future internet

MDPI

Article

Autonomic Semantic-Based Context-Aware Platform for Mobile Applications in Pervasive Environments

Adel Alti [1,*], Abderrahim Lakehal [1], Sébastien Laborie [2] and Philippe Roose [2]

[1] LRSD Lab, Computer Science Department, University of SETIF-1, Sétif 19000, Algeria; lakehal.abderrahim@gmail.com

[2] LUIPPA, University of PAU, Anglet, 64000, France; Sebastien.Laborie@iutbayonne.univ-pau.fr (S.L.); Philippe.Roose@iutbayonne.univ-pau.fr (P.R.)

* Correspondence: altiadel2002@yahoo.fr or alti.adel@univ-setif.dz; Tel.: +213-552-420-148

Academic Editors: Genoveffa Tortora and Giuliana Vitiello

Received: 19 June 2016; Accepted: 20 September 2016; Published: 29 September 2016

Abstract: Currently, the field of smart-* (home, city, health, tourism, etc.) is naturally heterogeneous and multimedia oriented. In such a domain, there is an increasing usage of heterogeneous mobile devices, as well as captors transmitting data (IoT). They are highly connected and can be used for many different services, such as to monitor, to analyze and to display information to users. In this context, data management and adaptation in real time are becoming a challenging task. More precisely, at one time, it is necessary to handle in a dynamic, intelligent and transparent framework various data provided by multiple devices with several modalities. This paper presents a Kali-Smart platform, which is an autonomic semantic-based context-aware platform. It is based on semantic web technologies and a middleware providing autonomy and reasoning facilities. Moreover, Kali-Smart is generic and, as a consequence, offers to users a flexible infrastructure where they can easily control various interaction modalities of their own situations. An experimental study has been made to evaluate the performance and feasibility of the proposed platform.

Keywords: context-aware; adaptation; ontology; constraint; semantic; autonomic reconfiguration

1. Introduction

Pervasive mobile applications are growing and their complexity is increasing dramatically. As a consequence, their maintainability and adaptability are becoming a challenging task. Moreover, in such systems, there is an increasing usage of different and heterogeneous mobile devices, as well as captors transmitting data (IoT). They are highly connected and can be used for different services, such as to monitor, to analyze and to display information to users. Therefore, data management and adaptation in real time are becoming challenging tasks. Context management is a key element for deriving semantically-rich context insights about mobile users (high-level adaptation task, preferences, intentions) from low level measurements (location, type of activity, etc.) to their online multimodal interactions, or more compellingly, from a combination of these. We argue here that users' mobility, users' situations (e.g., activity, location, time) and the limited resources of mobile devices (e.g., battery lifetime) need to ensure the service continuity on mobile devices.

Today, the field of smart-* (home, city, health, tourism, etc.) is highly multimedia oriented by nature; contents are heterogeneous; and it lacks a smart way to manage various modalities according to the current users' needs, usage situations and execution context. As a consequence, mobile applications certainly exist, but most often are inadequate according to users' expectations and, more precisely, the instant expectations. Moreover, the massive use of new technologies has led to a dramatic multiplication of a wide range of mobile applications, different usages and a huge amount of information. Using many different devices (home and professional PCs, set-top boxes, smartphones, etc.) make the user quite confused. This could imply that he or she would need to install a number of applications on various devices even if it is only just for short-lived interactions. That will provoke a huge multiplicity of applications (to install/uninstall/update), configurations and redundant context and user profiles. Therefore, it becomes mandatory to find a dynamic, intelligent way that can manage multiple devices at the same time. These devices need to communicate regardless of the difference of hardware/software. Our goal is to provide the suitable services and interactive applications in a transparent way for the user, according to his or her needs (personal, health, social and professional) and to his or her current context.

In order to design smart context-aware mobile applications, we need to exploit semantic web technologies, as well as the Kalimucho middleware [1], where the mobile application qualities are managed in an efficient way according to user needs and available context sources. Our proposed platform is implemented using an ontology-based approach. This ontology captures a shared conceptual schema common in location in specific application domains, such as tourism, healthcare, transport, sport, etc., and maintains semantic quality information in heterogeneous service providers for the service model. Our application will run on a flexible, extendable semantic model that will be able to evolve at every moment without any intervention of the users. We extend our previous knowledgebase set on a high semantic level on a cloud architecture [2] to include full distributed situation management in heterogeneous environments. To capture and characterize situations, we scan their environments (sensors and smart devices) and reason upon context changes in which multimodal and distributed behavioral adaptation is required. Our goal is to manage transparently all of the functionalities and additional modules that users may require in an ideal situation not available nowadays.

Our proposed platform is implemented using an ontology-based approach. This ontology maintains semantic quality information in heterogeneous service providers. We particularly focus on context-aware e-health mobile applications. To keep capturing the context in an efficient way, we need to have a platform able to monitor and handle continuous context changes. The distributed/centralized context monitor and event manager collect and manage any important information that could be important for the context regardless of its source and store it in a database. Then, it is represented and inferred within this context using our ontology and centralized context reasoner in order to deduce the current situations that are reported to the service controller. The latter is responsible for selecting the appropriate quality service to the user according to the inferred situations according to our strategy [3]. The execution is based on Kalimucho [1]. Such middleware allows a dynamic (re-)deployment strategy of services. Kalimucho is a platform that allows dynamic reconfiguration of applications on desktops, laptops and mobile devices. However, this platform is not currently focusing on the service selection and the service prediction, and then, it does not allow providing the appropriate service to the user.

Our main objectives are to extend the Kalimucho platform with a new layer called Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR) in order to: (1) dynamically monitor usage resources and user constraint changes among heterogeneous network protocols and mobile platforms (laptop, smartphone, etc.); (2) provide a centralized/distributed semantic multimodality event detection in order to manage relevant context information that could be important for the context regardless of its source; (3) provide a distributed action mechanism that will give the application the flexibility and dynamicity; (4) provide centralized semantic adaptation decision making to achieve the efficient adaptation of decisions; (5) find and select relevant semantic services for several heterogeneous mobile devices, many cloud services for a full usage multimodality and connected mobiles devices being shared every time; (6) maximize redundancy relays and switching mobile services; (7) autonomic optimization of the response time of situation matching under the criteria's priority (location, time, category). Our platform uses semantic technologies and the concept of multi-devices' context data representation to facilitate a seamless and interactive media service in a common contextual mobile environments.

Section 2 deals with related works in context-aware software platforms and possible types of adaptations. Section 3 focuses on our contribution, i.e., the ASSACR framework. In this section, we will detail our smart semantic-based context-aware service selection strategy. Section 4 describes our adaptation platform architecture, i.e., the Kali-Smart platform. Section 5 validates our proposal, and Section 6 concludes the paper with some future works.

2. Related Works

The first related area of research is some platforms involving the adaptations of component-based applications referring to the evolving needs of the users and the execution context by exploiting event-condition-action rules (e.g., WComp [4], MUSIC [5], OSGi [6], Kali2Much [7], Kalimucho [1]).

MUSIC [5] is the most well-known autonomous platform supporting self-adaptative mobile and context-aware applications. This platform can be adapted to the dynamic changes of the environment (e.g., location, network connectivity) in order to satisfy the user requirements and device properties (battery, memory, CPU). The adaptation process defined in MUSIC is based on the principles of planning-based adaptation. This work has not taken into account the multimodal aspects for user-machine interaction and the contextual information that can be gathered by bio-sensors and the distributed action mechanism. WComp [3] proposed a powerful framework for adapting multimodal mobile services in a pervasive computing environment by constructing a private mobile service adaption agent for each mobile user in the cloud. The main drawback of such a platform is the distributed adaptation action mechanism and some smart multimodal event detection mechanisms (Usb (Universal Serial Bus) inputs/outputs, social inputs, etc.).

Recently, Da et al. [7] have proposed a context management middleware Kali2Much to provide services dedicated to the management of distributed context at the semantic level on the shared domain. This work offered excellent smart service management and a predefined policies' deployment strategy, but disagrees in the user-defined policies and did not consider the prediction of user context changes.

Another interesting work is SenSocial [8], which defines middleware for integrating online social networks and mobile sensing data streams. This middleware is based on social networks for capturing and filtering context data. It proposes a generic solution to manage and aggregate context streams from multiple remote devices. This work provides richer contextual information from online social networks and did not take into account the semantics of services and that of the category, QoS and context constraint.

More recently, Taing et al.'s [9] work was based on the Context Toolkit infrastructure; it supports the change of XML files and fire events to an unanticipated adaptation component that can be associated to fully described situations, including time, place and other pieces of context. This work uses a transaction mechanism to ensure uniformly-consistent behavior for every smart object executing inside a transaction and supports only a notification as an action type without multimodality aspects that can be triggered as a result of situation identification and smart event detection.

The second related areas of research are multimodal adaptation projects [10–13]. However, these works lack a new way (i.e., using a detection function to detect modality (touch, gesture, voice) to activate it in another device) to respond to the user needs in a dynamic intelligent way.

Roberto Yus et al. [10] proposed a system that processes user requests continuously to provide up-to-date answers in heterogeneous and dynamic contexts according to the locations of the mobile users to offer customized information. Ontology techniques are used to share knowledge among devices, which enables the system to guide the user to select the service that best fits his/her needs in the given context. This work is efficient and shared between different services using OWL as the best candidate to describe and format it. However, SHERLOCK does not support multimodal standards for representing the application-specific semantics of user input.

Etchevery et al. [11] intended to focus on the Visual Programming Language (VPL), which allows designers to specify the interactions between the system and users who have a minimum computer-science background. The power of VPLs is the possibility of interpreting the meaning of diagrams in order to automatically generate executable code; each interaction is described by a diagram specifying the user action initiating the interaction, as well as the system reactions. VPL could be profitable for touristic or educational purposes. However, this work lacks the semantic expressiveness and efficient context management.

Primal Pappachan et al. [12] proposed (Rafiki: a Semantic and Collaborative approach to Community Health-Care in Underserved Areas) a system for mobile computing devices, which guides community health workers through the diagnostic process and facilitates collaboration between patients and healthcare providers. Interactions in community healthcare could be done by the Internet-based approach or a peer-2-peer (P2P) approach. Semantic context rules specify the desired reactive behavior and can be manually defined by designers or generated by applications.

To improve the interactions between machine and users, Joyce Chai et al. [13] have developed a tool called (Responsive Information Architect) RIA: A Semantics-based Multimodal Interpretation Framework for conversational Systems. The idea is whatever the use input (text, gesture, voice), the input should be interpreted according to the user situation and desires. The work proposed an event-condition-action rule specification that satisfies the functional requirements of covering the five semantic dimensions and handling various data collected from sensor devices and smartphone middleware, as well as supporting composite contexts.

All of the above described related works provide a mechanism for dealing with the inherent heterogeneity and complexity of ubiquitous environment. Hence, to compare to our proposal (see Table 1), they do not: (1) provide a distributed action mechanism with smart multimodality aspects and a service prediction strategy that will give the application the flexibility and dynamicity needed to run through the user's environment, which, to our knowledge, have not been proposed yet in this field; (2) support migration of context middleware components (i.e., event detection, situation reasoner, action mechanism) in a transparent and uniform way; and (3) distributed/centralized context monitor and semantic event detection in order to manage relevant information that could be important for the context regardless of its source. In addition, we implement a centralized semantic context reasoner making the decision a centralized process that will be handled by the main host (e.g., cloud, server, etc.). This choice is meant to prevent the redundancies of adaptation decisions.

Table 1. Related works' comparison.

Related Works	Context Monitor and Event Detection	Smart Multimodality Event Detection	Situation Reasoner	Action Mechanism and Prediction
[13]	Centralized	None	Centralized	None
[1,7,9,12]	Distributed	None	Centralized	Centralized
[8]	Centralized	Social mechanisms	None	Centralized
[4,5,12]	Distributed	multimodal	Centralized	Centralized

The combination of the Kalimucho middleware [1], mobile computing and IoT, with ontologies and rules-based-approaches, provides a new design approach for context-aware healthcare systems. In addition, the context model that we have proposed allows not only representing and reasoning about contextual information, but also providing a generic and flexible model to the developer that facilitates modeling the context and developing context-aware system.

We extended our previous works [3] by allowing a user to define his or her preferences to generate a primary context configuration as desired with a distributed action mechanism with multimodality aspects and a service prediction strategy. We propose an autonomic and dynamic service management that monitors, analyzes, plans and optimizes the service latency delay and maximizes service reliability. ASSACR (Autonomic Semantic Service Adaptation Controller and Reconfiguration), provides an automatic discovery of equivalent multimodal services/duplicated adaptation paths by analyzing the execution context and makes relevant semantic services for multi-heterogeneous mobile devices and many more one-cloud services for all full use, connected mobiles devices and mobile objects being shared every time.

3. Kali-Smart: Autonomic Semantic-Based Context-Aware Adaptation Platform

Kali-Smart extends the Kalimucho platform [1] to find the appropriate service/device with the multimodality aspect. Mainly, the platform should continuously monitor the all of incoming sensor events and immediately detect a certain abnormal situation from them. Furthermore, the platform needs to predict context changes for various user-oriented application services. As the solution to this, we extended the existing Kalimucho platform to be seamlessly integrated with the incremental service prediction strategy and the distributed adaptation action mechanism. This strategy encouraged us to efficiently minimize service switching risks, which may cause traffic congestion and a long blockage of mobile device.

Kali-Smart is based on a distributed semantic context monitor in order to manage important context information according to the current user's need when moving in his or her smart environment. We opted for a centralized context reasoner making the decision in a centralized process that will be handled by the main host of services (e.g., server, computer, cloud, etc.). This choice is meant to prevent redundancies of adaptation decisions. Nonetheless, we will defy the above related works by implementing a distributed action mechanism that will give the application flexibility and dynamicity.

3.1. Platform Architecture

Here is a general overview of the on the fly smart semantic-based services adaptation architecture (Figure 1). It is composed of four layers (Figure 1):

1. The **Knowledge Base (KB) Management Layer** is the part of the platform that is responsible for the representation of the context information in OWL-DL(Description Logics). In our approach, KB consists of the central component, which corresponds to a knowledgebase of different users' profiles, semantic multimodal services and reconfigurations files and a knowledgebase of semantic service description. We exploit most of the capabilities that OWL provides, such as reusability, sharing and extensibility, in way to offer wide representation of the context in smart-* domains.
2. The **Semantic Context-aware Services Management Layer** is responsible for managing the adaptation process and providing a continuous response to a user by adapting dynamically its context provisioning paths according to the change happened during its execution. This core layer relies on the following components:

    - The **User Context Manager** is responsible for capturing user context changes and storing these in a KB repository. The user context is enriched from different explicit constraints and implicit resource characteristics, services with various modalities and shared multimedia documents. This component includes:

        - *Semantic Constraint Analyzer*: interprets the profile, which expresses the users' preferences (e.g., explicit constraint), context information about the device (memory size, battery level, CPU speed, screen resolution, location, etc.), supported documents (media format, media type, content size, etc.), network characteristics (bandwidth, protocol type, etc.).
        - *Constraint Translator*: converts some semantic users' constraints specified in qualitative terms into triplet pattern in the OWL format.
        - *Context Collector*: collects static and dynamic context (user's information, device's information and sensor data) via different interfaces. The collected information will be integrated to make the low-level context and stored as the XML format.
        - *Context Pre-Processor*: is responsible for analyzing sensor data in order to remove duplicated data und unify sensor data that can have different measurement units.
        - *Context Translator*: is responsible to convert the context data into a triplet pattern in the OWL format.

- **Services Context Manager** is responsible for extracting and storing the context service multimodal description (gesture, voice, pen click and mouse click) with semantic context service constraints in a service repository. It is enriched by various QoS parameters (media quality, execution time, etc.).

3. **The Autonomic Semantic Service Adaptation Controller and Reconfiguration (ASSACR) Layer** supports the building of personalized and adapted process from available adaptation services. It is designed to follow an incremental dynamic strategy for provisioning services regarding the context changes during execution. We takes user's current context (battery level, CPU load, bandwidth, user location) and all of his or her surrounding resources as the input (local, remote). Solving the single user constraint in a low service space (local, neighbors) is easier and less time consuming.

- **Context Monitor:** is responsible of verifying the user's context change by contacting the *User Context Manager* component.
- **Context Reasoner:** is responsible for making inferences on the ontology information, determining and deploying the appropriate services according to the deduced situations. This component includes:
    - *Situation Reasoner*: This is based on the JESS (Java Expert System Shell) inference engine [14], executes the SWRL (Semantic Web Rule Language) situation rules and infers the current situations. The SWRL situation rules include user-defined rules written and executed on the available context information in order to infer additional knowledge and conclusions about it. We choose to follow the rules-based approach for the basic advantage that a user can simply reconfigure the behavior of the environment by defining new rules, without having to change the source code.
    - *Action Reasoner*: which is based on the JESS inference engine, as well [15], executes the SWRL service rules and determines the appropriate actions according to the deduced situations.
    - *Prediction Reasoner:* is responsible for generating predictions about the activity and situations happening in the environment.
- **Service Controller** is responsible for re-routing the data transfer to another path when there is poor QoS, connection failure, low battery, etc., in order to ensure service continuity and perform reconfiguration changes at the Kalimucho server.
    - *Automatic Semantic Service Discovery*: all gateways and services in the same local network are automatically discovered by sending SNMP (Simple Network Management Protocol: which represents the management information control that describes the router state) broadcast. Only mobile devices hosting the Kalimucho platform with SNMP will send a response to the SNMP request broadcast by ASSACR. SNMP responses will be processed by ASSACR to establish the neighbors/services individuals of the *OWL Host and Services Classes* in the ontology repository.
    - *Semantic Service Analyzer:* analyzes equivalent semantic services based on location, time and activity with different qualities (execution time, modality input/output). Two cases are possible for equivalent semantic services with different qualities: (1) semantic equivalent services with different service qualities at the same Kalimucho gateway; in this case, redundancy is supported by the Kalimucho gateway; and (2) semantic equivalent services with different service qualities selected from two (or more) gateways. In this case, redundancy is supported by the Kalimucho server.
    - *Reconfiguration Generator:* SNMP is a simple protocol to ensure dynamic service reconfiguration; ASSACR can automatically create configuration files to be saved in a Kalimucho repository.

- *Service Deplorer:* the configuration file created in the previous phase is copied in the Kalimucho cloud server, and all mobile devices affected by the equivalent path and the new services configuration are saved.

4. The **Kalimucho Platform Layer:** offers service-level functions: the physical (re-)deployment and dynamic incremental reconfiguration strategy according to its system (Android, laptop with QoS requirement, dynamic supervision of adaptation components and communication protocols between mobile nodes).

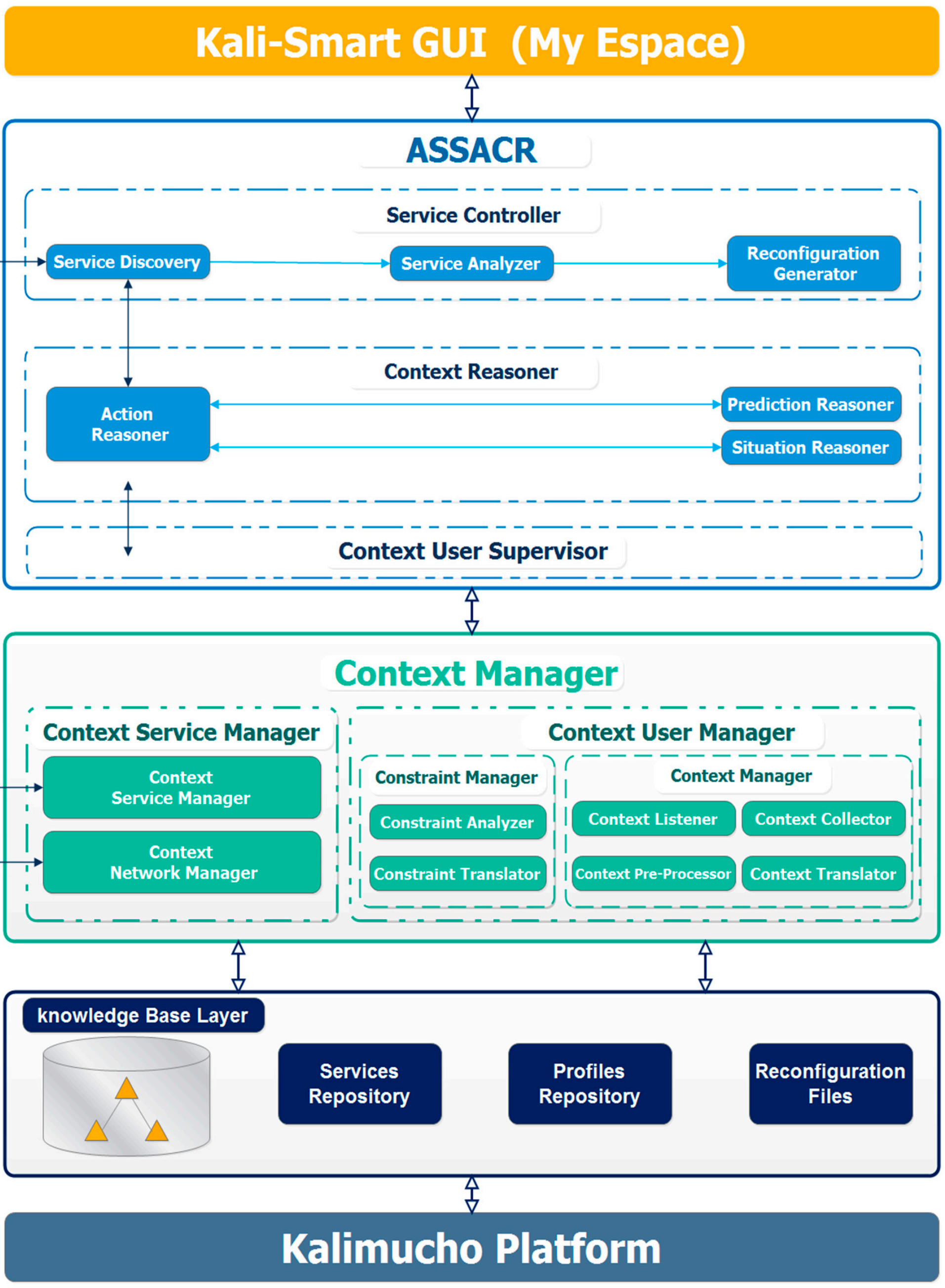


Figure 1. Kali-Smart: platform architecture. ASSACR (Autonomic Semantic Service Adaptation Controller and Reconfiguration).

3.2. Functional Model of the Smart Semantic–Based Context-Aware Adaptation Platform

The objective of the smart semantic-based context-aware platform is to provide a suitable service to users. This service has to fit with the usage context, such as: physical characteristics of the device, the bandwidth, the user's wishes, etc. An overview of the functional model of the smart semantic-based context-aware platform is provided in Figure 2.

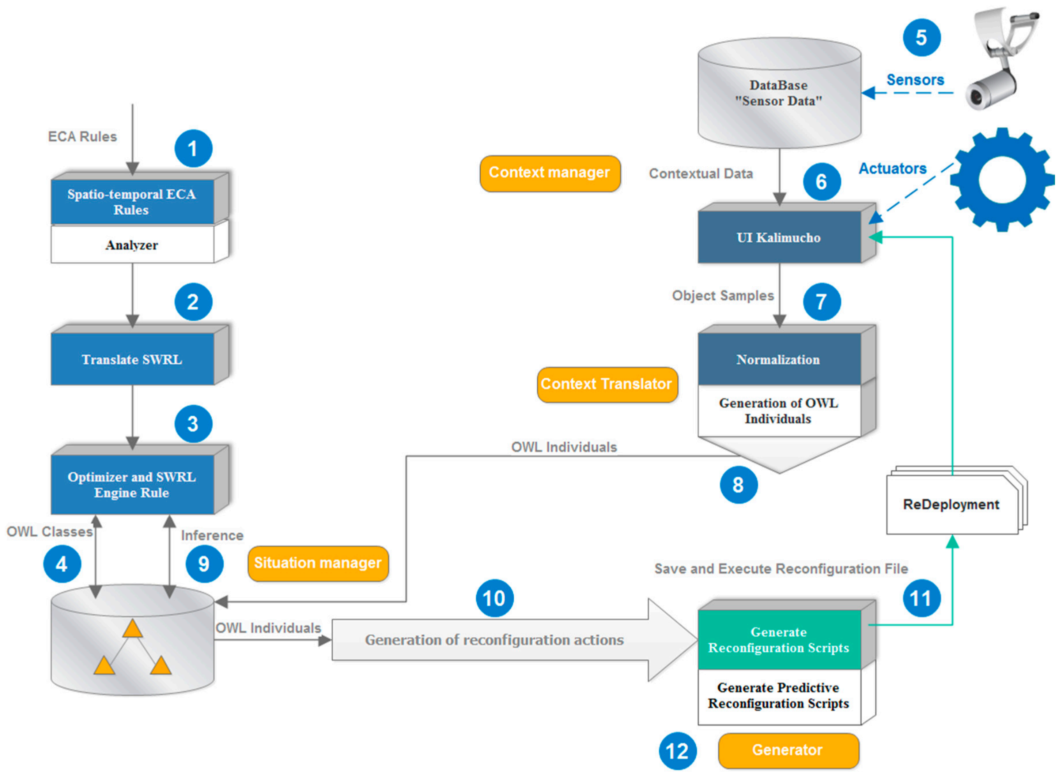


Figure 2. Model of the Kali-Smart adaptation platform. ECA (Event Condition Action); SWRL (Semantic Web Rule Language); OWL (Web Ontology Language).

As shown in Figure 2, the pervasive environment represents user, sensors, smart devices, smart objects and services. The pervasive environment is supervised by the *Context Monitor and the Service Controller*. The Context Monitor collects contextual information from the pervasive environment, which represents row and heterogeneous data, analyses and interprets these data with respect to high level information, which will present the adaptation of the system behavior and will be reported to the Service Controller. The Service Controller is used to generate the chain of quality services according to the situations raised. The Service Deployer deploys and executes the service chain.

In the following part, we give more details about the context model of our architecture.

4. Ontology and Rules-Based Context Model

4.1. Context Modeling

The main objective of our approach is to improve the efficiency and accuracy of users' adaptations tasks. This objective is achieved through finite sets of semantic relevant adaptation services and various users' contexts. A user has a context in which he or she wishes to adapt his or her multimedia documents within a specific activity in a known time and location using one of the offered modalities; any smart service can be used in a local way or using the cloud, which allows him or her to handle the data storage that he or she needs to run his or her applications.

In order to facilitate the conception and the development of our ontology, we divide it into four hierarchical levels: (1) *Contextual Information level*; (2) *Contextual Situations level*; (3) *Contextual Services level*; (4) *Contextual Constraint level*. These levels contain seven main classes, which are: ***Context class***, ***Event class***, ***Situation class***, ***Service class***, ***Context Constraint Class***, ***ContextProperty class***, ***ContextPropertyLogicValue class***. These classes represent generic concepts, which can be used in any pervasive context-aware distributed mobile application that aims to provide appropriate services to the user according to the current situations.

4.1.1. Context Sub-Ontology

We divided the context sub-ontology into seven sub-contexts (see Figure 3).

- The *RessourceContext* describes the current state of the hard equipment and soft equipment (memory size, CPU speed, battery energy, etc.).

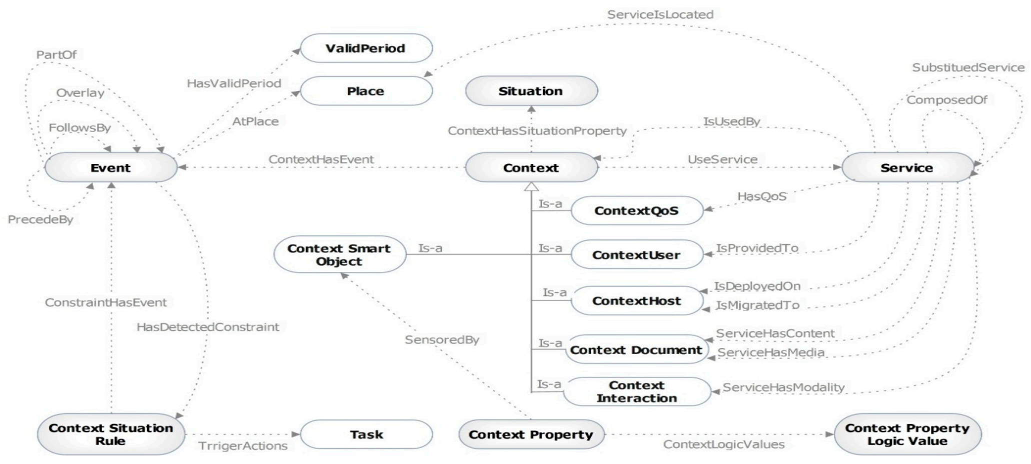


Figure 3. An overview of the structure of the common context ontology.

- The *User Context* describes information about the user, which can alter the adaptation service. User preferences include user age, preferred languages and preferred modalities (voice, gesture, pen click, mouse click, etc.). A user can select which multimedia object can be adapted (image, text, video or audio), for example if he or she receives audio when he or she is at work, he or she would rather receive a text instead; that means we need an adapting service to change the audio to a text. We can find also a description of the user's health situation, as the user can be healthy or handicapped (Figure 4).

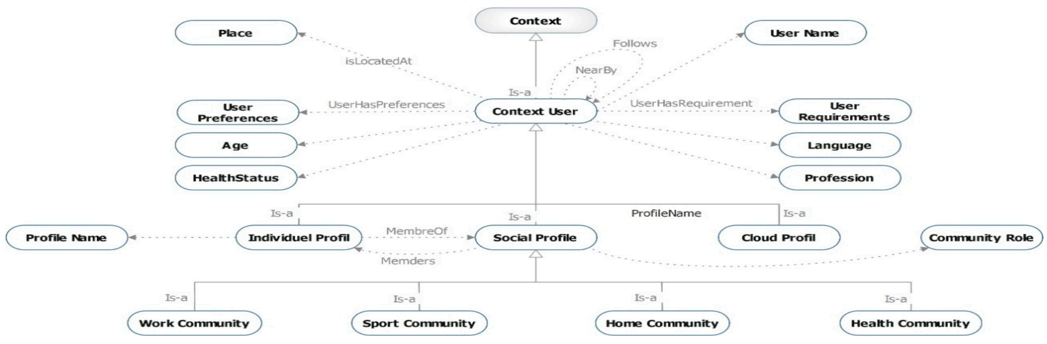


Figure 4. User context sub-ontology.

- The *Smart-Object Context* describes data that are gathered from different sensors and describe orchestrated acts of a variety of actuators in smart environments (Figure 5). We have three types of sensor data: (1) bio-sensor data represent data that are captured by bio-sensors, like blood pressure, blood sugar and body temperature; (2) environmental sensor data represent data that are captured by environmental sensors, like home temperature, humidity, etc.; (3) device sensor data represent data that are captured by sensors, like CPU speed, battery energy, etc.

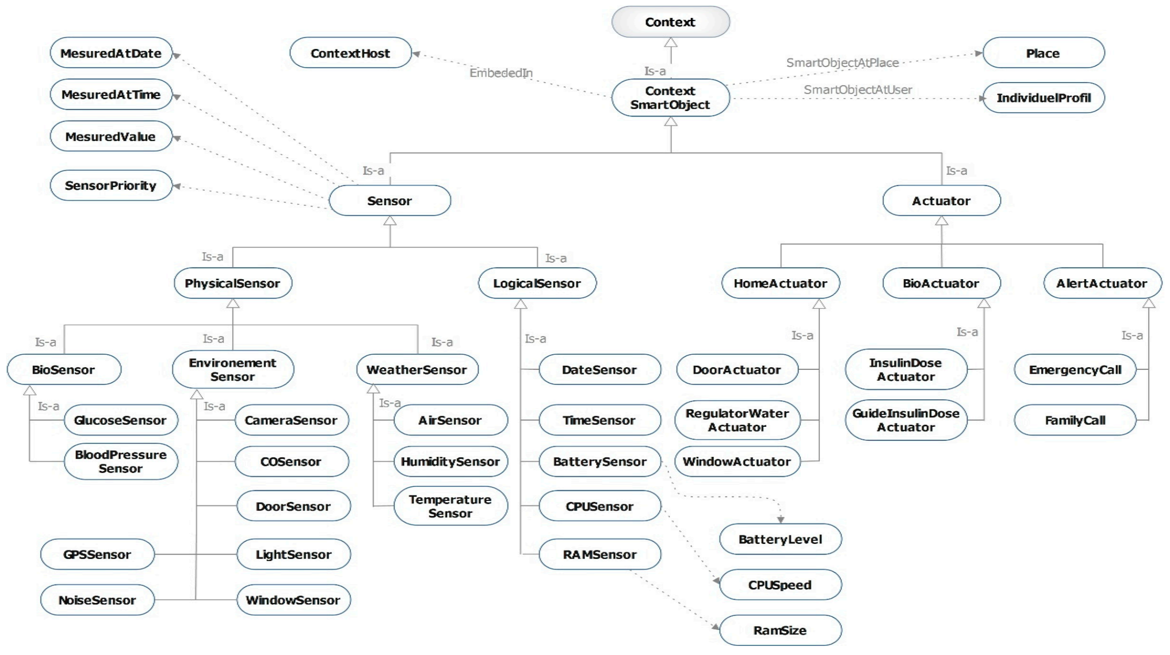


Figure 5. Smart-object context sub-ontology.

- The *QoS Context* describes the quality of any mobile-based application, which can be represented in our ontology, which is defined as a set of metadata parameters. These QoS parameters are: (1) continuity of service; (2) durability of service; (3) speed and efficiency of service; and (4) safety and security (see Figure 6).

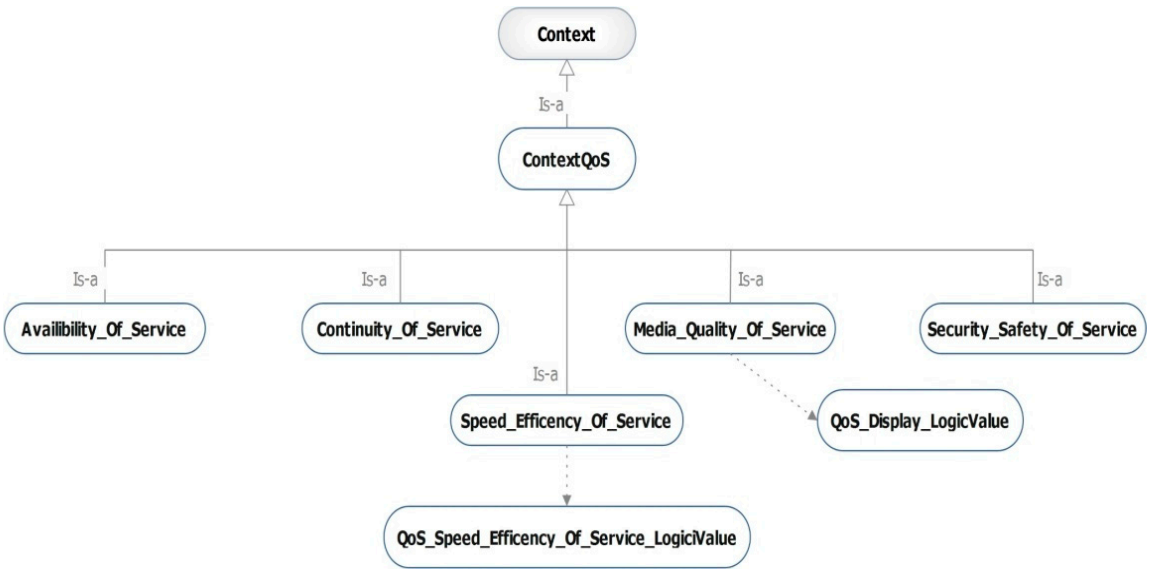


Figure 6. QoS context sub-ontology.

- The *Host Context* represents different hosts of services proposed by providers. For example, services can be hosted on local devices or on the cloud. The local device class contains information about fixed devices or mobile devices. Mobile devices have limited resources, such as battery, memory and CPU. The Cloud class contains information about the cloud server (e.g., Google cloud) that can be used for hosting services. The service is deployed and migrated on the host, and as the service has constraints, so a substitution of the service location could occur (a possible scenario is when the battery level is low, the service should be migrated on the cloud, so that the data could be stored separately and that could help minimize the use of energy). As mobile limited resources can break the mobile services, we are looking to the cloud or resources in proximity as a way to ensure the service continuity on mobile devices (see Figure 7).

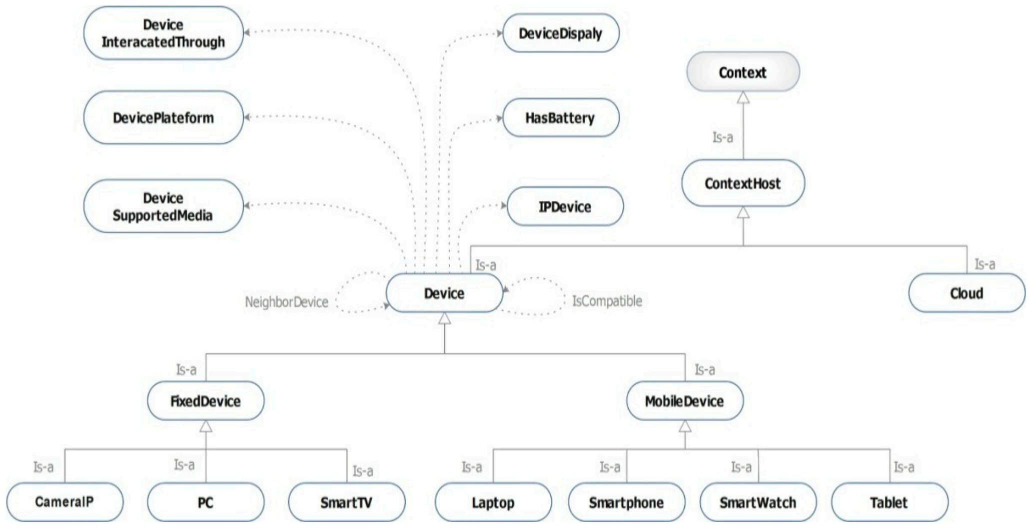


Figure 7. Hard context sub-ontology.

- The *Environment Context* describes spatial and temporal information (Figure 8):
    - Temporal information can be a date or time used as a timestamp. Time is one aspect of durability, so it is important to date information as soon as it is produced.
    - The *Place* describes related information about the user's location {*longitude*, *altitude* and *attitude*}, in a given location, where we can find available mobile resources. The mobile resources are mobile devices, such as tablets, smartphones, laptops and smart objects, such as bio-sensors, environment sensors and actuators, etc. Resources are accessible by users.
    - The *ActivityContext*: according to a schedule, a user can engage in a scheduled activity.

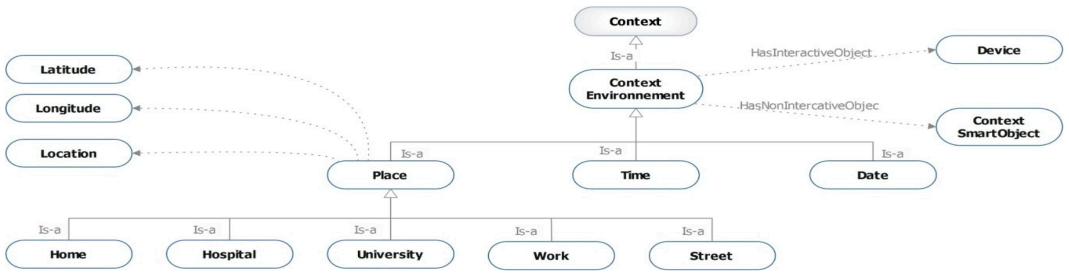


Figure 8. Environment context sub-ontology.

- The *Document Context* describes the nature of the documents (text, video, audio). The document context specifies a set of properties related to a specific media type: (1) text: alignment, font, color, format, etc.; (2) image: height, width, resolution, size, format, etc.; (3) video: title, color, resolution, size, codec etc.; (4) sound: frequency, size, resolution, and format (Figure 9).

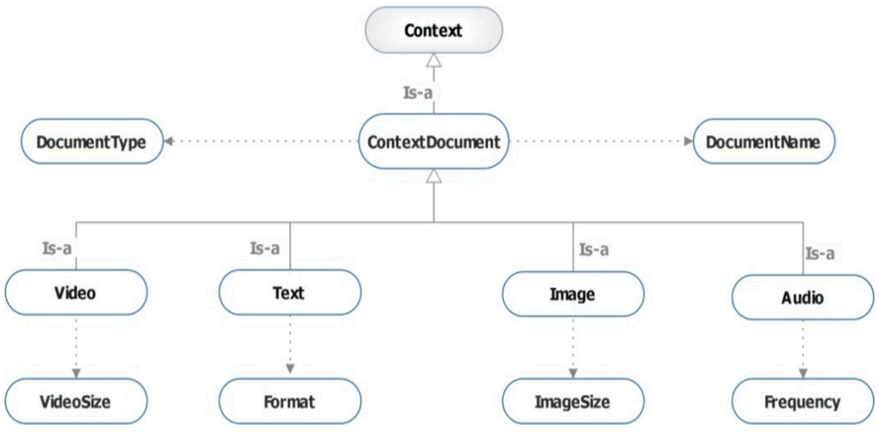


Figure 9. Document context sub-ontology.

4.1.2. Context Constraint Ontology

A context constraint is defined through the terms context parameters and context expression, which is further categorized by simple expression and/or composite expression, thus forming a multi-level context ontology as shown in Figure 10.

- **Context_property:** Each context category has specific context properties. For example, the device-related context is a collection of parameters (memory size, CPU power, bandwidth, battery lifecycle, etc.). Some context parameters may use semantically-related terms, e.g., CPU power, CPU speed.
- **Context_expression:** denotes an expression that consists of the context parameter, logic operator and logic value. For instance: *glucose level = 'very low'*.
- **Context_constraint:** consists of simple or composite context expression. For example, a context constraint can be *IF glucose level ='very high' and Time= 'Before Dinner' and Location='Any' Then Situation= Diabet_Type_1_Situation*.

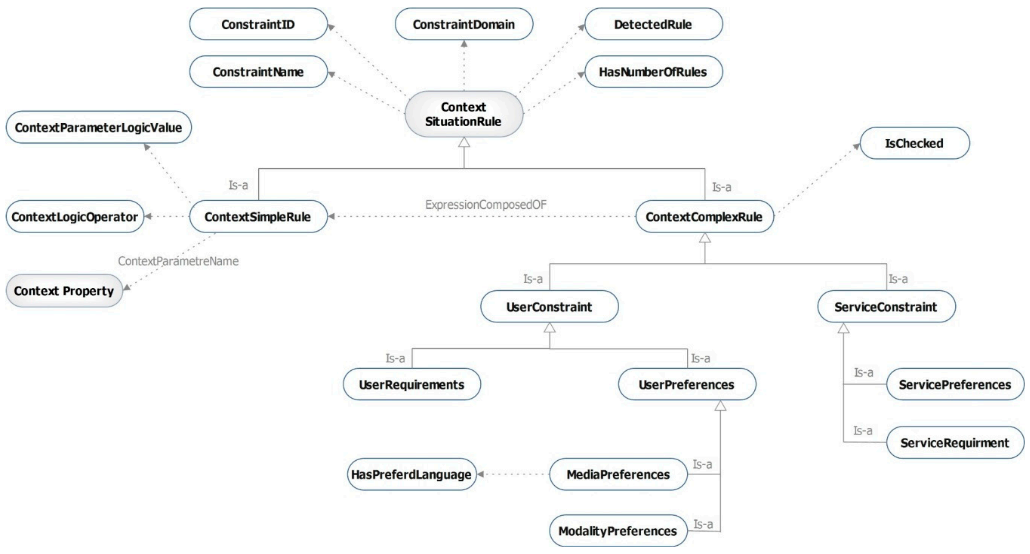


Figure 10. Context constraint sub-ontology.

4.1.3. Service Ontology

Nowadays, environments are getting smarter in order to reply to user requests anytime and anywhere according to his or her location; interaction between users seeks to get better services from providers. Service could be either smart or interactive. Any smart service can be used in a local way or using the cloud, which allows them to handle the data storage that users need to run their applications. Interactive services are unimodal and multimodal interactions. A service can be executed in various forms with various quality of experience and quality of service. Each user expects his or her own QoS when using a service. To ensure the QoS level, the service has its specific mobile device constraints (size of memory, CPU speed and battery lifetime). We can find also three services types: Smart Service, Interactive Service and Adaptation Service (Figure 11).

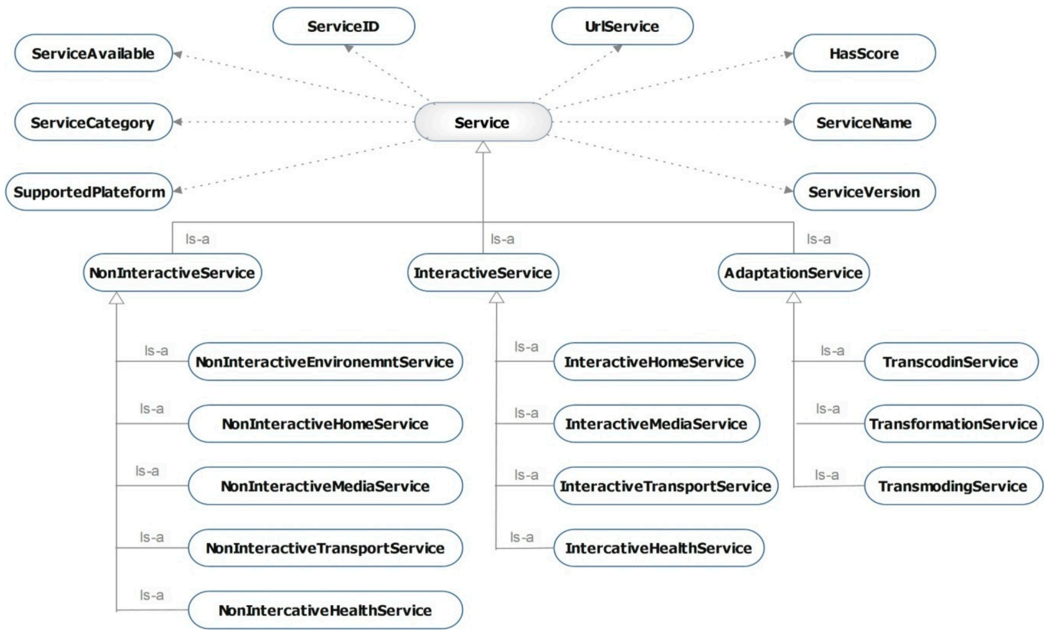


Figure 11. Service context sub-ontology.

4.1.4. Context Property Sub-Ontology

Some context parameters can use semantically-related terms, such as processor speed. Each parameter is described by the *ContextPropertyLogicValue* class to which is assigned a range of qualitative values (*Context Min Value*, *Context Max Value*); these values make it possible subsequently to determine the quantitative values (see Figure 12).

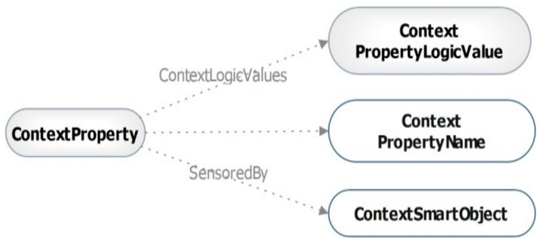


Figure 12. Context property sub-ontology.

4.1.5. Situation Ontology

This part represents the possible situations that we can define for each context. For instance, the home temperature context can have three situations: normal, cold, hot and very high situations.

We call these situations Contextual Situations since they depend on contextual information. This class contains two subclasses: *External-Situation class* and *Internal-Situation class*. The first one represents situations that are related to the user's environment and user's devices, such as home temperature situations and battery situations. The *Internal-Situation class* represents situations that are related to a specific domain, such as the person's health state, like blood sugar situations and blood pressure situations. Each situation has data type properties (see Figure 13), such as situation type, max value and a min value, that are defined by the developer and by the domain expert; in our work, the physician. An example of the situation rule is as follows.

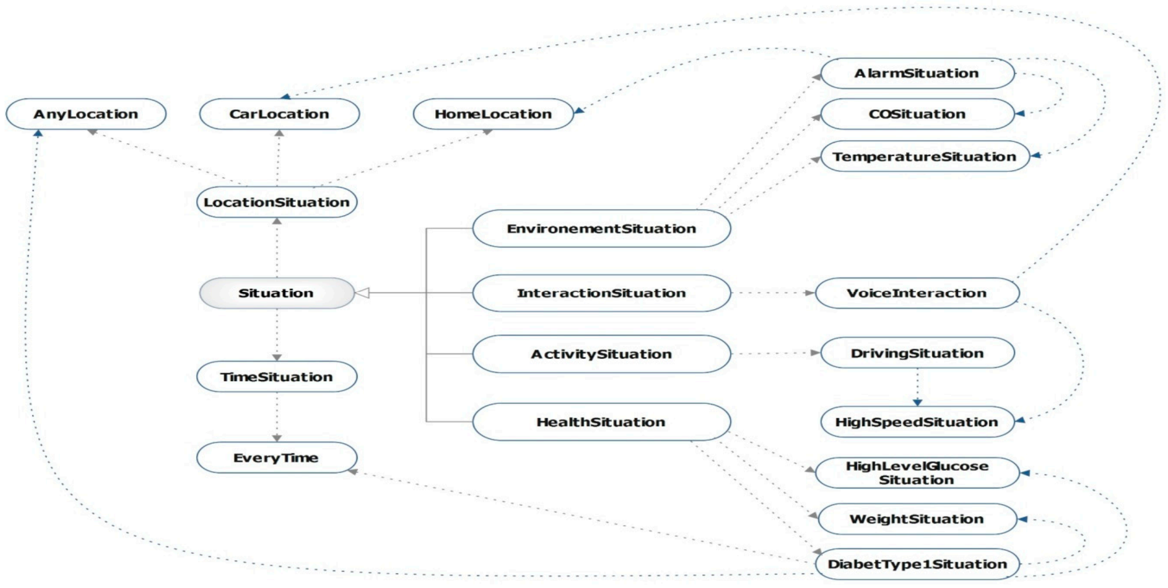


Figure 13. Situation sub-ontology.

4.2. Dynamic Situation Searching

There are several possible ways to identify a situation [16–19], and a proper identifying generic technique still has to be defined. Computing similarities is also quite difficult when integrating heterogeneous user profiles. Several services composing a situation are generally connected semantically. Our similarity measure extends the properties of *Sim* defined in [19]. It is formalized as follows:

$$Sim(Q,S) = \frac{a}{a+b} = \frac{a\sum_{i=1}^{a} w_i * sima(Q_i, Sj)}{(a\sum_{i=1}^{a} w_i) + \left(b\sum_{i=1}^{b} w_{a+i}\right)} \tag{1}$$

where "a" is the set of common concepts of Qi (*current situation of user*) and Sj (profile *constraint*) for the same domain and "b" is the set of concepts of Qi and not existing in Sj. *sima* is the atomic similarity between each context concept of situation Q, S. It is defined as a function that maps two concepts to the interval [0, 1]. First of all, local events have to be detected and local situations identified. If no events are detected, it should be detected whether events are in nearby mobile devices by knowing their exact situations and re-deploying distributed interactive services. The proposed algorithm (Algorithm 1) aims at matching the current situation with each constraint defined in the user's profile. It takes a list of current situation's concepts as input in order to calculate the atomic similarity sim_a of each pair of concepts (of situation and constraint). If the match is not a *Fail*, it returns the overall score *Sim* of each constraints and appends the advertisement to the result set. Finally, the result set is returned in ranked way. In so doing, we select the *higher* matching measure.

```
Algorithm 1: Situation Matching and Dynamic Service Selection
Input: Profile [], profile Status[],two contexts $C_1$ and $C_2$/$C_1$S and $C_2$ Q    // set of profiles and profile status
Output: Overall score Sim (Q,S)    // best semantically equivalent services
        Matched_Relation $\leftarrow \phi$    // EXACT, SUBSUMES, NEAREST-NEIGHBOUR, FAIL
        MatchedService_List $\leftarrow \phi$    // a set of services that meet the user's context preferences
            1. Matching preferences with local services and get each constraint similarity value defined in [1]:
                    $MatchedService\_List = MatchedService\_List \cup S_{Local};$
            2. Matching preferences with nearby services and get each constraint an overall similarity in [1]:
                    $MatchedService\_List = MatchedService\_List \cup S_{NearBy};$
            3. Selection of quality equivalent semantic services defined in [11].
            4. Generation of reconfiguration file with k-best services.
            5. Save new reconfiguration file in Kalimucho Server.
```

4.3. Context Provisioning

In order to be successful, a context-aware multimodal mobile application must have a full visibility of the person's context, including what, when and how to monitor, collect and analyze context data. In the case of changes according to the user context (e.g., less available memory, less processor power, change of user location), or the environment context (e.g., less bandwidth), or the user preferences, the ASSACR component will be provisioned by a set of contexts metadata in order to dynamically:

- Provision the next substituted service of the list of services offering the same service functionalities, which require less available resources and sorting the QoS of available services.
- Add new services into the list of found services. The new service is matched from the new provisioned changes in the profile constraint (e.g., the user context).

We consider each sub-context profile and its constraints. Solving a single user constraint in a low service space (local, neighbors) is easier and less time consuming. Therefore, we can start services discovering the optimum cost and good provisioning services chain by a combination of reconfiguration operations: create/update/migrate/remove from some context attributes changes with a single performance and approach the global optimum service reconfiguration chain.

Our approach divides the all of user context changes into several phases (Algorithm 2). One more preference (expressed as an *Event-Condition-Action rule*) is considered in each phase until the global reconfiguration chain is provisioned and generated. First, a set of services is discovered and matched with a certain concerned context conditions on some changing context adaptively, called *Single-Increment-Context-Evolution*. Secondly, we investigate the better quality services obtained from Step 1, and services from provisioning context attributes changes (low bandwidth, battery life time, etc.) are joined together in Step 2 to form an initial configuration, for which there are multi-concerned context constraints.

For an efficient and better provisioning and good management of the self-management adaptation process, we have used the "Poisson event-based" simulation model to predict context changes (i.e., mobility of users, usage resources), then we have provided the global system's behavior that requires adaptation and overcoming the rising complexity of the context-aware system.

Our process predicts minimum adaptive cost in low adaptation time, to result in the best configuration, such that the system quality is maximized subject to device resource constraints and user preference constraints. Firstly, the platform will be able to restrict the scope of the search into the range of configurations, which differ from the current configuration only by the service at the origin of the reconfiguration event. However, when this approach does not give any solution, we look for a new configuration to use, starting by switching from a new relay or by moving a service to a suitable device at run-time. The ASSACR generates a provisioned configuration model. When the reconfiguration is triggered, we start-up the provisioned saved reconfiguration.

Algorithm 2: Incremental Context Changes Prediction

```
Input: Predict user context changes
Output: Generates provisioned Services reconfiguration file
    1. Set $p$ = 1; where $p$ is the phase number; find a provisioned quality services list with regard
       to the first concerned context attribute change.
    2. Set $p = p + 1$. The next phase starts.
    3. Generate a fittest quality services list with regard to the $p$-th concerned context attribute
       change and joined with the $p - \text{-}1^{th}$ concerned context attributes.
    4. If remained context attributes changes, go to 2.
    5. Generation of provisioned reconfiguration file.
```

4.4. Context Reasoning

Context reasoning should take into consideration each user preference by virtue of giving him or her the best service according to the current location, environment constraints, schedule, etc. We have classified our rules into three categories; *Situation Rules*, *Smart Service Rules* and *Constraints Rules*. *Situation Rules* are used to infer situations from contextual information. Some of situations trigger appropriate services that must be provided to the user. However, other situations, like automatic gesture modality detection, can be used for proposing to assure the continuity of the service. Table 2 illustrates examples of the rules.

Table 2. Examples of the rules.

Situation Rules	**EVERY TIME** **IF** User Has Biosensor Data **AND** Glucose Level > Min Value **AND** Glucose Level < Max Value **THEN** Person HAS Situation "DiabetType1"
Services Rules	**EVERY TIME** **IF** Service IS Health-Care **AND** Service Has Type "Service Type" **AND** User HAS Situation Situation **AND** Situation Has Type "Situation Type" **WHERE** "SituationType" IS EQUAL TO "Service Type" **THEN** Situation Trigger Service Service **WHERE** Service isProvided To User
Smart Services Rules	**EVERY TIME** **IF** Service IS Health-Care **AND** Service is Deployed On Device "D1" **AND** User HAS Situation Situation **AND** Situation Is Detected By Detection_Function **AND** Situation Depends On Device "D2" **AND** Detection_Function Has Modality "Modality Type" **THEN** Situation Migrate Service Service **WHERE** Service is Deployed On Device "D2"

5. Potentials Scenarios and Validation

This section describes the implementation details of our Kali-Smart context-aware platform, potential usage scenarios and experimentation results.

5.1. Implementation of Kali-Smart

Firstly, based on OWL languages, we have implemented our ontology model using the Protégé tool. Protégé [20] is an open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. In addition, it implements a rich set of knowledge-modeling structures and actions that supports the creation, visualization and manipulation of OWL ontologies (see Figure 14).

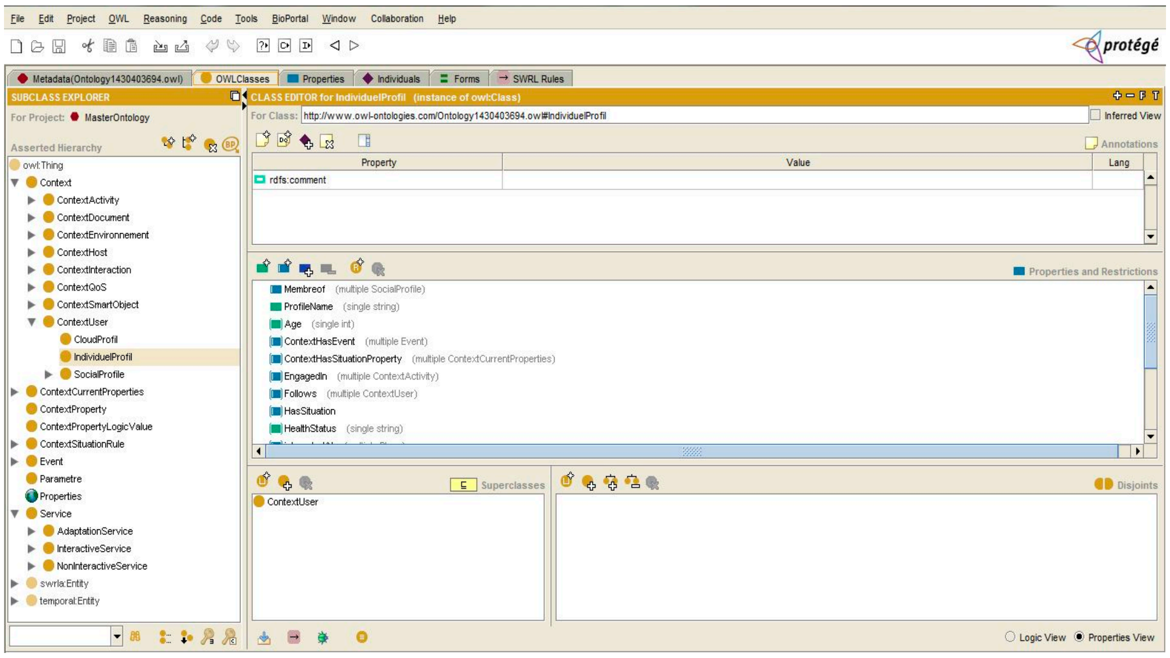

File Edit Project OWL Reasoning Code Tools BioPortal Window Collaboration Help
protégé
Metadata(Ontology1430403694.owl)
OWLClasses
Properties
Individuals
Forms
SWRL Rules
SUBCLASS EXPLORER
For Project: MasterOntology
Asserted Hierarchy
owl:Thing
Context
ContextActivity
ContextDocument
ContextEnvironnement
ContextHost
ContextInteraction
ContextQoS
ContextSmartObject
ContextUser
CloudProfil
IndividuelProfil
SocialProfile
ContextCurrentProperties
ContextProperty
ContextPropertyLogicValue
ContextSituationRule
Event
Parametre
Properties
Service
AdaptationService
InteractiveService
NonInteractiveService
swrla:Entity
temporal:Entity
CLASS EDITOR for IndividuelProfil (instance of owl:Class)
For Class: http://www.owl-ontologies.com/Ontology1430403694.owl#IndividuelProfil
Inferred View
Annotations
Property
Value
Lang
rdfs:comment
Properties and Restrictions
Membreof (multiple SocialProfile)
ProfileName (single string)
Age (single int)
ContextHasEvent (multiple Event)
ContextHasSituationProperty (multiple ContextCurrentProperties)
EngagedIn (multiple ContextActivity)
Follows (multiple ContextUser)
HasSituation
HealthStatus (single string)
Superclasses
ContextUser
Disjoints
Logic View
Properties View

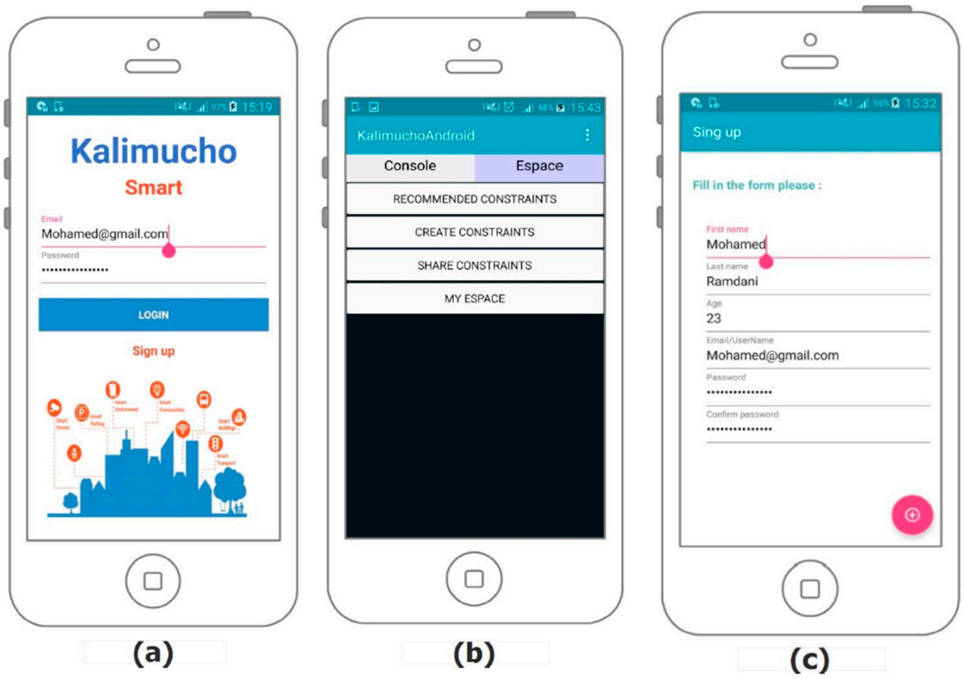
Figure 14. Implementation of our ontology.

SWRL as a rule-based context inference language is used to infer situations that are related to the user's environment and user's devices, such as home temperature situations and battery situations. Each situation has data type properties (see Figure 15), such as situation type, max value and a min value, that are defined by the developer and by the domain expert; in our work, the physician. For example, the High Blood Sugar Type 1 situation is a blood sugar situation that can have "High Blood Sugar Type 1" as a type, 1.45 as the min value and 2.5 as the max value.

Individual_Profile(?p) ^ ContextHasSituationProperty(?p , ?property) ^ HealthSituationProperty(?property) ^ SmartObjectAtUser(?s , ?p) ^ GlucoseSesnor(?s) ^ MeasuredValue(?s , ?v) ^ ContextProperty(?cp) ^ ContextPropertyName(?cp, ?np) ^ swrlb:equal(?np, "Glucose") ^ ContextLogicValues(?cp, ?clv) ^ ContextMinValue (?clv, ?min) ^ swrlb:graterThanorEqual (?v, ?min) ^ ContextMaxValue(?clv, ?max) ^ swrlb:lessThanorEqual (?v, ?max) ^ ContextLogicValue(?clv, ?logicvalue) ^ MeasuredAtTime(?s , ?time) ^ MeasuredAtDate(?s , ?date) ^ IsLocatedAt(?s , ?place)
➔ Glucose Situation (?property, ?logicvalue)

Figure 15. SWRL (A Semantic Web Rule Language) rules for the situation analysis.

A mobile user equipped with his or her smartphone enters into the smart environment. The smartphone automatically connects to Kali-Smart, which will allow it to authenticate (a), to view my space (b) or to create an account (c). The user has indicated that he or she wants to detect *diabetes type 1* and to detect abnormal situations in a given location (detection fire in home) (Figure 16). He or she can follow semantically his or her health status and/or detect abnormal situations in a given location and/or edit and/or update his or her profile.



Email
Mohamed@gmail.com
Password
••••••••••••••••

First name
Mohamed
Last name
Ramdani
Age
23
Email/UserName
Mohamed@gmail.com
Password
•••••••••••••••
Confirm password
•••••••••••••••

Figure 16. Screenshots of the Kali-Smart platform (**a**) authentication, (**b**) my space, (**c**) new account.

5.2. Real-Life Scenarios

We aim at demonstrating that our proposed Context Monitor/Context Reasoner/Service Controller and Ontology-Rule-based architecture has the ability to infer situations and determine services that must be provided to the user according to his or her current situations (at home, working, on the car, etc.) and other environmental factors, such as the current available devices and services.

User A is a diabetic person and needs to monitor his or her health. He or she uses an intelligent environment, which includes smart devices, sensors (glucose meter, weight scale, video camera) and services. This intelligent environment specifies an ambient system that allows him or her to live safely and to identify urgent situations. For example, the situation of "diabetic coma or too high temperature" is deduced by referencing the average glucose level and the temperature level in the profile. To make better informed choices, to help User A manage his or her situations in a more practical way by User B (e.g., doctor) (see Figure 17), User A needs to check his or her blood sugar level. He or she uses a bio-sensor that connects to his or her smart phone via Bluetooth technology. According to his or her blood sugar level, the system will execute the appropriate service. If his or her blood sugar level is out of the normal range determined by his or her doctor, there will be three possible situations (see Table 3). When "User A" takes a shower, he or she activates the faucet, and the "Adjusting water Temperature" service will be executed automatically in order to adjust the water temperature. When "User A" has breakfast, he or she reads his or her personal emails, and so, the email app is deployed and launched, whereas the previous "Adjusting water Temperature" service is removed silently. As he or she enters his or her car, the GPS is automatically connected to the phone; the mail app is automatically enhanced to integrate the vocal reading email functionality, whereas a traffic guide is launched and indicates the best way to go to work. He or she preferred to translate text to speech (see Table 4). As he or she is a bit late, an audio conference is dynamically deployed in order to ensure his or her participation in a scheduled conference that he or she has to attend. When entering the office, the audio conference is migrated to the desktop PC, and the video functionality is added to continue the discussion. In his or her smart office environment, when the system detects abnormal situations representing too high temperature and too low humidity, the "Fire Station Call" service is automatically triggered, and the service notifies the emergency status to the nearby fire station (Table 5).

Table 3. Constraints specified by the doctor.

Location	Situation	Constraint Description
Any	*High Blood Sugar*	**Every time IF** blood sugar level is **high THEN** adjust insulin dose and activate diabetes guide
	Danger Blood Sugar	**Every time IF** blood sugar level is **danger THEN** activate emergency call
	Low Blood Sugar	**Every time IF** blood sugar level is **low THEN** activate diabetes guide

Table 4. Constraints specified by the developer.

Location	Situation	Constraint Description
Any	*Low battery level*	**Every time IF** battery level is **low THEN** migrate service
	High battery level	**Every time IF** battery level is **High THEN** replace service
	Schedule Time	**IF** activity scheduled time is come **THEN** activate service
	Schedule Time	**IF** activity scheduled time is out **THEN** remove service

Table 5. Constraints specified by the user.

Location	Situation	Constraint Description
Car	*Driving situation*	**Every time IF** driving speed is **high THEN** activate text to speech
Home	*Shower situation*	**IF** morning time **AND** bath room faucet is activated **AND** Temperature level is **high THEN** adjust water temperature
	Drink situation	**IF** morning time **AND** breakfast is detected **THEN** activate email
Office	*Work situation*	**Every time IF** Temperature level is **high AND** CO level is **Low THEN** call emergency

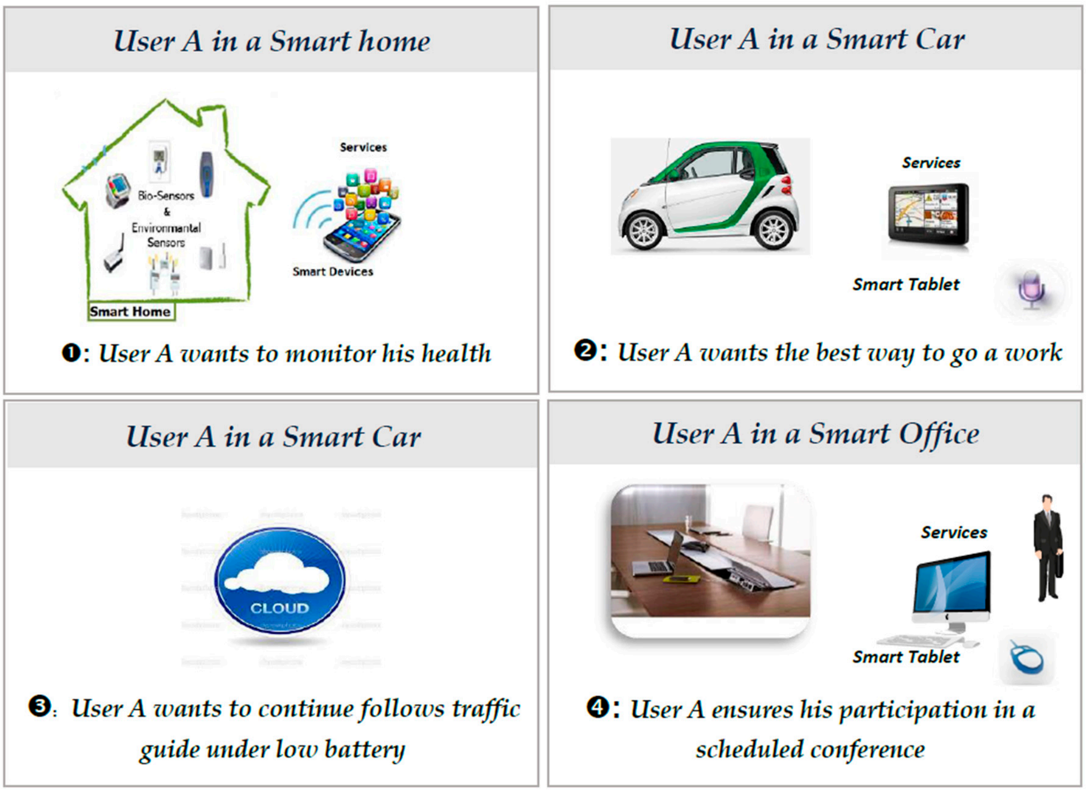
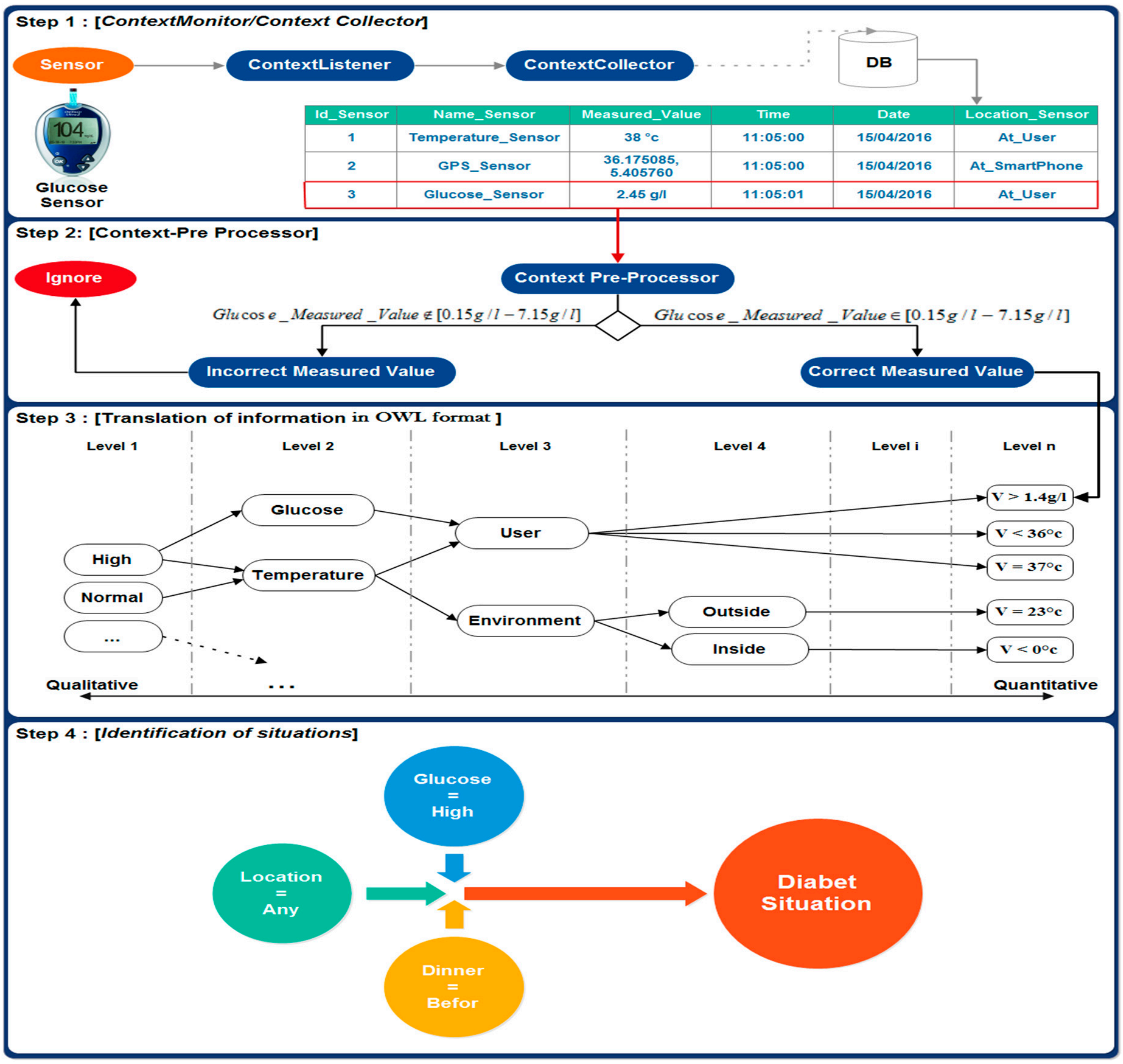

User A in a Smart home
Services
Bio-Sensors & Environmental Sensors
Smart Devices
Smart Home
❶: User A wants to monitor his health
User A in a Smart Car
Services
Smart Tablet
❷: User A wants the best way to go a work
User A in a Smart Car
CLOUD
❸: User A wants to continue follows traffic guide under low battery
User A in a Smart Office
Services
Smart Tablet
❹: User A ensures his participation in a scheduled conference

Figure 17. Potential scenarios.

5.2.1. Scenario I: User A in Home Using Smartphone (or Smart TV or Smart Watch)

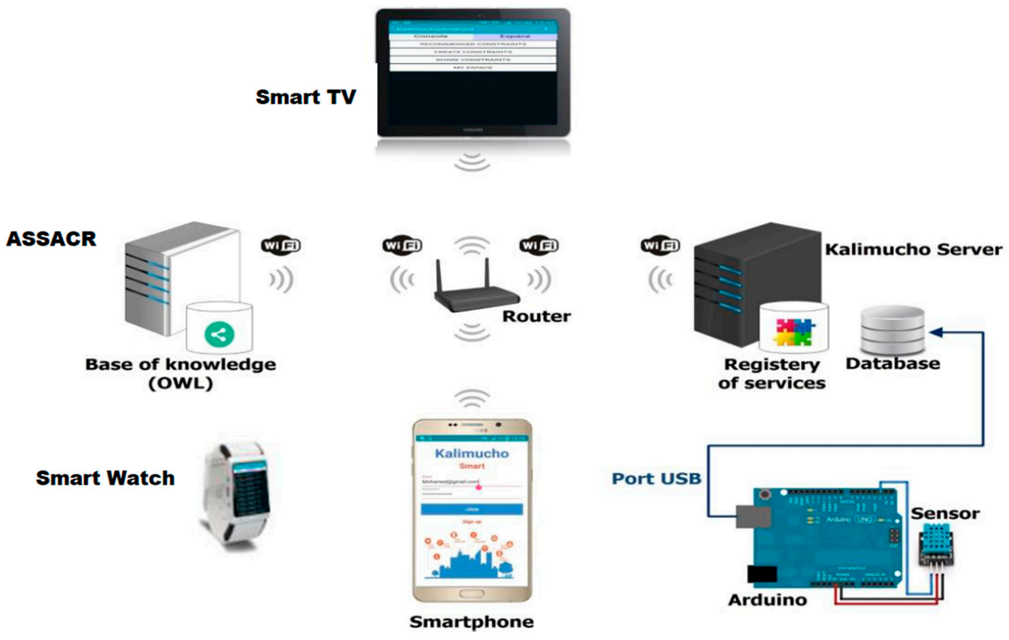
A first simple scenario is made of a User A, which is at *home* using a Samsung smartphone. User A can use our platform to monitor his or her health in a more practical way. He or she uses an intelligent environment (smart home), which includes smart devices, sensors (glucose meter, weight scale, video camera) and services. As proposed in our architecture, the principal role of the *Context Monitor* is to supervise User A in order to detect abnormal situations. The *Context Monitor* collects and preprocesses context information from the smart sensors and stores it in a database. Then, it represents and reasons about this context using our ontology and *Context Reasoner* in order to deduce the current situations that are reported to the *Service Controller*. The latter is responsible for providing the appropriate service to the user according to the inferred situations. In fact, as illustrated in Figure 18: (1) contextual information is collected from the smart home by the Context Collector, and it is stored in a database; this means that the information simulated via the prototype interfaces is stored automatically in a database; (2) this information is pre-processed by the Context Pre-processor and (3) converted into a triple pattern in OWL format and inserted in our ontology by the Context Translator; then, (4) the Context Reasoner uses the Situation Inference Engine, which is based on JESS, which executes the Situation Rules presented in Table 2 (translated in SWRL by *Constraint Translator*) and infers the current situations. These situations are transmitted to the Service Controller.



Id_Sensor	Name_Sensor	Measured_Value	Time	Date	Location_Sensor
1	Temperature_Sensor	38 °c	11:05:00	15/04/2016	At_User
2	GPS_Sensor	36.175085, 5.405760	11:05:00	15/04/2016	At_SmartPhone
3	Glucose_Sensor	2.45 g/l	11:05:01	15/04/2016	At_User

Figure 18. Ontology/rule-based architecture simulation.

Once User A is connected and clicks "**My Space**", the Kali-smart platform sends the initial actions configuration file to the Kalimucho Server, launching automatically the GPS location service. The script bellow is generated by the reconfiguration generator after scanning the environment with the help of Kalimucho in order to know the available devices and to orchestrate the best way to deploy the components (Figure 19).



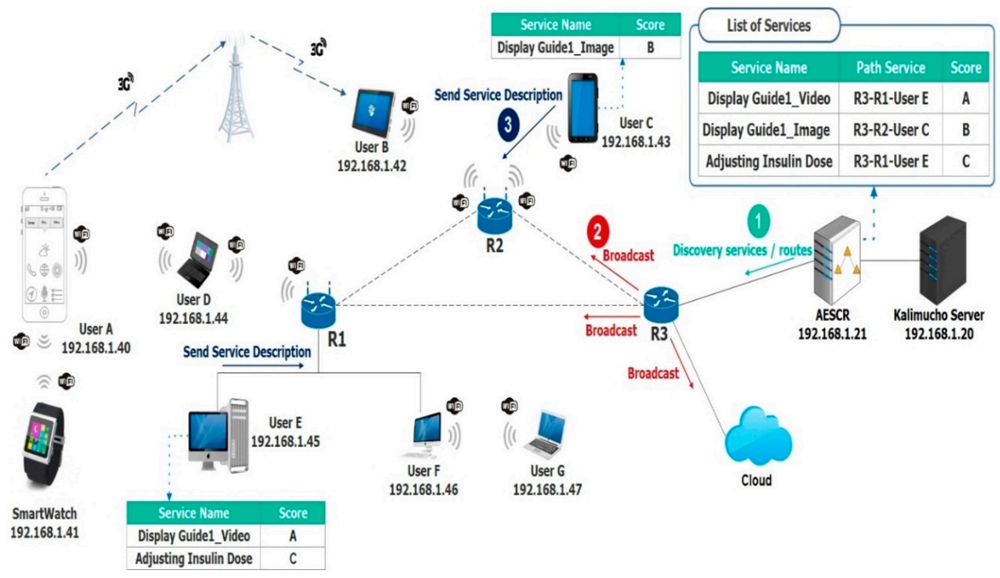
PC	CreateComponent	Service_Location_SendGPS_Coordiantes	application.sendGPSLocation.SendGPSLocation
Smartphone	CreateComponent	Service_Location_ViewGPS	application.viewGPSLocation.ViewLocation
SmartTV	CreateComponent	Service_Location_ViewGPS	application.viewGPSLocation. LocationInformations
SmartWatch	CreateComponent	Service_Location_ViewGPS	application.viewGPSLocation. LocationNotification

PC	CreateComponent	Service_Health_SendGlucose	application.sendGlucose.SendGlucose
PC	CreateComponent	Service_Health_SendWeight	application.sendWeight.SendWeight
Smartphone	CreateComponent	Service_Health_ViewGlucose	applications.ViewGlucose.ViewGlucose
Smartphone	CreateComponent	Service_Health_ViewWeight	applications. ViewWeight. ViewWeight
SmartWatch	CreateComponent	Service_Health_ViewGlucose	applications. HealthNotif. GlucoseNotif
SmartWatch	CreateComponent	Service_Health_ViewWeight	applications. HealthNotif. WeightNotif
SmartTV	CreateComponent	Service_Health_ViewGlucose	applications. HealthNotif. GlucoseInformations
SmartTV	CreateComponent	Service_Health_ViewWeight	applications. HealthNotif. WeightInformations

Figure 19. Available devices in a smart home and the initial script generation.

Once a user location is identified (e.g., smart home, smart car, smart office), the *Action Reasoner* receives a notification that triggers the search for appropriate services according to the identified user location. The script bellow is generated by the reconfiguration generator after scanning the environment with the help of Kalimucho. Our Prediction Reasoner deploys interactive multimodal services in a smart TV using the voice modality and in a smart watch using the gesture modality.

Once an abnormal situation is identified, the ASSACR is responsible for discovering available services around the user location. This is done thanks to the SNMP protocol and the middleware Kalimucho server side offering such a service; the ASSACR uploads different available services' profiles and starts semantic service matching that fulfills the user context and preferences according to Equation (1) (Section 4.2). User A is looking for a quality multimedia health service with low cost with implicit constraints: the screen resolution of the smartphone is limited to = 400 × 600, which is less than the document video. Now, studying both local and remote service results and taking into consideration the user preferences, the degree of the similarity is always high if the set of context attributes suggested by the health service and requested by User A is great, no matter the number of attributes provided by the service, but not asked by the client. Three available health services with different modalities with different multimedia contents across User A's location are as depicted in Figure 20. We used [18] for classifying the relevant interactive services that have a potential benefit. It takes into account the usage domain (the list of devices that the user can access or control in a certain situation or context) to illustrate the multimodality aspect and finds the appropriate device. Table 6 shows the evaluation results, meaning that the service orchestration "*Adjusting Insulin Service (User E) + Display Guide1 Video (User E)*" is selected as the best orchestration service. This selection is based on the high score compared to other paths because the preference desired by User B is satisfied by two neighbor users (User C and User E), but User E is top classified because of the available laptop resources. The ASSACR plugs in the "*Adjusting Insulin Service Client* and *Display Guide1 Video (User E)*" and invokes the service.



Service Name	Score
Display Guide1_Image	B

Service Name	Path Service	Score
Display Guide1_Video	R3-R1-User E	A
Display Guide1_Image	R3-R2-User C	B
Adjusting Insulin Dose	R3-R1-User E	C

Service Name	Score
Display Guide1_Video	A
Adjusting Insulin Dose	C

Figure 20. Semantic service discovery and different service composition paths.

Table 6. Evaluations results [11].

Services Composition Paths	Score
Adjusting Insulin Service (User E) + Display Guide1 Video (User E)	0.83
Adjusting Insulin Service (User E) + Display Guide1 Image (User A)	0.62

In order to support the needs of a dynamically changing context, ASSACR generates reconfiguration files (*Reconfig_C_E*, *Reconfig_E_C*) with two semantic equivalent services paths, but with different QoS in order to easily add or remove services at run-time.

5.2.2. Scenario II: User A in a Car using a Car Tablet

A second scenario when a User A uses a car, he or she can use a car tablet for a larger view. User B cannot read his or her personal emails in a car. ASSACR has SNMP attached as the discovery protocol and GPS as the client service. The latter reinitiates the adaptation process, infers the actions of discovering that the available text to speech services are supported by the user location: *update Text email service with Text to Speech Encoder/Decoder service*. After finding the adequate adaptation services from the cloud and after their selection and integration, the user can follow the email service while using a car. When a user asks for the traffic guide, the ASSACR discovers two map types: a black and white map and a colored map. The ASSACR deploys the second one because of the available resources.

5.2.3. Scenario III: Low Battery Level

In a third scenario, for instance when a current service delivers a high density color map, a notification was sent to the ASSACR that there is not enough available energy to continue displaying such an image. In this case, ASSACR migrates the service to the cloud.

5.3. Performance Comparison and Discussion

To test the feasibility of our ASSACR approach, we have to test the performance of our centralized/distributed semantic event detection algorithm and service semantic prediction algorithm on a laptop running Windows 7 with 6 GB of RAM and a i7-2630QM quadruple core-processor (2 GHz).

We have considered almost 80 users with different constraints (different number of conditions). Each constraint is translated as an event-condition-action rule. We have measured the computation time of the distributed/centralized semantic event detection mechanism. As shown in Figure 21, the centralized event detection gets alarmingly slow when the ontology grows in user profile instances and in the number of constraints. The difference between the centralized event detection and our centralized/distributed semantic-based dynamic event detection is the larger time amount being consuming in the centralized event detection. While checking the event, we have noticed that the more the users there are, the greater the time that is consumed; we improved the execution time by applying our algorithm, which performs each event detection separately; thus, we reduce size of the rules, thereby reducing the time it takes.

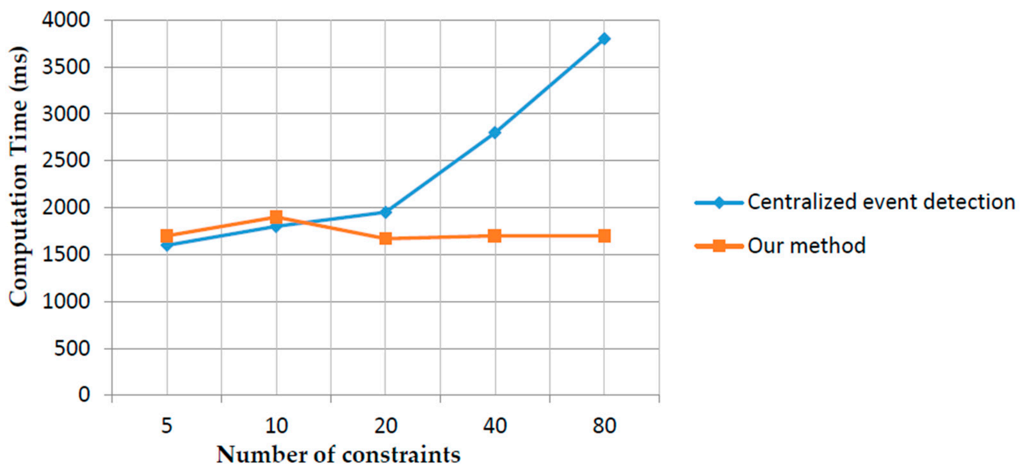


Figure 21. The comparison of the computation time between our distributed event detection and the centralized/distributed event detection in terms of the number of constraints.

To evaluate the matching accuracy, we compare the probability of the high quality service matching precision with/without the prediction strategy. The repository of services increases from 10 to 100. The system is modeled as a Poisson-process, and each mobile device's CPU speed is initialized with a random value in the range of [100, 800] and reduced automatically by a random value in the range of [0, 5]. We gathered results consisting of services in the Kalimucho server and the number of services selected from the Kalimucho server and deployed locally and/or in nearby devices. As we can see in Figure 22, in the majority of the cases, the selection frequency of the predicted services was higher than the probability of their selection without the prediction strategy. For example, with a repository size of 50, the probability of its selection was about 95% using our prediction strategy. Any modifications to those complex user contexts may require less checking time, thus increasing the flexibility of the user applications.

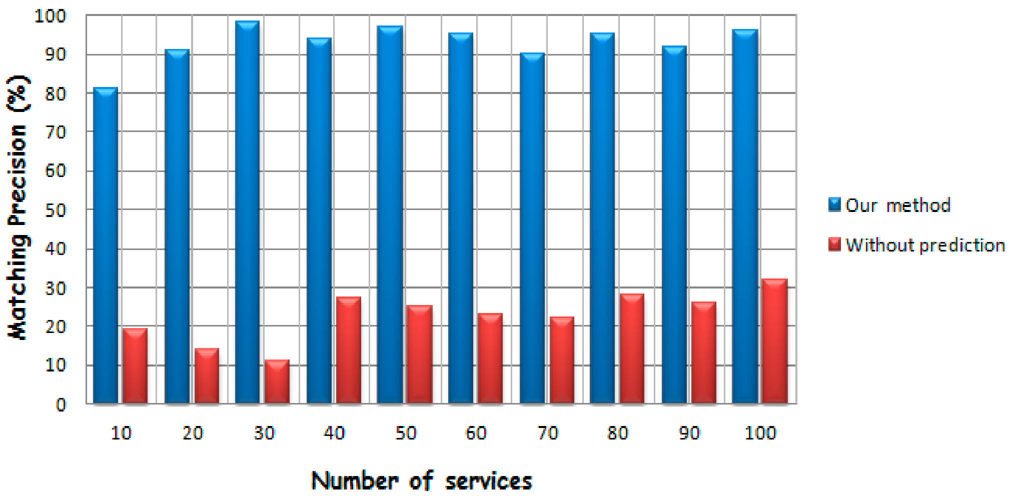


Figure 22. Services matching precision with/without the prediction strategy.

5.4. Advantages and Limitations of Kali-Smart Platform

The proposed platform is able to predict the person's context changes by using the Poisson distribution. It tries to provide the user appropriate services at the right time, in the right place and with the right manner that satisfies all of their requirements. The requirement of the user always varies every day in real life. Thus, the architecture must be dynamic and flexible. These last aspects are the principle advantages of our approach.

- Flexibility: our platform allows a user to describe his or her requirements in the file profile. Therefore, it facilitates modification. Our platform can add, remove or modify easily and quickly certain services in the user environment in a transparent and uniform way according to the user needs and contextual usage. Hence, the user can execute a service corresponding to any context and any device. The introduction of the distributed action mechanism introduces much flexibility and many opportunities for further extensions regarding how context entities should behave.
- Semantic intelligence and autonomic optimization: the proposed platform is a complete solution that minimizes the response time of the situation matching under the criteria's priority (location, time, category), which facilitates inference in the ontology and maximizes redundancy relays and switching mobile services.
- Interactive service experience: our platform uses semantic technologies and the concept of multi-device context data representation to facilitate seamless and interactive media services in a common contextual mobile environments.

The drawbacks of our context middleware, Kali-Smart, are:

- High flexibility: the proposed platform depends on users' constraints and the service repository. If the user requires constraints with any solved services, he or she cannot obtain the content
- More dynamicity: our platform needs to dynamically change the executing environment the same way as other context-aware mobile applications.
- Scalability: our architecture lacks several aspects (timing, scalability, etc.) related to the usage in realistic smart environments.

6. Conclusions

An ideal future where our applications know what we need before we even have to ask is coming. The opportunity is becoming possible to provide more smart and pervasive multimodal services in a smart environment. These services allow users to live safely and inexpensively in their own smart environments. The complexity of pervasive systems in general and ubiquitous-health systems in particular is steadily increasing. In these systems, there is a growing variety of mobile computing devices, which are highly connective and can be used for different tasks, in particular for dynamically-changing environments. Those systems must be able to detect context information over time, which comes from different and heterogeneous entities, deduce new situations from this information and then adapt their behavior automatically according to the deduced situations.

In this paper, we have proposed our context platform, Kali-Smart. The main objective of this platform is the collection of contextual data that are captured directly by sensors. It supports distributed action mechanisms with an incremental context change prediction strategy, as well as automatically adapting data contents to users. It also provides its clients three types of Context Reasoners (Situation Reasoner, Action Reasoner and Prediction Reasoner) programmable to handle more complex client constraints. Moreover, Kali-Smart is supported by a generic and flexible API; it uses an ontology-based model for building a wide amount of applications in various domains. This API hides the complexity and heterogeneity of contextual smart devices in a ubiquitous environment.

In this type of environment, due to the mobility and the limited resources of mobile devices (e.g., battery lifetime), it is difficult to provide the appropriate services at the right time, in the right place and in the right manner. Consequently, we cannot disregard the importance of how the adaptive environments will be able to reason semantically about context information and adapt their behavior according to the dynamic changes of this context. Instead of provisioning context changes in advance, Kali-Smart offers a dynamic searching service that allows its clients to use the current context services available in a requested location.

Finally, our proposal is based on the combination of the Kalimucho middleware [1], mobile computing and the Internet of Things with ontologies and rule-based approaches. This combination permits one to get most of their benefits for realizing a new autonomic adaptation approach for pervasive systems in general and pervasive healthcare systems in particular. Our approach allows: (1) supervising the system, detecting useful contextual information, reasoning about this information and adapting the behavior of the system according to the current context by providing the appropriate service to the user; (2) a distributed/centralized context monitor and semantic event detection in order to manage relevant information that could be important for the context regardless of its source; (3) a centralized semantic context reasoner and an incremental context prediction process making decisions that will be handled by the main host (e.g., computer, server, etc.); this choice is meant to prevent the redundancies of adaptation decisions; (4) the novelty and originality of our approach compared to previous related works by implementing a distributed action mechanism with multimodality aspects that will give the application the flexibility and dynamicity needed to run through the user's smart environment, which, in our knowledge, has not been proposed yet in this field; and (5) maximize redundancy relays and switching mobile services. Compared to current related works, our method improves the matching accuracy greatly by considering the whole meaning of the context conditions while dramatically decreasing the time cost due to the centralized/distributed event detection. In future works, we intend to extend the context model with new concepts and to evaluate our architecture in more complex case study scenarios. Moreover, we will develop our mechanisms for the dynamic components' reconfiguration, including the migration of context middleware from local mobile devices to the cloud.

Author Contributions: Adel Alti proposed the new adaptation idea and wrote the paper; Abderrahim Lakehal analyzed the data and gave some valuable suggestions; Philippe Roose contributed materials/analysis tools. Philippe Roose and Sébastien Laborie performed the research and revised the paper; all four authors designed the experiments and revised the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cassagnes, C.; Roose, P.; Dalmau, M. Kalimucho: software architecture for limited mobile devices. *ACM SIGBED Rev.* **2009**, *6*, 1–6. [CrossRef]

2. Alti, A.; Laborie, S.; Roose, P. Cloud semantic-based dynamic multimodal platform for building *mHealth* context-aware services. In Proceedings of the IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi, UAE, 19–21 October 2015.
3. Alti, A.; Laborie, S.; Roose, P. Dynamic semantic-based adaptation of multimedia documents. *Trans. Emerg. Telecommun. Technol.* **2014**, *25*, 239–258. [CrossRef]
4. Nicolas, F.; Vincent, H.; Stéphane, L.; Gaëtan, R.; et Jean-Yves, T. WComp: Middleware for Ubiquitous Computing and System Focused Adaptation. In *Computer Science and Ambient Intelligence*; ISTE Ltd.: London, UK; John Wiley and Sons: Hoboken, NJ, USA, 2013.
5. Romain, R.; Barone, P.; Ding, Y.; Eliassen, F.; Hallsteinsen, S.; Lorenzo, J.; Mamelli, A.; Scholz, U. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In *Software Engineering for Self-Adaptive Systems*; Springer: Berlin/Heidelberg, Germany, 2009.
6. OSGi Platforms. Available online: http://www.osgi.org/Main/HomePage (accessed on 27 May 2007).
7. Da, K.; Roose, P.; Dalmau, M.; Novado, J. 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) @ MIDDLEWARE'2014, Bordeaux, France, 25–30 December 2014.
8. Mehrotra, A.; Pejovic, V.; Musolesi, M. SenSocial: A Middleware for Integrating Online Social Networks and Mobile Sensing Data Streams. In *Proceedings of the 15th International Middleware Conference (MIDDLEWARE 2014)*; ACM Press: Bordeaux, France, 2014.
9. Taing, T.; Wutzler, M.; Spriner, T.; Cardozo, T.; Alexander, S. Consistent Unanticipated Adaptation for Context-Dependent Applications. In *Proceedings of the 8th International Workshop on Context Oriented Programming (COP'2016)*; ACM Press: Rome, Italy, 2016.
10. Yus, R.; Mena, E.; Ilarri, S.; Illarramendi, A. SHERLOCK: Semantic management of Location-Based Services in wireless environments. *Pervasive Mob. Comput.* **2014**, *15*, 87–99. [CrossRef]
11. Etcheverry, P.; Marquesuzaà, C.; Nodenot, T. A visual programming language for designing interactions embedded in web-based geographic applications. In Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces, Haifa, Israel, 14–17 February 2012.
12. Pappachan, P.; Yus, R.; Joshi, A.; Finin, T. Rafiki: A semantic and collaborative approach to community health-care in underserved areas. In Proceedings of 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Miami, FL, USA, 22–25 October 2014.
13. Chai, J.; Pan, S.; Zhou, M.X. MIND: A Semantics-based Multimodal Interpretation Framework for Conversational Systems. In Proceedings of the International Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems, Copenhagen, Denmark, 28–29 June 2002.
14. Clark, K.; Parsia, B. JESS: OWL 2 Reasoner for Java. Available online: http://clarkparsia.com/Jess (accessed on 17 July 2014).
15. Kim, J.; Chung, K.-Y. Ontology-based healthcare context information model to implement ubiquitous environment. *Multimed. Tools Appl.* **2014**, *71*, 873–888. [CrossRef]
16. Khallouki, H.; Bahaj, M.; Roose, P.; Laborie, S. SMPMA: Semantic multimodal Profile for Multimedia documents adaptation. In Proceedings of the 5th IEEE International Workshop on Codes, Cryptography and Communication Systems (IWCCCS'14), El Jadida, Morocco, 27–28 November 2014.
17. Bahaj, M.; Khallouki, H. UPOMA: User Profile Ontology for Multimedia documents Adaptation. *J. Technol.* **2014**, *6*, 448–457.
18. Hai, Q.P.; Laborie, S.; Roose, P. On-the-fly Multimedia Document Adaptation Architecture. *Procedia Comput. Sci.* **2012**, *10*, 1188–1193. [CrossRef]
19. Anagnostopoulos, C.; Hadjiefthy Miades, S. Enhancing situation aware systems through imprecise reasoning. *IEEE Trans. Mob. Comput.* **2008**, *7*, 1153–1168. [CrossRef]
20. Protégé 2007. Why Protégé. Available online: http://protege.stanford.edu/ (accessed on 27 August 2007).