Achieving Pro-Active Guidance of Patients through ADL via Knowledge-Driven Activity Recognition and Complex Semantic Workflows

William Van Woensel[1], Patrice C. Roy[1] and Syed Sibte Raza Abidi[1]

[1] NICHE Research Group, Faculty of Computer Science,
Dalhousie University, Halifax, Canada
{william.van.woensel, patrice.roy, raza.abidi}@dal.ca

Abstract. Assisted Ambient Living (AAL) focuses on self-sufficiency, assisting disabled people (in particular, cognitive decline) to perform activities of daily living (ADL) such as housework and taking medication, by automating assistive actions in smart environments. We argue that AAL provides opportunities for pro-active assistance of cognitively disabled patients, which involves dynamically guiding them through an ADL and correcting their actions when required. Activity recognition is a pivotal task in this effort, since it allows detecting when an ADL is started by recognizing its constituent activities. When dealing with diseases such as cognitive decline, activity recognition should be able to detect when activities are performed incorrectly as well – e.g., performed out-of-order, at the wrong location or time, or with the wrong objects (e.g., utensils) – which is nevertheless not a common goal in knowledge-driven activity recognition. In this paper, we present an approach to computerize complex ADL workflows, using an OWL ontology to represent tasks and their temporal relations, in order to realize continuous, pro-active patient assistance. This process is supported by fine-grained, knowledge-driven activity recognition, which employs semantic reasoning to recognize both correct and incorrect actions based on their associated context and temporal relations.

Keywords: assisted ambient living, activity recognition, semantic reasoning.

1 Introduction

In the current context of ageing populations and longevity of chronic patients, reducing the burden on the healthcare system, while ensuring patients' quality of life, requires patients to remain self-sufficient for as long as possible. Assisted Ambient Living (AAL) [1] focuses on self-sufficiency, assisting disabled people (in particular, cognitive decline) to perform activities of daily living (ADL) [2, 3], such as housework, medication adherence, and healthy activities. To that end, AAL utilizes smart environments to automate assistive actions, such as influencing environment conditions (e.g., setting temperature, light intensity), providing instructions to the patient (e.g., how to perform an ADL), and issuing alerts in case of unusual activities (e.g., falling). Currently, smart hardware, including sensors, actuators and displays, is available off-the-shelf (home automation), and software frameworks [4] for interacting with smart environments, as well as ontologies (e.g., SOUPA [5], HomeADL [6]) for describing context in smart homes, have been developed and published over the last decade. Consumer smart phones and smartwatches have become ubiquitous and affordable, adding accurate personal sensors (movement, activity) and communication capabilities to a smart home.

In the context of cognitive decline, AAL provides opportunities for the pro-active assistance of patients, dynamically guiding them through an ADL while it is being performed, and correcting their actions when needed. As for any AAL process, activity recognition will be a pivotal task as it enables the detection of constituent activities of complex ADL. Data-driven, machine-learning based techniques have proven their usefulness in achieving high-accuracy activity recognition, but these require a training dataset and are less suitable to recognize high-level, complex ADL activities. Knowledge-driven activity recognition approaches [7, 8] model high-level ADL activities and their constituent actions, and utilize semantic reasoning to classify unknown activities into a known, high-level activity class. Nevertheless, when dealing with cognitive decline, coping with incorrect activities is paramount; e.g., out-of-order, at the wrong time or location, or utilizing wrong objects (e.g., utensils). To the best of our knowledge, this is not a main goal of current knowledge-driven approaches. Moreover, any real-world ADL, even relatively simple ones such as taking medication or making tea, will include complex temporal relations that go beyond what is featured in the state of the art.

We present an initial approach to computerizing complex ADL workflows, using an OWL ontology to represent ADL tasks and their temporal relations. As the main knowledge artifacts, these workflows form the basis for fine-grained, knowledge-driven activity recognition, which employs semantic reasoning to recognize both correct and incorrect actions. We formally define the semantics of our proposed temporal relations, which are borrowed from general and specialized (e.g., Clinical Practice Guideline) workflow languages, as well as high-level UI design methods, using state transition rules. On top of this component, a continuous process pro-actively guides patients through the recognized ADL, based on recognized tasks and their associated ADL workflows, by issuing prompts when necessary.

This paper is structured as follows. Section 2 proposes a set of useful temporal relations and elaborates on their formal definitions. Section 3 details the activity recognition process, and Section 4 discusses a pro-active guidance process. In Section 5, we present our prototype together with preliminary experimental results, and Section 6 summarizes related work. Finally, Section 7 presents conclusions and future work.

2 ADL Knowledge Model

2.1 Modeling Tasks and Temporal Relations

To model useful ADL workflows, we draw inspiration from formalisms for general-purpose workflows (e.g., UML activity diagrams, BPMN), computerizing Clinical Practice Guidelines (CPG), as well as high-level UI-design task models. Workflow languages typically include constructs to indicate start- and endpoints, and sequential, choice and parallel (with split and join) relations between tasks. CPG workflow languages typically support nesting as well, with high-level clinical tasks having multiple sub-tasks; and pre- and post-conditions (effects), e.g., referring to the patient's condition [9]. Also, tasks may be explicitly assigned to a patient or physician. In the UI design domain, Paterno et al. introduced the ConcurTaskTree method [10] for designing high-level UI task models. CTT similarly supplies a hierarchical structuring of tasks, a set of temporal relations, task assignment to different parties, and associated task objects/attributes. To reflect the logical task structure, CTT utilizes a tree-like hierarchical graphical syntax. Applying a UI design formalism in this setting is not as unfit as it may sound: comparable to using a PC, users interact with a smart home, utilizing smart hardware and everyday household objects outfitted with embedded sensors. Indeed, the strong focus of CTT on task hierarchies, as well as its set of diverse temporal relations, suits the context of AAL quite well; ADLs are typically decomposable into multiple levels of tasks, and many temporal constraints bind the correct performance of an ADL.

We propose six workflow relations when modeling ADL (based on Paterno et al. [10]). We note that their precedence (i.e., when used at the same hierarchy level) is in the same order as presented.

– *Hierarchical* (tree structure): indicates the decomposition of a composite task into multiple lower-level tasks. The composite task is considered complete once all of its (non-optional) constituent tasks are completed.
– *Sequential* (T_1 >> T_2): defines an ordering between the two operands; i.e. the second task (T_2) cannot be begin before the first task is completed (T_1).

  We further introduce the *timeout-sequential conditional-sequential* subtypes:
  - *Timeout-sequential* (T_1 $>>_{<timeout>}$ T_2): after the first task (T_1) is completed, the second (T_2) may only start once a certain timespan has passed.
  - *Conditional-sequential* relation (T_1 $>>_{[condition]}$ T_2): after the first task (T_1) is completed, the second task (T_2) may only start once a condition is fulfilled.
– *Order independent* (T_1 |=| T_2): the operands (T_1, T_2) can be performed in any order.
– *Alternative* (T_1 [] T_2): the user can choose to perform either task operand; once a choice is made, the other task can no longer be started.

  We further introduce the following subtype:
  - *Conditional-alternative* (T_1 [cond] T_2): the left operand (T_1) can only be performed if the condition is satisfied; else, the right operand (T_2) is performed.
– *Optional* ([T]): performing the task is not mandatory and may be skipped.
– *Iterative* ($T_{\{n, m\}}$): a task may be carried out once or multiple times

Compound tasks are indicated as an empty circle (○); depending on whether an atomic task is assigned to the user (i.e., patient) or to the smart home system, a filled circle (●) or a filled square (■) is utilized, respectively. Fig. 1 shows an example ADL for taking medications, decomposed using the presented workflow relations. As can be seen, the ADL workflow allows many degrees of freedom: getting a glass and opening the water tap can be done in any order, as well as closing the tap and putting the glass away. However, both the former two tasks (*GetWater*) must be completed before the latter two tasks (*StopGettingWater*); it would not make sense to put the glass away before the water tap is opened or vice-versa, to close the tap before getting a glass. This sequential relation can have either a time interval (3s) or condition (glass half-full) depending on available sensors. Note that getting a pill, which consists of opening and then closing the pillbox, may also be interleaved (order independent) with any of the other tasks.

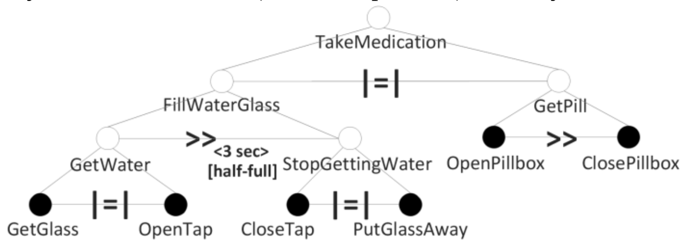

Fig. 1. Example ADL workflow, *TakeMedication*.

2.2 SmartAssist Ontology (SAO)

In this section, we introduce the SmartAssist Ontology (SAO), which formally defines ADL tasks and the proposed set of temporal relations (Section 2.1). Fig. 1 shows an ER diagram with the classes and relations in this ontology.

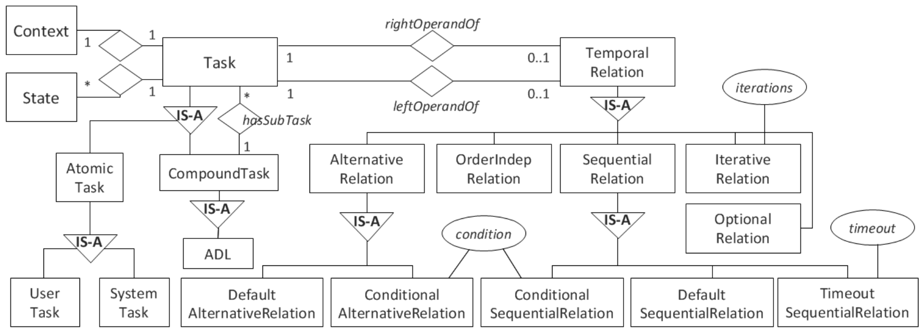

Fig. 2. SmartAssist Ontology: classes and relations.

Note that the ontology also connects each *Task* to a particular *Context* with regards to e.g., time, location and utilized objects. To support the formal semantics of temporal relations, each task also has an associated *State (inactive*, *active*, *started*, *completed*, or *error)*. Fig. 3 illustrates these states and the potential transitions between them.

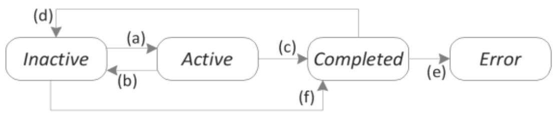

Fig. 3. Task states and transitions between them.

A task is *active* when it is next in line for execution, according to the workflow of the ADL (multiple tasks may be *active* at the same time). Inversely, *inactive* means that the task should not be executed at this time. The *completed* state indicates that the task was executed by the user. In case an error is detected with regards to the task's execution, the state will transition to the *error* state.

The formal semantics of a temporal relation is defined in terms of rules that govern transitions between task states, in line with their descriptions from Section 2.2. These formal semantics are included in the OWL ontology. We list them below using Description Logic notation [11]. To simplify these DL expressions, we assume each *Task* is assigned a type based on its current state: e.g., a task in the *completed* state will be assigned the *Completed* type; and a task in any other state will be assigned *Incompleted*. Each task is initially in the *inactive* state. We note that these rules focus on the temporal relations in particular, and do not consider e.g., checking context (e.g., time, location).

(a) **Inactive → Active**

(a.1) $ADL \sqcup (Task \sqcap \exists subTaskOf.(Active) \sqcap \neg(\exists rightOperandOf.(TemporalRelation))) \sqsubseteq \exists hasState.\{active\}$

(a.2) $Task \sqcap \exists rightOperandOf.(SequentialRelation \sqcap \exists hasLeftOperand.(Completed \sqcup Optional)) \sqsubseteq \exists hasState.\{nextInSequence\}$

(a.3) $NextInSequence \sqcap \exists operandOf.(DefaultSequentialRelation) \sqsubseteq \exists hasState.\{active\}$

(a.4) $NextInSequence \sqcap \exists operandOf.(ConditionalSequentialRelation \sqcap \exists hasCondition.(Satisfied)) \sqsubseteq \exists hasState.\{active\}$

(a.5) $NextInSequence \sqcap \exists operandOf.(TimeoutSequentialRelation \sqcap \exists hasTimeout.(Done)) \sqsubseteq \exists hasState.\{active\}$

(a.6) $Task \sqcap \exists operandOf.(OrderIndepRelation \sqcap \exists hasOperand.(Active)) \sqsubseteq \exists hasState.\{active\}$

(a.7) $Task \sqcap \exists operandOf.(DefaultAlternativeRelation \sqcap \exists hasOperand.(Active)) \sqsubseteq \exists hasState.\{active\}$

(a.8) $Task \sqcap \exists rightOperandOf.(ConditionalAlternativeRelation \sqcap \exists hasLeftOperand.(Active) \sqcap \exists hasCondition.(Unsatisfied)) \sqsubseteq \exists hasState.\{active\}$

(b) **Active → Inactive**

(b.1) $Task \sqcap Active \sqcap \exists leftOperandOf.(ConditionalAlternativeRelation \sqcap \exists hasCondition.(Unsatisfied)) \sqsubseteq \exists hasState.\{inactive\}$

(c) **Active → Complete**

(c.1) $Task \sqcap \forall operandOf(AlternativeRelation \sqcap \exists hasOperand.(Completed)) \sqsubseteq \exists hasState.\{completed\}$

(c.2) $CompoundTask \sqcap \neg(\exists hasSubTask.(Incompleted)) \sqsubseteq \exists hasState.\{completed\}$

(d) **Completed → Inactive**

(d.1) $Task \sqcap Active \sqcap Completed \sqcap (ADL \sqcup (\exists transitiveSubTaskOf.(ADL \sqcap Completed)) \sqsubseteq \exists hasState.\{inactive\}$

(e) **Completed → Error**

(e.1) $Task \sqcap Inactive \sqcap Completed \sqsubseteq \exists hasState.\{error\}$

Rule (a.1) allows each ADL to be started at any time, by assigning the active state to all high-level ADL tasks as well as each (constituent) “starting” task, i.e., not occurring as a right relation operand and with an active supertask. (As noted, this rule would need to be extended to check whether associated context is satisfied.). Firstly, this rule “left-activates” each ADL, assigning the active state to all of its left-most tasks (at any hierarchy level) – meaning the user may complete them at any time, and thus start the ADL. Secondly, when a compound task is activated in another way, this rule will activate its left-most subtask(s), thus allowing it to be completed by the user.

Rule (a.2) ensures that the right operand task of a sequential relation is only activated when the relation's left operand task is either *Completed* or *Optional*. It does this by assigning the *nextInSequence* temporary state to the right operand task, which is utilized by rules (a.3) – (a.5) to then activate the right operand task by default (a.3), or only when its given condition is met (a.4) or timeout has elapsed (a.5). Rule (a.6) and (a.7) state that any operand task of an order independent relation, or a default alternative relation (i.e., no associated condition), becomes active when one of the operand tasks was activated (e.g., due to rule (a.1) or (a.3)-(a.5)). When the relation's condition is not met, rule (a.8) *activates* the right operand of a *conditional-alternative* relation, and rule (b.1) *deactivates* the left operand task. Else, the left operand task will simply remain active (since it will have been left-activated). Rule (c.1) ensures that a task in an *alternative* relation is completed whenever the other operand task is completed, meaning the user may only execute one alternative task. Rule (c.2) marks a compound task as completed once it no longer has any incomplete subtasks. Note that an atomic task will be marked as completed depending on the detected context (Section 3). Once the ADL is completed, completed constituent tasks, as well as the high-level ADL task, will transition back to inactive (d.1). In case a task was completed while being in the inactive state (transition f), rule (e.1) assigns the error state to a task, since it was not executed in line with the ADL workflow. Note that detected context may also indicate an error (e.g., using the wrong utensil). Finally, although a task can thus be in multiple states (e.g., (d.1), (e.1)), the *inactive* state is retracted once a task is activated.

3 Knowledge-Driven Activity Recognition Using SAO

By applying semantic reasoning based on the proposed task state transition rules (Section 2.2), based on ADL workflows and detected user actions (i.e., low-level tasks), our approach is able to recognize the current states of ADL activities (i.e., high-level tasks). In doing so, we realize high-level, knowledge-driven activity recognition (AR): recognizing the start and completion of higher-level tasks, based on the execution of their constituent, lower-level tasks (hierarchical relation); and flagging incorrectly performed tasks (e.g., out-of-order) as erroneous, based on temporal relations. On top of state transition rules, the system will also raise an error when, once the ADL is started (i.e., one of its constituent atomic tasks is completed), an activated atomic task is not completed within a reasonable time. We note that this approach allows dealing with multiple, simultaneously started ADL, with the user performing their constituent tasks in any interleaved way. To detect the actions that drive a knowledge-driven AR, i.e., indicating task completion and individual task errors, we rely on sensor data processing techniques, as elaborated by Ni et al [12]. Each time an atomic action is detected, the semantic reasoning process is executed. Activity recognition results are passed to the top-level component, i.e., knowledge-driven pro-active assistance (Section 4).

4 Knowledge-Driven, Pro-active Assistance using SAO

Pro-active assistance can be divided into two facets: (1) guidance and (2) troubleshooting. In the first facet, assistive acts are issued to guide the patient through their daily ADL routines. When an ADL is overdue (e.g., based on its associated context), the system prompts the patient with increased urgency until the ADL is carried out, thus ensuring that their daily routine continues as expected. In line with the increased urgency of executing the ADL as time passes, we apply an *evolving* notification lifecycle [13]; i.e., where interaction resources (e.g., icons, audio & haptic feedback) increase in obtrusiveness over time, together with notification frequency, until the activity is performed. (This step is not the focus of this paper.) Secondly, our knowledge-driven approach allows, once an ADL is started, to keep the patient appraised of their current progress in the workflow. E.g., in case a patient is carrying out a cooking activity, assistive acts provide information about the activity's progression (current subtask) and its current state (instructions, location of utensils and ingredients, etc.) on the patient's smartphone or nearby devices (e.g., TV or tablet) [14, 15].

The second facet, i.e. troubleshooting, occurs when an activity is performed incorrectly. In that case, assistive acts are needed to prompt the patient about the error and explain the appropriate workflow. There are multiple ways in which a patient can incorrectly carry out an activity. First, an error occurs when the user did not perform an activated atomic task in time while carrying out an ADL, which will lead to the system issuing a reminder to the patient about the started activity. Our novel activity recognition process is able to detect a second type of error as well, occurring when a task was recognized (i.e., completed) but not yet activated (Section 2.2, rule (e)). In this case, the system prompt depends on the relevant temporal relation(s); we note that the recognized task may contradict more than one temporal relation (e.g., the completed task is located at the end of the workflow but currently active tasks are at the front).

5 Knowledge-driven Activity Recognition Prototype

We implemented a prototype utilizing the SAO ontology to evaluate the feasibility of our activity recognition approach. The SAO prototype ontology[1] includes the *TakeMedication* workflow from Fig. 1. Our prototype performs the following operations, utilizing the Hermit reasoner (v. 1.3.8.4) for the semantic reasoning step, i.e., reasoning over the DL transition rules:

- *Ontology loading*: utilizing the Hermit API to load the SAO ontology and encoded ADL workflow.
- *Initialization*: performing the semantic reasoning step once after loading, to "left-activate" each ADL.
- *Detected action*: performing the semantic reasoning step after a single low-level action has been detected. An error will be flagged if the detected action is not in line with the temporal relations in the ADL workflow.

[1] https://niche.cs.dal.ca/ontologies/sao.owl

We performed a preliminary evaluation of our activity recognition approach, executing 10 simulated scenarios for the *TakeMedication* ADL (Fig. 1) with the patient performing the 6 low-level tasks (actions) in different orders; with 5 scenarios where the patient "correctly" performed the ADL and 5 scenarios where the ADL is "incorrectly" performed (i.e., out-of-order tasks). We ran the experiments on a Lenovo ThinkPad T530 laptop running Windows 7, with an Intel Core i7-3520M CPU (2.90 GHz) and 8Gb of RAM. We ran each scenario 10 times and retrieved the average performance results of each operation (Table 1). The prototype was able to properly detect each of the incorrectly performed actions in the simulated scenarios.

Operation	**Average processing time (ms)**
Ontology loading	26
Initialization	148
Detected action	214

Table 1. Activity recognition performance.

6 Related Work

Data-driven activity recognition is defined as applying machine learning techniques to train an activity model, based on a (labeled) dataset. Such approaches do not need an a priori designed knowledge artifact (e.g., workflows), and have been shown to achieve high accuracy. However, they require an initial, patient-specific dataset, and are less suitable for recognizing high-level, complex activities [8]. Instead, knowledge-driven activity recognition relies on an a priori designed knowledge artifact. A subcategory of knowledge-driven approaches (including ourselves) applies logical reasoning to infer high-level activities from detected low-level actions. To that end, Chen et al. [7] defines a set of activity classes with constraints on which properties (e.g., hasLocation, hasContainer) and which values (e.g., HotDrink, Kitchen) they can be associated with. At runtime, low-level sensor observations (e.g., type of container, location) are attached to an object instance, and activity recognition is executed by attempting to classify the instance as an activity class (e.g., MakeDrink). As more properties are attached to the instance, it becomes possible to classify it as a more specific activity (e.g., MakeTea). The authors raise the option of reminding patients when an activity cannot be recognized in time, but do not elaborate on such guidance. A clear drawback of this approach is its inability to cope with tasks being performed incorrectly or in the wrong temporal sequence, since this would lead to an incorrect classification. Also, no temporal relations, aside from order independent, are considered.

Helaoui et al. [8] aim to solve the uncertainty issue by representing activity models using log-linear DL, which integrate Description Logics with probabilistic log-linear models. Similar to Chen et al., the work proceeds by applying semantic reasoning to progressively infer high-level activities; in case of incompatibilities due to (inaccurate/incomplete) sensor observations, the most probable activity is inferred based on confidence values of the different activity definitions. Their approach only considers a *sequential* temporal relation, which is represented via an ordinal number associated with a subtask; this means that other temporal relations cannot be plugged in without significantly restructuring the ontology. Okeyo et al. [16] incorporates composite, interleaving and concurrent task relations based on Allen's temporal logic, and AR is realized using SWRL rules, However, the approach does not support conditional, alternative or optional relations, and does not focus on dealing with incorrect actions.

We note that our approach is inspired by our work in runtime CPG execution [17, 18], where current task states in the CPG workflow depends on patient and physician actions together with utilized workflow constructs (e.g., decision, pre-conditions).

7 Conclusions and Future Work

In this paper, we proposed a novel approach to computerizing complex ADL workflows. We presented the SmartAssist Ontology (SAO), which formally defines these temporal relations via a set of rules governing the transition between task states. A fine-grained, knowledge-driven activity recognition process utilizes these state transition rules to perform high-level activity recognition. Based on recognized activities, a proactive assistance process then realizes guidance features such as keeping the patient appraised of their progress and next steps; and correcting the patient's actions in case they are incorrect or performed in the wrong temporal sequence. Compared to other knowledge-driven approaches, our system allows defining rich, hierarchical ADL workflows utilizing a diverse set of temporal relations, as well as coping with incorrectly performed activities – an important feature when dealing with cognitive decline, as is common in AAL.

Future work involves studying how to deal with uncertainty resulting from faulty sensor observations, e.g., by utilizing Probabilistic Description Logics [19]. Allowing for the personalization of ADL workflows, albeit manually or (semi-)automatically based on periodic patterns (e.g., [20]), is an avenue of future work as well. An important goal is to combine our knowledge-driven activity recognition approach with behavioral self-management, which computerizes behavioral strategies to engage patients to perform health activities; including exercise routines, avoiding unhealthy activities (smoking, alcohol abuse, unhealthy diet), and complying with medication regimen. In case of non-adherence, as detected using activity recognition, ambient self-management systems can remind patients of their prescribed regimen or suggested activities, while also educating them on importance of compliance.

References

1. Rashidi, P., Mihailidis, A.: A Survey on Ambient-Assisted Living Tools for Older Adults. IEEE J. Biomed. Heal. Informatics. 17, 579–590 (2013).
2. Katz, S., Ford, A.B., Moskowitz, R.W., Jackson, B.A., Jaffe, M.W.: Studies of illness in the aged: the index of ADL: a standardized measure of biological and psychosocial function. JAMA. 185, 914–919 (1963).
3. Lawton, M.P.P., Brody, E.M.M.: Assessment of Older People: Self-Maintaining and Instrumental Activities of Daily Living. Gerontologist. 9, 179–186 (1969).
4. Razzaque, M.A., Milojevic-Jevric, M., Palade, A., Clarke, S.: Middleware for Internet of

Things: A Survey. IEEE Internet Things J. 3, 70–95 (2016).
5. Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: standard ontology for ubiquitous and pervasive applications. The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. pp. 258–267 (2004).
6. Hong, X., Nugent, C.D., Finlay, D.D., Mulvenna, M.: HomeADL for adaptive ADL monitoring within smart homes. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. 2008, 3324–3327 (2008).
7. Chen, L., Nugent, C.D., Wang, H.: A Knowledge-Driven Approach to Activity Recognition in Smart Homes. IEEE Trans. Knowl. Data Eng. 24, 961–974 (2012).
8. Helaoui, R., Riboni, D., Stuckenschmidt, H.: A Probabilistic Ontological Framework for the Recognition of Multilevel Human Activities. Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 345–354. ACM, New York, NY, USA (2013).
9. Jafarpour, B.: Ontology Merging Using Semantically-Defined Merge Criteria and OWL Reasoning Services: Towards Execution-Time Merging of Multiple Clinical Workflows to Handle Comorbidities, (2013).
10. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer-Verlag, London, UK, UK (1999).
11. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The description logic handbook: theory, implementation, and applications. (2003).
12. Ni, Q., Garcia Hernando, A.B., de la Cruz, I.P.: The Elderly's Independent Living in Smart Homes: A Characterization of Activities and Sensing Infrastructure Survey to Facilitate Services Development. Sensors (Basel). 15, 11312–11362 (2015).
13. Van Woensel, W., Roy, P.C., Abidi, S.R., Abidi, S.S.R.: A Mobile and Intelligent Patient Diary for Chronic Disease Self-Management. Stud. Heal. Tech. Inf. 216, 118–122 (2015).
14. Groussard, P.-Y., Bier, N., Giroux, S., Pigot, H., Macoir, J., Milhau, J., Descheneaux, C., Roy, P., Arab, F., Chikhaoui, B., Medini, S., Kammoun, M.F., Parakh, Y.: SemAssist: Assistance and assessment tools for semantic memory rehabilitation. Gerontechnology. 9, 106–107 (2010).
15. Bier, N., Macoir, J., Joubert, S., Bottari, C., Chayer, C., Pigot, H., Giroux, S., Team, S.: Cooking "Shrimp à la Créole": A pilot study of an ecological rehabilitation in semantic dementia. Neuropsychol. Rehabil. 21, 455–483 (2011).
16. Okeyo, G., Chen, L., Wang, H.: Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. Futur. Gener. Comput. Syst. 39, 29–43 (2014).
17. Jafarpour, B., Abidi, S., Abidi, S.: Exploiting Semantic Web Technologies to Develop OWL-Based Clinical Practice Guideline Execution Engines. IEEE J. Biomed. Heal. Informatics. PP, 1–1 (2014).
18. Jafarpour, B., Abidi, S.R., Abidi, S.S.R.: Exploiting OWL Reasoning Services to Execute Ontologically-Modeled Clinical Practice Guidelines. 13th Conference on Artificial Intelligence in Medicine, AIME 2011, Bled, Slovenia, July 2-6, 2011. pp. 307–311 (2011).
19. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Int. 172, 852–883 (2008).
20. Valiente-Rocha, P.A., Lozano-Tello, A.: Ontology and SWRL-Based Learning Model for Home Automation Controlling. Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence. pp. 79–86. Springer Berlin Heidelberg (2010).