Intelligent Task Management Platform for Healthcare Workers

Femke Ongenae, Thomas Vanhove, Femke De Backere, and Filip De Turck

Information Technology Department (INTEC), Ghent University - iMinds, Gaston Crommenlaan 8, bus 201, 9050 Ghent, Belgium

ABSTRACT

The medical staff in a hospital could benefit from a specialized task management system, considering their high workload covering different patients. This paper presents an intelligent task management platform that automatically prioritizes and (re-)assigns tasks to the appropriate caregivers based on the current healthcare context captured in a continuous care ontology. Moreover, this platform provides the caregivers with a smartphone allowing them to easily view and process their assigned tasks.

KEYWORDS

eCare, Task Management, Ontology, Rules

CORRESPONDENCE TO:

Femke Ongenae
Department of Information Technology
Internet Based Communication Networks and Services (IBCN)
Ghent University - iMinds
Gaston Crommenlaan 8 (Bus 201), B-9050 Gent, Belgium
T: +32 9 33 14938
F: +32 9 33 14899
E: femke.ongenae@intec.UGent.be

1. INTRODUCTION

1.1. BACKGROUND

Task management systems have been around for some time and with the growing market share of smartphones and tablets, they have been introduced in everyday life. However, the adoption of such systems within healthcare settings is lagging behind. The medical staff in a hospital could benefit from a specialized task management system, considering their high workload covering different patients.

Time is a valuable resource in healthcare settings and caregivers respond to this time pressure by attempting to work as efficiently as possible by establishing a routine and prioritizing their work tasks [1]. However, each nurse has his or her own way of reaching a routine, e.g., by organizing work per patient or task. Nurses also use different criteria to prioritize tasks, e.g., the complexity of the task or the consequences of a task for the patient, the nurse and other tasks [2]. These strategies also cause nurses to lose the flexibility to respond to events and people as any non-scheduled event is perceived as a disruption or something to be prevented [1]. Continuously monitoring their task load and assessing the priorities of these tasks is a tiresome job. When overloaded with work, nurses also attempt to delegate the work better amongst themselves [3]. However, efficient delegation and re-assignment of tasks is often hindered by the social context. Today, caregivers have to decide for themselves who would be most qualified or able to take over the task and have to locate and contact his person themselves to hand over the task. Assigning tasks to other staff members personally can be a hassle, as different problems can occur. For example, it is not always possible to find or reach a person able to perform the task. Keeping the workload balanced is hard as well if there is no general overview of the division of tasks.

To address these issues, this paper presents an intelligent task management platform that automatically prioritizes and (re-)assigns tasks to the appropriate caregivers based on the current healthcare context, e.g., profile and current condition of the patients, current workload of the caregivers and kind of tasks. Moreover, this platform provides the caregivers with a smartphone allowing them to easily monitor their current workload, automatically re-assign tasks, keep track of tasks that have been assigned to them and order these tasks according to their preference, e.g., by task, by patient or by priority. The nurses can also indicate which tasks they are currently handling or have handled. Tasks can be created by the caregivers or by another application or service, e.g., decision support tools or regular tasks that can be created automatically.

Other task management platforms for nurses in hospitals have recently been presented in literature, of which the MEDo approach [4][5] is the most advanced. These platforms concentrate mostly on the creation and workflow management of the tasks. Consequently, most of these research endeavors focus on the front-end design of the application. Our research focusses more on the design of an intelligent context- & profile-aware back-end system that is able to automatically assign and prioritize tasks in a timely manner in order to allow nurses to achieve a balanced workload and work more efficiently. Moreover, the MEDo approach focusses on supporting rounds and not task assignment in general.

Task assignment algorithms have also been proposed to support the planning and assignment of patients in Home Health Care (HHC) [6][7][8]. However, in this case, the required tasks are known in advance and the goal is to achieve an optimal distribution of tasks and patients amongst the available nurses and available time schedule. Our management system focusses on the optimal and timely assignment of tasks that are created on the fly, while still trying to maintain a balanced workload amongst the nurses.

1.2. OBJECTIVES

The aim of this research is the design of a continuous care task management platform with automatic priority and task assignment. The application should offer the advanced features listed below:

- ***Delegation***: A caregiver can assign a staff member to a task, when the task is created. However, it must also be possible to automatically assign the most appropriate caregiver(s) to tasks based on the current context. Similarly, when a task is re-assigned, the caregiver can be assigned by the staff member of automatically by the system.
- ***Dynamic priority assignment***: A priority can be assigned to the task when it is created by the healthcare worker. However, the platform should also be able to assign or change the priority of the call based on the current context.
- ***Performance***: The smartphone application should be able to give an overview of all the tasks assigned to a specific person within a second. This means that the priorities assigned to these tasks should also be calculated with negligible delay. This list should also be updated correctly and timely to alert the user of new tasks that have been assigned.
- ***Scalability***: The platform should be able to support a lot of simultaneous users. An increasing amount of users may not cause a bottleneck for gathering the tasks assigned to a specific person and determining their priority.
- ***User-friendly***: The graphical user interface (GUI) of the task management application on the smartphone must be easy and intuitive to ease the integration of the application in the day-to-day work of the users. An intuitive overview of the different tasks assigned to a healthcare worker and their status is needed which can be easily searched and ordered. A user should also be able to quickly indicate that he or she has started or finished a task. Adding new tasks should require a minimum of input as text-based data entry on a smartphone is tedious. Finally, special alerts should be generated when tasks have been assigned with a high priority.
- ***Reliable***: Three kinds of faults can occur: the back-end server/database can go down, task information is not delivered to the smartphone of the staff member or the information of the smartphone is not delivered to the back-end. The platform has to be able to cope with each of these situations.
- ***Adaptability***: The application has to cope with internal and external changes. The user should be able to change the methods the system uses to assign or change priorities of tasks. The administrators should also be able to update or tests parts of the platform, e.g. the database, without downtime.
- ***Security***: Security is important to this platform as medical data from patients is accessed. First, violations on the inside must be avoided. It is not allowed that any nurse or doctor has access to tasks that are not assigned to him or her or the role that this person belongs to. Additionally, the priority assignment Rules and settings of the platform should only be changed by persons with the correct authorization. Second, the system must also ensure authentication, so that violations from the outside are impossible.

1.3. PAPER ORGANIZATION

The remainder of this paper is organized as follows. Section 2 details the architecture of the developed task management system. The continuous care ontology, which is used to capture the current healthcare context, is discussed in Section 3, while Section 4 dives deeper into the algorithm to assign the most appropriate caregiver to a task based on the current context captured in this ontology. Section 5 elaborates on the implementation details and the evaluation set-up. The results are discussed in Section 6. Finally, Section 7 highlights the conclusions.

2. ARCHITECTURE

Figure 1 visualizes the high-level architecture of the task management system. It consists of a combination of 2 architectural design patterns [9], namely *Publisher-Subscriber* and *Model-View-Controller*. The first is a messaging pattern, which allows that components, called *Publishers*, send a message to a *Message Handler*. Each message is associated with a type. *Subscribers* can then subscribe themselves through the *Message Handler* to specific types of messages. The publisher is thus unaware of which components, if any, receive the message. This leads to a very scalable and dynamic architecture in which the components are loosely coupled. The second pattern separates the representation of information from the users' interaction with it. The *Model* contains the logic and the data of the system, while the *View* is responsible for representing the data to the users. Because of this separation of concerns, different representations of the data can easily be implemented. The *Controller* is responsible for communicating the actions of the users on the *View* to the *Model*. In case the data in the *Model* changes, the *Views* are automatically updated.

In this case, the *Publishers* can either be the smartphones of the caregivers or various healthcare services, e.g., decision support tools or a work schedule application. These *Publishers* publish messages to the *CommunicationHandler*. The messages can for example be new tasks that are created or reassigned, staff members logging in or out or task information that is updated, e.g., the status changes or its priority is altered. *Subscribers*, i.e., the task applications running on the smartphones, which also play the role of the *Views*, can then subscribe to the messages they are interested in. These messages are automatically pushed to these *Subscribers*. However, based on interaction of users with the *View* is also possible that data is requested from the *CommunicationHandler*. This push/pull-model increases the flexibility of the system.

The *Model* consists of various components, which are able to process the received messages of the *Publishers*. The *CommunicationHandler* is responsible for all the communication between the *Publishers* and the *Subscribers*. The *TaskDataModel* component is responsible for managing all the data about the current tasks. Past tasks are stored in the *Database*. This *Database* is also used as back-up to cope with sudden failures of the system. The *TaskCreator* is responsible for creating new tasks based on the information received from the *Publishers*. If not all information is provided, e.g., the person who should handle the task or its priority, this information is requested from the *KnowledgeHandler*. When the task is created, it is passed to the *TaskDataModel*.

The *KnowledgeHandler* can again be broken down into different components. This component manages all the knowledge about the current context in an ontology, e.g., risk factors and medical information of patients and competences, roles, locations and tasks of the caregivers. An ontology [10] is a formal and semantic model of all the concepts within a particular domain and their relationships and properties. This common data-format can then be used to integrate all the healthcare data in a formal manner. The data, which adheres to the concept definitions in the ontology, is collected by the *DataWrapperOntology* from the *Clinical Databases* present in a healthcare institution, e.g., Electronic Patient Records (EPR), Laboratory databases or a database containing personnel information. A semantic *Reasoner* is a piece of generic software, which is able to infer logical consequences, i.e., new knowledge, out of the information captured in an ontology. For example, it can be used to determine which staff members have the appropriate competences to execute a certain task. More complex reasoning on data can be performed by a *Rule Engine*.

The *TaskProcessor* is responsible for processing the tasks it receives from the *TaskCreator*. It contains a queue to be able to process the different requests one by one in the correct order. The *TaskProcessor* is responsible for gathering the different information needed to process the task from the *Rule Engine* and the Ontology *Reasoner*. It then passes this information to the *Finders*,

which implement the algorithms to assign the most appropriate priority (*PriorityFinder*) and caregiver (*SpecialityFinder*) to the task based on the gathered information.

3. CONTINUOUS CARE ONTOLOGY

A modular continuous care ontology was developed, modeling context information and knowledge utilized across the various continuous care settings, i.e., hospitals, residential care and homecare. It consists of a high-level ontology and two low-level ontologies. The first, called the continuous care core ontology, contains knowledge that is applicable across all continuous care domains and is of interest to a plethora of healthcare applications and services. The core ontology was designed in a modular way instead of as one big semantic model, which facilitates (partial) re-use. The following seven core ontologies were developed: the *Upper*, *Sensor*, *Context*, *Profile*, *Role* & *Competence*, *Medical* and *Task* continuous care core ontologies. Two low-level ontologies were developed, modeling knowledge particular to a specific continuous care domain, namely the low-level *Cure* and *Care* ontology, which are respectively tuned towards the knowledge exchanged in hospital and continuous care settings. Each of these models also consists of a number of ontologies, extending specific core ontologies. More information about these ontologies can be found in Ongenae, et al. [11].

The prevalent concepts of the continuous care ontology for the task management platform are visualized in Figure 2. The people present in the healthcare environment are represent by the `Person` concept. Each person is associated with a telephone number. For the staff members, this is the phone number of their smartphone, which runs the mobile task application. People can also be associated with the devices they own and are logged into through the `loggedIntoDevice` and `ownsDevice` relations. This is used to know to which devices the notifications about the assigned tasks should be sent. The `Location` of a person is indicated, which can either be a `Coordinate` or a `Zone`. Each person is unique identified by his or her `ID`. The status of the person can also be expressed, e.g., `Busy`, `Free`, `Present` or not. To express the capabilities of the people, each `Person` is associated with one or more `Roles`. Each `Role` is defined by its `Competences` through classification axioms, e.g., the `Doctor` is defined as a role which has all the `MedicalCompetences`. Each person is associated with competences and roles through five relationships. The `hasFunction` relation indicates the primary role of a person, while `hasRole` models all the roles this person can have and the `hasCurrentRole` models his or her current role. The `hasDiplomaCompetence` and `hasExperienceCompetence` relationship model all the competences this person has acquired, either through their diploma and following courses or by experience. Some example roles and competences are shown in Figure 2. Finally, the medical information about patient is expressed using the `hasDiagnosis` relationship. Based on the medical information, the `MedicalRiskProfile` of a patient is determined.

To represent the tasks and continuous care workflows processes, the *OWL-S Process* [12] ontology was imported, of which the classes are preceded by the `owls` namespace prefix in Figure 2. The `Process` concept models a process, which can return information and produce a change in environment based on the context and the information it is given. This is described by `hasInput`, `hasOutput`, `hasPrecondition` and `hasEffect` relations. A process can be composed of several other processes. How these processes are combined is expressed by the `ControlConstruct` concept. The `Task` concept, introduced as subclass of `Process`, represents the various continuous care tasks. It is further divided into planned and unplanned tasks. Each task has also an associated `Status`, e.g., `Assigned` or `Finished`, and `Priority`. Each task is defined by the `Competences` which are needed to execute this task. Also the location at which this task is preferably executed can be indicated. Finally, the

time is indicated at which the task was created and when it was executed using concepts of the *SWRLTemporalOntology* [13]. Some example tasks are visualized in Figure 2.

4. TASK ASSIGNMENT ALGORITHM

The task management system uses the information captured in the continuous care ontology to assign the most appropriate caregiver to a task. The task assignment algorithm consists of four steps, namely assessing the priority of the task, determining the competences needed to execute the task, filtering the qualified possible caregivers and choosing one of these candidates to assign the task to.

When a task is created by a caregiver on the mobile application or by another service, the following information is specified: the name of the task, its priority and category and the associated patient. The *Rule Engine* contains rules, which are able to determine the competences needed to execute tasks of a particular category. For example, the following rule specifies that a task of the `Medication` category requires a caregiver with the `AdministerMedication` and `CheckMedicationSheet` competences:

```
rule ``Medication"
     when
          $t : Task (category == ``Medication")
     then
          ($t).setCompetency(``CheckMedicationSheet")
          ($t).setCompetency(``AdministerMedication");
end
```

A `Task` of the correct category is created in the ontology with the indicated name. The needed competence is specified using the `needsCompetence` relationship. The associated patient is modeled using the `executedOn` relationship, while the priority is associated using the `hasPriority` relation. The following two rules are specified in the ontology, which allow the *Reasoner* to determine the caregivers, who are able to execute the task because they have the competences required for the task and they are currently present in the healthcare setting:

```
Person(?p), Competence(?c), Task(?t),
hasDiplomaCompetence(?p, ?c), needsCompetence(?t, ?c),
hasStatus(?p, Present)
-> temporarilyAssignedTo(?t, ?p)

Person(?p), Competence(?c), Task(?t),
hasExperienceCompetence(?p, ?c),
needsCompetence(?t, ?c), hasStatus(?p, Present)
-> temporarilyAssignedTo(?t, ?p)
```

The caregivers, who fulfill these criteria, are temporarily assigned to the task.

Next, the priority of the task is assessed. The ontology contains rules, which specify whether the patient has a `MedicalRiskProfile` based on the medical information captured about this patient in the ontology. For example, when the patient has been transferred from the ICU in the last 72 hours or when he or she has recently had a heart attack or an epileptic seizure, he or she is considered at risk. When the patient has a medical risk, his or her priority is increased

one category. This algorithm is implemented in the *PriorityFinder*, which interacts with the information in the continuous care ontology.
Out of all these temporarily assigned caregivers, the most appropriate one is chosen using a weighted algorithm implemented in the *SpecialityFinder*. This algorithm takes into account the context information captured in the ontology. Which context information is taken into account and which weight is assigned to this information was determined by an expert panel of nurses, doctors and R&D engineers with years of experience in designing healthcare applications for communication amongst medical staff. Using the algorithm visualized in Figure 3, each temporarily assigned caregiver is given a weight. The algorithm takes three factors into account, namely the distance between the patient and the caregiver, the relationship between the patient and the caregiver and the workload distribution. The priority of a task determines how much these factors are taken into account. The relationship between caregivers and patients is taken more into account for lower priority tasks. A caregiver, who is more familiar with the medical situation of a patient will be able to perform the task more quickly and easily. Moreover, it gives the patient a feeling of security, trust and continuity when the same caregiver performs most tasks. The weight associated with this factor is first calculated. The distance between the caregiver and the patient becomes more crucial for higher priority tasks. The further the caregiver is removed from the patient, the longer it will take before he or she is able to handle the urgent task. Depending on the priority, the distance is subtracted three or less times from the weight, which was already calculated for this caregiver. Finally, it is important to evenly distribute the workload across the various caregivers. Based on the priority of the task, the number of tasks already assigned to the caregiver are subtracted two or one time(s) from the already calculated weight. For high priority calls, the workload distribution is taken less into account. This factor is less important than the other two factors, as caregivers, who are responsible for more patients, will naturally have more tasks assigned to them. It is mostly used to choose between two or more caregivers, who received an equal weight based on the previous two factors. Finally, the caregiver who received the highest weight is chosen.
The chosen caregiver is assigned to the task in the ontology using the `isAssignedTo` relationship. The `temporarilyAssignedTo` relationships are not removed from the ontology so that they can be reused when the task needs to be re-assigned, e.g., because the staff member logs out or because he or she is too busy or unable to handle the task. The caregiver can indicate that the task should be re-assigned on the smartphone.

5. IMPLEMENTATION DETAILS AND EVALUATION SET-UP

The continuous care ontology was implemented in the Web Ontology Language (OWL) [14] using the Protégé [15] ontology editor. The rules in the ontology were expressed using the Semantic Web Rule Language (SWRL) [16]. Drools [17] was used as Rule Engine. From a thorough literature study [18], we found that only RACER [19], Pellet [20] and Hermit [21] support reasoning on SWRL rules. As our ontology is quite complex and contains a lot of constraints, complete OWL-DL expressiveness is required. However, RACER only supports the SHIQ Description Logic. It is unable to perform reasoning on datatype properties, which is used quite extensively in the task assignment reasoning. Both Pellet and Hermit were employed as ontology reasoners to evaluate which has the best performance. PubNub [22] handles the communication between the smartphone and the back-end server. The mobile application was implemented in Android 2.3.
It is important to evaluate the performance, i.e., execution time and memory usage, of the developed task management system. Most healthcare environments have a limited amount of resources and delegating the processing to the cloud is often difficult because of privacy issues. Moreover, it is important that tasks are swiftly delegated, so that urgent tasks can quickly be assessed and handled. To evaluate the performance, each test consists of the creation of a task

on the task application and using the task management system to assign the most appropriate priority and caregiver. To evaluate the execution time and memory usage, the parameters of the system, e.g., the amount of caregivers in the ontology or the amount of rules, is gradually increased.
To achieve reliable results, each test was repeated 35 times, of which the first three and the last two were omitted during processing. Finally, the averages across the 30 remaining runs are calculated and visualized in the form of graphs. The tests were performed on an Acer Aspire 5920 with the following specifications: 4 gigabyte (GB) 800 megahertz (MHz) Double Data Rate Synchronous Dynamic Random-Access Memory (DDR2 SDRAM) and an Intel Core 2 Duo T5550 1.83 gigahertz (GHz) Central Processing Unit (CPU).

6. RESULTS & DISCUSSION

Figure 4 shows the execution time of the task management system when the amount of available healthcare workers in the ontology is gradually increased. Each healthcare worker also has the competence that is needed to execute the task that is being assigned during the test. Consequently, each present caregiver can possibly be assigned to the task. As mentioned previously, the task management system can be implemented using different semantic reasoners. The graph shows the execution time of the entire system when Pellet or Hermit are used as semantic reasoners. It can be noted that the implementation with Hermit performs significantly worse than with Pellet when more than 15 healthcare workers are present in the ontology. The performance of the implementation with Pellet stays belong 20 seconds, which is acceptable. The Pellet implementation has a linear trend, while the Hermit implementation has an exponential one. When 50 caregivers are present, the execution time of the Hermit implementation is around 2 minutes, which is unacceptable.
To analyze this result in more depth, Figure 5 shows the average distribution of the execution time across the three main components of the task management system, namely the communication using PubNub, the rule reasoning using Drools and the semantic reasoning using Pellet or Hermit. As PubNub is a cloud-based solution, it was difficult to accurately measure the execution time. However, PubNub publishes data on its website about the current response time and cloud uptime. At the time of the research the response time was on average 177 milliseconds (ms). The test was run with 15 caregivers and 23 patients, which is a realistic dataset based on data gathered about departments at Ghent University Hospital. 1000 Drools rules were present. As expected, the semantic reasoning consumes almost all the execution time.

To identify the bottlenecks in the semantic reasoning, Table I shows the average execution time of the different methods that require semantic reasoning. The same ontology is used as in the previous test. The `getPossibleCandidates` method retrieves from the ontology the caregivers with the appropriate competences to execute the task that is currently being assigned. The `getLocation` method retrieves the location of a particular person. This information is used to calculate the distance between a caregiver and a patient. The `isResponsibleCaregiver` method retrieves whether a caregiver is responsible for a particular patient. The `workloadPerson` method retrieves the number of tasks currently assigned to a person. The latter three methods are used by the weighted algorithm, which assigns the most appropriate caregiver to the task.

IT CAN BE NOTED THAT FOR THE PELLET IMPLEMENTATION THE `GETPOSSIBLECANDIDATES` REQUIRES THE MOST EXECUTION TIME. THIS RESULT IS HOWEVER MISLEADING. THIS IS THE FIRST SEMANTIC REASONING TASK PERFORMED DURING A TEST. AT THIS POINT, PELLET WILL CHECK THE CONSISTENCY OF THE ONTOLOGY AND CLASSIFY IT FOR THE FIRST TIME. CONSEQUENTLY, THIS METHOD REQUIRES A LOT OF TIME. AS NO INFORMATION IS ADDED TO THE ONTOLOGY BEFORE THE NEXT REASONING REQUESTS, THESE QUERIES

ARE PERFORMED ON THE ALREADY CLASSIFIED ONTOLOGY. THE FOLLOWING REASONING TASKS ARE THUS PERFORMED VERY SWIFTLY. THIS IS DEMONSTRATED BY THE RESULTS IN

Table II where the order of the reasoning methods in the implementation was changed. It can be noted that the `getLocation` method, which is now the first reasoning task, requires the most execution time.

FOR THE HERMIT IMPLEMENTATION, THE `GETLOCATION` METHOD REQUIRES THE MOST TIME. AS THIS IS NOT THE FIRST REASONING METHOD, IT IS STUDIED FURTHER TO ANALYZE ITS BOTTLENECK.

Table III shows the execution time of the different steps of the `getLocation` method. The `getObjectPropertyValues` retrieves the location of the person from the ontology. The `isIndividualOfClass` method checks whether this location is a `Zone` or a `Coordinate` and is responsible for mapping the `Zone` to its `Coordinate(s)`. The `getDataPropertyValues` method retrieves actual X and Y coordinates from the ontology, while the `getLiteral` method retrieves the double value from these coordinates. It can be noted that the `getDataPropertyValues` requires the most amount of time.

To resolve this issue, the implementation of the `getLocation` method was optimized, so that the datatype properties, namely `hasXCoordinate` and `hasYCoordinate`, do not need to be retrieved. To achieve this, the name of the `Coordinate` was formatted as follows: *<X-coordinate>:<Y-coordinate>*. For example, a coordinate with *X=50* and *Y=50* receives the name *"50:50"*. Now only the two first steps of the `getLocation` method, namely `getObjectPropertyValues` and `isIndividualofClass`, need to be executed. The name of the `Coordinate`, which is returned as a result of these two steps, can then be analyzed to retrieve the X and Y coordinate. Figure 6 shows the execution time of the task management system as a function of the amount of healthcare workers for the three possible implementations of the semantic reasoning, i.e., with Pellet, Hermit and with the optimized implementation of `getLocation` for Hermit. It can be noted that the latter has the best performance. The execution time stays below 5 seconds when at most 60 healthcare workers are represented in the ontology.

As mentioned previously, the properties and weights which the task assignment algorithm should take into account were determined by an expert panel. This is a limitation of the study as these choices were not grounded in a thorough empirical study of current clinical practices and procedures used by the staff to assign and prioritize tasks. However, collecting data out of which these properties and their weights could be derived, requires a very extensive study. This study would for example require that every task and its context are accurately logged and that nurses note down their assessment of the priority of the task. As there is no existing task management system, this would require a significant logging effort from the already time constrained caregivers. As the goal of the study was to first validate whether an intelligent task management system could be designed that uses extensive context and profile knowledge in a sophisticated assignment algorithm and still enables to assign and prioritize tasks in a timely manner, this study was not conducted. Basing the performed evaluation on the properties and weights determined by the expert panel gives us a first idea of the impact and feasibility of the task management system.

However, we do acknowledge that requiring these weights as input for the system could prove cumbersome to accurately determine or adjust by the various departments and hospitals where the system would be deployed. Therefore research is on-going on extending the designed system with self-learning techniques [23], which gradually and automatically adapt the weights to an optimal configuration based on data it collects about its performance, i.e., whether the nurses choose to follow the task assignments proposed by the system or not.

A second limitation of our study is the fact that an extensive clinical study in a real hospital setting of the designed task management system and its algorithms was not performed yet.

However, the designed system was evaluated with the expert panel in a high-fidelity mock-up of a near-future hospital, which contains two patient rooms and a hallway. Various use case scenarios were played, which differed in types of tasks, profiles of available nurses and the context of the tasks. The assignments made by the designed task management system were discussed and evaluated with the expert panel. The properties and the weights of the algorithms were adjusted to incorporate their feedback. Small updates to the GUI were also made.

7. Conclusion

In this paper an intelligent task management platform was presented that automatically prioritizes and (re-)assigns tasks to the appropriate caregivers based on the current healthcare context captured in a continuous care ontology. Moreover, this platform provides the caregivers with a smartphone allowing them to easily monitor their current workload, automatically re-assign tasks, keep track of tasks that have been assigned to them and order these tasks according to their preference. Finally, the performance of the system was studied in depth. A task can be assigned in less than 5 seconds when at most 60 healthcare workers are managed by the system. Future work will mainly focus on more intelligent algorithms to assign priorities to the tasks.

Acknowledgment

Femke Ongenae would like to thank the IWT, Institute for the promotion of Innovation through Science and Technology in Flanders, for supporting this work through her PhD grant. This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

Conflict of Interest

The authors declare that they have no conflict of interest.

REFERENCES

[1] Bowers BJ, Lauring C, Jacobson N. How nurses manage time and work in long-term care. J Adv Nurs. 2001 Feb; 33(4):484-491.

[2] Hendry C, Walker A. Priority setting in clinical nursing practice: literature review. J Adv Nurs. 2004 Aug; 47(4):427-436.

[3] Waterworth S. Time management strategies in nursing practice. J Adv Nurs. 2003 Sep; 43(5):432-440.

[4] Pryss R, Langer D, Reichert M, Hallerbach A. Mobile task management for medical ward rounds - the MEDo approach. In: La Rosa M, Soffer P, editors. Proceedings of the Business Process Management Workshops (BPM); 2012 Sept 3; Tallinn, Estonia. Berlin/Heidelberg: Springer; 2012. p. 43-54.

[5] Pryss R, Mundbrod N, Langer D, Reichert M. Supporting medical ward rounds through mobile task and process management. ISeB. 2015 Feb; 13(1):107-146.

[6] Bachouch RB, Guinet A, Hajri-Gabouj S. An optimization model for task assignment in home health care. Proceedings of the IEEE Workshop on Health Care Management (WHCM); 2010 Feb 18-20; Venice, Italy. IEEE; 2010. p. 1-6.

[7] Trabelsi S, Larbi R, Alouane AH. Linear Integer Programming for the Home Health Care Problem. In: Daniel F, Barkaoui K, Dustdar S, editors. Proceedings of the Business Process Management Workshops (BPM); 2011 Aug 29; Clermont-Ferrand, France. Berlin/Heidelberg: Springer; 2011. p. 143-151.

[8] Mutingi M, Mbohwa C. Task assignment in home healthcare: a fuzzy group genetic algorithm approach. Proceedings of the 25th Annual Conference of the Southern African Institute of Industrial Engineering (SAIIE); 2013 July 9-11; Stellenbosch, South Africa.

[9] Bass L, Clements P, Kazman R. Software Architecture in Practice. 2nd ed. Indianapolis: Addison-Wesley Professional; 2003.

[10] Gruber TR. A Translation Approach to Portable Ontology Specifications. KNOWL ACQUIS. 1993 Jun; 5(2):199-220.

[11] Ongenae F, Duysburgh P, Sulmon N, Verstraete M, Bleumers L, De Zutter S, et al. An ontology co-design method for the co-creation of a continuous care ontology. Appl Ontol. 2014 May; 9(1):27-64.

[12] Martin D, Burstein M, Hobbs J, Lassila O, McDermott D, McIlraith S, et al. OWL-S: Semantic Markup for Web Services [Internet]. World Wide Web Consortium; 2004 Nov [cited 2015 Jun 5]. W3C Member Submission. Available from: http://www.w3.org/Submission/OWL-S/.

[13] O'Connor MJ, Das AK. A lightweight model for representing and reasoning with temporal information in biomedical ontologies. In: Fred ALN, Filipe J, Gamboa H, editors. Proceedings of the International Conference on Health Informatics (HEALTHINF); 2010 Jan 20-23; Valencia, Spain. INSTICC Press; 2010. p. 90–97.

[14] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition) [Internet]. World Wide Web Consortium (W3C); 2012 Dec [cited 2015 Jun 5]. W3C Recommendation. Available from: http://www.w3.org/TR/owl2-overview/.

[15] Knublauch TH, Fergerson RW, Noy NF, Musen MA. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In: McIlraith SA, Plexousakis D, van Harmelen F, editors. Proceedings of the 3rd International Semantic Web Conference (ISWC); 2004 Nov 7-11; Hiroshima, Japan. Berlin/Heidelberg: Springer; 2004. p. 229-243.

[16] Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M. SWRL: A Semantic Web Rule Language Combining OWL and RuleML [Internet]. World Wide Web Consortium (W3C); 2004 May [cited 2015 Jun 5]. W3C Member Submission.

Available from: http://www.w3.org/Submission/SWRL/.
[17] Bali M. Drools JBoss Rules 5.0 Developer's Guide. 1st ed. Birmingham: Packt Publishing; 2009.
[18] Abburu S. A Survey on Ontology Reasoners and Comparison. IJCA. 2012 Nov; 57(17): 33-39.
[19] Haarslev V, Hidde K, Möller R, Wessel M. The RacerPro Knowledge Representation and Reasoning System. SWJ. 2012 Aug; 3(3):267-277. Available at: http://franz.com/agraph/racer/
[20] Sirin E, Parsia B, Grau BC, Kalyanpur A, Katz Y. Pellet: A practical OWL-DL Reasoner. J Web Semant. 2007 Jun; 5(2):51–53. Available at: http://pellet.owldl.com/.
[21] Glimm B, Horrocks I, Motis B, Stoilos G and Wang Z. HermiT: an OWL 2 reasoner. JAR. 2014 May; 53(3):245-269.
[22] PubNub, Inc. [Internet]. California, San Francisco; 2015 [Updated 2015 Jun 5; cited 2015 Jun 5]. Available from: http://www.pubnub.com/
[23] Ongenae F, Claeys M, Dupont T, Kerckhove W, Verhoeve P, Dhaene T, De Turck F. A probabilistic ontology-based platform for self-learning context-aware healthcare applications. ESWA. 2013 Dec; 40(18): 7629-7646.

FIGURES

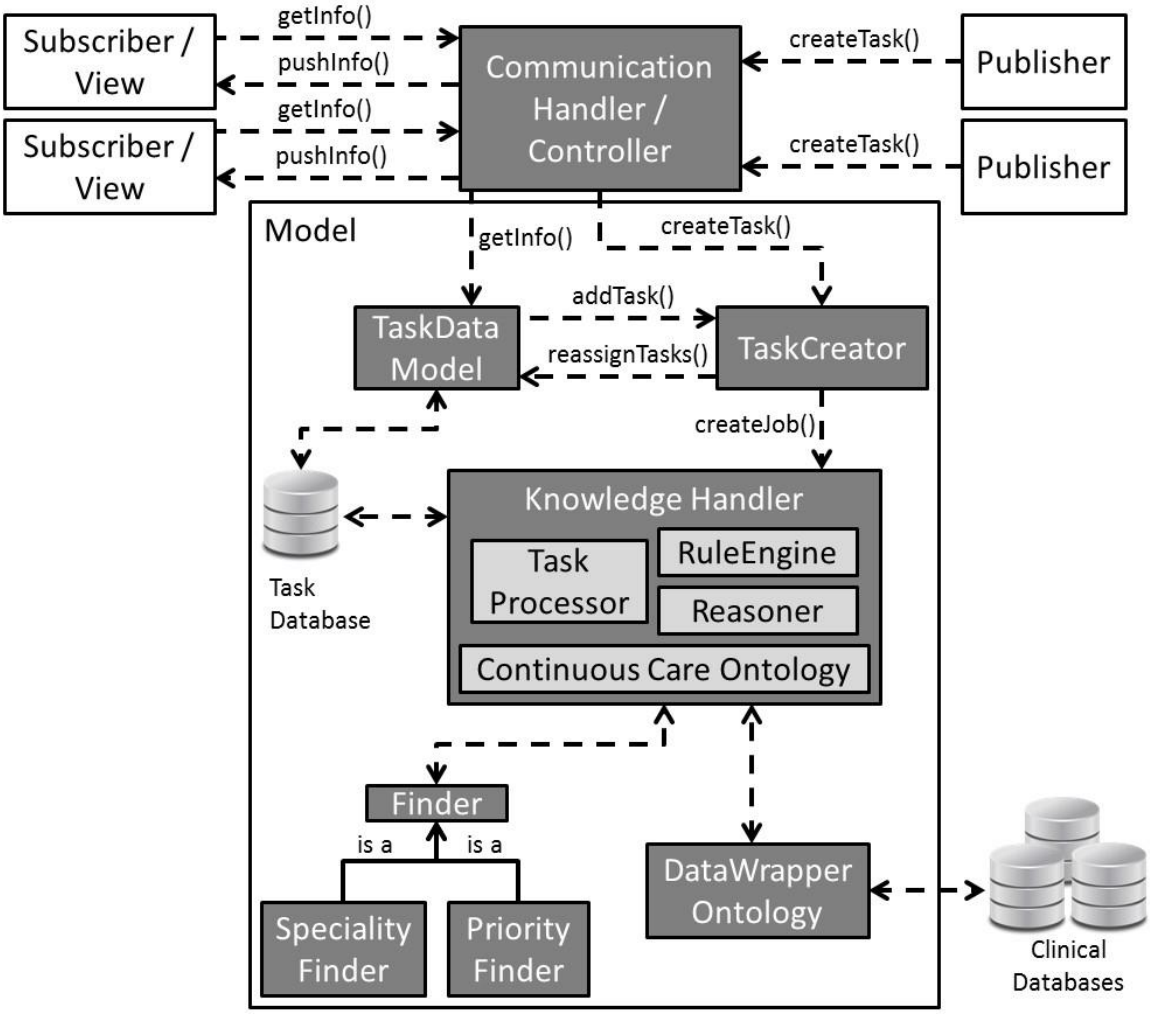


FIGURE 1 - HIGH-LEVEL ARCHITECTURE OF THE TASK MANAGEMENT SYSTEM

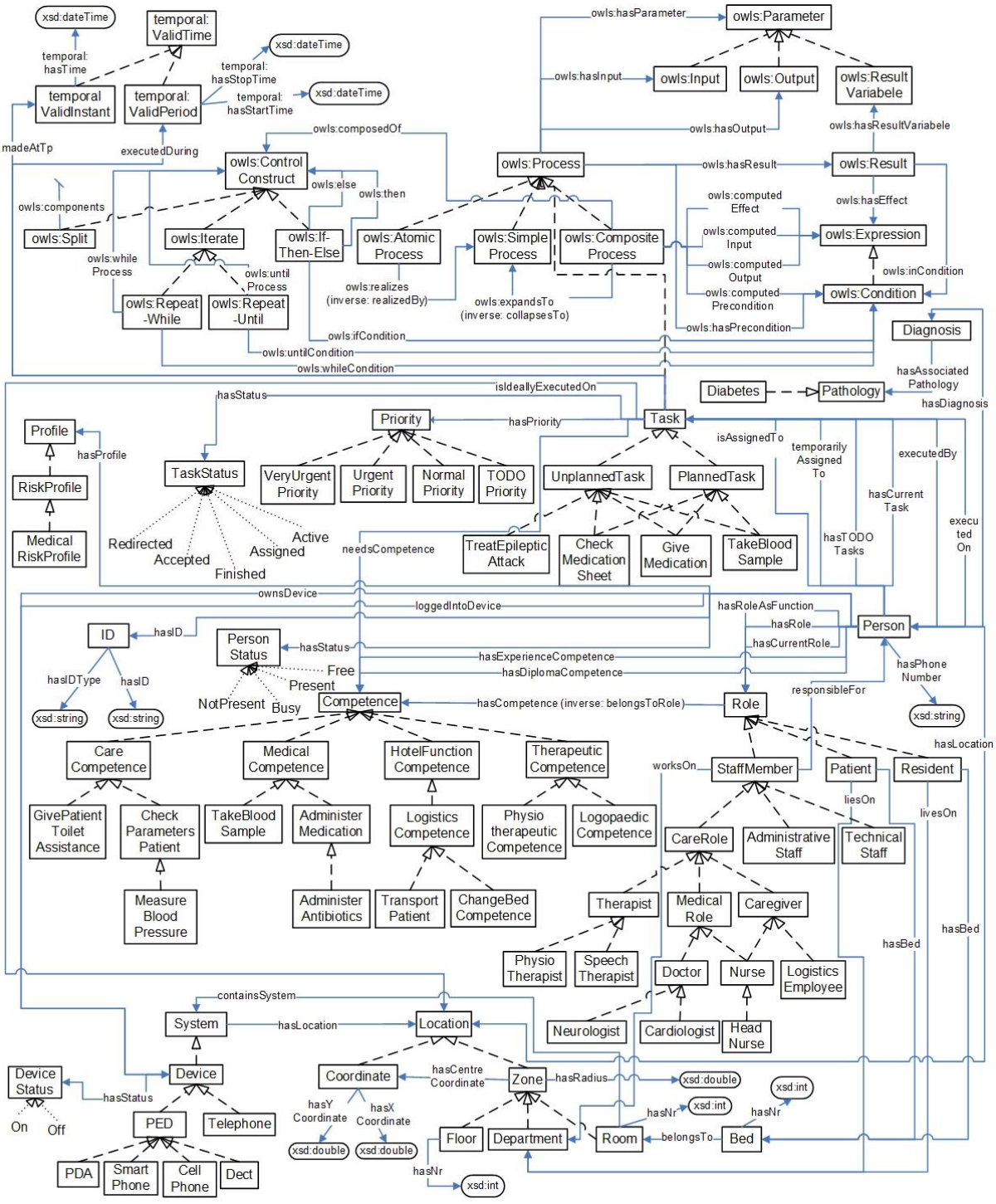


FIGURE 2 - OVERVIEW OF THE MOST PREVALENT CLASSES AND RELATIONS OF THE CONTINUOUS CARE ONTOLOGY FOR THE TASK MANAGEMENT PLATFORM

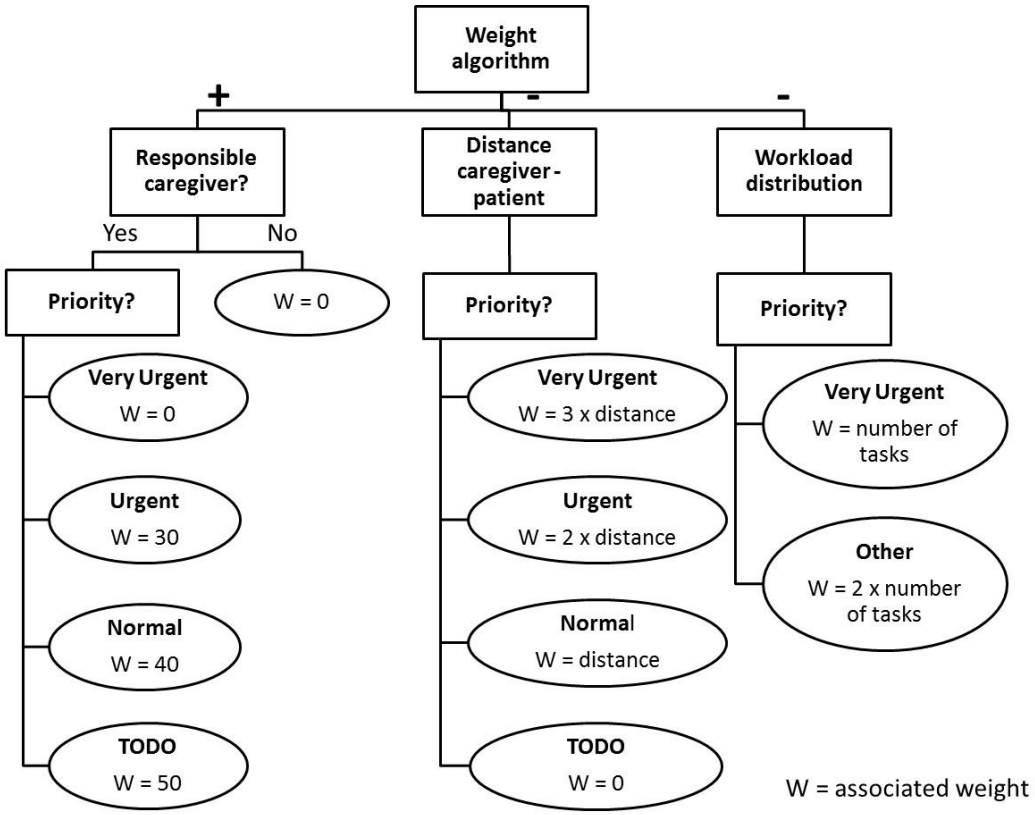


FIGURE 3 - ALGORITHM TO ASSIGN WEIGHTS TO THE QUALIFIED CAREGIVERS TO CHOOSE THE MOST APPROPRIATE ONE TO HANDLE A TASK

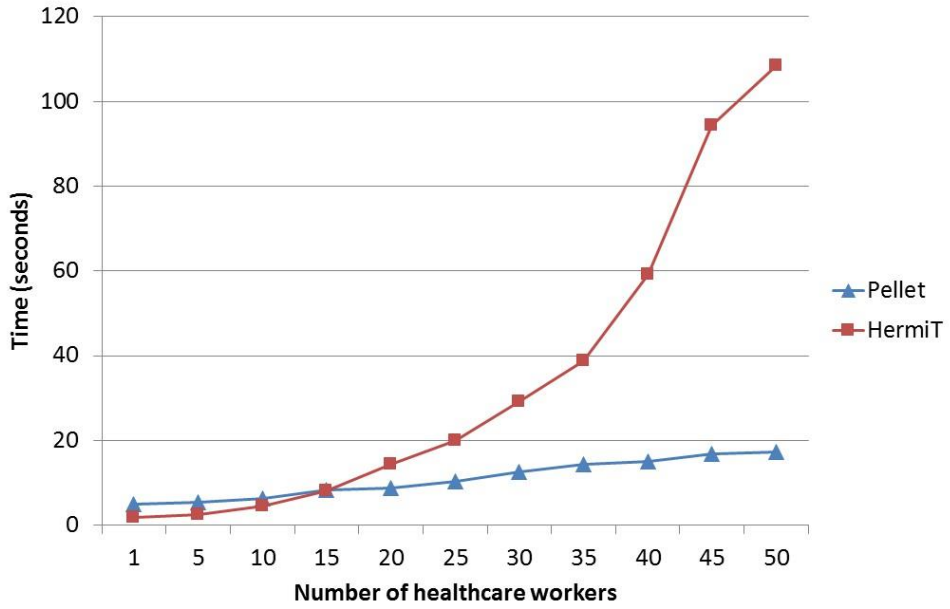
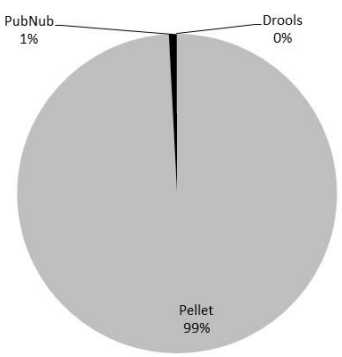
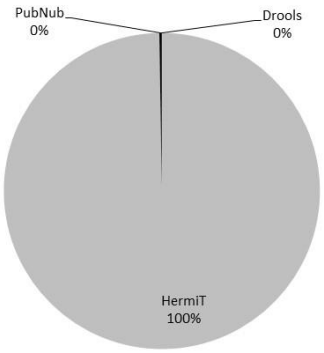


FIGURE 4 - EXECUTION TIME OF THE TASK MANAGEMENT SYSTEM AS A FUNCTION OF THE AMOUNT OF PRESENT HEALTHCARE WORKERS



B) TASK MANAGEMENT SYSTEM IMPLEMENTED WITH THE PELLET SEMANTIC REASONER



A) TASK MANAGEMENT SYSTEM IMPLEMENTED WITH THE HERMIT SEMANTIC REASONER

FIGURE 5 - DISTRIBUTION OF THE TOTAL EXECUTION TIME OF THE TASK MANAGEMENT SYSTEM ACROSS THE THREE MAIN COMPONENTS: PUBNUB, DROOLS RULE ENGINE AND THE SEMANTIC REASONER

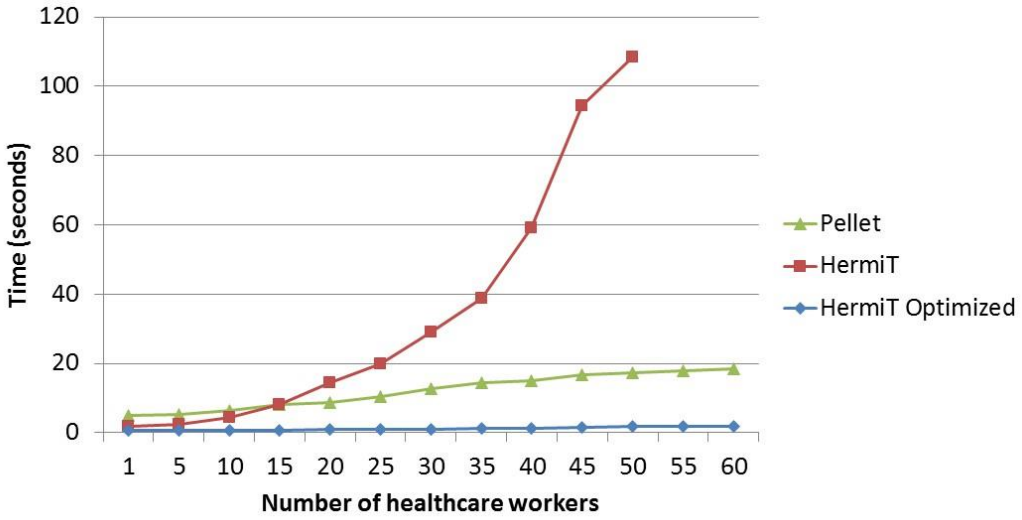


FIGURE 6 - EXECUTION TIME OF THE TASK MANAGEMENT SYSTEM AS A FUNCTION OF THE AMOUNT OF PRESENT HEALTHCARE WORKERS FOR THE THREE POSSIBLE IMPLEMENTATIONS OF THE SEMANTIC REASONING

TABLES

TABLE I: AVERAGE EXECUTION TIME OF THE DIFFERENT METHODS REQUIRING SEMANTIC REASONING

Methods	Execution time (ms)	
	Pellet	Hermit
getPossibleCandidates	16.98	0.43
getLocation	0	61.12
isResponsibleCaregiver	0	0
workloadPerson	0	0

TABLE II: AVERAGE EXECUTION TIME OF THE DIFFERENT METHODS REQUIRING SEMANTIC REASONING WITH PELLET WHEN THE GETLOCATION METHOD IS CALLED BEFORE THE GETPOSSIBLECANDIDATES METHOD

Methods	Execution time (seconds) Pellet
getLocation	15.76
getPossibleCandidates	0
isResponsibleCaregiver	0
workloadPerson	0

TABLE III: AVERAGE EXECUTION TIME OF THE DIFFERENT STEPS OF THE GETLOCATION METHOD WHEN HERMIT IS USED AS SEMANTIC REASONER

Methods	Execution time (seconds) Hermit
getObjectPropertyValues	0.18
isIndividualofClass	0
getDataPropertyValues	60.93
getLiteral	0