A Decentralized Agent-Based Semantic Service Control and Self-Adaptation in Smart Health Mobile Applications

Adel Alti, Abderrahim Lakehal, Philippe Roose

Research Article

A Decentralized Agent-Based Semantic Service Control and Self-Adaptation in Smart Health Mobile Applications

Adel ALTI [1,*], Abderrahim LAKEHAL[2], Philippe ROOSE[3]

a.alti@qu.edu, lakehal.abderrahim@gmail.com, philippe.roose@iutbayonne.univ-pau.fr

[1,*] Department of Management Information Systems, College of Business & Economics Qassim University, P.O. Box 6633, Buraidah, 51452, KSA

[2] LRSD, University Ferhat Abbas Sétif 1, Faculty of Sciences, Computer Science Department, P.O. Box 19000 Sétif, Algeria.

[3] LIUPPA, IUT of Bayonne, University of Pau and Adour Countries, Anglet, P.O. Box 64000, France.

Received: date; Accepted: date; Published: date

Abstract: In this paper, we propose a decentralized agent-based Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR) to solve the problem of context-aware service selection and deployment for distributed healthcare mobile applications, where functional and non-functional users' needs are expressed using domain-independent ontology. The main objective of this approach is to provide optimal personalized services as fast as possible. Indeed, a new ontology for semantic description and parallel management of the contextual services is presented. At first, a new dominance operator-based constraints violation degree is used to reduce the search space. Then, optimal personalized paths that meet user's needs, preferences and devices' availability are discovered and selected in a smaller search space more effectively and efficiently using dynamic services selection algorithm, based on new ASSACR multi-agent strategies. To validate the performance and efficiency of the proposed approach, we compared the proposed approach with existing multi-agent strategies of the diabetic follow-up use cases. Experimental results show that the proposed approach effectively addressed both the semantic service representation and agent aspects in terms of performance and quality service selection.

Keywords: IoT, Multi-Agent, Smart Health, QoS, Ontology Model, Mobile Application.

1. Introduction

The Internet of Things (IoT) has emerged as the dominant computing paradigm that enables ubiquitous connectivity between different smart objects [1]. IoT aims to support users anytime, anywhere and under any conditions. It leverages heterogeneous smart objects and IoT devices 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 activities and situations are increasing in daily life, thus a unified semantic model is required to easily identify situations whatever their natures, their expressed languages and their context sources. Ontology is a popular formal method for modeling different concepts of ubiquitous computing, which represents physical concepts (*i.e. smart object, device*) and abstract concepts (*i.e. context, situation, service*) and analyzing their behavior regarding system performance. Ontologies are simple, non-ambiguous, and efficient [32-33].

Existing ontological models [4 - 7] offer appropriate capabilities, including the semantic description of context and user's needs. However, the existing ontologies do not deal with various situations complexity, nor they do 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 get low priority. Nevertheless, the existing ontologies neglect the situation priority concept under several user's

locations (*moving across different smart domains*) during the run time. This may make reconfiguration applications suffer from reacting rapidly to urgent situations. This issue is noticeable when an urgent health situation (*e.g., heart-attack health situation*) needs to be firstly handled during a specific deadline which defined by an expert. We must ensure continuous monitoring and quick processing time regardless of user location and context changes in order to deal with urgent situations quickly. The semantic contextual description of smart services is an important aspect for managing a large number of heterogeneous smart objects, managing user-centric situations and offering appropriate services according to users' needs.

Several approaches and platforms [4-5; 8; 10-20] have been proposed for reconfigurable IoT systems in the field of smart health. The three main categories of approaches are semantic-based platforms [4-5;8;10-13], rules-based approaches [14], and metaheuristic-based approaches [**15-20**]. For example, Naqvi *et al.* [4] have proposed a semantic-based context-driven approach with QoC (Quality of Context) for mobile Cloud computing. This approach includes rich QoC constraints but lacks for efficient and consistent agent services cooperation. Rhayem *et al.* [14] focus on self-reconfiguring by changing the unavailable IoT sensor with another available equivalent, which is impractical due to the strong and multiple connections between the heterogeneous physical things that complicate the management of the system. Current approaches have poor scalability to identify the best service's candidate in terms of QoS from a large set of services. Moreover, the number of heterogeneous services and the QoS of service providers are not static and may change continuously over time. This increases the time of matching the semantic properties of different context environments. Faced with the evolving needs of users as well as the increase of services offering different QoS, the services selection and deployment requires the development of advanced decentralized semantic-based approaches related to parallel and/or distributed agents allowing more convenience, efficiency, and optimal deployment cost of application.

To address the above challenges, we extend the previous framework presented in [2;3] with a new layer called Agent-based Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR). Based on Context-aware Quality Smart Health ONTOlogy (CxQSHealthOnto) [3], this work adapts parallel settings of the quality services discovery agents to optimize performance in terms of execution time and deployment cost. Our main motivation is to propose a modular and hierarchical framework with strong scalability and flexibility. To achieve this flexibility, several kinds of agents are engaged in dynamic cooperative strategy to meet the patient's functional and QoS needs. The contributions of this work are as follows:

- A novel Context-aware Quality Smart Health ONTOlogy levels (CxQSHealthOnto) [3] which is defined on high semantic levels to help system developers manage heterogeneous services for smart-health domains supported by mobile Cloud at design and runtime. This ontology model allows the classification of health services with different QoS, contexts, locations and categories.
- A dominance relationship with a new constraints' violation degree to user QoS constraints. It allows semantic filtering of the quality of health services.
- To determine the optimal service, we develop a decentralized model based on ASSACR and CxQSHealthOnto ontology. It provides relevant health services according to the user's needs context and QoS preferences using situation reasoning ontology capabilities. Therefore, the proposed method uses a parallel discovery of the relevant services of distributed mobile devices from those close to the user's location. So it reduces processing load and network communication cost with the best QoS.
- To validate the efficiency of the proposed semantic-based optimization technique, IoT healthcare distributed mobile application is used in the experiment. The experimental results show that the proposed approach is superior to the current simulators [9] and provides better quality solutions as well as optimizes the service deployment cost in connected mobile devices.

The rest of the paper is organized as follows. Section 2 describes the current related works. Section 3 presents the ontology model. The proposed framework is presented in detail in Section 4.

Experimental results are given in Section 5. Finally, Section 6 concludes this work with some future research directions.

2. Related Work

Improving and managing health services selection has gained increasing attention in IoT environments and mobile applications. Despite the benefits of these works towards the quality services management concerns, there are research works that do not take into account context assumptions such as unhandled parallel semantic service discovery and selection from large services set as complexity increases with dynamic updates of QoS needs of users. We distinguish two approaches: semantic-based context-aware service selection techniques and heuristic/metaheuristic-based service selection techniques. Our goal is to overcome the limitations of existing recent relevant approaches and come out with a new decentralized system that manages reconfigurable mobile applications in optimal delay based on coordination and control of multiple agents-based ASSACR.

2.1 Semantic-Based Context-Aware Service Selection Techniques

Naqvi *et al.* [4] proposed a semantic-based context-driven framework in mobile Cloud computing. This ontology provides context information management at a semantic level of the shared domain. This work provides contextual monitoring on the mobile device to ensure reconfiguration decisions. Based on these decisions, this framework achieves service customization according to current user's situations and scheduled activities for deployments strategies. However, this context-driven management model lacks a parallel context processing strategy that can split context information into parallel sub-contexts (*e.g., defining "no one home" as an abstract complex context by aggregating all output values from motion sensors deployed in each room*).

To provide such support, Bayer *et al.* [5] developed an efficient and context-aware deployment schema of IoT Cloud environment that was mitigated by the user mobility and workload. This approach is provided with reduced application latency but the required maintenance cost is very high. In our work, we design and implement a decentralized agent-based autonomic semantic context-aware platform to predict smart service breaks (*e.g. limited life battery, user's mobility, workload*) through equivalent semantic paths and determine the necessary adaptations, so that the user has a long-life application.

Esposito *et al.* [8] proposed a smart mobile self-configuring and semantic-enabled context-aware architecture to rapid prototyping of personal health monitoring. Moreover, this approach was provided with health situation reasoning facilities through ontology fuzzy-based approach and smart emergency alert system. This approach requires a large time is considered as the major drawback of this approach. In addition, this work lacks of optimal services selection from large services set.

To provide durable pervasive healthcare systems with continuous context data monitoring and reasoning, Hameur Laine *et al.* [10] proposed a new hybrid model based on generic observer/controller paradigms and ontology/rule-based inference approach. The experimental analysis was conducted by evaluating inference rules, adaptation time per service and the results revealed that the efficiency of this approach was very high with a very low complexity rate. However, this work uses different and heterogeneous entities and completely lacking of any semantic techniques for services discovery to meet the application long-life QoS.

To address this problem, Angsuchotmetee *et al.* [11] presented Multimedia Semantic Sensor Network Ontology (MSSN-Onto), to model and handle complex events in real-time. The approach uses a pipeline-based technique to detect complex events by observing and combining multiple sensors. It provides distributed logic-based detection and continuous monitoring techniques to detect events in multimedia sensor networks. The proposed system can achieve high detection accuracy with a short processing time. However, it still has limitations as it does not detect critical situations (*i.e., heart attacks situations*). Therefore, the reconfiguration aspect of distributed

applications and semantic service discovery and selection from large services set are missing in this work.

Bhunia *et al.* [12] implemented a smart health monitoring system for measuring some physical properties of the human body including several sensors. This approach employed advanced fuzzy context-aware techniques to analyze various symptoms and discovering a specific disease. The system activates few sensors whereas sensors capture some basic parameters of a human body. The experimental analysis was conducted and the results revealed the context data was highly preserved using the proposed approach. The execution time and communication overhead are complex which is considered as the main drawback of this approach.

Christopoulou *et al.* [28] designed a virtual health Mentor (vhMentor) to automate the ubiquitous patient monitoring at run time. The approach is based on patient healthcare information ontology supported mobile agents where agents need to provide well-structured patient information to each other at run time. Therefore, it provides decision-making and reasoning services through autonomous and intelligent agents. The proposed work successes to increase the healthcare quality compared to previous rules-based approaches but neglects the context in the monitoring process.

2.2 Heuristic/Metaheuristic-Based Service Selection and Deployment Techniques

Name *et al.* [14] have proposed a new hybrid fog and cloud context-aware deployment environment to migrate services from fog nodes to Cloud. During the migration process, distributed optimization strategies are used to aggregate different context information related to bandwidth, network connectivity as well as available storage sizes. This work did not examine in detail their impact on QoS degradation and energy consumption. It will be necessary to consider the decentralized approaches related to the device-fog-cloud allowing the total (or partial) re-allocation, permanent or temporary of some services in order to achieve an optimal durability of applications.

Most delegate optimization techniques include Genetic Algorithms (GA) [15], Particle Swarm Optimization (PSO) [16], Ant Colony Optimization (ACO) [17] and Grey Wolf Optimization (GWO) [18]. These optimizers are most widely used for optimizing services qualities in the field of smart health. Different GA, PSO, ACO and GWO algorithms have been successfully applied to solve different problems in the last years, but PSO, GA, and GWO perform better in a variety of applications, including health service design. However, these optimizers still need more improvement in terms of computational complexity and solution quality in a wide variety of mobile services and handle continuous QoS change.

Valera *et al.* [19] proposed *DRACeo,* a simulator for evaluating and managing any type of network and heterogeneous devices to run distributed mobile applications based on services or microservices. *DRACeo* provides reconfiguration facilities (add/remove/update) to enable developers understanding the efficiency of deployment techniques. The evaluated variables include execution time, amount of energy spent, the current quality of service, and the amount of transmitted data. The simulator enables the user to define their own QoS requirements and energy consumption for developers to ensure monitoring of these requirements. The proposed simulator successes to test various management mechanisms: *centralized* and *non-centralized* compared to previous optimization approaches but neglects the best distribution behavior in any network of heterogeneous devices. However, this work does not provide equivalent configurations with different QoS to enhance the effectiveness and reduce deployment cost, where a new dominance relationship, which is based on constraint dominance between component-device, is proposed to compare configurations QoS values that are expressed as constraint violation degrees.

Simon *et al.* [20] presented a mathematical model with Cloud Quantum Computing for deploying large-size health applications. The performance measures utilized in this approach were computational time, storage overhead and computation overhead. The analysis showed great flexibility but the approach failed to consider small-scale quantum is considered as the main shortcoming of this approach.

2.3 Objectives and Motivations

We have analyzed and compared existing optimization models and semantic-based systems according to four main criteria: (1) resource capacity reasoning, (2) services grouped by location, type, and QoS, (3) (4) scalability, and (4) flexibility in order to increase the contribution of this work. Resource capacity reasoning refers to evaluate how the system behaves according to the number of available computing resources and growing number of events and urgent situations. This analysis allows predicting whether the system will satisfy users' needs efficiently. Services and sensors grouped by location, type, and QoS refer to how the model represents different sets of sensors. In our work, we analyze the system's behavior by changing the number of sensors and their frequencies. In our model, these locations can be seen as rooms in a hospital. This feature aims to make the model more realistic because, depending on the location and type of sensors, the data generated can be different. Scalability refers to how the system effectively manages the increasing number of sensors, the big volume of context data, continuous sensor events monitoring, and user's mobility. In our work, we analyze the system's behavior by changing the number of sensors and their frequencies. Flexibility refers to how the system represents a different set of sensors, contexts, events and situations of multi-smart domains, also that how it can respond continuously to users' needs regardless of context. The limitations and the strength of related works are summarized in Table 1. We have identified common limitations, particularly for reacting rapidly to situations under several moving locations. Further, to be independent of the different services providers and services technologies, we include the selection and deployment process in the service semantic description. A semantic model is used to describe services, user's needs and contexts to facilitate matching between them. Besides, the process of multi-objective Cloud service composition is optimized using multi-agent aspects to improve the performance in terms of execution time.

Table 1: Related work comparative study

	Related Works	Application Context	Resource Capacity Reasoning	Services Grouped by location	Scalability	Flexibility
Semantic-based	Naqvi *et al.* [4]	*Smart Health*	*Partial*	✗	*Low*	*Low*
	Bayer *et al.* [5]	*Smart Domain*	*Partial*	✗	*Low*	*Medium*
	Esposito *et al.* [8]	*Smart Health*	✗	✗	*Low*	*Medium*
	Hameur *et al.* [10]	*Smart Health*	*Partial*	✗	*Low*	*Medium*
	Angsuchotmetee et *al.* [11]	*Smart Building*	*Partial*	✗	*Low*	*Medium*
	Bhunia *et al.* [12]	*Smart Health*	✗	✗	*Low*	*Low*
Rule based	Name *et al.* [14]	*Smart Building*	✗	*Partial*	*Low*	*Low*
Metaheuristic -based	GA [15]	*Smart Health*	*Partial*	✗	*Low*	*Low*
	PSO [16]	*Smart Health*	*Partial*	✗	*Medium*	*Low*
	ACO [17]	*Smart Health*	*Partial*	✗	*Medium*	*Low*
	GWO [18]	*Smart Health*	*Partial*	✗	*Low*	*Low*
	Valera *et al.* [19]	*Smart Domain*	*Yes*	*Partial*	*Medium*	*Medium*
	Simon *et al.* [20]	*Smart Health*	✗	✗	*Medium*	*Low*
	This Work	***Smart Health***	***Yes***	***Yes***	***Yes***	***Yes***

3. CxQSHealthOnto ontology Model

3.1 Description of CxQSHealthOnto

We introduce a novel ontology called CxQSHealthOnto which stands for Context-aware Quality Smart Health Service ontology domains supported by mobile Cloud at design and runtime. The main purpose of CxQSHealthOnto is to provide support for efficient management of the end-to-end mobile distributed application building process as well as Cloud services selection approach dedicated to context-aware mobile applications. The novelty of the proposed ontology compared to existing ontologies is to implement novel generic semantic role-based smart services in hierarchical abstraction levels for better management facilities (*context health data management between heterogeneous health services, easy selection of new health services based roles regardless of its service provider through semantic services information description, a semantic description of health services, and to adapt and customize a selection service according to user's constraints and preferences*). Thus, it facilitates the semantic interoperability of heterogeneous services provided by different service providers and interprets qualities of services. It allows semantic filtering on quality of Cloud services. It enables semantic reasoning enriched by SWRL rules and facilitates users' profiles modeling, discovery and personalization of health services. The ontology presented in Figure 1 focuses on satisfying the user's functional needs and optimizing the deployment process in terms of QoS according to user's constraints and preferences. The ontology is built up of five key concepts:

- **User**: has a name, unique location, functional needs, constraints and contextual preferences that consist of simple or complex expressions. A user can specify their functional requirements, constraints, and preferences through GUI.
  - **Functional needs**: Each functional need consists of simple or composite logic expressions combined by 'AND' operator.
  - **Semantic constraints:** the user specifies more easily its QoS constraints in a graphical form. The constraint can be defined as value for each QoS attribute (*e.g. response time: fast, price: cheaper, etc.*). The system converts constraints expressed with qualitative terms to semantic values. Each semantic value corresponds to quantitative intervals. The semantic constraints may be classified into three logic values:
    - ✓ *High*: normalized QoS values in$[\mathrm{qmin}_i^{\mathrm{High}},\mathrm{qmax}_i^{\mathrm{High}}]$.
    - ✓ *Medium* : normalized QoS values in $[\mathrm{qmin}_i^{\mathrm{Medium}},\mathrm{qmax}_i^{\mathrm{Medium}}[$.
    - ✓ *Low*: normalized QoS values in $[\mathrm{qmin}_i^{\mathrm{Low}},\mathrm{qmax}_i^{\mathrm{Low}}[$.

    The ith explicit user constraint of the quality attribute q_i (denoted as Cst_i) is converted to numerical value as follows:

$$\mathrm{Cst_i} = \mathrm{qmin_i^{semantic-value}} \quad (1)$$

  - **Preferences**: The user specifies QoS preferences in an ordered list (from the most important-significant preference to the least one). For example, a user-preferred response time, price (1- response time, 2-price), while another prefers the opposite (*1-price, 2-response time*). Currently, each quality criterion is valued by a user according to its importance. It is an integer number that indicates the importance of QoS constraint (*0 for none, 1 for low, 2 for medium and 3 for high*).
- **Context:** the user plays an important role in the selection of quality services. The applications within the environment must adapt to the users' needs, not the other way around. Context is any information collected from the *environment*, *service execution* and *constraints*.
  - **Environment Context** describes contextual information such as available network connections (*WIFI, GSM, etc.),* time, date, and geographic location constraints. Connections between mobile devices (*PDAs, Smartphones*) with limited bandwidth. The location, date and time represent a specific geographic location where the service is to be performed
  - **Service execution context*:*** CPU speed, battery power, memory size.
  - **Constraint** allows us to describe a set of constraints using logical expressions.

- **Service:** Ontology provides unified health services such as monitoring, emergency and recommendation (*diets of a patient, medication, security….etc.*). Each service is defined as a set of QoS parameters. Services are organized in class hierarchies (*service category, service role, input /output parameters*). Each service depends on the state of the patient, location and time. In this work of discovery of quality services, ontology and QoS-based filtering with knowledge graph can play a crucial role to implement context-aware services selection that enhances the relevancy of recommended services. The smart-health domain ontology covers the domain of medical services. It includes many aspects such as patient care, clinical decisions, diagnostics, and mobile devices that help patients. In our mission to model the smart-health domain, our choice is fixed on certain health services such as (1) – assessment of Anemia, (2) – assessment of Seasonal flu, (3) – assessment of food poisoning, (4) – assessment of hyperglycemia type1 and (5) – assessment of hyperthyroidism.
- **Quality of Service (QoS):** The quality of any service is defined as a set of QoS and metadata parameters. QoS parameters concern not only the services but also the *trust level* and *throughput* where the execution is taking place. There are two categories in QoS attributes: *positive* and *negative* [**29**]. Both categories are configured using the same graphical tool. If a high QoS value is desirable then it is called *positive attribute*, and if a low QoS value is desirable then it is called as *negative attribute*. The classification of QoS attributes is shown in Table 2.

Table 2: Positive and negative attributes of QoS.

QoS Attributes		Description
Positive	**Availability**	The qualitative aspect of whether the service is present or ready for immediate use.
	Reliability	It represents the degree of ability to maintain the service.
	Security	It represents the security degree of a cloud service. It aims to ensure the following objectives: confidentiality, authentication and integrity.
Negative	**Execution Time**	The time needed to process and complete a service request.
	Price	It measures the cost of invoking a service for each request.

The QoS values range widely, we need to normalize all attribute values to fetch them in the same range [0, 1]. So that no attribute dominates other attributes. Since, we have two categories of attributes that need to be managed them individually while being normalized.

The QoS of a candidate service s_i is represented as a vector $QoS\ (s_i) = \langle q_{i,1}, q_{i,2}, \cdots q_{i,n} \rangle$ where n represents the number of QoS attributes required by the user and $q_{i,j}$ represents the value of the j^{th} QoS attribute ($1 \leq j \leq n$) of service s_i .

The formula for normalization of positive attributes is as follows:

$$q^{+}{}_{(i,j)} = \begin{cases} \frac{q_{(i,j)} - q_j^{min}}{q_j^{max} - q_j^{min}} & , \quad If\ q_j^{max} - q_j^{min} \neq 0 \\ 1 & , \quad Otherwise \end{cases} \tag{2}$$

The formula for normalization of negative attributes is as follows:

$$q^{-}{}_{(i,j)} = \begin{cases} \frac{q_j^{max} - q_{(i,j)}}{q_j^{max} - q_j^{min}} & , \quad If\ q_j^{max} - q_j^{min} \neq 0 \\ 1 & , \quad Otherwise \end{cases} \tag{3}$$

where q_j^{max} and q_j^{min} denote respectively the highest and lowest values of the j^{th} QoS attribute for the service. The normalized values will be one if they are the same.

- **Semantic Service Path:** This class is made up of a set of equivalent services (*parallel, sequential, alternated, loop*) providing the same functionalities with different QoS. In a sequential structure, all tasks are processed in sequential order. In a parallel structure, all parallel tasks are executed to proceed to the subsequent tasks. In a conditional structure, at least one task is executed among multiple branches to proceed further. In a loop structure, a task executes multiple times.

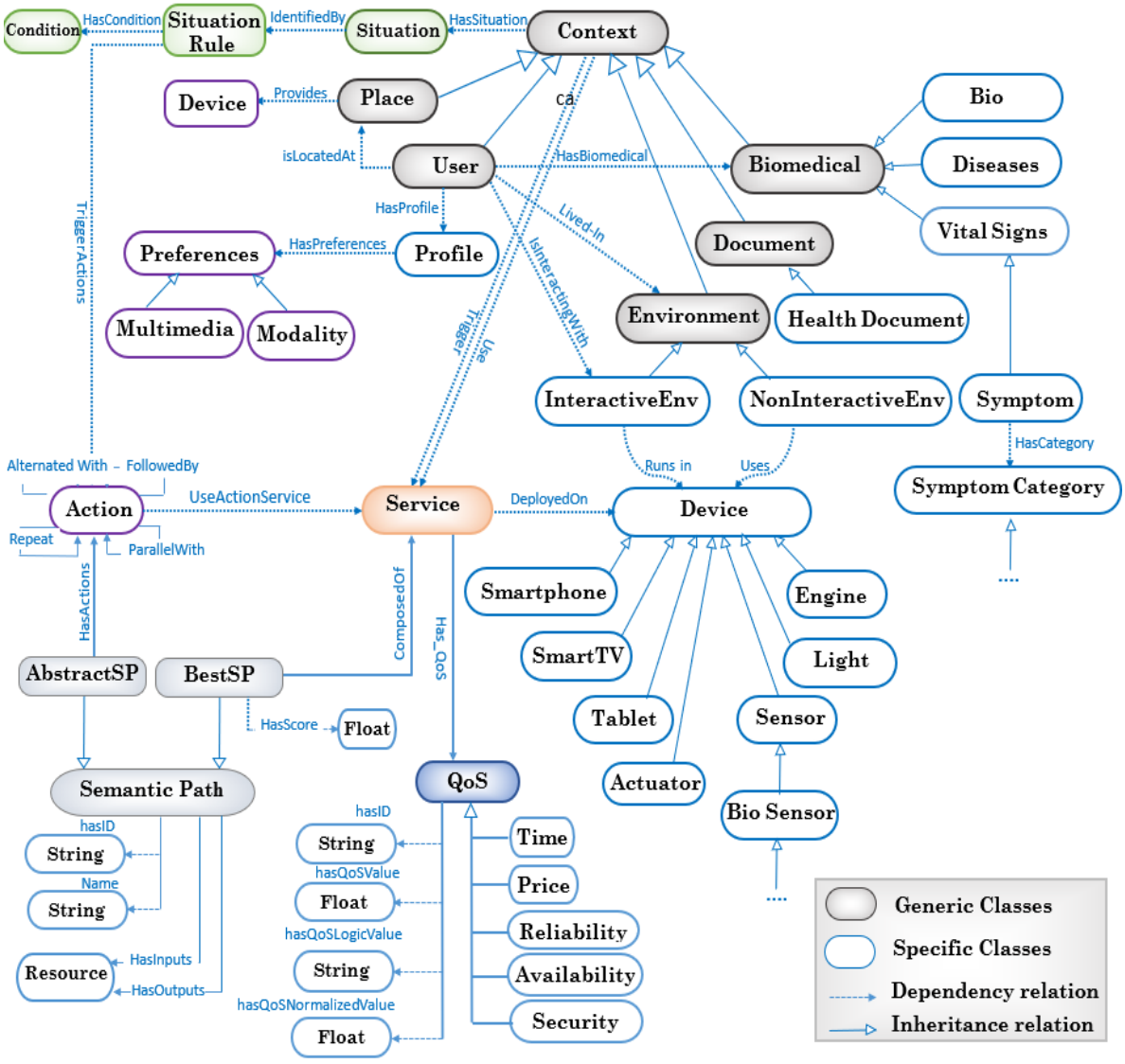

Figure 1. CxQSHealthOnto Ontology Model

3.2 Consistent rules of CxQSHealthOnto

Verifying the consistency of CxQSHealthOnto instances is an essential part of our validation, especially the consistency structure of composite situations and QoS constraints (*with various context properties*) while providing more formal semantics. To deal with it, we use the Description Logic (DL) [30] to represent CxQSHealthOnto in a formal and structured way. The CxQSHealthOnto expressed in a DL is constituted by two components, traditionally called TBox and ABox [31].

— **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: $C1 \sqsubseteq C2$, where $C1$ and $C2$ 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 reasoning mechanism is fundamentally based on three parts: 1) – the source of knowledge CxQSHealthOnto, 2) – the interaction between the ontology and the system through the SWRL query language and 3) – the reasoning engine based on parallel semantic-based composite situation identification framework. Table 3 presents DL examples of our ontology.

Table 3: Example TBox and ABox in our ontology.

TBox	ABox
$Context \subseteq Th\ ing$	
$UserContext \subseteq Context$	$UserContext(User\_A)$
$\geq HasID \subseteq String$	$HasID\ (User\_A, U\_1)$
$\geq HasName \subseteq String$	$HasName(User\_A, Adem)$
$\geq HasAge \subseteq Int$	$HasAge\ (User\_A, 25)$
$\geq HasSituation \subseteq Situation$	$HasSituation\ (User\_A, Situation\_S1)$
Service $\subseteq Th\ ing$	$Service\ (Service\_Inject\_Insulin)$
$\geq Has$QoS $\sqsubseteq$ QoS	HasQoS $(Service\_Inject\_Insulin$, QoS_1)
$Situation \subseteq Th\ ing$	$Situation\ (Situation\_S1)$
$\geq HasID \subseteq String$	$HasID\ (Situation\_S1$, ID_1)
$\geq HasName \subseteq String$	$HasName\ (Situation\_S1, High\ \_Glucose)$
$\geq HasCondition \subseteq Condition$	$HasCondition(Situation\_S1, Condition\_High\ \_Glucose)$
$\geq HasService \subseteq$ Service	$HasService\ (Situation\_S1, Service\_Inject\_Insulin)$
$\geq$ QoS $\subseteq Thing$	QoS (QoS_1)
$\geq$ HasName $\subseteq String$	$HasName$ (QoS_1,Price)
$\geq$ HasQoSNormalizedValue$\subseteq String$	$HasQoSNormalizedValue$ (QoS_1, 0.78)

4. Proposed Platform

Users (*patients, physics, nursing*) are expecting optimal personalized solutions to automate their health activities. In addition to their functional needs, they explicitly seek to optimize non-functional qualities (*minimize costs and delays, maximize security, etc.*). It is necessary to support flexible customization of dynamic context-aware mobile applications to meet the current needs of clients. A complete dynamic agent-based semantic selection and deployment of services in the field of smart health seems to be an appropriate solution to meet user needs and context evolutions. In this section, we propose a new approach that aims to meet user's functional needs and optimize their constraints and evolved preferences. For that purpose, we use a context-aware generic CxQSHealthOnto ontology that implements the concept of multi-agent and Context-aWare Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR) [3]. We have chosen ASSACR approach [3] that has proven its efficiency and effectiveness in the various smart domains (Home, City, Health, etc.). We take advantage of multi-agent approach to improve the execution time of the selection and deployment process while running cooperative control and/or distributed agents.

The main goal is to select the optimal personalized service in terms of QoS with lowest violation degree of user's constraint in reduced execution time. Services to be selected from a large set of services are stored respectively in service registry. An additional layer of service selection and optimization is integrated which provides: (1) a valuable balance between solution quality and good performance, scalability and efficiency; and (2) a dominance relationship to reduce the search space of context-aware quality service selection by purring the dominated candidate service. A dynamic

services selection algorithm based on new ASSACR multi-agent strategies has been performed to solve context-aware services selection in a reduced search space more effectively and efficiently.

4.1. General Architecture

We propose a new approach that includes a context supervisor agent to observe context changes and/or user preferences changes in order to continuously provide suitable service. The presented work uses different local controller agents that communicate and coordinate with each other to build reconfigurable distributed mobile applications and solve context-aware quality service selection optimization problems as shown in Figure 2**.** The use of the found solutions makes it possible to exploit the research experience acquired by the solution-building agents in the future iterations of the algorithm. To bring about an enhancement in the service selection process, we develop a multi-agent decentralized platform that improves the execution time of the optimization process through multi-agent collaborative and coordination strategies and involves the change of user constructs about the new environment. The proposed platform manages a large number of heterogeneous services described with both contexts QoS and user's needs to obtain optimal personalized service. The architecture of the Agent-based Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR) consists of four layers :

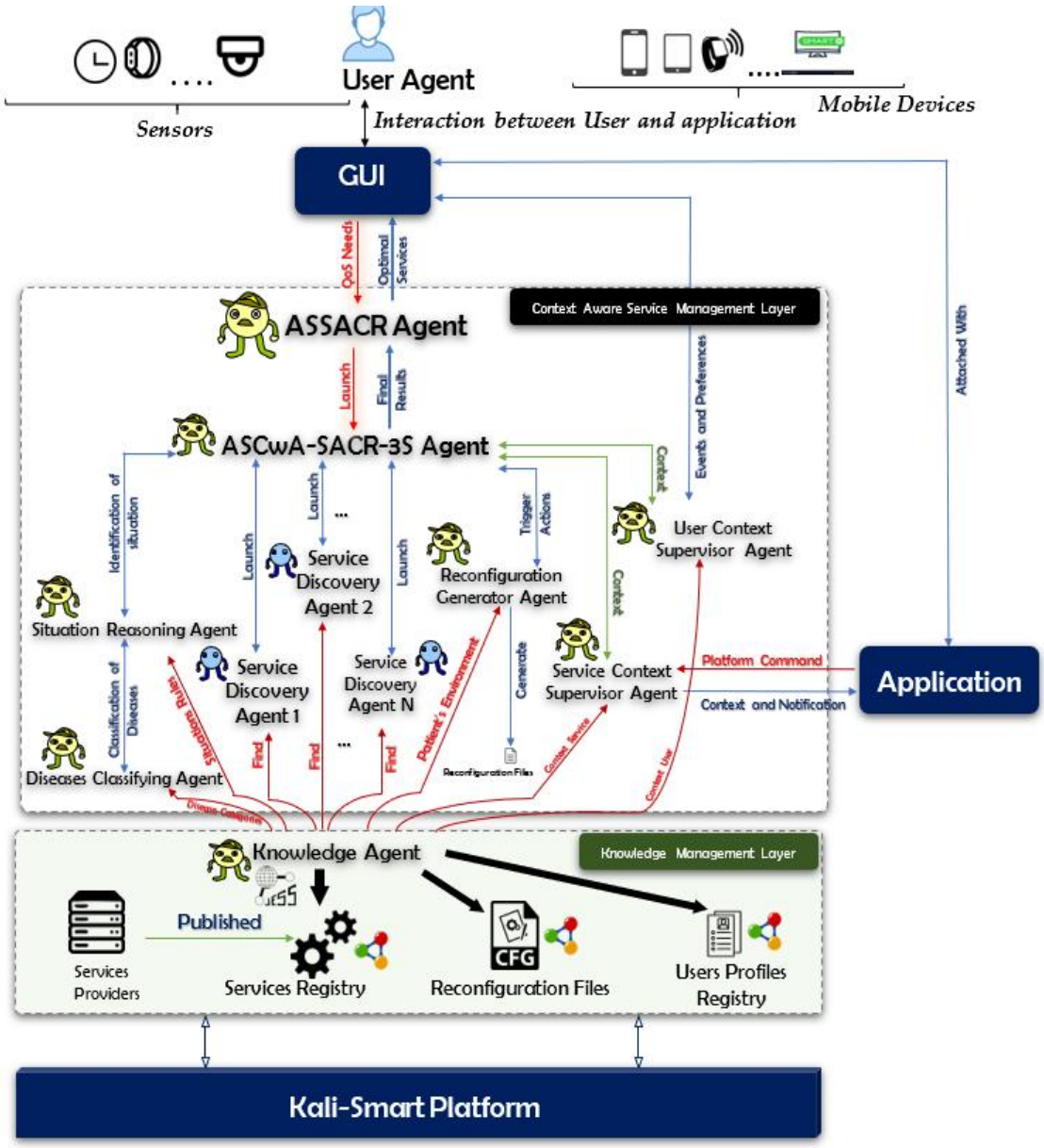

Figure 2. Multi-Agent-based management framework in Smart Health.

- **User Layer:** represents the patient that wants personalized health services.
- **Context-aware Service Management layer:** is seven kinds of agents, which communicate and collaborate with each other to manage services selection process. Each agent has its specific functionalities while accessing ontology knowledge. The description of all agents in our system is explained as follows:
  - **Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR) Agent:** allows patients expressing their functional and non-functional needs and to receive their expected results. It calls:

    (1) – *Situation Reasoning Agent* to identify automatically the health situations of patient.
    (2) – *Diseases Classifying Agent* to classify health situations into predefined classes.
    (3) – *ASCwA-SACR-3S Agent* to find the optimal customized service in terms of QoS.
    (4) – *Reconfiguration Generator Agent* to generate the reconfiguration file.
    (5) – *User Context Supervisor Agent* to monitor patient context.
    (6) – *Service Context Supervisor Agent* to monitor service context.

  - **Situation Reasoning Agent:** this agent is a semantic inference engine responsible for identifying different health situations of different patients as well as same patient.
  - **Diseases Classifying Agent:** this agent is a semantic inference engine responsible for identifying different health situations of different patients as well as the same patient.
  - **ASCwA-SACR-3S Agent:** the role of this agent is to find an optimal selection of services that can manage effectively all the identified health situations. It takes as input the patient's constraints and preferences described in our ontology and description of semantic services, then it produces an optimal service according to the patient's constraints and preferences.
  - **User Context Supervisor Agent:** this agent allows as much discovering and updating the activities related to a patient and his current context on an individual level. For example, where are the user (*in the car, at work, outside*), his mobile equipment and resources (*CPU speed, battery level, type of network*), and all current medical information (*his symptoms, his signs vitals, and his treatment*).
  - **Service Context Supervisor Agent:** it permits to manage the list of available services, the capabilities of the mobile device to deploy and run the required service. Each service is enriched by a set of contextual parameters (availability, location and execution time, hardware and software requirements, service execution preferences) and QoS parameters.
  - **Reconfiguration Generator Agent:** it scans the smart environment with the help of the Kali-Smart middleware [2] in order to know the available devices and orchestrate the best way to deploy the services. It receives the orchestrated actions provided by the Actions Orchestrator agent and generates the output reconfiguration file.
- **Knowledge management layer**: contains a single K**nowledge-based agent**. This agent performs ontology semantic management of Cloud services and customer profiles. It stores the description of services of all service providers. It also responds to requests sent by the *Service Context Supervisor Agent* and can update the ontology when it is required. It includes the following registries and files :
  - **The Services Profiles registry**: contains the semantic description of services for all services providers within our ontology.
  - **The Users Profiles registry** contains the various semantic description of users' profiles within our ontology.
  - **The Reconfiguration files** : contains all generated reconfiguration files.
- **Kali-Smart middleware layer** is a services-based context-aware platform [2] that ensures the execution of services on the devices and/or their redeployment in the Cloud from the generated reconfiguration file. It adapts mobile applications on different mobile devices, desktops, and laptops. The Reconfiguration Generator Agent uses the Kali-Smart in case of low device resources situations in order to ensure service continuity. Kali-Smart platform retrieves commands related to the services that must be adapted and their hosts. Then Kali-Smart platform reconfigures or redeploys these services.

The ASSACR agent manages distributed health mobile applications which run on different operating systems, either in mobile devices or from the Cloud. It consists of different health services that are used to manage and identify the different situations of patients. The ASSACR agent provides optimal personalized services in optimal delay and ensures dynamic reconfiguration of managed services and their health sensitive data in the mobile cloud. The sequence diagram of agent interaction is described in Figure 3 and detailed as follows:

- When a patient configures its session, he sends his profile and QoS needs. The current profile is stored in the ontology model using the *Knowledge agent*. This profile is sent once (or updated when needed) when asking for a new situation identification.
- Since the *ASSACR agent* receives the profile, it calls the *User Context Supervisor Agent* to monitor patient health data and capture new incoming events from deployed sensors. If any context

changes, the *ASSACR agent* calls the *Situation Reasoning Agent* to identify possible situations from the incoming events and contextual information.

— If some situations (*such as heart situations*) exist, The *ASSACR agent* calls *ASCwA-SACR-3S agent* in order to find the optimal services to manage these situations. It launches in parallel several *Services Discovery Agents* to retrieve suitable services from a large number of available Cloud services. We assume that each agent has a registry of all health services classified in our ontology. Each agent retrieves the best local services and sends it to the *ASCwA-SACR-3S agent*. The *ASCwA-SACR-3S agent* in turn determines the global optimal service that merges and sorts all best local services.

— The *ASSACR agent* calls the *Reconfiguration Generator Agent* in order to trigger appropriate action services to the user. When the Kali-Smart middleware [**2**] is active, it sends the description of all the available devices to the *Reconfiguration Generator Agent*. Therefore, the *Reconfiguration Generator Agent* has the descriptions of all available devices around the user, and the simple algorithm is executed to generate the actions reconfiguration file.

— The *Reconfiguration Generator* reads the actions reconfiguration file and executes available services.

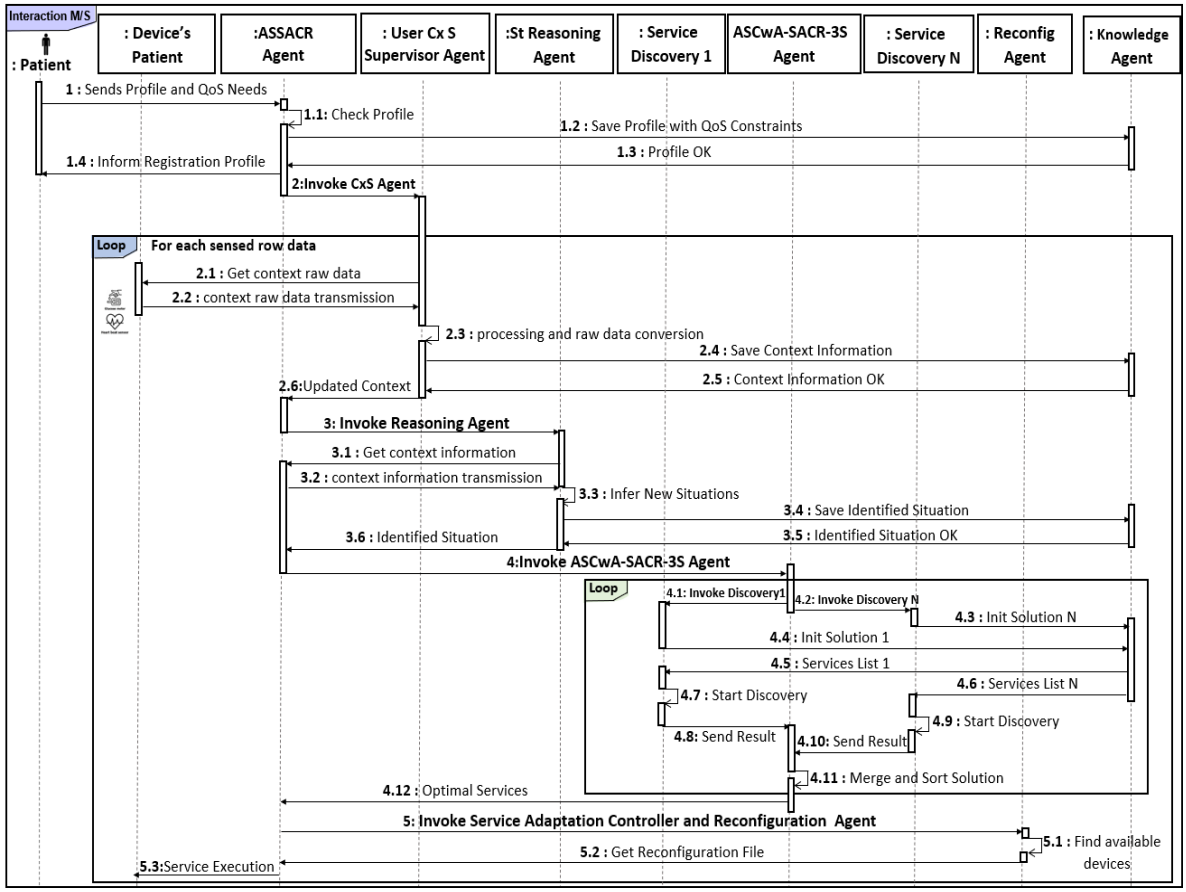

Figure 3. Sequence diagram of our platform.

4.2 Functional description of the proposed approach

The proposed approach aims to quickly provide an optimal personalized health services according to the patient's needs (see Figure 4). First, the patient specifies his functional and QoS needs. The QoS needs are expressed as constraints and preferences that can be static or dynamic. The current patient profile is stored in the ontology model and it is sent once (*or updated when needed*) when querying for optimal health service. Since the *ASSACR Agent* receives the patient profile and their needs, it calls the *User Context Supervisor Agent* ❶ to monitor the patient's health data. Then, the *ASSACR Agent* calls the *Situation Reasoning Agent* to identify situations based on the patient's health

context data through ontology inference rules❷. When a situation is identified, the *ASSACR Agent* calls the *ASCwA-SACR-3S agent* to select, optimize and deploy the best service in terms of QoS that should respect all the patient constraints. This process is a three-stage algorithm for optimal health services selection based on multi-agent approach and the dominance operator. The first stage consists of parallel discovering and selecting a set of optimal services respecting patient's constraints❸. This stage uses the dominance operator within multi-agent approach❹ to reduce the global execution time of the system. To improve the performance service selection and optimization, the approach uses cooperative multi-agent that is launched simultaneously for reducing global execution time of the system. Several intelligent agents attempt to cooperate dynamically in parallel for the discovery and selection of optimal health services from a large number of candidates❺. Each agent provides local solutions as a set of optimal health services and communicates those solutions to other agents. The global solution is the set of optimal health services selected among all the best local solutions❻. In the second stage, the *Reconfiguration Generator Agent* finds the available devices, orchestrates the best way to deploy them and generates the output reconfiguration file❼. In the third stage, the *Reconfiguration Generator Agent* executes the available services❽.

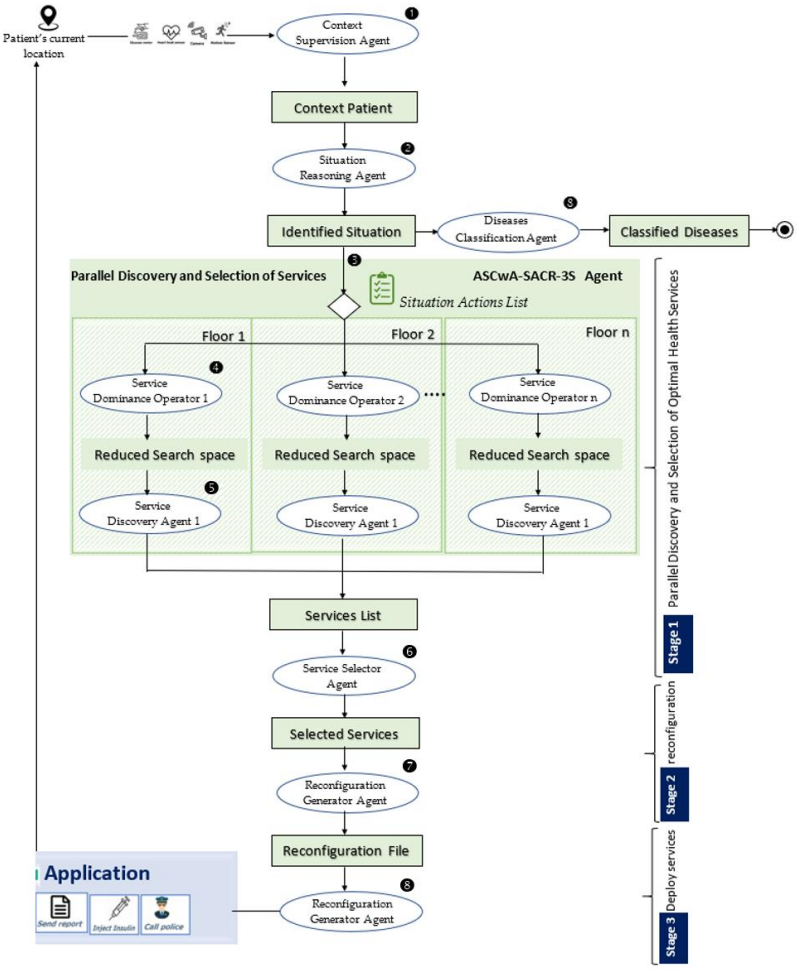

Figure 4. Proposed situation reasoning and parallel service selection and deployment process.

4.3 Problem Statement and Proposed Algorithm

In this section, a Semantic Context-aware Decentralized Quality Service Selection with 3 Stages (SCwD-QSS-3S) algorithm for dynamic services selection and deployment is detailed. We deal with the services selection problem with multi-objectives functions of QoS model and we search the optimal customized solutions considering the functional and non-functional needs of a user. Different collaborative strategies (*e.g. centralized and decentralized*) are proposed to find optimal service based on dynamic semantic-based multi-agent selection and deployment process. The

different kinds of agents that collaborate and cooperate together will enable better exploitation of research experiences acquired in building more accurate and quality solutions in future iterations of the algorithm.

4.3.1 The Semantic Context-aWare Quality Service Selection Problem (SCw-QSS)

The problem statement takes into account the following notations description:

- $SP = \{sp_1, sp_2 \dots, sp_n\}$, indicates n service providers of a mobile cloud system. Each service provider sp_i can be categorized into two types: *mobile* or *Cloud*, and contains a set hardware resources $\{CPU_i, RAM_i, AC_i\}$.
- $SS = \{ss_1, ss_2 \dots, ss_P\}$ with $ss_{i\ (i=1,2,\dots p)}$ are the semantic services that comprise it and its size is p tasks. For each semantic service ss_i , a service class $S_i = \{s_i^1, s_i^2 \dots, s_i^M\}$ of M concrete services having the same functionalities with different QoS where s_i^j represent the jth concrete service in the service class S_i.
- The set $QoS = \{q_1, q_1, \dots q_r\}$ of r parameters for each s can be classified into *positive* and *negative* QoS parameters denoted as QoS^+, QoS^-. Where for each negative QoS, lower values indicate better services. However, for each positive QoS, larger values indicate better services.
- Each user can define his own global QoS constraints $GCst = \{Cst_{q1}, Cst_{q2} \dots, Cst_{qr}\}$ and his own global QoS preferences $GPref = \{pref_{q1}, pref_{q2} \dots, pref_{qr}\}$.
- The service s_i is defined by QoS attributes different $QoS(s_i) = \langle q_{i,1}, q_{i,2}, \cdots q_{i,r} \rangle$ where $q_{i,j}$ represents the value of the j^{th} QoS attribute $(1 \le j \le n)$ of service s_i. A violation degree is assigned for each QoS attribute to penalize services that don't respect the user constraints. This degree is calculated as follows:

$$f_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} w_i \times \left(C_j - q_{i,j}\right) \quad (4)$$

  where

  - n is the number of discovered services.
  - m is the number of QoS attributes.
  - w_i is the weight of i^{th} QoS attribute;
  - $q_{i,j}$ is the value of the j^{th} QoS attribute of service i ;
  - C_i: is the user's constraint of the j^{th} QoS attribute.
- **The distance of next device from the user's location:** Devices with the shortest distance closer to user's current device tend to be selected as the best devices by minimizing f_1. Accordingly,

$$f_2 = \sum_{i=1}^{m} \left(D_{d_i}\right) \quad (5)$$

Where D_{d_i} is the normalized Euclidean distance between the device d_i and the current user's location; m represents the number of nearest devices from the user's location. This parameter covers optimal device selection as it takes less energy consumption to transmit data towards the user's device due to the short distance. The best devices have the smallest distance and vice versa. The normalized Euclidean distance D_d of the device d with the current user's location is:

$$D_d = \frac{\sqrt{\left(x_u - x_d^c\right)^2 + \left(y_u - y_d^c\right)^2}}{\sqrt{\left(x_d^{max} - x_d^c\right)^2 + \left(y_d^{max} - y_d^c\right)^2}} \quad (6)$$

where

- x_u,y_u: current location coordinates of user u,
- x_d^c,y_d^c: position center coordinates of device d,
- x_d^{max},y_d^{max} maximum coordinates coverage of device d.

— **The distance intra-devices:** The minimum distance between the devices and the selected nearest devices from a user is established. Devices with lower intra-communication costs tend to be chosen as rely by minimizing f_2. Accordingly,

$$f_3 = \sum_{i=1}^{m} \left(\sum_{j=1}^{l_j} D_{d_j} \right) \quad (7)$$

— **The CPU load**: The CPU load of the devices is specified to perform sensing, computation, and communication operations. By reducing f_2, lower CPU load devices tend to be chosen as the best devices. Accordingly,

$$f_4 = \frac{1}{\sum_{i=1}^{m} (CPU\_Load(d_i))} \quad (8)$$

We aim to select the appropriate service s_i^j to construct the best path services that trigger the corresponding set of actions for the identified situation and fulfill the global patient's QoS constraints and preferences. This selection is based on semantic agent-Based decentralized service adapter and controller and dominance relationship selection and deployment process such as (1) the total CPU load is minimized, (2) the global preferences of the patient are satisfied in the minimum time (*minimum number of iterations*) and (3) the distance of next device from the user's location and the total distance intra-devices is minimized. From such a perspective, the optimal path services P_s among all the possible ones which respect to the fitness function and time consumed:

$$Fitness\ (P_s) = Max\ (f_1); Min\ (f_2, f_3, f_4) \quad (9)$$

Subject to k patient's global constraints :

$$\forall t = 1..k \begin{cases} P_{s_{q_t}} \leq_{min} Cst_{q_t}, & If\ q_t \in QoS^+ \\ P_{s_{q_t}} \geq_{max} Cst_{q_t}, & If\ q_t \in QoS^- \end{cases} \quad (10)$$

4.3.2 Detailed algorithm

The main steps of the proposed algorithm are:

Step 1 – Automatic Identification of Patient's Health Situation. First, the *Situation Reasoning Agent* is able to identify automatically the patient's situation according to the health context data. It is based on SWRL (*Semantic Web Rule Language*) inference rules for each health context data sensed in real-time (*every 20 milliseconds*) using the *User Context Supervisor Agent*. The health context data consists of different vital signs and symptoms. It is also converted into a triple pattern in OWL (*Web Ontology Language*) format and inserted in CxQSHealthOnto ontology using a *Knowledge Agent*. The patient's health situations are dynamic and subject to change of context. Therefore, the platform must be enough adaptive to accept any updated context at any given time.

Step 2.1 – Parallel SCwD-QSS-3S optimization at the level of each agent. In this step, several agents aim to satisfy patient constraints by selecting the best services path for generating optimal service deployment. The optimization is done by adopting a penalty function-driven and multi-agent collaborative strategies to rapidly select the best service in terms of QoS while respecting the patient's constraints.

— **Solutions coding:** Each path service (solution) is represented by an array of *n* services.
— **Search space reduction:** The dominance operator is used to reduce the search space of SCw-QSS by penalizing the candidate services that violate patient's constraints.
— **Parallel services discovery:** Using the cooperative and collaborative services discovery strategy, local agents are evolved in parallel to find a set of optimal solutions. Each agent finds its best local solutions and sends it to the *ASCwA-SACR-3S* agent in each iteration for further improvement.

— **Evaluation:** This step aims to support personalized dynamic selection services with heterogeneous services providers. The QoS of path services (P_s) is calculated for each patient using the normalized QoS values of health services according to Table 4. The values of QoS attributes are normalized by the following formulas:

- **Normalization of aggregate positive attributes** (i.e., $\boldsymbol{q_t \in QoS^+}$)

$$q_t'(P_s) = \begin{cases} \frac{agg(q_t) - agg(q_t^{\min})}{agg(q_t^{\max}) - agg(q_t^{\min})} & , \quad If\ agg(q_t^{\max}) \neq agg(q_t^{\min}) \\ 1 & , \quad Otherwise \end{cases} \tag{11}$$

- **Normalization of aggregate negative** attributes (i.e., $\boldsymbol{q_t \in QoS^-}$)

$$q_t'(P_s) = \begin{cases} \frac{agg(q_t^{\max}) - agg(q_t)}{agg(q_t^{\max}) - agg(q_t^{\min})} & , \quad If\ agg(q_t^{\max}) \neq agg(q_t^{\min}) \\ 1 & , \quad Otherwise \end{cases} \tag{12}$$

where $agg(q_t^{\max})$ is the maximum aggregate values and $agg(q_t^{\min})$ is the minimum aggregate values of the j^{th} QoS criterion for the path services P_s. The $agg(q_j)$ presents the aggregated value of the j^{th} QoS criterion of P_s.

Table 4: QoS Aggregation functions.

QoS Attributes	Sequential	Parallel	Conditional	Loop
Execution time	$\sum_{j=1}^{n} Time_j$	$Max_{j=1}^{n} Time_j$	$Max_{j=1}^{n} Time_j$	$Time_j \times k$
Availability	$\prod_{j=1}^{n} Av_j$	$\prod_{j=1}^{n} Av_j$	$Min_{j=1}^{n} Av_j$	$(Av_j)^k$
Reliability	$\prod_{j=1}^{n} Re_j$	$\prod_{j=1}^{n} Re_j$	$Min_{j=1}^{n} Re_j$	$(Re_j)^k$
Security	$Min_{j=1}^{n} Sec_j$	$Min_{j=1}^{n} Sec_j$	$Min_{j=1}^{n} Sec_j$	Sec_j
Price	$\sum_{j=1}^{n} Cost_j$	$\sum_{j=1}^{n} Cost_j$	$Max_{j=1}^{n} Cost_j$	$Cost_j \times k$

— **Selection**: In this step, a list of semantically equivalent services (*location, time, and category*) with different QoS are evaluated and ranked by QoS constraints violation degree. Path services with the lowest constraints violation are assigned with a high priority value (Eq.9).

— **Merge and (re-)selections of local SCwD-QSS-3S optimal**. At the end of all iterations, the obtained ordered results at the level of discovery agents are merged. Then, the best path is selected. In this step, a parallel discovery and selection strategy is proposed and evaluated to reduce the execution time of the optimization task.

— If the stopping criterion is not reached return to **Step.4** with $i = i + 1$; otherwise: **end.**

Step 2.2 – Generation of (re-)deployment files. This step identifies the available devices and orchestrates the nearest available devices to deploy services selected previously. It takes into account the list of identified situation and the user's domain to ensure the multimodality interactions and find the appropriate device. Only mobile devices installing SNMP (Simple Network Management

Protocol)[1] with Kali-Smart platform [2] can detect automatically the list of available devices around the user and their capabilities. The retrieved devices and paths around the user are stored in the ontology model. When the patient's interaction facility is reached then the application generates a new re-deployment script to those devices that also received the interaction facility as well as the activation of redundancy of health service routing paths. The re-deployment script will be placed in a Kalimucho Cloud server.

Step 2.3 – Copy configuration files and services execution. The generated configuration files in the previous phase will be copied to the level of the routers concerned by the redundant route. It follows an optimal (re-)deployment strategy in regards to the capabilities of the available devices.

Algorithm 1: Semantic Parallel discovery strategy for dynamic service selection and deployment

Inputs: health context data, max_itr, local_registry_services
Outputs: $P_{OPTIMAL}$: Path Optimal Services, C_{FILE}: Configuration files

Begin
1 : Collect health context data
2 : Identify patient's health situations using SWRL rules
3 : $P_{OPTIMAL} \leftarrow \emptyset$
4 : Iteration $\leftarrow 0$
5 : **While** (Iteration ≤ max_itr) **do**
6 : reduce search space using dominance-based constraints violation degree
7 : **Fork (**service discovery agent using *L_services* in local_registry_services**)**
8 : Evaluate the objective function for each site *L_services*
9 : Local $P_{OPTIMAL}$ ← Select non dominate solution from current *L_services*;
10 : $P_{OPTIMAL} \leftarrow P_{OPTIMAL}$ U Local $P_{OPTIMAL}$
11 : **End Fork**
12 : $P_{OPTIMAL}$ ← Select optimal services from all agent-based site-optimal solutions
13 : Notify all discovery agents with new global services optimal solutions ;
14 : Iteration ++;
15 : **End**
16 : C_{FILE} ← Generate (re-)deployment files
17 : Copy C_{FILE} and deploy services
End

5. Possible scenarios and Validation

5.1. Prototype Implementation

To validate the proposed approach, we have implemented a prototype based on multi-agent platform Jade for cooperative services selection. Jade is widely used in applications involving artificial intelligence (AI) techniques and intelligent managing of health processes **[**21**]**. The CxQSHealthOnto ontology is implemented using Protégé tool **[**22**]** to semantically describe the different concepts of smart health and match functional user needs to situations using inference rules.

[1]Simple Network Management Protocol (SNMP) is an Internet Standard protocol for collecting and organizing information about managed devices on IP networks

We use Apache Jena [22] to interpret an OWL file expressing CxQSHealthOnto ontology and transform each OWL class into a corresponding java class including inheritance, data properties and object properties of that class formalized as JADE predicates. Figure 5 presents snapshot of the interactions between agents of the proposed work.

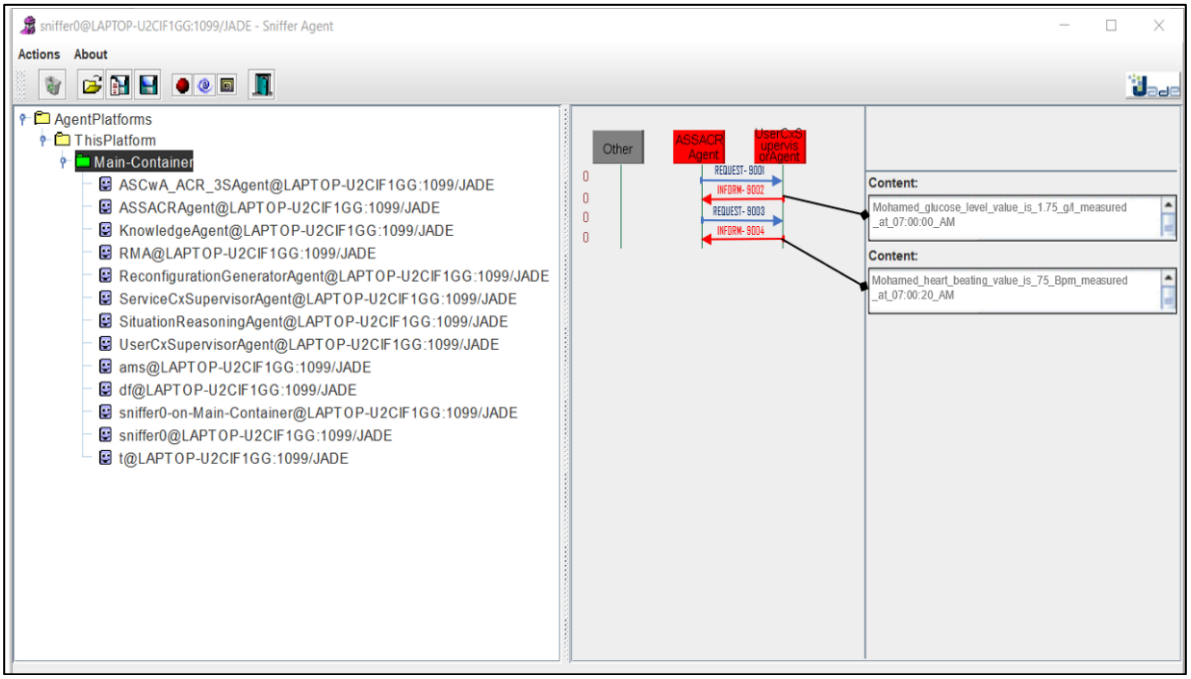

Figure 5. Snapshot of the interactions between agents.

The prototype has been developed in collaboration with physicians and programmer students. The physician specifies conditions in case of health situations while the programmer specifies conditions in case of services management. The prototype can able to manage and identify 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. Also, it can frequently identify user situations (*using SWRL rules*) 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.

The prototype is smart enough to identify any urgent situations and trigger appropriate action services to the user. In case of urgent situations like health-heart-situations, the expert (*i.e. physicians in the case of smart health domain*) sets situations priorities based on the user's context condition (*i.e. patient's health and disease's type*). The main objective of our Java-based prototype is to automatically manage the health situations in different smart environments (smart-home, smart-office, smart-hospital, etc.) in an efficient way using a parallel paradigm in order to perform the service delivery process.

5.2. Use case scenarios and validation

To evaluate the proposed approach in rich situations, we condense into investigating a case study in the healthcare field involving a diabetes follow-up management for orderly patients. We intend to (*add/remove/migrate services*) basing on the patient's health information using the *reconfiguration generator agent*. First of all, we assume that the patient has a registry of all his situation rules classified as urgent/daily-life in our CxQSHealthOnto ontology. We used our platform to deal with several situations in four smart domains (smart health, smart home, and smart hospital). 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.

Now, let us suggest this scenario:

We are interested in the daily life of a user (*Mr. Adam*). He lives in a smart home. 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 health situations (*e.g., high-level glucose or low-level glucose*). He is equipped with wearable smart objects and mobile devices (*glucose sensor and smartwatch*) that are responsible for monitoring raw sensor data (*e.g., the value 1.75 as raw data acquired from the glucose sensor*). The collected data were stored in a MySQL database. The *User Context Supervisor Agent* processes and represents the acquired raw sensor data into context information where metadata is added (*e.g., Mr. Adam glucose level value is 1.75, its unit is g/l, and measured at 07:00:00 AM*). The *Situation Reasoning Agent* uses this context information in order to infer user's situations (*e.g., Mr. Adam has high glucose situation*). The application ensures the deployment of appropriate services regarding inferred situations (e.g., injecting insulin). For instance, we cover different types of diseases and allows a physics and a nurse in other sites (i.e. hospital or clinic) to manage automatically the updates of a patient and advise quality services with the help of the proposed approach. When it is time for *Mr. Adam* to go to a hospital, the system migrates the appropriate health services to the physician tablet to ensure the continuity of service. The battery level is low; the only solution is to migrate the service in the Cloud. The system deploys the Google map and selects the fastest route to the hospital. The intelligent agent-based traffic control system can be deployed to avoid crowded streets, and offer a test road on the map, regardless of the potential traffic congestion. Figure 6 depicts the elements that constitute the infrastructure of Diabetes follow-up system,

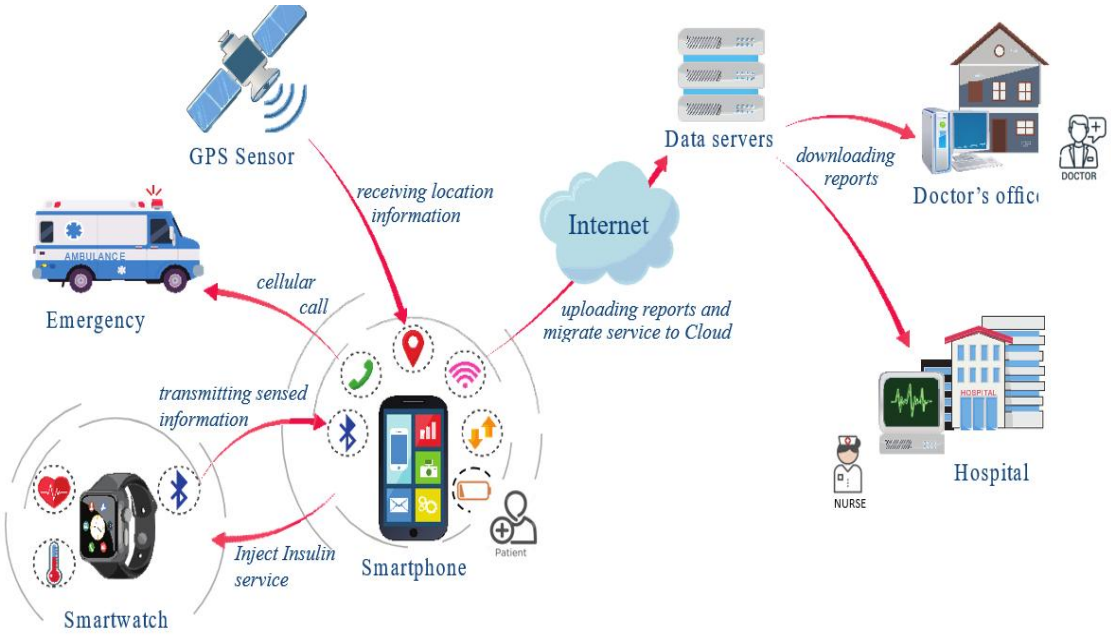

Figure 6. The infrastructure of the diabetes follow-up system

5.3. Modeling Diabetes follow-up with CxQSHealthOnto

Based on *CxQSHealthOnto* ontology model described in Section 3, several required subclasses have been identified for the realization of a *Diabetes follow-up*. Some individuals of the instantiation model for the case study are shown in Table 5. The subclasses involved are User, Device, Location, Service, Symptoms, and Situation.

Table 5. Modeling the Diabetes follow-up.

Main Class	Subclasses
User	Patient, Physician, Nurse.
Device	Sensor, Actuator.
Sensor	Glucose sensor, GPS sensor, IP camera, weight sensor.
Actuator	Inject Insulin, motion sensor, Smartphone, Tablet, SmartWatch.
Location	Home, Hospital.
Time	Before dinner period.
Symptoms	Glucose is 'High', Glucose is 'Low', Walk is 'less'
Situation	Diabetic, Risk of death
Service	New sensed data, assess patients conditions, inject insulin and emergency actions

Figure 7 depicts some examples with objects of the main classes that comprise part of the CxQSHealthOnto instantiation model used in our case study. Particularly, the smart home acts as IoT environment that is composed of many sensors. *Mr. Adam* is located in his smart home, the system checks *high glucose level situation*. GPS sensor is used to monitor the user's locations and other sensors (e.g. *glucose sensor and time sensor*) are used to measure other contextual information (*e.g. glucose level and current time*). The high glucose situation receives glucose event, location event and time event and triggers the inject insulin action.

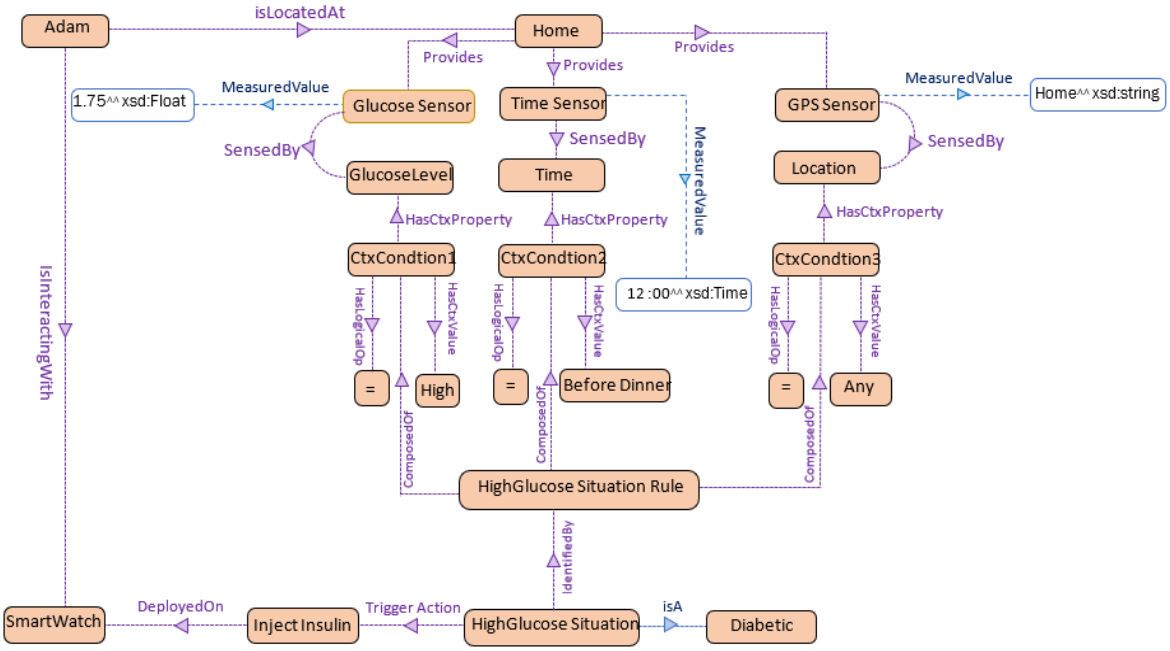

Figure 7. Examples of class objects using CxQSHealthOnto.

The implemented subclasses and produced contextual information model all health situations that are used to trigger appropriate actions depending on the user's environment, preferences and devices capabilities. Examples of such contextual information are shown in Table 6, which shows two health situations: *high glucose situation*, *risk of death situation*. The first situation represents the measurement of high glucose level that triggers the inject insulin action. The second situation represents the risk of death situation combines high glucose, high heart beating and trigger inject insulin and call for emergency actions.

Table 6. Examples of contextual information

ID	Health Activity	Smart Object	Context		Situations
0139	Glucose event	Glucose sensor	Time	: **7:00:00**	Situation: HG

			Location : **Home** Value : **1.75 g/l**	Rule : SS_0
0140	Heart beating event	Heart beat sensor	Time : **7:00:20** Location: **Home** Value: **75 Bpm**	Situation: RD
0141	User's state	Camera	Time : **7:00:55** Location: **Home** Value: **Lay down**	Rule : CS_1
0142	Inject Insulin	SmartWatch	Exact timestamp	Rule : SS_0
0143	Send Report	Physician Tablet: 01	Every 10 seconds	-
0144	Call 911	Friend Device: 02	Exact timestamp	Rule : CS_1

5.4. SWRL Inferences and Interrogations Rules for CxQSHealthOnto

SWRL (Semantic Web Rule Language) [24;34] with Java Expert System Shell (Jess) [25] engine is used to express queries across diverse data sources that can be used for semantic data extraction and situation detection. We utilize SWRL queries to extract semantic data from our CxQSHealthOnto ontology. Thus, we develop many inference SWRL rules based on CxQSHealthOnto. The main goal of these rules is identifying the health situations related to monitored patient context. We define a customized Java Class called *SemanticDataSetSWRL* based on OWL API [26] that creates, manipulates, and serializes CxQSHealthOnto ontology model. We utilize SWRL API [27] and Jess engine [25] to execute OWL-based SWRL rules at run-time. Table 7 shows some inferences and interrogations rules.

Table 7. SWRL inference and interrogation rules.

Rule 1 allows migrating services to the Cloud

Profil(?p) ∧ Device(?d) ∧ BatteryLevel(?d, ?level) ∧ **swrlb:equal***(?level,* **"Low"***) ∧ Health_Service(?s) ∧ DeployedOn(?s, ?host) ∧ DefinedAs(?d, ?host) ∧ Cloud(?cloud) ⟶ Migrate (?s, ?cloud)*

Rule 2 allows identifying the diseases seasonal allergies based on vital signs, patient's body parameters and symptoms

Profil(?p) ∧ DependOn(?p, ?d) ∧ Device(?d) ∧ DefinedAs(?d, ?host) ∧ Health_Service(?s) ∧ DeployedOn(?s, ? host) ∧ Category(?s, ?category) ∧ **swrlb:equal***(?category,* **"Infected"***) ∧ Inputs(?s, ?i1) ∧ Seasonal (?i1) ∧ ExistenceValue (?i1, ?value1) ∧* **swrlb:equal***(?value1,* **"Yes"***) ∧ Inputs(?s, ?i2) ∧ feebleness(?i2) ∧ ExistenceValue (?i2, ?value2) ∧* **swrlb:equal***(?value2,* **"Yes"***) ∧ Inputs(?s, ?i3) ∧ Articular_Pain(?i3) ∧ ExistenceValue (?i3, ? value3) ∧* **swrlb:equal***(?value3,* **"Yes"***) ∧ Inputs(?s, ?i4) ∧ Headache(?i4) ∧ ExistenceValue(?i4, ?value4) ∧* **swrlb:equal***(?value4,* **"Yes"***) ∧ Inputs(?s, ?i5) ∧ Body_temperature_sensor(?i5) ∧ QualitativeValue(?i5, ? ?value5) ∧* **swrlb:equal***(?value5,* **"High"***) ∧ Outputs(?s, ?m) ⟶* **sqwrl:select***(?m)*

Rule 3 allows displaying treatment for hyperglycemia type 1, Seasonal influenza, anemia"

Hyperglycemia_1(?g)∧ ManagedBy(?g, ?t) ⟶ **sqwrl:select***(?t)*

Seasonal (?g)∧ ManagedBy(?g, ?t) ⟶ **sqwrl:select***(?t)*

Anemia(?g)∧ ManagedBy(?g, ?t) ⟶ **sqwrl:select***(?t)*

Rule 4 returns list all services with their corresponding QoS.

Service(?s) ∧ *Has_QoS(?s, ?qos)* ∧ *Has_ServiceID (?s, ?id)* ∧ *Has_QoS(?qos,* ?qos_value*)* ⟶ ***sqwrl:select****(?s, ?qos,* ?qos_value*)*

Rule 5 returns all the available devices around the user's location.

User(?u) ∧ *IsLocatedAt(?u, ?location)* ∧ *IsNear(?location, ?d)* ∧ *Availability(?d,* ***"Yes"****)* ⟶ ***sqwrl:select****(?d)*

5.5. Service Quality Selection and Deployment in a Smart Healthcare Diseases Diagnosis

An example is given below is intended to show the applicability of SCwD-QS-3S both for evaluation and selection of the best path services from some alternatives. The medication delivery is realized using four relevant paths (see Figure 8). In this experimentation, the user specifies his constraints in terms of price and response time as follows: the medication should be delivered with a cheaper price in faster execution time from their request. The *ASCwA-SACR-3S Agent* must respect quality user's preferences and selects the optimal path having the best score (Eq. 4) according to Adam's preferences. All QoS attributes of services with « high » values correspond to interval] 0.7, 1].

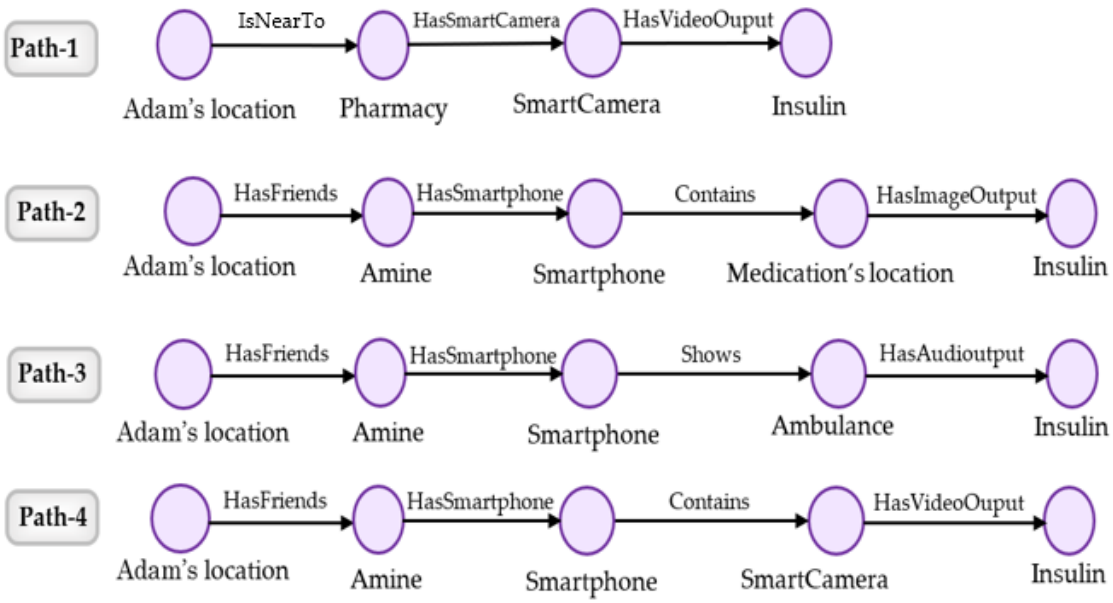

Figure 8. Possible semantic medication paths.

To do this, we sort the search results by score after the evaluation of path distance and quality of service (Table 8). Path #3 turns out to be the best. Differences can be seen in the high benefit and minimum cost of this path and other paths, which is due to the low processing load compared to other paths (*e.g. low violation constraints compared to other paths*).

Table 8. Discovered semantic paths for the diabetic scenario.

User	Health Situation	Path	Score
Adam	High-level glucose	{Adam's location, IsNearTo , Pharmacy ; Pharmacy, HasSmartCamera , SmartCamera; SmartCamera, HasVideoOuput , Insulin}	Medium

	{Adam's location, HasFriends , Amine ; Amine, HasSmartphone , Smartphone; Smartphone, Contains , Medication's location; Medication's, HasImageOutput , Insulin}	Medium
	{Adam's location, HasFriends, Amine ; Amine, HasSmartphone, Smartphone; Smartphone, Shows, Ambulance; SmartCamera, HasAudioutput, Insulin}	High
	{Adam's location, HasFriends, Amine ; Amine, HasSmartphone , Smartphone; Smartphone, Contains , SmartCamera; SmartCamera, HasVideoOuput , Insulin}	Medium

5.6 Experiments and Results

In this section, the performance of ASCwA-SACR-3S is validated through several experiments, which are carried out on real data dataset. These experiments were performed on a PC with an Intel i7, 3 GHz processor, 12GB of RAM, Windows 10 (64 bits) operating system, NetBeans and Jade.

5.6.1 Experimental dataset

The experiments are carried out on a real dataset collected by Zheng *et al.* [23] containing 25 tasks annotated by CxQSHealthOnto and classified on six main diseases (*anemia, seasonal flu, food poisoning, hyperglycemia type1, and hyperthyroidism*), service provider and QoS. Each service is linked to a set of 100 health services (the size of search space is 25^{100}). Each service considers five QoS (*availability, reliability, response time, price and security*) attributes whose values are normalized. For simplicity, each service considers two QoS attributes: the response time and the throughput as the QoS criteria.

The comparison between the different strategies (centralized and decentralized) depends on the score calculated using Eq. 12 of each solution. The performed experiments vary the values of some parameters to evaluates different strategies in terms of accuracy and computational time. The used parameters are situations rules-number and agents-number. The compared values are calculated by the average of ten fitness results for each execution to minimize the probabilistic mistake.

5.6.2 Evaluation metrics

We used four performance metrics: execution time, optimality, application's lifetime and number of discovery m-health services. The computation time metric indicates the required time to detect potential health diseases and find out the near-optimal solutions of the SCwD-QSS problem. The optimality metric indicates the quality of the obtained final path services solutions by the compared algorithms in terms of their utility value as defined by the aforesaid optimality equation :

$$Optimality = \sum_{t=1}^{r} w_{q_t} \times \overline{(Cst_t - q_t)} \quad (13)$$

This function adopts the Simple Additive Weighting (SAW) method by normalizing the value of violation degrees $\overline{(Cst_t - q_t)}$ between the of the t^{th} QoS attribute and the t^{th} global constraint. These violation degrees are aggregated using the weights w_{q_t} with $w_{q_t} \in [0,1]$ and $\sum_{t=1}^{r} w_{q_t} = 1$ which represent the importance and importance of each q_t by the user.

5.6.3 Results and Discussion

The result obtained from the proposed method for the identification of health situation and quality services selection will be discussed in this section

5.6.3.1 Execution time comparison

In this section, we compare the performance of different strategies in terms of execution time. The execution time is the average response time needed to accomplish the detection of potential health diseases and triggering appropriate services. The services dataset is fixed to 100 and the number of iterations to 100 divided over the number of agents. Figure 9 shows that the best execution time is obtained when the number of agents is five. This is due to the increase in the number of communications between agents. When the number of agents exceeds four, the number of messages exchanged between agents increases, so it slows down the optimization process and increases the overall execution time.

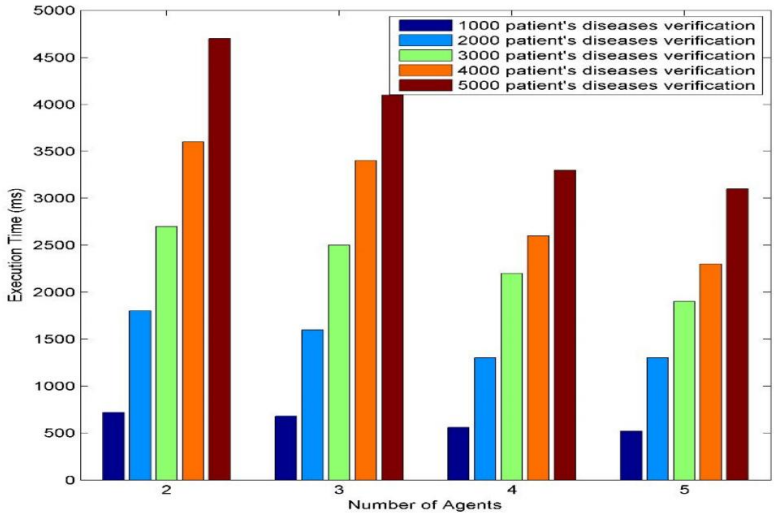

Figure 9. Execution time under various numbers of diseases' verification in the proposed approach.

5.6.3.2 Optimality comparison

We have compared the optimality of the results obtained from the Kali-Smart using different strategies (centralized and decentralized). The results are evaluated using Eq.13 which calculates the score of each solution (optimal services set). Figure 10 shows the different score values of different agent strategies by fixing the number of agents to 4 and the number of services between 50 and 1000 such that the services dataset is fixed to 100. Varying the size of services set, the decentralized strategy is more accurate in all cases compared to centralized strategy. This can be explained by the fact that the decentralized strategy benefits from the complete and exploring of search space.

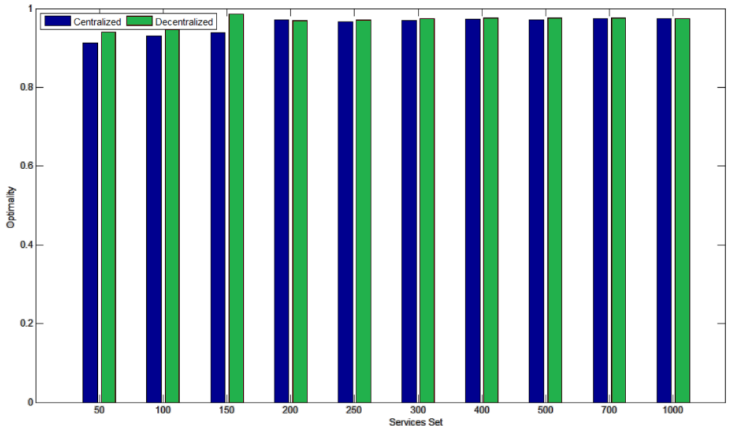

Figure 10. Optimality comparison of different strategies agents (centralized and decentralized).

5.6.3.3 Discovery health services comparison

Total discovered services is the summation of all the individual discovered services. Figure 11 represents the cumulative services number of both platforms and it may be observed that the proposed platform outperforms Kali-smart platform [2] in terms of number of the discovered

services. The total discovered services of Kali-smart platform [2] is **31** services while proposed extension was able to achieve **42** services. The proposed scheme was able to achieve discovery number greater than that of the Kali-smart platform [2]. As observed from the experiments above, the multi-agent-based Kali-Smart is better than the original Kali-Smart for managing and identifying potential health diseases. The multi-agent-based Kali-Smart is more efficient and decreases considerably the execution time, but also makes it practicable in spite of using near-real-time smart health applications. However, the execution time must be improved in future works with large-size health applications on a high-performance machine.

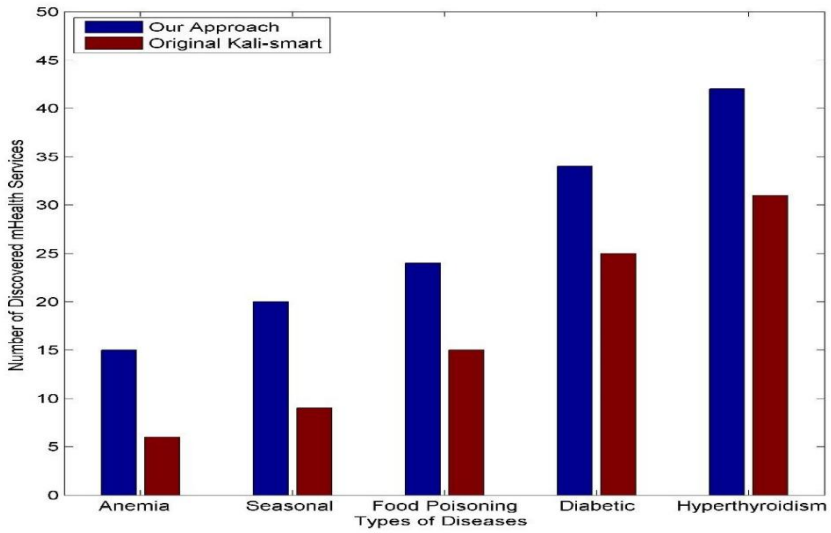

Figure 11. Discovery mhealth services comparison

5.6.3.4 Application's lifetime comparison

To evaluate the proposed approach, we performed many experiments on small and large configurations. We use five configurations: the first configuration consists of 200 sensors with a fixed frequency of 500ms and 400 events. The second configuration consists of 400 sensors with a fixed frequency of 500ms and 800 events. The third configuration consists of 600 sensors with a fixed frequency of 500ms and 12000 events. The fourth configuration consists of 800 sensors with a fixed frequency of 500ms and 16000 events. The last configuration comprises a large scale of 20000 events, 10000 sensors with a fixed frequency of 500ms. 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 condition high_temp is associated with the temperature sensor to detect if the sensed temperature is higher than 22 °C. An atomic condition face_detected is associated with the camera sensor to detect if the image frame contains at least one person. All these events are reported and processed by a server PC running Windows 10 (64 bits) on an Intel processor core i7 3 GHz, and 8 GB RAM. We evaluated and compared the impact of sensor availability on application's lifetime of the proposed work with other related tools such Kali-Smart [2] and MSSN-Onto [11] in five configurations as detailed in Table 9. The goal is to maximize the application's lifetime. Figure 12 shows the lifetime comparison for three tools. The proposed platform performs on service migrating/updating algorithm while application running under moving scenarios. Kali-Smart with multi-agent approach yields better service continuity until all devices become unavailable on all application configurations involved in the simulation. The durability of the application is caused by dynamic reconfiguration with a low deployment cost and short-distance interaction leads to a reduction in energy consumption performance of extended Kali-Smart as compared to other tools.

Table 9. Experimental configurations

Configurations	Sensors	Events	Frequency (ms)	Events Size (Kbytes)
Configuration 1	200	400	500	512
Configuration 2	400	800	500	512
Configuration 3	600	12000	500	512
Configuration 4	800	16000	500	512
Configuration 5	1000	20000	500	512

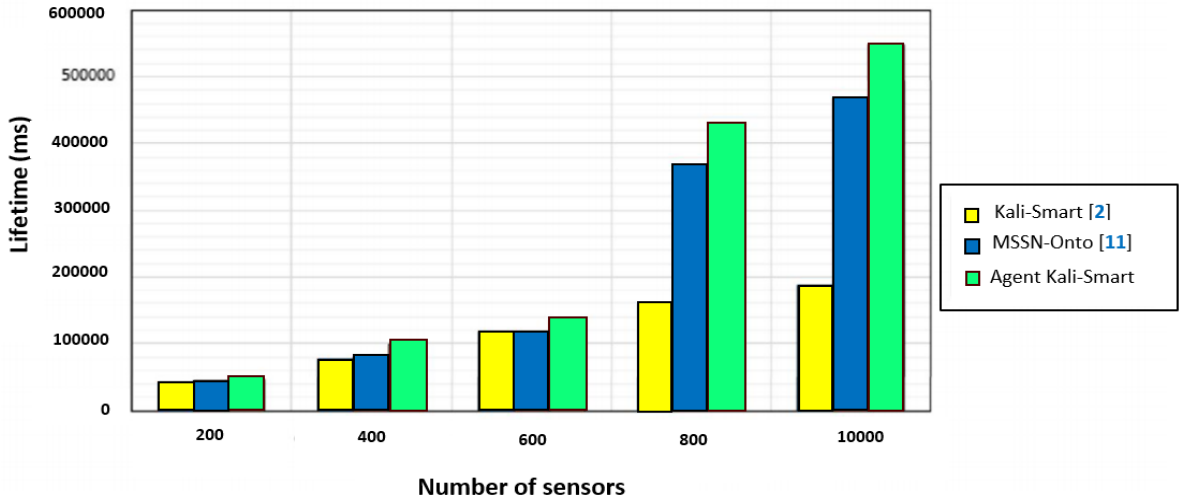

Figure 12. Application's lifetime comparison between the proposed work and other tools.

5.7 Comparative Analysis

The experiments on the proposed decentralized approach using real dataset attain promising results compared to the centralized approach in terms of accuracy and execution time. As a result, the proposed decentralized approach is much better than the centralized approach in managing and identifying health situations. The proposed decentralized approach achieves near-real-time optimal and more precise solutions for context-aware health applications. The proposed approach also gave improved accuracy results when compared with different strategies used on multi-agent as shown in Figure 10. It can be observed in Figure 10, that our proposed method shows better results of optimality than another multi-agent technique due to the good melanoma region localization. Further, we have achieved satisfying scalability in terms of services management and agents execution on an Intel processor core i7-2430 3 GHz in which the system can manage 10 agents easily. Bad configuration of agents will increase the execution time and create a huge transfer of data through the network. In future works, we intend to experiment proposed approach using real-world connected objects. Therefore, we plan to go into a real IoT infrastructure to perform large-size distributed applications and improve the execution time using a hybrid strategy.

6. Conclusion

We have proposed a novel approach based on a decentralized agents-based Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASACR) for accurate and automatic identification of health situations. Our approach consists of three steps: patient context monitoring, health situation identification and quality service selection and deployment. In comparison to previous state-of-the-art approaches, the presented work shows a very good representation of the patient's health context, an accurate situation detection rate and better system performance by selecting appropriate health services. The proposed approach can detect a variety of diseases of different patients as well as the same patient. Experiments were carried out on real-dataset using

different agents strategies (*centralized and decentralized*) for situation identification and service selection. Performance analysis has shown that the approach is more efficient than existing approaches in terms of respecting customer's constraints and preferences as well as the multi-agent factor that improves the computational time of the system comparing to the origin Kali-Smart platform. In addition, results confirm that the proposed approach achieved high optimality and low computational time, ranging from 0.977 to 720ms respectively. In future work, we will investigate other diseases by using this ontology model and extend this work by elaborating more semantic data for better management of location-aware situations.

Data Availability

The data that support the findings of this study are available from the corresponding author, upon reasonable request.

References

1. Mendez Mena, D., Papapanagiotou, I., & Yang, B. (2018). Internet of things: Survey on security. *Information Security Journal: A Global Perspective*, *27*(3), 162-182.
2. Alti, A., Lakehal, A., Laborie, S., & Roose, P. (2016). Autonomic semantic-based context-aware platform for mobile applications in pervasive environments. *Future Internet*, *8*(4), 48.
3. Slimani, S., Alti, A., Laborie, S., & Roose, P. (2016). An Autonomic Semantic Approach for the Detection and Monitoring of Diseases. *Workshop of Information System for decision support and the dissemination of alerts*, pp. *1-10*.
4. Angsuchotmetee, C., Chbeir, R., & Cardinale, Y. (2020). MSSN-Onto: An ontology-based approach for flexible event processing in Multimedia Sensor Networks. *Future Generation Computer Systems*, *108*, 1140-1158.
5. Bayer, T., Moedel, L., & Reich, C. (2019). A Fog-Cloud Computing Infrastructure for Condition Monitoring and Distributing Industry 4.0 Services. In *CLOSER* (pp. 233-240).
6. Mansour, E., Chbeir, R., & Arnould, P. (2019). HSSN: an ontology for hybrid semantic sensor networks. In *Proceedings of the 23rd International Database Applications & Engineering Symposium* (pp. 1-10).
7. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., & Taylor, K. (2017). IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. *Personal and Ubiquitous Computing*, *21*(3), 475-487.
8. Esposito, M., Minutolo, A., Megna, R., Forastiere, M., Magliulo, M., & De Pietro, G. (2018). A smart mobile, self-configuring, context-aware architecture for personal health monitoring. *Engineering Applications of Artificial Intelligence*, *67*, 136-156.
9. Keling, D., Roose, P., Dalmau, M., & Novado, J. (2014). Kali2Much: An autonomic middleware for adaptation-driven platform. In *Proceedings of the International Workshop on Middleware for Context-Aware Applications in the IoT (M4IOT 2014)* (pp. 25-30).
10. HameurLaine, A., Abdelaziz, K., Roose, P., & Kholladi, M. K. (2017). Towards an observer/controller and ontology/rule-based approach for pervasive healthcare systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *26*(3), 137-156.
11. Angsuchotmetee, C., Chbeir, R., & Cardinale, Y. (2020). MSSN-Onto: An ontology-based approach for flexible event processing in Multimedia Sensor Networks. *Future Generation Computer Systems*, *108*, 1140-1158.
12. Bhunia, S. S., Dhar, S. K., & Mukherjee, N. (2014, October). iHealth: A Fuzzy approach for provisioning Intelligent Health-care system in Smart City. In *2014 IEEE 10th international conference on wireless and mobile computing, networking and communications (wimob)* (pp. 187-193). IEEE
13. Rhayem, A., Mhiri, M. B. A., Drira, K., Tazi, S., & Gargouri, F. (2020). A semantic-enabled and context-aware monitoring system for the internet of medical things. Expert Systems.
14. Name, H. A. M., Oladipo, F. O., & Ariwa, E. (2017). User mobility and resource scheduling and management in fog computing to support IoT devices. In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)* (pp. 191-196).
15. Sasikaladevi, N., & Arockiam, L. (2012). Genetic approach for service selection problem in composite web service. *International Journal of Computer Applications*, *44*(4), 22-29.

16. Long, J., & Gui, W. (2009). An environment-aware particle swarm optimization algorithm for services composition. In *2009 International Conference on Computational Intelligence and Software Engineering* (pp. 1-4).
17. Yu, Q., Chen, L., & Li, B. (2015). Ant colony optimization applied to web service compositions in cloud computing. *Computers & Electrical Engineering*, *41*, 18-27.
18. Al-Tashi, Q., Kadir, S. J. A., Rais, H. M., Mirjalili, S., & Alhussian, H. (2019). Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access*, *7*, 39496-39508.
19. Valera, H. H. A., Dalmau, M., Roose, P., Larracoechea, J., & Herzog, C. (2020). DRACeo: A smart simulator to deploy energy saving methods in microservices based networks. In *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 94-99).
20. Mills, D., Sivarajah, S., Scholten, T. L., & Duncan, R. (2021). Application-motivated, holistic benchmarking of a full quantum computing stack. *Quantum*, *5*, 415.
21. Bellifemine, F., Caire, G., Poggi, A., & Rimassa, G. (2008). JADE: A software framework for developing multi-agent applications. Lessons learned. *Information and Software Technology*, *50*(1-2), 10-21.
22. Musen, M. A. (2015). The Protégé Project: A Look Back and a Look Forward. AI matters, 1 (4), 4-12.
23. Zheng, Z., Zhang, Y., & Lyu, M. R. (2014). Investigating QoS of real-world web services. *IEEE transactions on services computing*, *7*(1), 32-39.
24. SWRL: A Semantic Web Rule Language Combining OWL and RuleML (Version 0.7). Available online: https://www.w3g.org/Submission/SWRL/#3 (accessed *on 28 June 2020*).
25. Ameen, A.; Khan, K.U.R.; Rani, B.P. (2015). SemRPer—A rule based personalization system for semantic web. *Int. J. Web Appl*. 7, 23–38.
26. The OWL API. Available online: https://github.com/owlcs/owlapi/wiki (*accessed on 28 June 2020*).
27. O'Connor, M.; Shankar, R.D.; Musen, M.; Das, A.; Nyulas, C. (2008). The SWRLAPI: A development environment for working with SWRL Rules. In Proceedings of the 5th *OWLED Workshop on OWL:* Experience and Directions, Karlsruhe, Germany, 26–27 October 2008.
28. Christopoulou, S. C., Kotsilieris, T., Anagnostopoulos, I., Anagnostopoulos, C. N., & Mylonas, P. (2017). vhMentor: An ontology supported mobile agent system for pervasive health care monitoring. In *GeNeDis 2016* (pp. 57-65). Springer, Cham.
29. Chandrashekar, J., Gangadharan, G.R. (2015). QoS-aware web service composition using quantum inspired particle swarm optimization, in Proceedings of the 7th International KES Conference on Intelligent Decision Technologies, Springer, pp. 255–265.
30. F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi. (2003). The description logic handbook: Theory, implementation and applications. Cambridge university press.
31. G. De Giacomo and M. Lenzerini, . (1996). TBox and ABox reasoning in expressive description logics.," KR, vol. 96, no. 316–327, p. 10, 1996.
32. Munir, K., & Anjum, M. S. (2018). The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, *14*(2), 116-126.
33. Chen, L., Lu, D., Zhu, M., Muzammal, M., Samuel, O. W., Huang, G., & Wu, H. (2019). OMDP: An ontology-based model for diagnosis and treatment of diabetes patients in remote healthcare systems. *International Journal of Distributed Sensor Networks*, *15*(5), 1550147719847112.
34. O'Connor, M. J., & Das, A. K. (2009, October). SQWRL: a query language for OWL. In OWLED (Vol. 529, No. 2009).