The OCarePlatform: A Context-Aware System to Support Independent Living

F. De Backere[a,*], P. Bonte[a], S. Verstichel[a], F. Ongenae[a], F. De Turck[a]

[a] *Department of Information Technology (INTEC), Ghent University – iMinds,*
Gaston Crommenlaan 8, bus 201, B-9050 Gent, Belgium

Abstract

Background: Currently, healthcare services, such as institutional care facilities, are burdened with an increasing number of elderly people and individuals with chronic illnesses and a decreasing number of competent caregivers.

Objectives: To relieve the burden on healthcare services, independent living at home could be facilitated, by offering individuals and their (in)formal caregivers support in their daily care and needs. With the rise of pervasive healthcare, new information technology solutions can assist elderly people ("residents") and their caregivers to allow residents to live independently for as long as possible.

Methods: To this end, the OCarePlatform system was designed. This semantic, data-driven and cloud-based back-end system facilitates independent living by offering information and knowledge-based services to the resident and his/her (in)formal caregivers. Data and context information are gathered to realize context-aware and personalized services and to support residents in meeting their daily needs. This body of data, originating from heterogeneous data and information sources, is sent to personalized services, where is fused, thus creating an overview of the resident's current situation.

Results: The architecture of the OCarePlatform is proposed, which is based on a service-oriented approach, together with its different components and their interactions. The implementation details are presented, together with a run-

---

[*] Corresponding author; Tel: +32 9 331 49 38; Fax: +32 9 331 48 99
Email addresses: `femke.debackere@intec.ugent.be` (F. De Backere), `pieter.bonte@intec.ugent.be` (P. Bonte), `stijn.verstichel@intec.ugent.be` (S. Verstichel), `femke.ongenae@intec.ugent.be` (F. Ongenae), `filip.deturck@intec.ugent.be` (F. De Turck)

ning example. A scalability and performance study of the OCarePlatform was performed. The results indicate that the OCarePlatform is able to support a realistic working environment and respond to a trigger in less than 5 seconds. The system is highly dependent on the allocated memory.

Conclusions: The data-driven character of the OCarePlatform facilitates easy plug-in of new functionality, enabling the design of personalized, context-aware services. The OCarePlatform leads to better support for elderly people and individuals with chronic illnesses, who live independently.

Key words: eCare, ontologies, SOA, OSGi

---

1. Introduction

There are many challenges involved in caring for the increasing number of elderly individuals and those with chronic illness in independent living facilities (“residents”) [1]. One of these challenges is a shortage of skilled caregivers to provide help for residents [2] and a wide range of caregivers, some being formal (such as home nurses) and others being informal (such as neighbors).

As new technologies become available, they are also used to facilitate aging in place [3]. Several reviews have already proven and indicated the rising importance of the use of information technology (IT) in healthcare [4, 5]. The past years, there has been an uptake in the use of personal, smart devices [6, 7]. Other IT devices are used in our daily lives; for example, sensors [8, 9]. These devices and technologies generate large amounts of data. Within the pervasive or ubiquitous computing domain, data can be used to extract new knowledge. This knowledge makes it possible to make context-aware or situation-aware decisions [10]. Together with the uptake of pervasive computing, this approach has been used in specific domains, such as healthcare [11].

Next to pervasive healthcare [12], Ambient Assisted Living (AAL) solutions [13] have been introducted. AAL solutions offer IT products, services and systems, with their focus being on the improvement of the Quality of Life (QoL) of the individual. This technology can target specific parts of the QoL experience [14].

In recent years, AAL solutions have seen an increased uptake and have risen to the foreground. Based on reviews in the domain of AAL [13, 15, 16], it becomes clear that current AAL solutions often lack a strong base using a requirement analysis and a mapping of the needs of elderly and people with chronic diseases [15, 13]. Moreover, only a limited number of these solutions takes the whole AAL ecosystem into account [15]. Ontologies facilitate communication, collaboration and integration. Context-aware applications can benefit from incorporating ontologies as these models enable knowledge sharing, reasoning and increase interoperability [17]. Currently, semantic knowledge components in AAL systems [18, 19, 20, 21, 22, 23] are always deployed as a central entity within the framework. Such an approach negatively impacts the scalability and performance of the entire system. Other issues arise when ontologies are used at different endpoints, as discussed in [24, 25], which entails that both ontologies should be kept in sync.

Unlike other solutions described in the literature, our solution was created within the interdisciplinary OCareCloudS (OCCS) project [26]. In this project, all stakeholders of the AAL ecosystem were brought together to get insights into their needs and incorporate these features into the system. The core of this OCCS system is the OCarePlatform, an intelligent, semantic, modular back-end system. The OCarePlatform is designed in a scalable and extensible manner in order to address the shortcomings in existing AAL solutions. This makes it possible to easily add new functionality, based on the needs of the stakeholders. Moreover, the OCarePlatform is be able to process the data in a timely manner and respond within acceptable time ranges.

Previous publications focussed on the use cases where the OCareCloudS system and OCarePlatform can be used, namely in meeting the daily needs of residents and caregivers in home care [26] and handling falls using multiple sensors and context [27].

2. Data flows and manipulation

To achieve a modular, scalable system, a generic, structured format was created to communicate data in the OCareCloudS system and OCarePlatform. This format is called a MetaCareFragment (MCF) and has several properties, such

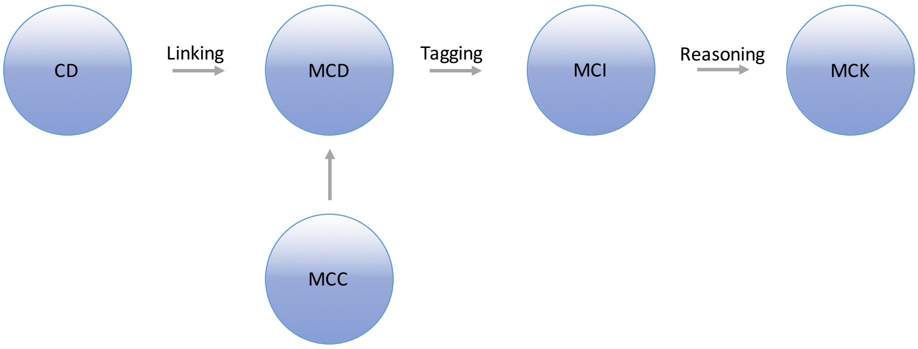

Figure 1: Visualization of the dependencies and definitions of the commonly adopted MetaCareFragment

as timestamps and identification of data source and user. The MCF is formatted as a JavaScript Object Notation [28] (JSON) fragment.

Based on the stage in the life cycle and the location in the OCareCloudS system, this fragment is enriched with new information. As a consequence, the fragment changes its name. This life cycle is shown in Figure 1. Data generated by data sources, such as sensors and devices, are called Care Data (CD). This is just raw data. This body of data is gathered in a central collection point in the resident's home, where it is linked with information of the resident and the caregiver, such as time information and identification of the individual. At this stage, the body of data is called Meta Care Data (MCD). Meta Care Concepts (MCC) are used to tag MCD with concepts corresponding from the ontology used in the OCarePlatform. By tagging MCD, the fragment is transformed into Meta Care Information (MCI). Adding MCC enables the OCarePlatform to interpret the type of data. After the processing by services in the OCarePlatform, MCI is transformed into Meta Care Knowledge (MCK), which is new knowledge inferred out of the combination and/or processing of MCI and MCK.

3. The OCareCloudS system, OCarePlatform components and their interactions

The OCarePlatform is an intelligent, semantic, modular back-end, which adopts a data-driven approach. This means that all data from the resident, devices, sensors and caregivers are gathered and directly sent to the platform. The platform will then forward the data to services, which have indicated their interest in specific types of data.

Data, such as sensor data, gathered from within the resident's home, are sent to the Controllers by the Local Gateway. This Local Gateway is responsible for translating CD into MCD by adding identification data. This Local Gateway forwards all data to the Controllers. The Controllers are able to add MCC to the fragment and transform it into MCI. Moreover, the Controllers directly receive data from the smartphones from the resident or caregivers. The Controllers are also able to contact the caregiver, based on the information received by the OCarePlatform. After the processing step in the Controllers, the data are sent to the OCarePlatform.

The OCarePlatform can be split up in 4 different parts (2), the main features of each being described in the following subsections.

3.1. Preprocessing the data

The collected data enter the OCarePlatform through the Gateway. The Gateway receives all the data packets and forwards them to the Matching Service. The Matching Service will analyze MCI, structured as a JSON object. Based

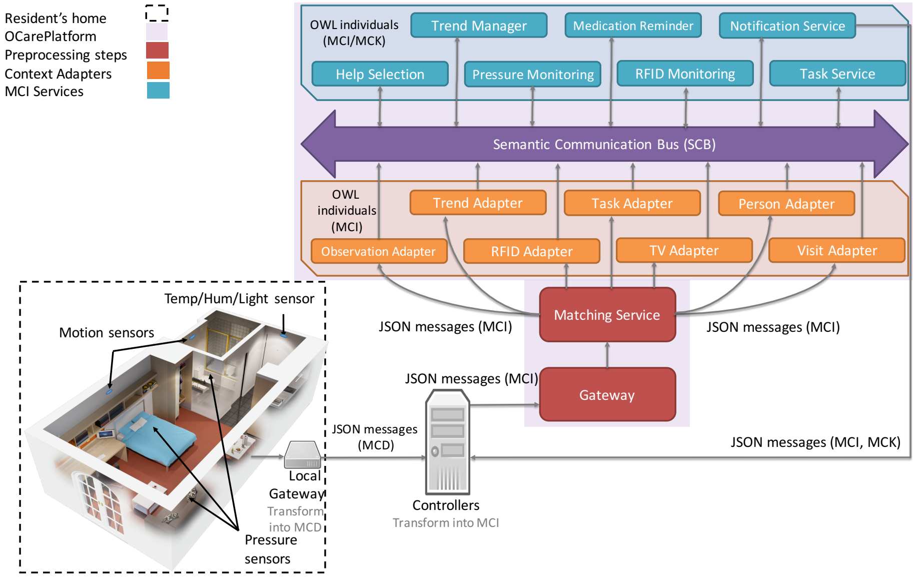

Figure 2: Detailed architecture of the OCarePlatform

on the tagged MCC, the Matching Service will then decide which Adapter needs to receive the data.

3.2. Context Adapters: Transforming the data into individuals

Context Adapters within the OCarePlatform are responsible for translating the JSON fragments into Web Ontology Language (OWL) individuals. The OWL individuals represent the semantically enriched received data, which are interpretable for the ontologies used in the next parts of the OCarePlatform. Each Context Adapter is responsible for the translation of one or more specific types of fragments.

Within the OCarePlatform, the Context Adapters were designed thus:

- Observation Adapter: This transforms all data concerning sensory observations done in the resident's home, for example, this involves the observations made by sensors.
- RFID Adapter: This receives and enriches input whenever caregivers register their presence in the home of the resident by using an RFID card.
- TV Adapter: The home of the resident is also equipped with a smart TV. This TV is used as a sensor. This means that the TV sends logs of performed actions to the OCarePlatform, which can then be used as context information. The OCarePlatform can also control the TV, for example by sending notifications to the device or by controlling the ambilight. Data sent from the TV are processed by the TV Adapter.
- Visit Adapter: Care organizations use planning tools to plan, update or delete visits of the formal caregivers to the patients/clients. This information can easily be inserted in the OCarePlatform using the Visit Adapter. For example, data from Google Calendar can be gathered by the Controllers, where it is also transformed into MCI. This MCI can then be fed to the OCarePlatform, through the Visit Adapter.
- Trend Adapter: When caregivers visit, they can request some trends, using for example the TV (as this screen is big enough to see the graphs). Examples of trends that can be shown are the walking or sleeping behavior

of the resident, based on the input of the sensors. The requests to collect specific trends are handled by the Trend Adapter.

- Task Adapter: This handles various tasks. If an event concerning a task enters the OCarePlatform, such as the acceptance or refusal of a specific task, the Task Adapter will handle it.
- Person Adapter: Residents needing care at home are often helped by several informal and formal caregivers. They sometimes have better relationships with specific caregivers. Therefore, the OCareCloudS system makes it possible to define trust relationships with a specific degree between resident and caregivers. New caregivers for a resident thus need to create a trust relationship. These relationships can be taken into account in case of non emergency calls for example. The Person Adapter is responsible for the creation, deletion and editing of trust relationships.

3.3. Semantic Communication Bus: Intelligently and semantically filtering the data

As the OCarePlatform is a data-driven platform, it has to be able to process a huge amount of data within a limited time period. To this end, the Semantic Communication Bus [29] (SCB, sometimes called the Bus in short), acts as the central component of the OCarePlatform and offers an intelligent filter mechanism.

The SCB is designed based on the publish/subscribe design pattern [30], enabling high performance and modifiability. The SCB uses ontologies to semantically filter the individuals published by the Context Adapters and MCI Services. More information on the specific MCI Services can be found in Section 3.4. Subscribers, in this case MCI Services, pass filter rules to the SCB defined as OWL classes. These OWL classes are added to the internal ontology of the Bus. When publishing new individuals, the SCB will reason and derive which MCI Service have indicated an interest to receive the data and will act upon this request.

As the size of the internal ontology of the SCB and the number of filter rules can have a large impact on the reasoning process and hence, on the performance of the Bus, a cache was added.

3.4. MCI Services: Reasoning on the data

The actual processing of the data in the OCarePlatform is executed in the MCI Services. MCI Services are atomic services with specific functionality. To this end, each MCI Service has its own internal ontology and reasoner. The ontology is kept small to make the service as efficient as possible. MCI Services publish filter rules to the SCB, receive individuals from the SCB and process them. After processing, they will again publish their findings to the Bus, enabling other services to further process it.

Currently, there are seven MCI Services defined in the OCarePlatform. New services can easily be added. This only requires the implementation of the specific service functionality and registering the filter rules on the SCB. From the moment they have registered these rules on the Bus, they will immediately receive relevant information.

- Help Selection MCI Service: This service is responsible for determining which caregiver is best suited to execute a task. To do this, context information, fed to the OCarePlatform, is taken into account. Examples of such information is location, availability, travel time of the caregiver and their trust relationship with the resident.

- Pressure Monitoring MCI Service: The MCI Service analyzes the pressure sensors, which are being used in the home. Individual rules per resident can be defined to detect abnormal situations.

- RFID Monitoring MCI Service: Whenever a caregiver registers him/herself to the registration system in the home of the resident using his/her RFID card, the RFID Monitoring MCI Service receives this information. Information generated by this service can be of interest to other MCI Services, such as the Task MCI Service, which will then generate a task list.

- Task MCI Service: Relevant task data are sent to this MCI Service. It keeps track of all the tasks of the different caregivers and can generate a task list when, for example, a caregiver has registered his/her presence in the resident's home. The generation of a personalized task list, based

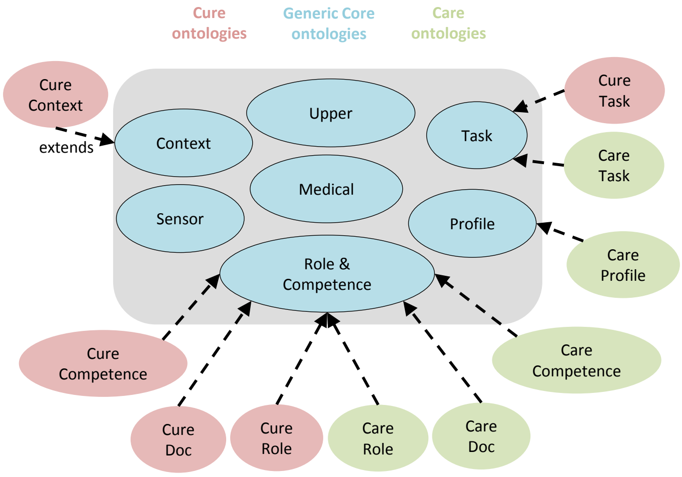

Figure 3: Overview of the Ambient-aware Continuous Care Ontology

on the profile and capabilities of the caregiver, will be triggered when the service receives the individual generated in the RFID Monitoring MCI Service, confirming the presence of a specific caregiver in the resident's home.

- Trend Manager MCI Service: When trends are requested by the caregiver, the Trend Manager MCI Service will gather the requested information.
- Medication Reminder MCI Service: This service can be connected to external data sources, which contain for example the medication scheme of the resident.
- Notification MCI Service: This service is the only service able to communicate with the outside world. This means that every MCI Service wanting to notify or communicate with caregivers or the resident has to push a message on the Bus, which then will be picked up by this MCI Service. It will pass this information through to the Controllers. The Controllers will then process the information.

4. Ontologies

The OCarePlatform makes use of the Ambient-aware Continuous Care Ontology [31]. As shown in Figure 3, it consists of 2 parts. The core ontologies model general knowledge that is applicable across all continuous care settings, e.g., hospitals, independent living facilities and nursing homes. Seven core ontologies are provided, each with its own focus:

- **Upper ontology**: Model generic classes and relations, such as IDs, events and temporal information.
- **Sensor ontology**: It is an extension of the W3C Semantic Sensor Network Ontology (SSN) [32], which models sensors, devices and actuators used within the continuous care domain together with the observations they make and the actions they can take.
- **Context ontology**: This model captures the contextual environment information, such as the layout of the care setting, the purpose of the various rooms and the available furniture. Most importantly, it also models localization information of people and objects.
- **Role & Competence ontology**: This ontology models the various roles and competences that the various (in)formal caregivers can have and how they map on each other and the tasks they can execute.
- **Profile ontology**: This model contains the profile information about the (in)formal caregivers and the care receivers. It captures the biological, sociological, psychological profile as well as the behavioral and medical risk profile.
- **Task ontology**: This ontology models continuous care process workflows. A workflow represents a sequence of related continuous care tasks, which are conducted in a particular order. For this, the OWL-S Process ontology [33] was imported.
- **Medical ontology**: This model contains all the medical knowledge pertaining to a particular resident.

This modular approach of the core ontology allows that MCI Services or applications can easily select the parts of the ontology they require instead of the whole model. It also facilitates the creation of domain-specific extensions of a particular module, as a smaller, focused ontology is easier to interpret and extend with new concepts, relations and definitions. As can be seen in Figure 3, various domain-specific extensions were made, e.g., modelling particular roles and competences within a certain care setting and how they map on each other. The Cure ontologies model domain-specific extensions for hospital settings, while the care ontologies focus on the independent living and residential care settings. More information about the ontologies can be found in Ongenae et al. [31].

5. Implementation

The following sections present the different frameworks and technologies used to implement the OCarePlatform system.

5.1. OSGi

In order to realize a truly modular and modifiable back-end, the OSGi framework [34, 35] was selected. OSGi makes it possible to design a modular, service-oriented platform using the Java Programming language [36, 37]. The OSGi service platform is built of bundles, which can be compared to Java Archive (JAR) files [38]. These bundles can be dynamically installed, started, stopped, updated and uninstalled without requiring a restart of the Java Virtual Machine (JVM). This concept is known as the life cycle management of the software components and is the most important feature of OSGi. Large and complicated applications can be divided into smaller pieces, called services. These services can be re-used and exchanged between applications, leading to collaborations. Bundles can register their functionality as services, providing an interface which can be used by other services [37].

As the OSGi framework facilitates the design of Service-Oriented Architectures, it is ideally suited to implement the OCarePlatform. Moreover, the OCarePlatform will not have to be taken offline to add, update or delete services and different versions of the same service can be deployed.

5.2. Web Ontology Language Application Programming Interface (OWL API)

OWL 2 [39] is the current version of the Web Ontology Language (OWL). OWL is the most populare language currently used to describe an ontology. The OWL 2 language is designed in order to ease the development and sharing of ontologies.

The OWL API [40] supports the creation and manipulation of OWL ontologies. The API is implemented in Java and is open source. The OWL API does not work at the level of triples, but at the level of axioms, which is a higher level of abstraction.

5.3. Reasoners

Ontologies can be processed by a reasoner [41]. The two reasoners used within the OCarePlatform are discussed in the following sub-sub-sections.

5.3.1. Hermit

The SCB and the MCI Services use a Hermit reasoner to derive knowledge from the ontologies and the data within the model. The Hermit reasoner [42] is the first reasoner using hypertableau calculcus, resulting in a more efficient reasoning process than other reasoners.

5.3.2. Pellet and SPARQL Protocol and RDF Query Language (SPARQL)

The Pellet reasoner [43] was used in the Jena ARQ query engine to process the SPARQL queries used in the MCI Services. The SPARQL Protocol and RDF Query Language [44] (SPARQL) is the query language for Resource Description Framework (RDF) content. SPARQL consists of similar operations as the SQL language.

6. Example application

Within the OCareCloudS project [26], an interdisciplinary design methodology was used, namely the Innovation Binder approach [45]. During this iterative approach, personas and scenarios were designed to get insights into the envisioned system [46]. One of these scenarios was used to technically evaluate the

OCarePlatform. A more elaborate version of the scenario and the accompanying personas can be found in [26]. This scenario, together with the more technical details, is discussed in the following paragraphs.

6.1. Scenario

Yousuf, 80 years old, lives with his daughter Fatima and is experiencing increased mobility problems since a recent fall. Fatima depends on formal caregivers when she is at work. Lydia is Yousuf's regular home care nurse, while Karen fills in if needed. Ann is an informal caregiver, living next door and helps out occasionally or in case of an emergency.

The home of Fatima and Yousuf is equipped with the OCareCloudS system, which utilizes the OCarePlatform as an intelligent, cloud-based back-end. Several sensors are installed within their home, such as a bed pressure sensor, movement and PIR sensors. The OCareCloudS registration system, using RFID cards, is also installed in the home. The system is configured in such a way that when the pressure sensor in Yousuf's bed is still activated at 10 o'clock in the morning, an alarm is triggered.

1. On Monday, Fatima works an early shift at work and relies on Ann to help Yousuf out of bed.
2. However, Ann has forgotten that she agreed to help out that morning.
3. At 10 o'clock, the system detects that Yousuf is still in bed.
4. A message is sent to Yousuf to ask whether he needs assistance to get out of bed or if he is willing to wait until Lydia arrives, later that day.
5. Yousuf indicates that he needs help to get out of bed, using his tablet.
6. The system acts upon this information and searches the most appropriate caregiver to assist Yousuf.
7. The system sends Lydia a message to help Yousuf out of bed.
8. Lydia receives the message on her smartphone and accepts the request.
9. Lydia arrives at the home and registers her presence using the registration system and her RFID card. This way the system knows that somebody is taking care of Yousuf's needs.

6.2. Technical realization of the scenario

The technical scenario can be realized using the OCarePlatform. The scenario is split up into 3 large parts. In the first step, it is past 10 o'clock and the OCarePlatform detects that Yousuf is still lying in bed. In step 2, Yousuf indicates he wants assistance to get out of bed. Finally, in step 3, Lydia is notified of this incident and indicates she will offer assistance. Data from sensors are collected by the Local Gateway, which is installed in the home of the resident. The Controllers are operating in the cloud, as well as the OCarePlatform. The following subsections go into detail on how decisions and actions are made/taken in the OCarePlatform.

6.2.1. Step 1: OCarePlatform detects Yousuf still lying in bed

The pressure sensor in Yousuf's bed sends out signals in frequent intervals, indicating whether the sensor is pressed (= 1) or not (= 0). The pressure sensor sends out such a signal every minute. At 10 o'clock in the morning, the pressure sensor will still transmit 1 as a value. The Local Gateway processes this data and transforms the CD into MCD. Next, the information is sent to the Controllers, responsible for translating the MCD into MCI. This in fact means adding the correct concepts from the ontology to the MCF.

Then, MCI is forwarded to the OCarePlatform. All data enter the OCarePlatform through the Gateway. The Gateway adds an internal identification number to the fragment and sends the data to the Matching Service. This Matching Service analyzes the MCI and decides, based on the MCC within the MCI, which adapter is able to translate the JSON fragment into OWL individuals. In this case, the Matching Service sends the data to the Observation Adapter, responsible for processing all observations done in the home of the resident. Within the Observation Adapter, the MCI is transformed into OWL individuals and then published on the SCB.

The SCB knows which MCI Services are interested in this type of data based on the hasContext ObjectPropertyAssertion, and forwards it accordingly to the Pressure MCI Service, based on the filter rule the Pressure MCI Service registered to the SCB when the MCI Service bundle was started. This filter rule is shown below.

hasContext **some**(isObservationOf **some**(hasSensorPart **some** PressureSensor))

The Pressure MCI Service uses a SPARQL query to deduct that Yousuf should already be out of bed. The observation created by the Observation Adapter is filled (x) and the time the resident is expected to be out of bed also (y). For Yousuf, this time is set to 10 o'clock in the morning.

Based on the results of the reasoning process, the Pressure MCI Service publishes new individuals on the SCB with a hasContext of taskEvent. Thus, a task is generated to ask Yousuf if assistance is needed.

Based on the type taskEvent, the SCB knows that the Help Selection MCI Service and the Task MCI Service are interested in this new information and forwards it accordingly. The Help Selection MCI Service updates the status of the task to pending, while the Task MCI Service indicates that the status has been updated in parallel. This information is again published on the SCB using the notificationEvent as hasContext type. The Bus forwards this information to the Notification MCI Service, as the filter rules indicate that the Notification MCI Service is interested in notification events. The Notification MCI Service sends this information, formatted in a JSON format, to the Controllers, which in its turn forwards the information to Yousuf's tablet, asking him whether he needs assistance.

6.2.2. Step 2: Yousuf requesting help

Yousuf receives the message on his tablet, asking him if he is in need of assistance. Yousuf indicates that he can use some assistance to get out of bed. Data from the smart devices are directly sent to the Controllers. Thus, data from these devices are sent as MCD, as the personal data of the user can be added on the device. The Contollers transforms the request of Yousuf into MCI by adding MCC and then forwards the information to the OCarePlatform. The Gateway again sends the MCI to the Matching Service, which now analyzes that this type of information should be sent to the Task Adapter. The Task Adapter translates the JSON format into OWL individuals and pushes it on the SCB using the taskEvent type. This type of individuals is of interest for the Help

Selection MCI Service and the Task MCI Service. The Help Selection MCI Service reasons that Lydia is the most appropriate caregiver to assist Yousuf and updates the status of the task to “assigned”. The Task MCI Service indicates the status of the task is changed. This information is pushed to the Bus using the notificationEvent type. The Bus forwards this information, based on the filter rules, to the Notification MCI Service. This service transforms the message from individuals to a JSON format and sends it to the Controllers, where it is forwarded to Lydia.

6.2.3. Step 3: Lydia accepting task

Lydia receives this notification on her smartphone. As she is nearby, she accepts the task, using her smartphone. The message, informing the OCarePlatform that she accepts the task, is sent as MCD to the Controllers. The Controllers add the necessary ontology concepts as tags and forwards this information to the OCarePlatform and, thus, the Gateway. The Gateway receives the MCI and redirects it to the Matching Service. As this is an acceptance of a task, the Matching Service forwards the information to the Task Adapter. The Task Adapter transforms it into OWL individuals using the type taskEvent and publishes it on the Bus. The Bus, as in the previous step, will forward the task information to the Help Selection MCI Service and the Task MCI Service, which enables these services to use this information, and new reasoning processes are started. In the end, Yousuf is notified by the Notification MCI Service and the Controllers that help is on its way.

7. Experimental evaluation

7.1. Evaluation set-up & approach

To get more insights into the scalability and performance of the OCarePlatform, several tests were conducted. All tests were performed using the same server with 4 x Dual-Core AMD Opteron™ Processor 2212 CPU with 12 GB RAM and running Debian 3.2.65-1+deb7u1 x86_64 GNU-Linux.

Each data fragment entering and leaving a component in the OCarePlatform is timed. This way, the execution time of each component can be calculated. Each test was executed 35 times, the first three and last two runs were omitted to eliminate any influence of the warm-up and cool-down phases. The mean and standard deviation were calculated over the remaining 30 iterations.

7.2. Evaluation results

The running example was evaluated in terms of execution times. The evaluations, following this analysis of the running example, focus on getting a better understanding of the scalability of the platform, the ontology was scaled up to resemble a realistic working environment.

7.2.1. Evaluation of the example application

In a first test, the standard scenario as described in Section 6 was evaluated.

In the first step of the scenario, the OCarePlatform detects that Yousuf is still lying in bed. Table 1 shows the results. The Pressure MCI Service needs the most processing time, on average 233.06 ms or 68.13%. The other components performing reasoning, namely the Help Selection MCI Service and the Task MCI Service, needing respectively 15.70% (53.80 ms) and 8.97% (30.77 ms), complete the top 3. The other components use about 7.19% of the processing time. As can be seen in the table, the Gateway, responsible for merely delegating data fragments to the Matching Service, is negligible. Data are pushed to the SCB 3 times, and the results are mentioned separately.

Table 1: Percentage, mean and standard deviation for the first step in the scenario

Component	**Percentage (%)**	**Mean (ms)**	**Std dev (ms)**
Gateway	0	0	0
Matching Service	0.17	0.57	0.50
Observation Adapter	1.25	4.3	1.1
SCB 1	0.47	0.14	0.5
Pressure MCI Service	68.14	233.6	24.51
SCB 2	0.09	0.3	0.50
Help Selection MCI Service	15.69	53.80	4.81
Task MCI Service	8.97	30.77	10.03
SCB 3	0.07	0.23	0.43
Notification MCI Service	5.48	18.8	2.04
Total	100	342.83	44.36

In the second phase of the scenario, Yousuf confirms he needs help. The results of the evaluation can be found in Table 2. Again, the Help Selection MCI Service

(57.70%) and Task MCI Service (23.36%) consume the most processing time, followed by the Notification MCI Service, which uses 15.33% or 19.53 ms. This is as expected because of the reasoning processes within these services. The SCB and the Matching Service utilize in total 0.94% or 1.20 ms. Again, the Gateway is negligible.

Table 2: Percentage, mean and standard deviation for the second step in the scenario

Component	**Percentage (%)**	**Mean (ms)**	**Std dev (ms)**
Gateway	0	0	0
Matching Service	0.29	0.37	0.48
Task Adapter	2.67	3.4	1.17
SCB 1	0.29	0.37	0.48
Help Selection MCI Service	57.7	73.53	12.70
Task MCI Service	23.36	29.77	9.26
SCB 2	0.36	0.47	0.50
Notification MCI Service	15.33	19.53	4.68
Total	100	127.43	29.28

The last step of the scenario is similar to the second step, as the same components are called. The results are shown in Table 3. Again, the MCI Services consume most of the total execution time in step 3 (96.41% or 130.73 ms).

Table 3: Percentage, mean and standard deviation for the third step in the scenario

Component	**Percentage (%)**	**Mean (ms)**	**Std dev (ms)**
Gateway	0	0	0
Matching Service	0.27	0.37	0.48
Task Adapter	2.51	3.40	1.05
SCB 1	0.44	0.60	0.49
Help Selection MCI Service	49.73	67.43	67.54
Task MCI Service	31.98	43.37	69.40
SCB 2	0.37	0.50	0.50
Notification MCI Service	14.7	19.93	6.36
Total	100	135.60	148.82

To obtain an overview of the performance for this scenario, the execution times for each component were added (Table 4). In Figure 4, a pie-in-pie chart is shown to visualize these results. The smallest pie resembles the four least consuming components in the scenario. The MCI Services and Context Adapters, responsible for the reasoning in the platform and for the transformation of the data into individuals consume the most time. However, the execution times of the Context Adapters are negligible compared to those of the MCI Services.

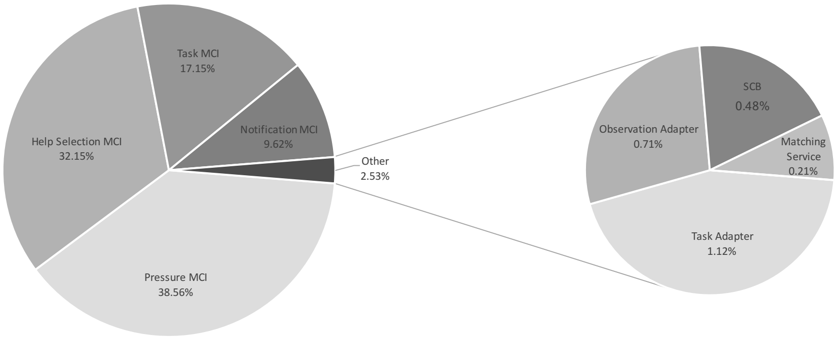

Figure 4: Execution time of the complete scenario

7.2.2. Evaluation of an increasing number of residents supported by the OCare-Platform

In a realistic working environment, formal caregivers handle requests for help between 10 to 12 residents per day. This is based on Gellatly et al [47] where nurse aide ratios in nursing homes are set to 8 to 12 patients and based on the rules used in home care worker schedules [48] as travelling should also be taken into account. As the scenario discussed in Section 6 contains 2 formal caregivers, the number of residents in this evaluation is steadily increased to 23 residents. This way, a prediction can be made on how the number of residents in the ontology influences the behavior of the OCarePlatform. The results of the evaluation are shown in Figure 5. As can be seen in Figure 5, the execution time increases with the growing number of patients in the ontology. The first iteration with 1 patient and 2 formal caregivers takes about 600 ms. The total

Table 4: Total execution times of the technical scenario

Component	**Percentage (%)**	**Mean (ms)**	**Std dev (ms)**
Pressure MCI Service	38.56	233.60	24.51
Help Selection MCI Service	32.15	194.77	85.05
Task MCI Service	17.15	103.90	88.68
Notification MCI Service	9.62	58.27	13.08
Task Adapter	1.12	6.80	2.22
Observation Adapter	4.30	1.1	0.71
SCB	0.48	2.93	3.35
Matching Service	0.21	1.30	1.46
Gateway	0	0	0
Total	100	605.87	219.46

execution time for an ontology with 23 residents and 2 formal caregivers is about 700 ms. A linear trend can be identified, depending on the number of residents in the ontology. The error bars in the graph visualize the standard deviation. As can be seen, some standard deviations are bigger than others. The larger deviations are caused by the garbage collection of the Java Virtual Machine (JVM).

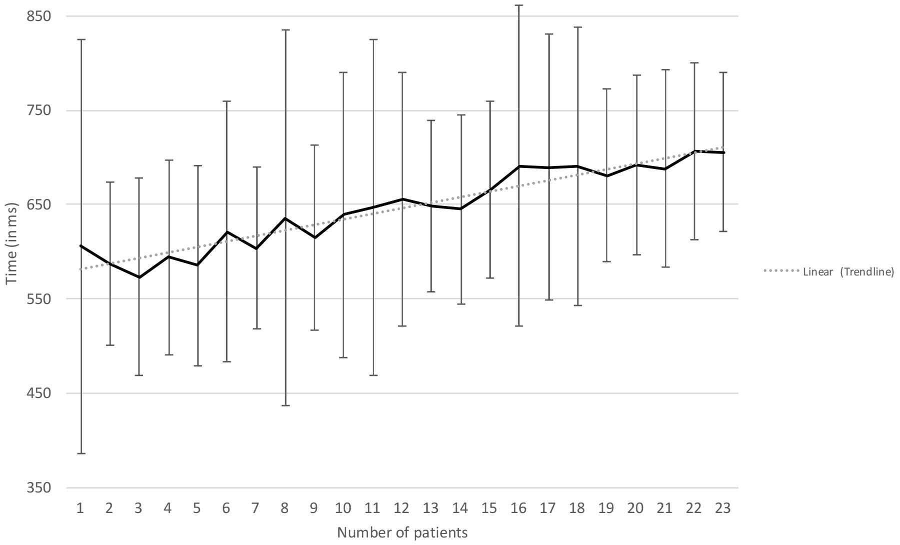

Figure 5: Influence of the number of residents in the ontology

7.2.3. Evaluation of an increasing number of (in)formal caregivers and residents

Within this evaluation, the number of residents, their informal caregivers and formal caregivers is steadily increased. In each step, one formal caregiver is added to the ontology, together with 12 residents, for which this formal caregiver is responsible. Each resident has zero to three informal caregivers in their care network, which are assigned at random based on a normal distribution. Per iteration, one pressure sensor of one patient chosen at random triggers the OCarePlatform. In a realistic work setting, formal caregivers work together in teams of 8 to 10 people. Therefore, the number of formal caregivers in the ontology was increased from 1 to 10, meaning the number of residents rose to 120.

Pressure sensor indicates unexpected situation in the home. This evaluation starts from the scenario as discussed in Section 6. The results of the evaluation are illustrated in Figure 6. For an ontology with 10 formal caregivers and 120 residents, each with their own informal caregivers, the execution time climbs up to 3680 ms. Again, this is a linear trend in terms of the increasing number of caregivers.

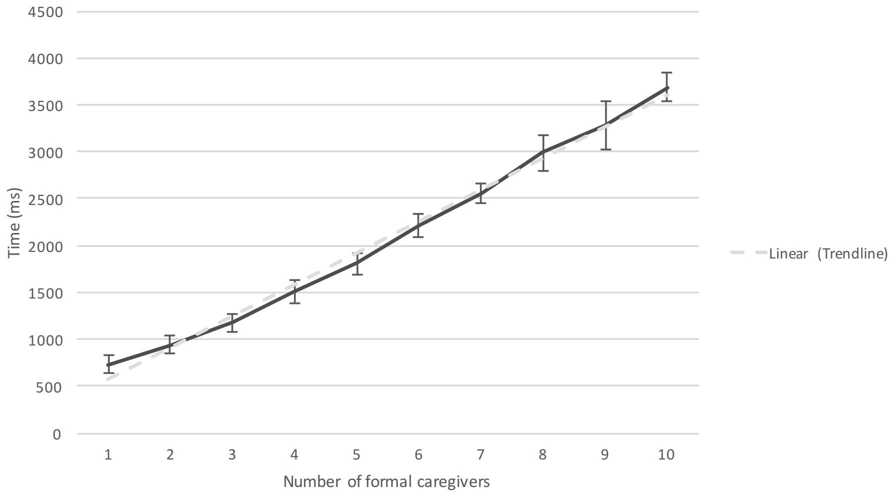

Figure 6: Influence of the number of residents, formal and informal caregivers in the ontology when pressure sensor triggers an unexpected situation

Registration in the home and task assignment. Within this evaluation, a caregiver enters the home of the resident and registers him/herself using an RFID card. Based on the person entering the house, the personal task list is generated and sent to this person. This way, the caregivers knows which tasks, together with the associated priority, can and should be executed and the associated priority. As can be seen in Figure 7, the execution time for an ontology with 10 formal caregivers, which have each been assigned zero to four tasks, based on a normal distribution, and 120 residents, climbs up to 450 ms.

7.2.4. Influence of memory allocation on the execution times

A final evaluation focuses on the influence of the amount of memory, allocated to the server, on the execution times of the OCarePlatform. In order to compare this, the technical scenario was executed on a server with 4 GB RAM and

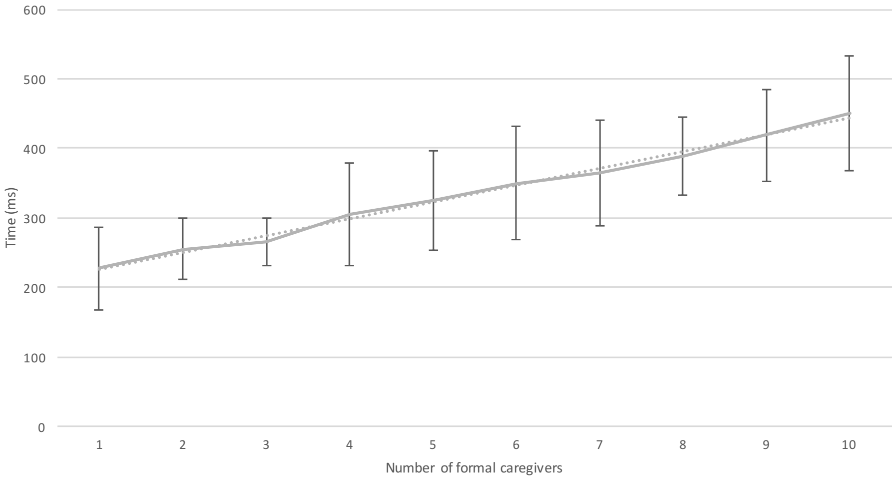

Figure 7: Influence of the number of residents, formal and informal caregivers in the ontology when caregiver registers in the home

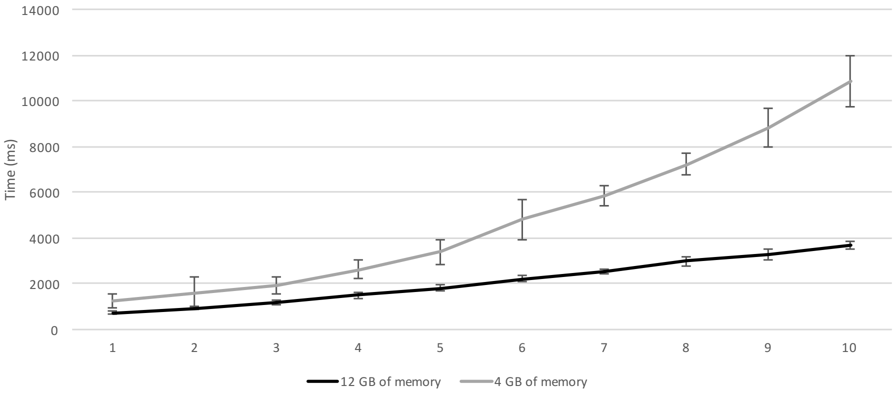

Figure 8: Comparison of the execution times of the technical scenario with an increasing number of (in)formal caregivers and residents using different memory sizes

12 GB RAM. The results are shown in Figure 8. As can be seen from this graph, the execution times increase more rapidly when only 4 GB RAM is used.

8. Discussion

In the coming years, healthcare services will be increasingly burdened with an increasing number of elderly people and people with chronic diseases. Moreover, the number of available caregivers is decreasing steadily. A solution for this problem is home care, where people can receive care in the comfort of their own home, supported by technology. State of the art research indicated that

AAL solutions often lack a thorough requirements analysis and do not meet the needs of all the involved stakeholders, such as residents and caregivers [15, 13]. Ontologies can be used in these solutions to enable context-awareness, reasoning and interoperability. However, as ontologies often are centralized [18, 19, 20, 21, 22, 23], the system may be robbed of its scalability and performance. In this paper, we presented the OCarePlatform, a semantic modular back-end system. The OCarePlatform is designed as a SOA, in which various semantic services are able to process heterogeneous data, originating from both resident and caregiver. These services form dynamic workflows, tailored to the needs and to support all involved stakeholders of the AAL ecosystem in a scalable and efficient manner.

The OCarePlatform is designed in a scalable and extensible manner, making it possible to easily add new functionality. The core of the OCarePlatform uses the Semantic Communication Bus (SCB), which is responsible for redirecting the plethora of data. While the SCB has an overall picture of the system using the core ontologies, the services only use a domain-specific extension of (a subset of) the core ontologies. By designing multiple services responsible for very specific functionality, dynamic workflows are created that are loosely coupled. These advantages all contribute to a more scalable and extensible platform. One shortcoming of this system is that during the realization of the first only limited focus was given to privacy and security.

Unlike other solutions, this system was developed using an interdisciplinary methodology, bringing together all involved stakeholders and focussing on their needs. By using a communication bus with core ontologies, responsible for forwarding the data to specific services, the SCB performs efficient and is no bottleneck for the system, unlike the systems discussed in [18, 19, 20, 21, 22, 23]. Other solutions like [24, 25] propose an approach in which a system is deployed on two endpoints. Both endpoints use an ontology to process the data, which entails that both ontologies should be kept in sync. By using a cloud-based solution, the OCarePlatform does not have to keep all ontologies of the services in sync. Only the core ontologies in the SCB should be stable.These design decisions guarantee a more efficient and scalable use of ontologies, resulting in a timely delivery of notifications to caregivers and residents. By using the OCarePlatform, caregivers are at ease as they know they will be informed whenever something happens. In summary, the OCarePlatform facilitates aging in place and living independently at home for as long as possible.

The most important limitation of the OCarePlatform is ensuring the privacy of the data of the resident and the caregivers. As the OCarePlatform is used to communicate health and care information, it is important that security and privacy are guaranteed. Currently, this is done by using trust circles. When a caregiver is added to the trust circle of the resident, the caregiver is added to the platform and will also receive notifications from the platform. In the future, integration with external data sources, such as Vitalink and the eHealth platform, would make this process easier, as this platform takes trust relationships into account. One drawback of using OWL API is its in-memory representation, limiting the size of the processed ontologies [40]. It can be concluded that allocating enough memory is of crucial importance when deploying the OCarePlatform in order to deliver messages in a timely manner. By deploying the OCarePlatform in the cloud, the memory allocation is not an issue. Moreover, if memory does become problematic, MCI Services could be deployed on different servers in order to provide sufficient memory to maintain a good performance. Finally, the core ontologies within the SCB should be stable. If core ontologies are updated this will influence backward compatibility and already deployed MCI services. By using intermediate concepts to make changes to the ontology, this issue can be avoided.

9. Conclusions

In the present contribution, we describe the design and implementation of a semantic, data-driven platform, referred to as the OCarePlatform. This system facilitates independent living by offering information and knowledge-based services to the resident and his/her (in)formal caregivers. This is realized by gathering data collected from various and heterogeneous sources within the home of the resident. Moreover, context information of the resident and the caregivers is also collected. The OCarePlatform processes this data semantically by making use of ontologies. The Semantic Communication Bus (SCB) filters and forwards data to services, which have indicated an interest in that specific type of such data. Several MCI Services are plugged onto this Bus, providing specific functionality.

The evaluation of the OCarePlatform shows that the OCarePlatform is able to deal with an increasing number of residents and caregivers within a realistic working environment. Further analysis showed that the performance of the platform is highly dependent on the allocated memory. Future work will focus on deploying the OCarePlatform on devices that have limited resources.

Acknowledgment

Part of this research was supported by the iMinds Project OCareCloudS co-funded by the IWT, iMinds and the following partners: Televic Healthcare, TPVision, Telecom IT and Boone NV. We thank OCMW Kortrijk, OCMW Ghent and Familiehulp, for their cooperation on the project.

References

[1] C. Röcker, Ambient assisted living: Chances and challenges of intelligent homecare solutions, in: 9th Japanese-German Frontiers of Science Symposium, 2012, pp. 6–14.

[2] B. d. Ruyter, E. Pelgrim, Ambient assisted-living research in carelab, Interactions 14 (4) (2007) 30–33.

[3] CDC, Healthy places – healthy places terminology (2009).
URL `http://www.cdc.gov/healthyplaces/terminology.htm`

[4] S. J. Gentles, C. Lokker, K. A. McKibbon, Health information technology to facilitate communication involving health care providers, caregivers, and pediatric patients: A scoping review, Journal of Medical Internet Research 12 (2).

[5] B. Lindberg, C. Nilsson, D. Zotterman, S. Söderberg, L. Skär, Using information and communication technology in home care for communication between patients, family members, and healthcare professionals: A systematic review, International Journal of Telemedicine and Applications (2013) 2.

[6] K. Dery, D. Kolb, J. MacCormick, Working with connective flow: How smartphone use is evolving in practice, European Journal of Information Systems 23 (5) (2014) 558–570.

[7] A. Oulasvirta, T. Rattenbury, L. Ma, E. Raita, Habits make smartphone use more pervasive, Personal and Ubiquitous Computing 16 (1) (2012) 105–114.

[8] G. Goggin, Cell Phone Culture: Mobile Technology in Everyday Life, Routledge, 2012.

[9] M. F. Dennedy, J. Fox, T. R. Finneran, Technology evolution, people, and privacy, in: The Privacy Engineer's Manifesto, Springer, 2014, pp. 3–24.

[10] M. Satyanarayanan, Pervasive computing: Vision and challenges, IEEE Personal Communications 8 (4) (2001) 10–17.

[11] C. Doukas, I. Maglogiannis, Bringing iot and cloud computing towards pervasive healthcare, in: 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), IEEE, 2012, pp. 922–926.

[12] U. Varshney, Pervasive healthcare: Applications, challenges and wireless solutions, Communications of the Association for Information Systems 16 (1) (2005) 3.

[13] P. Rashidi, A. Mihailidis, A survey on ambient-assisted living tools for older adults, IEEE Journal of Biomedical and Health Informatics 17 (3) (2013) 579–590.

[14] G. van den Broek, F. Cavallo, C. Wehrmann, AALIANCE Ambient Assisted Living Roadmap, Vol. 6, IOS press, 2010.

[15] D. Calvaresi, D. Cesarini, P. Sernani, M. Marinoni, A. F. Dragoni, A. Sturm, Exploring the ambient assisted living domain: A systematic review, Journal of Ambient Intelligence and Humanized Computing (2016) 1–19.

[16] W. Ludwig, K.-H. Wolf, C. Duwenkamp, N. Gusew, N. Hellrung, M. Marschollek, M. Wagner, R. Haux, Health-enabling technologies for the elderly – an overview of services based on a literature review, Computer Methods and Programs in Biomedicine 106 (2) (2012) 70–78.

[17] H. Chen, T. Finin, A. Joshi, An ontology for context-aware pervasive computing environments, The Knowledge Engineering Review 18 (03) (2003) 197–207.

[18] A. Forkan, I. Khalil, Z. Tari, Cocamaal: A cloud-oriented context-aware middleware in ambient assisted living, Future Generation Computer Systems 35 (2014) 114–127.

[19] M. Amoretti, S. Copelli, F. Wientapper, F. Furfari, S. Lenzi, S. Chessa, Sensor data fusion for activity monitoring in the persona ambient assisted living project, Journal of Ambient Intelligence and Humanized Computing 4 (1) (2013) 67–84.

[20] H. Kuijs, C. Rosencrantz, C. Reich, A context-aware, intelligent and flexible ambient assisted living platform architecture, Cloud Computing.

[21] R. F. Navarro, M. Rodriguez, J. Favela, Intervention tailoring in augmented cognition systems for elders with dementia, IEEE Journal of Biomedical and Health Informatics 18 (1) (2014) 361–367.

[22] V. F. S. Fook, S. C. Tay, M. Jayachandran, J. Biswas, D. Zhang, An ontology-based context model in monitoring and handling agitation behavior for persons with dementia, in: 4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops, IEEE, 2006, pp. 5–pp.

[23] D. Zhang, Z. Yu, C.-Y. Chin, Context-aware infrastructure for personalized healthcare, Studies in Health Technology and Informatics 117 (2005) 154–163.

[24] N. Lasierra, A. Alesanco, J. Garcia, Designing an architecture for monitoring patients at home: Ontologies and web services for clinical and technical

management integration, IEEE Journal of Biomedical and Health Informatics 18 (3) (2014) 896–906.

[25] F. Paganelli, D. Giuli, An ontology-based system for context-aware and configurable services to support home-based continuous care, IEEE Transactions on Information Technology in Biomedicine 15 (2) (2011) 324–333.

[26] F. De Backere, F. Ongenae, F. Vannieuwenborg, J. Van Ooteghem, P. Duysburgh, A. Jansen, J. Hoebeke, K. Wuyts, J. Rossey, F. Van den Abeele, K. Willems, J. Decancq, J. H. Annema, N. Sulmon, D. Van Landuyt, S. Verstichel, P. Crombez, A. Ackaert, D. De Grooff, A. Jacobs, F. De Turck, The ocareclouds project: Toward organizing care through trusted cloud services, Informatics for Health and Social Care 41 (2) (2014) 159–176.

[27] F. De Backere, F. Ongenae, F. Van den Abeele, J. Nelis, P. Bonte, E. Clement, M. Philpott, J. Hoebeke, S. Verstichel, A. Ackaert, et al., Towards a social and context-aware multi-sensor fall detection and risk assessment platform, Computers in biology and medicine 64 (2015) 307–320.

[28] D. Crockford, The application/json media type for javascript object notation (json) (2006).

[29] J. Famaey, S. Latré, J. Strassner, F. De Turck, An ontology-driven semantic bus for autonomic communication elements, in: Modelling Autonomic Communication Environments, Springer, 2010, pp. 37–50.

[30] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Computing Surveys 35 (2) (2003) 114–131.

[31] F. Ongenae, P. Duysburgh, N. Sulmon, M. Verstraete, L. Bleumers, S. De Zutter, S. Verstichel, A. Ackaert, A. Jacobs, F. De Turck, An ontology co-design method for the co-creation of a continuous care ontology, Applied Ontology 9 (1) (2014) 27–64.

[32] W3C Semantic Sensor Network Incubator Group, Semantic sensor network ontology (2011).
URL `http://www.w3.org/2005/Incubator/ssn/ssnx/ssn`

[33] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, OWL-S: Semantic markup for web services, W3C Member Submission (2004).
URL `http://www.w3.org/Submission/OWL-S/`

[34] O. Gruber, B. Hargrave, J. McAffer, P. Rapicault, T. Watson, The eclipse 3.0 platform: Adopting osgi technology, IBM Systems Journal 44 (2) (2005) 289–299.

[35] O. Alliance, About the osgi service platform, technical white paper revision 4.1, 7 june 2007 (2007).

[36] C. Lee, D. Nordstedt, S. Helal, Enabling smart spaces with osgi, IEEE Pervasive Computing 2 (3) (2003) 89–94.

[37] R. Hall, K. Pauls, S. McCulloch, D. Savage, OSGi in Action: Creating Modular Applications in Java, Manning Publications Co., 2011.

[38] M. Campione, K. Walrath, A. Huml, The Java Tutorial: A Short Course on the Basics, Vol. 1, Addison-Wesley Professional, 2001.

[39] D. L. McGuinness, F. Van Harmelen, Owl web ontology language overview, W3C Recommendation 10 (10).

[40] M. Horridge, P. F. Patel-Schneider, Owl 2 web ontologyllanguage manchester syntax, W3C Working Group Note.

[41] X. H. Wang, D. Q. Zhang, T. Gu, H. K. Pung, Ontology based context modeling and reasoning using owl, in: Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, Ieee, 2004, pp. 18–22.

[42] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, Hermit: An owl 2 reasoner, Journal of Automated Reasoning 53 (3) (2014) 245–269.

[43] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical owl-dl reasoner, Web Semantics: Science, Services and Agents on the World Wide Web 5 (2) (2007) 51–53.

[44] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of sparql, in: International Semantic Web Conference, Vol. 4273, Springer, 2006, pp. 30–43.

[45] A. Jacobs, P. Duysburgh, L. Bleumers, F. Ongenae, A. Ackaert, S. Verstichel, The innovation binder approach: A guide towards a social-technical balanced pervasive health system, in: Pervasive Health, Springer, 2014, pp. 69–99.

[46] J. Pruitt, J. Grudin, Personas: Practice and theory, in: Conference on Designing for User Experiences, ACM, 2003, pp. 1–15.

[47] L. Shi, et al., Managing human resources in health care organizations, Jones & Bartlett Publishers, 2010.

[48] C. Akjiratikarl, P. Yenradee, P. R. Drake, PSO-based algorithm for home care worker scheduling in the UK, Computers & Industrial Engineering 53 (4) (2007) 559 – 583.