séquence, alternative et récurrence entre des situations basées composants. A partir de la spécification de situations à base de composants, nous générons l'orchestration des services à partir des règles sémantiques et scénarios de mouvements. Nous avons aussi proposé une nouvelle approche basée sur les microservices pour la gestion et l'identification des situations composites en temps réel. Un système de recommandation basé sur une ontologie sensible au contexte pour l'enrichissement et l'adaptation des règles de situation est également proposé. Enfin, pour valider l'efficacité de notre approche, nous proposons un prototype de gestion des situations avec plusieurs expérimentations. Les résultats expérimentaux montrent que l'approche proposée est efficace en termes de gestion et d'identification des situations composites où l'utilisation des microservices factorisés améliore le temps d'exécution du système.

Keywords : ontologies, objets connectés hétérogènes, environnements omniprésents, situations contextuelles et urgentes, Multi-OCSM, reconfiguration automatique.

Preface

The works presented in this thesis have been published in the following international conferences and journals:

National conferences

- *A. Lakehal.* Approche sémantique de génération automatique d'applications agiles en environnement pervasif. Doctoral Day, First Edition, UFAS-1 University, Auditorium Mouloud Kacem Nait Belkacem-El Bez, 18 May 2017.

International conferences

- *A. Lakehal.* Ontology based Context Model for automatic generation of Agile Applications in Pervasive Environments. First Symposium International of Doctorate in Computer Science Tuniso-Algériens (SDTA). Tunisie, Hammamet, 24-26 Avril, 2017.
- *A. Lakehal, A. Alti, S. Laborie, P. Roose.* Ontolgy-based Context-aware Recommandation Approach for Dynamic Situations Enrichment. 2018 IEEE 13th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), Zaragoza, Spain, 6-8 November, 2018, (SJR 0.15) (Core: B). DOI: 10.1109/SMAP.2018.8501880
- *A. Lakehal, A. Alti, P. Roose.* A Semantic Event-Based Framework for Complex Situations Modeling and Identification in Smart Environments. 7th International Conference on Computing and Informatics 2019 (ICOCI 2019), Malaysia, 27-29 March, 2019 (Core: C).

International journals

- *A. Lakehal, A. Alti, P. Roose.* A Semantic Event-Based Framework for Complex Situations Modeling and Identification in Smart Environments. International Journal of Advanced Computer Research, Vol 9(43) – DOI: 10.19101/IJACR.PID33 – 2019 (Q4 IF 0.85).
- *A. Lakehal, A. Alti, S. Laborie, P. Roose.* A Semantic Agile Approach for Reconfigurable Distributed Applications in Smart Environments. International Journal of Ambient Computing and Intelligence (IJACI): Volume 11, Issue 2, pp. 48-67, 2020. DOI: 10.4018/IJAC. (Q2 IF 0.527).
- *A. Lakehal, A. Alti, S. Laborie, P. Roose.* Semantic-Based Automatic Generation of Reconfigurable Distributed Mobile Applications in Pervasive environments. Accepted and will be published in journal Transactions on Large-Scale Data and Knowledge-Centered Systems (tldks), Springer-Verlag, Vol.44 (2020).
- *A. Lakehal, A. Alti, S. Laborie, P. Roose.* Real-Time Ontology-Based Context-Aware Situation Reasoning Framework in Pervasive Computing. Submitted to Multimedia Tools and Applications, Springer, 2020.

Table of contents

List of Figures

List of Tables

List of Abbreviations

ADL	Activities of Daily Living
ADSL	Asymmetric Digital Subscriber Line
AGGIR	Autonomy Gerontology ISO-Resources Groupe
ANFIS	Adaptive Neuro-Fuzzy Inference System
API	Application Programming Interface
CF	Collaborative Filtering
CFG	Context-Free Grammar
CNN	Convolutional Neural Network
CoMeR	Context-aware Media Recommendation Platform
COSAR	Combined Ontological/Statistical Activity Recognition
CPU	Central Processing Unit
CT-PCA	Coordinate Transformation and Principal Component Analysis
DL	Description Logic
DSL	Domain Specific Language
ECA	Event Condition Action
EM	Expectation-Maximization
FOAF	Friend of a Friend
GPS	Global Positioning System
HMM	Hidden Markov Model
HSSN	Hybrid Semantic Sensor Network
ID	Identifier
IoT	Internet of Things
IP	Internet Protocol
KCAR	Knowledge-driven approach for Concurrent Activity Recognition
MSSN-Onto	Multimedia Semantic Sensor Network Ontology
Multi-OCSM	Multi-layered Ontology-based Composite Situation Model
NGCF	Neural Graph Collaborative Filtering
OISVM	Online Independent Support Vector Machine
OWL	Ontology Web Language

PDA	Personal Digital Assistant
POI	Point-of-Interest
PRS	Publication Recommendation System
QoS	Quality of Service
QSTR	Qualitative Spatio-Temporal Reasoning
RAM	Random Access Memory
RDF	Resource Description Framework
RFID	Radio Frequency Identification
RIMER	Rule-based Interface Methodology using the Evidential Reasoning
SAREF	Smart Application Reference
SNS	Social Networks Systems
SOA	Service-Oriented Architecture
SOSA	Sensor Observation Sample Actuator
SPARQL	Protocol and RDF Query Language
SSN	Semantic Sensor Network
SVM	Support Vector Machine
SWRL	Semantic Web Rule Language
TF-IDF	Term Frequency and Inverse Document Frequency
Wi-Fi	Wireless Fidelity
WSSQ	Weighted Set Similarity Query

General Introduction

1. Thesis context

Within the era of pervasive computing and smart-* (health, home, cities, car, office, etc.), it is not surprising that many users own more than one mobile device (e.g., smartphone, smartwatch, and headphone) to deal with multiple services simultaneously (6.58 devices per person in 2020 [1]). Nowadays, pervasive computing integrates sensors and computational entities that can communicate and interact transparently both with users and the environment in which they operate [2]. The integration of connected objects in our daily lives allows us to retrieve contextual information directly from the user's physical environment such as location, brightness, neighborhood, and interactions. This contextual information can be used to identify different situations depending on the user's environment, his preferences and devices capabilities. In fact, traditional applications were unaware of context information in which they neglect information related to user's environment such as his location, time, accessible services, and nearby devices. Neglecting context information may reduce the quality of reconfigurable applications that provide relevant services according to the user's current state. Therefore, several applications have adopted the use of context to improve the relevancy of provided services. Recently, context and situation have been widely used in pervasive computing systems to describe entities that interact with the application. More specifically, context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [3].

Pervasive environments incorporate a wide range of sensors to collect context data from the physical world. Sensors can be embedded in specific devices or integrated in human bodies. In other words, sensing technologies have made great progress in collecting various types of context data. In fact, context-aware applications relay on context data acquired from sensors to make adaptation and provide services at run-time according to the user's current state. Nevertheless, considering the entire context in pervasive applications to provide relevant services needs a powerful computational capability which is not always available. Therefore, considering a meaningful part of context that represents only expressive information namely situation, enhances the system's performance and reduces the computational time. This raises a significant challenge in situation modeling and identification. Generally, situation is the set of things (events/context) that happens at a given location (where?) and at a given time (when?), and the conditions (clauses) that can verify those events, thus identifying the situation.

With the growing number of connected objects, managing heterogeneous sensors and devices raises a significant challenge regarding continuous context monitoring and services (re-) deployment. The current vision of pervasive computing aims to develop an environment that incorporates flexibly heterogeneous communication and computing capabilities. Therefore, the use of semantic models and middlewares is mandatory. Thus, several standard ontologies of context and connected objects modeling have been proposed [4] [5] [6] to hide the heterogeneity of communications protocols and software features through semantic abstraction models. Besides, several middlewares have been also proposed [7] [8] [9] [10] [11] to act as a mid-level between sensors and pervasive applications. Further, they ensure the monitoring of context data from the involved sensors and devices regardless of hardware and software features.

This thesis focuses on handling distributed pervasive applications through the modeling and management perspective in the field of smart-*. We are interested in the management of context information and the identification of situations in smart environments by handling situations in a specific time, starting by serving the most urgent situations. Building a context-aware pervasive application based on situation management is becoming the most challenging task in pervasive environments. Therefore, we need to construct a flexible and parallel architecture that handles rapidly parallel inputs (i.e., context data) and outputs (i.e., action services). However, pervasive applications still have issues to be addressed such as scalability and flexibility to support dynamic evolutions of context changes.

2. Challenges

In this context, it is necessary to have a flexible and agile approach for situation management and identification able to understand the user's daily life activities regardless of his context changes. Nevertheless, before we propose an agile architecture, we should specify problems and challenges facing context-aware pervasive applications. In fact, context-aware pervasive applications have dynamic environment in which context changes continuously. This dynamicity raises the issue to provide reconfigurable applications regardless of context changes.

The heterogeneity of sensors and devices raises the issue of unifying monitored context data through a semantic model. Different approaches in the area of context modeling have been proposed [3] [12] [13] [14]. However, they only cover a specific domain such as health domain. We seek for proposing a generic model for smart-* domains that models and unifies the heterogeneity of monitored context data.

Situation modeling raises another challenge in representing user's daily activities. In real-life, users have repetitive and parallel activities (i.e., composite situations) that they can perform on a regular time. However, existing works [4] [15] [16] [17] lack flexible models that enabling the representation of parallel composite situations using composition operators. We seek for designing a flexible semantic model for covering all composite situations to describe real-world activities. The context-awareness and the

adaptation of personalized mobile applications are considered as a difficult task due to the dynamic evolution of user's preferences and the heterogeneous nature of execution contexts. This emphasizes the need to design a semantic context-aware framework that targets the management of user's constraints, his preferences, and his usage context.

Several middlewares based on component architecture have been built for reconfigurable pervasive applications [8] [9] [11] [18] . Nevertheless, due to the high connectivity of multiple sensors and devices in reconfigurable pervasive applications, performing dynamic reconfiguration (i.e., adding/removing/updating services) on the fly may cost a high adaptation time which significantly reduces system performance. Moreover, the fast evolution of context generates many reconfiguration scripts to adapt to context changes. Therefore, a loosely coupled architecture is needed to reduce the reconfiguration time.

The big amount of context data collected and processed by pervasive systems may take significant computational time. In fact, real-time context processing is an essential factor in pervasive computing systems. Therefore, systems must provide relevant services according to identified situations in the required time, especially when it comes to urgent situations (e.g., heart-attack situation). Moreover, most of the interactions in pervasive applications are expected to be executed in real-time. Nevertheless, with the growth of user's expectation and the huge number of users' situations, it is hard to respond to all users' needs in real-time. Traditional centralized computing systems cannot achieve this goal within a reasonable time. Existing middlewares [8] [9] [11] [18] lack efficient methods of managing and identifying composite situations in a parallel way. Thus, a flexible and parallel framework is needed to handle rapidly parallel incoming events and urgent situations. Besides, to accelerate the execution of urgent situations, we seek for proposing a prioritized mechanism to improve situation's identification process and speeding up queued urgent situations that have high priority.

3. Objectives

The main objective of this thesis is to generate a distributed context-aware mobile application in the field of smart-* (Home, City, Health, Tourism, etc.) in order to exploit the variety of interactive services capabilities according to the user's current situation and usage context. We take advantage of semantic models and IoT infrastructures based on existing middlewares to propose a new intelligent semantic composite situation management approach in high reconfigurable smart environments. Overall, the objective of this proposal is to provide a user-centered situation-based mobile application able to manage everyday situations and react to them by providing relevant services to the user's needs. Our aim is to improve the application flexibility and reusability at design and run time.

Our proposal is dedicated to assisting users in their everyday activities through widespread devices. It provides (re) deployment reconfiguration scripts including adding, updating, removing services to

dynamically adapt to user's context changes. Therefore, we aim at proposing an efficient approach for monitoring context changes from deployed sensors.

Our aim is to design a modular and hierarchical architecture that ensures a good level of reusability and flexibility. Modular architecture allows to reason on composite situations using deployable parallel components. The reasoning module may change the design model of the application to react to context changes by adding, updating, removing deployable components. The objective of this architecture is to design a flexible loosely coupled application that can resist for context changes.

4. Contributions

Our research serves to manage context-aware distributed mobile applications for smart-* domain. Therefore, we aim at developing a new framework for composite situation management that adopts semantic modeling based on microservice implementation to identify user's daily life activities. The main contributions of this research are:

- **Multi-OCSM (Multi-layered Ontology-based Composite Situation Model)** [19], a novel user-centric composite situation ontology based on multi-layered context-aware semantic services. It describes both simple/composite situation and its semantic description including new composition operators (parallel, sequence, recurrence and alternative), prioritized situations, and reconfiguration action services in order to ensure service continuity and immediate response time. This ontology provides a full modeling of heterogeneous smart objects and context data. Also, it provides a flexible and loosely coupled hierarchical model that can adapt rapidly for real-time context changes.
- **A novel generic of composite situations parallel identification framework for smart environments** [19], which allows the parallel identification of user's situations in several smart domains. The main features of our identification framework are: (i) ensuring monitoring of parallel streaming context data, (ii) prepossessing of context data, (iii) analyzing and reasoning on context changes using our Multi-OCSM ontology to identify context situation changes.
- **An extended Multi-OCSM ontology-based context-aware recommendation approach for dynamic situations enrichment** [20], that enriches automatically profile's situation rules, in order to provide an auto-complete of user's daily life requirement in different domains (shopping, work, travel, etc.). This approach exploits other users' experiences by offering to the users a high level of comfort and a better-customized user experience. Our objective is to automate the injection of new situation rules using a new

concept called role-situation-services, which aims to enrich dynamically mobile application with rich context rules.

- Development of **a prototype for composite situations parallel identification based on the Multi-OCSM reasoning** for services reconfiguration and adaptation. The prototype enables the identification of any abnormal situations and prioritizes the processing of urgent situations (e.g., health-heart-situation). Also, it is able to manage and identify composite situations under different user's context changes and triggers relevant action services for identified situations. The identification process relies on Java 8 concurrency APIs [21] to manage multithreading tasks that process parallel incoming events to improve the parallel identification of composite situations.

5. Motivating case study: multi-smart domains

We consider here a real case study that involves assisting users in several smart environments (smart-home, smart-car, and smart-office). These smart environments can react to users according to daily-life events that occur in different domains (e.g., health, social and professional). Figure 1 illustrates three smart environments (smart-home, smart-car and smart-office) with both wearable smart objects and mobile devices. The application's purpose is to manage all the context data of the health domain and daily activities in an efficient way by enhancing the monitoring and reporting of critical health conditions.

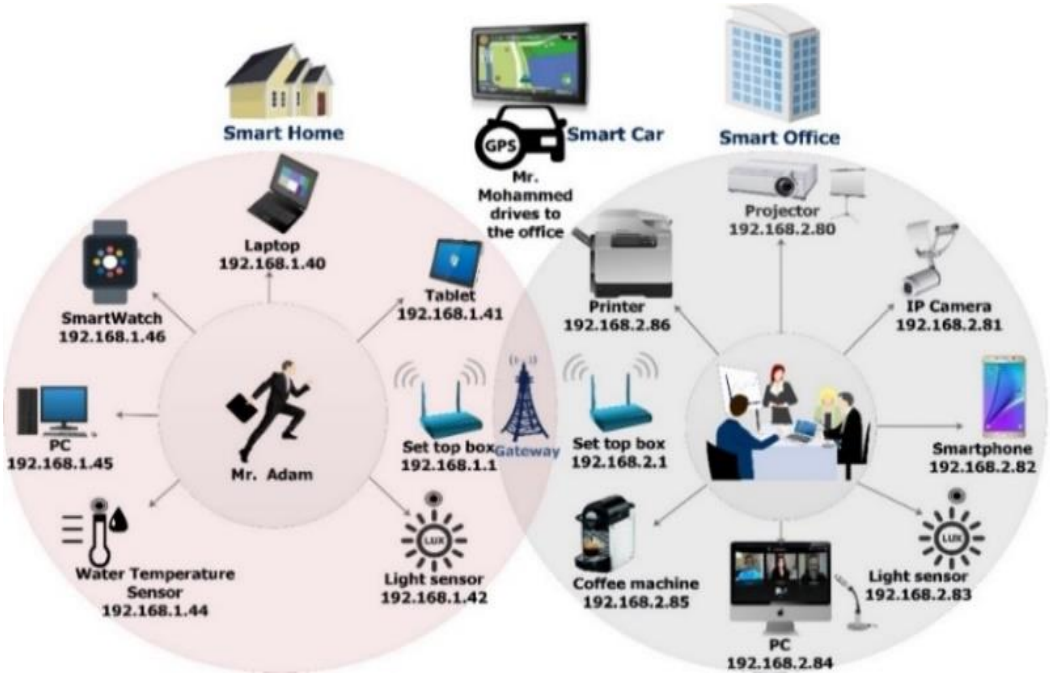


Figure 1: Smart environments.

As we can see in figure 1, there are various smart objects. Each smart object serves to monitor information such as a motion sensor to monitor the movement, a smart camera to monitor the activity (e.g., watches TV, drinks a coffee or checks emails), a smartwatch to monitor the user's locations and other sensors to

detect other contextual information (temperature and humidity level, states of doors, lights, user glucose and user movement states). In our case study, the smart home acts as a pervasive environment that is composed of many rooms: kitchen, living room, bathroom, study, garage and two bedrooms (bedroom#1 and bedroom#2) (see figure 2).

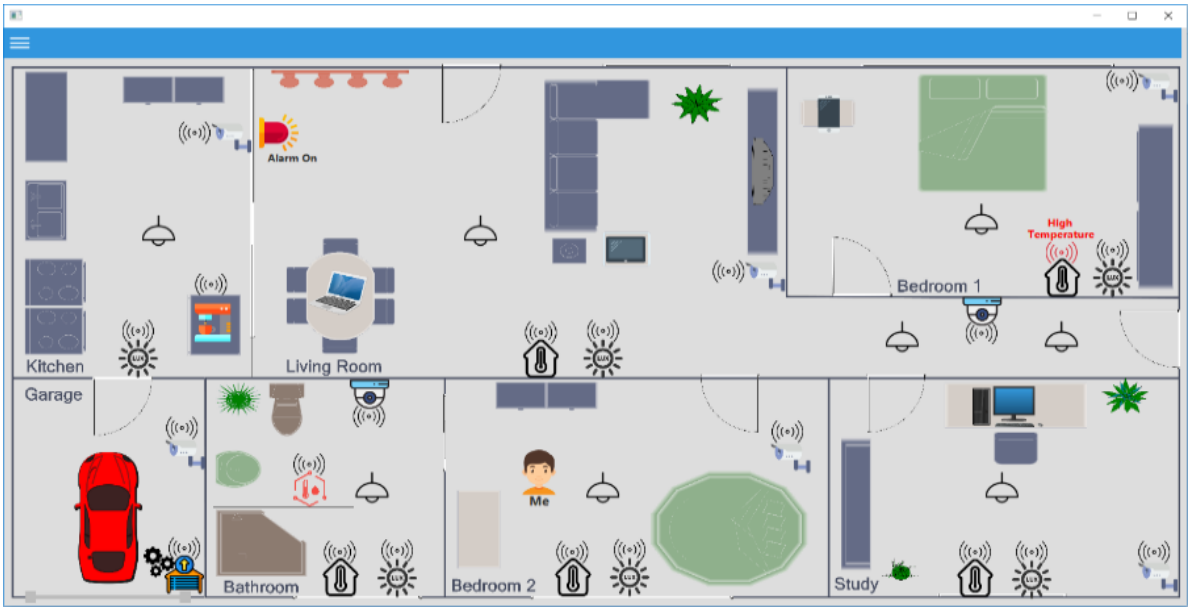


Figure 2: An overview of a smart home.

Each room is equipped with smart objects. Table 1 details a list of available smart objects and devices in the smart home. The context information is collected by the set-top-box (e.g., position, time and video). This contextual information is useful to identify a set of users' situations.

Table 1: Partitioning of different smart objects in the smart home.

Smart home-rooms	Sensors	Actuators and devices
Bedroom#1 Bedroom#2	Ambient temperature sensor, light sensor, state of door sensor, IP camera.	Light switch actuator, door locking.
Living room	Ambient temperature sensor, light sensor, motion sensor, IP camera.	Light switch actuator. Laptop, Tablet, Smart TV.
Kitchen	Light sensor, IP camera, coffee machine sensor.	Light switch actuator, smart machine coffee, Smart TV.
Bathroom	Ambient temperature sensor, humidity sensor, water temperature sensor, motion sensor, Light sensor.	Light switch actuator, Smart mirror.
Study	Ambient temperature sensor, light sensor, IP camera	Light switch actuator, PC
Garage	State of garage door, IP camera.	Garage door locking.

Now, let us suggest this scenario: we are interested in the daily life of a user (Mohammed). He lives in the smart home described above. He is a regular worker of a bank office. He has specific needs according to his current context, profile, and preferences (preferred language, media preferences, transport preferences, text size, etc.). He is a diabetic, so the system must continuously monitor his health status to identify any urgent situations (e.g., high-level glucose or low-level glucose). He is equipped with wearable devices (glucose meter and smartwatch). He has an agenda, which accurately describes his planned activities. These activities can be performed in a sequential and/or parallel order. Mohammed's agenda can be used to identify various situations during the morning period, from waking up until going to the office. Below, we give a short detail about his daily-life activities.

In the morning, the system checks his personal Google agenda to find out about his working days, then it sets the alarm application on his smartphone. When the time comes, the system increases gradually and automatically the lighting of the bedroom. The camera in the bedroom can automatically recognize if he wakes up or not. When he wakes up, the system checks Mohammed's next scheduled task, which is: (takes a shower). Therefore, when he leaves the bedroom, the system adjusts the water temperature and displays it on the smart mirror of the bathroom. When he leaves the bathroom, he heads towards the kitchen; the system sends an order to the smart coffee machine to prepare the coffee. After that, he takes his coffee and moves towards the study where he drinks his coffee quietly. The system deploys email application on his desktop PC with all tools that he can find useful in his working field (e.g., user profile) and deploys morning news on his Smartphone. Mohammed's smartwatch regularly sends GPS coordinates to the system to locate his precise location. When it is time to go to work, the system migrates the morning news service on his smart car dashboard to ensure the continuity of service. When he enters the garage, the system deploys the Google map and selects the fastest route to the office. He takes his car and goes to work. The garage is automatically opening and closing according to GPS coordinates of his car. When he leaves his smart home (i.e., the situation that nobody is at home is identified) the system triggers the intrusion alarm application and fire alarm application for his home security (for each anomaly, Mohammed can be notified by an SMS). For any unexpected traffic congestion situation on the current route, the system must be notified automatically and must react dynamically to select a new better road towards the office. At this time, a new incoming planned event extracted from Mohammed's agenda (a conference meeting). The system identifies this new event as a new situation. As he is late, an audio conference is automatically deployed on his Smartphone. When he enters the office, the audio conference is dynamically migrated to its desktop PC and the video functionality is added to continue the discussion. The conference application is migrated and deployed simultaneously on each participant's mobile device.

This case study illustrates the variety of sensors (simple or multimedia) used to interpret and detect simultaneous events and various users' situations. This scenario raises several needs and challenges:

- **Need 1:** Ensuring extensibility, unambiguous and semantic meaning of exchanged context data and events.
- **Need 2:** Representing and handling composite situations using composition operators in real-time.
- **Need 3:** Ensuring flexibility and continuity of services regardless of user's location and context.
- **Need 4:** Enriching and suggesting relevant situations for users.
- **Challenge 1:** How to define and manage simultaneously semantic context monitoring and processing? How to interpret parallel incoming events? For instance, a presence detector (motion sensor) can provide Boolean values as input (True/False), others may give you a vector of displacement (GPS), it is the same event, but the interpretation is very different.
- **Challenge 2:** What are the methods that can be exploited for inferring and managing both daily and urgent situations in real-time and effectively?
- **Challenge 3:** What are the strategies that can be used to replace a smart object by another semantically equivalent in order to ensure the continuity of service?
- **Challenge 4:** How to inject new situations to make the application more dynamic and complete using external situation rules and various context sources?

6. Organization of the thesis

This thesis has been organized into seven chapters. The first two chapters are devoted to a literature review related to our research, while Chapters 3, 4, 5, 6, and 7 present our contributions. The resume of each chapter is provided below as follows:

- **Chapter 1** gives an overview of pervasive computing systems. It also includes the application domains, features and challenges of pervasive environments as well as both context and situation concepts. Further, it gives an overview of semantic technologies related to standard ontologies in pervasive computing that help us to design specific domain knowledge.
- **Chapter 2** provides a literature review of the research area. It covers the existing literature on situations modeling and reasoning approaches and platforms. Moreover, this chapter gives comparisons between different approaches to understand both advantages and limitations of each solution in order to tackle research problems.
- **Chapter 3** presents an overview of our proposed contributions. Further, it introduces the general architecture of the parallel composite situation identification framework as well as its functional model at design and run time.

- **Chapter 4** presents the Multi-OCSM (Multi-layered Ontology-based Composite Situation Model) based on standard models such as Semantic Sensor Network (SSN) [49], DOLCE Ultra-Light Ontology (DUL) [52], and OWL Service (OWL-S) ontologies [110]. Moreover, it presents a novel semantic microservices-based user-centric prioritized composite situations modeling and reasoning for pervasive applications.
- **Chapter 5** presents in detail the parallel identification process of semantic-based composite situations over heterogeneous smart objects.
- **Chapter 6** presents an extension of the Multi-OCSM ontology to classify new situation rules. Further, it presents a new recommendation system based on Multi-OCSM ontology for injecting new situation rules into the user's agenda.
- **Chapter 7** presents the implementation of our framework prototype for parallel identification of composite situations. Further, it shows comparisons between parallel reasoning approach results and the sequential reasoning approach results in order to prove the effectiveness of our proposed approach.

This thesis is concluded by stating the research summary along with research contributions, limitations, then suggestions and future directions in order to achieve further research studies.

Part 1: Background and Related Work

Chapter 1

Context-aware and Situations in Pervasive Computing: Background and Standard Ontologies

Summary

1.1 Introduction

Pervasive computing incarnates a new era of computers that are perfectly integrating into users' daily life activities, and responding to their needs after processing a set of information provided by sensors deployed in various environments. In literature, the term pervasive signifies *"existing everywhere"*. Pervasive computing is the embedding sensing and computational entities that can communicate and interact transparently both with users and with the environment in which they operate [2]. Therefore, Systems can customize services to users in a context-aware manner based on these pervasive environments with the help of these entities. Currently, pervasive applications have many potential application domains, such as intelligent spaces (e.g. smart home, smart city, smart car, etc.), healthcare, transport, etc.

With the growing number of connected devices, sensors are deployed everywhere and embedded on any device or human bodies. Their ultimate objective is collecting different kind of context data such as user's location, ambient temperature, humidity, biomedical information. Pervasive context-aware applications use this data to provide customized services to users. Moreover, with the huge number of sensor data, contextual information exhibits high complexity causes by the inter-dependency links between sensor data. Further, this inter-dependency can raise other challenges related to reconfigurable pervasive applications when adding/removing/updating service components, resulting in low flexibility during context changes (e.g., appearance of new sensors/devices in the network, change of user needs or interaction modalities, etc.).

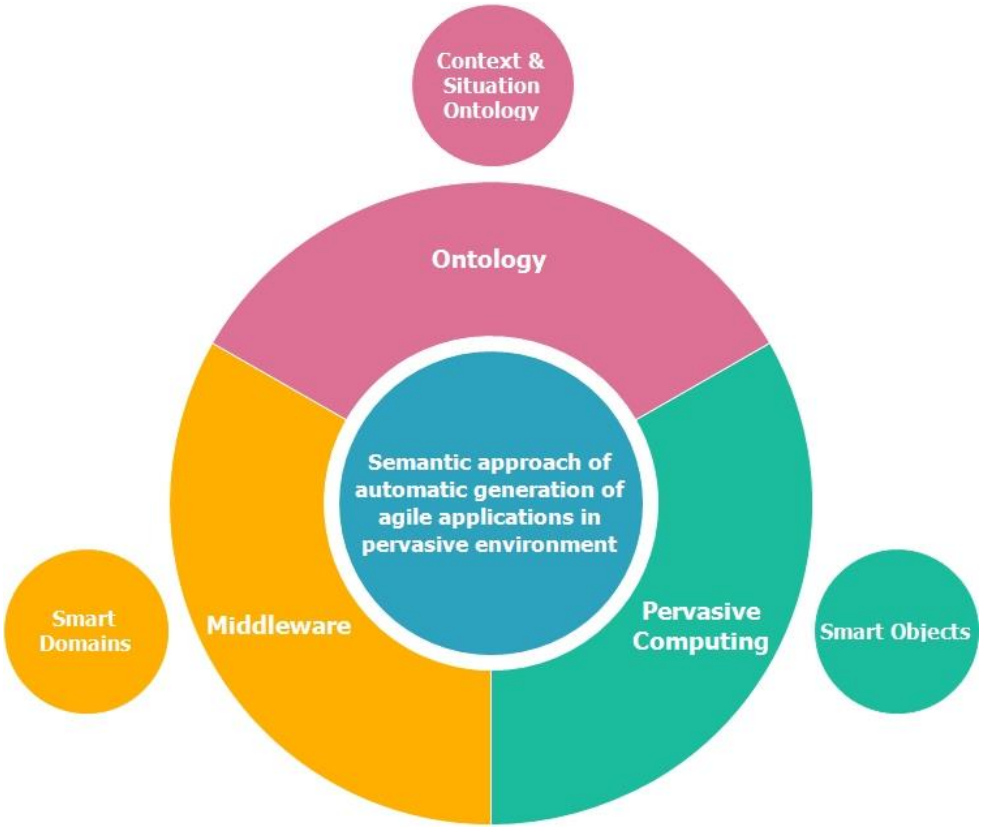


Figure 1.1: Thesis context.

In this chapter, we describe some background and preliminaries necessary to understanding the related works and our contributions detailed in the rest of this thesis accomplished in the domain of pervasive environments. First of all, it is important to underline that our thesis is located at the intersection of three main areas of research which are: Pervasive Computing, Middlewares, Context and User's Situations based ontologies as shown in Figure 1.1. First, we start by defining and setting up pervasive computing and give an overview of different challenges in this research area. Then we present standard definitions of context-awareness and situation-awareness. After that, we present the relationship between context-aware and situation in pervasive computing. Then, we present some standard ontologies of context modeling and its related heterogeneous connected objects, as well as semantic context representation in heterogeneous environments. Finally, we present some existing standard middlewares for pervasive application components and context-aware situation models in order to select a middleware solution.

1.2 Pervasive computing

1.2.1 Evolution of technology: from ubiquitous to pervasive

In 1991, Mark Weiser [22] coined the term ubiquitous computing. He defines the ubiquitous computing as a new vision of computing technology where data is everywhere and great number of devices is embedded unobtrusively in several smart spaces. This vision is summarized by his famous opening statement: "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*" Ubiquitous computing (or "ubicomp") means that computers are embedded, invisible and provide transparent services everywhere and every time. More specifically, all computers communicate and cooperate in order to achieve the sole purpose of satisfying our humanly needs. However, in 1991 the internet has just started to catch the new era. In this decade radio frequency and infrared were the most popular technologies used for devices communications. Therefore, Mark Weiser envisioned radio frequency and infrared to link computers to cooperate (see figure 1.2).

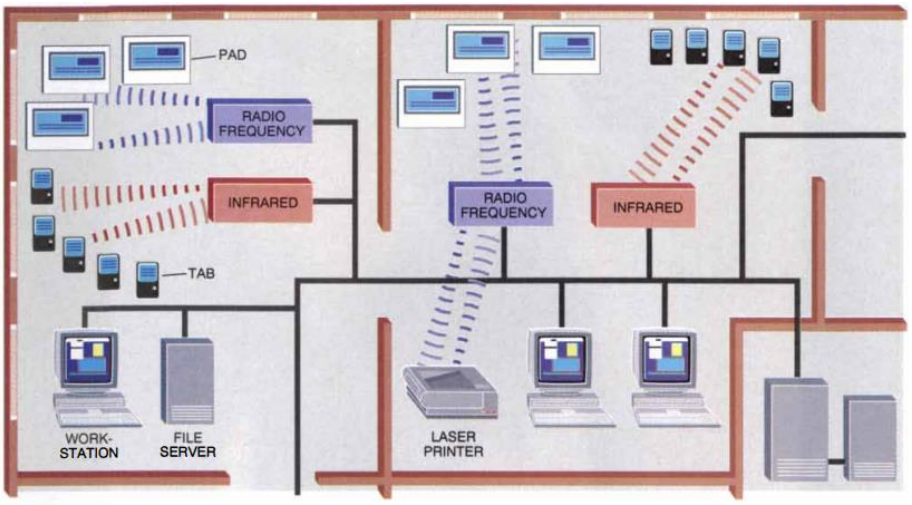


Figure 1.2: Vision of Mark Weiser, wired and wireless networks link computers to cooperate [22].

In the last decade of the 90s, the Internet was at the top of communication technologies and became mandatory support to link and connect different devices. In 1999, Kevin Ashton [23] takes benefit from the Internet where he envisioned that any things can be connected through Internet (e.g. devices, computers, laptops, PDA, vehicles, TV, watch, etc.) namely "Internet of Things" (IoT). He was the first person who coined this term. In fact, many overviews of IoT including Kevin Ashton rely on the definition given by Mark Weiser. According to Friedemann et *al.* [24] they define the IoT as: "*The Internet of Things represents a vision in which the Internet extends into the real world embracing everyday objects. Physical items are no longer disconnected from the virtual world, but can be controlled remotely and can act as physical access points to Internet services. An Internet of Things makes computing truly ubiquitous*", this concept has been initially put forward by Mark Weiser in the early of 90s.

Naturally, terms and concepts can change and evolve over time according to incoming new technologies. New terms can be based beforehand on old terms that may have same definitions with some improvement to cope with new technologies. Historically, pervasive computing has its roots in the definition of ubiquitous computing coined by Mark Weiser. The term pervasive computing has a very similar meaning to ubiquitous computing. In 2002, Mark Weiser [25] introduced a pervasive computing environment as connected objects have processing power with wireless or wired connections to a global network in which these objects can be spread everywhere in the user environment. The vision of pervasive computing characterized by the development of an environment that incorporates heterogeneous communication and computing capabilities with smooth integration of human interventions. This vision and aspirations to develop an entire pervasive system were limited due to the required hardware. After the tremendous advancement in hardware and software technologies that has known currently, the required elements and technologies in pervasive systems have become available. This new generation of hardware can be wearable, implanted in the human body or embedded in the environment, mobile, wireless, and intelligent with sensing and control capabilities. Therefore, we are living in a new era where we can take benefits from these new generations of hardware and software technologies (e.g., embedded sensors in human body, small and sophisticated sensors, actuators, smart watch, smart TV, mobile devices with different protocol of communication ad-hoc, Bluetooth, Wi-Fi, GPS, 4G, ADSL) to pursue the vision stated by Mark Weiser. Figure 1.3 illustrates the evolution of the new era of computers for the 21st century.

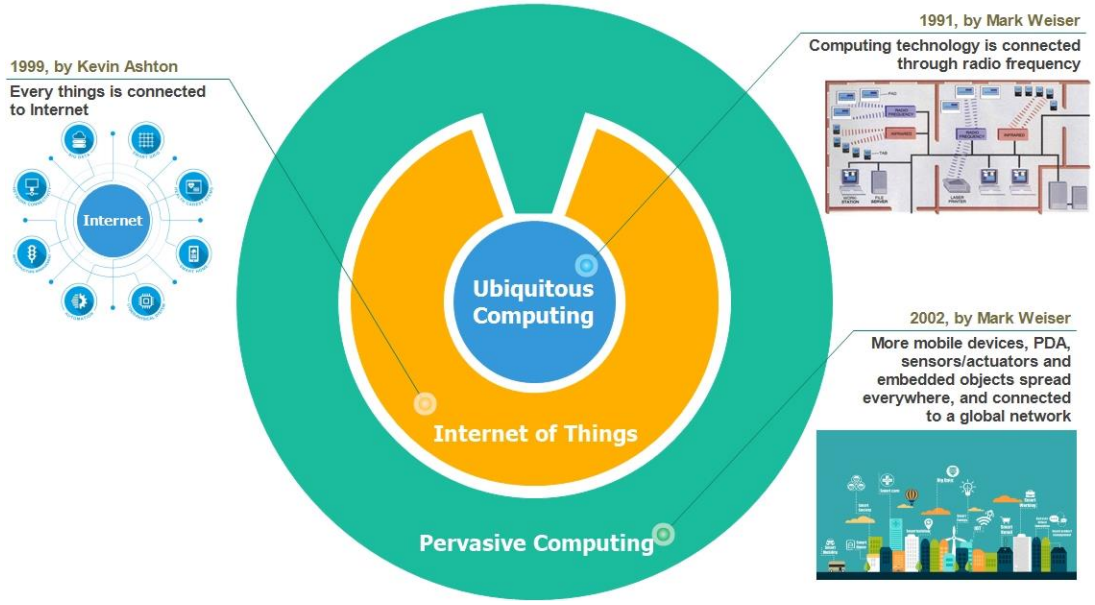


Figure 1.3: The evolution of technologies, ubiquitous computing, IoT and pervasive computing.

Ubiquitous/Pervasive Computing vs IoT

Ubiquitous/pervasive computing and IoT are terms that can be used interchangeably. Pervasive computing and IoT face similar problems and challenges. Both concepts support similar use cases such as smart homes, cities, healthcare, transport, agriculture. However, there is a slight difference between them. Pervasive computing focuses more on computational capability in linked objects (smart TVs, smart cars, smart homes, wearable devices and implanted sensors in the human body). On the other hand, IoT focuses to have these objects connected to the Internet.

1.2.2 Smart-* domains applications in pervasive environments

Pervasive environments can have a significant impact on our daily lives in many domains. In fact, pervasive computing provides an environment that can sense the localization of a person in a given smart area to react intelligently to its needs. In other words, intelligent systems in pervasive environments provide useful information to the user based on his location and other user-centered context information. Further, new infrastructures including smart-* domains (e.g., smart-home, smart-city, smart-office, smart-car, smart-shopping, etc.) that have been deployed in different spaces, facilitate the creation of new applications in several business domains. These applications will automate different tasks in our life such as make our home safe, increase the productivity of our work, and customize our shopping experiences. Therefore, pervasive systems will intelligently execute these tasks based on contextual information collected from these smart-* domains in order to improve the use of resources such as water and electricity. Figure 1.4 shows some smart-* application domains. Common and specific concepts of different smart domains (Appendix C) will be analyzed in chapter 3.

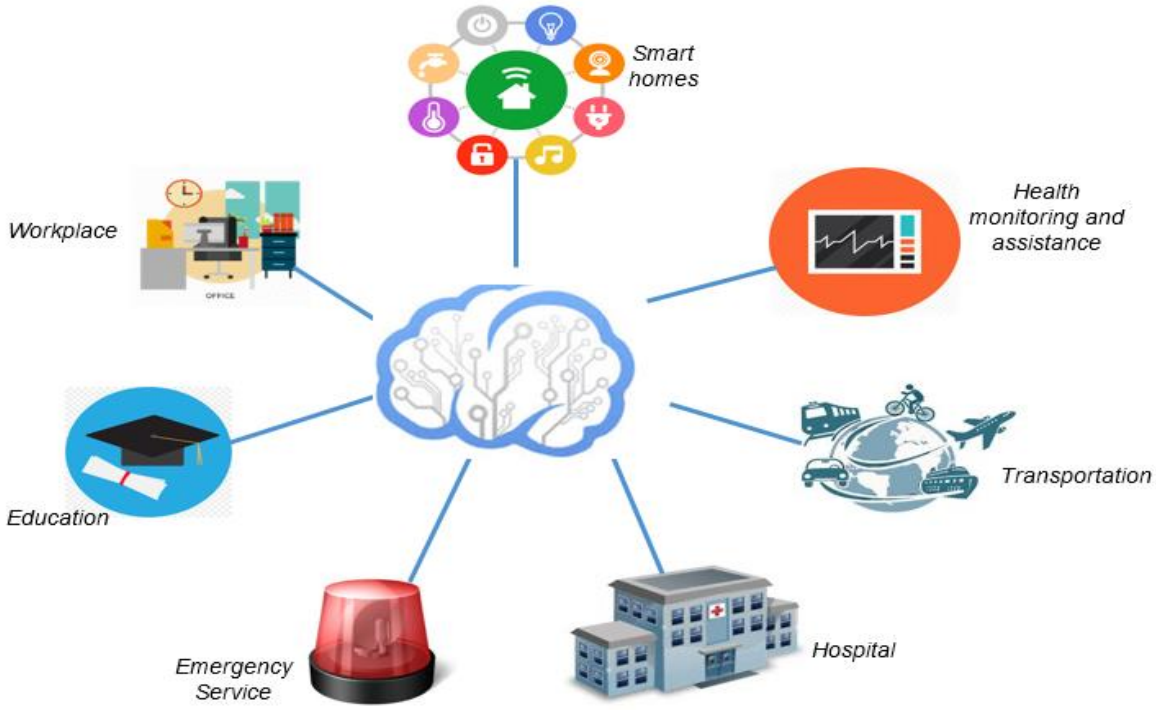


Figure 1.4: Application domains of pervasive computing environments.

For instance, pervasive computing systems may support many potentials use cases including smart-health, smart-transport, smart-home, and smart-office as follows:

- **Smart Health Domain**

Since humans can be implanted with sophisticated sensors, pervasive computing has been integrated widely in the health domain. The main purpose of smart-health systems is to ensure that patients or our loved ones are safe and healthy while performing their daily life activities. In fact, smart-health systems rely on pervasive environments to recognize human activities, follow their diet, and detect any abnormal changes or anomalies. In other words, pervasive applications in the health domain follow patients' health state by collecting and analyzing their context information. The context information of patients can be monitored either by implanted sensors in human body (e.g., glucose sensor, blood pressure sensor, heart rate sensor, body temperature sensor, respiratory rate, etc.) or by embedded sensors in pervasive environment (e.g., camera, movement detector, etc.). The system analysis this context information and identifies abnormal situations (e.g., high glucose level) to intervene automatically (e.g., inject insulin). For instance, in case of critical anomalies (e.g., like heart-attacks), the system triggers an immediate response to the closest urban gateway and requests an urgent medical intervention where the coordinates of the victim are determined and sent. After that, the system searches for the closest available ambulance and requests the nearest hospital for immediate preparation for hosting the patient. Therefore, pervasive environment capabilities can be used in healthcare applications to enhance intervention services while reducing the heavy load of nurses in hospitals.

- **Smart Transport Domain**

Currently, pervasive computing has found its way to integrate into the transport domain. Further, stations, buses, trains, airplanes and cars can be equipped with several sensors (e.g., camera/for image processing and vehicle identification, GPS sensors/for vehicle tracking, distance sensor/for preventing collision, speed sensor, temperature sensor, etc.) that can provide pervasive system with context information. Pervasive system analyzes this context information to make predictions, preventive actions, and plans for the future (e.g., cancellation of flights in advance due to the weather conditions, change trajectory in the satnav due to an accident occurred in the previously selected trajectory).

We spend a significant part of our lives traveling to different destinations in various ways. The user's experience during traveling can be customized based on his preferences to make transportation smoother, more efficient, and safer. Therefore, numerous pervasive applications in the transport domain intersect passenger safety, trajectory planning and optimization, entertainment and communication systems, and data collection for maintenance and prediction. For instance, smart-car applications span several domains including vehicle tracking, communications between vehicles (e.g., interchanging context information such as speed, destination, distance between them, etc.), coordination between vehicles and traffic control systems to avoid congestions. Therefore, an intelligent traffic control system is deployed to avoid crowded streets based on potential traffic congestion, and offer more efficient navigation systems for users.

- **Smart Home Domain**

Pervasive environments have been integrated widely in smart home domains. In fact, a smart home can be embedded with rich sensors (e.g., temperature sensor, humidity sensor, luminosity sensor, movement sensor, camera, noisy sensor, kitchen sensor, etc.), devices (e.g., smart tv, laptop, temperature dashboard, alarm system dashboard, curtain system dashboard, etc.), and actuators (e.g., temperature regulator, light switch, door mechanism, curtain mechanism, etc.) connected to the Internet through the set-top box. These sensors and devices capture home context information and monitor user daily life activities in his smart home without human intervention. Currently, people are able to control their smart home through pervasive context-aware applications using their smartphone or tablet (e.g., open the curtain, adjust the lighting, turn on the air conditioner, open the garage, activate the alarm system, turn on the kitchen, etc.). These commands can be either triggered manually based on user decisions, or automatically based on home context information and user preferences. Pervasive smart-home applications may incorporate many fields including security (e.g., providing assistance when a potentially dangerous situation occurs such as fire detection, where the system can trigger automatically the fire extinguishing system), comfort (e.g., adjusting the temperature automatically), and energy saving (e.g., controlling the use of lights).

- **Smart Office Domain**

Pervasive applications have been integrated widely in enterprises to achieve better goals and benefits. These applications may facilitate the management and distribution of tasks to workers. Therefore, a smart office is an intelligent workspace enriched with pervasive technology to enhance important aspects of the work process, such as employee safety and task management. Most smart offices use pervasive technology to deduce where the staff is by following their locations and what tasks they have performed. For instance, the system relies on radio frequency identification (RFID) and global positioning system (GPS) technologies to tag and track staff in the workplace. As workers are equipped with RFID readers, the system can track automatically the development of activities and indicates the remaining daily tasks of the employee.

1.2.3 Issues and challenges in pervasive environments

Pervasive computing raises several problems due to its specific design, methodology, and tools. The main important challenges in this domain of application are as follows:

- **Heterogeneity.** Millions of devices will be connected all running on different processor architectures and operating systems, and communicating through various protocols, and communication supports.
- **Adaptability and Scalability.** Pervasive computing is far from a static environment. It is ultimately dynamic and subject to change of components. Therefore, the system must be scalable and adaptive enough to accept any added functionality at any given time. More precisely, pervasive computing is composed of a growing number of smart objects where these objects communicate with each other. The system should have the ability to effectively manage the increasing number of objects and perform their functions with the required efficiency.
- **Flexibility.** Pervasive computing requires fast responses to user's current needs, his situations and usage context (i.e., at run time) and services availability cannot be known a priori. The service selection and reconfiguration are not easy due to the dynamic context changes. This raises other challenges when one or more services are no longer available (e.g., due to network capacity or poor network connection) while running the application. So, the system must be flexible enough with less cost for basics service reconfigurations (add/migrate/remove service).
- **Real-time processing (precision).** Pervasive computing is a system that consists of three main phases: event monitoring, context reasoning, and query processing. Pervasive computing is a real-time system only and only if it performed all these phases in real-time which is a very complex task especially with the growth of users' expectations (e.g., user's preferred modality interaction). Further, most of the interactions in pervasive environments are expected to be performed in real-

time. However, this paradigm has been rarely addressed by the research community in pervasive computing systems. Another important factor that we can find in real-time systems is precision. When dealing with precision machines that can fail if timing is off by a millisecond, adhering to strict requirements becomes pivotal to the health and safety of the machine operators. Originated in distributed computing, but more challenging in pervasive computing due to the important cruciality of network latency. Best example of crucial precision is airplanes' communication and coordination with airdromes.

- **Mobility.** A coherent and stable state of the network topology is an important factor for the management of pervasive applications. However, mobility of nodes (e.g., smartphones, wearable sensors or actuators, smartwatch, etc.) in a pervasive environment can cause the loss of connection to certain devices, or new connections to different devices can be established.
- **Distributed and parallel aspect.** The massive amount of information that can be collected and processed by pervasive systems may result in a significant computational time. Traditional centralized computing systems cannot process all these data within a reasonable time. Further, centralized architecture suffers from the traditional problem the single point of failure. Therefore, a distributed strategy based on parallel computing is needed to accelerate the processing time allowing autonomy and mobility. However, this strategy requires patterns to guaranty the context consistency and integrity.

1.3 Context and context-awareness

Context is a key concept of any pervasive computing system. In order to understand the concepts of context and context-awareness used in this section, it is necessary to provide some useful definitions of concepts that will appear later.

1.3.1 Sensor

Sensor is an electronic component that can sense physical phenomenon such as movement, temperature, and humidity. All pervasive systems must rely on a wide range of sensors to collect data from the physical world. Nowadays, sensing technologies have made significant progress in designing sensors with a smaller size, lighter weight and longer battery life [26]. Thus, sensors can be embedded in particular objects and human bodies. Sensors can capture a different range of context information as the following aspects:

- Environment: temperature, barometric pressure, humidity, light level, noise level in an ambient environment;
- User: location, schedule agenda, motion, biometrical data like heart rate, glucose level, and blood pressure;
- Device: Size of memory, CPU speed, the resolution of screen, device availability;

- Interaction: interacting with devices through virtual sensors using different modalities (click, touch, gesture, and voice), interacting with real objects through RFID [27], etc.

The huge number and diversity of sensors in pervasive environment lead to high complexity in terms of interpreting: huge data volumes, different modalities, inter-dependence and real-time update. Moreover, in real world, sensors produce imperfect data that my result in misunderstanding of user's activities or environment state. As a consequence, the execution of pervasive applications leads to incorrect behavior. There are also many limitations as incomplete, redundant and contradictory context data that may produce uncertainty issue. In fact, pervasive applications should take into account all these issues which complicate the process of making these data immediately understandable, efficient and consistent to applications [28]. To cope with this problem, developers need to know how to monitor these imperfect data using recognizing patterns that may give us better predicting of human activities [29].

1.3.2 Raw sensor data and context information

To differentiate between raw sensor data and context, Sanchez et al. [30] define raw sensor data and context information as follows:

- **Raw sensor data:** is unprocessed data that is observed directly from the data source such as sensors (e.g., GPS coordinate).
- **Context information:** is generated by processing raw sensor data. In other words, it represents checked data for consistency where metadata is added (e.g., geographical location extracted from GPS coordinate). In other words, context information represents well-structured information that describes interrelationships and properties between concepts of an environment or a user.

Generally, the raw sensor data that are used to generate context information can be identified as context. Raw data values have different types, including numeric, binary, and feature values. Each type has different chosen techniques to analyze it. The diversity of sensors categories and their data types is a major challenge for pervasive applications to analyze and process each type separately. Numeric values are sensed by the majority of sensors, including ambient sensors; Biometric sensors, spatial sensors, etc. The basic data type is binary that relies on two values: true (1) or false (0) (e.g. motion sensor produces true value for the human presence or false instead). The featured values are the most complex type, generally produced from more sophisticated sensors based on images such as a camera. Table 1.1 summarizes a list of used sensors and their types of context data in a smart environment [2].

Table 1.1: A list of used sensors and their types in the smart environment [2].

Sensor types	Type of values
Positioning detection	Numeric
Interaction witch objects	Binary, numeric
Acceleration	Numeric
Biometric parameters	Numeric
Resource usage	Numeric
Ambient parameters	Numeric, binary
Processed video	Features

1.3.3 Actuator

An actuator is a component based on mechatronic mechanism (i.e., technology combining electronics and mechanical engineering) that is responsible for moving a mechanism or controlling systems. It operates in two phases including control signal and source of energy for a given action (e.g., opening a door, activating alarm, etc.). The control signal may be electric energy, hydraulic pressure, or pneumatic pressure. Generally, the control signal is sent by the system to control an environment (e.g., home, car, etc.). After receiving the signal to perform an action, the actuator needs a source of energy that may be electric energy, hydraulic fluid pressure, or pneumatic pressure.

Sensor and actuator are primordial components to build and control a pervasive environment. Figure 1.5 illustrates the interrelationship between sensors, pervasive system, and actuators. Sensor sense physical phenomenon from physical world resulting in raw sensor data. These data are used to generate context information which in turn is used by the system. The system processes and analyses context information then takes decisions resulting in sending control signals to actuators. Actuators move or control a mechanism in the pervasive environment.

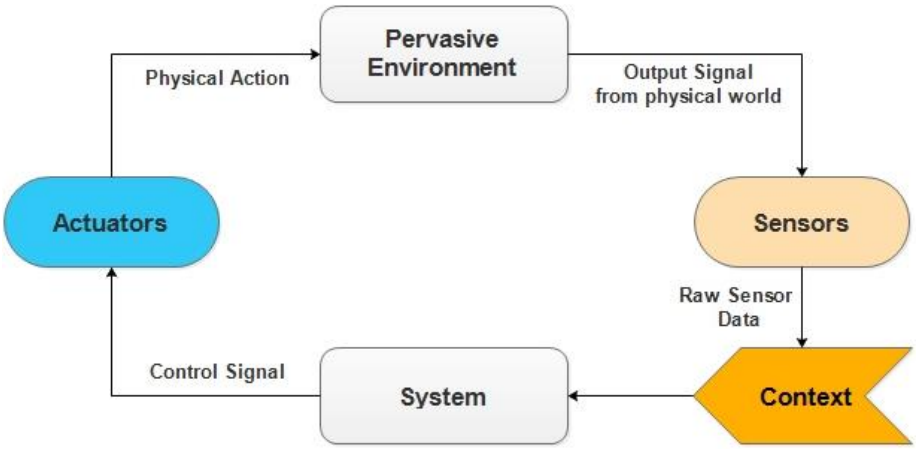


Figure 1.5: The interrelationship between sensors, system, and actuators in pervasive computing.

1.3.4 What is context?

In the literature context is the interrelated conditions in which something exists or occurs. More precisely, the context is the mutual relationship between a set of conditions that define a given state of an object in which some events can occur.

According to the Oxford Dictionary, context is defined as "*The circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood*" [31].

According to the Cambridge Dictionary, context is defined as "*The situation within which something exists or happens, and that can help explain it*" [32].

These definitions are very large and when applied to the field of pervasive computing it is hard to specify which elements of the context are important. Therefore, many definitions in the scientific literature were given to the term *context*. Researchers in diverse domains tried to define the term *context* basing on the necessities and the investigated field to be more applicable to computer applications. In fact, there are two kinds of definitions, a user-centered focus and a system centered focus. In the following, we present the most common of context definition proposals ordered chronologically:

- Schilit *et al.* [12] were among the first researchers who present a formalization of the user-centered understanding of context applied to the field of mobile computers as "*three important aspects of context are: "where you are, who you are with, and what resources are nearby* ". They stated that context is more than just knowledge of the user's location; there are other important aspects of interest to the user that may also mobile and constantly changing. So, they formalize the contextual information as the relationship between the user and the different things around him. The formalization defined by Schilit *et al.* [12] gives a more specific and tangible definition that we can apply to computer applications. Later researchers adapted this definition of context.
- Brown *et al.* [13] considered context as "location, identities of the people around the user, the time of day, season, temperature, etc."
- Pascoe [14] defined context as "the subset of physical and conceptual states of interest to a particular entity".
- Brézillon *et al.*[33] considered context as "all the knowledge that constrains a problem-solving at a given step without intervening in it explicitly".
- One of the most complete definitions was given by Dey and Abowd [3] in 1999, they refined the definition given by Schilit et al. and defined the context in the following way: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*". This definition is a more general definition of context. It operates with any information that can be used to identify the situation of an entity. An entity is every element

of the context that can be found in an environment, including the user, the application, and the interaction between them. Unlike other user-centric or application-centric approaches, Dey and Abowd have focused on what is relevant to the interaction between them. This approach gives designers an abstract level to capture all the context elements that are relevant to a given design, regardless of the design view on which the application is based (user-centric approach or application-centric approach).

In this thesis we have relied on this definition to express what context is, basing on three aspects of the context monitoring: capturing new incoming events from sensors, context updating, and context analyzing.

- Schmidt [34] defined context as "the knowledge about the state of the user and device, including surroundings, situation, and tasks".
- Strassner *et al.* [35] give a specific definition of context as: "The Context of an Entity is a collection of measured and inferred knowledge that describes the state and environment in which an Entity exists or has existed'.

1.3.5 What is event?

After defining the context, there is a clear relationship between context and event. Ahn *et al.* [36] define the interrelation between context and event as follows "context is a set of interrelated events with logical and timing relations among them". Further, they define an event as an occurred data tagged with occurrence time that triggers a condition in a target area. Hence, a sensor deployed in a target area has to detect the given event.

We distinguish two categories of events including discrete event and continuous event. If we consider that a sensor has a specific frequency in which it is periodically awakes at every sampling period p to observe an event:

- **Discrete event:** an event occurs at time t as first event instance, and $t+p$ as second event instance, both event instances are considered as separated (e.g., a door open, kitchen on, etc.)
- **Continuous event:** an event occurs at time t as first event instance, and $t+p$ as second event instance, both event instances cannot be considered as separated (e.g., eating, driving, raining, etc.)

After analyzing the previous definitions of the term *context and event*, both are semantic terms. They are similar but not identical. The difference between "state" and "transition" is the same as between "context" and "event" respectively. From my point of view, we define the interrelation between context and event as follows:

Context vs Event

Context is the state of the application at a specific time, while event is the transition that can change the state of the application (i.e. context change). In the field of pervasive computing, applications undergo many context changes. Thus, applications must be aware of these changes to react to new incoming events and to provide relevant information and services according to the user's current needs and usage context.

1.3.6 Types of events: simple and complex event

We distinguish two types of events including simple events and complex events.

- **Simple Event.** A simple event is the value of a context-attribute (e.g., 1.45 g/l is the simple event sensed by the glucose meter sensor for the glucose context attribute). The context attribute can be associated with qualitative comparators (e.g., inside, outside, near and far for Location; before, after and equal for Time, etc.). Each simple event is detected by using a sensor. Urgent events (e.g., health events and social security events) require immediate actions to deal with the situation, such as high glucose level and very high home temperature. Each event is defined formally as a set of metadata: event location, appearance time, valid period, sensed value.
- **Complex Event.** A complex event is considered as a set of aggregated simple events in order to deduce a composed value of a complex event (e.g., a four-room flat with a motion sensor in each room, a motion event in a single room is considered as a simple event, a flat motion event is the aggregation of all the motion room's events). A complex event can be interpreted as an event that requires multiple sensors at the same time.

In the next section, we will present the foremost definitions given to the term context-awareness.

1.3.7 Context-awareness

The original of the term context-aware computing was first introduced by Schilit and Theimer [37] in 1994, they defined it as: "the ability of a mobile user's applications to discover and react to changes in the environment they are situated in".

- Next, Schilit et al. [12] redefined the first definition of context-awareness as the ability to" adapt according to the location of user, the collection of nearby people and objects, as well as changes to those objects over time". This means that software is able to collect the contextual information of the environment that surrounds it and reacts according to the context changes.
- Abowd and Dey [3] have given a more general definition of what is a context-aware system. They define it as: "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". It is obvious that the term context

is within the definition which gives a tangible link between context-aware and context and shows the dependency between both terms. This definition is more suitable and compatible with the definition of context given by Abowd and Dey in [3]. Thus, in this thesis, we have relied on this definition to express what context-awareness is.

From the presented definitions, we notice that the context-awareness can be based on two important mechanisms: the ability to monitor the contextual information and the ability to exploit and react for environment changes. In the next section, we are interested in presenting different categorizations and the classifications of context information given by different researchers.

1.3.8 Context types and categorization

The context and its related information can be classified into different types and categorization schemes. Therefore, researchers have categorized context based on different perspectives. Van *et al*. [38] classified the context categorization into two broader categories including operational and conceptual categorization.

- **Operational categorization:** consists of classifying context based on the way it is acquired, modeled, and treated.
- **Conceptual categorization:** consists of classifying context based on the meaning and conceptual relationships between context information.

a) Generally, from the operational perspective, Perera *et al.* [39] classified the context into two general classes as primary and secondary contexts. The collected information from sensors is classified as primary context (i.e., raw sensor data). On the other side, information that combines data from different sources to find new significant information is called secondary context. In other words, secondary context is generated using data fusion operations-of the primary context. For instance, if we collect the glucose level directly from a sensor implanted in the patient, this context information is classified as a primary context. Besides, if we infer the same context information from combining multiple sources of primary context data such as heart rate and sweating rate, this context information is classified as a secondary context.

b) From the conceptual perspective, there are several categorizations that highlight relationships between contexts. Several researchers propose different categorization schemes to the context. Schilit *et al.* [12] classified context into three categories based on three questions:

- **Where you are:** consist of all contextual information related to location such as GPS coordinates (e.g., home, university, office, shopping mall, etc.).
- **Who you are with:** represents contextual information about the people nearby the user.
- **What resources are nearby:** represents contextual information about available resources

around the user's location (e.g., devices nearby the user in a specific area).

Henricksen *et al.* [40] define four context categories as follows:

- **Sensed:** sensor raw data received from sensors such as humidity value measured by humidity sensor. In this category values are changing dynamically over time with a high frequency.
- **Static:** this category represents static information that will not change over time such as device's capabilities, sensor's capabilities, etc.
- **Profiled:** this category represents information that changes over time with a low frequency such as IP of sensor and ID of sensor.
- **Derived:** this category represents information calculated using other context information such as distance between two users calculated using two GPS sensors.

Perera *et al.* [39] classify context from the conceptual perspective in four categories including location, identity, time and activity. Figure 1.6 illustrates context categorization based on two perspectives (*conceptual and operational*) as defined by Perara *et al.* [39].

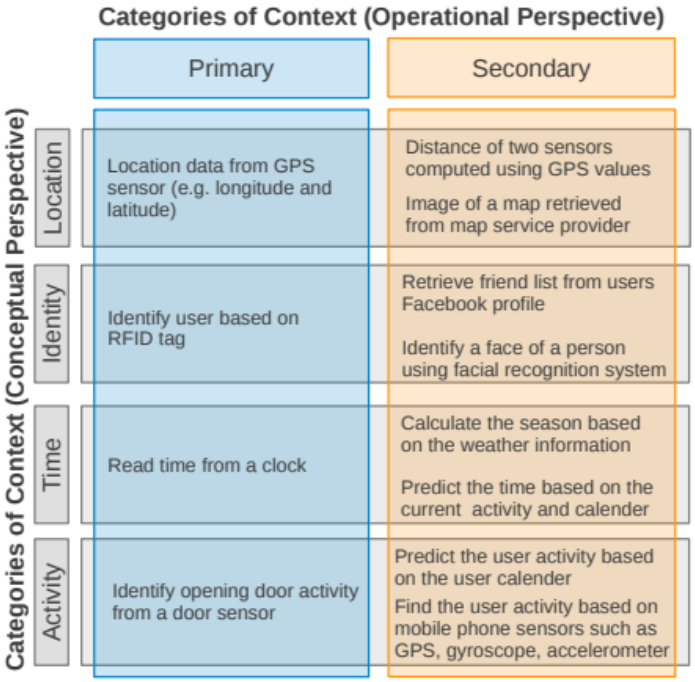


Figure 1.6: Context categorization based on conceptual and operational perspectives [39].

None of the presented categorization schemes can be considered as the best classification for context. Therefore, each categorization scheme can be suitable and reasonable in a specific situation. However, pervasive computing system needs to consider more comprehensive categorization schemes in hierarchical manner, such as generic categories, subcategories, and so on. The hierarchical categorization schemes of context can play a significant role in classifying and arranging context information in a smooth way. This hierarchical categorization of context information facilitates to give a clear vision of context's changes over time and thus react rapidly to these changes.

1.3.9 Context life cycle

Context life cycle represents data moves from several software modules. More specifically, it shows the life cycle of perceived data from the generation until consumption. Bernardos *et al.* [41] design four phases in a typical context life cycle management system including context acquisition, context modelling and processing, context reasoning, and context decision. Figure 1.7 illustrates the four-phase context life cycle proposed by Bernardos [41]. The first phase consists of acquiring data from different sources including sensors and user inputs (context acquisition). The second phase consists of collecting data and representing them according to a conceptual model (context modeling). The third phase consists of taking modeled data and processed them to drive high-level context from low-level raw sensor data (context reasoning). The final phase consists of disseminating high-level context information to corresponding applications in order to make decisions according to context changes (context dissemination).

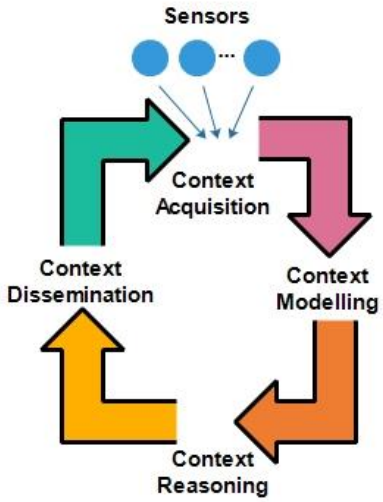


Figure 1.7: Context life cycle in context management system [39].

Considering the entire context in a pervasive system needs a powerful computational capability that is not always available. Besides, context is changing dynamically over time which may make context-awareness a hard task to achieve. Therefore, it may reduce the performance in terms of precision and real-time reaction. Further, some changes in context are meaningless and can produce an additional computational time. Hence, considering a part of context that represents only meaningful information namely situations, reduces this additional computational time and enhances the system's performances. In the next section, we are going to present the foremost definitions given to the term situation and situation-awareness.

1.4 Situation and situation-awareness

Situation and situation-awareness are subjective concepts in pervasive systems, their definitions rely on context and context awareness concepts. This section presents definitions given to situation and situation-awareness concepts.

1.4.1 What is situation?

According to Ye *et al.* [2] "*a situation can be defined by collecting relevant contexts, uncovering meaningful correlations between them, and labeling them with a descriptive name*". The descriptive name represents a descriptive definition of a situation, that is to say how a simple user or an expert defines this situation in reality. Further, users or experts can specify a situation logically using logical expressions of correlated context predicates. Therefore, with situation description and situation specification, a situation can bridge sensor data and applications. In other words, sensors raw data can be abstracted to a specific situation by evaluating its specification, while this situation is identified logically, the system triggers automatically actions correlated for the identified situation.

For instance, figure 1.8 shows a general architecture of information flow in pervasive computing, including the three main hierarchical concepts in any pervasive system, namely sensor raw data, context, and situation. As we see at the bottom of figure 1.8, sensor produces raw sensor data which can be characterized into a set of concepts namely context. A location context *studyRoom* is mapped to a set of coordinates GPS sensor data. This context has semantic relationships between other context information that can give the same interpretation as GPS sensor data such as user's interaction with the keyboard located in the desktop of the *studyRoom*. *StudyRoom* context and *KeyboardAccessed* context are considered relevant in which they share semantically the same location. These semantic relationships can be explored through a specification-based approach or learning-based approach. We will show the difference between these two approaches in chapter 2. After exploring the context information, a situation "*working*" can be identified – *working* is its descriptive name. A logical specification can define this situation as follows: "if a user is in the study room and accessing the keyboard of the desktop, this user is considered in a *working* situation". From the application perspective, it consumes the identified situation and runs automatically a set of actions associated with this situation (e.g., adjusting the sound level of the background music).

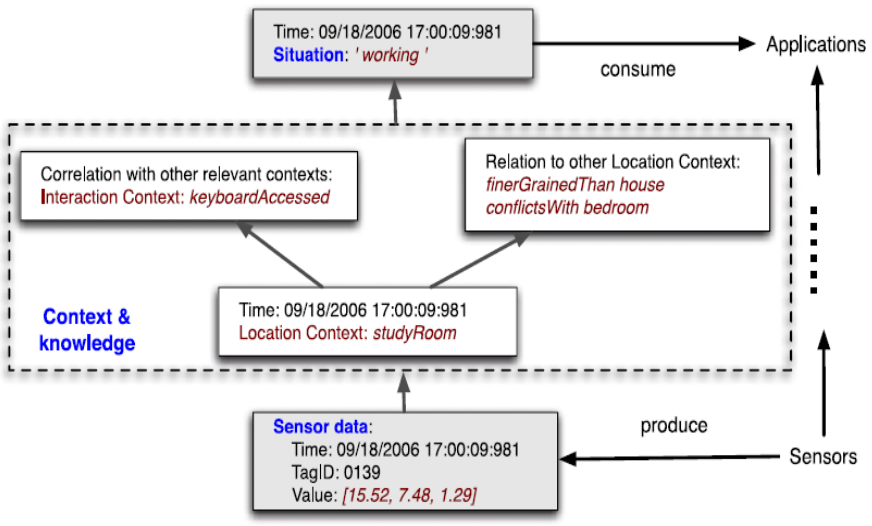


Figure 1.8: Information flow in pervasive computing [2].

Karchoud *et al.* [42] define situation as a projection on three axes including *time*, *location* and *activity*. Where activity represents the task that the user is performing. At some level user's activity and user's situation seem similar terms. In fact, what distinguishes activity from situation, is that the situation includes rich context information, namely location and time. A situation is specified with a particular location and time where/when it may happen.

After analyzing the previous definitions of the term *context, event,* and *situation*, we can give a definition that interrelates between all these concepts as follows:

Context, Event and Situation

Situation is the set of things (i.e., events/context) that happens at a given location (i.e., where?) and at a given time (i.e., when?), and the conditions (i.e., clauses) that can verify those events, thus identifying the situation.

We specify a situation through a set of clauses namely a situation rule. When all the clauses are verified (i.e., the situation is identified), the system triggers automatically the set of actions associated with the identified situation. For example, a high glucose situation results in injecting insulin in the human body automatically.

1.4.2 Types of situations: simple and composite situation

The situation can be either a simple situation or a composite situation. A simple situation is defined by a situation rule as a set of conditions and a set of actions. A composite situation is a composition of multiple situations using different composition operators (e.g., parallel, sequence, recurrence, and alternative). For example, a diabetic who has a not so good diet situation that may be modeled as a composite situation, which consists of a high glucose recurrence situation. The high glucose situation, without the recurrence composition operator, is considered as a simple situation. A simple high glucose situation results in injecting insulin. In contrast, the composite situation results in sending diagnostic results of the patient to the doctor.

1.4.3 Situation-awareness

Situation and situation awareness are different concepts but they complement each other. The situation concept represents the conceptual model specified by the designer to identify a situation. Whereas, situation-awareness is a cognitive process in decision-making after identifying a situation during context changes. Endsley *et al.* [43] define situation-awareness as "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future". They modeled situation-awareness system in three main layers including *perception*, *comprehension*, and *projection*. The perception layer represents all the necessary information collected from

the environment through sensors and devices. More specifically, we can consider the perception layer as context layer. The comprehension layer interprets and explores semantic relationships of received information in order to understand and identify the current state of the environment, namely situation. The projection layer has a direct relationship with the identified situation to predict its future state.

Situation-awareness is a related concept to the notion of context-awareness. These two concepts can be used interchangeably. However, a system is a context-aware if it uses context to provide relevant information and services to the user [3]. Based on this definition, we can say that a system is a situation-aware if it uses situation to provide relevant information and services to the user.

1.4.4 Situation management process

The situation management process in pervasive computing systems aims to facilitate mobile services delivery in various smart domains through a situation identification process where each situation is linked semantically to a set of mobile services. Moreover, contextual information acquired from pervasive environments must be represented and modeled into situation specification to unify and facilitates the situation identification mechanism. The situation management process consists of four main steps as follows:

- **Situation representation:** aims to define logic primitives (e.g., temporal primitive, location primitive, etc.) that are used to construct situation rules called situation's logical specification.
- **Situation specification and modeling:** aims to specify and model how a situation can be built using a situation's logical specification based on situation model (e.g., ontology model for situation specification). Several situation modeling techniques proposed in pervasive computing will be detailed in the next chapter.
- **Situation identification and reasoning**: represents how to infer and identify situations from a huge amount of raw sensor data. Several identification techniques are used to reason on situations' relationships. We distinguish two essential approaches for situation identification namely specification-based approaches and learning-based approaches. Both approaches will be detailed in the next chapter.
- **Situation recommending**: situation recommendation techniques are responsible for enriching user's agenda with useful situation rules. Several recommendation techniques will be discussed in the next chapter.

After defining the concept of context and situation, we need to clarify the relationship between pervasive computing systems and these concepts. The next section presents the conceptual architecture of pervasive computing systems based on context and situation.

1.5 Context and situation in pervasive computing

Traditional applications were unaware of the context in which they neglect context information related to the user's environment. In other words, they do not discern where the user is, what is he doing, who is nearby, and other context information. They settled only for explicit inputs from the user (e.g., know his ID, pointing to an icon with the cursor, etc.). Recently, several applications adopt the use of context to improve the quality of provided services by adapting them relevantly to the user's current context. Therefore, context has been integrated into different computing domains such as: IoT [39], healthcare [44], and even pervasive computing [45]. In this thesis, we are focused on context-awareness in the pervasive computing.

Pervasive computing is a complex system that requires a combination of existing technologies and mechanisms. To build up a full pervasive computing system, there are important elements needed which are hardware, business model, and software. Therefore, the development of a pervasive system involves three main elements:

- **Hardware:** sensors, actuators, and devices.
- **Model:** Context and situation models.
- **Software:** represents the application part that can control: context life cycle, situation identification, and adaptation.

Figure 1.9 provides an overview of a full hierarchical layers of pervasive system. Its layers consist of four main modules including raw sensor data acquisition, context processing, situation identification, and adaptation. The adaptation module is responsible for triggering actions according to context changes. The context can be determined after the processing of raw sensors data and user inputs. The first step consists of receiving heterogeneous input data, where they are collected as raw sensor data. These data are viewed as low-level context. After that, these data are processed (i.e., add metadata to raw sensor data, verify consistency, aggregation, semantic interpretation) to generate context information known as high-level context [46]. In pervasive computing, situation identification and situation-awareness are built on a high-level context to formalize and deduce real-life situations out of context. In other words, the situation concept is properly designed to extract the most relevant information from the context, while situation-awareness takes place to perform the adaptation for identified situations. Finally, according to the provided inputs and situation-aware system, the adaptation module sends commands to actuators to execute various actions (e.g., adjust the air conditioner temperature according to the sensed temperature in the office).

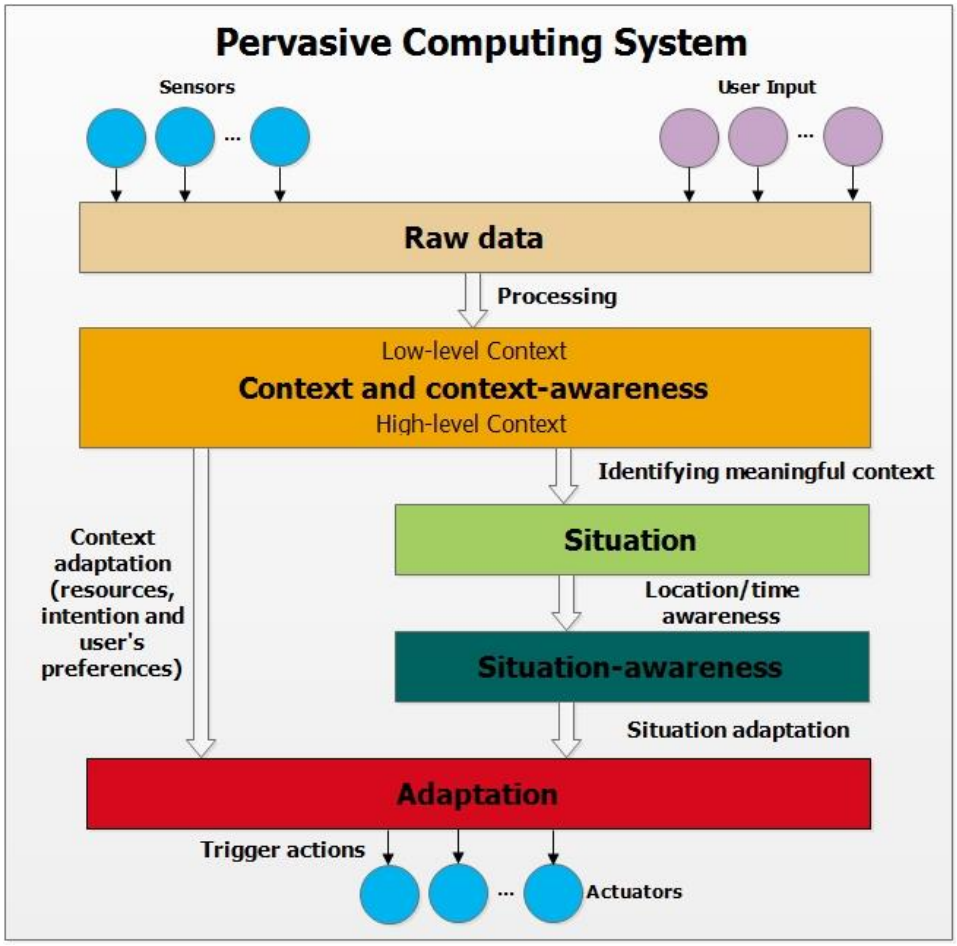


Figure 1.9: Conceptual Architecture of context processing in pervasive system.

In the next section, we will investigate the importance of ontology that aims to represent heterogeneous smart objects from the technical view in pervasive computing.

1.6 Ontology and standard ontologies for pervasive environment

Recently, various standard ontologies have been modeled in pervasive computing to provide semantic level facilities that allow the management of user situations based on heterogeneous services in order to adapt to its current needs. In addition, ontologies enhance application dynamicity and interoperability based on common context information for a better unification, flexibility, adaptation, and customization of mobile applications.

This section presents both ontology and semantic representation of heterogeneous smart objects, and shows some standard ontologies designed for pervasive environments.

1.6.1 What is ontology?

An ontology is a set of concepts that provide information semantically. The concepts are organized in a graph whose relations can be *semantic relations*, *composition,* and *inheritance* relations. However, the most mentioned definition is Gruber's: "An ontology is an explicit and formal specification of a particular area

of knowledge" [47]. An explicit and formal definition allows software agents to reason and infers new knowledge. The main purpose of an ontology is to model a set of knowledge in a specific domain. In the domain of the semantic web, there are two main description languages for ontologies: RDF (Resource Description Framework) and OWL (Ontology Web Language).

Ontologies provide a common vocabulary of a particular domain and define the meaning of terms and relationships between them. The knowledge in the ontologies is mainly formalized using four types of components [48] called: *concepts (or classes)*, *relations (or properties)*, *axioms (or rules)* and *instances (or individuals)*.

1.6.2 Semantic representation of smart object

A pervasive environment consists of various smart-* domains with various heterogonous smart objects (serval sensors and actuators). To represent such domain, we need an ontology domain. Ontology is a semantic model that provides the relationship between smart objects whatever their communication protocols and context data type. This allows the system to access and reason with the context data in a uniform way. For instance, figure 1.10 presents a hierarchical representation of sensors and actuators for pervasive smart objects ontology. While figure 1.11 illustrates an ontology reasoning for controlling concretely a radiator in a smart home under abnormal temperature detected by a temperature sensor (i.e., low temperature implies turn on the radiator, high temperature implies turn off the radiator).

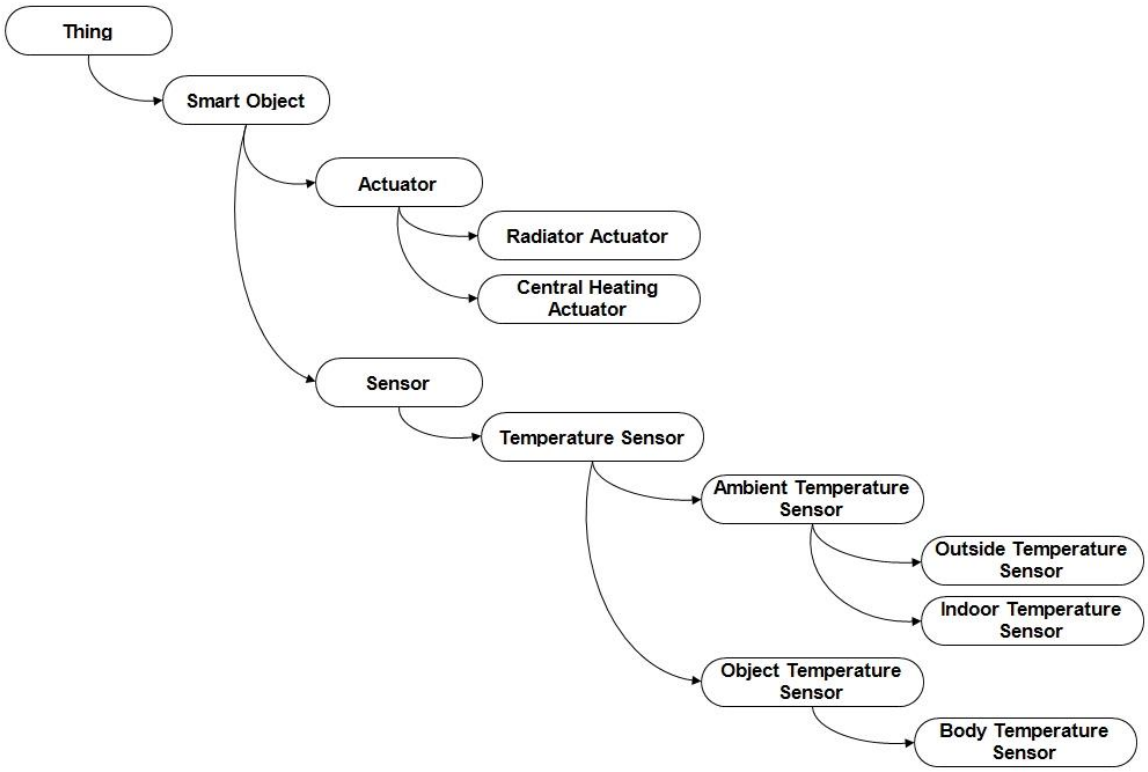


Figure 1.10: Semantic Model of smart objects.

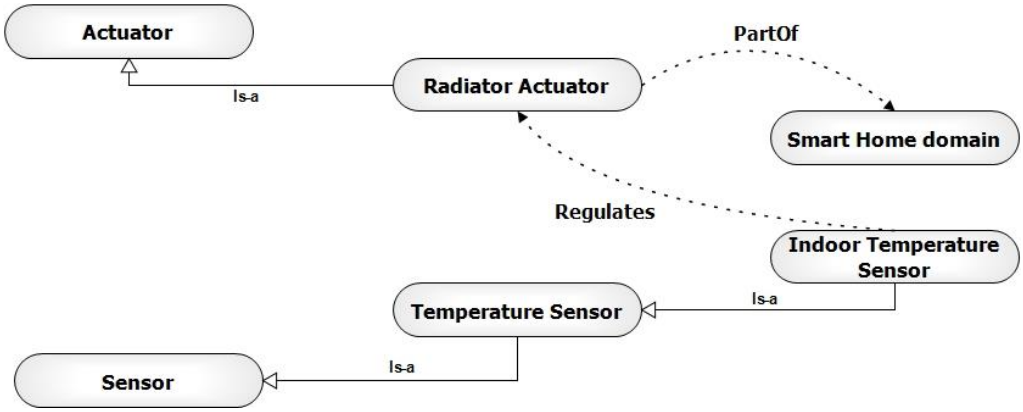


Figure 1.11: Ontology reasoning based on smart objects.

A semantic representation of smart objects helps to describe capabilities and additional information useful for interoperability and reasoning tasks. Table 1.2 illustrates some semantic information of a temperature sensor.

Table 1.2: Semantic information of a temperature sensor.

Temperature sensor	
Property	Type Values
is_sensor	Boolean, true, as Default
sensor_type	Domain (temperature)
sensor_id	Sensor identifier
sensor_location	Domain (home, office, indoor)
frequency	Numeric
temperature_value	float, valid > 22 && < 38

1.6.3 Standard ontologies for pervasive environment

During the analysis of certain ontologies relevant to our research area, we found several ontologies that model pervasive environments, smart objects, and user's profiles as follows:

- **Semantic Sensor Network (SSN)** [49]: is a standardized W3C ontology that describes five fundamental concepts (sensor, sensor's observations, sensor data samples, actuator, and services) and also the relationships between these concepts.
- **Sensor, Observation, Sample, and Actuator (SOSA)** [50]: offer sophisticated capabilities including social sensing, observation-driven and context monitoring.
- **Smart Applications Reference Ontology (SAREF)** [51]: is a reference ontology for IoT devices and IoT applications for energy efficiency solutions.
- **DOLCE Ultra-Light Ontology (DUL)** [52]: is an ultra-light version of ontologies DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) and DnS (Descriptions and Situations ontology), which provides simplification and enrichment of some concepts of DOLCE.

- **Friend of a Friend Ontology (FOAF)** [53]: is an RDF ontology for modeling users, their personal data, their relationships, and social concepts.

1.6.4 Ontology design process for pervasive environment

This section presents shared guidelines that help developers to design an ontology. Ontologies are used to cover the knowledge sharing between users, smart domains, and smart services to provide a better representation of heterogeneous context information in order to infer users' situations for smart mobile applications. In knowledge engineering, a top-down approach is a good technique to reduce design complexity by enabling hierarchical knowledge representation and ensuring a high level of abstraction in pervasive environments. There are different approaches for modeling an ontology for pervasive environments and their smart-* domains (home, city, health, car, etc.). Before we start developing such ontology, we should consider reusing existing standard ontologies to save costs and achieve interoperable, flexible, semantic, and open linked models. The ontology design process consists of the flowing fundamental steps (see figure 1.12):

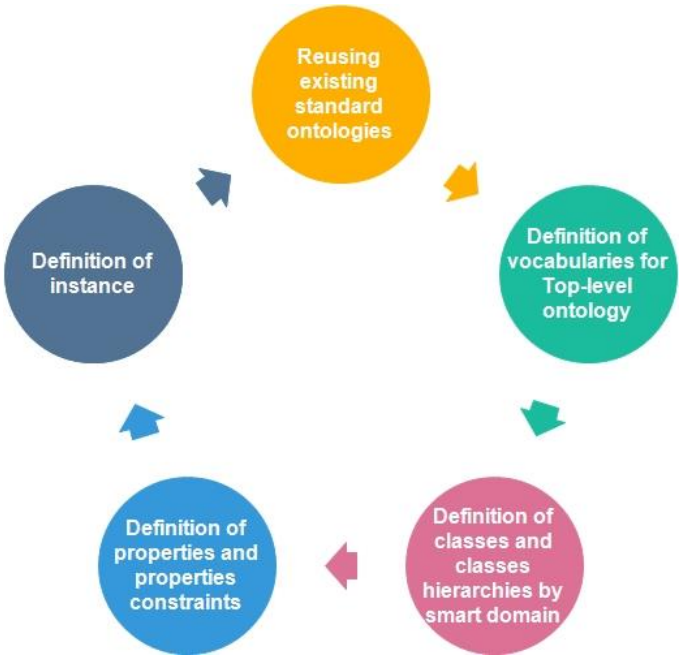


Figure 1.12: Ontology Design Process for pervasive environment.

- Firstly, we build a top-level ontology (upper ontology) that is simple and use common concepts in different smart-* domains. Based on the common building concepts of these smart domains, we identify ontology vocabularies that should be covered in all-pervasive domains.
- Identified key vocabularies in the previous step are used to define most general concepts (classes) and the relationship between concepts in the top-level ontology.

- For each specific smart-domain, we follow a top-down approach in which we derive from top-level ontology concepts, hierarchical domain-specific concepts. The main goal is to ensure cohesion, consistency, and a high level of reuse of top-level ontology.
- Definition of properties and properties' constraints for smart domain ontology.
- Creation of instances for the classes.

In our research work, we are interested in the management of situations-oriented mobile application at different levels:

- Semantic-level that depends on generic top-level ontology model to facilitate the design of mobile applications and the management of various composite situations with multiple handled devices. We use an ontology model to cover all context information and situations that respond to the user's daily life activities for context-aware mobile application structured on three levels: (1) context upper ontology, (2) specific domains ontology, and (3) application of context ontology for a given smart domain.
- Design-level that describes situation-oriented application design model at different smart domains (health, home, tourism, health, etc.) as a set of situations components. The use of components in such applications reduces the complexity of situations management into small manageable parts, namely situation component, and thus provides a more flexible architecture. The situation component is a combination of knowledge-based ontology modeling, situation rules (constraints), and usage context-based ontology (scenarios).
- Runtime-level that ensures the management of application based on context monitoring mechanisms from different sensors, composite user's situation reasoning, and the auto-generation of (re-) deployment script of action services.

We note that design-level and runtime-level integrate the use of components and services to manage and deploy action services in order to respond to user context changes. Managing these components and services (add/remove/migrate) on different devices and platforms without tool support is a difficult task to achieve. Therefore, a dedicated middleware is needed to manage and deploy these services on mobile devices. The next section presents the standard middlewares for pervasive systems.

1.7 Middleware dedicated to pervasive systems

1.7.1 Background

Pervasive computing is a heterogeneous environment where the modularity of mobile applications becomes crucial. Therefore, the use of middleware in such environments is mandatory. In pervasive computing architectures, a middleware is a distributed layer of software that provides services to pervasive applications. It acts as a mid-level agent between service providers (i.e., sensors) and service consumers

(i.e., pervasive applications). In other words, middleware is a software layer used to bridge the gap between sensors and top-level applications (see figure 1.13). In addition, it ensures the communication between the involved devices and applications regardless of network protocols, hardware and software features. The most popular technique of communication and information exchange between devices and applications is to invoke remote action services by adding, removing, or migrating component-based services. From the perspective of Service-Oriented Architecture (SOA), a component is viewed as the cornerstone to build a service in which a service is the combination of one or more components. Middleware handles these components and services to provide context information for users through specific devices.

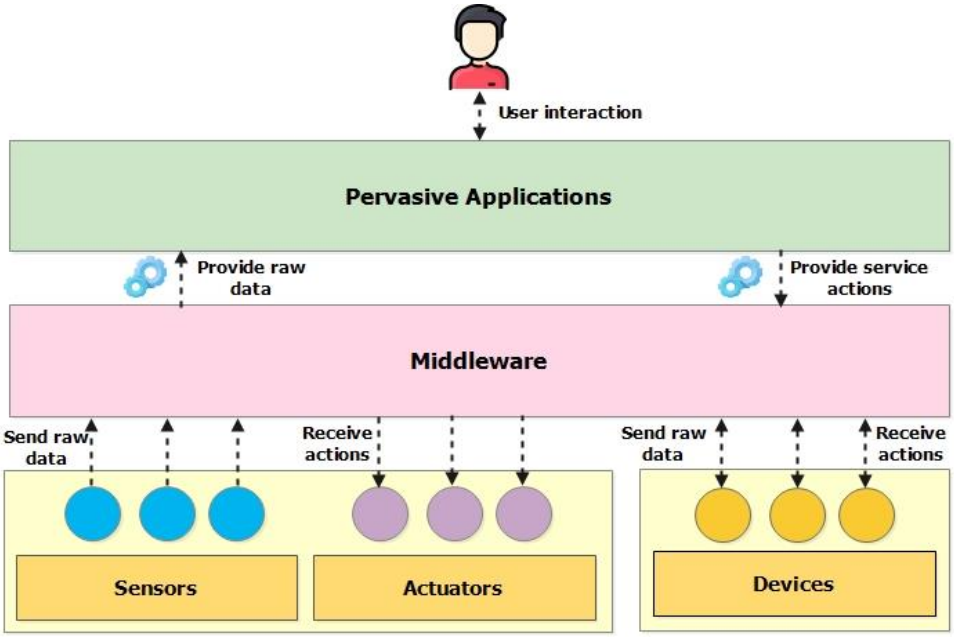


Figure 1.13: Pervasive computing architecture based on middleware.

In the next section, we present some existing middlewares solutions for pervasive applications.

1.7.2 Existing middlewares for pervasive applications

Several middlewares have been proposed for building context-aware applications. These middlewares unify and harmonize the reception of context information from deployed sensors in pervasive environments according to the current users' needs and usage situations. In the following, we present some middleware solutions for pervasive applications:

- **SATIN** [10] is a middleware based on components architecture. It provides reusable components that the developer can reuse them to create new applications. It relies on a modular architecture that allows us to add new functionalities at runtime. Therefore, to build an application based on SATIN middleware, it is mandatory to decompose it into modular functionalities. This feature provides reusability and interoperability between applications running on the same middleware.

Nonetheless, all applications built on SATIN depend on the middleware core component which leads to a centralized point of failure.

- **MUSIC** [7] is a middleware that enables reconfiguration of context-aware mobile applications. It supports component-based architecture adaptation. MUSIC is based on predefined adaptation rules that allow dynamic application reconfiguration according to context changes. Applications built based on MUSIC are modeled as component frameworks respecting the MUSIC implementations. Hence, this feature may reduce the flexibility of developed applications.
- **CARISMA** [11] is a middleware that adopts a peer-to-peer architecture. it enables to build context-aware adaptive application based on install/uninstall services on the online. CARISMA can detect context changes and triggers adaptation actions automatically for deployed applications. However, CARISMA does not provide service discovery in distributed environments.
- **WComp** [18] is a middleware for context-aware applications based on three main paradigms: event-based web services, a lightweight component-based application, and adaptation approach using the assembly aspect. These paradigms ensure a dynamic design model for ubiquitous computing applications through a decomposition/composition aspect. The adaptation approach uses a set of composite services in reaction to context changes. However, the adaptation design model is strongly coupled which may reduce the response time.
- **Kalimucho** [8] is a supervision middleware that manages the execution of services on available distributed devices. It is based on component architecture. Context changes is monitored through the components and connectors deployed on sensors and devices. The middleware ensures the communication between devices through connectors that Kalimucho sets up between the components. The middleware supports applications reconfiguration through supervised scripts as they are running by adding/removing/migrating component-based services while considering context changes.
- **Kali-smart** [9] is a middleware based on Kalimucho that allows dynamic reconfiguration of distributed mobile applications on desktops, laptops and mobile devices. It relies on component architecture to represent context information and user situation. The middleware is distributed on all mobile devices involved in the application and enables automatic decision-making. Moreover, Kali-Smart is a generic and flexible infrastructure that enables dynamic management of components to generate reconfiguration scripts on the fly.

Table 1.3 summarizes the limitations and strengths of studied middlewares. These middlewares are portable and runs on Smartphones, PCs and Laptops. Moreover, they ensure a better management of

context-aware functionalities. In this work, to improve flexibility and achieve high scalability (handling a big amount of context data and user's mobility), we rely on Kali-smart middleware [9] that enables automatic decision-making and semantic situations modeling.

Table 1.3: Middlewares' comparison.

Middlewares	Context Modeling	Mobility	Architecture	Adaptation	Portability	Flexibility
SATIN	✗	✓	Component	✗	✓	*Low*
MUSIC	✓	✓	Component	✓	✓	*Low*
CARISMA	✓	✓	Peer-to-peer	✓	✓	*Low*
WComp	✓	✓	Web service	✓	✓	*Low*
Kalimucho	✓	✓	Component	✓	✓	*Medium*
Kali-Smart	✓	✓	Component	✓	✓	*Medium*

1.8 Conclusion

The evolution of context-aware situation-based applications is leading us through pervasive computing which is a dominating paradigm. These applications have great potentials to facilitate people's lives and make the pervasive environment more friendly for citizens. In this chapter, we have presented three research domains related to our work's context, namely Pervasive Environments, Context-awareness and Situations based ontologies, and Middlewares. Firstly, we have described the pervasive environments and their different features challenges. After that, we have given potential definition related to context-awareness and situations concepts. We have investigated how these concepts offer a well understanding of user's context and easy management of user's situations through context and situation modeling. Then we have shown the relationship between pervasive computing and context-aware system in which pervasive applications adopt the use of context to improve the quality of provided services by adapting them relevantly to the user's current context. Then we have examined some existing standard ontologies which are used to unify the representation of heterogeneous context information and facilitate the management of a large number of heterogeneous connected objects and services. Finally, we have seen some middlewares dedicated to pervasive applications.

In the next chapter, we will describe the state of the art in modeling, identifying, and recommending services related situations. We will present the main research works in the field of situations identification and recommendation in pervasive environments.

Chapter 2

Literature Review on Situation-Based Approaches in Pervasive Computing

Summary

2.1 Introduction

Nowadays, we live in an age of pervasive computing that have the ability to serve users in sustainable context changes regardless of their locations, activities, and situations. However, human activities in real life know an endless possibility of user scenarios and situations. One of the main challenges in pervasive computing refers to the real-time identification and management of user's situations. The diversity of user's scenarios makes the management of user's situations very complex. In addition, identifying a huge number of situations under several moving scenarios (e.g., at home, at work, at car, etc.) and dynamic context evolution exhibiting varying degrees of precision, accuracy and processing time. Therefore, pervasive systems must offer a flexible, extensible, and dynamic model for collecting context information from sensors, identifying situations, then providing appropriate and adaptive services for identified situations. Further, with the huge number of situations in real-life scenarios, variety of user's needs, and multiplicity of devices to adapt to context changes, it becomes a necessity for pervasive systems to provide a dynamic solution that guarantees accurate recognition of situations. Nevertheless, the situation identification process may be ineffective or incomplete as it could be based beforehand on predefined situation rules, identified either by experts or users. In fact, it is a tedious task for each user to define his specific situation rules in any daily life scenario. Thus, a dynamic injection mechanism for new specific situations based on recommendation techniques is needed to predict and recommend the right situation in the right context for the right person.

Research into human activity recognition and situation identification in pervasive environments have been widely exhibited. This chapter explores different fields and approaches that serve as a frame for our work including situation modeling, reasoning and recommendation that have been proposed in the last decades. In this scope, we analyze the strengths and limitations of available works proposed in literature. Figure 2.1 summarizes different techniques and approaches for situation-based modeling, identification and recommendation. Several aspects are considered to synthesize the specification of each approach, namely composite situation modeling, priority, accuracy, scalability and flexibility. Our goal is to overcome the limitations of existing approaches and come out with a new flexible system that models composite situations model based on priority in multi-domain smart environments.

This chapter is organized as follows: Section 2.2 presents situations modeling approaches that aim to present different approaches addressing situation modeling. While section 2.3 presents situation identification approaches. Section 2.4 presents different recommendation approaches. Section 2.5 presents a synthesis section. Finally, section 2.6 presents the conclusion.

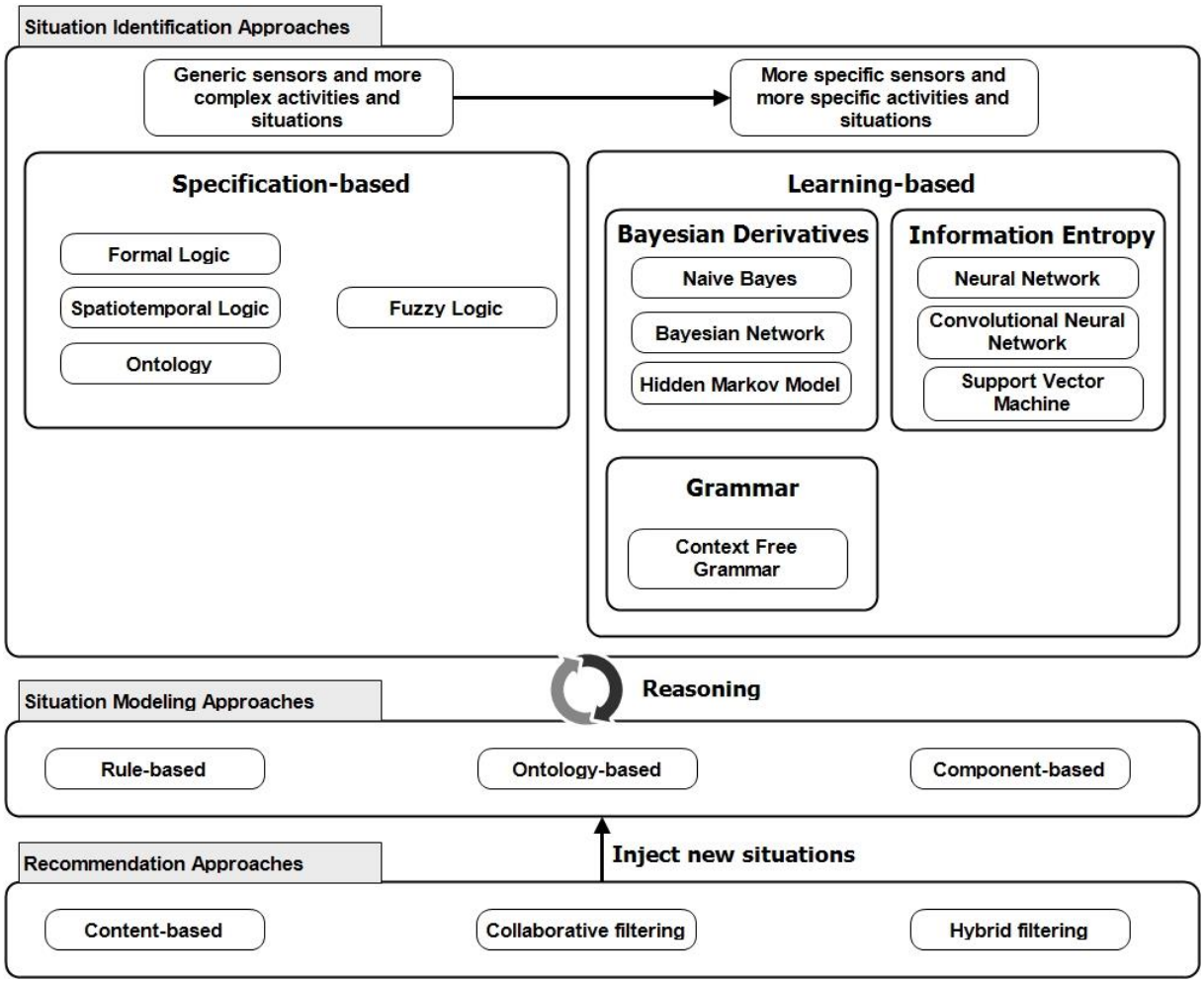


Figure 2.1: Recommendation techniques, situation modeling and identification approaches.

2.2 Situations modeling approaches

In this section, we review some recent relevant approaches addressing situation modeling and identification in pervasive environments. Most existing approaches able to identify everyday situations for pervasive and mobile environments. However, the related works suffer from various limitations including composite situations modeling and parallel situations reasoning. Our work is based on the semantic-based context-aware composite situation design model for mobile distributed applications. We first proposed a semantic representation of both complex events and composite situations of smart environments using composition operators parallel, sequence, recurrence and alternative for identifying and deducing composite situations.

2.2.1 Rule-based situation modeling approaches

In [1], the author uses the Event Condition Action (ECA) rule model to perform adaptation actions regarding incoming events checked by appropriate conditions. This model may be used to enable any adaptation behavior in pervasive applications. ECA rule consists of three parts including event, condition, and action. The event part describes and represents incoming events from sensors. The condition part represents a condition for checking incoming events (e.g., true or false). The action part performs planned

actions. This work ensures better management of context changes through monitoring and evaluating incoming events using ECA rule model. It covers context information representation and reasoning on various data collected from several sensors and devices. However, the main drawback of this work is the sequential checking of parallel incoming events that may increase the system's response time. Therefore, parallel checking techniques are needed to enhance the system's response time.

The CybreMinder [54] is an Eclipse plug-in that allows the explicit specification of a context as a combination of sub-contexts using diverse relations, such as =, <=, >=, <, or > as well as pre-defined contextual events. However, CybreMinder does not provide a range of composite operators such as parallel, sequence, alternative, and recurrence for mobile context-aware applications. Besides, CybreMinder neither supports multimodal actions to perform multi adaptations nor the combination of atomic distributed multimodal actions.

Recently, Karchoud *et al.* [55] propose a novel vision of user's long-life applications that relies on a rule-based spatiotemporal model for continuous context understanding and situation identification. However, this work lacks the support to manage accurately all contextual constraints. In other words, it does not take into account semantic composite relations between simple situations to create composite situations (e.g., recurrence of high glucose level as a composite situation).

More recently, Butt *et al.* [56] propose a priority model to manage and respond to patients' urgent situations. This work uses an objective function based on the patients' locations and their diseases to prioritize the situations. The goal of prioritizing situations is to reduce the system workload where only situations with high priority are firstly handled. However, the objective function used in this work neglects other important metrics such as user's current activity and situation's category. To deal with this problem, we have used four metrics including situation's category, user's activity, user's location, and situation's coverage to calculate the situation's priority. A weighted linear combination of these metrics is used to compute the final score for each situation (more details about the calculation of situation priority are given in chapter 4).

2.2.2 Ontology-based situation modeling approaches

Many initiatives have been taken in ontologies specification for semantic situation modeling and reasoning in pervasive environments. Despite the benefits of these works towards situation identification concerns, there are research works do not take into account context assumptions such as unhandled composite situations and situations' priority. In our work, a composite situation is described using different composition operators, while a situation priority is defined through predefined policies (for more details see chapter 4).

Roberto *et al.* [57] propose SHERLOCK system that provides semantic techniques to exchange contextual knowledge between devices, and to assist user in different context changes by selecting appropriate services for his current needs. SHERLOCK relies on OWL for knowledge modeling and reasoning using local ontology where several smart objects and user situations are specified. However, there is an apparent lack of a new way to manage composite situations and users' needs at one time in a dynamic and intelligent way.

Da *et al.* [8] propose an ontology-based context-driven management for Kalimucho middleware. This ontology offers the management of context information at a semantic level on the shared domain. This work provides monitoring techniques for context changes to ensure dynamic reconfiguration decisions. Based on reconfiguration decisions, Kalimucho offers predefined control policies for deployment strategies. However, this context-driven management model fails in distributed situation inferring and complex event processing due to the lack of parallel event processing strategy that can split the events into parallel sub-events (e.g., define "nobody at home" as an abstract complex event by aggregating all outputs values of motion sensors deployed in each room). To address this issue, Angsuchotmetee *et al.* [15] presented an ontology model (see Figure 2.2), known as Multimedia Semantic Sensor Network Ontology (MSSN-Onto), to model and process in real-time complex events in Multimedia Sensor Network. The approach uses pipeline-based technique for complex events detection by observing and combining several sensors. It provides a distributed logic-based detection and continuous monitoring technique to identify events that occur on multimedia sensors networks. The proposed system can achieve high accuracy detection with short processing time. However, it still has limitations, as it does not identify dynamically composite situations and critical anomalies (e.g., fire situations, heart attacks and other urgent situations). Therefore, this work lacks the reconfiguration aspect and priority integration among situations on distributed applications.

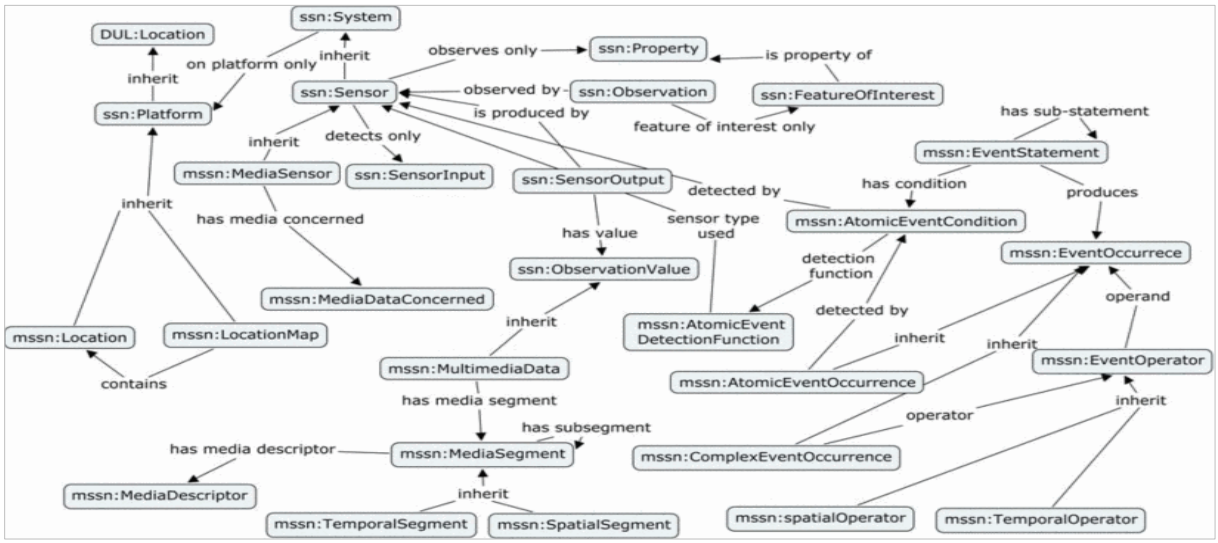


Figure 2.2: Complex events-model [15].

Other ontologies emerged based on Semantic Sensor Network (SSN) [4] such as HSSN Ontology (Hybrid Semantic Sensor Networks) [6], HealthCare Ontology [16], and SmartHome Ontology [17]. They provide a direct mapping between event-based sensors and OWL context-oriented patterns. However, these ontologies lack a high semantic-based description with the growth of heterogeneous smart objects. Our proposed work ensures the continuity of service reconfiguration at the semantic level and enhances the response time through explicit semantic dependency between situations and their relevant context sources.

Lu *et al.* [5] present a core lightweight ontology for IoT (IoT-Lite) which has been extended from the standard SSN ontology [4]. IoT-Lite ontology models the key concepts of IoT environment (i.e., coverage, actuators, services, entity, and virtual entity, attribute and measure unit). This ontology allows the interoperability and discovery of sensors data on heterogeneous platforms with low complexity and fast processing time. However, IoT-Lite does not describe essential concepts such as user profiles, its preferences, explicit descriptions of user's composite situations and its context usage. This poor description may lead to a lean user experience resulting in weak service customization. In our work, similarly to that IoT-Lite ontology, we describe sensors/actuators, event, situations as semantic services to obtain the same genericity. Furthermore, we incorporate the concept of microservices (i.e., small unit of code) in these semantic services that allow different functionalities such as observing sensors, and identifying parallel situations at a semantic level in a specific smart domain. In comparison to the IoT-Lite ontology, the objective of our research is to conduct a user-centric approach, where user needs and its composite situations in multi-domain smart environments are well described.

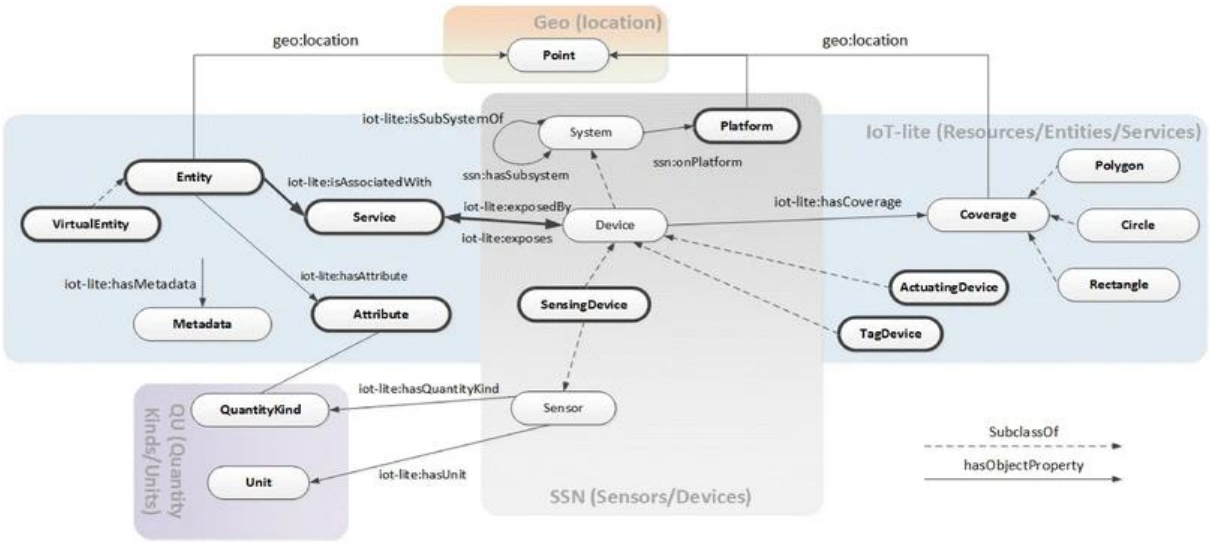


Figure 2.3: IoT-Lite Ontology model [5].

In an attempt to recognize complex daily living activities (ADLs) [58] in a smart environment, Okeyo *et al.* [59] propose an ontology-based approach for composite activity (e.g., interleaved and concurrent activities) recognition in smart homes. The knowledge bases are modeled as daily living activity

(ADL) ontology for both simple and composite activities (see figure 2.4). This approach provides a multi-agent framework to enable multiple activities to be simultaneously monitored and recognized. This framework uses SWRL with JESS engine to infer logical composite activities. However, this work lacks the concept of activity's priority that can play a significant role in human activities scheduling to determine urgent activities.

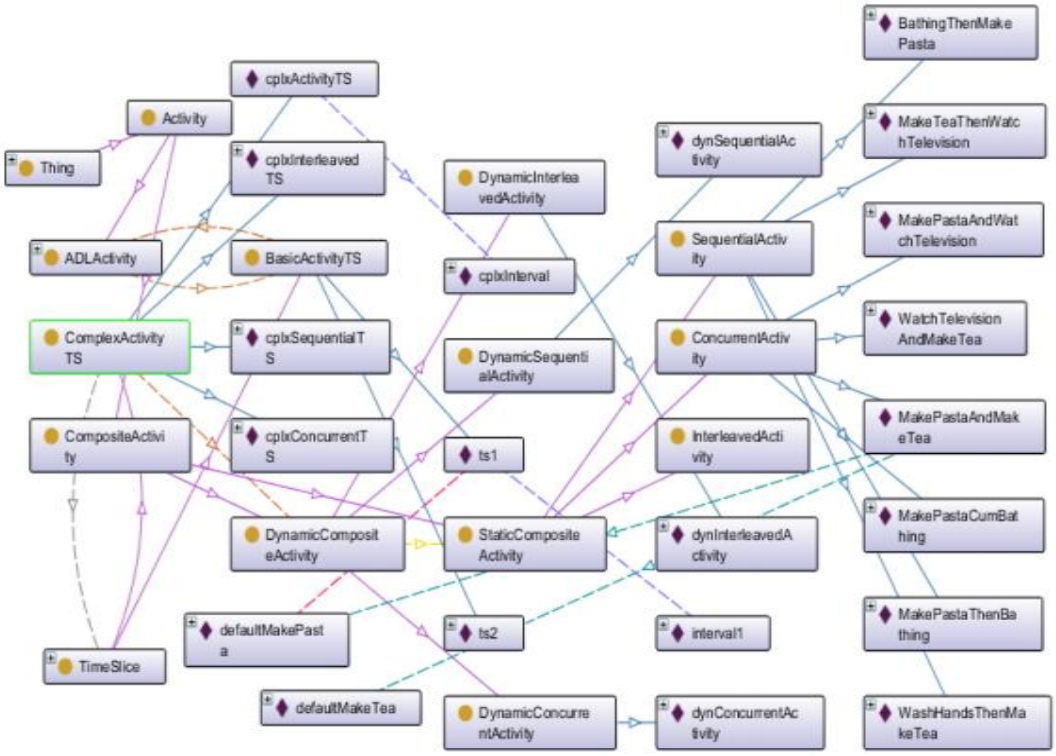


Figure 2.4: ADL ontology [59].

Ramírez *et al.* [60] introduce a graphical framework based on DSL (Domain Specific Language) for specification and evaluation of elderly/handicapped Daily Living Activities (ADL) according to the AGGIR (Autonomy Gerontology ISO-Resources Groups) grid variables. This DSL has been used for integrating spatiotemporal operators to manage time and location of composite activities and complex events within a home environment. Therefore, they define a situation as a set of occurred events (or set of composite activities as shown in figure 2.5). Three of the AGGIR variables (i.e., dressing, toileting, and transfer) have been picked for evaluation purposes. Results demonstrate the accuracy of the proposed framework to monitor and manage the occurred events and activities of the elderly. However, this DSL-based complex event has been criticized for the lack of monitoring situations of elderly outside home domains which makes the assistance of the elderly restricted in limit areas. This DSL does not handle the composition of multi-domain situation in which situations from different smart domains (e.g., health with transport) can be combined through combination operators such as parallel, sequential, recurrence, and alternative. The multi-domain situation combination can offer a new user's experience by incorporating different smart domain situations where users can manipulate and take benefit from all the existed smart-

* domains. In addition, this work lacks the concept of activity's priority that can play a significant role in human activities scheduling to determine urgent activities for elderly.

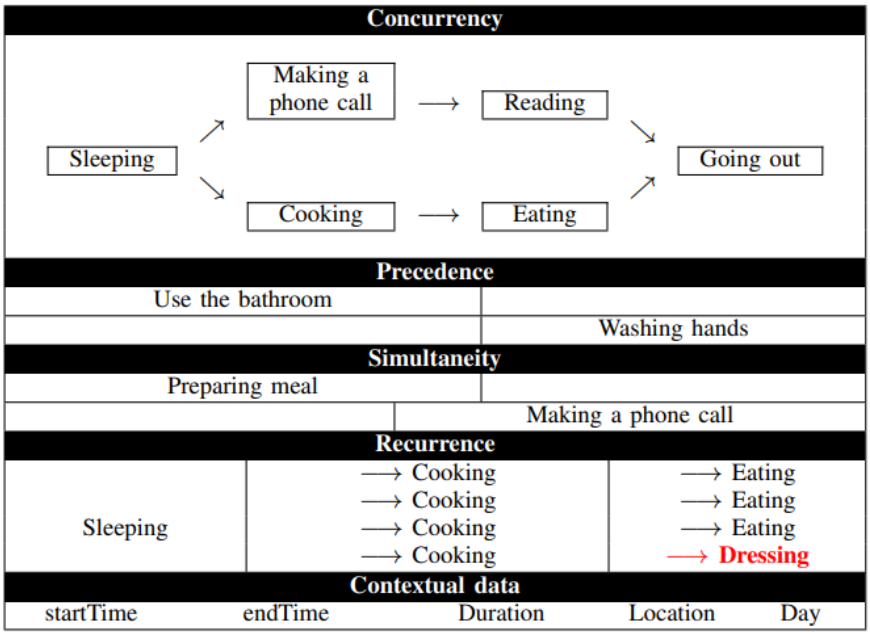


Figure 2.5: Composite activities over the time dimension [60].

2.2.3 Component-based situation modeling approaches

Other works use component-based architecture for context and situation modeling such as WComp [61], DiaSuite [62], DynamicWright [63], Kali-Smart [8] [9], and SensSocial [64] to manage continuously context changes and respond to users' needs.

Ferry *et al.* [61] propose WComp middleware that relies on context-aware component-based architecture for pervasive applications. They design events as component-based services to collect context data from different sources according to user's current situation. This paradigm ensures a flexible design model for the system through reconfiguration decisions by easily adding/migrating/removing event components. This work reacts rapidly and efficiently to deal with context changes, but lacks semantic mechanisms to handle complex events at run-time.

Kali-Smart [9] is an autonomic semantic-based context-aware middleware. It relies on component architecture to represent context information and user situation. This architecture supports dynamic application reconfiguration through automatic decision making during its running by adding/removing/migrating situation component-based services while considering context changes. Moreover, Kali-Smart is a generic and flexible infrastructure that enables dynamic management of situation components to generate reconfiguration scripts on the fly. However, this work neither handles parallel incoming events nor models composite situation.

2.2.4 Comparison and discussion

Table 2.1 is a summary of the limitations and strengths of the situation modeling approaches cited bellows.

The works are compared in terms of composite situation modeling, multi-smart domains situation modeling, situation's priority, scalability and flexibility. These works have many limitations regarding several reasons. The first reason, major solutions lack scalability (handling big volumes of context data, continuous sensor's events monitoring, and user's mobility), which directly influences the whole reasoning process. The second reason, major solutions have been carried out in supporting specific smart domain that affect flexibility under several moving locations in several smart environments.

2.3 Situations identification approaches

Research into situation identification in pervasive environments has been extensively studied by researchers. As we show above in Figure 2.1, various techniques and approaches for situation identification have been classified into two categories, namely specification-based approaches and learning-based approaches.

2.3.1 Specification-based situation identification approaches

Specification-based approaches consist of representing expert knowledge in logic rules, concepts, or probabilistic models in which reasoning engines can be applied to infer new situations from context raw data. Different approaches have been developed from earlier efforts resulting in first-order predicate logic towards more formal ontologies models. Ontologies models have shown expressive capability to support formal analysis, and provide efficient reasoning engines. This section introduces the mainstream of specification-based approaches, namely predicate logic, spatial and temporal logic, fuzzy logic, and ontologies.

2.3.1.1 Predicate logic

There are several approaches based on predicate logical rules such as formal logic models to define a situation, including Gu *et al.* [65], Ranganathan *et al.* [66], Henricksen *et al.* [40]. These approaches provide formal foundation based on rule representation, which offers a theoretical foundation for building a situation-aware system. Henricksen *et al.* [40] express situations using a predicate logic. They define a situation as logical expressions of the form $S(v_1, \ldots, v_n): \phi$, where S is the name of situation, v_1 to v_n are variables, and ϕ is a logical expression that contains free variables correspond to the set $\{v_1, \ldots, v_n\}$. The logical expression uses logical connectives such as, and ($\wedge$), or ($\vee$), and not ($\neg$), and other quantifiers. For instance, figure 2.6 illustrates an *Occupied* situation for a person. This situation describes the fact that a person can be engaged in an activity that generally should not be interrupted: in meeting or talking call.

Table 2.1: Comparison of situation modeling approaches.

Approaches		Situations Modeling Concepts			Extension Mechanism	Extensibility	Scalability	Flexibility
		Composite Situation Modeling	Multi-Domains Situation Modeling	Situations ' Priority				
Rule-based situation modeling approaches	[67]	✗	*Generic*	✗	*XML*	*Partial*	✗	✗
	[54]	*Partial*	*Generic*	✗	*Toolkit*	*Partial*	✗	✗
	[56]	*Partial*	*Smart-Health*	*Partial*	✗	✗	✓	✗
	[55]	✗	*Generic*	*Defined*	✗	✗	*Partial*	*Partial*
Ontology-based situation modeling approaches	[8]	✗	*Generic*	✗	*OWL & XML*	✓	*Partial*	✗
	[57]	✗	*Smart-City*	✗	*OWL/RDF*	✓	*Partial*	*Partial*
	[15]	*Partial*	*Generic*	*Defined*	*OWL & DSL*	✓	*Partial*	✓
	[6]	✗	*Generic*	✗	*OWL/RDF*	✓	✗	✗
	[16]	✗	*Smart-Health*	✗	*OWL/RDF*	✓	✗	✗
	[17]	✗	*Smart-Home*	✗	*OWL/RDF*	✓	✗	✗
	[5]	✗	*Generic*	✗	*OWL/RDF*	✓	✗	✗
	[59]	×	*Smart-Home*	✗	*OWL/RDF*	✓	✗	✗
	[60]	*Composite Activity*	*Smart-Home*	✗	*DSL*	*Partial*	✗	✗
Component-based situation modeling approaches	[63]	✗	*Generic*	✗	*Component*	*Partial*	✗	*Partial*
	[61]	✗	*Generic*	*Defined*	*Comp&Service*	✓	✗	*Partial*
	[62]	✗	*Generic*	✗	*DSL*	*Partial*	✗	✗
	[64]	✗	*City*	✗	✗	✗	✗	✗
	[9]	✗	*Generic*	*Defined*	*Comp & OWL*	✓	✗	*Partial*
	[68]	✗	*Generic*	✗	✗	✗	*Partial*	✗

$$
\begin{aligned}
&Occupied(person):\\
&\quad \exists t_1, t_2, activity \bullet \texttt{engaged\_in}[person, activity, t_1, t_2]\bullet\\
&\quad (t_1 \leq timenow() \wedge (timenow() \leq t_2 \vee isnull(t_2)) \vee\\
&\quad (t_1 \leq timenow() \vee isnull(t_1)) \wedge timenow() \leq t_2) \wedge\\
&\quad (activity = \text{“in meeting”} \vee activity = \text{“taking call”})\\
&\\
&CanUseChannel(person, channel):\\
&\quad \forall device \bullet \texttt{requires\_device}[channel, device]\\
&\quad \bullet \texttt{located\_near}[person, device] \wedge \texttt{permitted\_to\_use}[person, device]\\
&\\
&SynchronousMode(channel):\\
&\quad \forall mode \bullet \texttt{has\_mode}[channel, mode] \bullet \texttt{synchronous}[mode]\\
&\\
&Urgent(priority):\\
&\quad priority = \text{“high”}
\end{aligned}
$$

Figure 2.6: Situation formalization based predicate logic [40] .

2.3.1.2 Spatial and temporal logic

The spatial and temporal logic specifies its logical expressions with spatial and temporal primitives and knowledge. It has been used to representing and identifying situations based on both objects' locations and the current time. Juan *et al.* [69] use the methodology RIMER (Rule-based Interface Methodology using the Evidential Reasoning) [70], extended within a spatiotemporal logic to monitor and recognize human activities in smart homes. They have introduced two temporal operators: *ANDlater* (*i.e. A ANDlater B;* means that A is true and later B is true), and *ANDsim* (*i.e. A ANDsim B*; means that A is true and simultaneously B is true). They rely on Event-Condition-Action [67] rules to specify situations based on temporal knowledge and human activities. For instance, the following situation rule describes an urgent situation that may happen when a user fainted in a kitchen:

“**IF** at_kitchen_on **ANDlater** tdRK_on **ANDlater** no_movement_detected **THEN** assume the occupant has fainted”. Where:

- “at_kitchen_on” represents the motion sensors detect movement in the kitchen,
- “tdRK_on” represents that a person triggers an RFID sensor while passing through the kitchen door that communicates with the reception area,
- “no_movement_detected” represents no movement has been detected.

However, this work lacks other complex temporal operators such as overlaps, before, recurrence, during, equals, etc.

Michael *et al.* [71] propose a Qualitative Spatio-Temporal Reasoning (QSTR) system in smart environment for assisting the elderly at home. QSTR is an essential pillar in the field of knowledge representation and reasoning. It is used to obtain qualitative descriptions from different types of sensors deployed in smart homes, in order to monitor a spatial timeline of the elderly’s daily life activities. Figure 2.7 illustrates Allen’s temporal logic utilized to represent, constrain, and reason on temporal sequences between two events. The reasoning system is enriched with time constraints in which the activity is

represented as an event tagged with its occurrence time where each event is evaluated using a precondition (i.e., specific situation). An event has a certain life-time which represents a temporal distance from its occurrence time. Each event is expected to be identified during a certain time interval [t-A, t-B] which is predefined in its precondition (see figure 2.8).

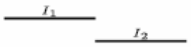
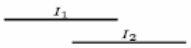
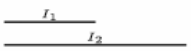
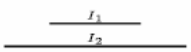
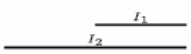
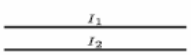
Relation	*Illustration*	*Interpretation*
$I_1 < I_2$ $I_2 > I_1$		I_1 before I_2 I_2 after I_1
$I_1 m I_2$ $I_2 mi I_1$		I_1 meets I_2 I_2 met by I_1
$I_1 o I_2$ $I_2 oi I_1$		I_1 overlaps I_2 I_2 overlapped by I_1
$I_1 s I_2$ $I_2 si I_1$		I_1 starts I_2 I_2 started by I_1
$I_1 d I_2$ $I_2 di I_1$		I_1 during I_2 I_2 contains I_1
$I_1 f I_2$ $I_2 fi I_1$		I_1 finishes I_2 I_2 finished by I_1
$I_1 = I_2$		I_1 equals I_2

Figure 2.7: Allen's Temporal Logic [71].

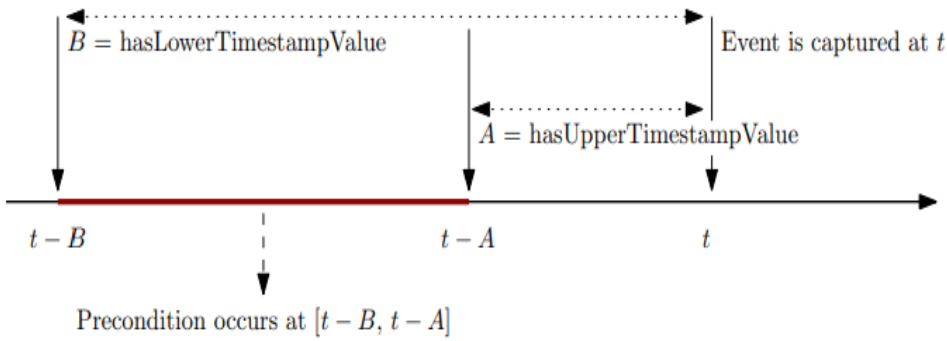


Figure 2.8: Temporal constraint between an event occurs at time t and its time interval [t-A, t-B] [71].

2.3.1.3 Fuzzy logic-based situation identification

There are some related learning techniques to identify accurately contextual situations. The fuzzy logic presented in [72] has been employed to handle context information by relying on the Fuzzy rules to classify situation rules according to fuzzy context attributes before applying the situation identification. However, this work does not offer the specification of composite situations based on the composition of atomic situations nor the definition of priority-based situations.

Previous works on logic-based situation identification techniques specify user's situations and their

preferences [40] [65] [66] [69] [70] [71] [72] using logical and temporal expressions. The studied techniques strictly depend on specific syntactic vocabularies and do not take into account all smart domains' terms and concepts. In the next section, we will describe how ontologies define the semantic description of different terms and concepts in smart domains.

2.3.1.4 Ontologies-based situation identification

Ontologies have been used widely for formal representation and reasoning on complex activities and situations in different domains knowledge through semantics relationships and formal axioms [73]. Ontology offers a hierarchically structured terminology to formally represent a pervasive environment with its different concepts such as sensors, context data, smart places, human activities, user situations, services with different QoS, etc. Different from the learning-based approaches, ontology as a knowledge-driven approach does not need any training data for a potential complex model and thus reduces the computational complexity.

Riboni et *al.* [74] adopt ontologies for ontological reasoning combined with statistical inferencing to recognize context-aware human activities (COSAR/Combined Ontological/Statistical Activity Recognition). They defined an otology that models different concepts such as human activities, person, symbolic locations, etc. This technique uses semantics relationships to infer and reason on context sensor data collected from sensors worn by a community of people performing different activities. However, this work consumes a significant time for identifying human activities which makes this work non-adaptable for real-time identification. In addition, it does not handle concurrent human activities.

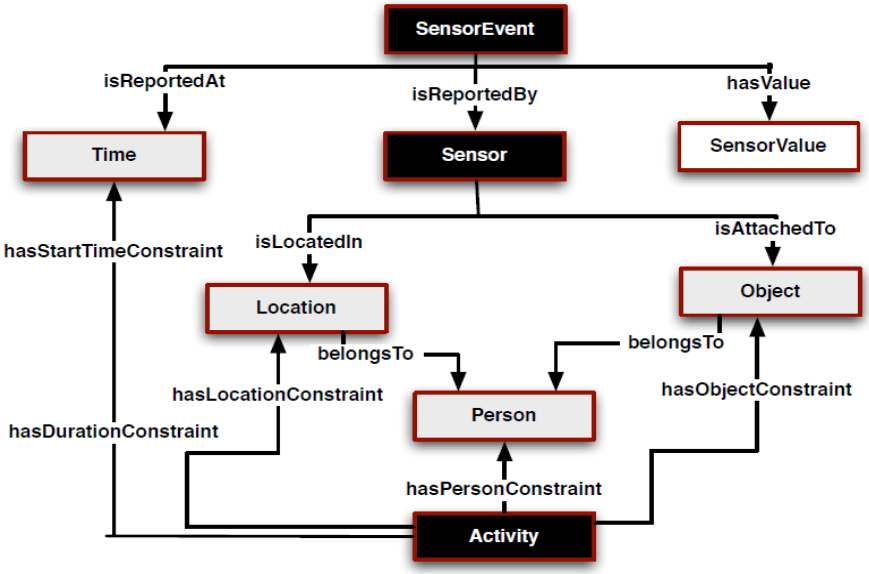


Figure 2.9: General structure of KCAR ontological model [75].

To address this issue, Ye et *al.* [75] propose a Knowledge-driven approach for Concurrent Activity Recognition (KCAR) to recognize multi-user concurrent activities. KCAR adopt an ontology model based on semantic hierarchical structure that allows the extraction of semantic properties of sensor events into

spatiotemporal and thematic aspects. In this ontology model, the human activities are described through a number of semantic properties linked with different conditions in which they are linked with sensor events. Figure 2.9 illustrates a general structure of KCAR ontological model. It receives and segments the continuous sensor events and processes the input using the ontology models through semantic relationships between conditions and sensor events (see figure 2.10). However, KCAR is used only to assist users in a specified domain without addressing the need to consider adopting multiple domains.

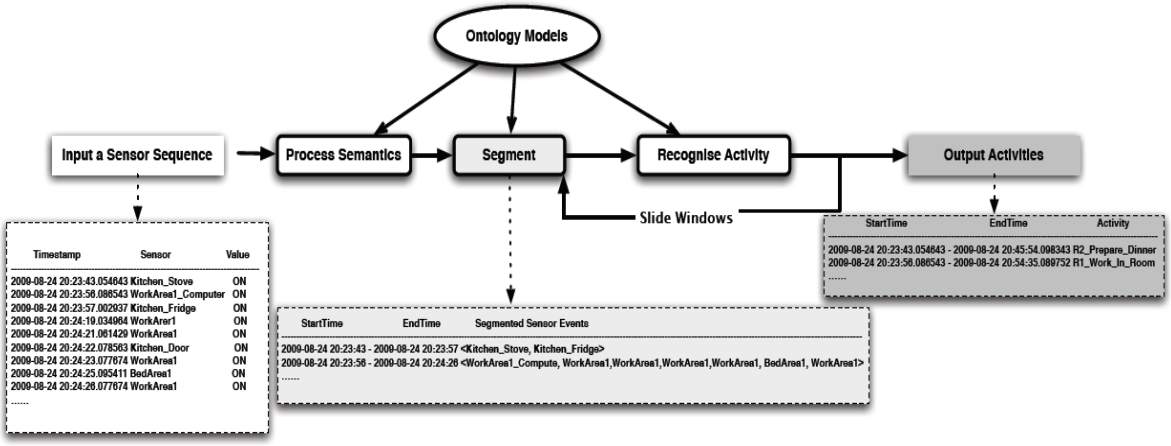


Figure 2.10: An illustrative example of KCAR segmentation and recognition [75].

Recently, Meditskos *et al.* [76] merge ontology-driven and data-driven approaches to reach a great daily life activities recognition accuracy and better activities representation. They define activities as lightweight DUL-based situation descriptors and build RDF graph with the goal to capture the contextual information between event-based observations and high-level activities. RDF graph describing partial contexts from the observation and enrich the representation of complex activities and relations among them. This work achieved an accuracy of 82% through SPARQL rules. However, this work ignored multi smart-domains within different situations and combination between them. Thus, this approach needs lots of additional work to identify properly interlinked situations of several smart-domains.

Previous models on situation-based identification ontologies [74] [75] [76] provide the semantic representation of heterogeneous context data and situations rules to assist users in their daily activities. However, existing semantic models based on predefined situation rules, not able to support undefined user situations. To provide such support, we will detail in the next section the existing learning-based situation identification techniques that enable the automatic recognition of user's daily activities without predefined rules.

2.3.2 Learning-based situation identification approaches

Recently, sensors have been evolved to more specific sensors such as 3D accelerometers, and 3D gyroscope. Advances in sensor technologies compromise the performance of specification-based solutions. In other words, it is inadequate to use only expert knowledge to identify situations from a huge number

of these specific sensors data. Therefore, learning-based techniques have been adopted to explore more complex relationships between sensor data and specific situations. The knowledge base and semantic links between sensor data and situations can be gained from the learning process that requires a large amount of training data to set up the context model and determine the correlations of relevant contexts. This section introduces the mainstream of learning-based approaches, namely, Bayesian classifier, Hidden Markov Model, Context-Free Grammar, Neural network, and Support Vector Machines.

2.3.2.1 Bayesian classifier

Jiménez *et al.* [77], propose a situation classification and identification model based on the Bayesian classifier. They use the Bayesian classifier to group and classify contextual situations into predefined classes. Despites that, this work lacks the semantic classification of situations that provides new insights on identifying efficiently hierarchical composite situations. In addition, this work has poor scalability for identifying parallel situations. On the other hand, Swapna *et al.* [78], propose a hybrid machine learning algorithm as well as situations classification techniques, which accurately improve the identification of user's situations. However, this work neither supports composite situation modeling using composition operators (parallel, sequence, recurrence, and alternative) nor provides priority mechanisms to react rapidly to urgent situations.

2.3.2.2 Hidden Markov Model

Wojek *et al.* [79] propose an approach for multi-user activity recognition in an office environment based on a multi-level Hidden Markov Model. It uses gathered audio and video features in five offices with one camera and microphone per room. These features are used as inputs to employ the multi-level Hidden Markov Model (HMM). It can track users by distributed camera networks to identify users' activities. Besides, this work has handled only a small range of users' activities (e.g. nobody in the office, meeting activity, a user has a phone call). In contrast, a real-life scenario can have more complex human activities. Although this work employs a multi-level Hidden Markov Model that can recognize plenty of users' situations, it is hard to identify a large number of real-life situations in a very specific time. In addition, this work can't handle with dynamic reconfiguration (e.g., camera unavailability is a challenging issue for HMM) on the fly during the run time which influence the flexibility of the system.

2.3.2.3 Context-Free Grammar

Ryoo *et al.* [80] propose a general approach based on a context-free grammar (CFG) for automated recognition of complex human activities. Naturally, human activities can be composed of multiple sub-action. These human activities are categorized into three categories including *atomic action*, *composite action*, and *interaction*. This approach uses CFG to formally represent complex human activities. The system follows the production rules of CFG to recognize and identify represented composite actions and

interactions. However, this work suffers from a poor grammar that can represent only small training sets where it can identify only eight interactions: *shake-hands, depart, approach*, *point*, *hug*, *punch*, *kick* and *push*. A pervasive environment is assumed to be highly dynamic in the sense that a user can move rapidly introducing new moves which may produce ambiguous interpretations in such a system.

2.3.2.4 *Neural networks*

Yang *et al.* [81] propose a learning-based approach based on multilayer neural networks (see figure 2.11) to recognize human activities using a triaxial accelerometer sensor. The acceleration data is collected from a triaxial accelerometer sensor mounted on the user's dominant wrist. Based on these data, the neural networks can generate complex discriminating surfaces for human activity recognition and classification (e.g. sitting, standing, walking, running, scrubbing, working at a PC, vacuuming, and brushing teeth). This work is limited to identifying a small range of human activities and neglecting spatiotemporal context information. Besides, as we know, human activities in a pervasive environment are several and can be sensed using many sensors that require large and deep neural networks with a significant processing time to identify such activities.

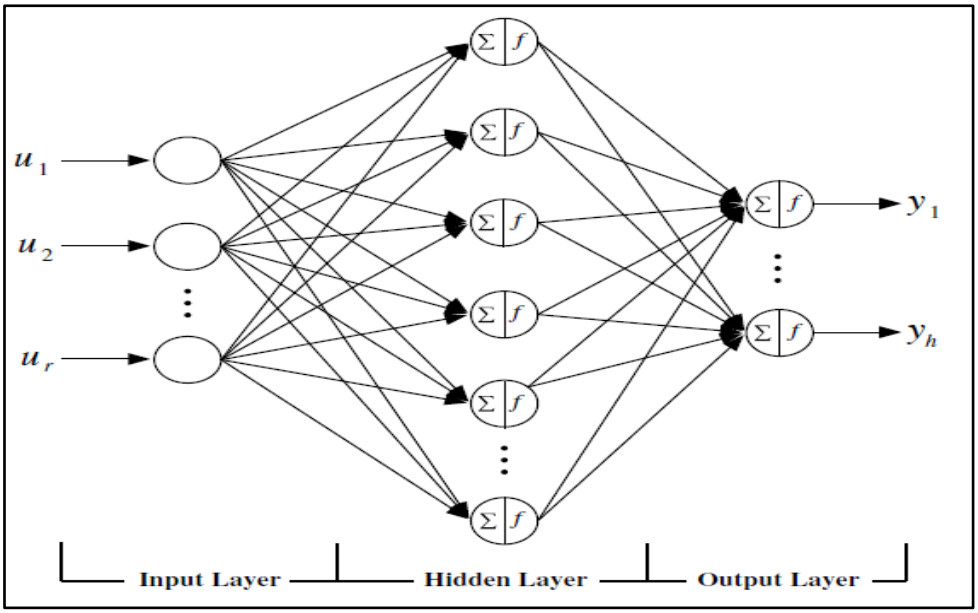


Figure 2.11: General architecture of a neural classifier [81].

As an improvement of neural network, Ronao *et al.* [82] develop a deep convolutional neural networks (CNN) to recognize human activity using a 3D accelerometer sensor (acc-x, acc-y, and acc-z) and a 3D gyroscope sensor (gyro-x, gyro-y, and gyro-z). Figure 2.12 illustrates raw time-series sensor data for both 3D accelerometer and 3D gyroscope. The CNN extracts reliable features and knowledge from raw data and classifies human activities such as sitting, standing, laying, walking, walking upstairs, and walking downstairs. The major lack of this approach is the complexity level and the computational time that increase with every additional layer in purpose of extracting more complex features.

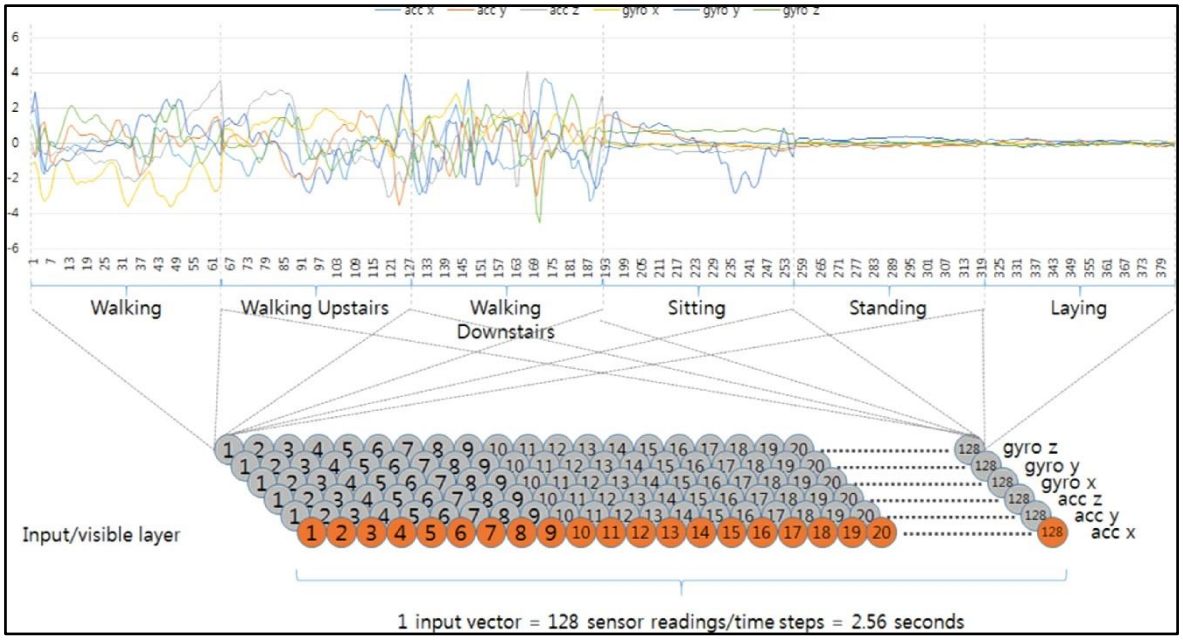


Figure 2.12: 1D time-series multi-axes sensor input signal [82].

2.3.2.5 Support Vector Machines

Recently, Support Vector Machines (SVM) have been used in pervasive computing for human activity recognition. SVMs are one of classification methods for both linear and nonlinear data. SVMs classifiers have been used to handle a wide range of feature spaces since they use overfitting protection, which does not necessarily depend on the number of features [83]. Kanda *et al.* [84] present a series of techniques to differentiate potential buyers or customers, such as window-shoppers, from those in a public space such as a shopping center. These techniques are used by a robot for anticipating people's behavior based on accumulated trajectories to offer different services for potential customers. This work uses SVMs to classify human motion trajectories based on their velocity (i.e., stop, wait, run, fast-walk, and idle-walk), direction (i.e., right-turn, left-turn, straight, wandering, U-turn, and stop), and shape features. Based on ubiquitous location sensors (i.e., GPS) and motion sensors (i.e., laser range finder) implanted in both the shopping mall and the robot, the robot can predict the areas in which people are probable to perform these behaviors a few seconds in the future. However, this work does not take into consideration users' profiles and preferences and neglect others context information such as QoS, user interest.

Chen *et al.* [85] propose a robust human activity recognition system based on Coordinate Transformation and Principal Component Analysis (CT-PCA) and Online Independent Support Vector Machine (OISVM). It discriminates different human activities including walking, static, running, going upstairs, and going downstairs. The sensed data (e.g., acceleration and orientation) are extracted from acceleration and orientation sensors embedded in smartphones. Due to the orientation variation of using a smartphone, the recognition accuracy will decrease with different orientation and position of the smartphone (e.g., holding it in the hand, in a backpack, in the pocket, etc.). CT-PCA scheme was proposed

to avoid the orientation variation effect which can give abnormal acceleration and gyroscope orientation. After the prepossessing of coordinate data, the recognition system uses OISVM with a portion of sensed data to update the parameters of the SVM on the fly in order to classify human activities. Experimental results show the effectiveness of this approach in terms of orientation variation and human activity recognition. However, this work neglects other important context data such as living behavior, aging, gender, etc. Furthermore, the feature extraction and selection are also crucial for activity recognition, which are not well represented in this work.

2.3.3 Comparison and discussion

Table 2.2 shows the comparison of the different situation identification techniques. The works are compared in terms of composite situation reasoning, real-time parallel situation reasoning, comparison techniques, scalability and flexibility. These works have many limitations regarding several reasons. The first reason, most of the existing works lack the identification of composite situations using combination operators such as parallel, sequence, recurrence, and alternative. The second reason, several works do not define a priority mechanism to advance urgent situations. The third reason, the majority of works lack a real-time parallel situation management strategy under several moving locations.

Previous studies of situation identification approaches showed that we cannot ensure a complete identification process and govern all user situations. Further, the situation identification process may be ineffective or incomplete as it could be based beforehand on predefined situation rules, defined either by experts or users. In fact, it is a tedious task for each user to define his specific situation rules in any daily life scenario. Thus, a dynamic injection mechanism for new specific situations based on recommendation techniques is needed to predict and recommend the right situation in the right context for the right person. To ensure such a recommendation, we will study in the next section different recommendation techniques that enable to enrich automatically different items in different domains (shopping, work, travel, etc.). The main goal is to build a comparative analysis of existing recommendation techniques that allows us to drive the challenges facing future pervasive computing systems and help us to ensure a comprehensive and adaptative mechanism for pervasive recommendation system.

Table 2.2: Comparison of situation reasoning approaches.

Situation identification approaches		Composite Situation Reasoning	Real-time Situation Reasoning	Validation and Comparison Techniques	Scalability	Flexibility
Specification-based						
▸ *Predicate Logic*	Henricksen *et al.* [40]	✗	✗	*Use Case (Comm-Channels)*	✗	✗
▸ *Spatial and temporal Logic*	Augusto *et al.* [69]	*Complex Activity*	✗	*Use Case (smart-home)*	*Partial*	✗
	Sioutis *et al.* [71]	*Complex Activity*	✗	*Use Case (smart-home)*	*Partial*	✗
▸ *Ontology-based*	Riboni *et al.* [74]	✗	✗	*Use Case (walking activities)*	✗	✗
	Ye *et al.* [75]	✗	✗	*Real-world Sensed Data Set*	✓	*Partial*
▸ *Fuzzy logic*	Al-Jawad *et al.* [72]	✗	✗	*RFID Synthetic Data Set*	✗	✗
Learning-based						
▸ *Bayesian Classifier*	Jiménez *et al.* [77]	✗	✗	*Real-Time Generated Data*	✗	✗
▸ *Context-Free Grammar*	Ryoo *et al.* [80]	✗	✗	*Gestures Collected Videos*	✗	✗
▸ *Neural Networks*	Yang *et al.* [81]	✗	✗	*Acceleration Collected Data*	✗	✗
▸ Convolutional *Neural Networks*	Ronao *et al.* [82]	✗	✗	*Acceleration Collected Data*	*Partial*	✗
▸ *Support Vector Machine*	Kanda *et al.* [84]	✗	✗	*Motions Collected data*	✗	✗
	Chen *et al.* [85]	✗	✗	*Acceleration Collected Data*	✗	✗

2.4 Recommendation techniques

Recently, recommendation systems have been used in several domains such as movies, tourism e-commerce, health and social network. A recommendation system is a computer program able to predict which items are the most relevant for a specific user according to its preferences. So far, several works based on different filtering techniques have been proposed in the field of recommendation systems, including content-based filtering, collaborative filtering, and hybrid filtering (see figure 2.13). In this section, we present and categorize these filtering techniques and strategies proposed in the literature.

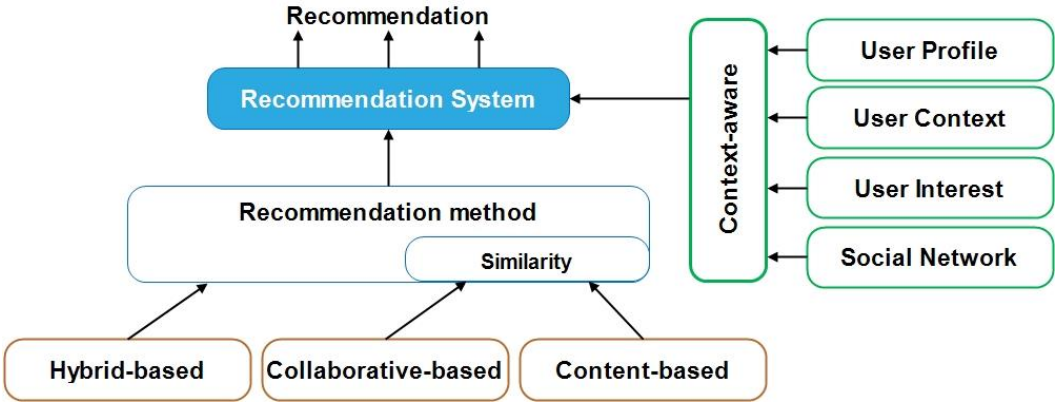


Figure 2.13: Recommendation filtering techniques.

2.4.1 Content-based filtering

Content-based filtering approaches provide recommendation based on analyzing the content of items which have been rated previously by the users and the content of items to be recommended [86]. Namely, content-based filtering approaches aim to recommend similar items to those rated items by a user in the past. These approaches can be used in different domains such as recommending restaurants, hotels, web pages, movies, and journals. Many algorithms have been proposed for content-based filtering that predict a numeric value (i.e., rating of an item) shows how far an item can interest a user.

Tkalčič *et al.* [87] propose a content-based filtering approach for images recommendation. This approach utilizes affective metadata (i.e. metadata that describe emotions of user, "*this image gives me a sad feeling*") and the content of an image to recommend. This work shows that the use of affective metadata in a content-based recommendation system for images yields a significant improvement in the performance of the recommendation much better than using generic metadata (e.g., genre). Further, a Support Vector Machine algorithm is used for the calculation of items' rating estimates, where it shows its effectiveness. However, affective metadata models have a restricted field to imply in recommendation systems.

Videos are one of the most difficult items to extract their content automatically (i.e., implicit content). More specifically, video files that have no meta-data, such as popular video sharing websites (e.g., YouTube) where users upload videos with no associated information (i.e., explicit content). Deldjoo et al. [88] propose a content-based video recommendation system that includes a technique to automatically

analyze video content and extract a set of videos' features (i.e. low-level stylistic features such as lighting, color, shadows and camera motion). These features can be used to predict intended emotional, aesthetic, or informative effects in order to provide users with recommendations. This technique extracts few low-level videos' features and neglect high-level semantic features such as, title of film, year of production, nationality, video genre, names of the actors, original language and subtitle. Despite that, the collection of high-level semantic features is more costly because it requires an editorial effort which is not always available.

To help authors find the most appropriate journal and speed up the submission process, Wang *et al.* [89] propose a content-based recommendation system for journals or conferences publication known as Publication Recommendation System (PRS). This system can suggest the best conferences or journals based on the abstract of manuscript through a priority order process. The PSR uses two methods including Term Frequency and Inverse Document Frequency (TF-IDF) and Chi-Square feature selection. The first method TF-IDF offers a way to recognize the important terms of an article by associating a weight for each term [90]. While the second method chi-Square statistic calculates the degree of dependence between the term t and a class c, such as computer journals or conferences [91]. Further, the PSR adopts a real-time online system that uses a web crawler to automatically update the training model. However, this system achieves 61.37% accuracy for paper recommendation, which is quite low. We believe that accuracy can be improved significantly if the system takes into account other criteria such as authors' preferences, impact factors, first decision, and special issues.

The major problem in content-based filtering approach in recommendation systems is the overspecialization problem. In which the system provides recommendations based only on the user's profile and neglects the user's future interests. In fact, the system retrieves items similar to those the user has already liked. In other words, it does not provide novelty from the user perspective and prevents recommending surprising items that users have not discovered yet.

2.4.2 Collaborative-filtering

Collaborative Filtering (CF) approaches are based on collecting and analyzing users' experiences in the past, then predicting users' interests in the future. As collaborative filtering approaches rely on user's feedback, they are able to recommend accurately complex items (e.g., video or music content) than content-based filtering. As we have seen in the previous section, the extraction of visual features of videos is quite difficult. However, recommending relevant items to a user based on their preferences extracted from their feedbacks appears naturally easier.

One of the major issues in CF is the sparsity of users' rating. To address this issue, Zhang *et al.* [92] present an item-based collaborative filtering recommendation algorithm based on Slope one scheme

smoothing. The Slope one scheme offers an intuitive principle to predict differential between items [93]. This approach predicts unrated items by employing two main phases to produce recommendations. The first step is calculating the average deviation of two items based on Slope one scheme. The second step is producing the prediction based on Pearson correlation similarity.

Hernando *et al.* [94] propose a collaborative filtering technique based on Bayesian probabilistic model for predicting the tastes of users. This technique relies on the factorization of the rating matrix into two non-negative matrixes (i.e., item matrix and user matrix). Unlike the classical matrix factorization, this technique associates a vector k ranging from [0, 1] with an understandable probabilistic model which allows identifying groups of users sharing the same tastes. In fact, the new factorization technique offers a significant improvement in terms of quality of prediction and recommendation accuracy. However, this technique imposes a significant computational time during the decomposition of the initial matrix into two matrixes.

Recently, deep learning-based methods find a way to submerge in recommendation systems. Wang *et al.* [95] propose a recommendation framework based on Neural Graph Collaborative Filtering (NGCF). This framework integrates user-item interaction graph into high-order connectivity graph-based embedding function preserving collaborative signal. Figure 2.14 illustrates the difference between user-item interaction graph and high-order connectivity graph. The user of interest for recommendation is user $u1$. The right subfigure illustrates a rich semantics tree knowledge that carry collaborative signal extracted from the left subfigure for the user $u1$. For instance, the behavior similarity between users can be extracted easily, the path $u1 \leftarrow i2 \leftarrow u2 \leftarrow i4$ (of layer $l = 3$) indicates that $u1$ has similar behavior with $u2$. More precisely, the path indicates that item $i4$ is likely to interest $u1$. To explore different knowledge layers, they design a neural network to propagate embeddings recursively on the high-order connectivity graph. This work was tested using three public benchmarks to verify the performance of the NGCF, which demonstrates the rationality, and effectiveness of incorporating the user-item interaction graph into the embedding learning process. However, the user-item interaction structure used for understanding user behaviors is quite poor and based only on collaborative signal. More specifically, there are many other forms of knowledge models that can be used to better understand user behavior, such as context-aware model [96] [97] [98] and social networks [99] [100].

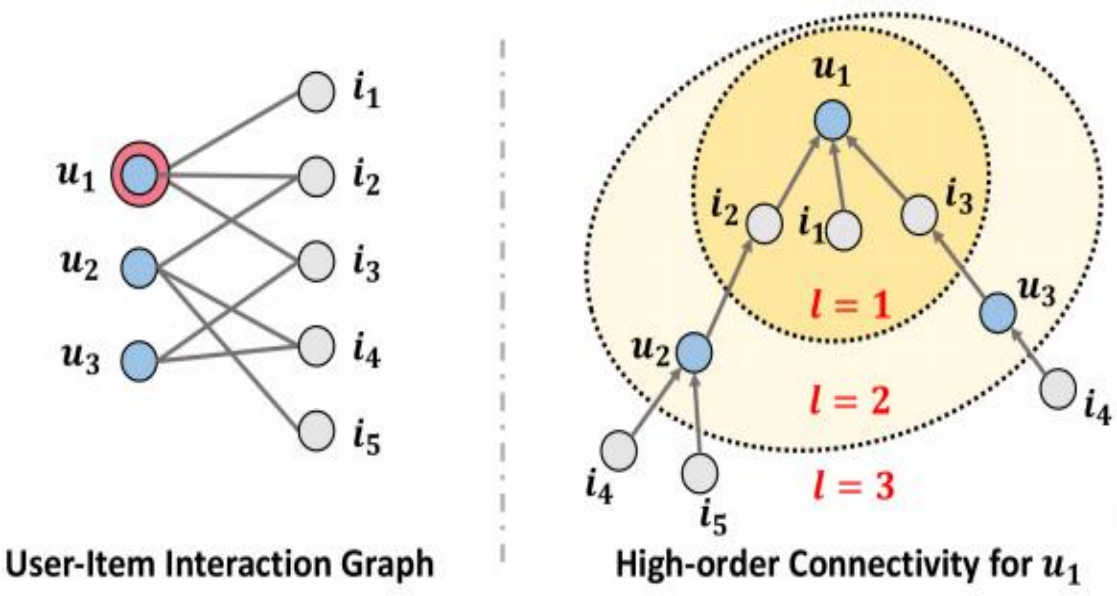


Figure 2.14: An example of the user-item interaction graph and the high-order connectivity [95].

So far, several collaborative filtering recommendation systems have been proposed and evaluated. The major problem in collaborative filtering is the cold start (i.e., when a new user has not provided enough ratings, or a new item has not been rated yet, the system is not able to generate reliable recommendations). Furthermore, the cold start problem procreates a related issue, namely the sparsity of the rating data. The sparsity measures the ratio of available ratings to all possible ratings [101]. In real recommendation systems, the sparsity is very close to 1 (100%) because users rate a very small subset of the overall database [101]. For this reason, it is challenging for CF to provide accurate recommendations addressing these problems. To deal with these problems, researchers use hybrid recommendation techniques where they combine two or more techniques, e.g., combine collaborative and content-based filtering [102].

2.4.3 Hybrid filtering

As we have shown above, each collaborative and content-based technique has its limitations, like the overspecialization, the clod start, and the sparsity problem. In fact, hybrid approaches have been proposed to enhance the recommendation accuracy and avoid the above-mentioned limitations of collaborative and content-based techniques.

Soboroff *et al.* [102] propose a hybrid recommendation technique combining collaborative and content-based filtering to get out the best of both approaches. Content-based filtering is used to build user profile from the content of past relevant documents. Furthermore, collaborative filtering is used to recommend new items based on correlation of users' rating of past items where a correlation coefficient is calculated between each user. However, this work neglects the contextual aspect of users such as location, time and role-based user preferences which reduces the quality of recommendation.

Yu *et al.* [103] develop a context-aware media recommendation platform (CoMeR) based on hybrid-processing approach. It combines three different approaches, including, content-based approach, Bayesian-classifier approach, and rule-based approach. Furthermore, it employs a context-aware model that represents the user profile including, spatiotemporal information, and user preferences. This platform supports media adaptation and recommendation for smart mobile applications. This work shows good results with precision values around 0.8 and recall values ranging from 0.63 to 0.75. However, CoMeR platform lacks flexibility. It provides a static model in which the database is updated in an offline manner. Therefore, we intend to fix this problem by using a learning process classifier that updates automatically and dynamically the database in an online manner.

Nilashi *et al.* [104] develop a hybrid method using multi-criteria based collaborative filtering for hotel recommendation (see figure 2.15). Further, to improve the recommendation accuracy of multi-criteria CF, they combined three techniques, including, Expectation-Maximization (EM) algorithm, Adaptive Neuro-Fuzzy Inference System (ANFIS), and Principal Component Analysis (PCA) for dimensionality reduction. Experimental results on the TripAdvisor dataset demonstrate the accuracy of this approach. However, items' model and users' model are updated in an offline manner, which makes this work incapable to incrementally adapt new data ratings. Therefore, incremental learning is needed to consider new updates on the fly.

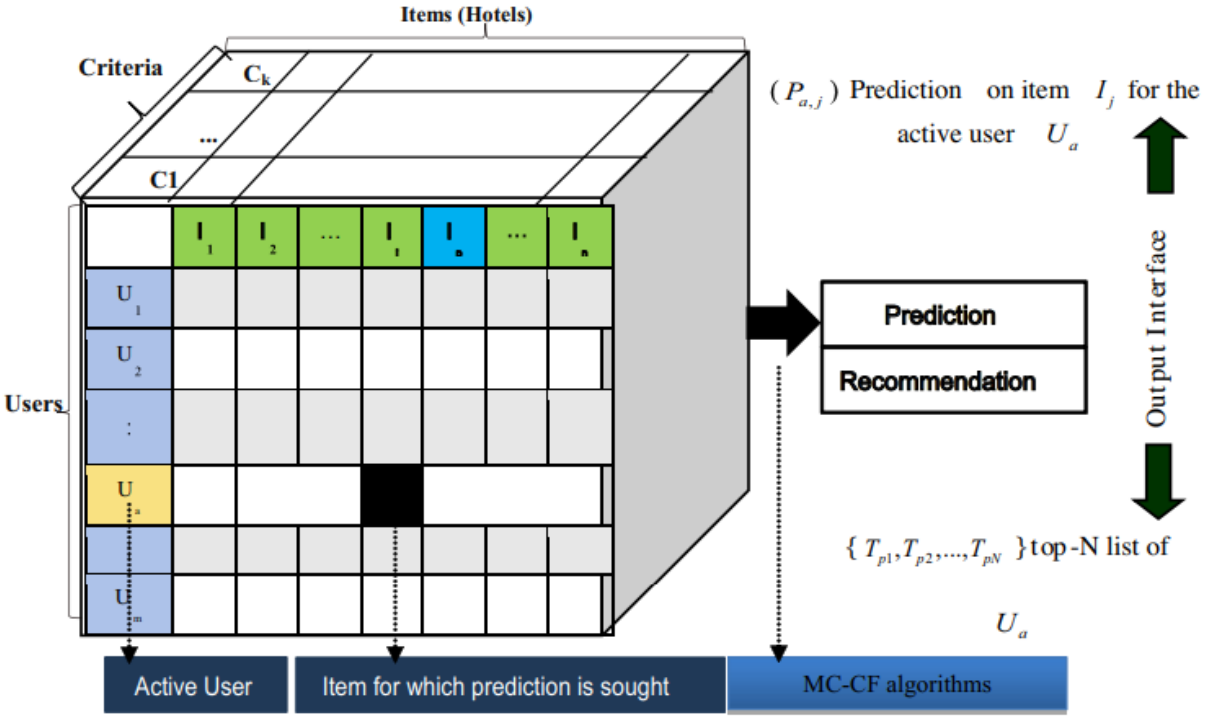


Figure 2.15: The multi-criteria CF process [104].

Services recommendation becomes an important aspect in pervasive environments due to their dynamicity and heterogeneity. Various recommendation approaches that we have studied in previous

sections focus only on rating and similarity measurement and neglect context information. Neglecting the context in recommendation systems remains inappropriate for such dynamic environments (i.e., context changing). Therefore, it will be necessary to consider context-aware approaches in recommendation systems in order to recommend more relevant services for the user's needs and usage context. We will review existing context-aware recommendation systems in the next section.

2.4.4 Context-aware recommendation system

Recently context has been widely used in multiple disciplines [96]. For instance, classic recommendation systems can take advantage of contextual information such as location, time, user activity, and social information, in order to improve recommendation accuracy. In fact, the integration of context information has been adopted by many researchers in many areas including pervasive environments, marketing, smart domains, and IoT [96].

Some works exploit the time and location information to reduce the search space that enables to enhance recommendation accuracy, namely time-aware and location-aware recommendation [105] [106]. Point-of-interest (POI) recommendation is one of the most context-aware recommendation systems that is based on spatiotemporal context information. It recommends places where users have not visited before. Yuan *et al.* [97] introduce a spatiotemporal-aware POI recommendation system to recommend a list of POIs for a given user's location at a specific time, e.g., recommend for a user to visit a nearby restaurant at midday. In fact, they develop a POI recommendation model based on human behaviors, namely, temporal behaviors and spatial behavior. Moreover, temporal and spatial behavior can play a significant role in analyzing users' activities and improve the recommendation accuracy. For instance, if many users visit a restaurant at midday but very few users visit a library at the same time, then the restaurant should be given a higher priority than the library during recommending POIs for a user to visit at midday. However, this work tends to produce a pool of unrelated POIs that the user may not continuously visit due to the lack of related associations between POIs. To cope with that issue, Zhou *et al.* [98] propose a recommendation system based on user effective Point-of-interest path (POI) that can recommend POIs by considering both possible associations and diversity features of POIs. In that case, they develop a top-k POI recommendation model based on effective path coverage to improve the performance of the recommendation algorithm.

People in real life tend to seek advice from their entourage based on social context before trying or purchasing something new. Integrating social context in recommendation systems increase significantly the performance of the recommendation in term of accuracy. Researchers have used social information (e.g., family, followed and followers, friends list, relationships, trusted and untrusted users) to improve the prediction when providing recommendations. Hong *et al.* [99] propose context-aware recommendation using a role-based trust network. They use the term *role* to model common context-

aware interests within a group of users. The user can play different roles (e.g., diner, movie fan, and reader), which may change dynamically by context changes. They use Weighted Set Similarity Query (WSSQ) algorithm to calculate the value of trust between two users in order to build the user's role-based trust network in a given context. The recommendation is made based on both calculated user's role-based trust network and user-item rating matrix (see figure 2.16). However, user context and role sets are predefined in an offline manner, which makes this work incapable to incrementally adapt to new context users. Young *et al.* [100] propose a personalized recommendation system based on friendship strength using Twitter as a source of big Social Networks Systems (SNS). It recommends interests to users by considering various characteristics of big SNS. This approach is based on social information for measuring closeness between users that is defined as friendship strength. Friendship strength is calculated using three categories of social data, including contents generated by users, relationships information, and interaction information. More precisely, three similarities are calculated for each social data category, including personal, group, interaction similarity, where the friendship is the combination of these three similarities. However, this work ignores user context and may suffer from the cold start problem especially when the user has few friendships in his social circle.

We also cite a method based on a collaborative social environment and external providers. Currently, Karchoud *et al.* [107] propose a proactive injection mechanism to inject new situations from external context sources into the user's long-life application. This mechanism uses a collaborative social environment to generate a user-friendly situation model in order to inject new situations considering user's context information. Wen *et al.* [108] propose a rule-based recommendation system for mobile applications based on user's context information. They use a semantic model to define correlations among user's context information. To perform the recommendation, they use a probability model to predict user's interests based on the context semantic model.

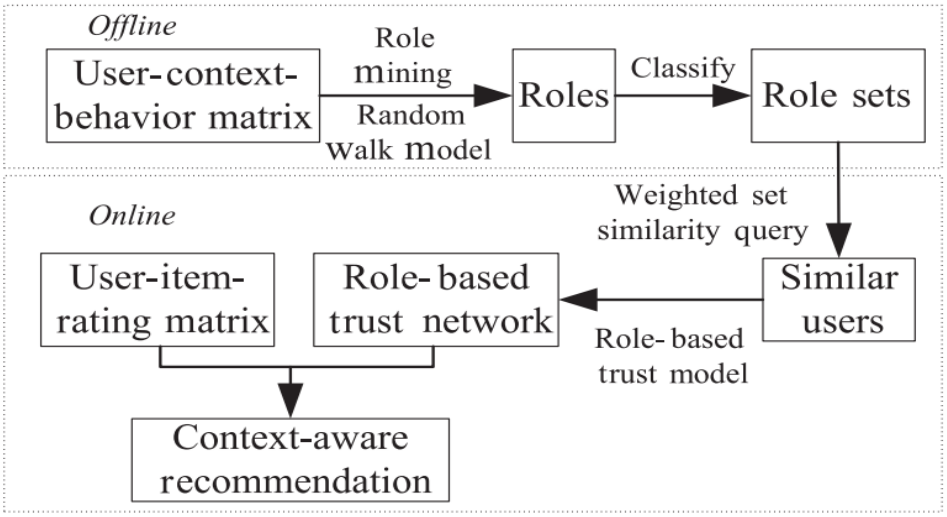


Figure 2.16: Framework of context-aware recommendation based on role trust network [99].

2.4.5 Comparison and discussion

In recent years, various recommendation techniques [86] [92] [96] [102] have been developed to improve accuracy, adaptation and flexibility using different filtering techniques and strategies. Table 2.3 illustrates a comparison between several approaches in terms of accuracy, online learning process, data set, scalability, adaptation, and flexibility. We have concluded that the content-based recommendation systems [87] [88] [89] support the cold start problem in case of updating the data set with new items. However, they suffer from low accuracy compared to collaborative filtering systems. In fact, collaborative filtering techniques [92] [93] [94] provide a better accurate recommendation but suffer from the cold start problem. The existing hybrid approaches [102] [103] [104] aim to combine different recommendation approaches to yield better recommendations across the board. However, these approaches are not flexible against context changes and evolution of user's needs. In other words, they neglect context awareness during the recommendation process. Considering context information during the recommendation process improves the recommendation accuracy. Thus, several recommendation approaches adopt context awareness [100] [105] [106] [108] to achieve a high recommendation accuracy.

Table 2.3: Comparison of recommendation approaches.

Approaches		Accuracy	Online Learning	Dataset	Scalability	Adaptation	Flexibility
Content-based filtering	Tkalčič *et al.* [87]	*63.00 %*	✗	*images*	✗	✗	✗
	Wang *et al.* [89]	*72.30 %*	✗	*Scientific papers*	✗	✗	✗
	Deldjoo *et al.* [88]	*70.50 %*	✗	*Movies*	✗	✗	✗
Collaborative-Filtering	Zhang *et al.* [92]	*80.70 %*	✗	*Movies Lens*	✓	*Partial*	✗
	Hernando *et al.* [94]	*84.60 %*	✗	*NetFlix*	✓	*Partial*	✗
	Wang *et al.* [95]	*84.60% - 97.30 %*	✗	*Gowalla - Book*	✓	*Partial*	✗
Hybrid Filtering	Soboroff *et al.* [102]	*83.50 %*	✗	*Movies*	✓	*Partial*	✗
	Yu *et al.* [103]	*80.00 %*	✗	*Movies*	✓	✓	*Partial*
Context-aware-based							
◗ *Location-aware*	Suria *et al.* [105]	*64.50 %*	✗	*Web Services*	✓	✗	*Partial*
◗ *Time-aware*	Zhong *et al.* [106]	*43.40 %*	✗	*Web Services*	✓	✗	*Partial*
◗ Spatiotemporal-aware	Yuan *et al.* [97]	*51.00 %*	✗	*Foursquare- Gowalla*	✓	✗	*Partial*
◗ role-based	Hong *et al.* [99]	*63.50 %*	✗	*Epinions - Dianping*	✓	✗	*Partial*
◗ *Trust-aware*	Young *et al.* [100]	*80.39% - 87.84 %*	✗	*Web Services*	✓	✓	*Partial*
◗ *Environment-aware*	Karchoud *et al.* [107]	-	✗	*Case Study- Tourism*	✓	✓	✓
◗ *Mobile-aware*	Wen *et al.* [108]	-	✗	*Tourism activities*	✓	✓	*Partial*

2.5 Synthesis and discussion

We have analyzed and compared existing semantic models and ontology approaches according to five criteria: 1) complex events and composite situations modeling, 2) situations' priority, 3) parallel/sequential centralized/distributed based situations reasoning and managing, 4) scalability and 5) flexibility. Most related works lack a real-time management and reaction to composite situations under dynamic user's real-life scenarios. To cope with it, a flexible and parallel model for pervasive applications is needed to respond quickly to user's current needs. Thus, a semantic model is used to describe smart domains, user's domain, user's mobility, both user's simple and composite situations, priority-based situations, and user contexts to guide lightweight and automatic reconfiguration. The objective is to ensure:

- **Semantic relationships among composite situations:** currently, there is no semantic model capable of fully representing semantic relationships among composite situations (parallel, sequential, recurrence, alternate). Existing works do not represent various smart objects and their advanced data formats (multimedia and features vector) with different mobile platforms. Providing a more detailed representation of both simple and composite situations as well as simple and complex events among heterogeneous smart objects regardless of their data source and data format that may help us to improve dynamic monitoring and continuous situations reasoning.
- **Parallel distributed ontology-based composite situations reasoning:** few existing frameworks offer the management of distributed context at the semantic level such as [9] [55]. We define three layered infrastructure (*user's constraints preprocessing layer, autonomic microservice-based composite situation processing layer and Knowledge management layer*) that includes several independent distributed events listeners and composite situations identifiers for detecting simultaneously various user's composite situations and trigger their corresponding actions services. We opt for parallel ontology-based microservices reasoning system in order to filter and factorize the relevant events and situations according to user's location and time (more details are presented in chapter 4).
- **Priority-based Situations:** existing approaches do not consider customized criteria as situations' priorities and do not cover all criteria that are relevant to situation management. In other words, they do not provide an efficient mechanism for reacting rapidly to urgent situations. To address this issue, we opt to provide a new strategy to introduce a situation

priority model based on situation category, user activity, user location, and situation coverage (more details are presented in chapter 5).

- **Context and Situations' scalability:** Most existing approaches lack a support to manage efficiently the explosion of user's situations and relations among them, and to monitor parallel incoming events for mobile applications. Besides, they are heavily linked to a specific application domain such as smart-health [16] or smart-home [17]. We opt for a generic parallel semantic-based approach in order to cover all smart-* domains and handle large context information according to the user's current needs and his context.
- **Flexibility:** In spite of the diversity and heterogeneity of users' profiles, user's domains, and smart objects, the existing works have adopted rigid strategies. To address this issue, we come up with new context-aware semantic composite situation microservices model, through a hierarchical loosely coupled infrastructure based on open semantic standards. Thus, by adding, removing, or replacing context/situation-oriented services in real-time, user's constraints and preferences are managed much better, and distributed mobile applications will be flexible enough to evolve and adapt according to the environment's requirements.

To sum up, all of the related work cited above use different techniques to deal with heterogeneity and complexity of smart environments for achieving intelligent reconfigurable mobile applications. In general, none of the existing works provides a semantic representation of high-level microservices at the run-time level. We believe that the composition of simple situations must be managed using a hierarchical model. We suggest a powerful semantic modeling approach, which has the ultimate purpose of describing new concepts and relationships that may help to overcome related works limitations: (semantic representation of both simple and composite situations as well as simple and complex events among heterogeneous smart objects regardless of their data source). For the management perspective, we opt for context-aware parallel situation reasoning approach according to the current user's needs and his context usages. This will give the application the flexibility and the dynamicity under different user's mobility and context changes. A naive solution could be the simple combination of different ontologies or context data models for composite situations modeling and reasoning, and the build of a framework based on parallel microservices. Thus, a more complex solution could be also the alignment of the best-suited top-k smart domains ontologies. Nonetheless, this would partially solve some current limitations as shown in Table 2.1. Furthermore, we aim to provide a novel flexible, modular and hierarchical loosely coupled framework that is able to adapt in specific domains (e.g., the smart home) but also can be extended into any other smart-* domain.

2.6 Conclusion

In the last years, different researchers have addressed various situations modeling, identification and recommendation techniques. Each technique has its advantages as well as drawbacks. In this chapter, we have detailed many situations modeling, identification and recommendation methods and comparisons between them. The performances of these approaches are compared through many factors: composite situation consideration, priority, flexibility, missed situations ratio and missed events ratios. From the comparative study of these methods, we conclude that the challenging problem of situation identification is still ongoing to develop a framework that can provide parallel and agile technique while keeping continuous user's evolution context, composite situation management and priority among situations. In light of this context, in the future works we try to propose a method that can provide better performances.

The next chapters will focus on our contributions in parallel semantic-based composite situation modeling and identification approach.

Part 2: Contributions

Chapter 3

A Semantic Agile Approach for Reconfigurable Pervasive Environments

Summary

3.1 Introduction

Nowadays, physical sensors and mobile applications are the key aspects in pervasive computing. Exchanging data between these two domains through a semantic level makes pervasive computing as an intelligent environment. Intelligent environments provide meaningful context data for mobile applications through several sensors. Mobile applications analyze context data and provide appropriate services for users to carry out their daily activities. However, mobile applications must generate a high semantic level of knowledge to receive and classify these raw sensor data. In fact, several sensors and mobile applications in pervasive environment are able to provide information for users. The main recognized challenges refer to the huge information exchanged in pervasive environments. This amount of information makes the management and the monitoring of sensors a very complex task. However, current user's needs usage context and interactions modalities need to be customized and respond at the right time in the right way according to user's mobility and limited resources capabilities (low bandwidth, etc.). Moreover, the huge number of users' situation rules increases the processing time of the system which leads to missing some urgent situations. Now, context-aware smart mobile applications have become increasingly demanding a lot of speed in terms of data processing and situation identification.

Most existing smart mobile applications have neglected the evolution of users' current needs and their contexts. So, a new flexible semantic conceptual model and new effective techniques are needed in

order to resolve these challenges. Therefore, our aim is to process all these data in a specific period of time by providing a new framework offering loosely coupled architecture combined with semantic reasoning. This combination would perform the connectivity between different components and reduce the processing time.

In this chapter, we present the major contributions that we have proposed. For the first, we present a smooth and semantic flexible model based on ontology to guarantee a shared vision of the conceptualization of the user profile, his preferences, connected objects, smart environments, context, situation, etc. We design new loosely coupled semantic hierarchical layers able to represent pervasive environments and facilitate the application maintenance under dynamic user mobility and limited devices capabilities (low battery, etc.). The separation of layers reduces the cost of the application reconfiguration under context changes, where each layer handles small part of the application model rather than reconsidering the whole model. In the second contribution, we fully exploit all available computing resources (*i.e.* processor cores). Thus, we propose a parallel composite situation identification framework that uses all the available processor cores to monitor parallel incoming events and identify situation rules. Also, we propose a new prioritization model assigned to parallel incoming events and situation rules to speed up urgent situations. In the third contribution, after the situation identification, it comes the phase of service orchestration to meet all the requested services for identified situations. It also provides a dynamic reconfiguration (i.e., add, update or migrate, remove of services) to ensure service continuity and modality adaptation while finding an appropriate device. To complete the task of deployment, the service orchestration relies on the Kali-smart platform [9] that scans the available devices and orchestrate the best way to deploy services. Finally, we have noticed that newly registered users in the system suffer from the cold starting problem where their profiles are empty or have few information and situations rules. Otherwise, the identification process may be incomplete as it could be based beforehand on predefined situation rules, defined either by users or by developers. Moreover, it is a cumbersome task for each user to define his specific rules in any daily life cases. Therefore, we need to develop an injection mechanism to inject new situation rules extracted from users' agendas to rapidly fill up the agendas of newly registered users. This emerges the need to recommend the right situation rule in the right context for the right person. A context-aware recommendation system for situation rules enrichment and adaptation is proposed. We aim at providing a personalized context-aware recommendation framework that can automatically enrich profile rules. We use hybrid classification methods in various levels of the recommendation process for better user profile enrichment to respond to evolving situations in different domains (e.g., shopping center, at work, etc.).

This chapter presents a global flexible composite situations identification and service adaptation approach. We first start with a general description of our main contributions. Then we will focus on the

proposed farmworker and its different modules at design and run time. Finally, we conclude this chapter with a conclusion.

3.2 Semantic agile approach: overview and contributions

Our aim is to provide a new parallel management approach for context-aware semantic composite situations in high reconfigurable smart environments. Therefore, we aim to process parallel incoming events in a specific period of time by providing a new framework offering loosely coupled architecture combined with semantic reasoning. This combination would perform the connectivity between different components and reduce the processing time. In this context, we exploit semantic information at multiple levels, i.e., from users' profiles to the deployment of adapted interactive services.

The main goal is the development of a new framework for situation management that is the intersection of the adoption of semantic web technologies and IoT domain with microservice-based approach in the field of smart-* (home, health, cities and vehicles). In this section, we briefly summarize the major contributions of this thesis. We classify them according to the area they naturally belong to:

- In the area of ontology semantic conceptualization and formalization, our contribution is:
  We conceptualize a new Multi-layered Ontology-based Composite Situation Model (Multi-OCSM) to identify and deal with user's daily lives composite situations [19]. The main properties of our model are the following ones: (i) full modeling of heterogeneous smart objects and context data, (ii) describing both simple and composite situation, both simple and complex event, (iii) including new composition operators (parallel, sequence, recurrence, and alternative) that can compose between simple or composite situation, (iv) a flexible and loosely coupled hierarchical model that can adapt rapidly for real-time context changes, (v) an extensible model that ensure extensibility with different other smart domains.
- In the area of situation-based component composition architecture, our contribution is:
  Proposed composition operators such as *parallel, sequence, alternative, and recurrence* among component-based situations provide a flexible architecture during the run-time level. To construct a loosely coupled component architecture we separate the design level from the execution level. Thus, this contribution includes the separation of the business logic application-concerns from the composition/connections of situation component concerns. We opt for using middleware for component management (adding/removing/migrating). Also, we rely on Kali-smart middleware [9] for decentralized context and situation components management. However, existing middlewares do not handle composite situations based on composition operators. Therefore, we extend Kali-smart middleware with composite situations based on composition operators at design and run-time levels. This

contribution allows the modeling of composite situations in smart-* domains and provides a loosely coupled architecture which implies better and flexible adaptability. The extended Kali-smart middleware performs the following features: (i) offering a high-semantic level on both user-defined constraints and remote smart services, (ii) providing an easy way to manage distributed mobile applications (iii) ensuring flexible and data exchanges between distributed services [109].

- In the area of situation-aware identification process, our contribution is:

  We design a novel generic distributed framework for smart environment that responds to users' context changes, current user's needs and preferences [19]. The main features of our framework are the following ones: (i) parallel monitoring of parallel streaming context data, (ii) accelerating and optimizing the situation identification process using the parallel paradigm, (iii) assisting users in their daily lives activities using parallel services-oriented ontology-based management system, (iv) providing higher efficiency in term of system's response time and identification of urgent situations in very specific time (v) supports pseudo real-time, and for instance, it can be easily integrated for real lives scenarios.

- In the area of service orchestration, our contribution is:

  We develop a new actions orchestrator module that scans the environment with the help of the kali-smart platform [9] in order to know the available devices and orchestrate the best way to deploy services. It measures the distance of available interactive mobile devices from the user's location and stores all distances in the ontology. After that, the application sends a request to the nearest available device for the user's interaction facility. It provides a dynamic reconfiguration (i.e., add, update or migrate, remove of services) to ensure service continuity and modality adaptation while finding an appropriate device.

- In the area of machine learning and recommendation, our contribution is:

  We extend our multi-OCSM for ontology-based dynamic context-aware recommendation system that enriches automatically profile's situation rules in different smart domains (shopping, work, travel, etc.). It exploits users' experiences by offering to the users a high level of comfort and better-customized user experience. A new hybrid customized situation rules' learning process is proposed to classify situation rules according to rule ontology before applying the recommendation process to achieve a high quality of recommendation considering three context categories (user preference, situation context, and device capability) [20].

3.2.1 A Multi-layered Ontology-based Composite Situation Model

Real-life scenarios and user's daily lives situations are getting more and more complex in recent trends. Moreover, the enormous and heterogeneous context data collected by different sensors embedded in smart spaces must be able to be quickly exploited with distributed devices. A formal model is needed to conceptualize semantic aspects of heterogeneous context data and provide advanced reasoning capabilities for real-life scenarios and complex situations. We provide a novel multi-layered ontology called Multi-OCSM (Multi-layered Ontology-based Composite Situation Model). The main purpose of this ontology is to describe both simple and composite situation rules and its semantic description including composition operators (parallel, sequence, recurrence and alternative), prioritized situations rules, and reconfiguration actions in order to ensure service continuity and immediate response time. This ontology facilitates the management of a large number of heterogeneous smart objects and services, semantic features related to the context data for the interpretation of parallel events (e.g., a GPS sensor that provides coordinates, is semantically linked to give many interpretations such as interpreting these coordinates as a presence detection event or a user location change event), interoperability of different incoming events and highlights the semantic rules of user.

3.2.2 A middleware for managing composite situations

Several middlewares [8] [10] [11] [18] have been proposed for the design and management of context-aware mobile applications. However, these middlewares do not handle composite situation modeling through composition operators in which the adaptation model may trigger different actions for composite situations rather than simple situations (i.e., simple situations that compose composite situation may have different actions). For instance, a diabetic who has a not so good diet situation that may be modeled as a composite situation, which consists of a high glucose recurrence situation. The high glucose situation, without the recurrence composition operator is considered as a simple situation. A simple high glucose situation results in injecting insulin action service. In contrast, the composite situation results in sending diagnostic results of the patient to the doctor. The variation of triggered actions regarding simple/composite situations increases the complexity of adaptation process. Therefore, we introduce a composite situation-oriented component architecture for Kali-smart middleware that allows context management of component-based services on available distributed devices and ensure long-life applications. Composite situation-oriented component supports four composition operators including parallel, sequence, alternative, and recurrence. Nevertheless, with the huge number of users' situations, the management and reconfiguration of applications during the execution time regarding the design model requires more computational resources and high reconfiguration cost which are not always available. Thus, we introduce a flexible model that separates the design model from the execution model in order to adapt and improves the reusability and extensibility of deployable components.

3.2.3 Parallel context-aware semantic-based situations identification framework

Few years ago, factories have tried to build processors that offer better performance for users by increasing their speed. Now, the majority of factories try to add more cores so that systems can execute more than one task at time in different processor cores (i.e., parallelism). We propose a novel generic framework for parallel composite situations identification process for smart environments, which describes situation rules using context-aware microservices-based approach. We start with a generic set of situation rules and transform them into a factorized tree to support parallel events monitoring and relevant composite situation reasoning. We explicitly separate between the event, situation and action layers and generate a very weak coupling between the microservices in order to achieve reconfiguration scale at different levels, ranging from atomic component/service to composite components/services or complete application reconfiguration. Moreover, the weak coupling between layers speeds up the process of situation reasoning by launching simultaneously independent parallel event/situation microservices in different processor cores. The more cores a machine has, the more events detection and situations identification tasks can be performed simultaneously.

We develop a framework prototype for the parallel composite situations reasoning and services reconfiguration based on the Multi-OCSM. The prototype enables the filtering and the factorization of relevant situation rules as a dynamic context monitoring, parallel composite situations reasoning and actions services reconfiguration. This process relies on Java 8 concurrency APIs to exploit the power of multithreading and by receiving concurrent events and process them at the same time to make sure that the system can identify composite situations (parallel, sequence, recurrence, alternative.).

3.2.4 A modular flexible orchestration of services

Our framework is a modular and flexible system for the rapid orchestration of mobile and interactive services. It generates adaptive components representing the initial configuration based on the user's domain by exploiting semantic situations rules. Also, it offers a good mechanism to add/remove/update easily one or more events/situations/actions components to deal with new identified situations or context evolutions based on the semantic reconfiguration engine. New application reconfiguration scripts may be generated dynamically as soon as the new specification of situations rules filtering or user's domain are changed. This is achieved with the help of the Kali-smart platform [9] that is able to identify the available resources and orchestrates the nearest available terminals to deploy the action components.

3.2.5 Machine learning for situation rules recommendation and enrichment

The cold start facing adaptive systems based on users' agendas and rules is a classical problem to be solved imperatively. Recommendation of situation rules is one of the best techniques to inject and full the user's agenda automatically. Nowadays, recommendation systems are becoming extensively used for rule-based recommendation. In fact, a rule-based recommendation system is used to suggest meaningful rules and has the ability to guide users to useful and relevant situation rules, which meets their needs in a large space of available rules. We provide a personalized context-aware recommendation framework that can automatically enrich profile rules in order to respond to evolving situations and thus comply with an auto-complete of user's daily life requirement (e.g., his needs in the shopping center, at work, at home, etc.) in different domains. Our recommendation system considers three aspects: the *user's contexts*, *preferences* (e.g., preferred activities) and *constraints* (e.g., available time) while respects the device constraints (e.g. device characteristics or device capacity). We extended our multi-OCSM ontology for context-aware recommendation system. A new situation rules' learning process based on extended multi-OCSM is proposed to classify situation rules according to rule ontology before applying the recommendation process to achieve a high quality of recommendation considering three context categories (user preference, situation context and device capability).

3.3 Proposed framework

As noted in chapter 2, the issues related to composite situations modeling and processing are currently curbing the management of context-aware distributed smart mobile application designed to assist users in their daily activities. The identification of a composite situation plays a primary role in this context. It becomes more complicated due to the availability of a huge number of different and heterogeneous connected objects (IoT). However, with the large variety of user situation rules and multiplicity of devices in different smart domains, the aspects of Event-Conditions-Actions (ECA) [67] are diminished, and they lose their performance and effectiveness in terms of the response time.

To cope with these problems, we propose an ontology-based parallel context-aware reasoning microservices engine approach within a generic framework. The novelty of our approach is to provide: 1) parallel identification process of composite situations combined from multiple deduced atomic situations based on a generic ontology of smart domains and 2) factorized context-aware tree representing relevant situations regarding the specific smart environment. These two elements improve the performance of the identification process of the user's situations while detecting a huge number of parallel incoming events. When a situation is identified, the system calls the reconfiguration generator, which is responsible for triggering a list of actions (e.g., adding/deleting/updating a service or migrating of services from a device to another in the context of the users' mobility).

We propose a matching and filtering techniques to simultaneously manage spatiotemporal situation rules (i.e., rules required user's location and/or time). In other words, the system matches and filters situation rules according to user's current location and time. The filtered situations rules are transformed into a factorized tree based on the explicit links between situation rules and context attributes. Next, the factorized tree is explored in order to submit and deploy parallel autonomic microservices-based events listeners and situations identifiers. Finally, the application can yield a set of situations and their associated action services in an attempt to serve several users at the same time.

3.3.1 Framework's general architecture

This section presents a novel generic framework for a smart environment that responds to the current user's context usage, needs, and preferences in a parallel and intelligent way. The main purpose is to assist and support users in their daily activities and in urgent situations with a parallel micro services-oriented ontology-based management system. It provides higher efficiency of the user's situation identification process and efficient management of the user's situations over heterogeneous smart objects. As shown in figure 3.1, the architecture is made of three interconnected layers where each layer uses context data and services provided by other layers.

The main purpose of the **Knowledge management Multi-OCSM** layer is to describe composite/atomic situations and their related concepts in smart domains. It provides OWL (Web Ontology Language) description of different user's profiles and semantic services using a multi-layered Ontology-based Composite Situation Model (Multi-OCSM). This layer includes:

- **User's profiles registry:** it provides a semantic description of the user's profiles and his preferences.
- **Situation rules registry**: it provides all the situations rules set categorized by smart domain, location, time and situation type (urgent or normal).
- **Services registry**: it provides a set of all functional services, where these services will be deployed according to each identified situation, taking into account the semantic link between the situation and its suitable action services.
- **User's domain registry**: it provides relevant information about the available smart and interactive devices in the managed mobile application such as the status of device (used or not, context information, etc.).

The **Autonomic microservice-based Composite Situation Processing layer (At run-time)** contains several components for identifying situations based on user-defined rules. This is done through situations identifiers providing the user with appropriate action services according to their identified situations. It consists of four main components to identify the user's situations:

- **Situations Manager** is responsible for filtering a set of relevant situations rules according to user's current location and time from the situation rules registry followed by the situation rules factorization. This component outputs the factorized tree of situations and its related common context attributes.
- **Task Manager:** is responsible for encapsulating all situations and their context attributes from the factorized tree into several situations' identifiers and events listeners respectively.
- **Event Listener:** is responsible for monitoring a specific user's context data (events) using a specific sensor, then sends them to their corresponding situation identifier components in order to be semantically interpreted and analyzed.
- **Situations Identifier** is a parallel reasoning engine with two kinds of microservices: simple situations identifiers and composite situations identifiers. They check the situation rules and identify current situations.
- **Service Controller:** It generates and orchestrates the appropriate action services according to the identified situations and available mobile devices.

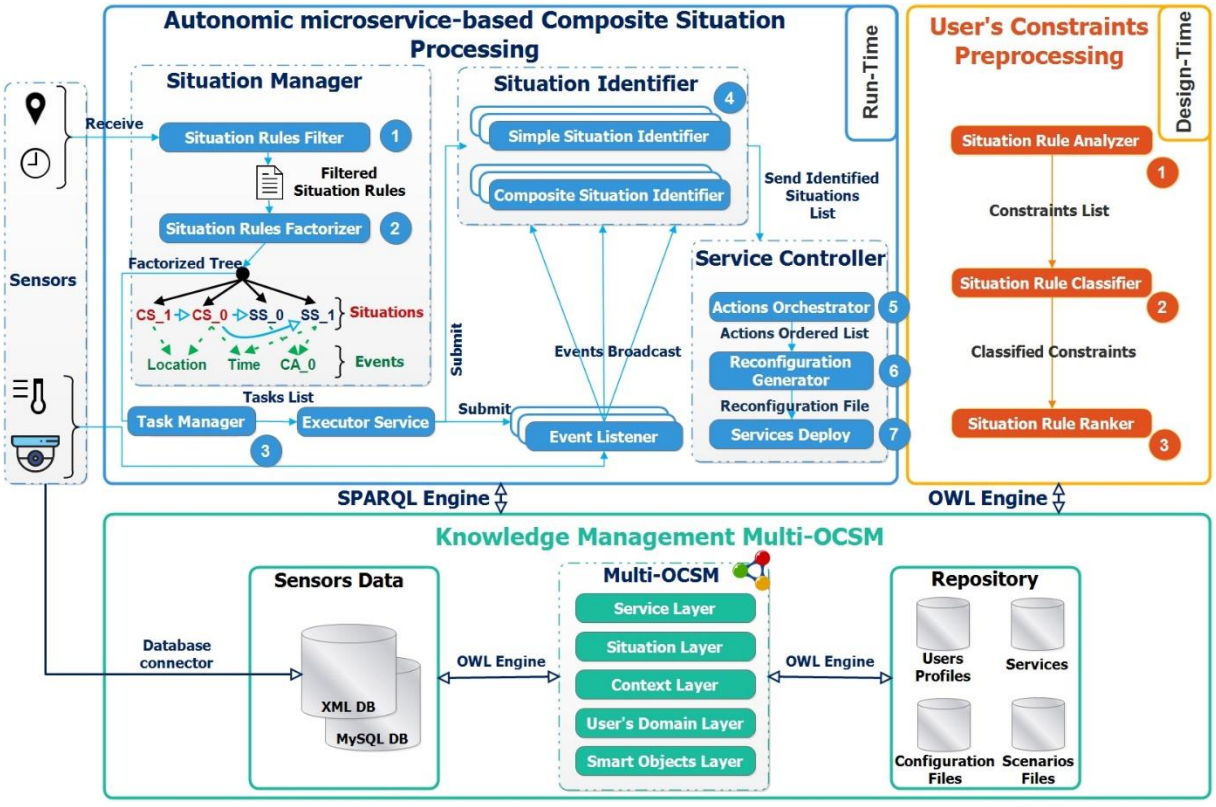


Figure 3.1: The general architecture of the framework.

The **User's Constraints Preprocessing layer (At design-time)** that provides a semantic user's situation rules preprocessing based on spatiotemporal user's specified constraints.

- **Situations Rules Analyzer** is responsible for receiving and understanding the user's rules specified using GUI and translates it into its internal knowledge structure.

- **Situations Rules Classifier** is responsible for grouping situations rules related to their similar location, time and smart domain.
- **Situations Rules Ranker** is responsible for providing a ranked list of rules according to expert's defined priorities.

The novelty of the proposed framework architecture is to provide: (i) a Knowledge Management module for preprocessing of raw sensor data using Multi-OCSM ontology, (ii) a User's Constraints module that easily defines and configures composite situation rules by combining several simple situation rules at design-time, (iii) an Autonomic Microservice-based Composite Situation module for events monitoring and situations management and reasoning at run-time.

We detail in the next section the functional model of the User's Constraints Processing module and the Autonomic Microservice-Based Composite Situation Processing module.

3.3.2 Functional model of the User's Constraints Processing (at design-time)

At design-time level, as shown in Figure 3.1, this module is used to build different usage contexts (i.e. context situation rules and their associated actions) to specify and define user situation rules that represent user's daily life activities. It consists of the following steps:

- **Step 1. Specification and analysis of situations rules.** The first step relates to the specification of situations rules through a graphical tool in which situation rules are specified and defined based on the Event-Condition-Action (ECA) model. With the new composition operators (i.e., parallel, sequence, negation and recurrence) among existing simple situations, users can create a hierarchy between ECA models and build composite situation application model. The ECA model can describe a situation rule using different concepts including simple/complex events, simple/composite conditions, and action services. The Situation Rule Analyzer component analyzes and checks the semantic coherence of user-defined situation rule. If there is no constraint violation, the Situation Rule Analyzer component translates checked situation rules to OWL individuals using JAVA code in order to unify and insert them into Multi-OCSM OWL model.
- **Step 2. Classification of situations rules.** The Situation Rule Classifier component classifies contextual situations rules based on their similar location, time and smart domain (smart-health, smart-tourism, smart-transport, smart-home, etc.). This classification improves the extraction of relevant situation rules and builds a global generic situation profile. Also, it enriches Multi-OCSM ontology with rich situation classes that helps to recommend and identify relevant situation rules according to user's current domain and needs.

- **Step 3. Ranking of situation rules.** The Situation Rule Ranker component classifies the situation rules into three categories, urgent, important and normal situation. The classification is performed through the calculated priority score, which falls into one of the predefined priority score intervals of each category. The calculation of priority score is based on four criteria including situation category, user current activity, user location, and situation coverage. The Situations Rules Ranker component sorts the situation rules from the highest to the lowest priority score. At run-time, during the management of situations, situations with high priority will be handled before situations with low priority.

3.3.3 Functional model of the Autonomic microservice-based Composite Situation processing (at run-time)

This module performs the identification of user's situations during his daily activities. It ensures the monitoring and analysis of incoming events and context changes during the run-time. This module is based on real-time agile situation identification mechanism using an autonomic microservices architecture. It consists of the following steps:

- **Step 1. Dynamic filtering of relevant situation rules.** The Situation Rules Filter component provides relevant user's situation rules list filtered by the user's current location and time. Initially, a list of the user's relevant rules contains a set of anytime and anywhere rules (i.e., situation rules that do not requires location and time constraints). Then, the Situation Rules Filter component refers to the situation rules registry to filter a list of relevant situation rules that match the user's current location and time.
- **Step 2. Construction of dynamic situation rules-based factorized tree.** The Situation Rule Factorizer component transforms the relevant situation rules list into a factorized tree. The factorized tree consists of three layers: composite situations layer, simple situations layer and common context attribute layer for all situations. The factorized tree is sent to Task Manager Component.
- **Step 3. Creation of events and situations microservices from the factorized tree.** In this step, the Task Manager component encapsulates each event monitor entity and situation identifier entity in a single deployable unit of code known as a microservice entity. Two types of microservices are used, the event listener microservice for monitoring sensed data and the situation identifier microservice for identifying user's situations. The Event Listener is continuously monitoring the surrounding environment to acquire dynamic context data and preprocesses context information from smart sensors. The Event Listener receives new context information (i.e. newly detected events or contextual changes), converts them into a triple

pattern in OWL format, and inserted them in our Multi-OCSM ontology. Finally, the Event Listener component dispatches the detected event to the appropriate situation identifier components.

- **Step 4. Parallel Identification of simple and composite situations.** In this step, the Situation Identifier component receives parallel incoming events from Event Listener component in order to analyses semantically context changes. The incoming events are checked semantically according to the predefined situation rules. The situation is identified if and only if the situation rule is checked. In this case, the Situation Identifier component dispatches the identified situation to its appropriate Composite Situations Identifier component. The Composite Situation Identifier component infers new composite situations from identified simple situations. Finally, the system sends all identified situations to Service Controller module to trigger appropriate action services.
- **Step 5. Orchestration of the situation's actions.** The Actions Orchestrator component scans the smart environment with the help of Kali-smart middleware to select available services and devices in order to orchestrate the best way to deploy action services.
- **Step 6: Services reconfiguration and deployment.** The Reconfiguration Generator component receives the orchestrated actions provided by the Action Orchestrator component and generates the output reconfiguration script via the semantic links between situations and services.
- **Step 7: Services execution.** The Services Execution component consults the available microservices (e.g., the availability of light switch service) located in services repository to execute them.

3.4 Conclusion

This chapter is devoted to exhibiting generally the contributions of our thesis with its various axes. First, we presented the necessity to have a semantic model that conceptualizes the environment. So, we proposed a new ontology model called Multi-OCSM that conceptualizes composite situations and different concepts in pervasive environments. Then, in the need to have an efficient and powerful system that can monitor context changes and identify a huge number of situation rules, we presented generally a new agile framework for composite situation identification process based on the multi-OCSM ontology. The strength of our framework is the simplicity of the situation identification mechanism basing on a modular structure. Therefore, we designed a services' orchestration module in order to achieve agility in distributed context-aware situation-driven mobile applications using our approach. Finally, we showed that newly registered users in the system suffer from the cold staring problem where their profiles agendas

are empty or have some few information and situations rules. Therefore, we presented a new context-aware recommendation system for situation rules enrichment.

The next chapter is dedicated to present the multi-layered Ontology for Composite Situation Management model with its different semantic concepts and relationships between them.

Chapter 4

Context-aware Multi layered Ontology for Composite Situation Model in Pervasive Computing

Summary

4.1 Introduction

Currently, the smart connected objects (IoT) are multiplied and distributed by different providers. It leverages heterogeneous smart objects with various communication protocols (i.e., ZigBee, 3G, Wi-Fi, etc.) and a variety of context data types (i.e., text, audio, image, video, etc.). In addition, different languages express the client's needs and constraints. A large number of daily life activities and urgent situations are increasing, thus a unified semantic model is needed in order to easily identify situations regardless of their natures, their expressed language, and their context sources. To identify user situations in pervasive computing, there are various approaches available such as specification-based approaches and learning-based approaches [77] [75] [85]. Ontology-based approaches have been widely used for modeling a set of concepts in the domain of pervasive computing, which represents physical concepts (i.e., smart object, device) and abstract concepts (i.e., context, situation, service).

The existing semantic ontologies models presented in chapter two offers appropriate capabilities, including the semantic description of context and user's needs. However, the proposed ontologies do not deal with various composite situations (*parallel, sequential, recurrence and alternative*), nor do they handle multiple devices and smart objects at run-time. In real-life at a semantic level, situations have different

priorities ordering from low priority, normal priority, and high priority where high priority situations are handled before the situations having low priority. Nevertheless, existing ontologies neglect the situation priority concept under several user's locations (*moving through different smart domains*) during the run time. This may make reconfiguration applications suffers from reacting rapidly to urgent situations. This issue is clearly noticeable when an urgent health situation (e.g., heart-attack health situation) needs to be firstly handled during a specific deadline defined by an expert. In fact, we must ensure continuous monitoring and quick processing time regardless of user location and context changes in order to rapidly perform urgent situation. The semantic contextual description of smart services is an important aspect for managing a large number of heterogeneous smart objects, managing semantic user-centric situations, and providing suitable services according to users' needs.

In this chapter, we designed a new model called Multi-OCSM (*Multi-layered Ontology-based Composite Situation Model*) [19] which is based on novel semantic microservices-based user-centric composite situations modeling and reasoning for pervasive applications. We offer a smart framework for efficient situations management based on the semantic of incoming events to achieve context-aware smart mobile applications. This ontology aims to model smart objects, smart domains, context, simple and composite situations, and applications services. In fact, real-life situations are more complex: parallel, sequence, interlacing, etc. This ontology has new modeling of composite situations pertaining to smart environments using specific combination operators: parallel, sequence, recurrence and alternative. The combination of these operators allows creating real-life composite situations.

4.2 Multi-OCSM ontology

Multi-OCSM ontology [19] is a novel generic ontology that can be applicable in different smart domains (health, home, cities, car, office, etc.). This ontology extends standard ontologies including the Semantic Sensor Network (SSN) ontology [49], the DOLCE Ultra-Light ontology (DUL) [52] and the OWL Service (OWL-S) ontology [110] which are already reusable. In fact, it includes concepts that model all the functionalities of smart environments (i.e., event services, situation services and action services). Multi-OCSM is a formal context and situation-based ontology that can play a vital role in facilitating reasoning by formally representing domain knowledge. It considers many concepts in smart space such as services, devices, places, context-awareness, events, situations and user profiles. The main objective of our proposed Multi-OCSM (MULTI-layered Ontology-based Composite Situation Model) ontology is to:

1) describe heterogeneous smart objects as well as their various protocols (*ZigBee, 3G, Wi-Fi, etc.*) and data formats (text, audio, image, video, etc.),
2) provide classification of simple and composite situations according to the context of user for efficient and effective situations managing,

3) provide formal semantics for user's domains structure as well as relations between them,
4) represent semantic service information such as service role (adaptation, interaction, context analysis, etc.), service category (health, home, transport, tourism, etc.), semantic of inputs/outputs,
5) unify the properties and constraints of user for easier managing and sharing,
6) automatic deduction of service events and actions corresponding to semantic specifications of user's constraints.

Moreover, this ontology accelerates the process of identifying relevant situations by matching user profiles and context information on a semantic level. A semantic classification of situations rules is useful for intelligent decision-making process according to the current user's needs and its context. The purpose of situations classification is to improve the efficiency of the identification process by prioritizing them according to one or more criteria such as situation category, activity of user, user's location and situation coverage. Our ontology aims to capture the user's context at the semantic level. Figure 4.1 presents an overview of Multi-OCSM layers for pervasive applications.

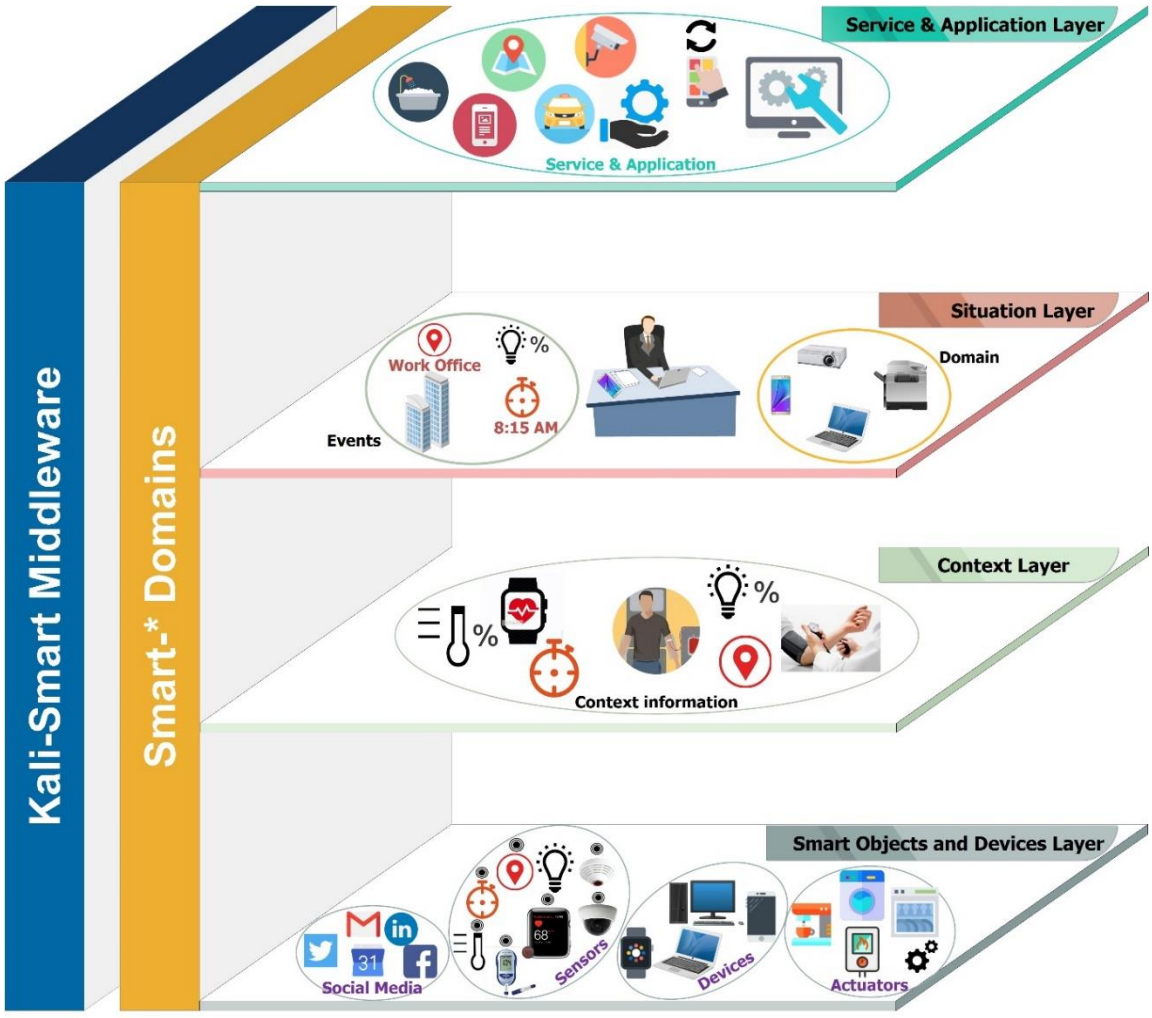


Figure 4.1: Multi-OCSM ontology layers.

Multi-OCSM ontology can be used for several ends. Members of a given community of a specific user's domain can communicate and share knowledge between them using Multi-OCSM. Also, it can be used to develop distributed mobile applications for several purposes: security of smart spaces, energy-saving, health monitoring, etc. To conceptualize our ontology, we have used a combination of the top-down and bottom-up approaches. First, we have created for each layer its main concepts to build the core model of Multi-OCSM. Then, we have appropriately identified their specialization and allocate them into their taxonomy of smart domain concepts. Finally, we have created a logical relationship between those concepts to enable reasoning using our ontology. The Multi-OCSM consists of four layers as follows:

- **Layer 1 (smart objects and devices layer):** this layer contains several physical/logical sensors and devices depending on its type and its features. These sensors serve as acquirers of raw contextual data, which will be processed by layer 2 to translate it to semantic context data.
- **Layer 2 (context layer):** this layer is used to classify contextual information provided by the smart objects and devices layer using multi-dimensional aspects such as environment context (e.g., location, time, network) and user's context (e.g., user's role such as a student, employer or teacher, user's current location, time and user's agenda). Our ontology uses context information to identify the corresponding user's situations.
- **Layer 3 (situation layer):** a composite situation is specified as a combination of sub-situations based on the semantics provided by the context layer. Within situations, diverse relations, such as parallel, sequence, alternative and recurrence can be applied.
- **Layer 4 (service and application layer):** as previously presented, each situation identified by the situation layer corresponds to a set of appropriate action services. Each service is described by its distributed microservices on mobile applications. It is possible to dynamically adapt to an application by reconfiguring its microservices according to the relevant context changes.

The proposed layers are applicable in different smart domains (*health, home, cities, car, office, etc.*) through Kali-smart middleware [9]. Each smart domain defines specific situations described in a domain-specific ontology and covers specific context data. We use Kali-smart for dynamic reconfiguration that enables adding/removing/updating services on distributed devices. The middleware is deployed on all mobile devices involved in the application. At the end of the design process, we got our Multi-OCSM ontology as shown in figure 4.2. The ontology is defined in five key classes: **Context** (a), **Situation** (b), **Smart Object** and **Device** (c), **Domain** (d), and **Service** (e). These classes represent generic concepts, which can be used in any smart context-aware environments that aim at providing appropriate services to users according to their current situations.

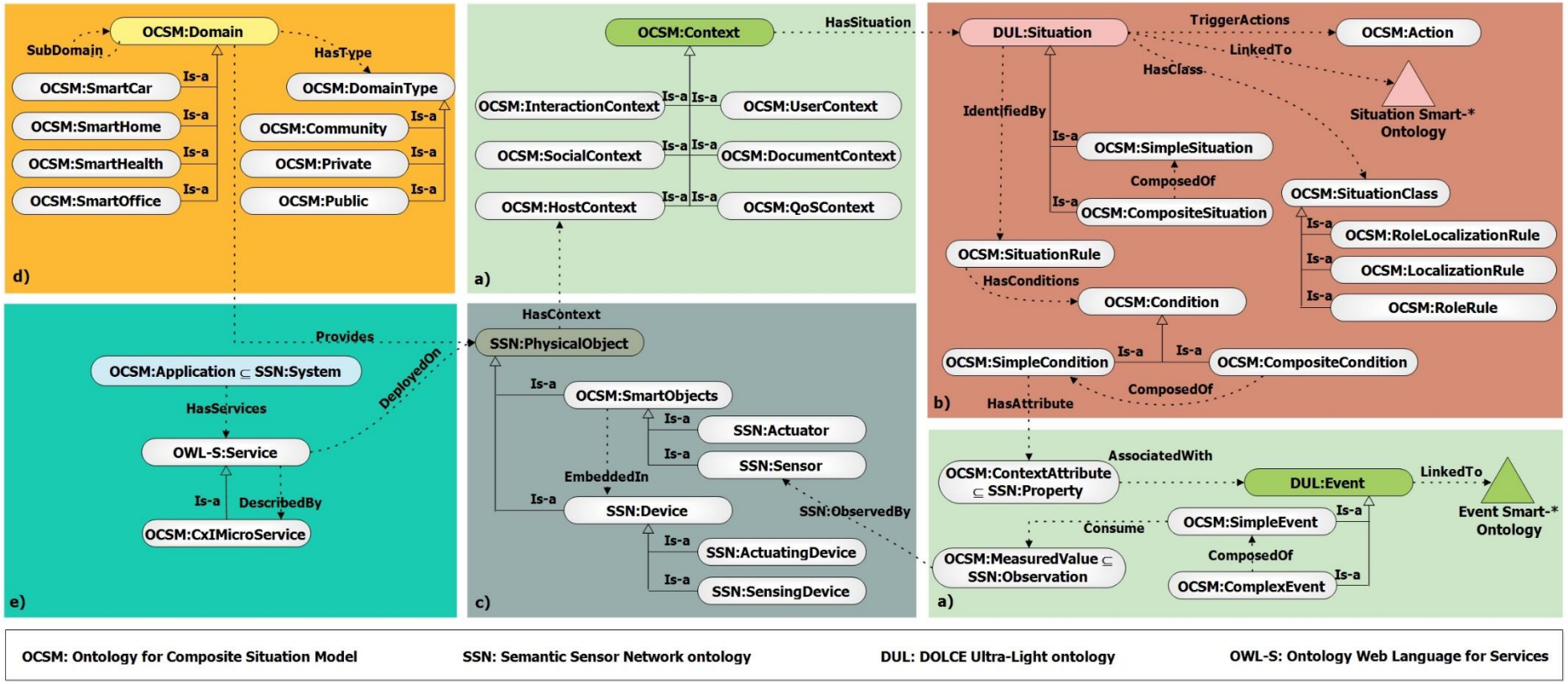


Figure 4.2: An overview of generic context-aware composite situation ontology.

4.2.1 Smart objects class

The first-class consists of several heterogeneous sensors and actuators. Each one has a unique identifier, a local name, a location and other attributes describing its properties (figure 4.3). It contains several sensors depending on its type (*logic or physic*) and its features (*size, frequency, name, URL*). These sensors acquire raw contextual data, which will be processed using the reasoning techniques based on common functionalities. We have two main types of sensors: logic and physical sensors. Physical sensors can include bio-sensor (e.g., data captured by bio-sensors, like blood pressure, blood sugar, body temperature, etc.), weather sensor, environment sensor (e.g., data captured by environmental sensors, like room temperature, humidity, etc.). Logic sensors include device sensors (e.g., data captured by sensors, like CPU speed, battery energy, etc.), network sensors, calendar sensors, email sensors, etc.

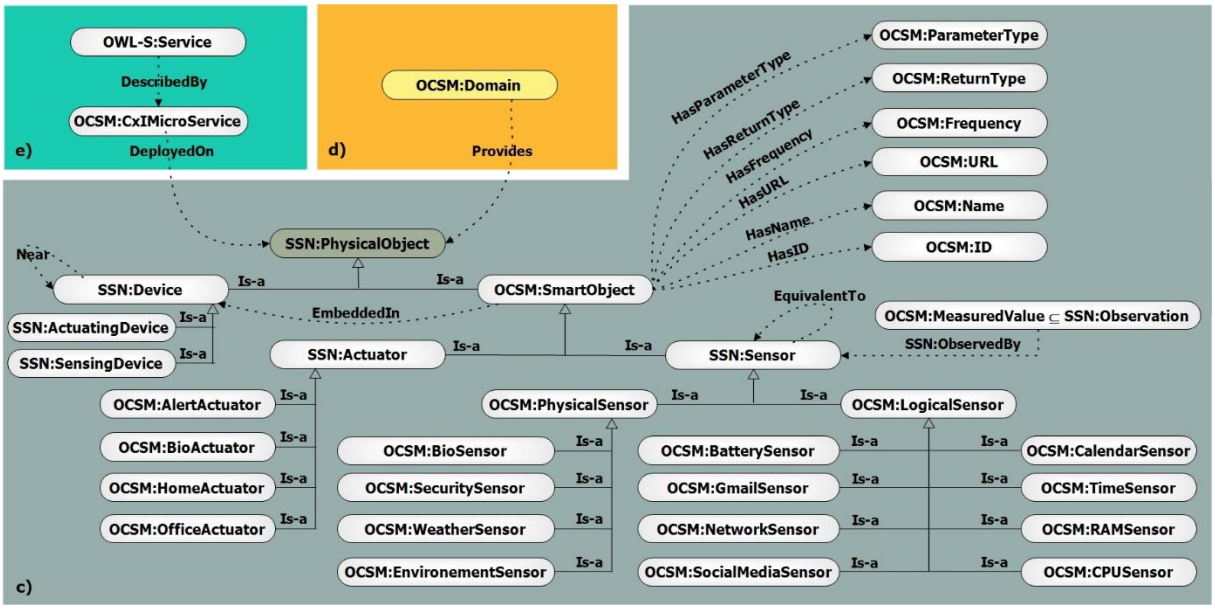


Figure 4.3: Context Smart Object Multi-OCSM sub-ontology.

4.2.2 User's domain class

A user's **Domain** (e.g., home domain or work domain) is composed of several devices. Each device is responsible for the execution of several services. Each service contains one or more Context-aware Intelligent Micro-Services (*CxIMicroService*). Each CxIMicroService is deployed on a specific smart device. It provides a specific need and has hardware and software requirements (*using CxIMicroService description profile*) in order to run in an optimal mode.

4.2.3 Context class

The third class **Context** plays a significant role to enable the provision of adequate services to users based on their surrounding environments (see figure 4.4). It aims to classify the contextual information using multi-dimensional aspects:

- The ***User Context*** can include a user's role, user's current location, time, and user's agenda that describes all tasks performed in daily life (e.g., watches TV or listens to music).
- The ***Environment Context*** describes both location and time, which are very relevant context information for situation identification. Location: is described either by geographical coordinates, text-description or by using GeoNames ontology [111]. Time: the time can support multiple values such exact time (h: 20, min: 30) or timestamp (start time, end time, current time) or a semantic keyword such `morning' or by using time ontology.

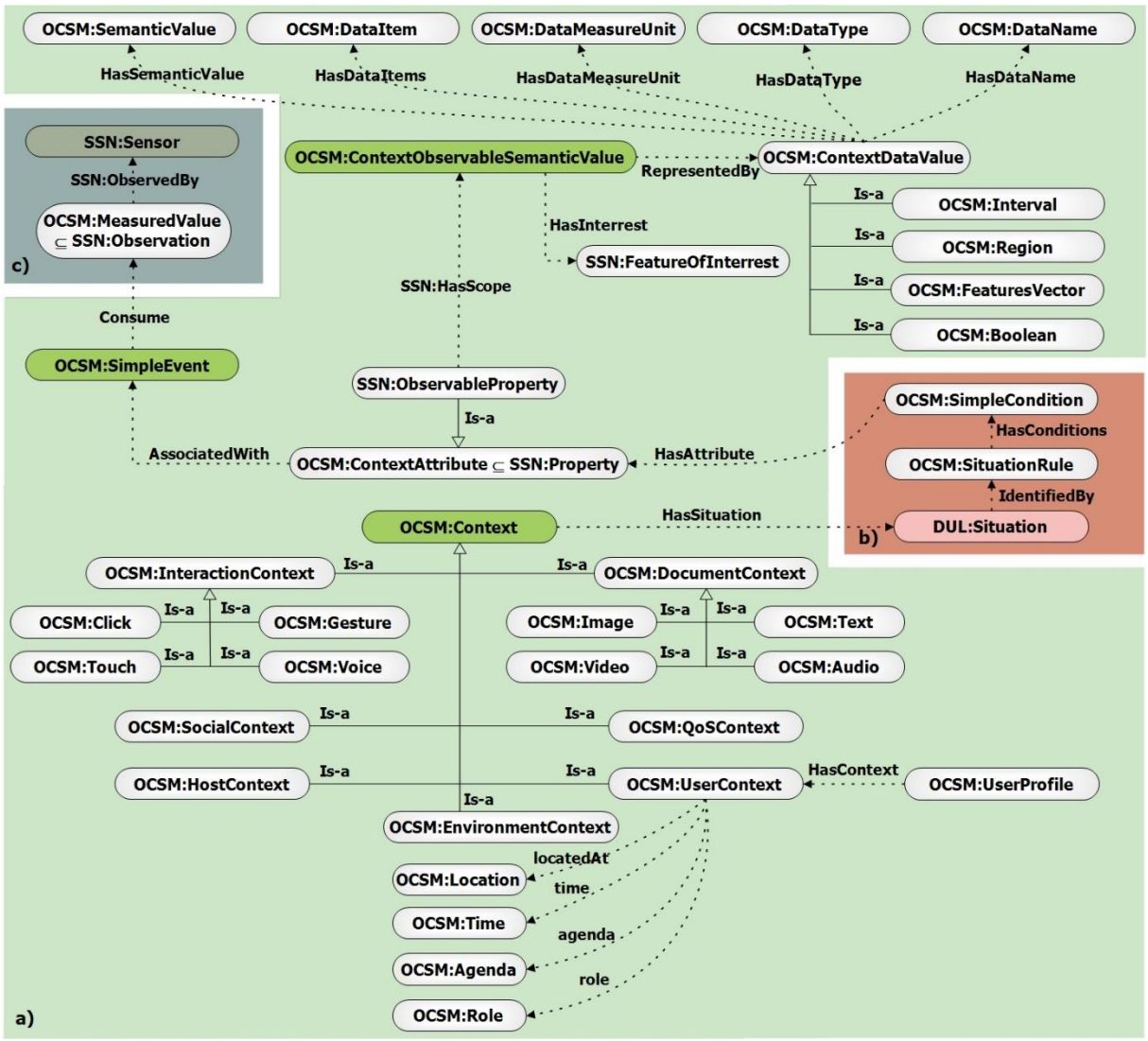


Figure 4.4: Context Multi-OCSM sub-ontology.

- The ***Host Context*** represents various physical object (smart objects, devices) and its related properties. For example, a sensing device is characterized by the *memory size, CPU speed and battery level*.
- The ***Interaction Context*** describes the different interaction modalities (click, gesture, voice, touch). The interaction context specifies a set of properties related to a specific modality type.

- The ***Document Context*** specifies the type of documents (text, image, video, audio) and its properties related to a specific media type.
- The ***QoS Context*** describes the quality of service and smart mobile application, which is defined as a set of QoS parameters. These QoS parameters are (1) continuity of service; (2) speed and efficiency of service; and (3) safety and security.
- The ***Social Context*** describes the context information generated from different social networks and high-level relations among multiple users.

Each context category has a set of different context attributes (SSN:Property). Each context attribute is associated with an event (i.e., the sensed data value by a specific sensor) as presented in figure 4.4. We extend the work of Dromzée & al. [112] by employing two types of attributes in interpreting events: *region*, a *vector of features* (*image, video, and audio*). It helps in interpreting events regardless of their sources using semantic value. Indeed, a semantic value, like "Low" may be applied on several context-attributes, such as the outside temperature level, the glucose level, the bandwidth. Hence, the meaning of a semantic can be completely different depending on the context used. Moreover, even in a specific context, for several smart domains (smart home, smart office), a semantic value may have different interpretations. For instance, the semantic value "Presence" which belongs to a user corresponds to a vector of GPS coordinates when we consider smartwatch as the data source, or it corresponds to different feature values when we consider surveillance camera as the data source or corresponds to Boolean value when we consider motion sensor as the data source. Actually, this proposal may be used for a wide variety of semantic values (see figure 4.5). For instance, one may define semantic values for spatial information, such as \closeTo", \farFrom", \intersectWith", etc.

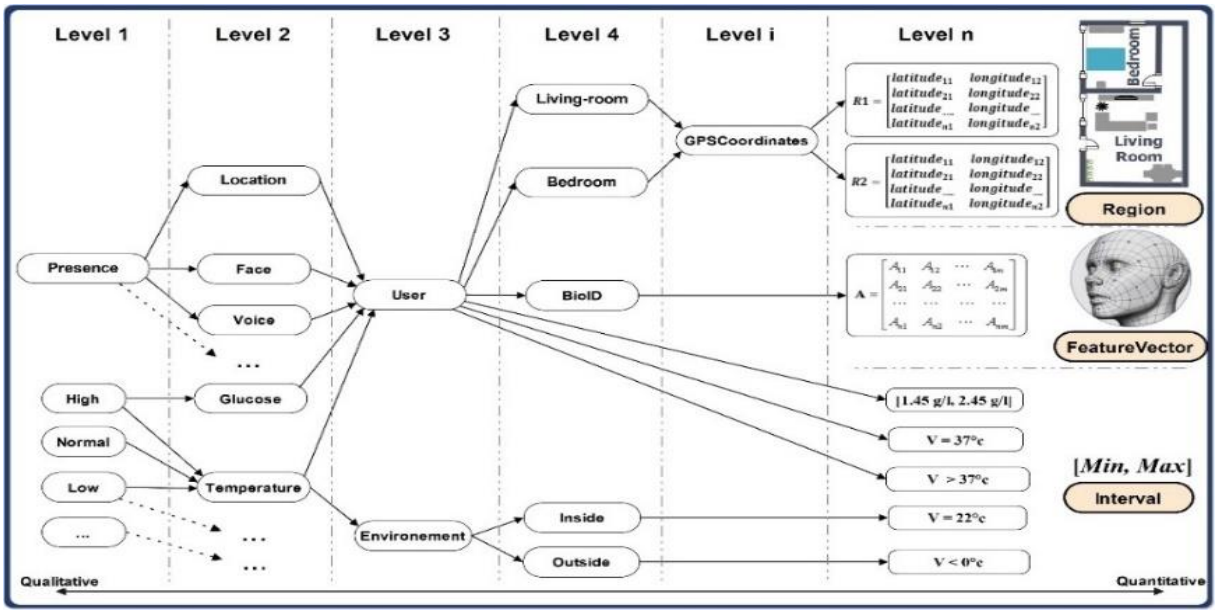


Figure 4.5: Semantic interpretation of events extended form [112].

4.2.4 Situation class

Situation is the basic concept of any context-aware application. It is the set of events that are occurring at a given location (where) and time (when), and the conditions (rules) that can verify those events, thus the need to identify the situation. Figure 4.6 presents a situation sub-ontology, delineating the user's context, rule modeling, data semantic value, situation. Each user context has distinct sets of situations. A situation is identified by a situation rule. A rule is represented by a combination of multiple conditions. Each condition has a context attribute (e.g., location, time, role or glucose level.).

The context attribute can be attached to a semantic value (e.g., home for the context attribute location, morning for the context attribute time, student for the context attribute role and high for the context attribute glucose level) using a comparator (e.g., *inside, outside, near* and *far* for Location; *before, after, while* and *equal* for Time; *is* for Role; etc.) to enrich the description of the condition. Besides the process of situation identification, we need to know what the user expects when this situation happens (i.e., situation rule is verified). So, we need to determine which actions to activate when the situation is identified through a semantic link between a situation and actions.

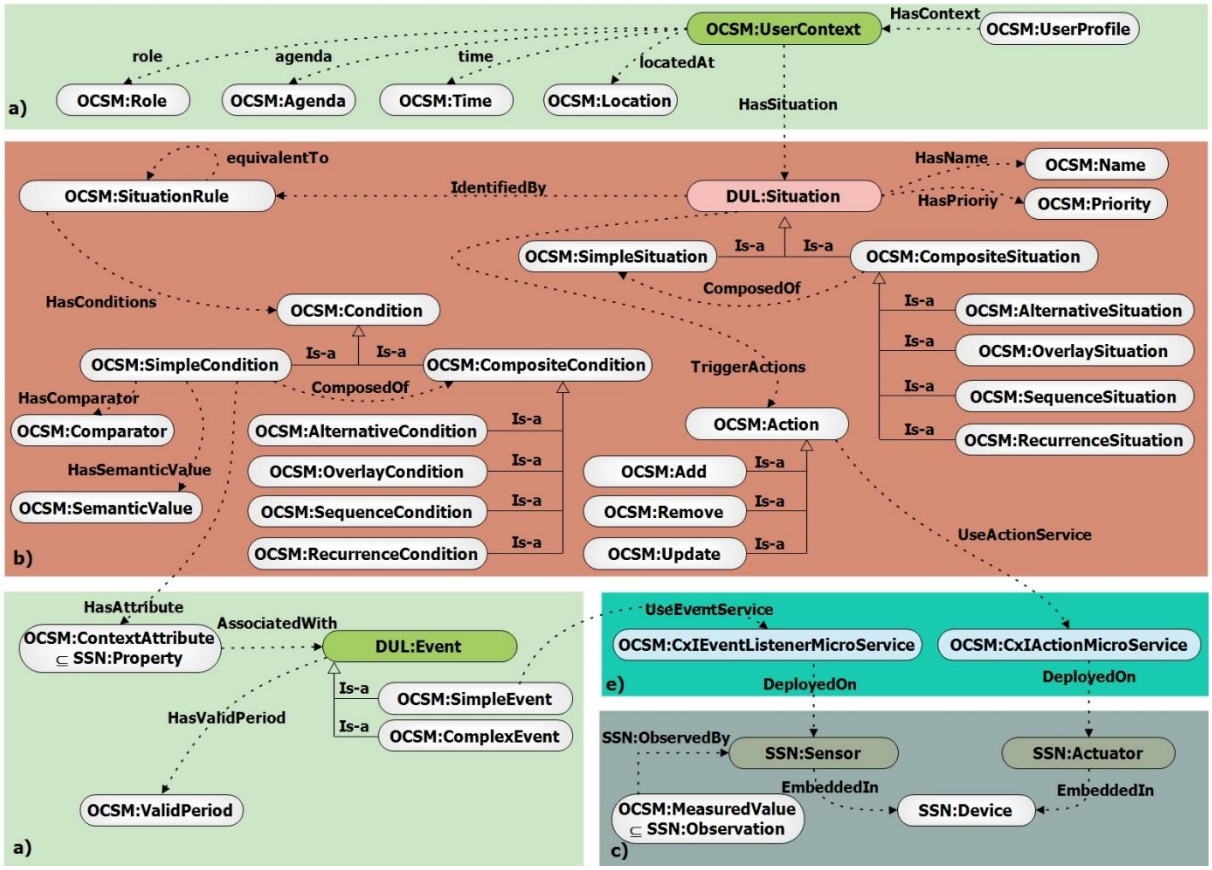


Figure 4.6: Situation Multi-OCSM sub-ontology.

4.2.5 Modeling prioritized situations

Once there are urgent situations with lives at stake such as heart-situations, time is a crucial factor for handling such situations in specific period. The application searches as quickly as possible for the closest available services to prevent damage and serious health evolution. At first, a priority value is set by an expert in the offline (e.g., a doctor specifies high priority value in case of a serious health situation). Then, the situation priority is updated online using a new weight based on the situation's category, user's activity, user's location, and situation's coverage.

- **Situation Category (P_1).** Situation category defines the important level of the situation depending on the emergency evaluation of the triggered actions. The priority value of a situation '*s*' is used to identify whether a given situation is normal, important or urgent. Therefore, the priority value is attributed based on the evaluation of situation rule that an expert can evaluate its importance level. Each situation is assigned a priority value from 1 to 3, where the lowest value represents the highest priority and vice versa. In our work, we used three different semantic levels:
  - **Normal situations:** situations with the least priority (i.e., priority value =3) representing the user's daily life situations.
  - **Important situations:** medium priority situations (i.e., priority value =2) representing the user's situations related to work, travel, etc.
  - **Urgent situations:** highest priority situations (i.e., priority value =1) representing the user's urgent situations (e.g. healthcare situations).
- **User Activity (P_2).** User activity is another factor that determines the priority level of a situation. In our proposed work, we defined three levels of activity priority depending on the user's state:
  - **Free:** this state has the lowest priority that allows daily life activities (e.g., priority value =3).
  - **Active:** this state has a medium priority that allows important activities (priority value =2).
  - **Busy:** this state has the highest priority for urgent activities (e.g., priority value =1).
- **User Location (P_3).** A user's location is another factor that determines the priority of a situation. The user with the closest distance to his situation's location should be given a higher priority to be filtered and identified because the probability that he leaves the situation's area is higher. Here, we used the normalized Euclidean distance between the situations' locations and the current user's location to compute the priority value. The highest priority value is assigned to the smallest distance and vice versa. The normalized Euclidean distance *Ds* of the situation *s* with the current user's location is:

$$P_3 = D_S = \frac{\sqrt{(x_u - x_s^c)^2 + (y_u - y_s^c)^2}}{\sqrt{(x_s^{max} - x_s^c)^2 + (y_s^{max} - y_s^c)^2}} \quad (1)$$

Where:

x_u, y_u: current location coordinates of user *u*,

x_s^c, y_s^c: position center coordinates of situation *s*,

x_s^{max}, y_s^{max}: maximum coordinates coverage of situation *s*.

- **Situation Coverage (P_4).** Situation coverage is another factor that determines the priority of a situation. The coverage of the situation can be found in different sizes (individual situations, community situations, and public situations). Furthermore, the population size of a situation changes with time. The situation with the highest population should be given a higher priority to be identified. Here, we used the average of population coverage over a period of time, instead of the instant value, to calculate its priority as follows:

$$P_4 = \frac{\sum_{t=0}^{N} C_s^t}{N} \quad (2)$$

Where C_s^t denotes the population coverage for a situation *s* in an instant *t*, and *N* is the number of time instances.

After the calculation of the four priorities (P_1, P_2, P_3, P_4), the weight value P_S of a situation *s* is calculated as follows:

$$P_S = w_1 * P_1 + w_2 * P_2 + w_3 * P_3 + w_1 * P_4 \quad (3)$$

Where w_i are the weighting factors that allow an expert to specify the importance of corresponding metrics, with $\sum_{i=1}^{4} w_i = 1$.

4.2.6 Service class

The distributed smart mobile application consists of a set of distributed services. A service is an autonomous Context-aware Intelligent microservices that make up an application and empower reasoning capabilities over context data (CxIMicroService see figure 4.7). The benefit of decomposing the application service into different small services is that it improves modularity and flexibility to provide support in managing situations under the mobility of users. Another benefit is that each CxIMicroService runs in its process, which collaborates and communicates with other microservices. Likewise, each CxIMicroService has an autonomic functionality (i.e., task implements the runnable interface in Java code) that serves as the body of a thread. The latter will be deployed using the Service Executor of our framework. We extend the CxIMicroService with a CxISituationIdentifierMicroService, CxIEventListenerMicroservice, and CxIActionMicroservice to define situation rule identifier, situation's event listener and situation's actions respectively. These parallel microservices are capable of making some useful reasoning tasks on its context smart domain (location, context entity, etc.) with the help of an

ontology-based model. Using our service Multi-OCSM sub-ontology, we can construct a hierarchy of collaborated CxIMicroservices using other specialized CxIMicroservice classes in the situation reasoning process. In this case, a supporting framework has the capability of sharing parallel-detected events and identified situations in different smart environments through distributed mobile applications.

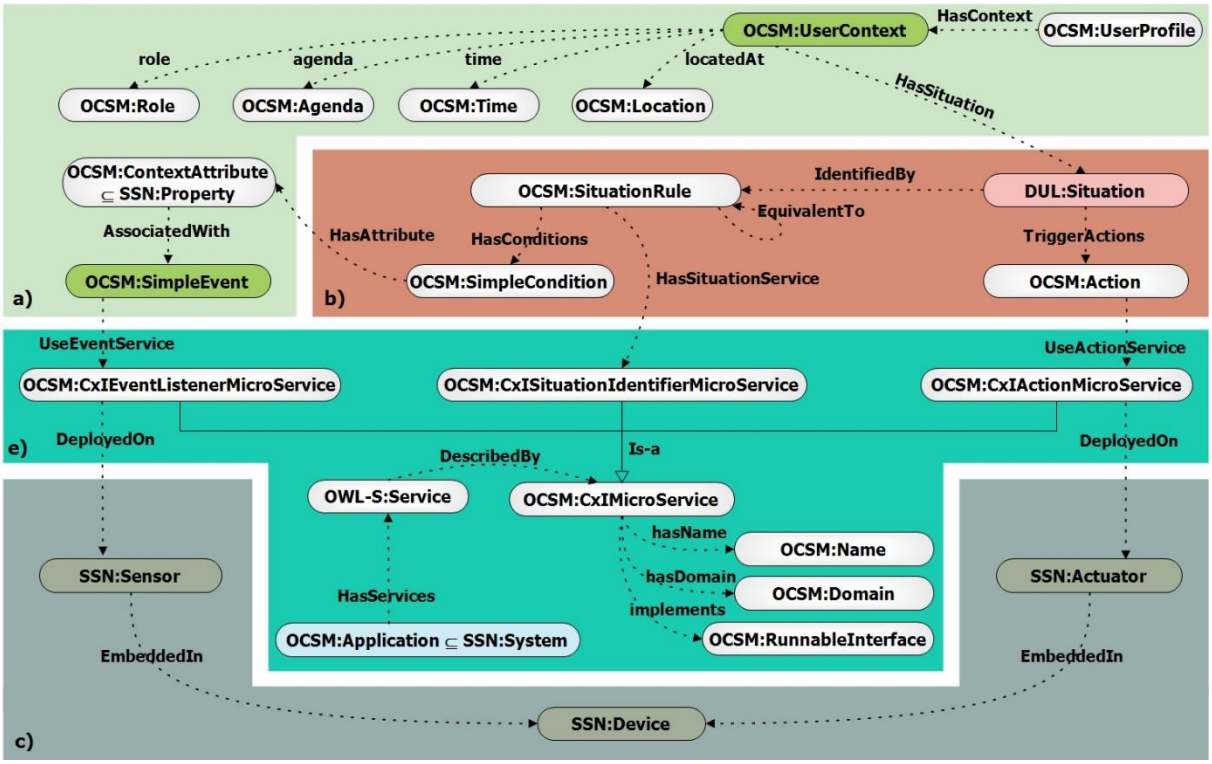
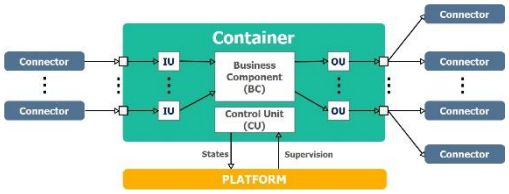
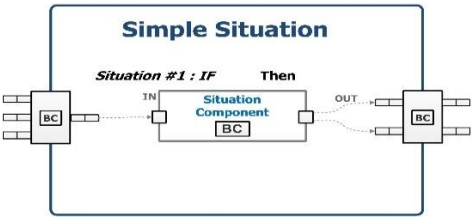
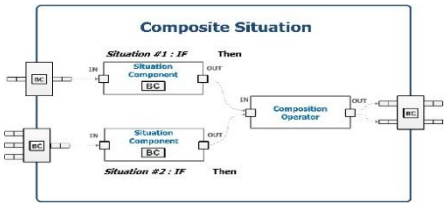


Figure 4.7: Service Multi-OCSM sub-ontology.

4.2.7 Modeling service using component-based approach

As mentioned earlier in section 4.2.6, mobile applications are built using several available microservices. In fact, a large number of microservices may affect negatively the application's flexibility and maintainability. Thus, a level of abstraction is needed in order to resolve this challenge. We exploit the architectural entities (i.e., components) of the Kali-smart middleware [9] to build microservice-based composite situations. We distinguish four types of composition component operators: sequence, parallel, negation and recurrence (as shown in table 4.1). We integrate these operators into Kali-smart middleware for managing composite situations at design and runtime levels. For instance, figure 4.8 introduce a design model of a composite situation called "Low Battery & Meeting composite situation" built with the combination of two simple situations "Low Battery simple situation, and Meeting simple situation" using parallel composition component operator. The identification of this composite situation triggers a set of action services including: exclude audio output and migrate the meeting application on another available device according to user's preferences.

Table 4.1: Distributed Mobile Application Extend Concepts on Kali-Smart.

Concept	Definitions	Graphic notations
Services and Components	The components are entities composing the service run simultaneously in a distributed environment composed of the users' devices. The components can be divided into three categories: **Input**, **Core**, and **Output**.	
Situations	**Situations** could be atomic or composite, represent a set of events that are happening at a given location and time and the conditions. Situations are mapped to services. Services represent the services performed by the application within each situation.	
Composite situations	A **composite situation** is an abstraction of multiple situations combined from multiple atomic situations.	
Sequence	**Sequence operator** allows the composition of two dependent situations in a sequence order	
Parallel	**Parallel operator** allows the composition of two independent situations in parallel order	
Negation	**Negation operator** specifies that a certain situation must not detect in any location in a given period.	
Recurrence	**Recurrence operator** specifies that a certain situation must be detected many times	

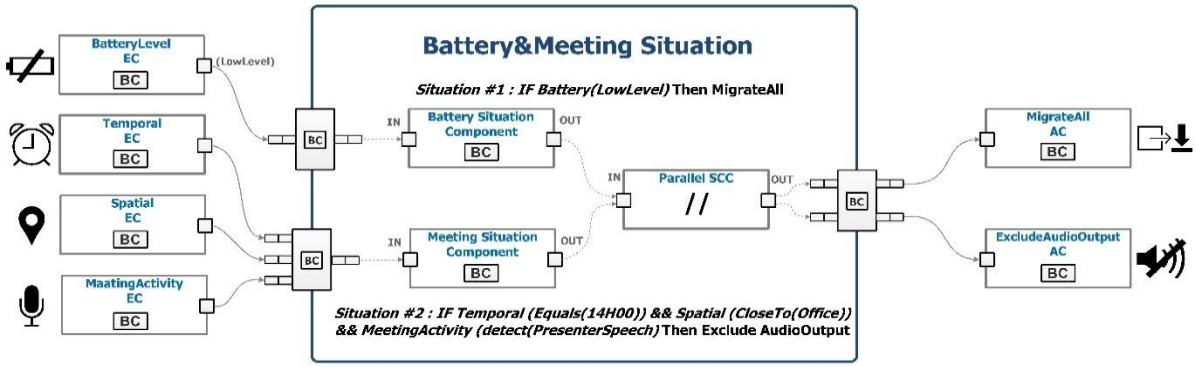


Figure 4.8: Example of Composite Situation Based Model.

4.3 Implementation and validation of Multi-OCSM

4.3.1 Implementation of Multi-OCSM

After the conceptualization of the ontology Multi-OCSM, we can start the implementation process. The implementation process consists of two steps. First, we implement the ontology Multi-OCSM (*classes, subclasses, and their object properties*) and instantiates some individuals from each class, while the second is to fill it by defining some formal logics as well as interrogation rules. Both steps were generated using the Protégé 7.1 editor [113]. This visual modeling tool provides OWL (i.e., Ontology Web Language) for describing our Multi-OCSM ontology that has been used to reason about its defined classes and individuals. The Multi-OCSM ontology is made-up of four upper-level classes: Smart Objects and Devices, Context, Situation, and Service & Application. First, we create Multi-OCSM hierarchical classes and its subclasses (Figure 4.9.a). For instance, the class *Context* has seven sub-classes: *ContextActivity, ContextDocument, ContextEnvironment, ContextHost, ContextInteraction, ContextQoS* and *ContextUser*. Then we add for each class its datatype properties (Figure 4.9.b) and objects properties (Figure 4.9.c) which defines the semantic relationships between classes. For instance, the *Situation* class has an object property "*HasCondition*" which is linked to *Condition* class. This relationship means that the *Situation* class can have one or more *conditions.*

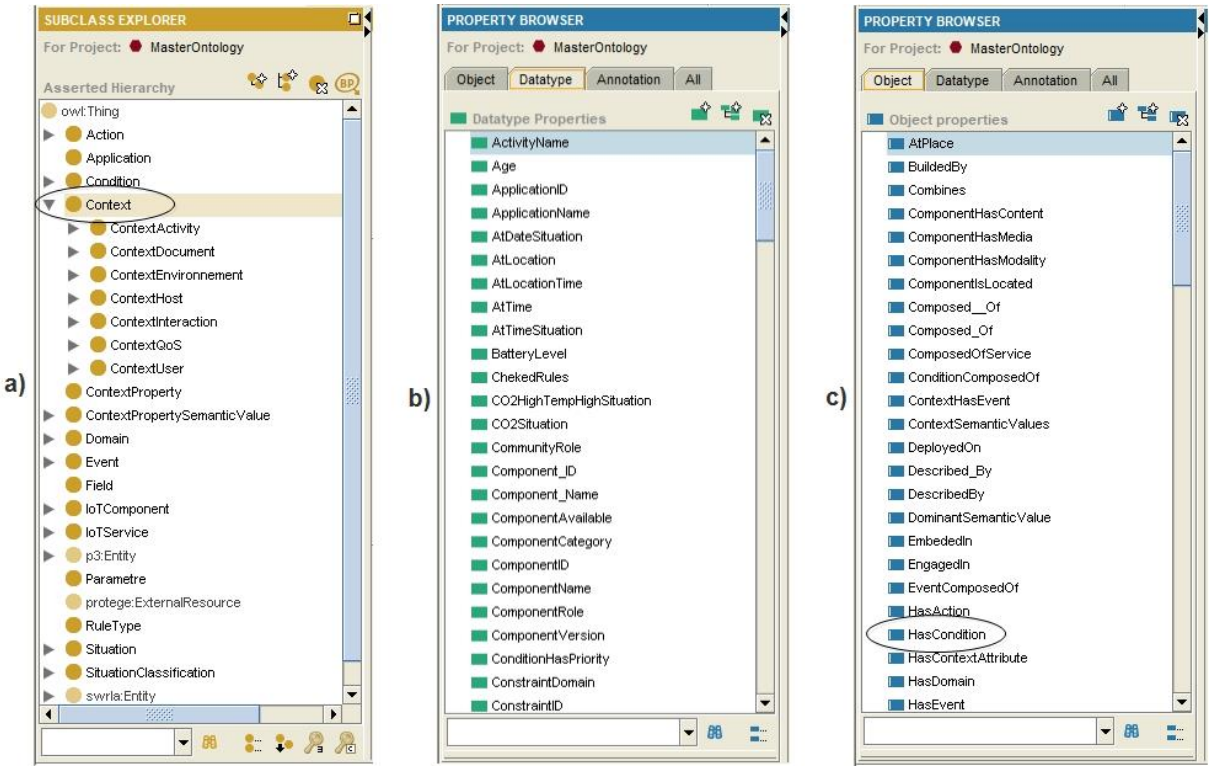


Figure 4.9: Implementation of Multi-OCSM ontology in Protégé.

Figure 4.10 illustrates a part of Multi-OCSM. It shows four fundamental classes: *Situation*, *Condition*, *Event* and *Action*. In addition, it shows the semantic links between these classes (e.g., Situation ***has*** Condition, SimpleCondition ***Is-a*** Condition, ComplexEvent ***ComposedOf*** SimpleEvent, etc.).

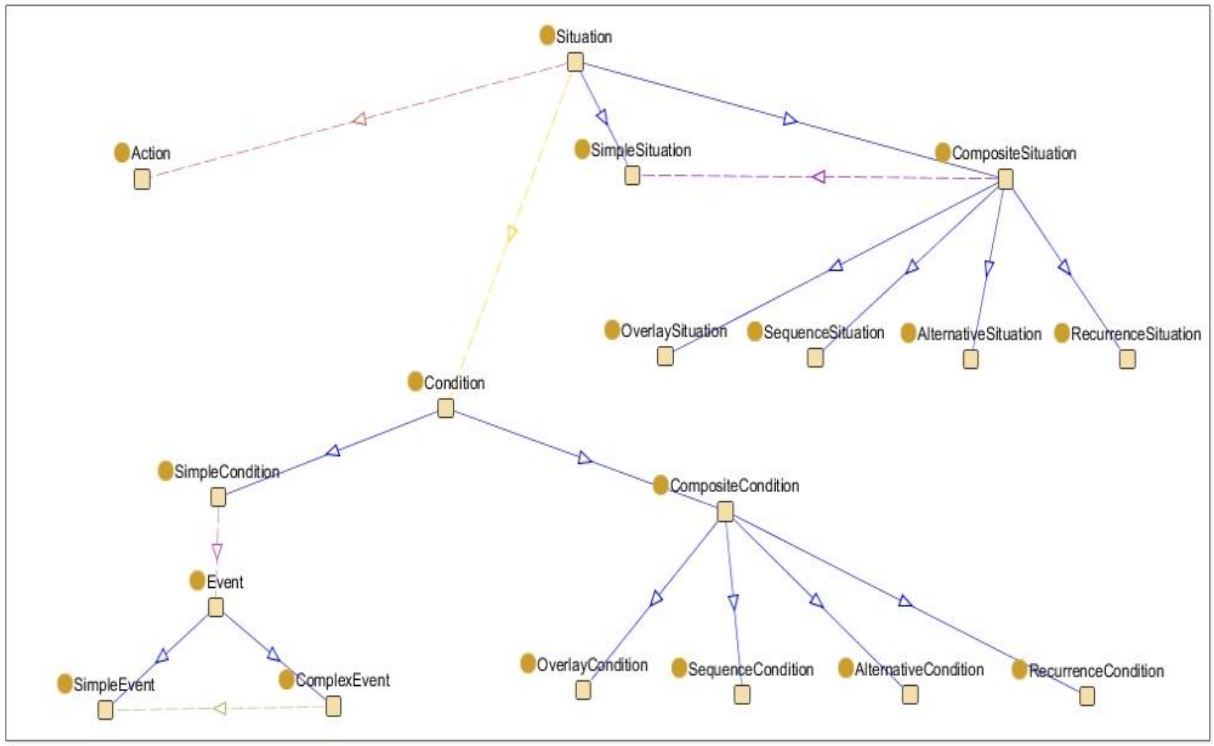


Figure 4.10: An illustrated example of a part of multi-OCSM generated with Jambalaya.

4.3.2 Consistent rules of Multi-OCSM

Verifying the consistency of Multi-OCSM instances is an essential part of our implementation, especially the consistency structure of composite situations and user-defined constraints (with various context properties) while providing more formal semantics. To deal with it, we use the Description Logic (DL) [114] to represent Multi-OCSM in a formal and structured way. The Multi-OCSM expressed in a DL is constituted by two components, traditionally called *TBox* and *ABox* [115]:

- **TBox (assertions on concepts):** describes terminological information by defining basic or derived concepts. Also, it defines how these concepts relate to each other.

  The TBox is a finite set of inclusion assertions of the form: $C_1 \sqsubseteq C_2$, where C_1 and C_2 are concepts. Note that: $C \equiv E$ as an abbreviation of $C \sqsubseteq E$ and $E \sqsubseteq C$.

- **ABox (assertions on individuals):** describes the information that characterizes individuals. This information is expressed by specific or local assertions.

The ABox is a finite set of instance assertions of the form: $C(\alpha)$, where C is a concept, and α is an individual name, or of the form: $P(\alpha_1, \alpha_2)$where P is a primitive role and α_1, α_2 two individuals names. Two fundamental aspects then characterize a DL system: 1) - the language used for building the concepts and the roles mentioned in TBox and ABox, 2) - the reasoning mechanisms on the knowledge bases expressible in the system. The first aspect was ensured via the protégé editor [113], where concepts,

relationships between concepts, concepts properties and creation of individual were created using the visual interface of Protégé. Besides that, the reasoning mechanism is fundamentally based on three parts: 1) – the source of knowledge Multi-OCSM, 2) – the interaction between the ontology and the system through the SPARQL query language and 3) – the reasoning engine based on parallel semantic-based composite situation identification framework. Table 4.2 presents DL example of our ontology.

Table 4.2: DL example of our ontology.

TBox	ABox
$Context \sqsubseteq Thing$	$UserContext(User\_A)$
$UserContext \sqsubseteq Context$	$HasID(User\_A, U\_1)$
$\geq HasID \sqsubseteq String$	$HasName(User\_A, Adem)$
$\geq HasName \sqsubseteq String$	$HasAge(User\_A, 25)$
$\geq HasAge \sqsubseteq Int$	$HasSituation(User\_A, S1)$
$\geq HasSituation \sqsubseteq Situation$	$HasSituation(User\_A, S2)$
$Situation \sqsubseteq Thing$	$Situation(Situation\_S1)$
$\geq HasID \sqsubseteq String$	$HasID(Situation\_S1, S\_1)$
$\geq HasName \sqsubseteq String$	$HasName(Situation\_S1, High\_Glucose)$
$\geq HasCondition \sqsubseteq Condition$	$HasCondition(Situation\_S1, Condition\_High\_Glucose)$
$\geq HasPriority \sqsubseteq double$	$HasPriority(Situation\_S1, 0.81)$
$\geq HasService \sqsubseteq Condition$	$HasService(Situation\_S1, Service\_Inject\_Insulin)$

4.3.3 SPARQL queries for Multi-OCSM

SPARQL (SPARQL Protocol and RDF Query Language) [116] is used to express queries across diverse data sources that can be used for semantic data extraction. The use of SPARQL is based on two functional properties, simplicity and rapidity. We utilize SPARQL queries to extract semantic data from our multi-OCSM ontology. Thus, we develop many interrogation SPARQL queries based on Multi-OCSM. The primary goal of these queries is extracting the semantic features related to monitored user context, then filtering relevant situations of the user in order to accelerate the situation identification process. Table 4.3 shows some selection and filtering queries.

Table 4.3: SPARQL selection and filtering queries.

Query 1 selects the user's location event. It takes as parameter "user name".

```
PREFIX OCSMOnto: <http://www.owl-ontologies.com/Ontology1430403694.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?idEvent
WHERE {
 ?user OCSMOnto:UserName\""+user.getUserName()+"\".
 ?user OCSMOnto:HasLocationEvent ?locationEvent .
 ?locationEvent OCSMOnto:IdEvent ?idEvent .
}
```

Query 2 selects all the situations that do not have any spatiotemporal constraint. It takes as parameter "user name".

```
PREFIX OCSMOnto: <http://www.owl-ontologies.com/Ontology1430403694.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?situation ?idSituation  ?locationTime
WHERE {
 ?user OCSMOnto:UserName\""+user.getUserName()+"\".
 ?user OCSMOnto:UserHasSituation ?situation.
 ?situation OCSMOnto:IdSituation ?idSituation .
 ?situation OCSMOnto:AtLocationTime ?locationTime .
 FILTER( ?locationTime = \"AnyWhere#AnyTime\").
}
```

Query 3 returns all situations that have a spatiotemporal constraint. It takes as parameter "user name, user location and time"

```
PREFIX OCSMOnto: <http://www.owl-ontologies.com/Ontology1430403694.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?situation ?idSituation  ?locationTime
WHERE {
 ?user OCSMOnto:UserName\""+user.getUserName()+"\".
 ?user OCSMOnto:UserHasSituation ?situation.
 ?situation OCSMOnto:IdSituation ?idSituation .
 ?situation OCSMOnto:AtLocationTime ?locationTime .
 FILTER(?locationTime = \""+location+"#AnyTime\"||
 ?locationTime = \""+location+"#"+time+"\"||
 ?locationTime = \"AnyWhere#"+time+"\").
}";
```

Query 4 returns all the sub situations of a given composite situation. It takes as parameter "id of composite situation"

```
PREFIX OCSMOnto: <http://www.owl-ontologies.com/Ontology1430403694.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?subSituation ?idSubSituation              WHERE {
 ?situation OCSMOnto:IdSituation
            \""+situation.getIdSituation()+"\".
 ?situation OCSMOnto:HasSubSituation ?subSituation.
 ?subSituation OCSMOnto:IdSituation ?idSubSituation
}
```

4.3.4 Evaluation of Multi-OCSM

Figure 4.11 shows an instance of individual profile and its properties. As we can see, the individual *MohammedProfile* of the class IndividualProfile has object properties (e.g., isLocatedAt, HasScheduledActivity, HasSituation, etc.) and datatype properties (e.g., UserName, LastName, Age, Profession, etc.).

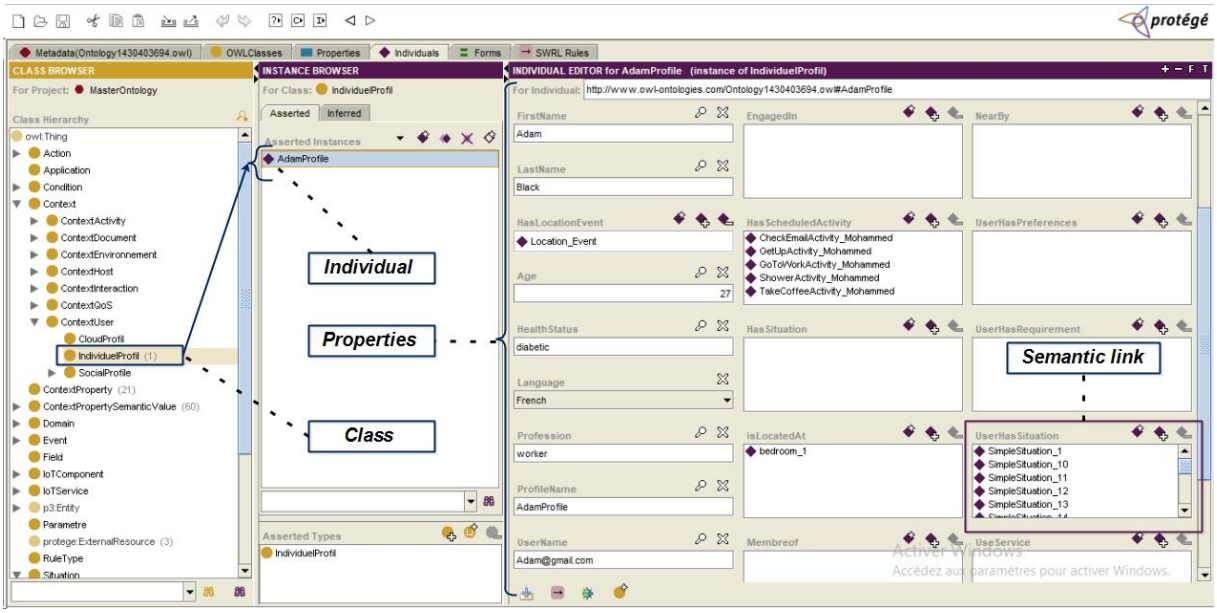


Figure 4.11: Description and properties of Mohammed profile in Protégé.

After the implementation of the multi-OCSM ontology, we would like to show a concrete example represented by OWL. Figure 4.12 shows an example of a concrete model based on the Multi-OCSM ontology represented by OWL. This example describes a garage within a smart home. The garage is a subdomain of the domain smart home (line 8). The datatype property "PlaceID" (line 4) represent a unique identifier of the garage. The garage has a domain where we can define provided smart objects (i.e. sensors and actuators) (lines 9-20). As we can see in figure 4.12, the garage has a camera sensor (lines 10-13) and a garage door opener actuator (lines 16-29). The camera sensor has an ID (line 11) and uses the service "Service_GetCamera" (line 12). In addition, the garage door opener actuator has ID (line 17) and uses the service "Service_Garage_Door_Opener" (line 18).

We have evaluated the execution time of several SPARQL queries on some ontology individuals. The goal is to demonstrate how Multi-OCSM ontology can perform SPARQL queries based on the combination of multiple operators among queries. Note that all evaluations have been performed on a PC running Windows 10 (64 bits) on an Intel processor core i5-2430 2.4 GHz, and 8 GB RAM. We have compared the execution time of different SPARQL queries (query 1, query 2, query 3, and query 4 of table 4.3) on Multi-OCSM with 100, 200, 300… 1000 times. First, we run query 1 and query 2 separately multiple times. Then we combine query 1, query 2, query3 and query 4 using sequential operator and run

them multiple times. Figure 4.13 shows the execution time of requesting Multi-OCSM with different SPARQL queries. The execution time of a single query repeated 500 times shows an average of 1000ms. Nevertheless, combining different SPARQL queries shows an average increase in the execution time of 2000ms. Our aim is to optimize the execution time. To achieve our goal, we propose a parallel microservices-based approach while combining multiple queries (for more details see Chapter 5).

```
<Garage rdf:ID="Garage_1">
    <SuperID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Home_1</SuperID>
    <PlaceID rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Garage_1</PlaceID>
    <HasDomain>
        <Community rdf:ID="Garage">
            <SubDomain rdf:resource="#SmartHome"/>
            <Provides>
                <CameraIPSensor rdf:ID="CameraIPSensor_garage">
                    <IdSensor rdf:datatype="http://www.w3.org/2001/XMLSchema#string">42</IdSensor>
                    <UseService rdf:resource="#Service_GetCamera"/>
                </CameraIPSensor>
            </Provides>
            <Provides>
                    <AutomationActuator rdf:ID="AutomationActuator_Garage_Door_Opener">
                        <IdActuator rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A_2</IdActuator>
                        <UseService rdf:resource="#Service_Garage_Door_Opener"/>
                    </AutomationActuator>
            </Provides>
        </Community>
    </HasDomain>
    <In rdf:resource="#Smart_Home_1"/>
</Garage>
```

Figure 4.12: A concrete example of OWL representation for multi-OCSM.

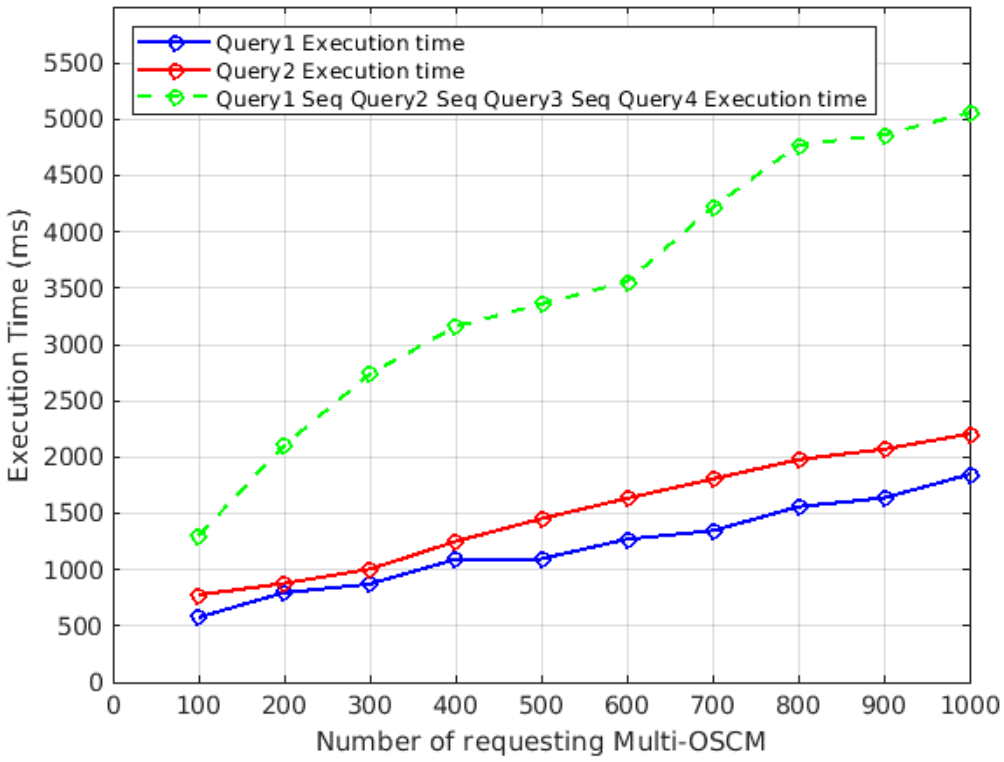


Figure 4.13: Multi-OCSM Query execution time comparison.

4.4 Conclusion

In this chapter, we have proposed a novel generic user-centric composite situation ontology called Multi-OCSM (multi layered ontology for composite situation model) for representing the majority of concepts of smart-* domains and user's profile. This ontology contains five key classes: Context, Situation, Smart objects, Domain, and Service. It describes both simple and composite situation rules and its semantic description including composition operators (parallel, sequence, recurrence and alternative). This ontology facilitates the management of a large number of heterogeneous smart objects and services, semantic features related to the context data for the interpretation of parallel events, interoperability of different incoming events and highlights the semantic rules of user. Moreover, the Multi-OCSM supports SPARQL queries, which allow users to explore the ontology and to customize the situations filtering according to their constraints and preferences.

In the next chapter, we will present a novel parallel semantic-based composite situation identification framework for optimizing the situation identification process. Also, we present the whole process of situation identification illustrated with a uses case and possible scenarios. To cope with that, we have based on the multi-OCSM ontology that plays a fundamental role in our approach.

Chapter 5

Real-Time Ontology-based Context-aware Situations Identification in Pervasive Computing

Summary

5.1 Introduction

Nowadays, users are looking for flexible and effective solutions that are suitable to automate their daily activities. The objective is to have a reconfigurable distributed mobile application adapted to the user's current needs according to the environment changes. Reconfigurations are needed due to the users' activities as well as their possible evolution scenarios. Besides, the major challenge we face today is the management of urgent situations that must be identified rapidly and processed in a short time. To address this issue, a parallel identification process of context-aware semantic-based composite situation is an effective and efficient solution for automatic and flexible reconfigurable mobile application.

In the previous chapter, we presented the Multi-OCSM ontology, which is considered as the backbone of our approach. This ontology is used to describe the semantic aspects of various concepts of distributed mobile application with its changing context. Furthermore, it unifies the description of heterogeneous smart devices and various composite situations of users' daily life to meet their needs in a specific context.

In this chapter, we present a novel parallel identification process of semantic-based composite situations. The main purpose of this process is to submit tasks as much as the number of user's situations running simultaneously across multiple processor cores. In addition, it provides users with a customized context-aware distributed mobile application based on our ontology. Our approach relies on the parallel identification of composite/simple situations and the automatic generation of reconfiguration actions

based on the Multi-OCSM. Moreover, we add the notion of priority to the situation rules to perform the situation's identification process and speeding up the queued urgent situation rules in order to process urgent situations with high priority before the normal situations having low priority.

The rest of this chapter is organized as follows. We detail the identification process of context-aware composite situation. After that, we define a priority strategy for speeding up pending urgent situations. Then we illustrate our process with a use of case and possible scenarios. Finally, we close this chapter with a conclusion.

5.2 Real-time dynamic semantic-based situations identification processes

This section presents an autonomic dynamic semantic-based composite situation reasoning and identification process over heterogeneous smart objects. It enables the filtering and factorization of the user's situations rules. In addition, it allows the parallel identification of composite/simple situations and provides an automatic generation of reconfiguration actions based on the Multi-OCSM as illustrated in Figure 5.1.

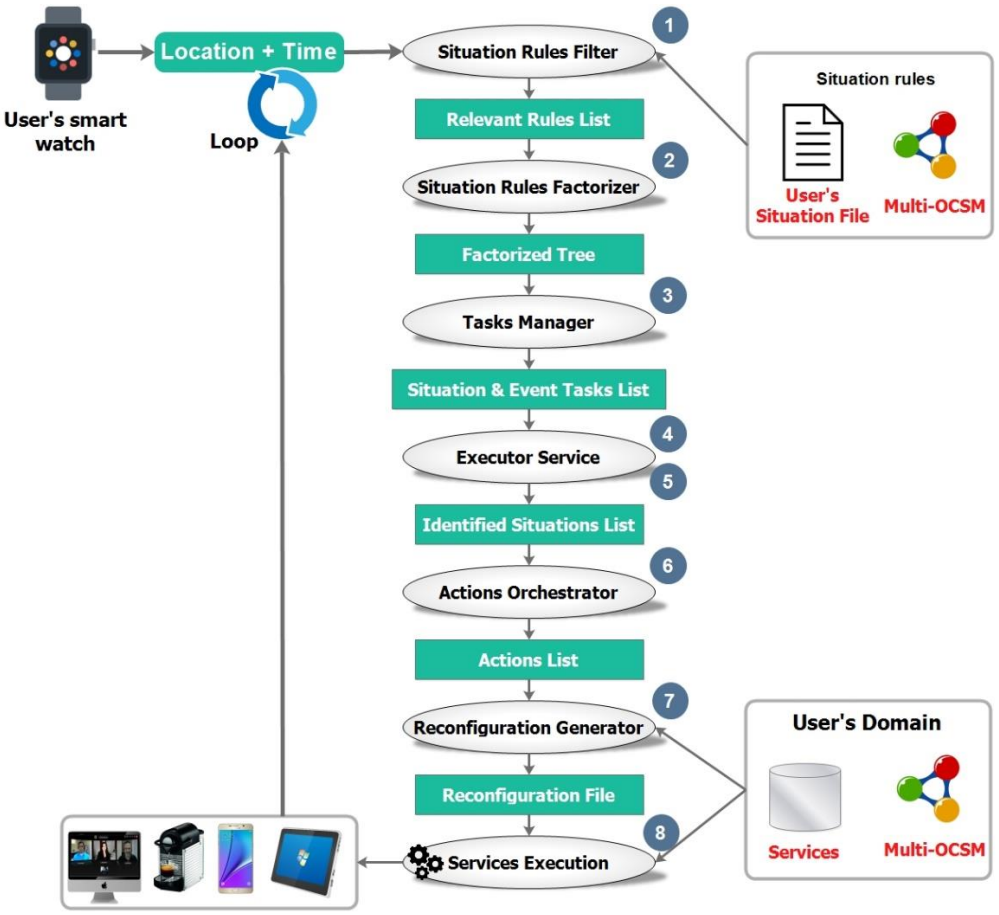


Figure 5.1: Proposed composite situation reasoning process.

The dynamic semantic-based composite situation identification process takes as inputs situations rules registry, user's domain, user's current location and time, and provides as output identified situations list and actions reconfiguration file. The process complete steps are detailed as follows.

The first step consists of filtering relevant situation rules according to new incoming events and context changes. The second step is based on the construction of the tree that factorizes common filtered situation rules with their attributes. The factorized tree gives us a semantic decomposition and accord between simple/composite situation and their context attributes. The third step consists of creating and encapsulating situation rules and their context attributes in micro-service classes. The fourth step consists of ordering the blocked situation rules by attributing each one a priority in order to forward them for execution. The fifth step is to run parallelly micro-services into threads to ensure the parallel identification of simple and composite situations. Threads management is going to be ensured through different synchronization mechanisms (e.g., semaphores, locks, monitors, atomic operation, etc.). The sixth step consists of service orchestration. The seventh step consists of services reconfiguration and deployment. The last step consists of reading the reconfiguration file and execute available services.

5.2.1 Dynamic filtering of relevant situation rules

As detailed in Algorithm 1, the Situation Rules Filter component provides relevant user's situation rules list filtered by the user's current location and time (see Figure 5.2). Initially, a list of the user's relevant rules contains a set of anytime and anywhere rules (line 1). For each iteration, we add a situation rule '*s*' from the user's situation rules registry to the relevant situation rules list (lines 2-14) that match the user's current location and time.

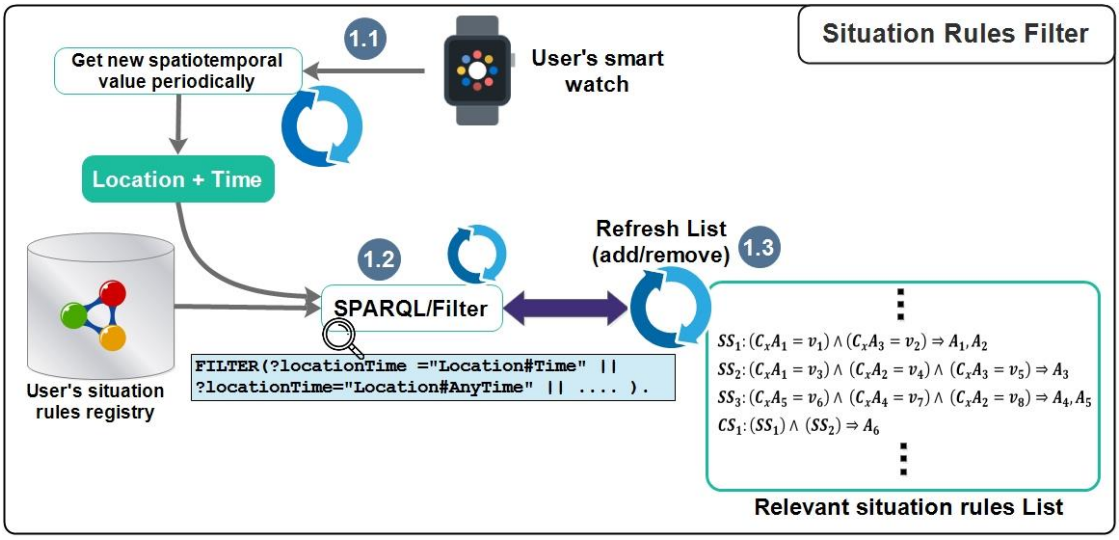


Figure 5.2: Filtering relevant situation rules.

We distinguish three main different spatiotemporal classes rules:(1) the rule-based localization and time that depends on both current user's location and time (lines 3-4), (2) the time-based rule depends only on the current time (lines 6-7), and (3) the localization-based rule depends only on the current user's location (lines 9-10). A list of relevant situation rules will be sent to the Situation Rules Factorizer component for further processing.

Algorithm 1: Dynamic filtering of relevant situation rules

```
Inputs:
        Current user's location: Location;
        Current user's time: Time;
        User's situation rules registry: Rules_registry;
Outputs:
        Relevant list of user's situation: Filtered_user_rules_vector = Ø;   /*each 20 milliseconds
                                                we refresh this vector in case of context changes */
Begin
1:   Filtered_user_rules_vector.add(Rules_registry.pullAnyWhere_AnyTime_Rules());
2:   For each s from Rules_registry do
3:      If (s.getLocation().closet(Location) && s.getTime().isCloseTo(Time)) then
4:          Filtered_user_rules_vector.add(s);
5:      else
6:          If (s.getLocation() == "Anywhere" &&  s.getTime().isCloseTo(Time)) then
7:              Filtered_user_rules_vector.add(s);
8:          else
9:              If (s.getLocation.isCloseTo(Location) &&  s.getTime() == "AnyTime") then
10:                 Filtered_user_rules_vector.add(s);
11:             end if
12:         end if
13:     end if
14:  end for
15:  return Filtered_user_rules_vector;
End
```

5.2.2 Construction of dynamic situation rules-based factorized tree

The Situation Rule Factorizer component transforms the relevant situation rules list into a factorized tree as detailed in Algorithm 2. This tree ensures an optimal representation of composite situations structure in order to enhance the situation identification process. The factorized tree consists of three layers: composite situations layer, simple situations layer and common context attribute layer for all situations. Figure 5.3 presents an illustration of a composite situation, three simple situations, and five contexts attributes. The factorized tree promotes the efficiency and the effectiveness of the reasoning process of the situation rules. The Situation Rule Factorizer component inspects if the new situation is yet to exist then creates a new root branch for it (lines 2-4) as a start node for the tree traversal of the next step. Alternatively, in case of a composite situation that has sub situations, if a sub situation already exists in the tree, the Situation Rule Factorizer component will assign it to its parent composite situation (lines 6-16) to construct composite situations progressively. Then it aggregates the common context attributes of the situation rules, which are created in the tree leaves. A context attribute may be assigned to one or more situations (lines 17-25) to dispatch sensed events to shared situations. This process is repeated until all the relevant situations have been treated. When Algorithm 2 finishes, it produces the factorized tree as shown in figure 5.3.

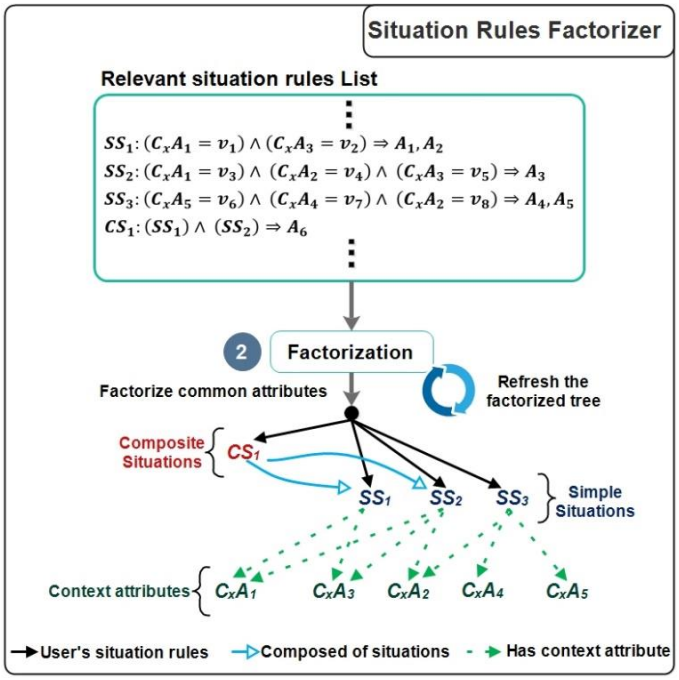


Figure 5.3: Construction of factorized tree from relevant situation rules list.

Algorithm 2: Factorized tree Construction

Inputs: Relevant list of user's situation: Filtered_user_ rules _vector;
Outputs: Factorized tree: Factorized_Tree = Ø;
Begin

```
1: For each rule from Filtered_user_rules_vector do
2:          If (Not(Factorized_Tree.hasSituation(rule.getSituation()))) then
3:                    SituationNode sn = new Node(rule.getSituation());
4:                    Factorized_Tree.addSituationNode(sn);
5:          end if
6:          If (rule.hasCompositeSituations()) then
7:              For each subSituation from rule.getSubSituations() do
8:                    If (Not(Factorized_Tree. hasSituation(subSituation))) then
9:                         SituationNode sn = new Node(subSituation);
10:                        Factorized_Tree.addSituationNode(sn);
11:                        Factorized_Tree.getSituation(subSituation).linkWith(rule.getSituation());
12:                   else
13:                        Factorized_Tree.getSituation(subSituation).linkWith (rule.getSituation());
14:                   end if
15:             end for
16:         end if
17:     For each attribute from rule.getAttributes() do
18:             If (Factorized_Tree.hasAttribute(attribute)) then
19:                   Factorized_Tree. getAttribute(attribute).linkWith(rule.getSituation());
20:             else
21:                   AttributeNode an = new Node(attribute);
22:                   Factorized_Tree. add(an);
23:                   Factorized_Tree.getAttribute(attribute).linkWith(rule.getSituation());
24:             end if
25:       end for
26: end for
27: return Factorized_Tree;
End
```

5.2.3 Parallel creation of events and situations microservices from factorized tree

In this step, the Task Manager component encapsulates each event monitor entity and situation identifier entity in a single deployable unit of code known as a microservice entity (i.e., task implements the *Runnable* interface of Java API) as detailed in Algorithm 3 (see figure 5.4). This encapsulation promotes autonomy, flexibility, reuse of code, and ensures a decomposition scale of different simple/composite situations and simple/complex events. Two types of microservices are used: the event listener microservice for monitoring sensed data, and the situation identifier micro-service for identifying user's situations. The Task Manager component explores the factorized tree to encapsulate situations in situation microservices entities taking into account all the dependencies between the composite situations and their simple situations (lines 1-9) that allow all granularity levels of situation microservices. Then we iterate all common context attributes of the tree leaves to encapsulate them in event microservices entities (lines 10-13). Later, all microservices will be sent to the Executor Service for submission (see figure 5.4).

Algorithm 3: Creation of events and situations microservices from the factorized tree

```
Inputs: Factorized tree: Factorized_Tree;
Outputs: List of events tasks:  Event_Task_List = Ø;
         List of situations tasks:  Situation_Task_List = Ø;
Begin
1:   For each situation from Factorized_Tree.getSituations() do
2:       If(situation.isComposite()) then
3:           SituationMicroService situationTask = new CompositeSituationMicroService(situation);
4:           Situation_Task_List.add(situationTask);
5:       else
6:           SituationMicroService situationTask = new SimpleSituationMicroService(situation);
7:           Situation_Task_List.add(situationTask);
8:       end if
9:   end for
10:  For each attribute from Factorized_Tree.getAttributes() do
11:      EventMicroService eventTask = new EventMicroService(attribute);
12:      Event_Task_List.add(eventTask);
13:  end for
End
```

5.2.4 Priority-ordered blocking queue

Once there is an emergency situation (such as heart-situations) with lives at stake, time is a crucial factor for a health and social organization where the platform searches as quickly as possible for the closest available services to prevent damage and serious health evolution. A priority value is set by an expert in the offline (e.g., a doctor specifies high priority value in case of a serious health situation) and the situation is updated online using a new weight based on the situation's category, user's activity, user's location, and situation's coverage (see Chapter 4 Section 4.2.5).

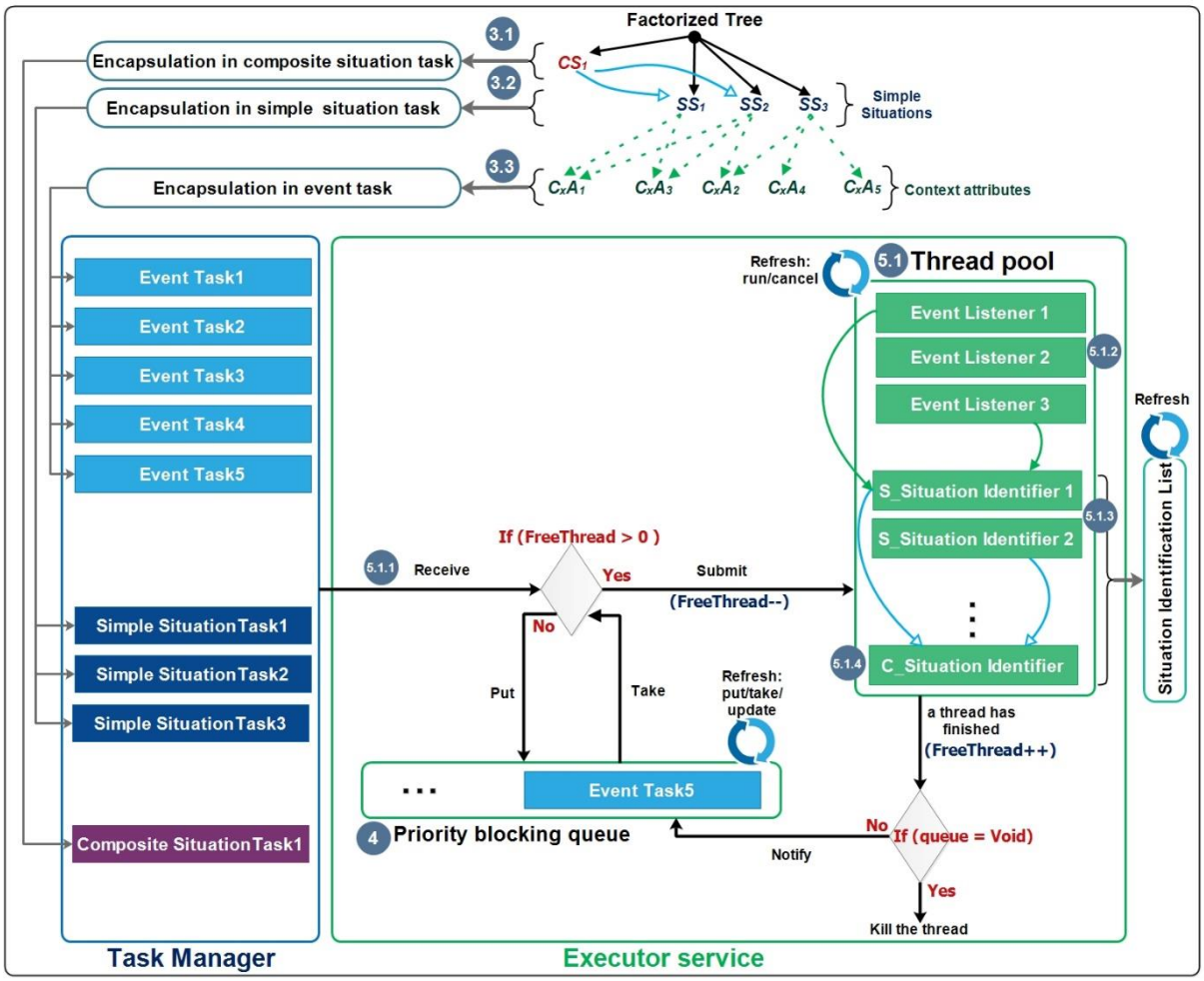


Figure 5.4: Parallel situations identification process.

5.2.5 Threads pool and parallel identification of simple and composite situation

In this step, the Executor Service component is responsible for exploring how intelligently microservices can be synchronized to identify new situations and how they can share the detected events to a group of situations identifiers. We use several structures including events buffers, hierarchical reasoning, thread pool, and priority-ordered blocking queue. We intend to optimize the number of missed situations and reduce the checking time by implementing a parallel approach. Another benefit is that our parallel approach exploits all available processor cores. Instead of only one processor executing the whole process of situations identification, we will split the process into several microservices, each one of them is going to be executed simultaneously by different processors.

The Executor Service component creates a thread pool and monitors the execution of the concurrent events listeners, composite and simple situations microservices. This step includes microservices reception and submission, concurrent context collection (occurred events), and parallel situations identifications (see the right side of figure 5.4).

a) Receiving and submitting microservices

The Executor Service component provides improved performance when executing large numbers of events and situations microservices simultaneously. It iterates both lists of events and situations tasks in order to send them for execution as detailed in Algorithm 4 (lines 2-13). The Executor Service component manages all the process of tasks execution from reception until submission using the submit function (lines 4 and 10). When receiving a new task for execution, the Executor Service component verifies if there is a free thread in the pool. If there is a free thread, then associate the task to this thread directly for launching. Otherwise, send the task to the priority-ordered blocking queue (see figure 5.4). The task will be placed in the right place in the queue based on its priority and waits until there is a free thread in the pool. The task with the highest priority in the queue (i.e., urgent task) will be pulled first and sent to the pool for execution.

Algorithm 4: Receiving and Submitting tasks algorithm

```
Inputs:
        List of events tasks:  Event_Task_List;
        List of situations tasks:  Situation_Task_List;
Outputs:
        List of submitted tasks in the executor: Submitted_Task_List = Ø;
Begin
1:  ExtendedThreadPoolExecutor executor = new ExtendedThreadPoolExecutor();
2:  For each situationTask from Situation_Task_List do
3:        If (Not(Submitted_Task_List.has(situationTask))) then
4:              executor.submit(situationTask);
5:              Submitted_Task_List.add(situationTask);
6:      end if
7:  end for
8:  For each eventTask from Event_Task_List do
9:        If (Not(Submitted_Task_List.has(eventTask))) then
10:             executor.submit(eventTask);
11:             Submitted_Task_List.add(eventTask);
12:       end if
13:  end for
Return Submitted_Task_List;
End
```

b) Context data monitoring

The event listener thread is continuously monitoring and capturing new context data (i.e. event formatted as <event time, sensed value, appearance time, event valid period>) from sensors as detailed in Algorithm 5. After the reception of a new event, it will be sent to the associated situations identifiers threads to identify the user's situations (line 6). In this step, we can notice that events listeners threads are executed in a parallel way. The synchronization between two or more threads is ensured to avoid any data race condition.

Algorithm 5: Event listener thread algorithm

```
Inputs:
        Factorized Tree: Factorized_Tree:
        Attribute: CxA;
Outputs:
        Sensed value of event: event;
Begin
1:   Boolean loop = true;
2:   While (loop) do
3:       event = CxA.receiveEvent(); /* receive event from sensor */
4:       For each situation from Factorized_Tree.getSituationRules() do
5:           If (situation.hasAttribute(CxA)) then
6:               situation.getBuffer().add(event));
7:           end if
8:       end for
9:       sleep(CxA.getSensor().getSensorFrequency().toMillis()); /*sleep, for example 20 ms)*/
10:  end while
End
```

c) Dynamic identification of simple situations

The pseudo-code of Algorithm 6 describes the simple situation identification process. With the concurrent simple situations identifiers threads, we have the possibility to identify several situations at the same time. The simple situation identifier component receives the occurred events and then checks them through simple situation conditions. Finally, it returns the situation's checked value. This is done in the following four major steps:

- Step1 (lines 4-6): When a new event is received, the simple situation identifier thread checks the simple situation's condition and returns the situation's checked value.
- Step2 (lines 7-9): During the check process of the simple situation's condition, if the simple situation thread finds at least one expired event (i.e. an event that exceeds its lifetime), it increments the number of missed situations and the number of expired events.
- Step 3 (line 11): The simple situation identifier calls the reconfiguration service that will trigger appropriate action services to the user.
- Step 4 (lines 12-14): Finally, the simple situation identifier thread shares the situation's checked value with its associated composite situation threads through buffers.

Algorithm 6: Simple Situation identifier thread algorithm

```
Inputs:
        Simple situation: SS;
        Factorized tree: Factorized_Tree;
Begin
1:    SS.setChecked(false);
2:    Boolean loop = true;
3:    While (loop) do
4:          Event event = SS.getBuffer().getEvent();  /*wait for receiving new event */
5:          SS.refreshNewEvent(event);
6:          SS.check();
7:          If (SS.isMissed()) then
8:              SS.incrementMissedSituation();
9:              SS.incrementExpiredEvent(SS.getNumberOfExpiredEvents());
10:         else
11:             Reconfig();
12:             For each compositeSituation from Factorized_Tree.getCompositeDependsOn(SS) do
13:                             compositeSituation.getBuffer().add(SS);
14:             end for
15:       end if
16:   end while
End
```

d) Dynamic identification of composite situations

The Composite Situation Identifier thread is responsible for identifying composite contextual situations based on the user's situations rules. It involves the collaboration of its sub situations identifiers' threads that share their situations' checked values with their parent composite situation. Algorithm 7 describes the composite situation identification steps. It receives as inputs the occurred events and the situations' checked values. Then it verifies the set of conditions and sub situations of the composite situation. Finally, it returns the situation s' checked value. This is done by following the four major steps:

- Step 1 (lines 4, 8 and 22): The first step relates to the detected events and situations checked values received from the appropriate events listeners threads, and situations identifiers threads respectively (line 4). The composite situation identifier thread verifies the set of conditions and sub situations of the composite situation (lines 8 and 22) and returns the situation's checked value.
- Step 2 (lines 9-11 and 23-25): During the check process of the composite situation, if the composite situation thread finds at least one expired event (i.e. an event that exceeds its lifetime), it increments the number of missed situations and expired events.
- Step 3 (lines 13 and 27): The composite situation identifier thread calls the reconfiguration service that will trigger appropriate action services for the user.

- Step 4 (14-16 and 28-31): Finally, the composite situation identifier thread shares the situation's checked value with its associated composite situation threads through buffers.

Algorithm 7: Composite Situation identifier thread algorithm

```
Inputs:
        CompositeSituation: CS;
        Factorized tree: Factorized_Tree;
Begin
1: CS.setChecked(false);
2: boolean  loop = true;
3:  While (loop) do
4:    Object object = CS.getBuffer().getObject(); /*wait for receiving new object*/
5:    If(object instanceof Event) then
6:          Event event = (event) object;
7:          CS.refreshNewEvent(event);
8:          CS.check();
9:         If(CS.isMissed()) then
10:               CS.incrementMissedSituation();
11:               CS.incrementExpiredEvent(CS.getNumberOfExpiredEvents());
12:        else
13:              Reconfig();
14:              For each Dependent_CS from Factorized_Tree.getCompositeDependsOn(CS) do
15:                        Dependent_CS.getBuffer().add(CS);
16:              end for
17:        end If
18:   else
19:         If(object instanceof Situation) then
20:              Situation situation = (Situation) object;
21:              CS.refreshSubSituation(situation);
22:              CS.check();
23:              If(CS.isMissed()) then
24:                    CS.incrementMissedSituation();
25:                    CS.incrementExpiredEvent(CS.getNumberOfExpiredEvents());
26:              else
27:                    Reconfig();
28:                    For each Dependent_CS from
29:                          Factorized_Tree.getCompositeDependsOn(CS) Do
30:                                Dependent_CS.getBuffer().add(CS);
31:                    end for
32:              end If
33:        end If
34:    end If
35:  end while
End
```

5.2.6 Orchestration of the situation's actions

The Actions Orchestrator scans the smart environment with the help of the Kali-smart middleware [9] in order to know the available devices and orchestrate the best way to deploy the microservices. It takes into account the identified situations' list and the user's domain to ensure the multimodality interactions and find the appropriate device.

5.2.7 Services reconfiguration and deployment

It receives the orchestrated actions provided by the Actions Orchestrator component and generates the output reconfiguration file.

5.2.8 Services execution

The Services Execution component consults the available microservices located in the Knowledge micro-services to execute them. For example, the availability of light switch service.

5.3 Use cases and scenarios

In this section, we illustrate the proposed approach using a motivating scenario described in the general introduction. The possible scenarios are varied and vast. We used our framework to deal with several situations in four smart domains (smart health, smart home, smart car, and smart office). We intend to support the user's mobility in different smart domains using the situation filtering process (add/remove/migrate services) basing on the user's spatiotemporal information. First of all, we assume that the user has a registry of all his situation rules classified as urgent/daily-life in our Multi-OCSM ontology. As shown in table 5.1, the user has four domains (Health domain, Security domain, Home domain, Office domain). We can distinguish two types of situation rules: Simple Situation (SS) and Composite Situation (CS).

A first scenario involves the user, Mohammed, who can be located in his smart home. The system filters relevant situations rules according to the current user's location and time. Figure 5.5 shows a list of relevant home situation rules (e.g., home fire situation, shower situation, etc.) and health situation rules that are going to be checked anywhere and anytime (e.g., high glucose level situation, diet is not a good situation, etc.). The list of relevant situation rules is passed to the Situation Factorizer component for further processing. This list of relevant situation rules will be updated at each filter by adding and/or removing situation rules.

Table 5.1: User's situation rules registry.

Smart Domain	Situation ID	Situation Name	Situation Rule
Health	SS_0	High Glucose Simple Situation	IF [GlucoseLevel = 'High'] THEN Deploy[(Inject_Insulin_Service)]
	SS_1	Low Glucose Simple Situation	IF [GlucoseLevel = 'Low'] THEN Deploy[(Inject_Glucagon_Service)]
	CS_0	Diet Is Not Good Composite Situation	IF [Recurrence(SS_0; 10; 5; day)] THEN Deploy[(Diet_Is_Not_Good_Service)]
	CS_1	Risk Of Death Composite Situation	IF [Sequential(UserPhysicalState = 'Rest'; SS_0; HeartBeating = 'High')] THEN Deploy[(Risk_Of_Death_Service)]
Security	SS_2	Home Fire Simple Situation	IF [HomeTemperature = 'VeryHigh'] THEN Deploy[(Firefighting_Service)]
	SS_3	Home Intrusion Simple Situation	IF [Overlay(Location = 'Outside' && HomeMotion= 'ture')] THEN Deploy[(Intrusion_Home_Service)]
Home	SS_4	Wake Up Simple Situation	IF [Sequential(Time = 'Wake up time'; WorkDay ='On')] THEN Deploy[(Wake_Up_Service)]
	CS_3	Shower Composite Situation	IF [Sequential(SS_4; UserState = 'waked up'; Location = 'Bathroom'; Time = 'Shower time')] THEN Deploy[(Shower_Service)]
	CS_4	Drink Coffee Composite Situation	IF [Sequential(CS_3; Location = 'Kitchen')] THEN Deploy[(Coffee_Machine_Service)]
	CS_5	Check Email Composite Situation	IF [Sequential(CS_4; Location = 'Study')] THEN Deploy[(Email_Service && News_Service)]
	CS_6	Going To Work Composite Situation	IF [Sequential(CS_5; Location = 'Garage')] THEN Deploy[(Car_Controlling_Service && Car_GPS_Service && Garage_Controlling_Service)]
Office	SS_5	At Work Simple Situation	IF [Location = 'Office'] THEN Deploy[(Work_Service)]
	SS_6	Meeting Simple Situation	IF [Overlay(Location = 'Meeting room' && Time = 'Meeting time')] THEN Deploy[(Meeting_Service)]

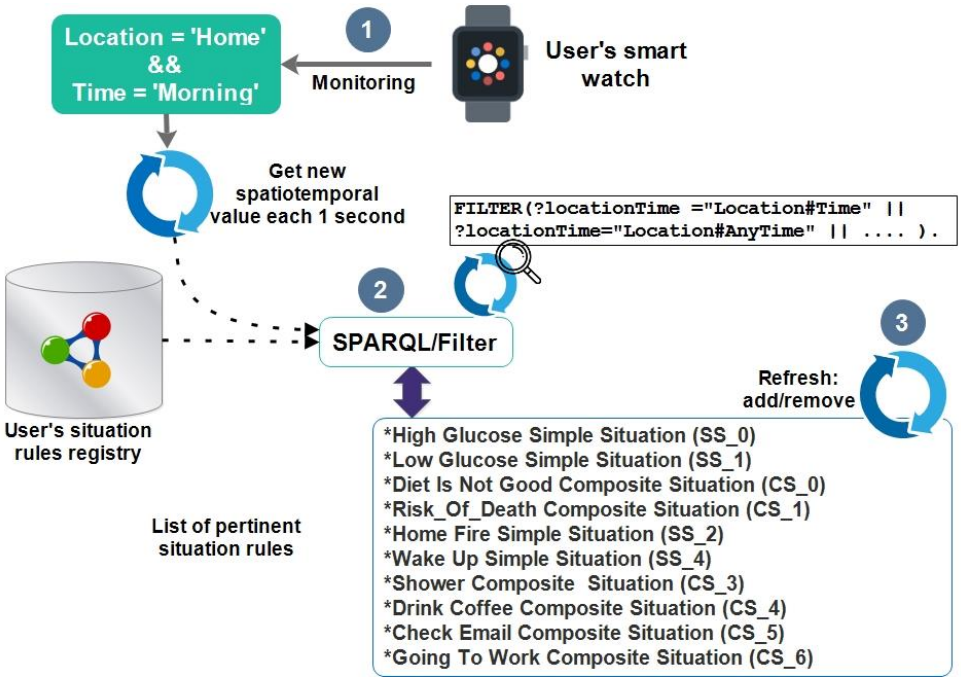


Figure 5.5: Illustrative example of relevant situations rules filtering.

After the situation filtering process, the Situation Factorizer component factorizes the list of relevant home's situation rules into common context attributes and situation rules in order to remove duplicate situation rules and context attributes. This factorization is useful for creating a loose coupling of events/situations and for optimizing the situations' identification process. Then, the Situation Factorizer component builds a factorization tree and creates dependency links between the situation rules and their context attributes or with other situation rules, as shown in Figure 5.6.

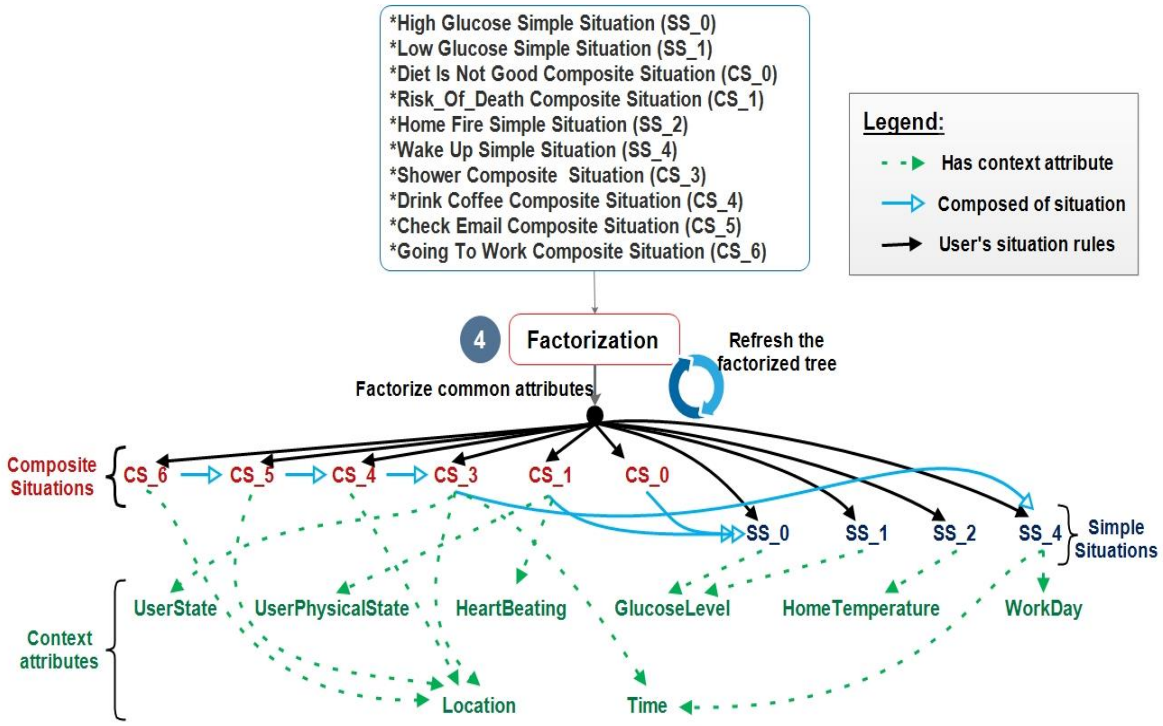


Figure 5.6: Construction of factorized tree.

After the factorization process, the factorized situation rules and their context attributes (i.e., event) will be encapsulated in small runnable microservices-based tasks. Task Manager takes these tasks and sends them to the *Executor Service*. The *Executor Service* has a thread pool and implements a priority-based blocking queue where it put pending tasks (see figure 5.7). If the number of free threads in the pool equals "0" then the submitted tasks will be put in the blocking queue. Otherwise, the submitted tasks will be sent to a free thread, and the number of free threads will be decremented by one. When a thread finishes its execution, the number of free threads will be incremented by one and the thread will be killed if and only if the blocking queue is empty. Otherwise, the priority blocking queue will be notified to take a pending task and passing it to submission.

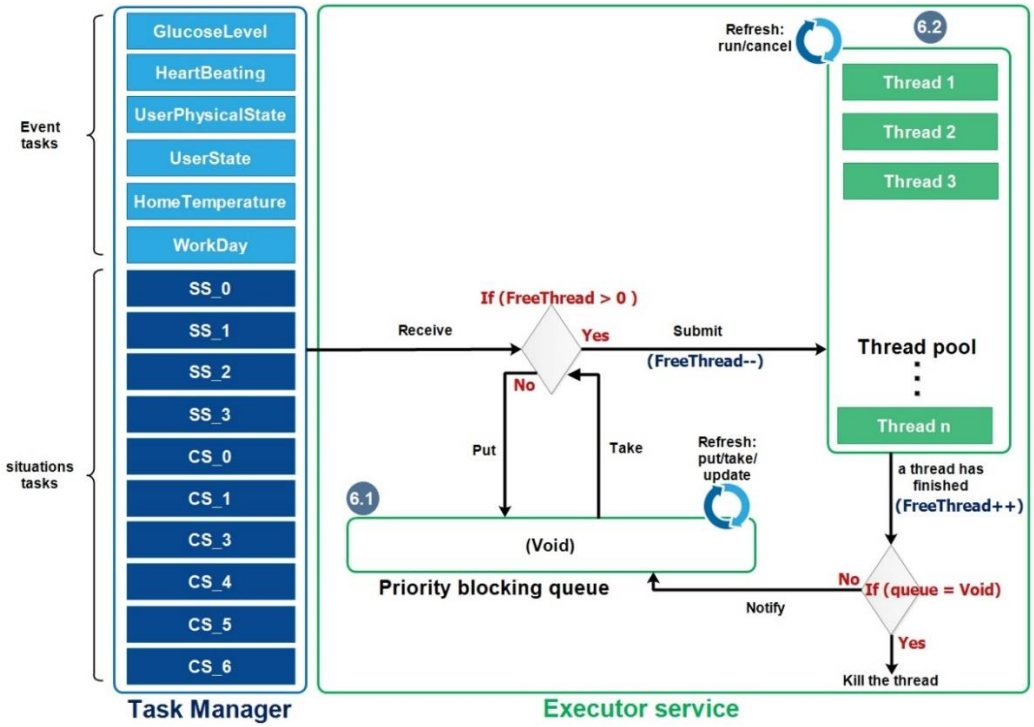


Figure 5.7: Submission of events and situations tasks.

A second scenario is when a user arrives at his office, a notification is sent to the Situation Filter Component, and the list of pertinent situation rules is dynamically updated (see figure 5.8). Of course, his office situation rules will be added to the list (e.g., at work situation, meeting situation). Previous home's situation rules will be removed (e.g., shower situation, drink coffee situation, etc.). The system deploys meeting services on his desktop PC with all tools that he can find them useful in his working domain. All this while he can always check the state of his health (e.g., glucose level, if it is high then deploy insulin service).

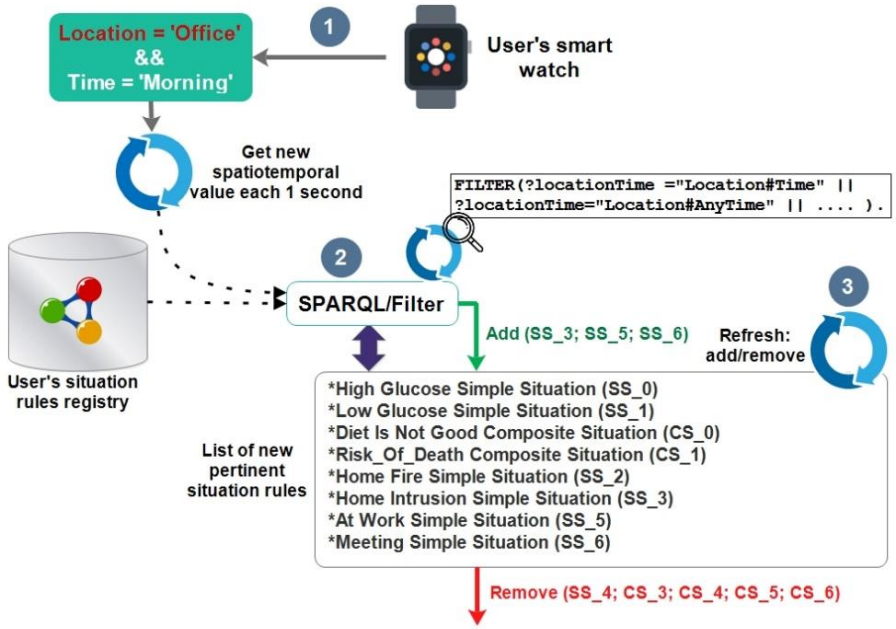


Figure 5.8: Refreshing of relevant situation rules list.

5.4 Conclusion

During this chapter, we present a composite situation identification process based on ontology and parallel approach in order to help user to have an adequate reconfigurable distributed mobile application. First, we give a detailed parallel approach for composite/simple situation identification exploiting all processor cores. In addition, we show how we filter and factorize user's situations rules through Multi-OCSM ontology. To perform our system and respond to urgent situations rapidly, a priority strategy is proposed for speeding up the queued urgent situation rules that run simultaneously across collaborating microservices. Finally, we illustrate the proposed approach using a motivating scenario using both simple and composite situation rules in order to illustrate the identification process and its different steps.

As next chapter, we aim to propose situations enrichment mechanism in order to achieve the complete work in research task.

Chapter 6

Dynamic Situation Enrichment and Adaptation Mechanism in Pervasive Environments

Summary

6.1 Introduction

Currently, we are living in the era of ubiquitous environments, which introduces the possibility to serve users in multiple context situations. Yet, with this large variety of situations, diversity of preferences, and multiplicity of devices to respond to users' needs, it becomes a necessity to have an effective solution that ensures the correct identification of specific situations. Otherwise, the identification process may be incomplete as it could be based beforehand on predefined situation rules, identified either by users or by the developers. As we see in figure 6.1, there are lot of gaps in the user's agenda that should be specified by the user himself. Moreover, it is a cumbersome task for each user to identify his specific rules in any daily life cases. This emerges the need to recommend effectively the right situation rule in the right context for the right person. In fact, a rule-based recommendation system is used to suggest meaningful rules and has the ability to guide users to useful and relevant rules, which meets their needs in a large space of available rules.

Nowadays, with the rapid development of both social, smart environments and their related technologies, a huge volume of metadata from crowdsourcing is available, such as context-aware properties, a large variety of situations, diversity of needs, a multiplicity of devices to respond to users'

needs, etc. This overloaded information is one of the main reasons that trigger the construction of an efficient recommendation system based on situation rules. Traditional recommender systems are mostly built to provide recommendations only based on user preferences, using the content or a collaborative approach and neglect situational context information, such as location, time and role. Recently, several surveys and studies in the field of rule-based recommendation [107] [108] have only focused exclusively on the techniques applied in the recommendation process regardless of the following questions: what is the context information taken into consideration for a better understanding and an effective result? Which device is the best to use considering the immediate situation? In this scope, to recommend the right situation rule in the right context for the right person, a context-aware semantic model for rule-based recommendation system needs to be considered.

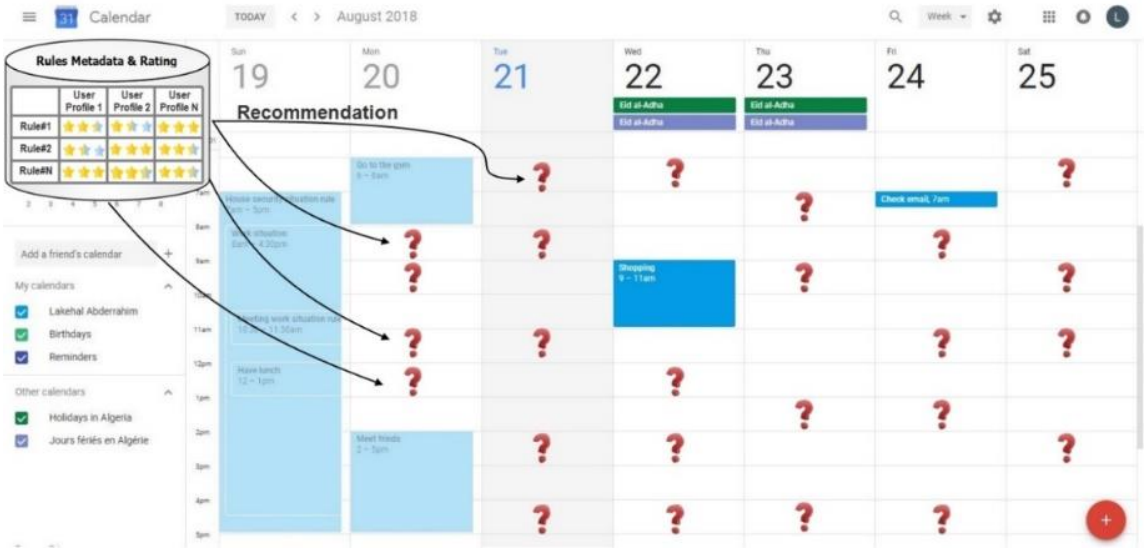


Figure 6.1: User' agenda**.**

In this chapter, we propose a new ontology-based context-aware recommendation system for dynamic situation rules enrichment. Thus, we extend our Multi-OCSM ontology to provide unified representation and classification of situations rules. Also, it provides a formal way to explicitly specify common meaning of users' contexts, their preferences (e.g., preferred activities) and their constraints (e.g., available time) while respecting the device constraints (e.g., device characteristics or device capacity). In this way, the system can exploit users' experiences and offers to-users an auto-complete daily life user requirement (e.g., his needs in the shopping center, at work, at home, etc.) with better-customized user experience. Moreover, a new situation rules' learning process is proposed to classify situation rules according to ontology rule before applying the recommendation process to achieve a high quality of recommendation. The learning process constituted of rules profiles collector, rules filter, rules analyzer and rules classifier. This classification is based on semantic rules ontology that helps in reducing the searching time and enhances the recommendation by offering useful and relevant rules depending on user's context and

preferences. Thus, the recommendation system considers three context categories: user's preferences, situation context and device's capability. The recommendation process built on three main steps. Starting by the calculation of the similarity of both user's explicit and implicit preferences. Passing to calculating the probability for recommending a situation rule corresponding to user's situation context, through a Bayesian-classifier, that focuses on the user's role in a specific location and time and in a particular domain (e.g., shopping, working, driving, etc.). Finally, the selection of the appropriate rules list by checking the semantic coherence of the devices capabilities constraints with the recommended rules requirements based on SWRL rules. To illustrate the applicability and adaptability of our approach, we present a case study using user's context preferences to detail how the whole process works with our recommendation system.

The rest of this chapter is organized as follows. We provide some useful concepts for situation rules enrichment. After that, we present the context-aware ontology-based recommendation system and the situation enrichment mechanism. Then we illustrate our process with a use of case. Finally, we conclude this chapter with a conclusion.

6.2 Situation-based contextual model: definitions and formalizations

Situations are increasing in unprecedented ways in different areas of pervasive environments. Users can be found in different situations surrounded by various mobile devices, such as personal mobile phones to accomplish their daily tasks. In practice, the specification of a huge number of situations for each user is quite difficult. However, previous studies [107] [108] reveal that the recommendation of the right situation rule in the right context is also a real problem for users, even beyond the users' experiences. Therefore, it is necessary to recommend useful rules through a recommendation process according to the available devices with current context to enrich automatically user's agenda. In this context, there are three types of recommendation systems: content-based filtering, collaborative filtering, and hybrid filtering. In this work of recommendation of relevant situation rules, ontology and hybrid filtering with Bayesian-classifier can play a crucial role to implement context-aware rules classification that enhance relevancy of suggested rules.

Realizing an ontology-based context-aware recommendation system for situations enrichment and adaptation gives two main benefits for both semantic and relevance recommendation. The first benefit is achieved through the generic and flexible ontology model Multi-OCSM to unify the representation of situation rules as well as the sharing and classification of context rules. The second benefit is attained through the combination of three context categories: user preference, situation context, and device capability with the ultimate goal of covering a multidimensional recommendation space and performing the recommendation system. To handle with altogether the three context categories, we deployed three different approaches and then combined them to create a hybrid approach.

6.2.1 The multidimensional recommendation space modeling

The recommendation model provides situation rules over multiple dimensions to generate output. As input, we define location, time, role and user preference as user context dimensions. In addition, we define CPU speed, RAM size, storage, network, and so on as device capability dimensions. Besides, we define rule item dimension, which describes the rule's content. These input dimensions are the base to develop the recommendation system intended to suggest useful and relevant situations rules to the user based on his context (i.e., user location, user roles) and his device capabilities. The dimension of user context reflects different spatiotemporal information related to prior usage rules as well as the assigned rules scores, information related to possible roles that a user can play in the smart space (driver, student, etc.), and information related to user's explicit and implicit preferences. The rule item dimension consists of multiple projections on different context attributes. Each projection is a combination of *location, time, role, etc.* The device dimension defines device's features and their capabilities. As output, we can define the rule item's score. The score represents the degree of user interest in the recommended situation rule item, ranging from -1 (least preferred) to 1 (most preferred) (see figure 6.2). In another word, the recommendation score is obtained as a combination of these inputs as follows:

$$R(UserContext, RuleItem, DeviceCapability) = score$$

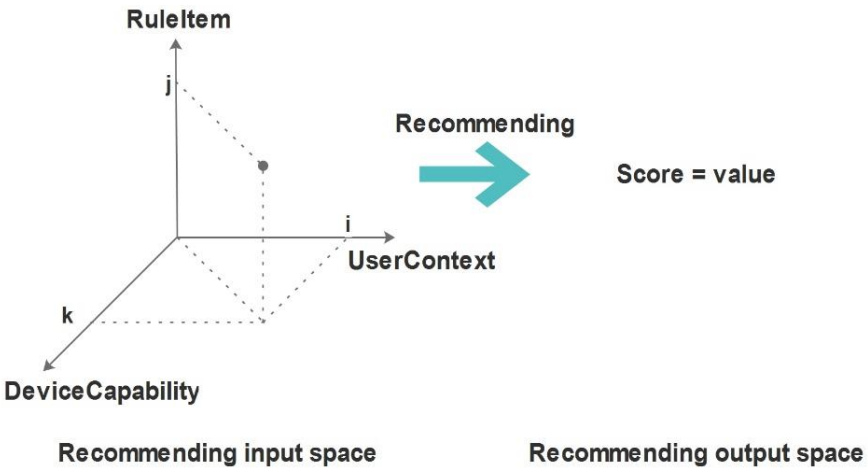


Figure 6.2: Proposed multidimensional recommendation space.

We assume as inputs:

- *UserPreference (i)* = {(security; 0.67), (health; 0.43), (home; 0.55), (work; 0.43)}.
- *RuleItem (j)* = {(domain;" home"), (day; "weekday"), (field; "security")}.
- *DeviceCapability (k)* = {(CPU; 4.5 Ghz), (RAM, 4 GB), (Modality, [text, audio, image, video])}.

We obtain as output:

- *Score* = 0.88.

6.2.2 The user context profile formalization

The user context profile contains the user-related information: location, time, agenda and role (e.g., role is *'student', 'citizen', 'driver',* etc.). User's role can be identified continuously using location, time and agenda:

$$\exists\, l, t \,/\, agenda\big(location(l); time(t)\big) \rightarrow\ Role_i$$

In another case, the location is not necessary to identify a user's role, so in such situation, we can use only the time and agenda:

$$\exists\ \ t \,/\, agenda\big(time(t)\big) \rightarrow\ Role_i$$

6.2.3 The rule preference formalization

The rule preference denotes explicit and implicit user's preference regarding the rule content. It is defined by the couple <*term, weight*>. The *Weight* ∈ [-1; 1] indicate the degree of the user interest in such terms (e.g., <*Home; 0.85*>; <*Health; −0.75*>).

6.2.4 The device context

The device context presents device capability (i.e., CPU speed, RAM size, storage space, network bandwidth and protocols).

6.2.5 The situation rules formalization

Situation is the basic concept of any context-aware application. It is the set of events that are happening at a given location (where?) and time (when?), and the conditions (rules) that can verify those events, thus identifying the situation. Each user has an agenda that in turn contains user situations. Each situation is detected by a situation rule.

At the abstract modeling level, the rule is represented by a combination of data and concepts. A rule is a combination of multiple projections. Each projection has different axes (e.g., Location, Time, Role, Sensor Value, etc.). The axis can attach with primitive (e.g. *inside, outside, near* and *far* for Location; *before, after, while* and *equal* for Time; *is* for Role; etc.) to enrich the description of the rule. Moreover, there may be exceptions that can give another dimension to rule modeling. Besides the process of situation identification, we need to know what the user expects when this situation happens (i.e., situation rule is verified). So, we need to determine which actions to activate when the situation is identified through a semantic link between a situation and actions. A situation rule is presented textually in the following format, where *P* is a projection.

$$\boldsymbol{R : Name\ ;\ P_1[Axis(Primitive(value, tolerance)\ \ldots);\ \ldots]}$$
$$\boldsymbol{P_n\ [\ldots]\ \ EXCEPT[Situation/Axis(\ldots)\ ] \rightarrow}$$
$$\boldsymbol{Actions[Action_1, Action_2, \ldots]}$$

6.2.6 Classification of situation rules

We propose to classify the situation rules according to their content.

- **Role-based rules:** This type of rules depends only on the role of user. For example, *R1* (*Work*) that verifies when the time is equal to the Planed Time (*PT*) "work" and user's role is a worker. The exception is being on holiday. This rule is described as follows:

$$\boldsymbol{R1 : Work\ ;\ P_1[Time(PT(work)); Role(is(worker))]}$$
$$\boldsymbol{EXCEPT[Situation(holidays)\ ] \rightarrow}$$
$$\boldsymbol{Actions[deploy\ work\ space, put\ the\ phone\ silent]}$$

- **Localization-based rules:** This type of rules depends only on the location of user. For example, *R2* (*Alarm home*) verifies when the user is either outside his home with 20 meters tolerance (*P1*) or if the time is between 21h and 6h with 20 minutes tolerance (*P2*). The exception is being on Sunday. *O* and *I* represent the location axis primitives outside and inside, respectively. *A* and *B* represent the time axis primitives after and before, respectively. This rule is described as follows:

$$\boldsymbol{R2: Alarm\ home\ ;\ P_1[Location(O(Home, 20))]\ OR}$$
$$\boldsymbol{P_2[Time(A(21,30)B(6,30); Location(I(Home, 10)]}$$
$$\boldsymbol{EXCEPT[Time(Sunday)\ ] \rightarrow}$$
$$\boldsymbol{Actions[deploy\ security\ alarm\ application]}$$

- **Rules-based on localization and role:** This type of rules depends on both location and role of user. For example, *R3*(*Meeting*) situation rule verifies when user is inside the meeting room with 2 meters tolerance and time axis is equal to the *PT* meeting and his role axis is worker.

$$\boldsymbol{R3 : Meeting\ ;\ P_1[Location(I("MeetingRoom, 2))}$$
$$\boldsymbol{Time(PT(meeting)); Role(is(worker))] \rightarrow}$$
$$\boldsymbol{Actions[deploy\ meeting\ application]}$$

6.3 Functional model of the situation enrichment and adaptation system

In this section, we propose ontology-based dynamic context-aware recommendation system that enriches automatically profile's situation rules in different domains (shopping, work, travel, etc.). The objective of the situation enrichment system is to provide suitably adapted rules to users. These rules have to fit with users' preferences, users' current situations and device's capabilities. Figure 6.3 exhibits an overview of the functional model of the enrichment system. This system is described in two main processes including respectively the learning process and the recommendation process. The learning process classifies situations rules according to their content after applying pre-processing and filtering techniques. It is based

on four main phases, 1) rules profiles collection, 2) pre-processing & filtering rules, 3) rules analysis & classification, and 4) situation rule storage in rules metadata repository. The recommendation process calculates the final score and recommends new rule items for the user. It is based on five main phases, 1) calculate the similarity of rules items, 2) calculate the probability-based context of rules items, 3) calculate the final score and sorting rules, 4) select compatible rules using SWRL, and 5) recommend new rule items. In the next section, we will highlight each part's different steps of the recommendation system.

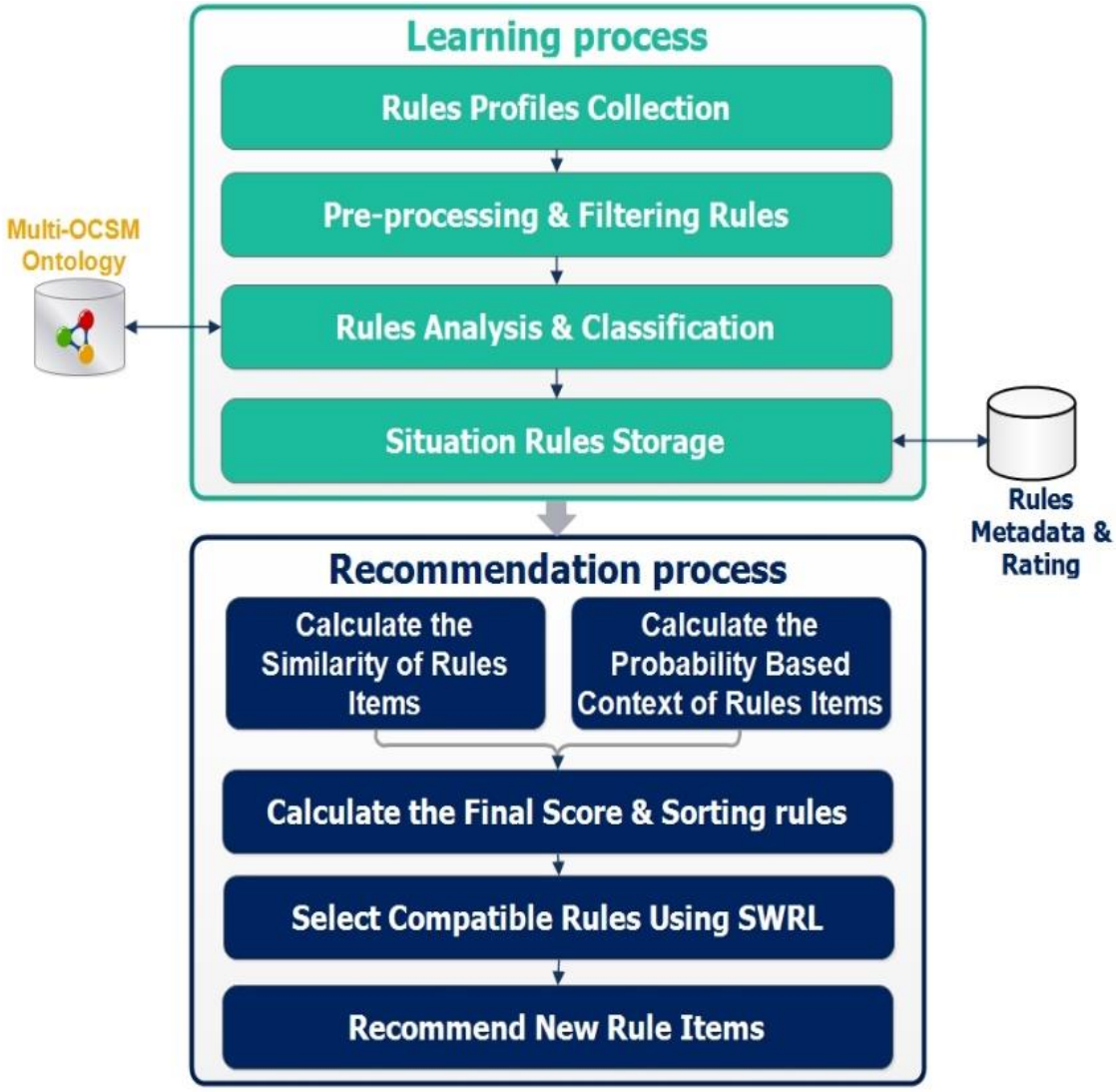


Figure 6.3: Proposed General architecture implemented using Multi-OCSM ontology.

6.3.1 Situation rules learning process

The proposed learning process classifies situation rules according to semantic rule ontology. The learning process takes a collection of situations rules and outputs classes of learned rules. It consists of the following four main steps (figure 6.4):

- **Step.1 – Collection of rules profiles.** In this step, we collect situation rules from users' agendas in which these situations have defined by other users or developers. Further, users can rate collected situation rules in order to provide a personalized context-aware recommendation by calculating their similarity scores and injecting new rules. After the collection of new situations, system sends them to the preprocessing module for filtering and prepossessing.
- **Step.2 – Pre-processing and filtering of rules profiles.** This step consists of eliminating redundant and insignificant rules (e.g. rule without consequence, rule without antecedent, etc.).

  Step.3 – Analysis and classification of rules profiles. We propose to analyze and classify the candidate rules with three context aspects (*location, time, and role*). From situation rules, we extract first a list of context data that explain the meaning of user's role, user's location or both and their synonyms from WordNet [117]. We define three classes that represent generic context rules according to the location and the role of user, namely, role & location-based, location-based, and role-based. A given situation rule is classified into one of these generic classes based on the location of user (Home, University, Office, Hospital, Car, City, Parking, Market) and his role (Citizen, Student, Worker, Driver). If the condition of rule contains both location and role terms then we classify it into role & location-based class. If not, we check if it contains only location terms, so we classify it into location-based class. If not, we classify it into role-based class. Then, we determine the most popular situation rules based on the users' ratings. The user's rating is ranging from 1 (useless rule) to 5 (excellent rule). Each user can give his opinion about a given situation rule. Situations with high popularity that match the user's current location and role are going to be used for the recommendation. The popularity of a situation rule is calculated using the following formula:

$$PS_i = \frac{\sum_{j=1}^{N} Rating\_S_i^{U_j}}{N} \quad (1)$$

  Where:

  PS_i: the popularity of situation i.

  $Rating\_S_i^{U_j}$: the rating of user j for situation i.

  N: the number of users who have rated the situation i.
- **Step.4 – Storage of situation rules**. Each situation rule is stored comprising the identifier and the search context-term.

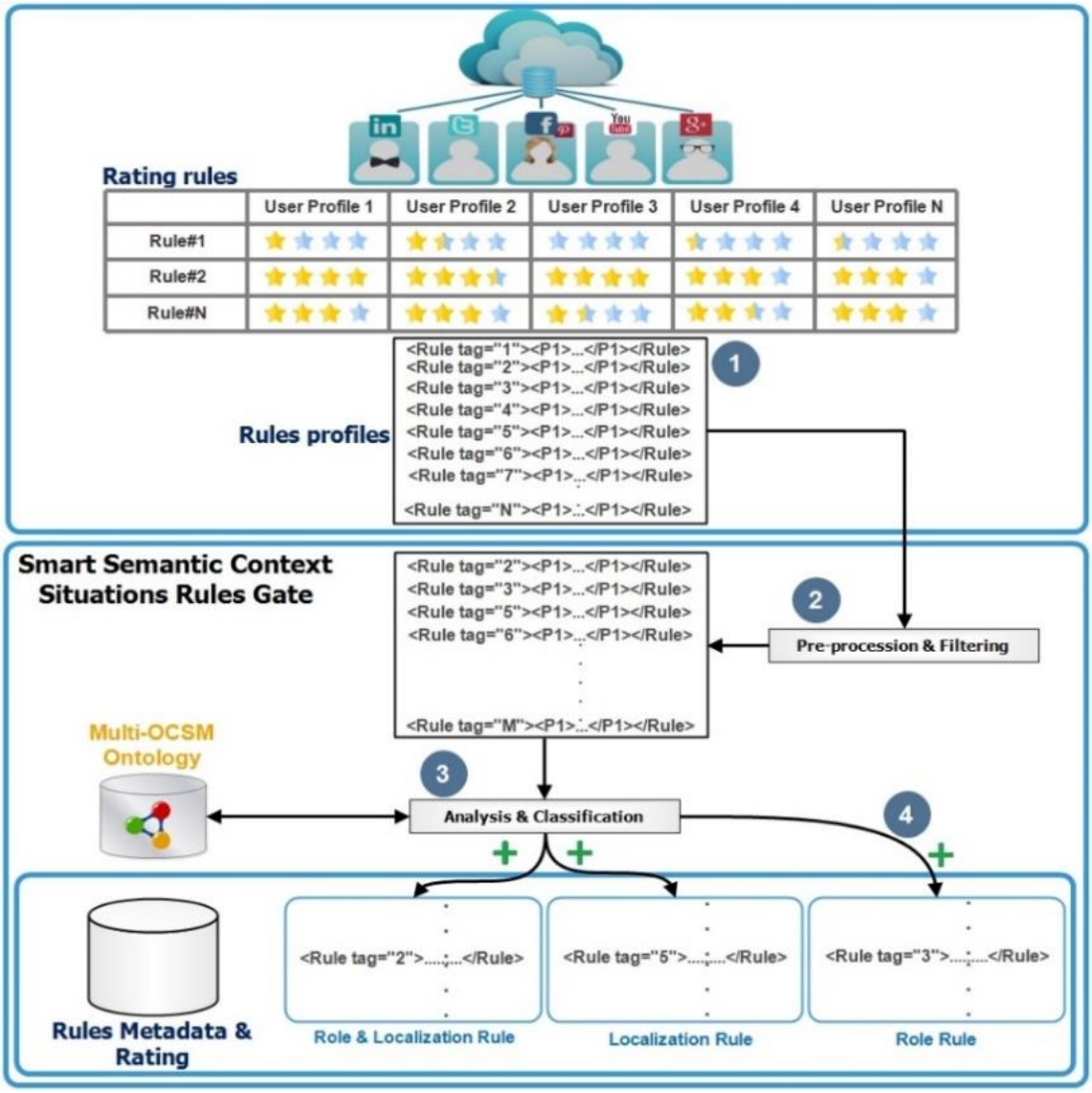


Figure 6.4: An overview of situation rules learning process.

6.3.2 Recommendation process of situation rules

This section presents a hybrid recommendation strategy for situation enrichment and adaptation based on a weighted linear combination equation. This latter, is a combination of two approaches, the first consist of rule's content-based semantic similarity, while the second is a Bayesian-classifier. The objective of the recommendation process is to provide suitable rules to users. It takes the user's preferences, situational context and device information as input to calculate the situation rules' scores and returns the relevant rules ranked according to their correspondence final scores. The proposed recommendation process operates in the following six main steps (see figure 6.5):

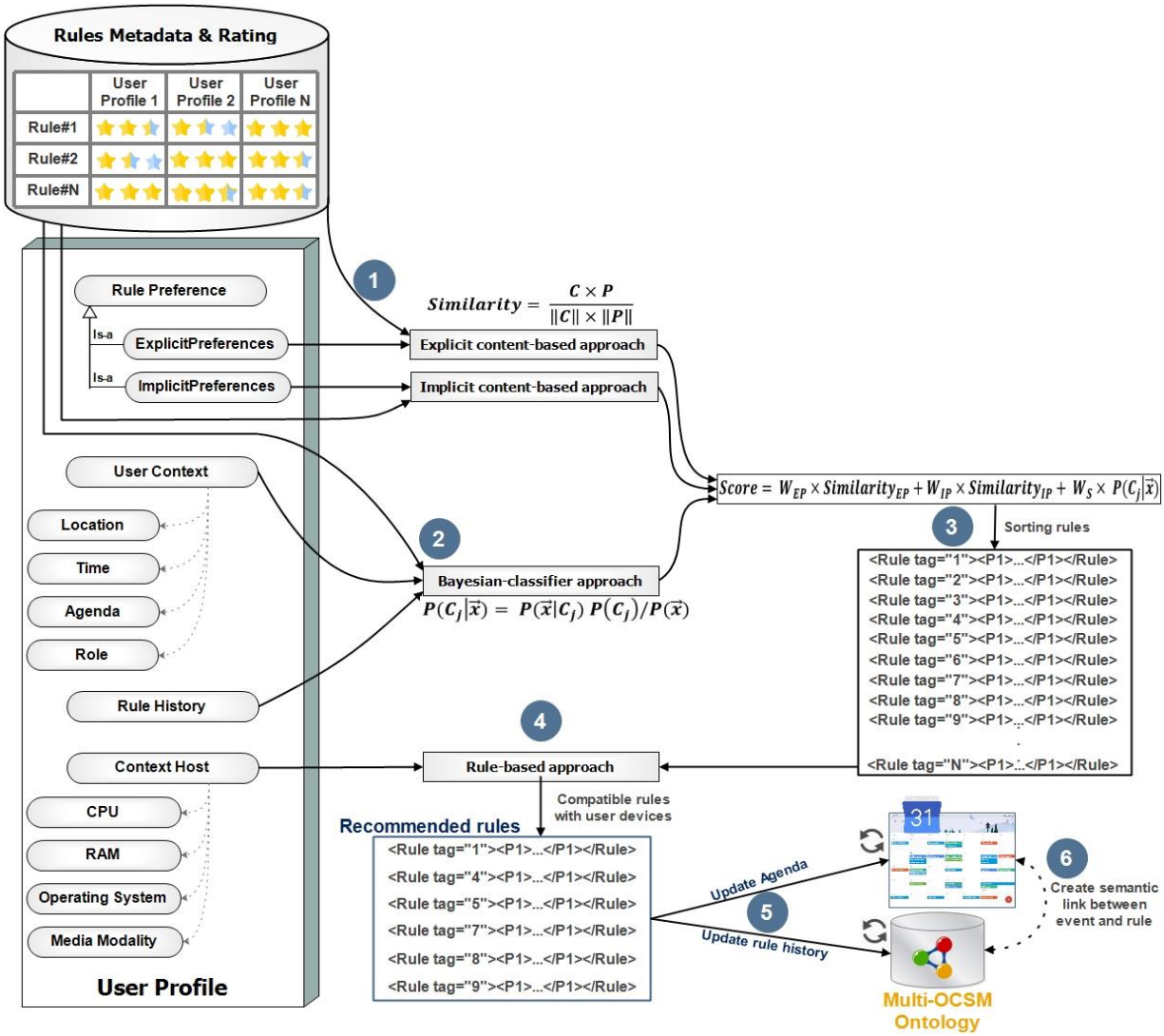


Figure 6.5: Hybrid rule-based recommendation process.

- **Step.1 – Calculation of rule semantic similarity based on content.** The first step consists of taking all the classified rules in order to calculate the similarity between rule items and user's preferences. It matches rule item content with the user's preferences-terms (explicit and implicit preference).
- **Step.2 – Calculation of rule probability based on situational context.** To deal with user situation context, we use the classical Bayesian classifier approach to evaluate rule items against the situation context. Thus, in the second phase, we calculate the probability for recommending the rule item knowing that we take into account the user situation context.
- **Step.3 – Calculation of rule final score and sorting rules.** In the third step of recommendation, we calculate the final score of each rule item including three aspects (i.e., explicit/implicit preferences and Bayesian classifier). Then, we use the final score to sort the rule items ordered from the highest to the lowest score.

- **Step.4 – Selection of compatible rules based on the device's capability.** The content-based approach and probability-based Bayesian classifier described above do not take into account if the available devices have the ability to check and use the recommended rule. In fact, the recommended item has to fit with devices capabilities and hardware compatibilities, for example, CPU speed, sensor availability, RAM size of the current devices. Moreover, the recommended item has a feature file, which describes several services and recommended devices often used to check and execute this rule item. We relied on rule-based approach to evaluate rule items against capability context to determine which rules can be used and executed by user domain, using SWRL language. The SWRL rule infers the situation rules which have the requirements (e.g., CPU speed) to run correctly in user domain and finally add them into the user agenda.
- **Step 5. Recommendation of new rules.** In this step, we take recommended rules with the highest scores and update both the ontology model and the user's agenda basing on user's needs.
- **Step 6. Agenda and ontology update.** In the sixth step, we create a semantic link between the user's agenda and situation rules stored in the ontology model. When the user's situation is evolved, the recommended list is changed accordingly.

6.3.2.1 Rule's content-based approach

The proposed approach deals with situation rule's content to evaluate rule items against user context. It compares rule item content with terms that characterize user preferences (explicit and implicit preference) to determine either the user likes or not that item. Only rule items with a high degree of similarity will be recommended. User's preferences can be extracted implicitly from his rules content or can be specified explicitly by the user. User can define many configurable preferences (i.e., Which domain he is interested in? What day? Which field? etc.). In addition, the user can specify the range of rules for the recommendation in terms of rules' popularity (i.e., low, medium and high popularity).

We define the user explicit preference as a vector $EP = (CP_1 \dots CP_n)$, where CP_i is configuration preference defined by the user. Each CP contains four features: domain of application (e.g. home, hospital, work, university, car, city, parking, market …etc.), days of rule's verification (e.g. weekday, weekend, etc.), time (e.g. morning, afternoon, evening …etc.) and the field of rules (e.g. study, demotics, security, shopping, work …etc.). From the vector EP we can extract a vector of terms (see Algorithm 1), where $EP\_Term = (t_1 \dots t_m)$, then we calculate the vector $W = (w_1, \dots, w_m)$, where w_i is the weight of term t_i. Given EP, w_i is calculated according to the equation (2):

$$w_i = \frac{number\ of\ occurrence\ of\ term\ t_i\ in\ EP}{e^{|EP|}} \quad (2)$$

Algorithm 1: Extract terms from Explicit Preferences (EP).

Input: Explicit preference vector $EP = (CP_1, \ldots, CP_n)$;
Output: Term vector $EP\_Term = \emptyset$;
Begin
1 for each element CP_i from EP
2 for each term t_j from CP_i
3 if ($t_i \notin EP\_Term$)
4 $EP\_Term = EP\_Term \cup t_j$;
End;

Similarly, we define Rules Repository (RR) for each user as a vector $RR = (r_1, \ldots, r_n)$, where r_i is a rule's metadata. Also, we define user implicit preference as a vector $IP\_Term = (t_1, \ldots, t_m)$, where t_i is a term extracted automatically from RR (see Algorithm 2). For each term t_i we assign a weight w_i where a weight vector can be also represented as $W = (w_1, \ldots, w_m)$. The weight w_i is computed according to the equation (3):

$$w_i = \frac{number\ of\ occurrence\ of\ term\ t_i\ in\ RR}{e^{|RR|}} \quad (3)$$

Algorithm 2: Extract terms from Rules Repository (RR).

Input: Rules repository vector $RR = (r_1, \ldots, r_n)$;
Output: Implicit preference vector $IP\_Term = \emptyset$;
Begin
1 for each element r_i from RR
2 for each term t_j from r_i
3 if ($t_i \notin IP\_Term$)
4 $IP\_Term = IP\_Term \cup t_j$;
End;

Finally, we can get two vectors of preference pair $PP = ((term_1, weight_1) \ldots (term_m, weight_m))$, for each explicit and implicit preference.

Using PP vector, we define the user preference as a vector $P = (w_1 \ldots w_m)$, where w_i is the weight of term t_i. Similarly, a rule item can be also represented as a vector with m elements, $C = (u_1, \ldots, u_m)$, where u_i is the weight assigned to term t_i, which is the same as that in the user preference vector P. It's clearly that terms aren't equally important (i.e., by their nature, terms in the location or field tag might be more important than subfield tag), so we assign importance factors to give each term's relative importance for rule item metadata.

We define tag set, $S = \{Location, Role, field, sub\ field\}$, and importance factor set, $W = \{W_x \mid x \in S\}$. W_x is the importance factor weight assigned to terms in tag x to give their relative importance (assuming $W_{location} = 0.75$, $W_{role} = 0.45$, $W_{field} = 0.75$, $W_{subField} = 0.35$). The vector C is generating using the algorithm 3 below (e.g. see figure 6.7):

Algorithm 3: Generating the vector C

Inputs: Rule item's metadata *R* (item to recommend);
The vector *PP*;
Output: The vector *C*;
Begin

```
for each term value t_i from PP
    if (t_i is merely included in tag x of R)
        u_i = W_x;
    if (t_i is included in two or more tags of R)
        u_i = Max{W_x}
    if (t_i isn't included in any tag of R)
        u_i = 0;
```

End;

Finally, we calculate the similarity between the two vectors of rule item and user preference. We use the commonly used similarity metric (i.e., the cosine value of the angle between the rule item and user preference). The cosine similarity is calculated as:

$$Similarity = \frac{C \times P}{\|C\| \times \|P\|} = \frac{\sum_{i=1}^{m} u_i w_i}{\sqrt{\sum_{i=1}^{m} u_i^2 \sum_{i=1}^{m} w_i^2}} \quad (4)$$

6.3.2.2 Bayesian-classifier approach

Content-based approach handles only with rule item content and user preference. To deal with user situation context we use the classical Bayesian classifier approach to evaluate rule items against the situation context. In our multidimensional recommendation model, we can group each situation context into classes (e.g., the situation (work, weekday and worker) can be represented as a class). Therefore, we have three dimensions that we can divide into classes.

For example, we can divide a user's location into four classes (home, university, office, outdoors: in a car for example), and divide day rule into two classes (Weekday, Weekend), also we divide a user's role into four classes (Citizen, Student, Worker, and Driver). Finally, we get the cartesian product for the 3 dimensions, $location \times day \times role$, *as a result, 32 classes.*

Suppose $C = \{C_1 \dots C_{32}\}$ are classes of a considered situation. In this approach, we evaluate the probability of a rule item belonging to a certain class of *C*. The general form of a user rules repository is $< location|day|role||rule\ metadata >$, where *location* and *day* determine where and when the rule was checked, respectively; *role* determine the role that the user has to play; and *rule metadata* show the other tags (field, subfield, etc.). For example, if we considered the class C_i = *(home, weekend, citizen),* then we can calculate the probability of rule situation item $\vec{x}$ belonging to this class C_i through mathematical repository analysis of the user rules using the equation below:

$$P(C_j|\vec{x}) = P(\vec{x}|C_j)\, P(C_j)/P(\vec{x}) \quad (5)$$

$$P(\vec{x}\,|C_j) = \prod_{i=1}^{4} P(f_i|C_j)$$

$$P(C_j) = k(C_j) \,/\, T$$

$$P(\vec{x}) = \sum_{j=1}^{k} P(\vec{x}|C_j)P(C_j)$$

$$P(f_i|C_j) = \frac{n(f_i, C_j) + 0.5}{n(C_j) + 0.5|V_j|}$$

Where :

- f_i denotes the *ith* tag *of* $\vec{x}$,
- $k(C_j)$ denotes the number of user rules in C_j,
- T denotes the total rules number in user repository,
- $n(f_i, C_j)$ denotes how many times f_i occurs in C_j,
- $n(C_j)$ denotes the sum of times all tags occur in C_j,
- $|V_i|$ denotes the total number of tags occurring in C_j.

For example, we can calculate the probability that a user will be interesting to add such rule item (house security's field) in his repository outdoors on a weekday (here the role isn't important in such situation, so we can base only on the two other dimensions).

At last, the final score of the rule item will be measured using both content-based approach and Bayesian-classifier approach. The final score is given by the weighted linear combination equation;

$$Score = W_{EP} \times Sim_{EP} + W_{IP} \times Sim_{IP} + W_S \times P(C_j|\vec{x}) \quad (6)$$

Where W_{EP}, W_{IP}, W_S are weight assigned to explicit similarity, implicit similarity and situation probability respectively ($W_{EP} + W_{IP} + W_S = 1;\ 0 \le W_{EP} \le 1;\ 0 \le W_{IP} \le 1; and\ 0 \le W_S \le 1$) given the relative importance of explicit and implicit preference and situation context.

Rule items will be ranked according to the scores achieved through the equation described above. Then, we use the rule-based approach to select which rule items are able to check and execute through the user domain (devices, sensors, actuators, etc).

6.3.2.3 *Rule-based adaptation approach*

Both recommendations described above, do not consider if the user domain (available devices) has the ability to check and use the recommended rule. Moreover, the recommended item has a feature file that describes the services and recommended devices to use to check and execute this rule item, for example,

CPU speed, sensor availability, RAM size and so forth should satisfy the capability context. We relied on rule-based approach to evaluate rule items against capability context. In this scope, after the generation of recommendation list of rule items, we have to determine which rules can be used and executed by user domain, using SWRL language (see Table 1). Our system applies an inference engine that is responsible to infer pertinent rules for a given domain situation. For example, the SWRL rule in Table 1 infers the situation rules which have the requirement (e.g., CPU speed) to run correctly on user domain and then add them to the user recommended agenda.

Table 6.1: SWRL rule that infers appropriate situation rules based on the terminal capabilities.

UserProfile (?user) ∧ userDomain (?user, ?device) ∧ hasCPU (?device, DeviceCPUSpeed) ∧ hasRAM (?device, DeviceRAMSize) ∧ network (?device, DeviceNetwork) ∧ SituationRule (?rule, "House_Alarm") ∧ ruleHasCPURrequirement (?rule, RuleCPUSpeed) ∧ ruleHasRAMRrequirement (?rule, RuleRAMSize) ∧ ruleHasNetworkRrequirement(?rule, RuleNetworkSpeed) ∧ greaterThan (DeviceCPUSpeed, RuleCPUSpeed) ∧ greaterThan (DeviceRAMSize, RuleRAMSize) ∧ greaterThan (DeviceNetwork, RuleNetworkSpeed) → agenda(?user, ?rule)

6.4 Illustrative case study

In this section, we illustrate our approach with an illustrative use case. As an example, we clarify the whole process of ontology-based context-aware recommendation. First of all, after the user sets his explicit preference (see Figure 6.6), we extract the *PP* vector (preference pair) for explicit preference as we show above (Also, the *PP* vector for implicit preference is extracted as we have shown previously).

Explicit preference

Configuration#1	Configuration#2	...	Configuration#N
<Domain>Home</Domain> <Day>Weekday</Day> <Time>Evening</Time> <Field>Study</Field>	<Domain>Home</Domain> <Day>Always</Day> <Time>Always</Time> <Field>Domotics</Field>		<Domain>Home</Domain> <Day>Always</Day> <Time>Always</Time> <Field>Security</Field>

Figure 6.6: An example of explicit preference.

Figure 6.7 illustrates the process of feature extraction and similarity measurement (we can use it for both implicit and explicit preference), we assume W a set of weight ($W_{location} = 0.75$, $W_{role} = 0.45$, $W_{field} = 0.75$, $W_{subField} = 0.35$). The rule item represents a house alarm situation rule that happens when the user is outside his home with 20 meters tolerance value and it has two fields value (Security and Domotics). The similarity calculated for explicit preference is 0.88 (same as explicit preference, the similarity calculated for implicit preference is 0.71).

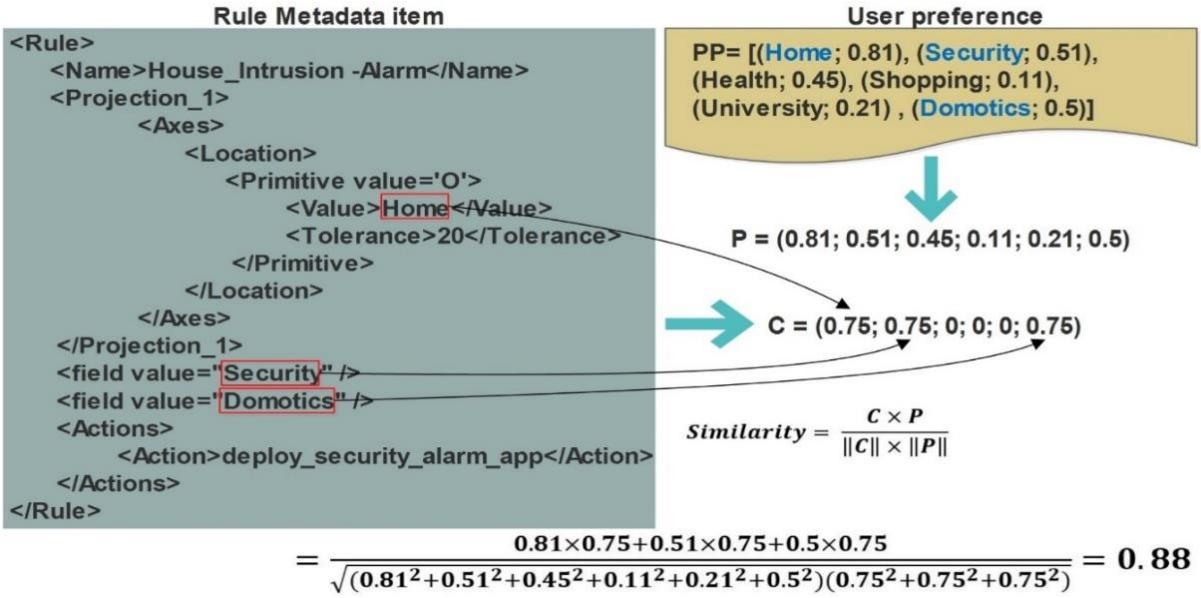


Figure 6.7: An example of feature extraction and similarity measurement.

After the similarity measurement, we can estimate the probability that a user will use the rule item "*House_Intrusion_Alarm*" outdoors on a weekday. The probability will calculate using the Bayesian classifier approach as we show above, where the class C_j = *(Outdoors, Weekday, Citizen).* Figure 6.8 shows an example of user agenda history. According to the Bayesian equations we have presented above:

$P(C_j \mid \text{"House\_Intrusion\_Alarm"}) = P(\text{"House\_Intrusion\_Alarm"} \mid C_j)\, P(C_j) \,/\, P(\text{"House\_Intrusion\_Alarm"}) = 0.48$

```
<Outdoors|Weekday|Citizen||House_Surveillance_Camera|Location(O(Home))|Field(Security, Domotics)>
<Outdoors|Weekday|Citizen||House_Surveillance_Fire|Location(O(Home))|Field(Security)>
<Office|Weekday|Worker||Meeting|Location(I(MeetingRoom))|Time(PT(meeting))|Field(Meeting)>
<Outdoors|Weekend|Citizen||Shopping|Location(I(Supermarket))|Time(PT(Shopping))|Field(Shopping)>
```

Figure 6.8: An example of user agenda history.

After that, the final score of the rule item (*"House_ Intrusion_ Alarm"*) will be measured using the weighted linear combination equation (assuming, $W_{EP} = 0.35$, $W_{IP} = 0.35$, $W_S = 0.30$). The final recommendation score is:

$$Score = 0.35 \times O.88 + 0.35 \times 0.71 + 0.30 \times 0.48 = 0.7$$

Finally, we rank all rule items according to the scores reached through these three steps and choose the rule items which have the highest score. Next, we use the rule-based approach to select rule items that can be used and executed by user domain, using SWRL language.

6.5 Conclusion

In this chapter, we have presented an ontology-based context-aware approach for user's situation rules classification and recommendation for adaptable context-aware mobile application. Our approach is mainly consisting on two processes, the learning process and the recommendation process. The former

can be summarized in the semantic classification of user's situations rules. While the latter is based on the dynamic injection of situations rules at run-time through the hybridization of the content-based approach, the Bayesian classifier approach, and the ontology-based SWRL rules. The recommendation process leverages the benefit of explicit and implicit user's preferences rules, user's roles and Bayesian network to increase significantly performances for relevant situation rules recommendation. Besides, the use of ontology model on learning and recommendation processes facilitates the filtering and customization of situation rules. The approach has been demonstrated through a simple case study to simulate its overall execution flow.

The next chapter is dedicated to present the implementation and validation of the proposed semantic context-aware rule-based composite situation management approach.

Chapter 7

Prototype and Evaluation

Summary

7.1 Introduction

The huge number and diversity of sensors in pervasive environment lead to high complexity in terms of interpreting: huge data volumes, parallel incoming events, different modalities, inter-dependence and real-time update. Moreover, in real world, sensors produce parallel data that my result in sensing of user's activities or environment state. Consequently, the execution of pervasive applications leads to monitor parallel incoming event in order to interpret different users' behaviors. Many limitations in pervasive environments as incomplete, redundant and contradictory context data that may produce uncertainty issue. Traditional context monitoring systems monitor parallel incoming event sequentially. This may slow down the identification process of user's composite parallel situations, which decrease the performance of systems. Our challenge is to provide a system that allows monitoring parallel incoming events and parallelly processing different users' situations.

This chapter presents the implementation and validation of the proposed semantic context-aware rule-based composite situation management approach in the context of distributed smart Environments. We developed a new framework with agile situations management process and parallel distributed microservices-based tasks that reduce the complexity of user's situations management and provides quick response time. The experiments were conducted by generating a specific number of situations and various types of attribute-based sensors events to demonstrate the efficiency and the effectiveness of the presented approach. We showed the interest of the proposed solution basing on two different approaches, sequential

approach and parallel approach. The sequential approach had only a single task that ran in a single processor to check all the users' situations. In contrast, the parallel approach had tasks as much as the number of user's situations that simultaneously ran at the same time in different processors' cores (multi-core processor).

In this chapter, we present in section 7.2 a prototype implementation of parallel identification process for composite situations. Then, we present in section 7.3 the different experimental evaluation of our prototype, where we define the used parameters and evaluated metrics, as well as results validation. Finally, section 7.4 concludes this chapter.

7.2 Prototype implementation

Our prototype is implemented with *IntelliJ IDEA 17.2*. The agile process of situation's identification is the main part of our framework. The prototype is able to manage and identify composite situations under different user context changes. The context changes are frequently captured and sensed (every 20 milliseconds) by different sensors. It allows as much capturing of incoming events as possible in a parallel way in order to process them in a parallel way, which improves the performance system and reduces the response time. In addition, it can frequently filter user situations (using SPARQL queries as shown in Table 4.3 Chapter 4) according to two parameters (current user's location and time) trying to submit only pertinent users' situations that are executed at a given location and/or time and thus enhancing the situation identification process. After the filtering process, the selected situations may include redundant events, as well as redundant situations. Before sending them to the Executor, they have to be factorized to eliminate all the redundant events and situations. After that, the factorized events and situations will be encapsulated in the *Runnable* microservices classes in order to send them to the *Executor* to execute them in different threads. The prototype manages the life cycle of a situation as a thread, according to user's spatiotemporal values by dynamically creating/updating/killing this thread.

Our framework is smart enough to identify any abnormal situations and prioritizes the processing of urgent situations. In case of urgent situations like health-heart-situations, the expert (i.e., a doctor in the case of smart health domain) set situations priorities based on the user's context condition (i.e., patient's health and disease's type). The system autonomously calls the urgent event processing. The Event Listener component monitors the user's events and sends them to the appropriate Situation Identifier component. When the situation is identified, the Service Deployer component searches for the closest available services and requests the nearest service for immediate management of identified user's situation.

The main objective of our Java-based framework is to automatically manage composite user situations in different smart environments (smart-home, smart-university, smart-car, smart-hospital, etc.) in an efficient way using parallel paradigm in order to improve the situation identification process. The

situation identification process is the core part of our framework, which is responsible for receiving and aggregating context data from the smart object layer and then send them to each specific situation as required. The Multi-OCSM offers an automatic matching between the events' context data and their specified situation rules using the semantic links between them.

7.3 Experimental evaluation

For evaluation purposes, we included two generator modules. The first one is the situation generator module that creates a specified number of situations in order to simulate the identification process of situations. The number of situations is varied in each execution; thus, we can simulate the scalability experiment for a different number of situations. Moreover, we have observed and calculated different metrics such as (execution time, expired/unexpired events, missed/unmissed situations, a number of worker threads), to evaluate the performance of the prototype. The second one is the dataset generator module, which is responsible for generating various types of sensor events. The event's valid period is set to a specific lifetime. During the verification process, if there is an event that exceeds its lifetime, it will be marked as an expired event.

7.3.1 Experimental setup

To evaluate the performance of parallel and sequential approach with their processing algorithms, we conducted many experimental evaluations by means of simulations. All simulations have been performed on a PC running Windows 10 (64 bits) on an Intel processor core i5-2430 2.4 GHz, and 8 GB RAM.

With simulation, we included two generator modules: the situation generator and the dataset generator. The situation generator module that creates a specified number of situations permits the identification process of situations to be simulated. We have simulated both approaches with different number of situations that gradually increased after each run. Thus, we can simulate the scalability experiment for a large number of situations. The dataset generator module is responsible for generating various types of sensor events (states of doors, lights, home temperature, user location, user glucose level, user movement states), resulting in 50,000 sensor events.

The performance of a parallel and sequential approach can be measured using different metrics such as an expired event ratio, missed situation ratio, number of submitted microservice and execution time.

7.3.2 Evaluation parameters and metrics

In order to evaluate the efficiency and effectiveness of the proposed approach, several parameters and metrics have been adopted. We have defined four parameters including the sensor's frequency time, the event's appearance time, the event's validity period and the event's expired time. The sensor's frequency-time defines the frequency-time of sending the sensing data to the system. The event's appearance time ($t_{capture}$) is the time when the event is captured by the sensor. The event's validity period is the interval

time between the event's appearance time ($t_{capture}$) and the event's expired time ($t_{expired}$). The event's expired time ($t_{expired}$) is its expiration time (see figure 7.1) after the validity period has expired.

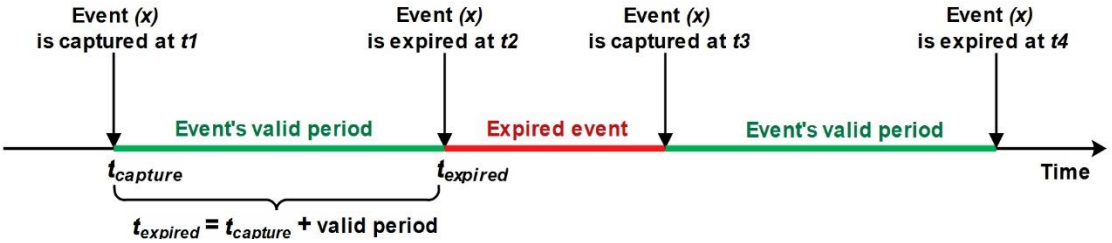


Figure 7.1: Event lifetime.

To evaluate the performance of parallel and sequential approaches, we have defined five evaluation metrics during the situation's verification process, including expired event ratio, missed situation ratio, number of situations' verifications, number of submitted threads, and execution time. Expired event ratio and missed situation ratio are used to assess the approach effectiveness and the quality of the proposed situation identification process. The number of situations' verifications and number of submitted threads are used to measure the scalability of the proposed approach. The execution time is used to assess the system performance in static and moving scenarios.

- **Expired event ratio:** the ratio of the number of expired events to the total amount of the events:

$$Expired\ Event\ Ratio = \frac{ExpiredEvent}{ExpiredEvent + UnexpiredEvent} \quad (1)$$

  Where,

  - ***Expired event*** is the total number of events that have exceeded their lifetime, where each event that exceeds its lifetime is marked as an expired event.
  - ***Unexpired* event** is the total number of events that have not exceeded their lifetime, where each event that does not exceed its lifetime is marked as an unexpired event.

- **Missed situation ratio:** the ratio of the number of missed situations to the amount of total missed and unmissed situations:

$$Missed\ Situation\ Ration = \frac{MissedSituation}{MissedSituation + UnmissedSituation} \quad (2)$$

  Where,

  - ***Missed situation*:** the total number of situations that have at least an expired event; each situation has at least an expired event is marked as a missed situation.
  - ***Unmissed situation*:** the total number of all situations that do not have any expired events: each situation that does not have any expired event is marked as unmissed situation.

- **Number of situations' verifications:** the total number of verifications of all situations. Each time a situation receives a required event, it starts its verification, then incrementing the number of situations' verifications.

$$Nb_{situations\_verifications} = \sum_{i=1}^{n} Nb_{verification}(S_i) \quad (3)$$

Where,

  - $\boldsymbol{Nb_{verification}(S_i)}$**:** number of verifications of the situation *i*.

- **Number of submitted threads (micro-services):** the sum of submitted events and situations as threads executed by different cores of the processor.

$$Nb_{microservices} = Nb_{submitted\_events} + Nb_{submitted\_situations} \quad (4)$$

Where,

  - $\boldsymbol{Nb_{microservices}}$**:** total number of microservices.
  - $\boldsymbol{Nb_{submitted\_events}}$**:** total number of submitted events.
  - $\boldsymbol{Nb_{submitted\_situations}}$**:** total number of submitted situations.

- **Execution time:** the time needed to accomplish the verification of a specific number of situations' verifications.

$$T_{exec} = \sum_{i=1}^{n_1} T_{filt}(i) + \sum_{j=1}^{n_2} T_{sensing}\big(E(j)\big) + \sum_{k=1}^{n_3} T_{verif}\big(S(k)\big) + \sum_{z=1}^{n_4} T_{overhead}(z) \quad (5)$$

Where,

  - $\boldsymbol{T_{exec}}$**:** the execution time of all the processes.
  - $\boldsymbol{T_{filt}(i)}$**:** the execution time of the filter *i*.
  - $\boldsymbol{T_{sensing}\big(E(j)\big)}$**:** the time needed for sensing the event *j*.
  - $\boldsymbol{T_{verif}\big(S(k)\big)}$**:** the time needed to verify the situation *k*.
  - $\boldsymbol{T_{overhead}(z)}$**:** the overhead time that can be spent in tasks management (i.e. synchronization, place new tasks into the queue and take queued tasks for execution)
  - n_1: number of filtering of the situations.
  - n_2: number of sensed events.
  - n_3: number of verified situations.
  - n_4: number of managed tasks.

Assume that several user's situations are intended to be analyzed and identified. Therefore, the main objective of an approach is to minimize the missed situations and expired events ratio for a specific period of time. We consider an approach to work well if it can verify as many situations in a very specific

time with a very low ratio of expired events and thus a very low ratio of missed situations. That is, the approach will be measured in four metrics, expired event ratio, missed situation ratio, number of submitted threads and execution time.

On the other hand, we intend to compare our proposed parallel framework vs pipelining-based framework [118]. Therefore, two metrics have been used to evaluate and compare performance between these frameworks, namely, event detection latency and throughput.

- **Event detection latency:** the time delay between event acquisition and event detection.

$$latency = \frac{\sum_{i=1}^{n}(T_{detection_i} - T_{acquisition_i})}{n} \quad (6)$$

Where,

- $\boldsymbol{T_{acquisition_i}}$**:** the acquisition time of event *i*.
- $\boldsymbol{T_{detection_i}}$**:** the detection time of event *i*.
- $\boldsymbol{n}$**:** number of detected events.

- **Throughput:** number of detected events per minute.

In the next sections, we evaluate and compare the performance of different proposed strategies including parallel and sequential approaches. Furthermore, we evaluate our parallel proposed approach vs pipelining-based approach [118].

7.3.3 Evaluation results and analysis

In this section, we show the evaluation of the proposed solution based on two different approaches, sequential and parallel. Parallel and sequential approaches have different strategies concerning user situations identification and event detection. The sequential approach uses only a single task that runs in a single processor to check all user's situations. In contrast, the parallel approach uses tasks as much as the number of user's situations that run simultaneously at the same time in different processor cores. The more cores a machine has, the more events detection and situations identification tasks can be performed simultaneously.

We evaluated and compared the performance of the implemented approaches by measuring four metrics: execution time, expired event ratio, missed situation ratio, a total of submitted microservices. The expired event ratio and missed situation ratio were crucial factors. The high ratio showed the poor performance of the approach and it created negative user experiences leading to the famous problem of oscillation. This problem concerning a user's mobility happened when the system took a significant delay to check a situation. The triggered actions were meaningless in some urgent cases because the user's context changes and this can have a negative impact on the system. To cope with this problem, we marked each event with an appearance time. Then we obtained the event's expired time by adding the

event's appearance time to the event's validity period and compared it with the current time. If the current time was after the event's expired time, then the event would be marked as an expired event. As a consequence, it causes a missed situation. In a cycle of situation verification, if a situation receives more than one event and there is more than one expired event, the number of missed situations would be incremented only by 1. Unlike the number of expired events, it would be incremented each time we had an expired event. When the situation was marked as missed, the actions would not be triggered and thus avoid meaningless triggered actions.

In the next sections, to validate the performance of the proposed implemented approaches, we present a comparison of four defined metrics on parallel and sequential approaches for identifying urgent composite situations. We used two kinds of scenarios: static and dynamic. For the static scenario, we focused on a specific user's location (i.e., user at home) in high workload user situations with a varied number of situations' verifications from 1,000 to 10,000. While for the dynamic scenario, we handled continuous context changes (i.e., the mobility of user) with dynamic filtering of situation rules according to each relevant context changes. In this case, we implemented two alternatives without/with the situation filtering process as a static number of situation rules and dynamic number of situation rules respectively. Thus, we can better observe the different effects that can be introduced using the filtering process. For all the evaluated experiments, the sensor's frequency-time it was set to 20 milliseconds, and the event's valid period was set to 50 milliseconds. During the situation verification process, if there was an event that exceeds its lifetime (i.e., 50 milliseconds), it will be marked as an expired event.

7.3.3.1 Static evaluation results and analysis

For the first scenario, a user has a specific location (i.e., at home). We want in this experiment to simulate a high workload case (i.e., a large number of situations and events) on sequential and parallel approaches. The goal is to minimize both the ratio of expired events and the ratio of missed situations, as well as the execution time.

a. Evaluation and comparison of Expired Event Ratio

We have evaluated the expired event ratio with different number of situation's verifications (1000 – 5000) under varied events set size (34 – 64). As shown in figure 7.2, we observe that the expired event ratio is very low, ranging from 0 to 0.073 that shows better performance of the parallel approach.

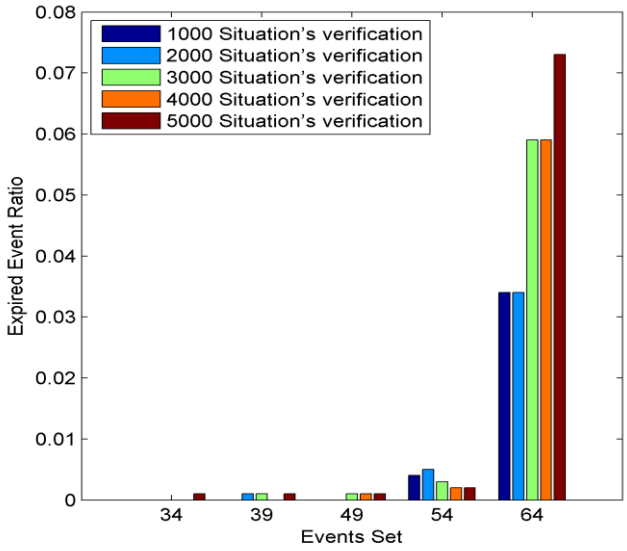


Figure 7.2: Expired event ratios under different events set in the parallel approach.

We compare the expired event ratios of parallel approach with the expired event ratios of the sequential approach. As shown in figure 7.3, we can notice that the obtained results of parallel are much better than using the sequential approach. Clearly, the parallel approach yields better-expired event ratios on all events set involved in the simulation. This caused by multiple events detection microservices deployed at the same time on the processor cores. Thus, a system can manage a large number of incoming events in a parallel way. However, the sequential approach uses only one single task running in a single processor to check all the incoming events and user's situations.

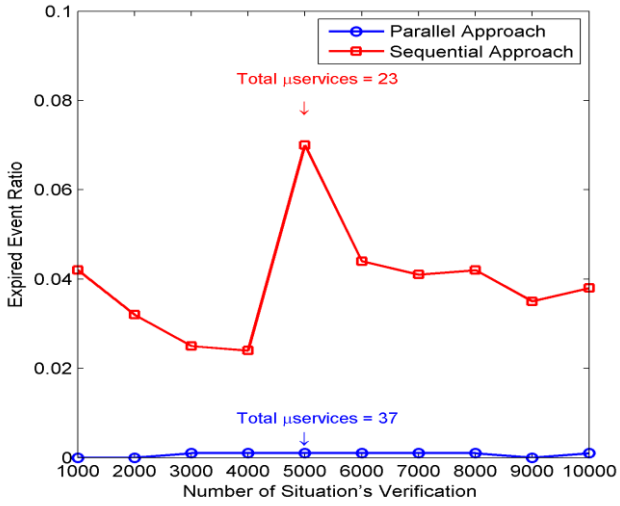


Figure 7.3: Expired event ratio comparison between the parallel and sequential approaches.

b. Evaluation and comparison of Missed Situation Ratio

The missed situation ratio is an important performance factor in pervasive environments, which reflects the ability of the system to check a large number of situations in a specific time. In order to evaluate the impact of different approaches on the missed situation ratio, we have evaluated the missed situation ratio

with a different number of situation's verifications (1000 – 5000) under varied situations rules (20 – 53). As shown in figure 7.4, we observe that the missed situation ratio is very low from 0 to 0.11, which shows better performance of the proposed parallel approach.

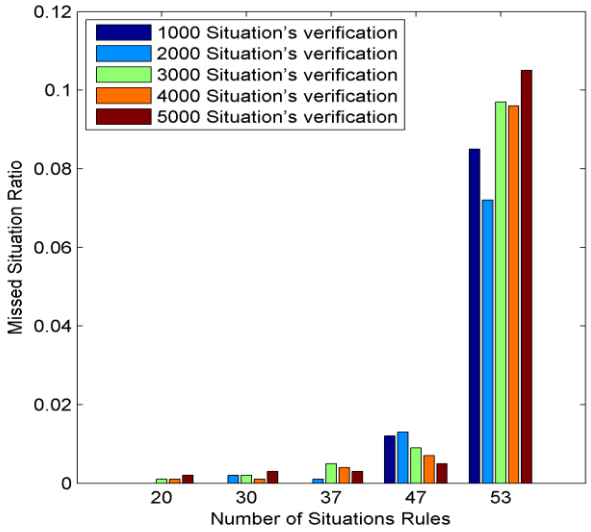


Figure 7.4: Missed situation ratios under different number of situations rules in the parallel approach.

The missed situation ratios of the parallel approach were compared to the missed situation ratios of the sequential approach. As shown in figure 7.5, the obtained results of the parallel approach are much better than the sequential approach. In 5000 situations verifications, the system with the parallel approach has better-missed situation ratios of 0.015 using 37 microservices while the system with sequential approach can achieve 0.073 using 23 microservices. Therefore, the parallel approach identifies various composite situations in several simultaneous microservices while the sequential approach adopts a single task for a situation identification.

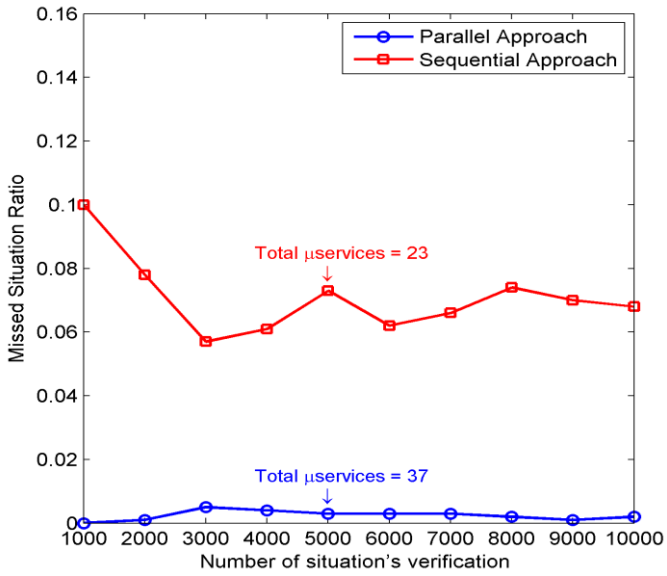


Figure 7.5: Missed situation ratio comparison between the parallel and sequential approaches.

c. *Evaluation and comparison of execution time*

We have evaluated the execution time with a different number of situations' verifications (1000 – 5000) under a varied number of microservices (20 – 53). As shown in figure 7.6, we observe that the execution time of the parallel approach is very satisfactory, ranging from 500ms to 4500ms.

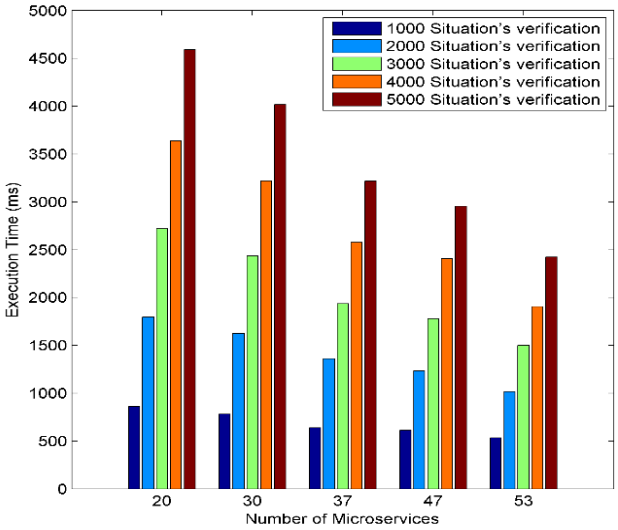


Figure 7.6: Execution time under different numbers of situation's verification in the parallel approach.

We compared the execution time of the parallel approach with the execution time of the sequential approach. As shown in figure 7.7, we can notice that the obtained results from 1000 till 3000 situations' verifications are nearly equal in both approaches. After the 3000th situations' verifications, the parallel approach became better than the sequential approach. The parallel approach yields lower execution time than the sequential approach due to the use of several situations and events microservices deployed simultaneously on different processor cores while the sequential approach considers only a single processing task inside the same processor.

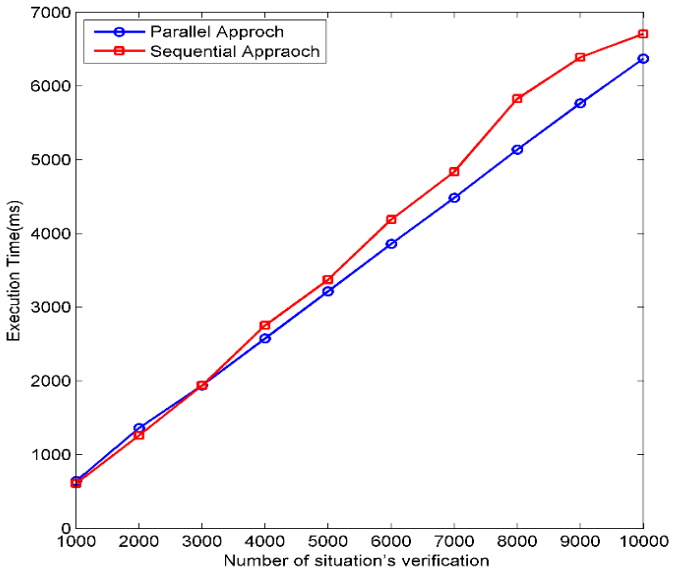


Figure 7.7: Execution time comparison between the parallel and sequential approaches.

d. Evaluation of Expired Event Ratio vs Number of microservices

Figure 7.8 shows the expired event ratio of the parallel approach versus the number of microservices. We set the number of situation's verification to 10000, and we increase the number of microservices from 20 to 53. Obviously, our parallel approach has very low expired event ratios between 0 and 0.13. The parallel approach has a high number of simultaneous processing microservices, which may increase the processing load and further lead to increase the expired event ratio. In addition, the parallel approach can rise the processing overload, so a high-performance machine is required to fully exploit available computing resources in order to handle a significant number of incoming parallel events.

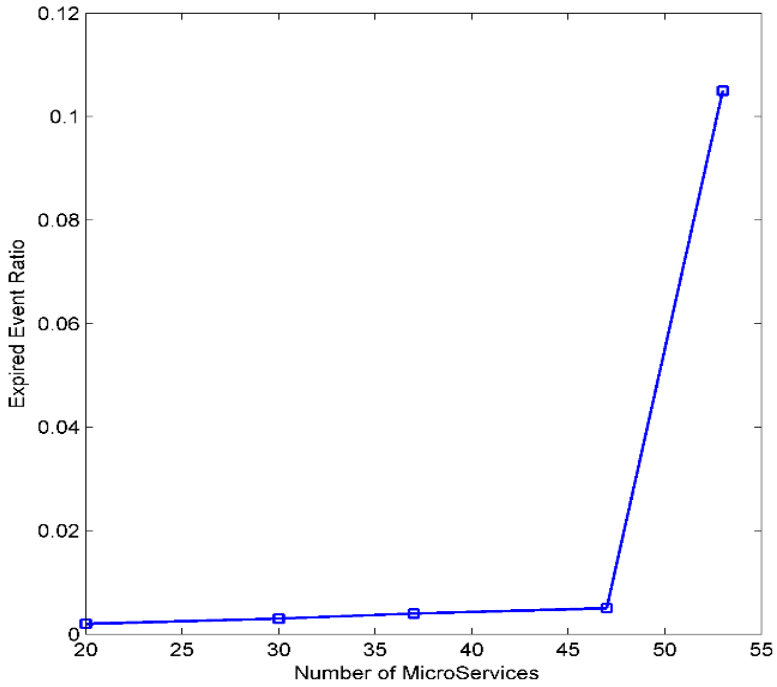


Figure 7.8: Number of micro-service vs expired event ratio in the parallel approach.

e. Evaluation of Missed Situation Ratio vs Number of microservices

Figure 7.9 illustrates the missed situations ratio of the parallel approach versus the number of microservices ranging from 20 to 53. The missed situation ratio for the parallel approach is considerably low in all the experiments. The parallel approach has a high number of simultaneous processing microservices, which may increase the processing load and further lead to increase the missed situation ratio. This ratio must be improved in future works using a high-performance machine.

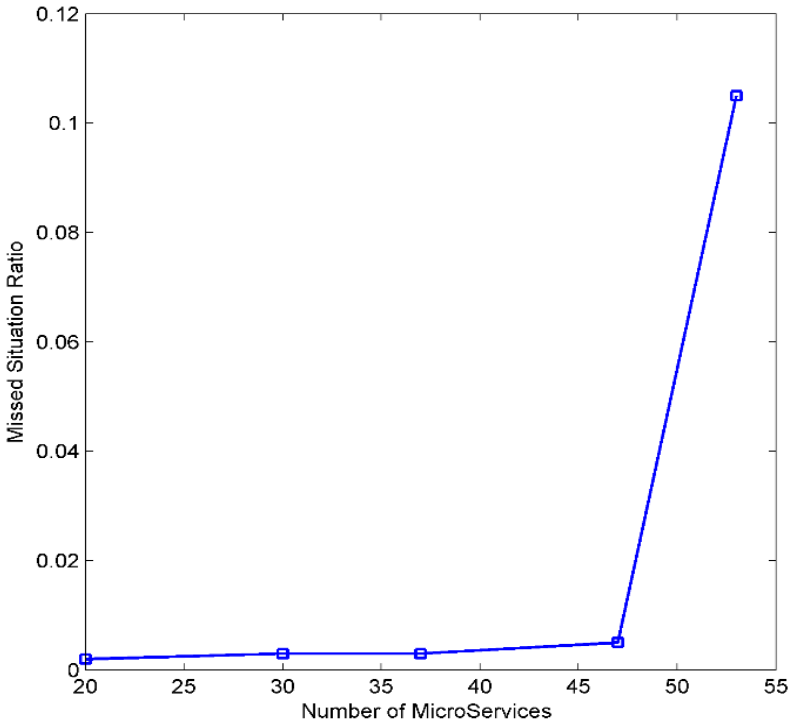


Figure 7.9: Number of micro-services vs missed situation ratio in the parallel approach.

7.3.3.2 Dynamic evaluation and analysis

Unlike the first evaluation scenarios approach, the dynamic evaluation scenario considers dynamic context changes (mobility of the user, various events for each smart-domain and various context sources). In order to verify and evaluate the proposed approach that we have conducted in this study to simulate user mobility, such that the user changes his location from one smart-domain to another to show the performance evolutions. Two experiments were conducted to demonstrate the efficiency of the proposed approach in terms of the number of expired events and the number of missed situations.

a. Expired Event vs. #Event's Microservices vs. # Situation's verification

Figure 7.10 shows the number of expired events of the parallel approach versus the numbers of events microservices ranging from 22 to 64 and the number of situation's verification ranging from 1000 to 8000. Obviously, when the number of situations verification increases with the number of events microservices, the number of expired events increases due to the limited number of processor cores and the time consumed by the filtering process. Therefore, the system will not be able to detect all occurred events.

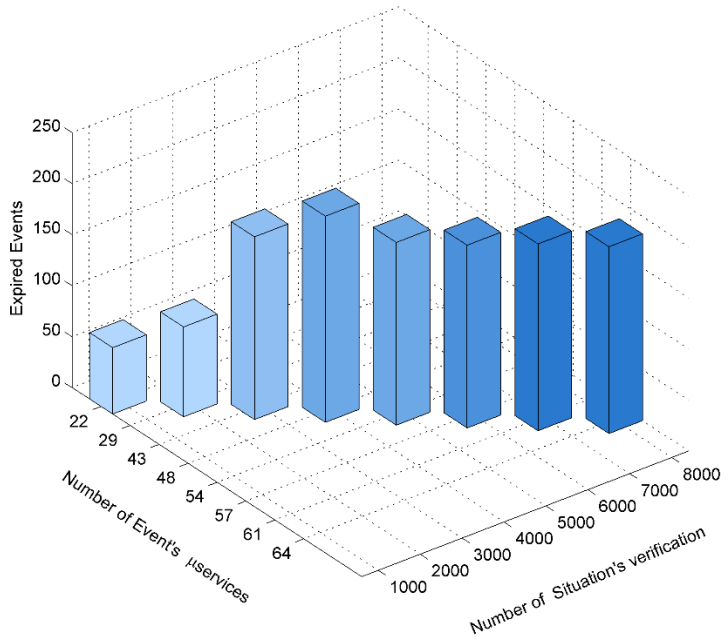


Figure 7.10: Evaluation of **e**xpired events on the parallel approach in dynamic scenarios.

b. Missed Situations vs. #Situation's Microservices vs. # Situation's verification

In this experiment, we have evaluated the number of missed situations versus the number of submitted microservices as well as the number of situation's verification. When the number of situation's verification increases with the number of situations microservices, the number of missed situations increases as shown in figure 7.11. Furthermore, the limited number of processor cores, the time consumed by the filtering process and simultaneous execution of situations identifier microservices cause the increase of the number of missed situations.

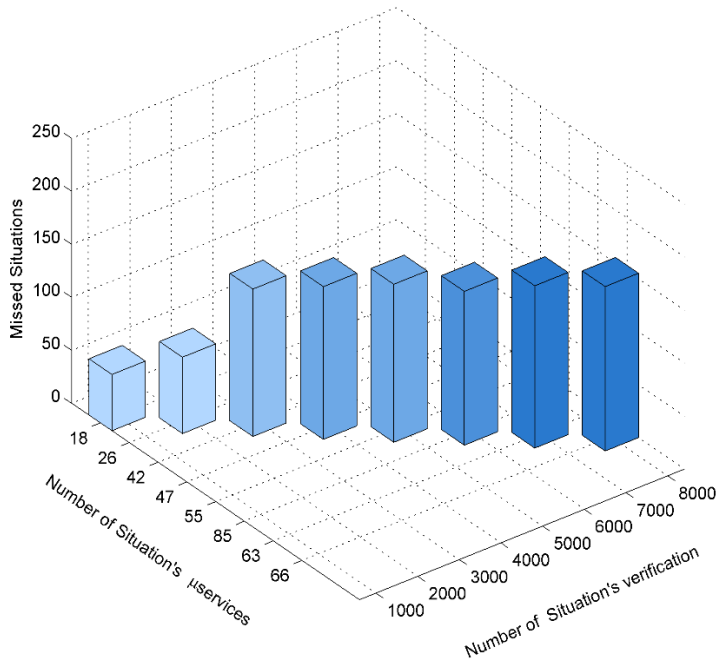


Figure 7.11: Evaluation of missed situations on the parallel approach in dynamic scenarios.

7.3.4 Comparison with pipelining-based framework

7.3.4.1 Impact of sensors number

In this section, we seek to study the impact of sensors number on the event detection latency and throughput. We have performed several tests under varied number of sensors ranging from 200 to 1000. Experiments are performed using two types of sensors: scalar (temperature sensor) and multimedia (camera sensor) with a fixed sensor frequency of 500ms to collect context data. Therefore, we have developed a simulator that generates random values of temperature based on standard deviation with configurable sampling frequency. In addition, it produces a random image frame that represents an image of an office with no one inside, or with at least people inside. Temperature and camera events are considered as simple events. An atomic event condition *high_temp* is associated with the temperature sensor to detect if the sensed temperature is higher than 22 °C. An atomic event condition *face_detected* is associated with the camera sensor to detect if the image frame contains at least one person.

To evaluate the performance of our parallel microservice-based framework, 4 main steps have been applied: (1) deploying 200 sensors with a fixed frequency of 500ms, where each sensor is executed by a microservice; (2) defining two complex events statement (using two combination operators: overlap and sequential) for temperature and camera stream detection associated with temperature and camera sensors respectively: *face_detected* OVERLAPS *high_temp* and SEQ(*face_detected*, *high_temp*), (3) performing complex events detection through a simulation of 10 minutes, and evaluating the throughput and event average detection latency per minute, (4) repeat all previous steps with 400, 600, 800 and 1000 sensors.

We have compared the proposed parallel microservice-based framework with the pipelining-based framework [118] by evaluating the events detection performance. Figure 7.12 shows a throughput comparison of the proposed approach vs pipelining-based framework with varied number of sensors (200 – 100) using overlap operator (part a) and sequential operator (part b). Results show that the proposed approach promotes better performance in terms of throughput than pipelining-based framework. Clearly, temporal operators including OVERLAPS (figure 7.12.a) and SEQ (figure 7.12.b) have slightly different performance since they have different detection processes that can affect the throughput. Furthermore, as we can see in figure 7.12, results show that the proposed approach can support up to 60 163 events per minute with OVERLAPS operator and 60 148 events per minute with SEQ operator. On the other part, the pipelining-based framework can up only to 11 631 events per minute with OVERLAPS operator and 10 034 events per minute with SEQ operator. The proposed approach outperforms the pipelining-based framework, which yields better throughput values across all number of sensors employing either sequential or overlap operators. This shows that our parallel microservice-based framework can support a high workload case compared to the pipelining-based framework.

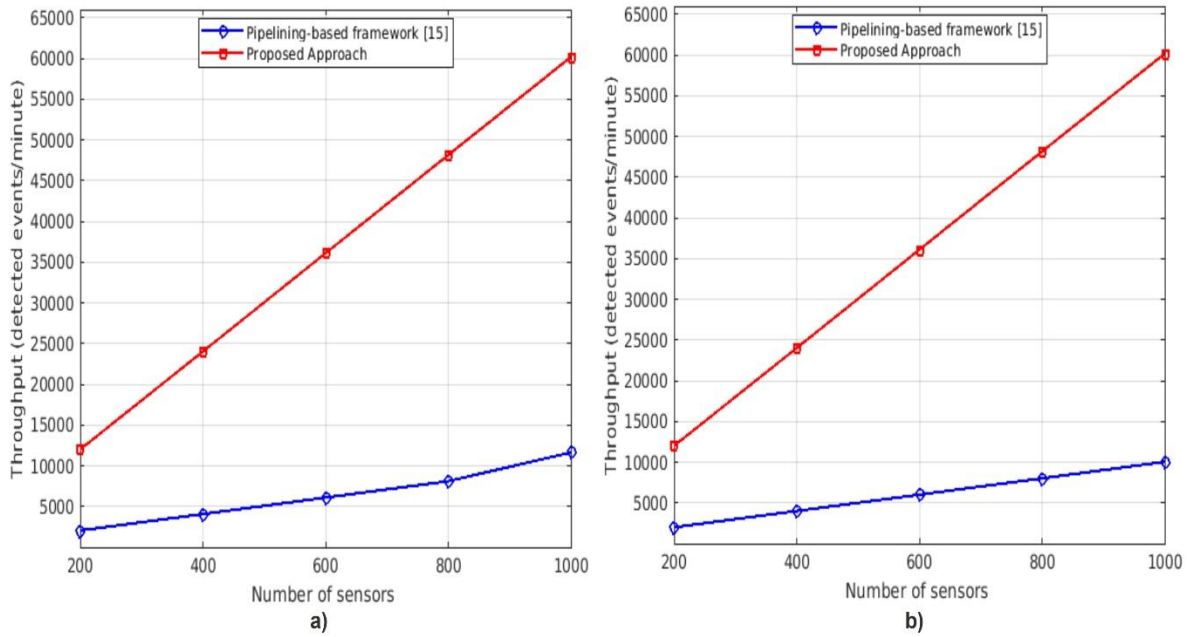


Figure 7.12: Throughput comparison of the proposed approach vs pipelining-based framework [118] using a) overlap operator, b) sequential operator.

Latency is one of the main metrics that can evaluate the system performance in terms of speed of event detection. Research studies in pervasive computing suggest that the defection latency should be less than 1000ms in most cases [118]. Figure 7.13 shows a Latency comparison of the proposed approach vs pipelining-based framework [118] using overlap operator (part a) and sequential operator (part b). Results show that increasing the number of sensors has no effect on the event detection latency in both approaches. Furthermore, as we can see in figure 7.13, results show that the proposed approach has an average event detection latency of 2.4ms with OVERLAPS operator and 20ms with SEQ operator. On the other part, the pipelining-based framework has an average event detection latency of 522ms with OVERLAPS operator and 588ms with SEQ operator. The proposed approach outperforms the pipelining-based framework, which yields low latency across all number of sensors employing either overlap or sequential operators.

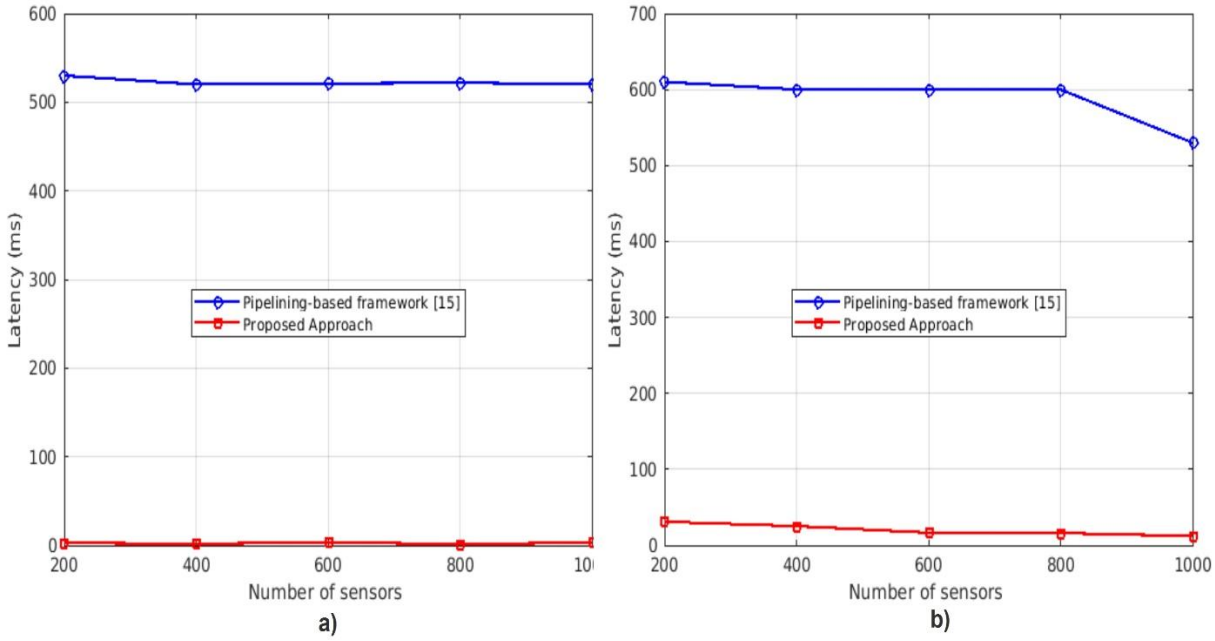


Figure 7.13: Average latency comparison of the proposed approach vs pipelining-based framework [118] using a) overlap operator, b) sequential operator.

7.3.4.2 Impact of sensor's frequency

In pervasive environment we can find various sensors with different sampling frequencies. In fact, small sampling frequencies may overload the system by receiving a huge number of incoming events in a small period of time, which may affect negatively the system's performance in terms of throughput and event detection latency. In order to verify how our proposed approach performs under different sampling frequency, we have conducted a set of experiments with the same applied steps in the previous experiments, except varying sensors frequency ranging from 50ms to 1000ms. In these experiments, we have fixed the number of sensors to 1000 sensors.

We have compared the impact of varying sensor's frequency on the event detection throughput on both proposed approach and pipelining-based framework [118]. Figure 4.14 shows a throughput comparison of the proposed approach vs pipelining-based framework using overlap operator (part a) and sequential operator (part b). As shown in figure 4.14 at 200ms sampling frequency, results show that the proposed approach can support up to 149 750 events per minute with OVERLAPS operator and 149 737 events per minute with SEQ operator. On the other part, the pipelining-based framework can up only to 28 600 events per minute with OVERLAPS operator and 25 000 events per minute with SEQ operator. The proposed approach outperforms the pipelining-based framework, which yields better throughput values across all varied sensor's frequency employing either sequential or overlap operators.

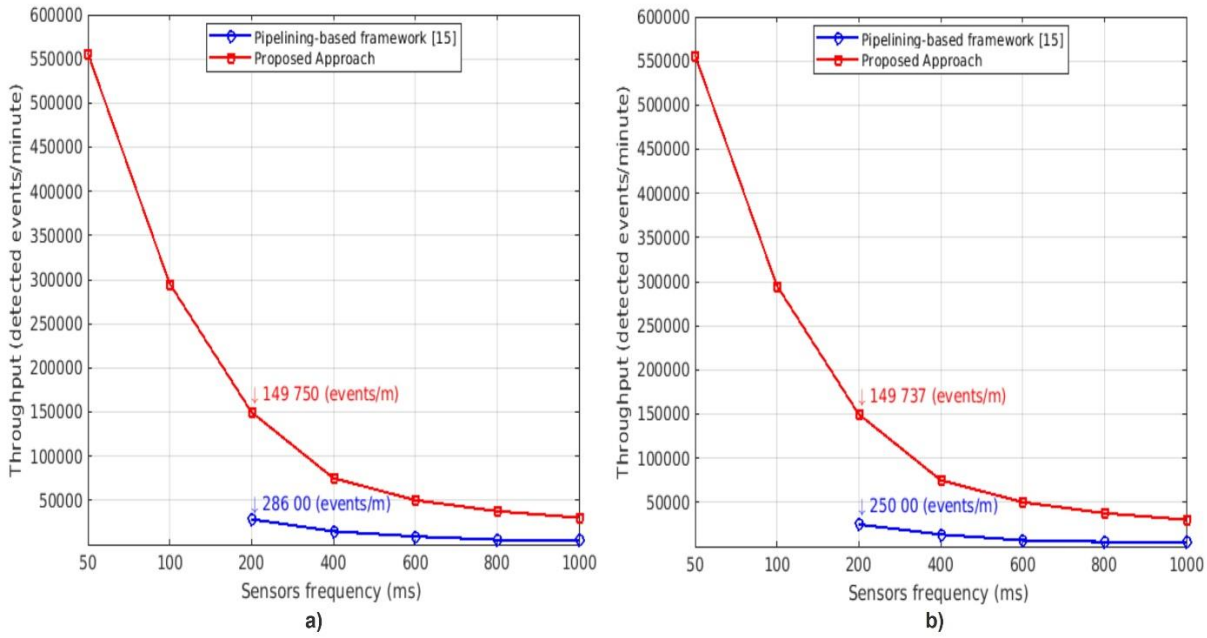


Figure 7.14: Throughput comparison of the proposed approach vs pipelining-based framework [118] using a) overlap operator, b) sequential operator (number of sensors is fixed to 1000).

In addition, we have compared the impact of varying sensor's frequency on the event detection latency on the proposed approach and pipelining-based framework [118]. Figure 4.15 shows an event detection latency comparison of the proposed approach vs pipelining-based framework using overlap operator (part a) and sequential operator (part b). As shown in figure 7.15 at 400ms sampling frequency, results show that the proposed approach has an average event detection latency of 1.3ms with OVERLAPS operator and 7.7ms with SEQ operator. On the other part, the pipelining-based framework has an average event detection latency of 800ms with OVERLAPS operator and 850ms with SEQ operator. The proposed approach outperforms the pipelining-based framework, which yields low latency across all varied sensor's frequency employing either overlap or sequential operators.

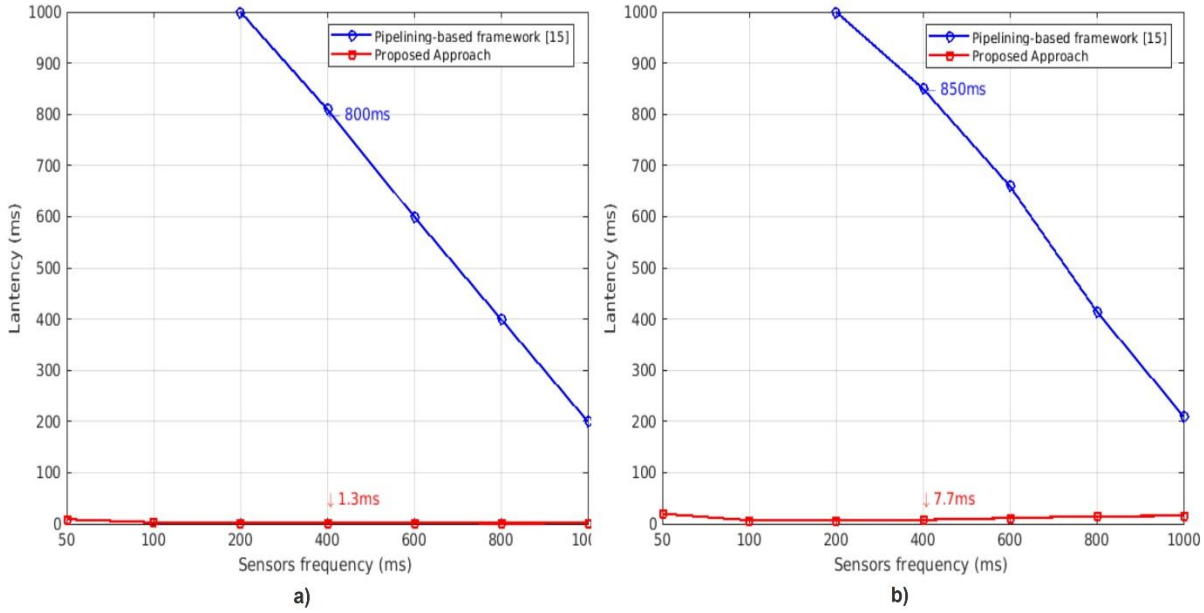


Figure 7.15: Average latency comparison of the proposed approach vs pipelining-based framework [118] using a) overlap operator, b) sequential operator (number of sensors is fixed to 1000).

As observed from the experiments above, the proposed approach is better than the pipelining-based framework in terms of managing and detecting complex events. Furthermore, the proposed approach is more efficient and decreases considerably the latency, which makes it practicable to be used in near-real-time pervasive environments.

To sum up, the pipelining-based framework adopts the process paradigm for handling event detections [118]. In fact, it creates for each event detection unit an entire process where each process is responsible for handling the detection of atomic or complex events. However, creating, running, and managing processes for small units of code (tasks) is not feasible and may slow down system performances. Therefore, we have used a lightweight implementation (threads) to improve the system performance. We conclude that using parallel threads as microservices for events detections improves event detection in terms of throughput and latency.

7.3.5 Synthesis of results analysis and discussions

The experiments on the proposed parallel approach using events random dataset attain promising results compared to the sequential approach in terms of expired event ratio, missed situation ratio, and execution time. As a result, the proposed parallel approach is much better than the sequential approach in managing and identifying composite situations. In fact, the proposed parallel approach achieves a near-real-time situation identification for context-aware pervasive applications. Further, we have achieved satisfying scalability in terms of task management and thread execution on an Intel processor core i5-2430 2.4 GHz in which the system can manage 100 thread easily (see figures 7.8 and 7.9).

Furthermore, we have conducted several experiments to compare the performance of the proposed parallel approach vs pipelining-based approach. Experiments examine the impact of increasing the number of sensors and varying sensor's frequency on both approaches. The proposed parallel approach outperforms the pipelining-based approaches. Compared to the pipelining-based approach, results show an average improvement of 82.4 % in terms of throughput. Also, we have seen an average improvement of -520ms in terms of event detection latency. As result, using shared threads instead of processes implementation yields better performance in terms of both throughput and latency.

In future works, we intend to experiment our proposal using real-world connected objects. Therefore, we plan to go into a real IoT infrastructure to perform large-size applications and improve the execution time using high-performance machines.

7.4 Conclusion

In this Chapter, we have proposed an implementation and evaluation of prototype framework for managing context-aware composite situations of the smart environment. It is based on a multi-layered ontology model to describe the composite situations semantically. They are expressed through numerous simple situations and rich composition operators (*parallel, sequence, alternative and recurrence*). We take into account a large number of situations on the fly. Thus, we dynamically and easily filter new relevant situations rules using flexible context-aware microservices-based architecture. Several experiments were conducted using a real-life motivating scenario. Experimental results have shown that the proposed approach provides better efficiency while ensuring a very low number of missed situations. A comparison with the sequential approach shows that the proposed parallel approach is more efficient and decreases the number of expired events. However, the situation checking number must be improved in future works. Further, we have conducted several experimental comparisons between the proposed parallel approach and the pipelining-based approach. Experimental results have shown that the proposed parallel approach outperforms the pipelining-based approaches in terms of throughput and event detection latency.

Future work can focus on the validation of the proposed architecture by implementing a global framework using real-world multimedia sensors devices (Arduino, Raspberry), smart objects and semantic web technologies.

General Conclusion

1. Summary

The emerging idea of context-aware pervasive computing systems is rapidly finding its path in assisting users everywhere, anytime, through widespread sensors and mobile devices. In this thesis, our main goal is to serve users during their daily life activities in various smart domains (health, transport, home, drive, etc.) by providing them with relevant services according to their current needs, preferences and situations. Despite the dynamicity of user's needs and the heterogeneity of smart objects in terms of hardware/software features, communication protocols, and context data types in pervasive environment, we have opted for using generic ontology models to hide this dynamicity and heterogeneity. This choice aims to achieve a shared and common knowledge representation model. Therefore, we have introduced a new ontology entitled Multi-OCSM (Multi-layered Ontology-based Composite Situation Model), which is based on SSN (Semantic Sensor Network ontology) [49], DUL (DOLCE Ultra-Light Ontology) [52], and OWL-S (OWL-based Web service ontology) [110] to represent and manage both simple/composite situations in pervasive computing systems including composition operators (parallel, sequence, recurrence and alternative). Moreover, it supports SPARQL queries allowing users to explore the ontology and customize the situations filtering according to their constraints and preferences.

Furthermore, we have found that the management and identification of a large number of user's composite situations cost a significant processing time. Therefore, we have opted to propose a real-time ontology-based approach for parallel composite situation reasoning to accelerate the identification of urgent situations. In this scope, we have used Java 8 concurrency APIs [21] to manage multithreading tasks that process parallel incoming events to improve the parallel identification of composite situations. In addition, we have proposed a priority strategy for speeding up the queued urgent situation rules that run simultaneously across collaborating microservices.

Due to the huge number of user's daily activities, it is a cumbersome task for each user to identify his specific rules in any daily life cases. Therefore, we have proposed a rule-based recommendation system for injecting new situation rules into user's agenda. In fact, we have relied on the Multi-OCSM ontology to suggest meaningful situation rules that meet users' needs during their movements through different smart domains.

To summarize, the main contributions of this thesis are:

The first contribution realizes a novel user-centric composite situation ontology based on multi-layered context-aware semantic microservices called Multi-OCSM (Multi-layered Ontology-based

Composite Situation Model) [19]. It described both simple and composite situation rules and its semantic description including composition operators (*parallel, sequence, recurrence and alternative*), prioritized situations rules, and reconfiguration actions in order to ensure service continuity and immediate response time. This ontology facilitated the management of a large number of heterogeneous smart objects and services, semantic features related to the context data for the interpretation of parallel events and interoperability of different incoming events and highlights the semantic rules of user.

The second contribution realizes an extension of Kali-smart middleware [9]. It provides composition operators including parallel, sequence, alternative, and recurrence in the modeling of composite situations. The main virtue of such extension is to automate the generation of reconfigurable mobile applications and to ensure a loosely coupled architecture, which implies flexible adaptability in pervasive environments.

The third contribution consists in proposing a novel parallel identification framework of composite situations for smart environments [19], which describes situation rules using context-aware microservices-based approach. It ensures parallel monitoring and processing of streaming context data in order to identify both simple/composite situations. The identification process is based on the Multi-OCSM ontology to identify dynamically situation changes. Further, a priority mechanism is proposed to improve the situation's identification process and speeding up queued urgent situations that have high priority.

The fourth contribution realizes a context-aware recommendation system for suggesting and injecting meaningful situation rules into user's agenda [20]. Further, it extends our Multi-OCSM to classify situation rules before applying the recommendation process to achieve a high quality of recommendation considering three context categories (user preference, situation context and device capability).

To validate our work, we have a prototype developed for composite situations parallel identification based on the Multi-OCSM reasoning for services reconfiguration and adaptation. To realize our prototype, we have relied on Protégé [113] to implement the Multi-OCSM and Java 8 concurrency APIs [21] to implement the parallel identification framework. The prototype provides several functionalities including monitoring and management of context, filtering and the factorization of relevant situation rules, prioritizing of situation rules, and parallel identification of both simple/composite situations. Further, we have conducted a set of experiments on a continuous context monitoring according to a real-life motivating scenario based on two approaches including parallel and sequential identification processes. Experimental results have shown that the proposed parallel approach provides better efficiency while ensuring a very low number of missed situations. A comparison with the sequential approach shows that the proposed parallel approach is more efficient and decreases both the number of expired events and the response time.

Our contributions are very interesting for various smart application domains. We have proposed an agile framework for parallel identification and injection of composite situation in smart-* domains. The main advantages of the proposed solution compared to existing related works are the real-time semantic composite situation management approach in reconfigurable smart environments, as well as, less execution time for identifying urgent situations, and remarkable effectiveness and flexibility.

2. Future Works

The work presented in this thesis has addressed various research areas in pervasive computing systems regarding situations modeling, identification, and recommendation through loosely coupled and flexible architecture. To this end, we have elaborated some perspectives that seem relevant to be addressed in future works in order to improve our proposal. These perspectives are:

- **Extending Multi-OCSM ontology**

  Technologies and concepts continuously evolve across generations. Therefore, we seek for extending our Multi-OCSM with rich new concepts and technologies such as new communication protocols (5G). We intend for proposing a Domain Specific Language (DSL) for easy specification of user constraints. In addition, we seek for integrating consistency techniques with the ontology model to check semantic violations of user constraints.

- **Improving the performance of situation identification system**

  We have noticed that overloading the processor (i5-2430 2.4 GHz) with a significant number of threads (>100) gradually increases the execution time of the identification process. Therefore, we intend to propose a new intelligent thread-based management algorithm that collects the current state of each processor (number of cores, number of threads in each core) then splits the queued tasks on processor cores with the slightest load. This makes the identification process more scalable while receiving parallel tasks for execution.

- **Ongoing to real-world smart objects**

  Our prototype has been tested on a dataset resulting in 50,000 sensor events generated using a random events generator. We intend to experiment our proposal using real-world multimedia sensors and devices (Arduino, Raspberry, etc.). Therefore, we plan to go into a real IoT infrastructure to perform large-size applications using high-performance machines.

Bibliography

[1] D. Evans, "How the Next Evolution of the Internet Is Changing Everything," p. 11, 2011.

[2] J. Ye, S. Dobson, and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive and mobile computing*, vol. 8, no. 1, pp. 36–66, 2012.

[3] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *International symposium on handheld and ubiquitous computing*, 1999, pp. 304–307.

[4] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," *Journal of Web Semantics*, vol. 17, pp. 25–32, 2012.

[5] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics," *Personal and Ubiquitous Computing*, vol. 21, no. 3, pp. 475–487, 2017.

[6] E. Mansour, R. Chbeir, and P. Arnould, "HSSN: an ontology for hybrid semantic sensor networks," in *Proceedings of the 23rd International Database Applications &Engineering Symposium*, 2019, pp. 1–10.

[7] R. Rouvoy *et al.*, "Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments," in *Software engineering for self-adaptive systems*, Springer, 2009, pp. 164–182.

[8] K. Da, M. Dalmau, and P. Roose, "Kalimucho: middleware for mobile applications," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, 2014, pp. 413–419.

[9] A. Alti, A. Lakehal, S. Laborie, and P. Roose, "Autonomic semantic-based context-aware platform for mobile applications in pervasive environments," *Future Internet*, vol. 8, no. 4, p. 48, 2016.

[10] S. Zachariadis, C. Mascolo, and W. Emmerich, "Satin: a component model for mobile self organisation," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2004, pp. 1303–1321.

[11] K. Zhai, B. Xu, W. K. Chan, and T. H. Tse, "CARISMA: a context-sensitive approach to race-condition sample-instance selection for multithreaded applications," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, 2012, pp. 221–231.

[12] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *1994 First Workshop on Mobile Computing Systems and Applications*, 1994, pp. 85–90.

[13] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the marketplace," *IEEE personal communications*, vol. 4, no. 5, pp. 58–64, 1997.

[14] J. Pascoe, "Adding generic contextual capabilities to wearable computers," in *Digest of papers. second international symposium on wearable computers (cat. no. 98ex215)*, 1998, pp. 92–99.

[15] C. Angsuchotmetee, R. Chbeir, and Y. Cardinale, "MSSN-Onto: An ontology-based approach for flexible event processing in Multimedia Sensor Networks," *Future Generation Computer Systems*, vol. 108, pp. 1140–1158, 2020.

[16] D. Zografisto, "Support for context-aware healthcare in Ambient Assisted Living," 2012.

[17] M. Alirezaie *et al.*, "An ontology-based context-aware system for smart homes: E-care@ home," *Sensors*, vol. 17, no. 7, p. 1586, 2017.

[18] J.-Y. Tigli *et al.*, "WComp middleware for ubiquitous computing: Aspects and composite event-based Web services," *annals of telecommunications-annales des télécommunications*, vol. 64, no. 3–4, pp. 197–214, 2009.

[19] A. Lakehal, A. Alti, and P. Roose, "A semantic event based framework for complex situations modeling and identification in smart environments," *International Journal of Advanced Computer Research*, vol. 9, no. 43, pp. 212–221, 2019.

[20] A. Lakehal, A. Alti, S. Laborie, and R. Philippe, "Ontology-based Context-aware Recommendation Approach For Dynamic Situations Enrichment," in *2018 13th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*, 2018, pp. 81–86.

[21] J. F. González, *Mastering Concurrency Programming with Java 9*. Packt Publishing Ltd, 2017.

[22] M. Weiser, "The Computer for the 21 st Century," *Scientific american*, vol. 265, no. 3, pp. 94–105, 1991.

[23] K. Ashton, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[24] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *From active data management to event-based systems and more*, Springer, 2010, pp. 242–259.

[25] M. Weiser, "The computer for the 21st century," *IEEE pervasive computing*, vol. 1, no. 1, pp. 19–25, 2002.

[26] D. J. Cook and S. K. Das, "How smart are our environments? An updated look at the state of the art," *Pervasive and mobile computing*, vol. 3, no. 2, pp. 53–73, 2007.

[27] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille, "A long-term evaluation of sensing modalities for activity recognition," in *International conference on Ubiquitous computing*, 2007, pp. 483–500.

[28] K. Henricksen and J. Indulska, "Modelling and using imperfect context information," in *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, 2004, pp. 33–37.

[29] L. Atallah and G.-Z. Yang, "The use of pervasive sensing for behaviour profiling—a survey," *Pervasive and mobile computing*, vol. 5, no. 5, pp. 447–464, 2009.

[30] L. Sanchez, J. Lanza, R. Olsen, M. Bauer, and M. Girod-Genet, "A generic context management framework for personal networking environments," in *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking &Services*, 2006, pp. 1–8.

[31] "Definitions, Meanings, Synonyms, and Grammar by Oxford Dictionary on Lexico.com." https://www.lexico.com/ (accessed Sep. 05, 2020).

[32] "Cambridge Dictionary | English Dictionary, Translations & Thesaurus." https://dictionary.cambridge.org/ (accessed Sep. 05, 2020).

[33] P. Brézillon and J.-C. Pomerol, "Contextual knowledge sharing and cooperation in intelligent assistant systems," *Le travail humain*, pp. 223–246, 1999.

[34] A. Schmidt, "Implicit human computer interaction through context," *Personal technologies*, vol. 4, no. 2, pp. 191–199, 2000.

[35] J. Strassner, S. van der Meer, D. O'Sullivan, and S. Dobson, "The use of context-aware policies and ontologies to facilitate business-aware network management," *Journal of Network and Systems Management*, vol. 17, no. 3, pp. 255–284, 2009.

[36] S. Ahn and D. Kim, "Proactive context-aware sensor networks," in *European Workshop on Wireless Sensor Networks*, 2006, pp. 38–53.

[37] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *IEEE network*, vol. 8, no. 5, pp. 22–32, 1994.

[38] A. H. Van Bunningen, L. Feng, and P. M. Apers, "Context for ubiquitous data management," in *International Workshop on Ubiquitous Data Management*, 2005, pp. 17–24.

[39] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys &tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[40] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach," *Pervasive and mobile computing*, vol. 2, no. 1, pp. 37–64, 2006.

[41] A. M. Bernardos, P. Tarrio, and J. R. Casar, "A data fusion framework for context-aware mobile services," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 606–613.

[42] R. Karchoud, P. Roose, M. Dalmau, A. Illaramendi, and S. Ilarri, "Long life application: approach for user context management and situation understanding," in *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, 2016, pp. 45–53.

[43] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[44] N. Bricon-Souf and C. R. Newman, "Context awareness in health care: A review," *international journal of medical informatics*, vol. 76, no. 1, pp. 2–12, 2007.

[45] A. Moon, H. Kim, K. Lee, and H. Kim, "Designing CAMUS based context-awareness for pervasive home environments," in *2006 International Conference on Hybrid Information Technology*, 2006, vol. 1, pp. 666–672.

[46] R. Mayrhofer, *An architecture for context prediction*. Citeseer, 2005.

[47] N. Guarino, D. Oberle, and S. Staab, "What is an ontology?," in *Handbook on ontologies*, Springer, 2009, pp. 1–17.

[48] S. Grimm, A. Abecker, J. Völker, and R. Studer, "Ontologies and the semantic web," in *Handbook of semantic web technologies*, 2011.

[49] "Semantic Sensor Network (SSN) W3C Incubator Group." https://www.w3.org/TR/vocab-ssn/ (accessed Sep. 05, 2020).

[50] "Sensor, Observation, Sample, and Actuator (SOSA) Ontology." https://www.w3.org/ns/sosa/ (accessed Sep. 05, 2020).

[51] "Smart Applications Reference Ontology (SAREF) Portal." https://saref.etsi.org/index.html (accessed Sep. 05, 2020).

[52] "DOLCE Ultra-Light (DUL) Ontology." http://www.ontologydesignpatterns.org/ont/dul/DUL.owl (accessed Sep. 05, 2020).

[53] "Friend of a Friend Ontology (FOAF) Ontology." http://xmlns.com/foaf/spec/ (accessed Sep. 05, 2020).

[54] A. K. Dey and G. D. Abowd, "Cybreminder: A context-aware system for supporting reminders," in *International Symposium on Handheld and Ubiquitous Computing*, 2000, pp. 172–186.

[55] R. Karchoud, A. Illarramendi, S. Ilarri, P. Roose, and M. Dalmau, "Long-life application," *Personal and Ubiquitous Computing*, vol. 21, no. 6, pp. 1025–1037, 2017.

[56] S. A. Butt, T. Jamal, M. A. Azad, A. Ali, and N. S. Safa, "A multivariant secure framework for smart mobile health application," *Transactions on Emerging Telecommunications Technologies*, p. e3684, 2019.

[57] R. Yus, E. Mena, S. Ilarri, and A. Illarramendi, "SHERLOCK: Semantic management of location-based services in wireless environments," *Pervasive and Mobile Computing*, vol. 15, pp. 87–99, 2014.

[58] A. B. James, "Activities of daily living and instrumental activities of daily living," 2009.

[59] G. Okeyo, L. Chen, and H. Wang, "An Agent-mediated Ontology-based Approach for Composite Activity Recognition in Smart Homes.," *J. UCS*, vol. 19, no. 17, pp. 2577–2597, 2013.

[60] J. M. N. Ramírez, P. Roose, M. Dalmau, and Y. Cardinale, "An event detection framework for the representation of the AGGIR variables," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2018, pp. 153–160.

[61] N. Ferry, V. Hourdin, S. Lavirotte, G. Rey, M. Riveill, and J.-Y. Tigli, *Wcomp, middleware for ubiquitous computing and system focused adaptation*. 2012.

[62] B. Bertran, J. Bruneau, D. Cassou, N. Loriant, E. Balland, and C. Consel, "DiaSuite: A tool suite to develop Sense/Compute/Control applications," *Science of Computer Programming*, vol. 79, pp. 39–51, 2014.

[63] R. Allen, R. Douence, and D. Garlan, "Specifying and analyzing dynamic software architectures," in *International Conference on Fundamental Approaches to Software Engineering*, 1998, pp. 21–37.

[64] A. Mehrotra, V. Pejovic, and M. Musolesi, "SenSocial: a middleware for integrating online social networks and mobile sensing data streams," in *Proceedings of the 15th International Middleware Conference*, 2014, pp. 205–216.

[65] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," *arXiv preprint arXiv:2003.05055*, 2020.

[66] A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell, "Reasoning about uncertain contexts in pervasive computing environments," *IEEE Pervasive computing*, vol. 3, no. 2, pp. 62–70, 2004.

[67] G. Papamarkos, "Event-Condition-Action Rule Languages over Semistructured Data," PhD Thesis, University of London, 2007.

[68] H. Mshali, T. Lemlouma, and D. Magoni, "Adaptive monitoring system for e-health smart homes," *Pervasive and Mobile Computing*, vol. 43, pp. 1–19, 2018.

[69] J. C. Augusto, J. Liu, P. McCullagh, H. Wang, and J.-B. Yang, "Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home," *International Journal of Computational Intelligence Systems*, vol. 1, no. 4, pp. 361–378, 2008.

[70] J.-B. Yang, J. Liu, J. Wang, H.-S. Sii, and H.-W. Wang, "Belief rule-base inference methodology using the evidential reasoning approach-RIMER," *IEEE Transactions on systems, Man, and Cybernetics-part A: Systems and Humans*, vol. 36, no. 2, pp. 266–285, 2006.

[71] M. Sioutis, M. Alirezaie, J. Renoux, and A. Loutfi, "Towards a synergy of qualitative spatio-temporal reasoning and smart environments for assisting the elderly at home," in *IJCAI Workshop on Qualitative Reasoning*, 2017, pp. 901–907.

[72] M. M. H. Al-Jawad, "A context-aware method for verifying user identity in pervasive computing environments," PhD Thesis, University of Salford, 2017.

[73] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Ontology-based models in pervasive computing systems," *The Knowledge Engineering Review*, vol. 22, no. 4, pp. 315–347, 2007.

[74] D. Riboni and C. Bettini, "Context-aware activity recognition through a combination of ontological and statistical reasoning," in *International Conference on Ubiquitous Intelligence and Computing*, 2009, pp. 39–53.

[75] J. Ye, G. Stevenson, and S. Dobson, "KCAR: A knowledge-driven approach for concurrent activity recognition," *Pervasive and Mobile Computing*, vol. 19, pp. 47–70, 2015.

[76] G. Meditskos and I. Kompatsiaris, "iKnow: Ontology-driven situational awareness for the recognition of activities of daily living," *Pervasive and Mobile Computing*, vol. 40, pp. 17–41, 2017.

[77] A. R. Jiménez and F. Seco, "Multi-Event Naive Bayes Classifier for Activity Recognition in the UCAmI Cup," in *Multidisciplinary Digital Publishing Institute Proceedings*, 2018, vol. 2, no. 19, p. 1264.

[78] G. Swapna, K. P. Soman, and R. Vinayakumar, "Diabetes Detection Using ECG Signals: An Overview," in *Deep Learning Techniques for Biomedical and Health Informatics*, Springer, 2020, pp. 299–327.

[79] C. Wojek, K. Nickel, and R. Stiefelhagen, "Activity recognition and room-level tracking in an office environment," in *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006, pp. 25–30.

[80] M. S. Ryoo and J. K. Aggarwal, "Recognition of composite human activities through context-free grammar based representation," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2006, vol. 2, pp. 1709–1718.

[81] J.-Y. Yang, J.-S. Wang, and Y.-P. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers," *Pattern recognition letters*, vol. 29, no. 16, pp. 2213–2220, 2008.

[82] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert systems with applications*, vol. 59, pp. 235–244, 2016.

[83] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds, and G. D. Abowd, "At the flick of a switch: Detecting and classifying unique electrical events on the residential power line (nominated for the best paper award)," in *International Conference on Ubiquitous Computing*, 2007, pp. 271–288.

[84] T. Kanda, D. F. Glas, M. Shiomi, H. Ishiguro, and N. Hagita, "Who will be the customer? A social robot that anticipates people's behavior from their trajectories," in *Proceedings of the 10th international conference on Ubiquitous computing*, 2008, pp. 380–389.

[85] Z. Chen, Q. Zhu, Y. C. Soh, and L. Zhang, "Robust human activity recognition using smartphone sensors via CT-PCA and online SVM," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3070–3080, 2017.

[86] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*, Springer, 2007, pp. 325–341.

[87] M. Tkalčič, U. Burnik, and A. Košir, "Using affective parameters in a content-based recommender system for images," *User Modeling and User-Adapted Interaction*, vol. 20, no. 4, pp. 279–311, 2010.

[88] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, P. Piazzolla, and M. Quadrana, "Content-based video recommendation system based on stylistic visual features," *Journal on Data Semantics*, vol. 5, no. 2, pp. 99–113, 2016.

[89] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowledge-Based Systems*, vol. 157, pp. 1–9, 2018.

[90] B. Chen, H. He, and J. Guo, "Constructing maximum entropy language models for movie review subjectivity analysis," *Journal of Computer Science and Technology*, vol. 23, no. 2, pp. 231–239, 2008.

[91] C. Jin *et al.*, "Chi-square statistics feature selection based on term frequency and distribution for text categorization," *IETE journal of research*, vol. 61, no. 4, pp. 351–362, 2015.

[92] D. Zhang, "An item-based collaborative filtering recommendation algorithm using slope one scheme smoothing," in *2009 Second International Symposium on Electronic Commerce and Security*, 2009, vol. 2, pp. 215–217.

[93] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005, pp. 471–475.

[94] A. Hernando, J. Bobadilla, and F. Ortega, "A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model," *Knowledge-Based Systems*, vol. 97, pp. 188–202, 2016.

[95] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[96] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*, Springer, 2011, pp. 217–253.

[97] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M.- Thalmann, "Time-aware point-of-interest recommendation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 363–372.

[98] G. Zhou, S. Zhang, Y. Fan, J. Li, W. Yao, and H. Liu, "Recommendations based on user effective point-of-interest path," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 10, pp. 2887–2899, 2019.

[99] L. Hong, L. Zou, C. Zeng, L. Zhang, J. Wang, and J. Tian, "Context-aware recommendation using role-based trust network," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 2, pp. 1–25, 2015.

[100] Y.-D. Seo, Y.-G. Kim, E. Lee, and D.-K. Baik, "Personalized recommender system based on friendship strength in social network services," *Expert Systems with Applications*, vol. 69, pp. 135–148, 2017.

[101] M. Elahi, F. Ricci, and N. Rubens, "A survey of active learning in collaborative filtering recommender systems," *Computer Science Review*, vol. 20, pp. 29–50, 2016.

[102] I. Soboroff and C. Nicholas, "Combining content and collaboration in text filtering," in *Proceedings of the IJCAI*, 1999, vol. 99, pp. 86–91.

[103] Z. Yu, X. Zhou, D. Zhang, C.-Y. Chin, X. Wang, and J. Men, "Supporting context-aware media recommendations for smart phones," *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 68–75, 2006.

[104] M. Nilashi, O. bin Ibrahim, N. Ithnin, and N. H. Sarmin, "A multi-criteria collaborative filtering recommender system for the tourism domain using Expectation Maximization (EM) and PCA–ANFIS," *Electronic Commerce Research and Applications*, vol. 14, no. 6, pp. 542–562, 2015.

[105] S. Suria and K. Palanivel, "An enhanced web service recommendation system with ranking QoS information," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 4, no. 1, pp. 116–121, 2015.

[106] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation in an evolving service ecosystem," in *2014 IEEE International Conference on Web Services*, 2014, pp. 25–32.

[107] R. Karchoud, P. Roose, M. Dalmau, A. Illarramendi, and S. Ilarri, "All for one and one for all: dynamic injection of situations in a generic context-aware application," *Procedia computer science*, vol. 113, pp. 17–24, 2017.

[108] W. Wen and H. Miao, "Context-Based Service Recommendation System Using Probability Model in Mobile Devices," in *2016 4th International Conference on Enterprise Systems (ES)*, 2016, pp. 178–182.

[109] A. Lakehal, A. Alti, S. Laborie, and P. Roose, "A Semantic Agile Approach for Reconfigurable Distributed Applications in Pervasive Environments," *International Journal of Ambient Computing and Intelligence (IJACI)*, vol. 11, no. 2, pp. 48–67, 2020.

[110] J. E. L. de Vergara, V. A. Villagrá, and J. Berrocal, "Application of OWL-S to define management interfaces based on Web Services," in *IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, 2005, pp. 242–253.

[111] "The GeoNames geographical database." http://www.geonames.org/ (accessed Sep. 05, 2020).

[112] C. Dromzée, S. Laborie, and P. Roose, "A Semantic Generic Profile for Multimedia Document Adaptation," in *Intelligent Multimedia technologies for networking applications: techniques and tools*, IGI Global, 2013, pp. 225–246.

[113] "protégé." https://protege.stanford.edu/ (accessed Sep. 05, 2020).

[114] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi, *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[115] G. De Giacomo and M. Lenzerini, "TBox and ABox reasoning in expressive description logics.," *KR*, vol. 96, no. 316–327, p. 10, 1996.

[116] P. Eric and S. Andy, "Sparql query language for rdf," 2006.

[117] "WordNet Search." http://wordnetweb.princeton.edu/perl/webwn (accessed Sep. 05, 2020).

[118] C. Angsuchotmetee, R. Chbeir, Y. Cardinale, and S. Yokoyama, "A pipelining-based framework for processing events in multimedia sensor networks," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 247–250.

Appendix A

Implementation of Event Task, Situation Identifier Task and Extended Thread Pool Executor

A.1 Source code of Event Task class

```
package sample.concurrent.tasks;

import sample.console.ColorConsole;
import sample.datastructure.event.Event;
import sample.datastructure.event.SimpleEvent;
import sample.knowledgebase.DatabaseConnector;
import sample.datastructure.buffer.Buffer;
import sample.runnableTask.SituationManagerTask;
import java.util.Vector;
import java.util.concurrent.Future;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.locks.ReadWriteLock;
import java.util.concurrent.locks.ReentrantReadWriteLock;

public class EventTask extends ConcurrentTask {
      private int nextValue
      private Event event;
      private DatabaseConnector databaseConnector;
      private Vector<Buffer> producerBufferVector;
      private Vector<SituationIdentifierTask> situationIdentifierTaskVector;
      private Future<?> future
      private boolean cancellable;
      private ReadWriteLock readWriteLock = new ReentrantReadWriteLock();
      public static AtomicInteger numOfEvents = new AtomicInteger(0);

      public EventTask(Event event, DatabaseConnector databaseConnector, int priority) {
        super(priority);
        this.nextValue = 1;
        this.event = event;
        this.databaseConnector = databaseConnector;
        this.producerBufferVector = new Vector<>();
        this.producerBufferVector.add(SituationManagerTask.consumerBuffer);
        this. situationIdentifierTaskVector= new Vector<>();
        cancellable = true;
      }

      @Override
      public void execute() {
         try
         {
            numOfEvents.incrementAndGet();
            SimpleEvent simpleEvent = (SimpleEvent) event;
            while(!Thread.currentThread().interrupted())
            {
               readWriteLock.writeLock().lock();
               databaseConnector.updateNextSensorMeasuredValue(simpleEvent, nextValue++);
               readWriteLock.writeLock().unlock();
               for(Buffer producerBuffer : producerBufferVector)
               producerBuffer.add(((SimpleEvent) event).clone());
               TimeUnit.MILLISECONDS.sleep(simpleEvent.getSensor().getSensorFrequency().toMil

            }
         }
         catch (InterruptedException e) {
            System.out.println("Event interrupted "+event.getEventName());
         }
      }
```

```
@Override
public Object getObject() {
   return this;
}

public Future<?> getFuture() {
   return future;
}

public void setFuture(Future<?> future) {
   this.future = future;
}

public boolean isCancellable() {
   return cancellable;
}

public void setCancellable(boolean cancellable) {
  this.cancellable = cancellable;
}

public void addEventBuffer(Buffer buffer){
  if(!producerBufferVector.contains(buffer))
   producerBufferVector.add(buffer);
}
public boolean addSituationIdentifierTask(SituationIdentifierTask sitIdTask){
  if(!situationIdentifierTaskVector.contains(sitIdTask)){
   situationIdentifierTaskVector.add(sitIdTask);
   situationIdentifierTask.getEventTaskVector().add(this);
   return true;
  }
  else {
   return false;
  }
}

public boolean removeSituationIdentifierTask(SituationIdentifierTask sitIdTask){
  return situationIdentifierTaskVector.remove(sitIdTask);
}

public boolean kill(){
   if (situationIdentifierTaskVector.size()==0 && cancellable)
   {
      return future.cancel(true);
   }
   else
   {
      if(situationIdentifierTaskVector.size()==0 && !cancellable)
      {
          SituationManagerTask.mapIdEventToSituations.remove(event.getEventId());
      }
  }
  return false;
}

public boolean killNow(){
   if (situationIdentifierTaskVector.size()==0)
      return future.cancel(true);
   return false;
}

public String getId() {
   return event.getEventId();
}

public boolean compareAndSetIfHighPriority(int priority) {
   //the highest priority has the lowest number
   if(this.priority > priority)
   {
      this.priority = priority;
      return true;
   }
   return false;
}

public Event getEvent() {
```

```
            try {
               readWriteLock.readLock().lock();
               return event;
            } finally {
               readWriteLock.readLock().unlock();
            }
         }

         public SimpleEvent getClonedEvent() {
            try {
               readWriteLock.readLock().lock();
               return ((SimpleEvent) event).clone();
            } finally {
               readWriteLock.readLock().unlock();
            }
         }
}
```

A.2 Source code of Situation Identifier Task class

```
package sample.concurrent.tasks;

import sample.console.ColorConsole;
import sample.datastructure.event.SimpleEvent;
import sample.datastructure.buffer.Buffer;
import sample.datastructure.situation.Situation;
import sample.datastructure.situation.SituationStatus;
import sample.datastructure.situationRule.condition.SimpleCondition;
import sample.runnableTask.SituationBinder;
import sample.runnableTask.SituationManagerTask;
import java.util.Vector;
import java.util.concurrent.ConcurrentHashMap;

public class SituationIdentifierTask extends ConcurrentTask{
       private Situation situation;
       private String status;
       private Buffer producerBuffer;
       private SituationBinder situationBinder;
       private Vector<EventTask> eventTaskVector;
       private ConcurrentHashMap<String, EventTask> eventTaskRegistry;
       private Object object;

       public SituationIdentifierTask(Situation situation, int priority) {
          super(priority);
          this.situation = situation;
          this.producerBuffer = consumerBuffer;
          this.eventTaskVector =new Vector<>();
          status = SituationStatus.IDLE;
          situationBinder = null;
       }

       public void refreshObject(Object object){
          this.object = object;
       }

       @Override
       public void execute() {
          if(situationBinder == null)
             this.situationBinder = new SituationBinder(situation);
             try {
                if(object instanceof SimpleEvent)
                {
                   SimpleEvent consumedEvent = (SimpleEvent) object;
                   SimpleCondition simpleCondition =
                      situationBinder.bindSimpleConditionToNewSimpleEvent(consumedEvent);
                   situation.check();
                   if(situation.isChangedStatus())
                   {
                      if(situation.isChecked())
                         System.out.println(+"Situation :" + situation.getSituationName() + "
                                   Is changed status now / Checked ="+situation.isChecked()
                      else
                         System.out.println("Situation :" + situation.getSituationName() + "
```

```
                                Is changed status now / Checked ="+situation.isChecked());
                    }
                    else
                    {
                          System.out.println("*Situation : " + situation.getSituationName()
                                  " Is not changed status / Checked ="+situation.isChecked());
                    }
                }
                else
                {
                    if(object instanceof Situation)
                    {
                        Situation consumedSituation = (Situation) object;

                        situationBinder.bindAndUpdateSituationWithNewSituation(consumedSituation)
                        situation.check();
                        if(situation.isChangedStatus())
                        {
                            if(situation.isChecked())
                            {
                                System.out.println(+ "Composite Situation :" +
                                situation.getSituationName() + " Is changed status now / Checked
                                =" + situation.isChecked());
                            }
                            else
                            {
                                System.out.println("Composite Situation :" +
                                situation.getSituationName() + " Is changed status now / Checked
                                =" + situation.isChecked());
                            }
                        }
                        else
                        {
                              System.out.println("Composite Situation : " +
                              situation.getSituationName() + " Is not changed status / Checked
                              ="+situation.isChecked()+);
                        }
                    }
                 }
                 if(situation.isInvolved())
                       producerBuffer.add(situation);
            } catch (NullPointerException e){
                e.printStackTrace();
            }
    }

    public Object getObject() {
        return this;
    }

    public void setEventTaskRegistry(ConcurrentHashMap<String,EventTask> eventTaskReg) {
        this.eventTaskRegistry = eventTaskReg;
    }

    public Situation getSituation() {
        return situation;
    }

    public void setSituation(Situation situationRule) {
        this.situation = situationRule;
    }

    public Buffer getProducerBuffer() {
        return producerBuffer;
    }

    public void setProducerBuffer(Buffer consumerBuffer) {
        this.producerBuffer = consumerBuffer;
    }

    public SituationBinder getSituationBinder() {
        if(situationBinder == null)
            this.situationBinder = new SituationBinder(situation);
        return situationBinder;
    }

    public Vector<EventTask> getEventTaskVector() {
```

```
        return eventTaskVector;
    }

    public void setEventTaskVector(Vector<EventTask> eventTaskVector) {
        this.eventTaskVector = eventTaskVector;
    }

    public String getId(){
        return situation.getIdSituation();
    }

    public void setActive(){
        status = SituationStatus.ACTIVE;
    }

    public void setKilled(){
        status = SituationStatus.KILLED;
    }

    public boolean isActif(){
        return status.equals(SituationStatus.ACTIVE);
    }

    public boolean isKilled(){
        return status.equals(SituationStatus.KILLED);
    }

    public boolean kill(){
        if (situation.getSituationIdentifierTaskOfParentSituationMap().size()==0)
        {
            for (EventTask event : eventTaskVector)
            {
                event.removeSituationIdentifierTask(this);
                if(event.kill())
                {
                    eventTaskRegistry.remove(event.getId());
                    SituationManagerTask.mapIdEventToSituations.remove(event.getId());
                }
            }
            return true;
        }
        return false;
    }
}
```

A.3 Source code of Extended Thread Pool Executor class

```
package sample.concurrent.executor;

import sample.concurrent.tasks.ConcurrentFutureTask;
import sample.concurrent.tasks.ConcurrentTask;
import sample.parallel.log.Logger;
import java.util.Date;
import java.util.Map.Entry;
import java.util.concurrent.*;

public class ExtendedThreadPoolExecutor extends ThreadPoolExecutor {

    private ConcurrentHashMap<Runnable, Date> startTimes;
    private ConcurrentHashMap<String, ExecutorStatistics> executionStatistics;
    private static int CORE_POOL_SIZE = 1000;
    private static int MAXIMUM_POOL_SIZE = 1000;
    private static long KEEP_ALIVE_TIME = 10
    private static RejectedTaskController REJECTED_TASK_CONTROLLER =
                                                    new RejectedTaskController();


    public ExtendedThreadPoolExecutor() {
        super(CORE_POOL_SIZE, MAXIMUM_POOL_SIZE, KEEP_ALIVE_TIME, TimeUnit.SECONDS,
      new PriorityBlockingQueue<>(),REJECTED_TASK_CONTROLLER);
```

```
            startTimes = new ConcurrentHashMap<>();
            executionStatistics = new ConcurrentHashMap<>();
        }

        @Override
        protected void afterExecute(Runnable r, Throwable t) {
            super.afterExecute(r, t);
        }

        @Override
        protected void beforeExecute(Thread t, Runnable r) {
            super.beforeExecute(t, r);
            startTimes.put(r, new Date());
        }

        @Override
        protected <T> RunnableFuture<T> newTaskFor(Runnable runnable, T value) {
            return new ConcurrentFutureTask<>((ConcurrentTask) runnable);
        }


        public void writeStatistics() {
            for(Entry<String, ExecutorStatistics> entry: executionStatistics.entrySet()) {
                String user = entry.getKey();
                ExecutorStatistics stats = entry.getValue();
                Logger.sendMessage(user+":"+stats);
            }
        }
}
```

Appendix B

GUIs Establishing and Testing of the Motivating Scenario

B.1 Login GUI

To start using the application prototype, the user Mohammed needs to install and subscribe to the application using a unique user email and password. Firstly, he logs in using the login GUI. Figure B.1 illustrates user login GUI.

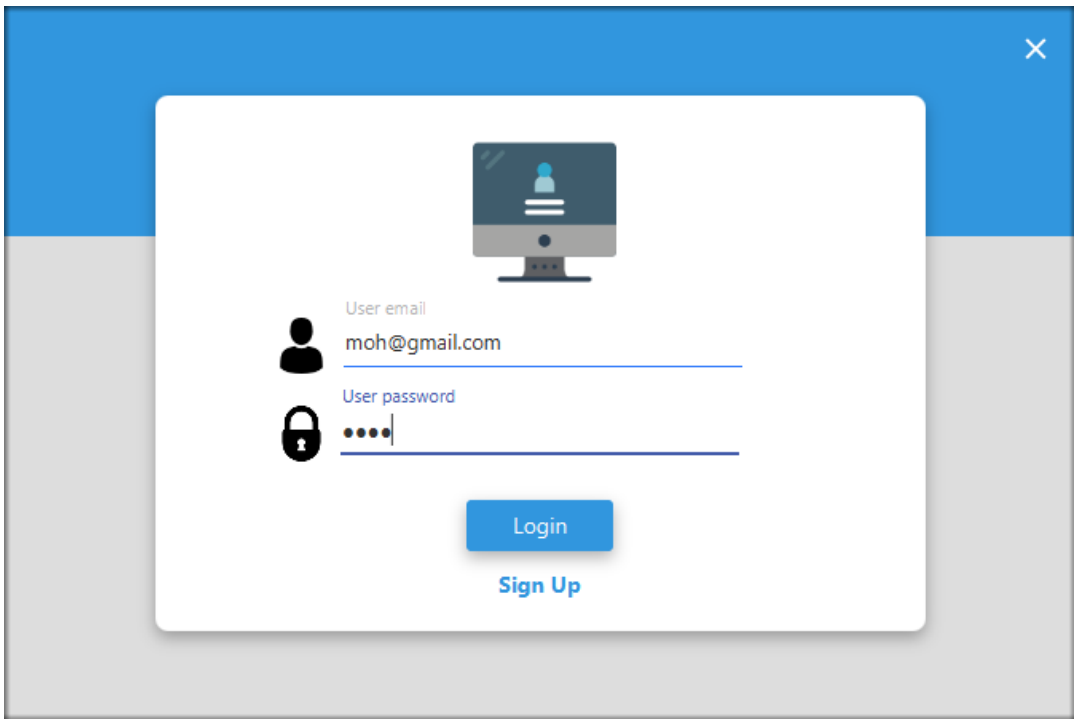


Figure B.1: Login GUI.

B.2 Main GUI

The login GUI checks the username and password. If they are correct, the application displays the main GUI with user details. Figure B.2 illustrates a screenshot of the main GUI. It contains three main options. The first one is Domain Management option that allows a user to manage his smart domains (smart-home, smart-office, smart-university, etc.). The second one is Agenda option that allows a user to manage his agenda. The third one is Constraint Management option that allows a user to define and manage his constraints in different smart domains.

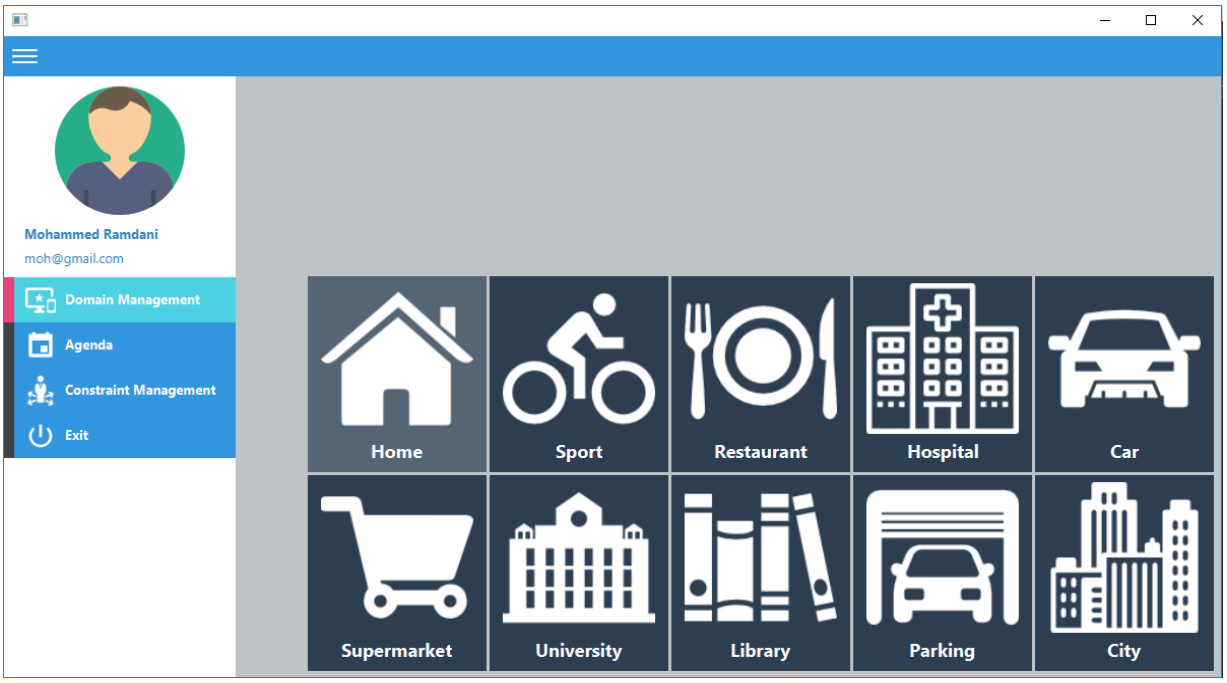


Figure B.2: Main GUI.

B.3 Home Domain GUI

When Mohammed is located in his smart home, the application can display him his smart home map with its all available devices as illustrated in figure B.3.

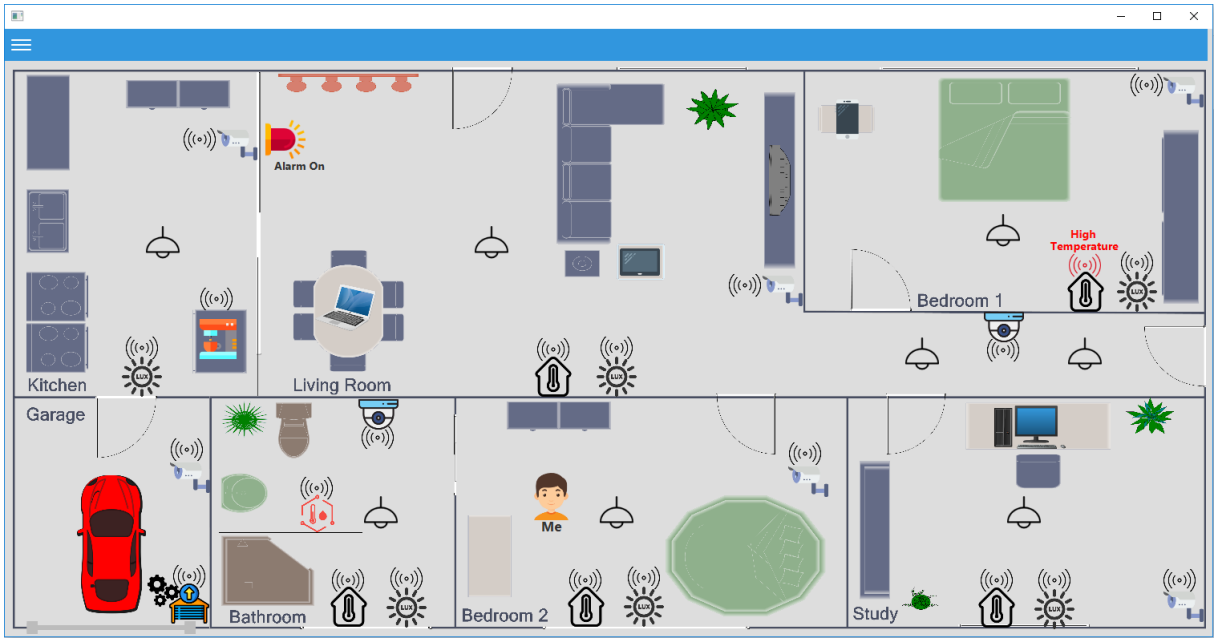


Figure B.3: A smart home schema with its embedded sensors and devices.

B.4 Situations definition GUI

Since Mohammed wants to exploit different available services in his smart home, he needs to define his constraint according to his preferences. Therefore, the application offers a situation definition GUI to manage and define his different constraints and situations as shown in figure B.4.

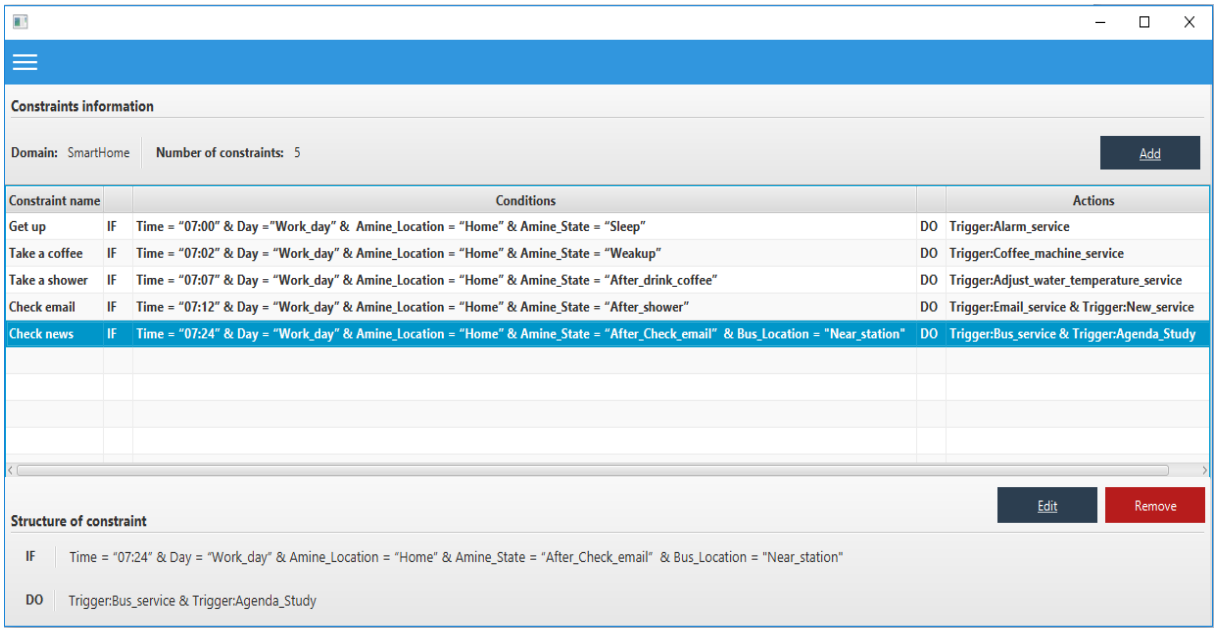


Figure B.4: Situation definition (Constraint Management) GUI.

B.5 Agenda GUI

While Mohammed can plan and define his daily activities using Constraint Management GUI, the application can display his parallel/sequential activities on the time axis. Figure B.5 illustrates scheduled activities of Mohammed in agenda GUI.

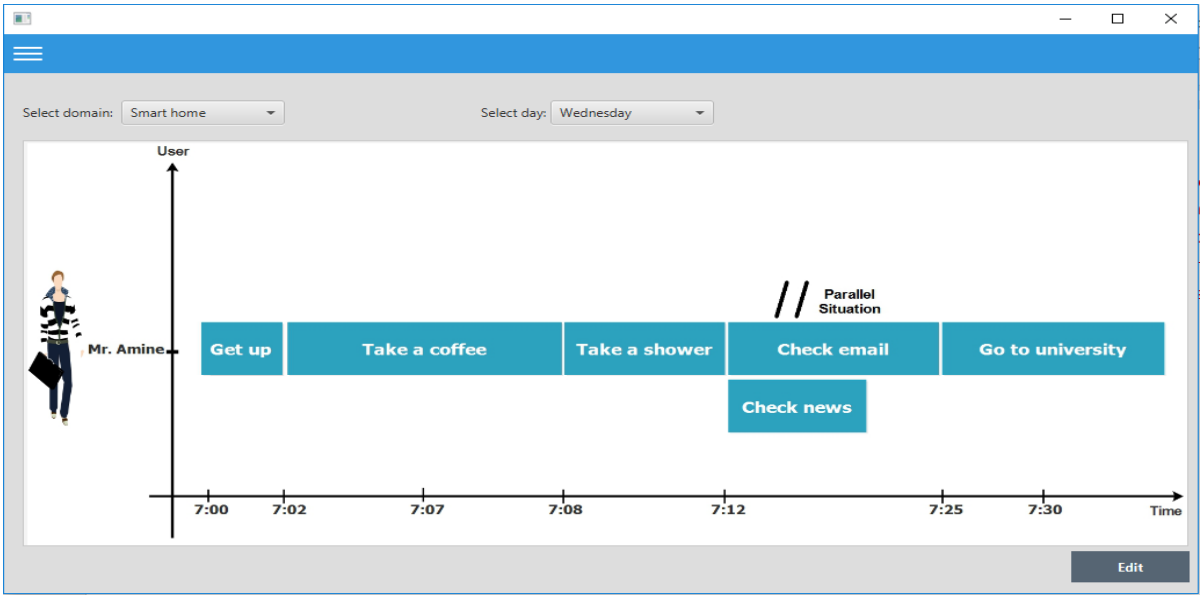


Figure B.5: Agenda GUI.

B.6 Automatic generation of reconfiguration script

After the definition of user's daily activities and situations, the reasoning engine filters relevant situation rules (recorded in the situation rules registry) according to current spatiotemporal contexts and current usage context. As the motivating scenario involves different locations, times and activities, the system cheeks continuously Mohammed's situations and triggers reconfigurations actions according to user's context and available devices in the smart home (see figure B.6).

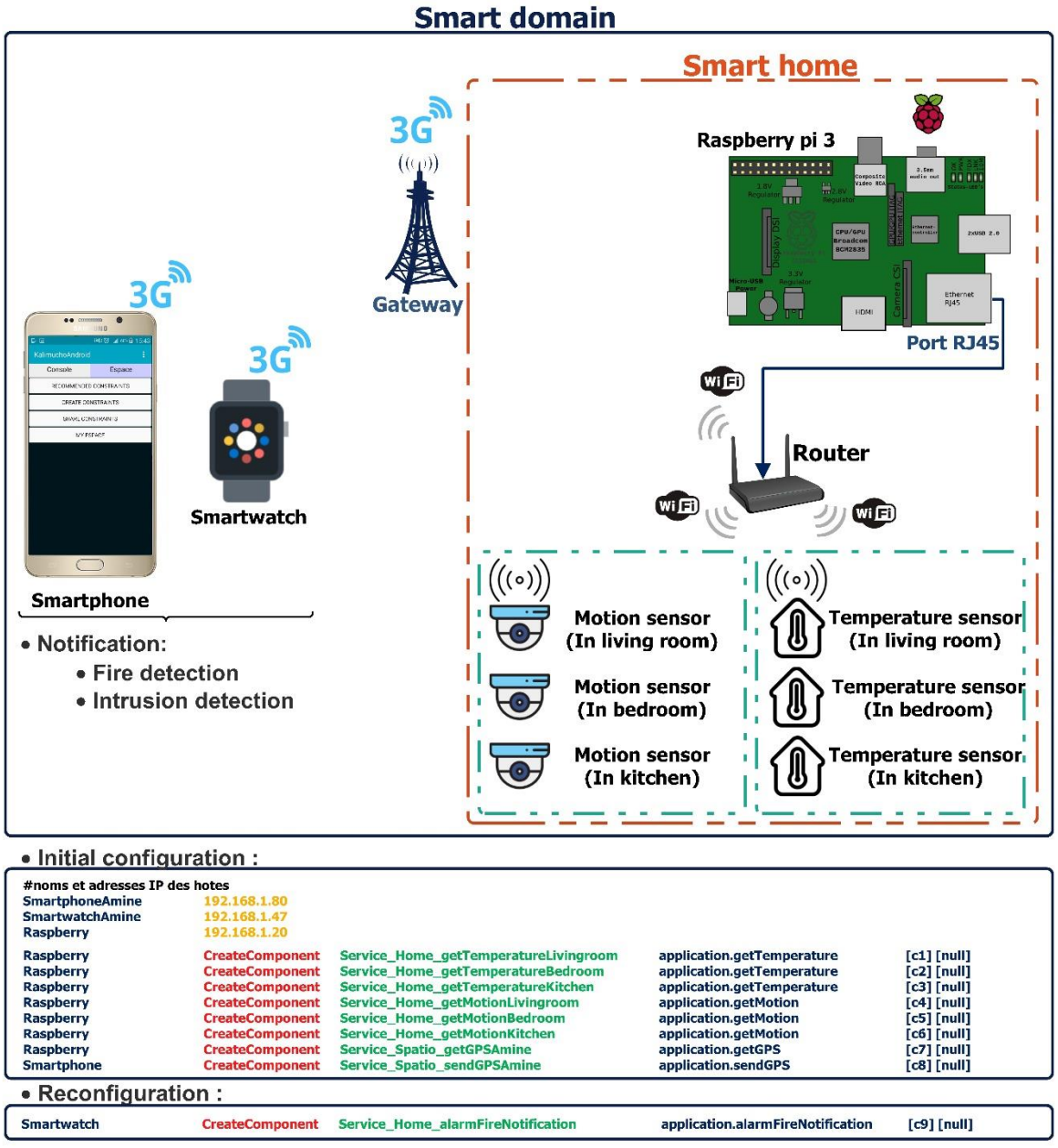


Figure B.6: The generation of initial script configuration.

Appendix C

Context and Situation Properties Analysis in Smart-* Domains

C.1 Concept extraction based on case studies

This section revealed four categories of smart case studies that are extremely important for extracting the common and specific concepts which represent context and composite situations properties in assisting users in various environments (smart assisted living, smart workspace, guided and control car and healthcare). These smart environments can react to users according to the daily-life events that occur in different domains (e.g., health, social and professional). The context properties can fall under any of the corresponding subclasses of the five key classes (Context, Situation, Smart objects, Domain, and Service). This will be followed by a comparison between the four categories of smart-* domains (home, health, car and office).

C.1.1 Smart-Home case study: Smart Assisted Living for People using Smart Devices

Looking first case study is about smart home that use heterogeneous sensors (e.g., temperature sensor, humidity sensor, luminosity sensor, movement sensor, camera, noisy sensor, kitchen sensor, etc.), assistive devices (e.g., smart-TV, laptop, temperature dashboard, alarm system dashboard, etc.) to capture home context information and activities of user in his smart home with daily situations such as open the curtain, adjust the lighting, turn on the air conditioner, open the garage, activate the alarm system, turn on the kitchen, etc. Home context information are analyzed for successful detection and generation of commands. These commands can be either triggered manually based on user decisions, or automatically based on home context information and user preferences. Moreover, the availability of smart devices has involved in many other fields including security (e.g., providing assistance when a potentially dangerous situation occurs such as fire detection, where the system can trigger automatically the fire extinguishing system), security (e.g., automated temperature adjustment), and energy saving (e.g., lights control). All the components of the smart home are summarized in Table C.1.

C.1.2 Smart-Home case study: Smart Assisted Living for People using Smart Devices

The next case study is about a proactive system that enable greatest fear for elderly people and afford healthy and safe lifestyle for patients or loved people. It is necessary to detect cancer, heart diseases, and diabetes as such urgent situations that speed up the care services for people. For this kind of monitoring there needs to be dedicated human body sensors (e.g., glucose sensor, body temperature sensor, etc.) or embedded sensors in pervasive environment (e.g., camera, movement detector, etc.). The system analysis this context information and identifies abnormal situations (e.g., high glucose level) to intervene automatically (e.g., inject insulin).

For instance, in case of critical anomalies (e.g., like heart-attacks), the system triggers an immediate response to the closest urban gateway and calls for immediate urgent medical attention where the victim's coordinates are found and submitted. After that, the system searches for the closest available ambulance and requests the nearest hospital for immediate preparation for hosting the patient. Therefore, pervasive environment capabilities can be used in healthcare applications to enhance intervention services while reducing the heavy load of nurses in hospitals. All the components of the smart healthcare monitoring system for diabetic people are summarized in Table C.1.

C.1.3 Smart-Car Case Study: Automated Guided and Traffic Control Smart Car

This case study about a pervasive monitoring system that spans several components including vehicle tracking, communications between vehicles (e.g., interchanging context information such as speed, destination, distance between them, etc.), coordination between vehicles and traffic control systems to avoid congestions. Therefore, an intelligent traffic control system is deployed to avoid crowded streets based on potential traffic congestion, and offer more efficient navigation systems for users. Looking outside of the immediate home environment, it is necessary that for any unexpected traffic congestion situation on the current route, the system must be notified automatically and must react dynamically to select a new better road towards the user's destination. The summary of system components based on the proposed model is shown in Table C.1.

C.1.4 Smart Office case study: Building a successful and efficient workspace Service

The final case study, smart office application can offer a great potential for the better goals and benefits. The application facilitates the management and distribution of tasks to workers and enhance employee safety. This is critical because, even if employee are totally stressed out, the system provides information on whether the employee is in good location of workspace, or whether tasks have performed, and whether the relative needs to take action. For instance, the system depends on RFID and GPS technologies to track and identify workplace employees. As workers are equipped with RFID readers, the system can track automatically the development of activities and indicates the remaining daily tasks of the employee. These are examples of the basic context information that we are summarizing in Table C.1.

Table C.1: The generation of initial script configuration.

Main Concepts		Domain Concepts	Smart-Home	Smart-Health	Smart-Office	Smart-Car
Context	**User**					
	Profile	User	√	√	√	√
		Patient	×	√	×	×
		Physics	×	×	×	×
		Worker	×	×	√	×
		Driver	×	×	×	√
	Preference	Implicit	√	√	√	√
	Host					
		Explicit	√	√	√	√
		Laptop	√	√	√	×
		Smart TV	√	×	√	×
		Tablet	√	√	√	√
		Mach. Coffee	√	√	√	√
		Smartphone	√	√	√	√
	QoS					
		Continuity	√	√	√	√
		Efficiency	√	√	√	√
		Safety	√	√	√	√
		Interoperability	√	√	√	√
		Scalability	√	√	√	√
	Environment					
	Location	Absolute	√	√	√	√
		Relative	√	√	√	√
	Time	Absolute	√	√	√	√
		Interval	√	√	√	√
	Condition	Noise	√	√	√	√
		Humidity	√	√	√	×
	Document					
	Multimedia	Video. Image	√	√	√	Text
	Modality	Click... Touch	√	√	√	Voice
	Social					
		Accounts	√	√	√	×
		Friends	√	√	√	V2V
User's Situation	**Simple Situation**					
	Normal	Wake Up	√	×	×	×
		Shower	√	×	√	×
		Drink Coffee	√	×	√	×
		Working	√	√	√	×
		Meeting	√	√	√	×
		Driving	×	×	×	√

	Urgent	Low Glucose	√	√	√	√
		Fire	√	×	√	√
		Intrusion	√	×	√	√
		Car Accident	×	×	×	√
	Composite Situation					
	Normal	Wake up-shower	√	×	×	×
		Shower-Drink	√	×	√	×
		Drink Email	√	×	√	×
	Urgent	ↄLow Glucose	×	√	×	×
		ↄHigh Glucose	×	√	×	×
		Glucose -Heart	×	√	×	×
Service	**Service Listener**					
		User Monitor.	√	√	√	√
		Health Monitor.	×	√	×	×
		Activity Track.	×	√	×	×
		Cardiac Implant	×	√	×	×
		Location	√	√	√	√
		Time Service	√	√	√	√
		Care Message	×	√	×	×
		Remote Ctrl	×	√	√	√
		Road Ctrl	×	×	×	√
	Action Service					
		Wake Up	√	×	×	×
		Shower	√	×	×	×
		Coffee Mach.	√	×	√	×
		Message	√	√	√	√
		Intrusion	√	×	√	×
		Fire Fighting	√	×	√	×
		Inject Insulin	×	√	×	×
		Risk of Death	×	√	×	×
		Remote Diag.	×	√	×	×
		Car Ctrl	×	×	×	√
Smart objects	**Sensor**					
		Temperature	√	×	√	×
		Light	√	×	√	×
		Motion	√	×	√	×
		IP Camera	√	√	√	√
		State Of Door	√	√	√	√
		Coffee Mach.	√	×	×	×
		Humidity	√	×	×	×
		Water Temp.	√	×	×	×
		Garage	√	×	×	×
		Body Temp.	√	√	√	√
		Heart Rate	√	√	√	√
		Insulin Level	√	√	√	√
		Location	√	√	√	√
		Time Sensor	√	√	√	√
		Traffic	×	×	×	√
		Video Recorder	√	×	√	√
		RFID Sensor	×	×	×	√
	Actuator					
		Light Switch	√	×	√	×
		Door Locking	√	×	√	×
		Smart Mirror	√	×	√	×
		Inject Insulin	×	√	×	×
		Fire Alarm	√	×	√	√
		Intrusion Alarm	√	×	√	√
		Alert	√	√	√	√
		Light Switch	√	×	√	×
Domain	User's Domain					
		Security	√	√	√	√
		Home	√	×	×	×
		Health	√	√	√	√
		Office	×	×	√	×

Thèse : Approche sémantique de génération automatique d'applications agiles en environnement pervasif.

Doctorant : Abderrahim LAKEHAL

Directeur : Dr. Adel ALTI

Co-Directeur : Dr. Philippe ROOSE

Abstract. Currently, pervasive computing incarnates a new era in assisting users during their daily activities everywhere and anytime. Although there are many mobile applications that may handle various context data, they neglect the real-time factor for composite/urgent situation management and identification. To ensure managing and identifying situations efficiently, we have brought many contributions in pervasive computing systems. Our main contribution is the proposition of a new ontology model called Multi-OCSM (Multi-layered Ontology-based Composite Situation Model) representing composite situations pertaining to smart environments using specific composition operators. The second contribution is the proposal of a novel flexible, modular, and hierarchical loosely coupled framework that can be employed for parallel composite situations management and reasoning. In addition, it ensures the automatic reconfiguration of distributed pervasive applications by providing relevant action services regarding context changes. The last contribution refers to a context-aware recommendation system for situation rules enrichment and adaptation.

Résumé. Actuellement, l'informatique pervasif incarne une nouvelle ère pour aider les utilisateurs dans leurs activités quotidiennes partout et à tout moment. Bien qu'il existe de nombreuses applications mobiles qui peuvent gérer diverses données de contexte, elles négligent le facteur temps réel pour la gestion et l'identification des situations composites / urgentes. Pour assurer une gestion et une identification efficaces des situations, nous avons apporté de nombreuses contributions aux systèmes informatiques pervasif. Notre principale contribution est la proposition d'un nouveau modèle d'ontologie appelé Multi-OCSM (Multi-layered Ontology-based Composite Situation Model) représentant des situations composites relatives aux environnements intelligents en utilisant des opérateurs de composition spécifiques. La deuxième contribution est la proposition d'un nouveau cadre flexible, modulaire et hiérarchique faiblement couplé qui peut être utilisé pour la gestion et le raisonnement parallèles de situations composites. De plus, il assure la reconfiguration automatique des applications pervasif distribuées en fournissant des services d'action pertinents concernant les changements de contexte. La dernière contribution fait référence à un système de recommandation sensible au contexte pour l'enrichissement et l'adaptation des règles de situation.

ملخص. في الوقت الحالي، تجسد الحوسبة المنتشرة حقبة جديدة في مساعدة المستخدمين أثناء أنشطتهم اليومية في كل مكان وزمان. على الرغم من وجود العديد من تطبيقات الهاتف المحمول التي قد تتعامل مع بيانات السياق المختلفة، إلا أنها تتجاهل عامل الوقت الحقيقي لإدارة الحالة المركبة/العاجلة وتحديدها. لضمان إدارة وتحديد المواقف بكفاءة، قدمنا العديد من المساهمات في أنظمة الحوسبة المنتشرة. مساهمتنا الرئيسية هي اقتراح نموذج جديد للأنطولوجيا يسمى Multi-OCSM (نموذج الموقف المركب متعدد الطبقات القائم على علم الوجود) الذي يمثل مواقف مركبة تتعلق بالبيئات الذكية باستخدام عوامل تكوين محددة. المساهمة الثانية هي اقتراح إطار جديد مرن، نمطي، وتسلسل هرمي ضعيف الاقتران يمكن استخدامه لتحديد واستنتاج المواقف المركبة المتوازية. بالإضافة إلى ذلك، فإنه يضمن إعادة التكوين التلقائي للتطبيقات المنتشرة الموزعة من خلال توفير خدمات متنوعة ذات الصلة فيما يتعلق بتغييرات السياق. المساهمة الأخيرة تمثل نظام توصية مدرك للسياق لإثراء قواعد الموقف وتكييفها.