An Ontology-based Hybrid Approach to Activity Modeling for Smart Homes

Liming Chen, *Member, IEEE*, Chris Nugent, *Member, IEEE*, and George Okeyo, *Member, IEEE*

***Abstract*—Activity models play a critical role for activity recognition and assistance in ambient assisted living. Existing approaches to activity modeling suffer from a number of problems, e.g. cold-start, model reusability and incompleteness. In an effort to address these problems, we introduce an ontology-based hybrid approach to activity modeling that combines domain knowledge based model specification and data-driven model learning. Central to the approach is an iterative process that begins with "seed" activity models created by ontological engineering. The "seed" models are deployed, and subsequently evolved through incremental activity discovery and model update. While our previous work has detailed ontological activity modeling and activity recognition, this paper focuses on the systematic hybrid approach and associated methods and inference rules for learning new activities and user activity profiles. The approach has been implemented in a feature-rich assistive living system. Analysis of the experiments conducted has been undertaken in an effort to test and evaluate the activity learning algorithms and associated mechanisms.**

***Index Terms*—activity model learning, activity recognition, ontology, semantic reasoning, smart homes.**

I. INTRODUCTION

SMART Homes (SH) have been widely accepted as being a promising paradigm for technology-driven assistive living for the aging population [1]. A SH can be described as a home environment augmented with a diversity of multi-modal sensors, actuators and devices along with information and communication technology (ICT) based services and systems [2]. By monitoring environmental changes and inhabitants' activities, an assistive system within a SH can process sensor data, infer an inhabitant's needs and take appropriate actions to support Activities of Daily Living (ADLs). As such, a SH can help older people prolong their independent living and enhance quality of life within their own homes.

Activity models play a crucial role in the realization of the SH concept. They are required to support reasoning based upon real-time streaming sensor data in order to infer the current activity for application-level functions. This may include, for example, to predict the next action within a specific task or to detect anomalies within the ADLs currently being undertaken. The completeness and accuracy of ADL models is therefore critical for an assistive system to function correctly. If an activity is not modeled or the model is not accurate, the activity will not be recognized by an assistive system. The system will therefore not be able to provide assistance and/or prediction with regard to this activity.

Modeling ADLs is a challenging task due to their unique characteristics. For example, there are a large number of ADLs in a diversity of categories which can all be modeled at multiple levels of granularity [4]. In addition, most ADLs involve performing a number of actions. The sequence of the actions to be performed is usually dependent on an individual's own preferences. Furthermore, the manner an ADL is performed evolves dynamically, for example the change in duration or the order of objects being used within a task. This is particularly the case for older people and those suffering from decline of cognitive capabilities.

Currently there are two mainstream approaches to modeling ADLs. One approach is to learn an individual's activity models from existing behavioral datasets using data mining and machine learning techniques. With this approach activity models are created based on two tasks, namely the creation of a probabilistic or statistical activity model and the training of the model to decide its parameters or mappings [5]-[17]. Given that the approach is based on intensive data analysis, it is usually referred to as a data-driven approach. A data-driven approach to ADL modeling has three major drawbacks. The first is the well-known cold-start problem, i.e. requiring a large representative dataset to support model training for each ADL. This problem is exacerbated in the context of assistive living as people are reluctant to disclose their behavioral data due to privacy and ethical considerations. The second drawback is related to model applicability and reusability. A data-driven approach is sensitive to unseen data which makes it difficult to apply the ADL models which have been learnt from one person to another person. This means that with a data-driven approach an activity model for different users has to be learnt separately. The third drawback is the incompleteness of activity models which is closely related to the aforementioned two issues. With the data-driven approach every activity model for all ADLs for every user needs to be learnt in order to create complete ADL models. Given the large number of ADLs and the cold-start problem, this is a huge challenge, if indeed not

Manuscript received on 3rd Feb. 2013, revised on 12th June and 12th August, accepted on 5th September 2013.

L. Chen is with the School of Computer Science and Informatics, De Montfort University, UK. CD. Nugent and G. Okeyo are with the School of Computing and Mathematics, University of Ulster, UK. Tel and fax: 44-28-90368837; (e-mail: limingchen1027@gmail.com; cd.nugent@ulster.ac.uk, Okeyo-G@email.ulster.ac.uk).

impossible, in practice. To mitigate the aforementioned problems, researchers have recently started applying transfer learning techniques to activity modeling and recognition by reusing resources and knowledge. This involves transferring the source datasets, or features or models, from one user to another in different settings [18] [19] [20]. Nevertheless, such research is still at its infancy with many open challenges [21].

An alternative to the data-driven approach is to manually define activity models by making use of rich, prior knowledge and domain heuristics. This approach is motivated by the observation that most ADLs are daily routines which normally take place within a specific circumstance of time, location and space with relatively fixed types of objects. Using formal knowledge acquisition and modeling technologies activity models can be created by means of various knowledge modeling tools [22]-[33]. As this approach is closely related to knowledge engineering, it is referred to as a knowledge-driven approach. A knowledge-driven approach overcomes the cold-start problem and improves model reusability by modeling activities at multiple levels of abstraction to create both generalized and specialized ADL models. For example, ontological activity modeling can model a generic ADL as an ontological activity class and an individual-specific ADL as an instance of the corresponding activity class. Nevertheless, given that ADL models are created manually by domain experts on a case-by-case basis, the approach is questionable in relation to its scalability of generating complete ADL models. As such, it also suffers from model incompleteness. In addition, ADL models created by knowledge-driven approaches are perceived as being generic and static. Adapting an individual's ADL models to their changing behaviors is still an open issue.

Rather than trying to reuse resources and knowledge among different users similar to the scenario with transfer learning based research, this paper introduces an ontology-based hybrid approach by incorporating data-driven learning capabilities into a knowledge-driven approach to address the aforementioned problems of activity modeling. The rationale is to provide generic activity models suitable for all users and then create individual activity models through incremental learning. The approach uses semantic technologies as a conceptual backbone and technology enablers for ADL modeling, classification and learning. The distinguishable feature of the approach from existing approaches is that ADL modeling is not a one-off effort, instead, a multi-phase iterative process that interleaves knowledge-based model specifications and data-driven model learning. The process consists of three key phases. In the first phase the initial seed ADL models are created through ontological engineering by leveraging domain knowledge and heuristics, thus solving the cold-start problem. Ontological activity modeling creates activity models at two levels of abstractions, namely as ontological activity concepts and their instances respectively. Ontological activity concepts represent generic coarse-grained activity models applicable and reusable for all users, thus solving the reusability problem. The seed ADL models are then used in applications for activity recognition at the second phase. In the third phase, the activity classification results from the second phase are analyzed to discover new activities and user profiles. These learnt activity patterns are in turn used to update the ADL models, thus solving the incompleteness problem. Once the first phase completes, the remaining two-phase process can be iterated many times to incrementally evolve the ADL models, leading to complete, accurate and up-to-date ADL models.

This paper makes three main contributions. Firstly, we develop a hybrid approach to activity modeling that combines the strengths of data- and knowledge-driven approaches to support an incremental modeling process. The approach is built upon the work in [31], however, extends it by incorporating the learning capabilities to provide a viable solution for addressing existing problems relating to ADL modeling. Secondly, we develop a learning method to discover activities that are performed by users but have not yet been modeled. Thirdly, we define the characteristics of a user activity profile and develop analysis methods and associated inference rules to learn a user's activity profiles, i.e. the specific way the user performs activities. The learning methods of activity profiles can detect the changing manner an activity is performed, thus allowing ADL models to adapt over time. We have implemented the approach in a feature-rich assistive living system. ADL discovery algorithms and profile learning methods have been tested and evaluated in a number of experiments by participants in a real sensorised environment. Initial results have demonstrated that the approach works and the algorithms are effective.

It is worth noting that the research presented in this paper is based on single-user single-activity scenarios. While complex activity scenarios, e.g. interleaved and concurrent activities, pose many research problems, it is beyond the scope of this paper to address them all. In addition, the activities this research is concerned with are basic ADLs and instrumental ADLs [45] which can be performed within home environments with clear model semantics, such as meal and drink preparation. Instrumental ADLs such as shopping and use of transportation which take place outside residential environments, and money management and housekeeping which do not have meaningful computational models, require special treatment, and are therefore also considered to be beyond the scope of this paper. Activity monitoring in this study is based on dense sensing [3], i.e. one miniaturized sensor is attached to individual objects that are used for monitoring individual tasks within ADLs. As such, by analyzing an inhabitant's interactions with objects of interest it is possible to infer the inhabitant's activity.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 introduces the hybrid approach and its core technical underpinnings. Section 4 describes learning methods to discover new activities and Section 5 presents analysis mechanisms for learning user

profiles. We discuss implementation, testing and evaluation in Section 6 and conclude the paper in Section 7.

II. Related Work in Activity Modeling

The data-driven approach to activity modeling contains two categories of methods. One of them involves the use of parametric probabilistic or statistical models to represent activities. Individual activity models are obtained by learning the structure and parameters through model training based on large-scale datasets. Major models in this category include naïve Bayes classifiers [5], Hidden Markov Models (HMMs) [6], Dynamic Bayesian Networks (DBN) [7], hierarchical clustering [8], partially observable Markov decision processes (POMDPs) [9] and the variants of HMM and DBN, e.g. Coupled Hidden Markov Models (CHMMs) [10] and linear dynamical system (LDS) [11].

The other category of data-driven activity modeling is to use classification techniques to establish the mapping from inputs of sensor data to outputs, i.e., activity labels. Methods in this category compare a sequence of sensor observations to a set of template sequences in a training dataset. Individual activity models are obtained by learning the activity labels of the most closely matching sequences in the training dataset. Examples of this type of method include nearest neighbor [12], Support Vector Machine (SVM) [13], Conditional Random Field (CRF) [14], decision trees [15], hierarchical CRF [16] and meta-level classifiers that combine the results of multiple base-level classifiers [17]. The advantage of data-driven activity modeling is that it can handle noisy, uncertain and incomplete data in addition to temporal information. Nevertheless, the approach suffers from several drawbacks as discussed in the previous Section.

The knowledge-driven approach to activity modeling also consists of two types of methods. The first is to discover activity models from existing publicly available sources, e.g. recipe handbooks or activity specifications on the Web. This type of method uses information retrieval and analysis techniques to retrieve activity definitions from specific sources and then extracts phrases and relationships to create activity models. Relevant studies in this area include [22] - [24].

The second type of method for knowledge-driven activity modeling is to view an activity model as a knowledge model. As such, activity modeling is essentially equivalent to knowledge modeling that can be formally performed using various knowledge engineering techniques and representation formalisms. There exists a number of works in this strand in terms of the underlying knowledge representation theories and formalisms. Kautz *et al*. [25] treated activity models as plans in the context of plan recognition, e.g. using first-order axioms to build a library of hierarchical plans. Wobke [26] used situation theory to address the different probabilities of inferred plans by defining a partial order relation between plans in terms of levels of plausibility. Bouchard *et al*. [27] used action description logic to formalize actions, entities and variable states in a SH within which an activity was modeled as a sequence of actions and represented as a lattice structure. Chen *et al*. [28] adopted the highly developed logical theory of actions, i.e., the event calculus for formalizing domain theories of a SH in which activities are modeled as sequential and/or parallel events. More recently, ontologies and semantic technologies have been used for activity modeling and representation. For example, Chen *et al*. have developed sensor ontologies for contextual data management [29] and activity ontologies for activity modelling and recognition [31]. Ye *et al*. [32] defined an upper ontology for smart environments that has been used to create a formal activity model for activity recognition. Riboni *et al*. [33] presented the details of using OWL2[1], in particular, rule modeling, for both modeling and reasoning with complex activities. Knowledge-driven activity modeling solves the cold-start problem. It is semantically clear and elegant in reasoning as the approach is based on the solid foundation of formal knowledge representation theories. The major weaknesses are that the pre-defined models are static and incomplete due to the limited knowledge of individual experts.

Though activity modeling is important and existing approaches suffer from various problems, research on this topic has in general received little attention. The main reason is that activity modeling has been predominantly carried out in the domain of pattern recognition and data mining [34]. In these research contexts, the emphasis is usually placed on recognition and classification algorithms and their performance. Activity modeling is often viewed as a supportive component for pattern recognition and model-based application-level functions. When a knowledge-driven approach is used for activity modeling in knowledge-based systems the emphasis is normally placed on inference and decision-support mechanisms. It is usually assumed that activity modeling using knowledge engineering techniques will create complete accurate models [31] [33]. Nevertheless, in reality it is very difficult, if not impossible, to manually create models covering all permutations of different users, activities and performance styles. This is actually a widely known drawback of knowledge engineering based approach to knowledge modeling. This paper places activity modeling as the focus of the investigation. It conceives and develops an ontology-based hybrid approach that is able to address all of the aforementioned main problems, namely cold-start, reusability and model incompleteness, in one systematic solution. The approach and associated learning methods presented in this paper have to date not been seen in related research communities and are therefore deemed to be novel.

III. The Hybrid Approach to Activity Modeling

Fig. 1 depicts the 3-phase process of the hybrid approach to activity modeling as introduced in this paper. In Phase I - Knowledge-driven Activity Modeling, ontological knowledge engineering techniques, are utilized to extract and create the

[1] OWL2, along with RDF, RDFS and SPARQL mentioned later are all W3C standards, which can be found at www.w3.org.

initial seed activity models based on domain heuristics and prior knowledge. In Phase II - Model-based Activity Recognition, the seed activity models are used as classifiers by activity-based application systems, e.g. an ambient assisted living system, to classify sensor data for the purposes of activity recognition. If an activity has been accurately modeled in the seed activity models the activity should be recognized. On the other hand, if an activity is not modeled or the model is not accurate the activity will not be recognized. Nevertheless, the outputs of Phase II provides valuable inputs for Phase III, Data-driven Activity Learning within which data mining based learning methods are used to learn new activities and a user's activity profile. The learning results from Phase III can then be used to expand or update the seed activity models created in Phase I. The 3-phase process can be iterated periodically, thus incrementally improving the completeness and accuracy of activity models. Among these three phases Phase I requires human intervention. This includes initial inputs of domain knowledge, manual specification of the seed ontological activity models and human validation and update of learnt activities at the end of a single iteration. Both Phase II and III are data-driven and completely automatic.

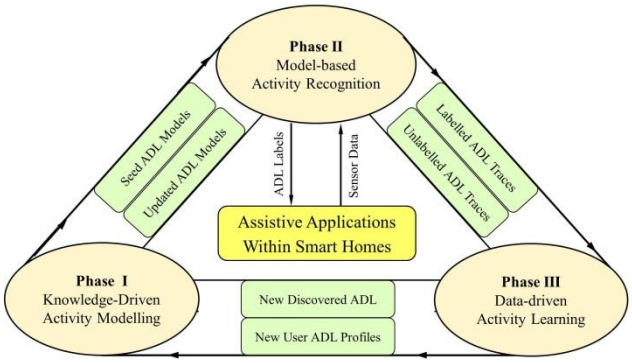


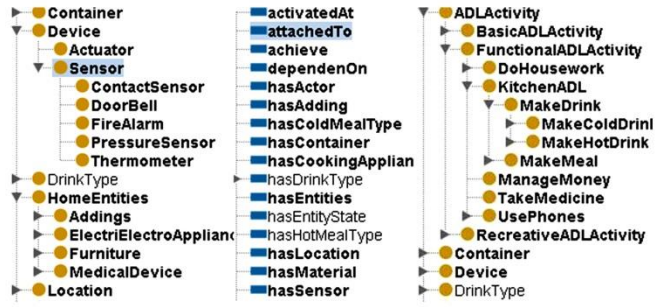
Fig. 1. The 3-phase hybrid approach to activity modeling.

In our previous studies we have developed ontological activity models [29], the mechanisms for dynamic sensor data segmentation [30][37] and ontology-based activity recognition [31] for Phase I and II. In this paper we concentrate on developing methods and algorithms for activity and user profile learning in Phase III. While details of Phase I and II can be found in the aforementioned work, to aid in the understanding of the discussion of the following Sections, we briefly outline the rationale and mechanisms of ontology-based activity modeling and recognition.

A. *Ontological Activity Modelling*

Ontological activity modeling relates to explicitly specifying activity models using the Description Logics (DL) formalism [42]. It defines an activity as an ontological concept and all actions that are required to perform the activity as the properties of the concept. In addition to action-based properties which are hereafter referred to action properties, an activity model also contains a number of descriptive properties to characterize the manner in which an activity is performed. For example, making tea involves taking a cup from the cupboard, putting a teabag into the cup, adding hot water to the cup, then milk and/or sugar. The ontological model of making tea, i.e. *MakeTea* concept, can be defined by action properties *hasContainer*, *hasTeabag*, *hasHotwater*, *hasMilk* and *hasFlavor* in conjunction with descriptive properties such as an activity identifier *actID*, start time *actStartTime*, duration *actDuration* and the sequential order of these objects in performing an activity *actObjSequence*. As action properties are mainly used for defining an activity, they play a crucial role in activity recognition. Descriptive properties, on the other hand, are not determinants in activity recognition. For example, making tea can happen at any time, it can be performed in different sequences and it may take variable amounts of time. Descriptive properties are mainly used to define user's activity profiles, namely to characterize the manner an activity is performed.

Activities can be modeled at different levels of abstraction. As such, ontological activity concepts are usually organized in a hierarchical structure to form super-class and sub-class relationships. For example, *MakeTea*, *MakeCoffee and MakeHotChocolate* activities can be modeled as the subclasses of *MakeHotDrink* activity, which is in turn the subclass of *MakeDrink*. Properties establish the relations between ontological activity concepts and the actions required for performing the activities. For example, the *hasContainer* property links the action of preparing a cup to the activity of making tea. Subclasses can inherit properties from superclasses. A leaf node of the hierarchy denotes a primitive activity that cannot be further classified. Fig. 2 presents an excerpt of activity ontologies and associated SH contextual concepts.



(a) Contextual classes (b) Properties (c) Activity classes

Fig. 2. An excerpt of the activity ontologies where (a) depicts the Sensor classes for describing sensor activations, (c) depicts the KitchenADL activity hierarchy of the ADL ontologies, and(b) illustrates the properties used by the context and ADL ontologies.

Ontological activity concepts define high-level generic activity models which are applicable to anyone. In addition to this, ontological activity modeling can also define the specific way that a person performs an activity, which is usually referred to as user activity profiles. For example, a user always makes English tea at 10am using skimmed milk and brown sugar. User activity profiles can be defined by creating an instance of a generic ontological activity concept in terms of the user's preference and habits. Ontological activity modeling

in Phase I can generate both generic activity models and user activity profiles, thus providing activity models at different levels of abstraction. Aside from activity concepts, other major entities from the domain will also be ontologically modeled. For example, a sensor concept and related properties are developed to establish the relationships between physical sensors, objects and their locations in addition to the sensor activation time. Further details of these concepts can be found in [29].

B. Ontology-based Activity Recognition

In dense sensing based activity monitoring [3] [34] an action of a user interacting with an object is detected through the sensor attached to the object. As such, the activation of a sensor implies that an action has been taken and subsequently an action property relating to the object will be assigned a value. Suppose that a number of sensors are activated along a time line and these sensor observations have been linked to corresponding action properties. At a specific time point the aggregation of these action properties will create a context denoting an ontological activity description. For example, the activation of the contact sensors on a cup and milk bottle can link the *cup* and *milk* to the activity being performed through *hasContainer* and *hasFlavor* properties. Assume that at a specific time, i.e. *hasTime*(*10am*), sensor observations *hasLocation*(*kitchen*), *hasContainer*(*cup*), *hasTeabag (English Teabag)* and *hasFlavor*(sand sugar) are generated, in aggregation this represents a context for an ongoing activity. If an activity concept in the ADL ontologies, e.g. *MakeTea*, has been defined by this set of action properties, then the activity can be deemed as the type of activity for the perceived context.

The rationale of inferring an activity from sensor observations described above can be formulated as follows: Given a set of action properties instantiated by sensor observations, identify the activity concept in the ADL ontologies that has the same set of action properties. Conceptually this problem of activity recognition can be mapped to the classification of the activity description using activity ontologies as the classifiers. Technically the problem can be solved using the subsumption reasoning in description logic, i.e. to decide if a concept description *C* created from sensor observations is subsumed by a concept description *D* within the activity models. Details of the theoretical foundation, reasoning algorithm and continuous recognition mechanisms for ontology-based activity recognition can be found in [31]. It is worth pointing out that the sensor data stream will be first partitioned into segments so that sensor activations within a segment can be aggregated to create an ontological activity description for activity recognition. We have developed a dynamic segmentation model based on the notion of varied time windows for real-time sensor data partition. The model can shrink and expand the window size of segmentation by using temporal information of activity models and sensor data. Further details about this can be found in [37].

To facilitate discussion we refer to the sequence of sensor observations within a segment as an action trace, i.e. the actions being undertaken within the segment. An action trace is equivalent to a set of action properties in an ontological activity description. With activity models from Phase I and streaming sensor data from applications within a smart home, activity recognition in Phase II can produce two types of action traces. If an action trace has a corresponding activity concept in the ADL models, this type of action trace is referred to as a **Labeled Action Trace** or *LAT* in short. Otherwise it is an **Unlabeled Action Trace** or *UAT*. *LATs* can be recognized from the set of action properties while *UATs* cannot be recognized as there are no corresponding activity models in the ADL ontologies or the models are not accurate.

The initial seed activity models generated in Phase I are inevitably incomplete due to the large number of ADLs, the different manner of users performing the ADLs and the changing user behaviors. As such, when an application within a smart home performs activity recognition over a period of time, it will generate large amount of *LATs* and *UATs* that contain information relating to un-modeled activities and the changing behaviors of a user. These action traces can be analyzed in Phase III to learn new activities and user activity profiles. New activities increase the completeness of activity models while user activity profiles improve the accuracy of activity models. Section IV and V describe the details of the learning mechanisms and methods for Phase III.

IV. Learning Unmodelled Activities

Activity learning aims to discover the activities that a user performs, however, which have not been modelled in the seed activity ontologies. As there are no models for these activities, they will not be recognised by the activity recognition process in Phase II. Subsequently, they are classified as unlabelled action traces, i.e. *UATs*. The essence of the activity learning is therefore to extract regular activities from *UATs* so that they can be modelled to improve activity models.

We have developed a 3-step learning method for this purpose. In the first step a semantic similarity metric is defined to measure the semantic similarity between two *UATs*. Based on this an algorithm is then developed to compute the semantic similarity. In the second step the semantic similarity between any individual *UAT* among all *UATs* are calculated. Based on the similarity metrics all *UATs* are classified into a number of subsets where each subset contains semantically similar *UATs*. In essence, each subset corresponds to one unmodelled activity and the number of *UATs* within each subset is the number of occurrences of the unmodelled activity during activity monitoring. Given that an unmodelled activity could be a one-off or random behaviour, it would be necessary to determine which discovered activities are regular activities and should be formally modelled. In the third step the frequency of the occurrence of these discovered unmodelled activities are calculated and a threshold is specified based on domain heuristics. If the occurrence frequency of an unmodelled activity is equal or greater than the threshold, then the activity will be formally modelled to update the ADL models.

As previously discussed, central to the activity learning method is the definition and computation of semantic similarity between *UATs*. We define $sim_{uat}(UAT_i, UAT_j)$ as the semantic similarity measure between two *UATs* and denote each *UAT* as a set of action property-value pairs represented as follows:

UAT_i={$prop_1$-$value_1$, $prop_2$-$value_2$ … $prop_k$-$value_k$}

UAT_j={$prop_1$-$value_1$, $prop_2$-$value_2$ … $prop_n$-$value_n$}

By semantic similarity we refer to the similarity of two *UATs* in terms of the types of the property values rather than the values themselves. This is because in ontological activity modeling an activity model is defined by the types of object rather than the objects themselves. For example, the *MakeTea* activity is specified by *hasContainer*(*x*), *hasTeabag(y)* ... and *hasFlavor(z)*. It is the types of the property values, rather than the specific *x*, *y* or *z* objects that define the activity. The value of a property, e.g. *x*, *y* or *z*, can be any object, e.g. cup_1 or cup_2 for *x*. *English tea* or *India tea* for *y*, *white sugar* or *brown sugar* for *z*. As such, the types of objects are the key discriminants to decide if two *UATs* refer to the same type of activity.

We have developed a method to compute the semantic similarity of two *UATs* in terms of the similarity of the two sets of property-value pairs in the two *UATs*. The method works as follows. We first map the set of action properties in a *UAT* to a corresponding set of objects and then derive the corresponding object type for each individual object. Both mappings from action properties to objects and from objects to object types are conducted by recursively unfolding the semantic relations based on ontological relationships modeled in the ADL ontologies. As a result of these mappings a *UAT* can be transformed into a description of a set of object types, as denoted in the formula below.

UAT_i={*objectType of* $prop_1$-$value_1$, *objectType of* $prop_2$-$value_2$ … *objectType of* $prop_k$-$value_k$}

Following this semantic explication and transformation, the similarity of two sets of property-value pairs is equal to the similarity of two sets of object types. As each object type is modeled as a concept in the ADL ontologies, the semantic similarity between two object types (concepts) can be computed based on the signatures of the object concepts. Specifically the similarity measure can be calculated using the Jaccard coefficient [35] which is the ratio of the number of shared elements from the intersection of the two sample sets to the number of total elements from the union of the two sets. This is represented as follows:

$$sim_{uat}(UAT_i, UAT_j) = (|UAT_{iot} \cap UAT_{jot}| / |UAT_{iot} \cup UAT_{jot}|$$

Here UAT_{iot} and UAT_{jot} refer to the set of object types in UAT_i and UAT_j, respectively.

Example 1 illustrates the transformation of two *UATs* and their semantic similarity. Even though the order and specific objects used for each activity is different, the semantic similarity measure equals one, indicating they refer to the same type of activities.

[Example 1:

UATs are in the form of property-value pairs.

UAT_i={*hasContainer*(mug_a), *hasTeabag*(English teabag), *hasFlavor*(brown sugar), *hasHotwater*($kettle_a$), *hasMilk*(semi-skimmed milk)}

UAT_j={*hasContainer*(mug_b), *hasTeabag*(India teabag), *hasHotwater* ($kettle_a$), *hasMilk*(skimmed milk), *hasFlavor*(white sugar)}

UATs are in the form of sets of objects.

UAT_i={mug_a, English teabag, brown sugar, $kettle_a$, semi-skimmed milk}

UAT_j ={mug_b, India teabag, $kettle_a$, skimmed milk, white sugar}

UATs are in the form of sets of object types.

UAT_i ={*Container*, *Tea*, *Sugar*, *Kettle*, *Milk*}

UAT_j ={*Container*, *Tea*, *Kettle*, *Milk*, *Sugar*}

$sim_{uat}(UAT_i, UAT_j) = 1$]

We can compute the semantic similarity of any two *UATs* using the described method and then use the resulting similarity metrics to classify all *UATs* into a number of subsets of *UATs*. For each subset, the semantic similarity sim_{uat} between any two *UATs* is equal to 1, hence each subset denotes a specific type of activity.

Once distinct activities are discovered through semantic classification, it is necessary to decide whether they are regular activities, random or one-off activities. To this end, we use the daily *frequency of occurrence* of a *UAT* as the significance measure for the activity it represents. For example, if the daily *frequency of occurrence* of UAT_k is *n*, this means the activity UAT_k occurs on average *n* times a day during the period of monitoring, e.g. once a day for *n*=1, twice a day for *n*=2 and once every two days for *n*=0.5. A threshold value can then be specified for the daily *frequency of occurrence* based on domain knowledge and heuristics. For example, given that most ADLs are performed on a daily basis, we can reasonably set 0.5 as the threshold value, namely a *UAT* happening once every two days can be regarded as a regular activity. If the daily *frequency of occurrence* of a *UAT* is greater or equal to the threshold value, the *UAT* can be formally designated as a regular activity. Subsequently, this activity will be modelled to update the activity models. Table 1 summarises the variables, their descriptions and the pseudo code of the algorithm for the presented activity learning method.

TABLE 1. THE ALGORITHM FOR LEARNING UNMODELLED ACTIVITIES

Variables	Descriptions
SU	*the whole set of UATs*
SSU_i	*the* i^{th} *subset of UATs within which all UATs are semantically similar*
fo_{uat}	*the daily frequency of occurrence of an UAT*
T_{fo}	*the threshold value specified for* fo_{uat}
D	*the duration of activity monitoring in days*

1. **set** *SU*, *D*, T_{fo} from Phase II outputs
2. for any UAT_i, $UAT_i \in SU$, **do**
3. semantic unfolding and transformation as illustrated in Example 1
4. **enddo**
5. **set** a counter *actNum* = 0, representing the number of new activities
6. **while** ($|SU| > 0$)
7. **set** UAT_{base} to an arbitrary member of *SU*
8. create a new subset SSU_{actNum} with UAT_{base} as the only member
9. **for** ($1 \leq i \leq |SU|$)
10. calculate $sim_{uat}(UAT_{base}, UAT_i)$, where $UAT_i \in SU$
11. **if** ($sim_{uat}(UAT_{base}, UAT_i) = 1$)
12. put UAT_i into the set SSU_{actNum}
13. remove UAT_i from *SU*
14. **else**
15. leave UAT_i in *SU*

```
16.       endif
17.    endfor
18.    Increase the counter actNum = actNum + 1
19. endwhile  // this will create actNum subsets SSU_i
20. for ( 1 ≤ i ≤ actNum )
21.    calculate fo_uat(UAT, UAT ∈ SSU_i) = |SSU_i | / D
22.    if ( fo_uat(UAT) ≥ T_fo ) recommend to an expert
23.       SSU_i represents a regular activity
24.    else
25.       SSU_i represents a random / one-off activity
26.    endif
27. Endfor
```

V. Learning User Activity Profiles

An activity can be performed in many different ways, e.g. using different items of the same object types, in different sequence of actions, at different times and within variable durations. A user activity profile is referred to the specific way of a user performing activities which is the key to personalised assistance in assistive living. To formally specify a user activity profile we use three attributes, namely an object pattern, duration and an activity pattern, to characterize the manner that an activity is performed. An object pattern refers to the unique order of objects that an activity is performed whilst an activity pattern describes the frequency and regularity of an activity occurrence, including the starting time(s).

Ontological activity modelling can model an activity profile as an instance of the corresponding generic activity concept. Nevertheless, the initial seed activity models do not contain user profile models. This is because the model of a user activity profile is user specific, it can only be defined once a user is identified. In addition, a user's behaviour can change due to physical or mental conditions, thus leading to the change of activity profiles. As such, learning user behaviours from their activity observations is an effective way to create user profiles.

An *LAT* represents an activity that has been modelled in the ADL ontologies and recognised in Phase II. Each *LAT* has a corresponding activity label and a sequence of sensor observations denoting the specific undertaking of the activity. Over time for each activity there will be a set of accumulated *LATs*, which provide a valuable source for user profile discovery. In the following Sections we describe the processes and methods of learning user profiles from real time observations of activity performance, i.e. the *LATs* generated in Phase II.

A. Learning object patterns

We have developed a 3-step learning method to discover whether or not a user follows a unique object pattern in performing an activity. In the first step, we define a similarity measure $sim_{lat}(LAT_i, LAT_j)$ in terms of object sequences and develop an algorithm to calculate the similarity of two *LATs*. In the second step we compute the similarity among all *LATs* of a specific activity and based on the similarity measures a classification algorithm is developed to classify the set of *LATs* into subsets of *LATs* of the same object pattern. In the third step we calculate the distribution of frequency of occurrences of all object patterns for the specific activity. The dominant object pattern can then be used to characterize the user activity profile for the specific activity.

Similar to a *UAT*, an *LAT* can be denoted as a set of action property-value pairs, i.e. LAT_i={$prop_1$-$value_1$, $prop_2$-$value_2$ … $prop_k$-$value_k$}. We define $sim_{lat}(LAT_i, LAT_j)$ as the similarity measure in terms of object sequences of the two *LATs*. To calculate the similarity measure we transform an *LAT* from a sequence of action property-value pairs to a sequence of objects through semantic unfolding of ontological concepts. The resulting *LAT* can be represented as a sequence of objects, i.e. *LAT*={$object_1$ *of* $prop_1$-$value_1$, $object_2$ *of* $prop_2$-$value_2$ … $object_k$ *of* $prop_k$-$value_k$} where each element $object_i$ is a specific object denoted by its signature. After this transformation, an *LAT* can be treated as an object signature vector, and the similarity of two *LATs* is essentially the similarity between two vectors in a high dimensional space. This can be computed using the generic cosine similarity algorithm [36], as formulated in the equation below.

$$sim_{lat}(LAT_i, LAT_j) = (LAT_i \,.\, LAT_j) / (\|LAT_i\| \, \|LAT_j\|)$$

$$= \sum_{k=0}^{n} (LAT_{ik} \times LAT_{jk}) \,/\, \left(\sqrt{\sum_{k=0}^{n} (LAT_{ik})^2} \times \sqrt{\sum_{k=0}^{n} (LAT_{jk})^2} \right)$$

The numerator is the dot product of the two *LAT* vectors and the denominator is the product of the magnitudes of the two vectors. *i* and *j* are an *LAT* respectively, $i \neq j$, and *n* is the total number of *LATs*. A value in the range [-1, 1] can be generated, where -1 signifies the exact opposite object pattern and 1 signifies exactly the same pattern.

In order to make use of the cosine similarity algorithm to compute similarity of *LATs* we convert the text notation of the elements of an *LAT* to numerical values by allocating each object an object identifier number. The object identifier numbers do not have any meaning, they are simply used to facilitate the similarity computation based on object sequences. Example 2 below illustrates three *LATs*, their object signatures, corresponding exemplar object identifier numbers and the similarity measures between them.

[Example 2:
LAT_1={mug_a(1), *teabag*(2), *hotwater*(3), *sand sugar*(4), *skimmed milk*(5)}
LAT_2={mug_b(9), *teabag*(2), *whole milk(8), hotwater*(3), *sand sugar*(4)}
LAT_3={mug_a(1), *teabag*(2), *hotwater*(3), *sand sugar*(4), *skimmed milk*(5)}
$sim_{lat}(LAT_1, LAT_2) = 0.7053$
$sim_{lat}(LAT_1, LAT_3) = 1$]

As shown in the above example, LAT_1 and LAT_3 will be classified into the same subset because they follow the same object sequences. Similarly we can compute the similarity measures for all *LATs* and classify the *LATs* that their similarity measures are equal to 1 into a subset. Each subset represents a unique object pattern.

To determine if there is a dominant object pattern for performing a specific activity, we calculate the probability of the occurrence of a unique object pattern for all object patterns

within the set of *LATs* for the activity. We then specify a threshold value for the probability of occurrence so that when the occurrence probability of a specific object pattern is greater than or equal to the threshold value, the corresponding subset can be viewed as the dominant object pattern. For example, suppose that there are five object patterns for performing an activity and the occurrence probability of the third object pattern is 0.83. This means that the activity is performed 83% of the time using the 3rd object pattern and only 17% using the other patterns. In this case, the 3rd object pattern can be reasonably regarded as the user profile for this specific activity. On the other hand, if all probability values are roughly evenly distributed and each value is very small, it can be assumed that the activity is performed in a random manner and there is not a specific preferred way of performing the activity. In our study we define 2/3 as the threshold value of the occurrence frequency. Table 2 summarizes the variables, their descriptions and the pseudo code of the algorithm for this object pattern learning method.

TABLE 2. THE ALGORITHM FOR LEARNING OBJECT PATTERNS

Variables	Descriptions
$SL(z)$	*The set of all LATs for the specific activity z*
pop_k	*The probability of occurrence of the object pattern k*
T_{pop}	*The threshold of pop = 2/3*

// discover unique object patterns
1. **set** $SL(z)$ and T_{pop} from Phase II outputs
2. for any LAT_i, $LAT_i \in SL(z)$, **do**
3. semantic unfolding as illustrated in step 2 in Example 1
4. **enddo**
5. **set** a counter *uopNum* = 0, which represents the number of the unique object patterns in $SL(z)$
6. **while** ($|SL(z)| > 0$)
7. **set** LAT_{base} to an arbitrary member of $SL(z)$
8. create a new subset $SSL(z)_{uopNum}$ with LAT_{base} as the only member
9. **for** ($1 \le i \le |SL(z)|$)
10. calculate $sim_{lat}(LAT_{base}, LAT_i)$, where $LAT_i \in SL(z)$
11. **if** ($sim_{lat}(LAT_{base}, LAT_i) = 1$)
12. put LAT_i into the subset $SSL(z)_{uopNum}$
13. remove LAT_i from $SL(z)$
14. **else**
15. leave LAT_i in $SL(z)$
16. **endif**
17. **endfor**
18. Increase the counter *uopNum* = *uopNum* + 1
19. **endwhile** // this will create *uopNum* subsets $SSL(z)$
20. **for** ($1 \le i \le uopNum$)
21. calculate $pop_i = |SSL(z)_i| / |SL(z)|$
22. **if** (pop_i) $\ge T_{pop}$)
23. $SSL(z)_i$ represents a dominant object pattern
24. **else**
25. No user profile for this activity
26. **endif**
27. **Endfor**

B. Learning an activity duration

Duration information of an activity model is useful in continuous activity recognition. It helps define the sliding time window for dynamic sensor data segmentation [37]. It is also a key indicator of a user's behavioural changes, which provide personalised assistance, e.g. specifying the waiting time for a reminder.

We calculate duration information using all *LATs* of an activity based on the time points at which the first and last sensor activations of the LAT_i are received. Table 3 displays the algorithm for calculating the minimum, maximum and average duration of a user performing an activity. The algorithm is a continuum of the object pattern learning algorithm in Table 2.

TABLE 3. THE ALGORITHM FOR LEARNING ACTIVITY DURATION

Variables	Descriptions
t_s, t_e	*the first and last sensor activation times*
Du_{min}, Du_{max}, Du_{ave}	*the minimum, maximum and average duration*

// discover the duration information
28. **Set** Du_{min}=initial value, Du_{max},*and* $Du = 0$
29. **for** ($1 \le i \le |SL(z)|$) // for all *LATs* of an activity
30. **if** ($Du_{min} > (t_{ei} - t_{si})$) $Du_{min} = (t_{ei} - t_{si})$
31. **if** ($Du_{max} < (t_{ei} - t_{si})$) $Du_{max} = (t_{ei} - t_{si})$
32. $Du = Du + (t_{ei} - t_{si})$
33. **endfor**
34. $Du_{ave} = Du / |SL(z)|$

C. Learning activity patterns

An activity pattern is crucial for providing proactive personalized activity assistance. For example, if an assistive system knows that a user takes medicine twice a day at 10am and 5pm respectively, then it can prompt the user to take medicine at these times. Nevertheless, it is difficult to decide an activity pattern and starting time as most ADLs could be carried out randomly dependent on personal preferences. Even with some kind of regularity, ADLs are most likely performed within a time period rather than at an exact time point.

We have developed a 2-stage approach to discover an activity pattern and starting time from *LATs*. In the first stage we calculate the daily frequency of occurrence of an activity, namely the average number of activity occurrences in a day during the period of monitoring. The daily frequency of occurrence is used as a criterion to decide if the activity is carried out on a regular basis. It can be determined based on domain knowledge during the initial *LAT* modelling. For example it could be 1/7, implying that it covers all weekly activities. A regular activity does not necessarily support an activity pattern. For example, a user makes tea twice a day, every day, however, the activity is always carried out at different times. This is a regular activity but does not have a pattern.

In the second stage we decide if a regular activity follows an activity pattern. To this end we firstly partition the 24 hours of a day into a number of fixed-length time slots. For example, if the duration of a time slot is 30 minutes, then a day can be partitioned into 48 time slots. Secondly, we map the starting time of all *LATs* of an activity into the corresponding time slots. Thirdly, we calculate the probabilities of the occurrence of the activity within each time slot against the total occurrence of the activity. Based on the probability distribution of occurrence and the threshold values of the occurrence probabilities, we can infer whether or not there is an activity pattern.

Table 4 displays the algorithm of learning activity patterns, which is a continuum of the algorithms in Tables 2 and 3. Three inference rules for learning activity patterns have been defined below, which are explained using the example depicted in Fig. 3.

TABLE 4. THE ALGORITHM FOR LEARNING ACTIVITY PATTERNS

Variables	Descriptions
fo_{lat}	*the daily frequency of occurrence of an activity*
Stime	*the starting time(s) of an activity*
prob	*the probability of an activity occurrence in a time slot*
$prob_{threshold}$	*the threshold values for prob*
t_{slot}	*the fixed-length duration of a time slot in minutes*
D	*the duration of activity monitoring in days*

// discover activity patterns and starting time(s)
35. calculate $fo_{lat}(LAT, LAT \in SL(z)) = |SL(z)| / D$
36. partition a day into time slots based on t_{slot}
37. map the t_s of all *LATs* in *SL(z)* into corresponding time slots
38. **for** ($1 \leq i \leq 24x60/ t_{slot}$) // for all time slots
39. $prob_i$ = (number of occurrence in the i^{th} time slot) / $|SL(z)|$
40. **endfor**
41. // apply the pattern learning rules
42. **if** ($fo_{lat} \leq 1$ *and prob* at a time slot $p \geq prob_{threshold}$)
43. *LAT* is a regular activity with an activity pattern
44. *Stime* $=(\sum_{i=0}^{K} t_{si}) / K$, *K* = number of occurrence in time slot *p*
45. **else** ($fo_{lat} \leq 1$ *and prob* at any time slot $p \leq prob_{threshold}$)
46. *LAT is a random activity, no need to calculate Stime*
47. **endif**
48. **if** ($fo_{lat} = n > 1$ and each *prob* at *n* time slots $\geq prob_{threshold} \times (1/n)$)
49. *LAT* is a regular activity with an activity pattern
50. *Stime*(at the n^{th} occurrence) $=(\sum_{i=0}^{K} t_{si}) / K$, *K* = number of occurrence in time slot p_i, $i=1, 2...n$.
51. **else** ($fo_{lat} = n > 1$ and all *prob* at *n* time slots $\leq prob_{threshold} \times (1/n)$)
52. *LAT is a random activity, no need to calculate Stime*
53. **Endif**

Rule 1: If an activity is a regular activity based on the daily frequency of the activity $fo_{lat}(LAT)$; and $fo_{lat}(LAT)$ is $n \leq 1$; and the occurrence probability of the activity in the p^{th} time slot is equal or greater than $Prob_{threshold}$; then the activity has a pattern - it happens once $1/n$ day(s) in the p^{th} time slot. The starting time *Stime* for the activity pattern can be estimated as the average time of the first sensor activation of all *LATs* within the p^{th} time slot. The *bath* activity in Fig. 3 illustrates this case. For example, if $fo_{lat}(bath) = 0.5$, $Prob_{threshold} = 70\%$, as $Prob(bath) = 80\% > 70\%$, then it can be inferred that the *bath* activity happens once every two days in the time slot starting from 7pm.

Rule 2: If an activity is a regular activity; and $fo_{lat}(LAT)$ is $n > 1$; and the occurrence probability for each time slot is greater than $Prob_{threshold} \times (1/n)$, i.e. the aggregated occurrence probability in the *n* time slots is greater than $Prob_{threshold}$; then the activity has a pattern - it is performed *n* times a day within the *n* time slots. The starting time *Stime* of the n^{th} occurrences can be estimated as the average time of the first sensor activation of all *LATs* within the n^{th} time slot. The *tea* activity in Fig. 3 illustrated this case, i.e. it happens three times a day in three time slots with the occurrence probability of each timeslot being greater than 23.3%.

Rule 3: If an activity is a regular activity and the occurrence of an activity is dispersed evenly among a number of time slots *k* where *k* is significantly greater than $fo_{lat}(LAT)$; and the occurrence probability in each time slot is significantly less than $Prob_{threshold}$; then the activity is a random activity during a day. As such, it makes no sense to infer the starting time of the activity. The phone call activity in Fig. 3 illustrates the nature of a random activity.

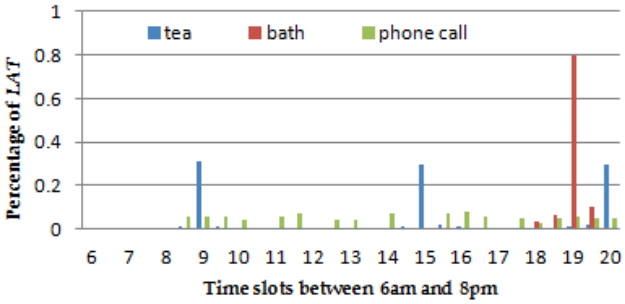


Fig. 3. Making tea, having a bath and making phone call activities, and their probability distribution of occurrence over a period of time.

D. Activity model evolution

Once a new activity is discovered as described in Section IV, it is necessary to decide the location of the activity in the hierarchy of the activity ontologies and also an appropriate label that should be assigned to the activity. The label should be meaningful and compliant with other activities' labelling rationale and also the ontological modeling conventions so that it can be easily referred to and understood later. The location of a newly discovered activity in the ontological activity hierarchy can be recommended through the subsumption reasoning of the *UAT* description. Nevertheless, human intervention is required to validate and finalize the position and label of an activity model in order to maintain the quality of the model. As such, the classification and naming process have been carried out manually using the standard practice of ontological engineering, i.e. a knowledge engineer encodes the new activities and edits the ontologies using an ontology editor, e.g. [38].

Similarly, once a user's behavioral features, i.e. activity profiles, are learnt as described in the previous subsections, the activity models should be evolved to reflect the unique manner a user performs activities, e.g. for the purpose of personalized assistance. Given that a user's activity profile is equivalent to an instance of a generic activity model, i.e. an ontological activity class, and for any *LAT* there is a corresponding ontological activity class, activity profile evolution amounts to creating a new instance or updating an existing instance. This can be undertaken automatically by using the standard APIs of the underlying semantic frameworks.

VI. IMPLEMENTATION AND EVALUATION

In this Section we initially outline the results of ontological activity modeling, system implementation and deployment. We then describe in detail the experimental design, data collection and evaluation for activity and profile learning. Based on the evaluation results, we discuss generic issues related to the presented approach.

A. *Modeling, Implementation and Deployment*

To test and evaluate the presented approach we have created the seed activity ontologies in Phase I using the Protégé ontology editor [38] (Fig. 2), through knowledge engineering practice [29]. We have implemented a feature-rich system for activity recognition and model learning in Phase II as presented in Fig. 4. The system was developed using C#, ASP.NET, Ajax and Silverlight for audio and graphical user experience and deployed within our smart Lab [2]. The creation, management and query of semantic data was handled using the SemWeb semantic technologies for C# [39] and SPARQL query language. Semantic reasoning was implemented using the Euler [40] and Pellet [41] inference engines.

When an actor interacts with objects in sequence in real time, sensor activations are continuously fed into the system. Sensor data series are dynamically segmented [37] and recognition operations are repeatedly performed to carry out continuous, progressive activity recognition [31]. As depicted in Fig. 4, the system can dynamically display the activated sensor sequence, the incrementally recognized activities and the system status in real time.

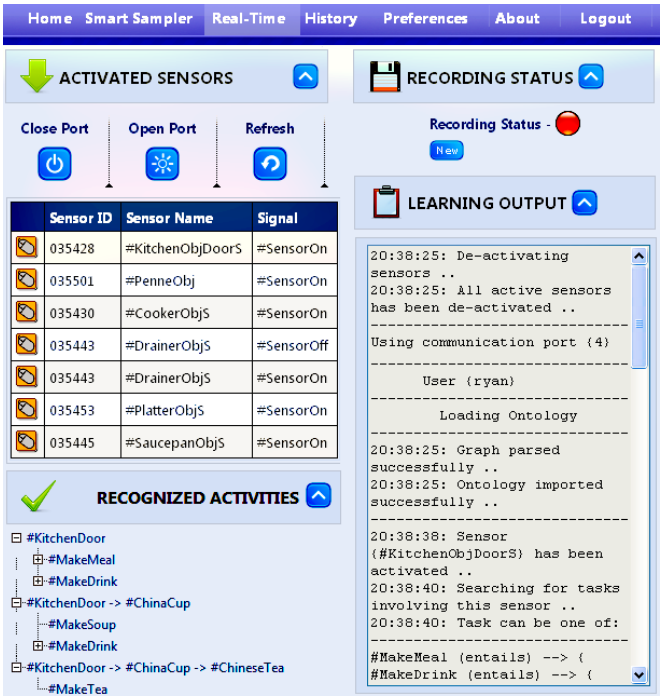


Fig. 4. The system interface operating in real time mode. In the left-hand side, the top panel is used for communication port setup; the middle panel displays the sequence of activated objects; and the bottom panel presents progressively recognized activities in a tree-like hierarchy. In the right-hand side, the top panel contains function buttons for data recording and playback; the bottom panel presents a temporal trace of events during the system operation. The system can import activity ontologies, specify reasoning and learning parameters, select the modality of audio reminders, configure hardware and define event priorities and user activity profiles.

B. *Experiment Design and Data Collection*

To systematically test and evaluate activity and profile learning in Phase III, eight typical ADLs as presented in Table 6, were selected for the purposes of experimentation. For each activity, the required objects for performing the activity were identified and for each of them a contact sensor was attached. Each activity was designed to be performed in three different ways, leading to three different types of activity specification as illustrated in Table 5. The Type 1 activity specification, namely TP1 in short, can be viewed as the "standard" way of performing a specific activity. The Type 2 activity specification has the same set of objects; however, they are interacted with in a different order. The Type 3 activity specification has a different set of objects as it is intended to simulate noise on the sensor data, i.e. a faulty sensor by omitting a user-object interaction or a false sensor reading by adding an irrelevant object interaction. In addition, in order to test the activity learning capability we deliberately remove activity models, *MakeChocolate* and *BrushTeeth*, two of the eight selected activities from the seed activity ontologies.

TABLE 5. TWO EXAMPLES OF ACTIVITY SPECIFICATIONS

Activity		Activity Specification (sequence of user-object interactions)
makeTea	TP1	GetCup, GetTea, PourWater, GetMilk, GetSugar
	TP2	GetCup, PourWater, GetMilk, GetTea, GetSugar
	TP3	GetCup*, GetTea, PourWater, GetMilk, GetSugar
BrushTeeth	TP1	RunSink, GetToothbrush, GetToothpaste, GetMouthwash
	TP2	GetToothbrush, GetToothpaste, RunSink, GetMouthwash
	TP3	RunSink, GetToothbrush, getSoap**, GetToothpaste, GetMouthwash

** faulty sensors that do not fire; ** false or extra sensor reading; TP-Type.*

Three actors took part in the experiments. Each of the participants interacted with the objects of each activity of the eight activities in accordance with the activity specifications for two rounds. This produced a total of 3 (types) x 8 (activities) x 2 (rounds) x 3 (actors) = 144 action traces. Following activity recognition in Phase II the system produced 100 *LATs* and 44 *UATs* as presented in Table 6.

C. *Analysis and Evaluation*

Our evaluation has focused on the performance of learning distinct activities from *UATs* and the performance of discovering the dominant object pattern from *LATs* in activity profile learning. This is due to the fact that semantic based similarity calculation and classification are the central underpinning mechanisms for the presented methods. In addition, evaluation of time-related metrics, e.g. duration or activity patterns will only make sense if the data are generated by real users performing real ADLs over a relatively long period of time. This has been proven to be difficult due to technical, privacy and ethical issues. Furthermore temporal information in these learning methods is mainly used for numerical calculation, i.e. the duration, starting time and frequencies, which has already been clearly illustrated in previous Sections.

1) Results and analysis on learning new activities

We apply the activity learning algorithm in Table 1 to the *UAT* dataset in Table 6 to learn new activities. Table 7 displays the activity learning results. The "Ground Truth" column presents what actually happened in the experiment whereas the "*UAT Subset"* column lists the classified subsets of the 44 *UATs.* Among six of the modeled activities three of them, i.e. *WashHands*, *WatchTV* and *HaveBath*, have been fully recognized without generating any *UATs*, so they are not listed in Table 7. The other three modeled activities, i.e. *MakeTea*, *MakePasta* and *MakeCoffee*, have generated four, two and two *UATs* respectively. This is because we randomly introduce sensor noise into the Type 3 activity specification, the activity traces from TP3 may be recognized or not depending on the nature of the noise, thus leading to *UATs*.

TABLE 6. RECOGNITION RESULTS OF THE 144 ACTIVITIES

Activities		Actor1		Actor2		Actor3		Sum L/U
Exp		1	2	1	2	1	2	
Make Tea	TP1	L	L	L	L	L	L	6/0
	TP2	L	L	L	L	U	L	5/1
	TP3	L	L	L	U	U	U	3/3
Brush Teeth	TP1	U	U	U	U	U	U	0/6
	TP2	U	U	U	U	U	U	0/6
	TP3	U	U	U	U	U	U	0/6
Make Coffee	TP1	L	L	L	L	L	L	6/0
	TP2	L	L	U	L	U	L	4/2
	TP3	L	L	L	L	L	L	6/0
Have Bath	TP1	L	L	L	L	L	L	6/0
	TP2	L	L	L	L	L	L	6/0
	TP3	L	L	L	L	L	L	6/0
Watch TV	TP1	L	L	L	L	L	L	6/0
	TP2	L	L	L	L	L	L	6/0
	TP3	L	L	L	L	L	L	6/0
MakeChoc olate	TP1	U	U	U	U	U	U	0/6
	TP2	U	U	U	U	U	U	0/6
	TP3	U	U	U	U	U	U	0/6
Make Pasta	TP1	L	L	L	L	L	L	6/0
	TP2	L	L	L	L	U	L	5/1
	TP3	L	L	U	L	L	L	5/1
Wash Hands	TP1	L	L	L	L	L	L	6/0
	TP2	L	L	L	L	L	L	6/0
	TP3	L	L	L	L	L	L	6/0
Sum L/U	All TPs	18/ 6	18/ 6	16/ 8	17/ 7	14 /10	17/ 7	100 /44

Here TP - the type of activity, Exp1 and Exp2 - the two rounds of experiments respectively, L and U - an LAT and UAT respectively, and Sum - the number of L and U for a particular type of activity and a particular actor respectively.

For the two unmodeled activities, *MakeChocolate* and *BrushTeeth*, each consists of 18 *UATs* which are classified into 7 subsets. One subset has 12 *UATs* and the other six subsets each have one *UAT*. This is because both Type 1 and Type 2 activity specifications use the same set of objects, thus leading to 12 *UATs* in one subset. The Type 3 activity specification simulates random sensor noise by introducing an irrelevant object into the activity, thus leading to 6 different action traces. The comparison between the *UAT* classification results and the ground truth proved that the semantic similarity based *UAT* classification is 100% accurate in terms of similarity criteria $sim_{uat}(UAT_i, UAT_j) = 1$. In the case that the duration of observation is available, it is straightforward to follow the activity learning algorithm to identify the distinct regular activities.

TABLE 7. THE ACTIVITY DISCOVERY RESULTS FROM *UATS*

	Ground Truth		***UAT* Subsets SSU_i**
Activity Label	***UAT***	***LAT***	**Total 21 subsets, SSU_1 - SSU_{21}**
MakeChocolate	18	0	12 in SSU_1, 1 in each $SSU_{6\text{-}11}$
MakeTea	4	14	1 in each $SSU_{18\text{-}21}$
MakeCoffee	2	16	2 in SSU_3
BrushTeeth	18	0	12 in SSU_2, 1 in each $SSU_{12\text{-}17}$
MakePasta	2	16	1 in each $SSU_{4\text{-}5}$

Here SSU_i - the i^{th} subset of UATs as defined in Table 1.

2) Results and analysis on learning object patterns

We apply the algorithm in Table 2 to all *LATs* in Table 6 to learn object patterns. Table 8 presents the analysis results for three of the six modeled activities. From left to right the first and second columns contain the activities and the total number of *LATs* in the corresponding activity. The third column displays the unique object patterns among all *LATs* of the activity while the fourth one shows the number of *LATs* for each unique object pattern. The fifth column presents the probabilities of occurrence of a unique object pattern. As can be viewed from the results, each activity has two major activity patterns with a similar percentage of occurrences. In addition, a number of patterns are also identified for each activity with each pattern having only one *LAT*. The learning results are in line with the ground truth of the experiment. The two major activity patterns correspond to the Type 1 and Type 2 activity specifications. The occurrence of a number of one-*LAT* patterns in each activity corresponds to the Type 3 activity that is performed randomly by introducing random noise, thus no sequence of objects are identical. The matching of the analysis results with the ground truth of the experiment proves that the method for learning object patterns is effective.

TABLE 8. PART OF THE ACTIVITY LEARNING RESULTS FROM *LATS*

Activities	***LAT* No.**	**Unique Object Patterns (*UOP*)**	***LAT* No. for each *UOP***	**pop_x (%) for each *UOP***
MakeTea	14	UOP_1	6	42.86
		UOP_2	5	35.71
		UOP_3 - UOP_5	1	7.14 each
MakePasta	16	UOP_1	6	37.5
		UOP_2	5	31.25
		UOP_3 - UOP_7	1	6.25 each
WashHands	18	UOP_1	6	33.33
		UOP_2	6	33.33
		UOP_3 - UOP_9	1	5.55 each

There are a number of object patterns for each activity in Table 8. This is because the activity specifications are deliberately designed to contain two major object patterns, i.e.

Type 1 and Type 2, and a number of random patterns in Type 3, to test and evaluate various aspects of activity and profile learning methods. In a real situation a user may have one dominant object pattern or simply perform in a random way. Nevertheless, the experiments and analysis results demonstrate the learning method and process. For example, if we set the threshold of the probability of occurrence of the object pattern to 36%, then the unique object pattern for both *MakeTea* and *MakePasta* will be identified as the dominant object patterns. For the *WashHands* activity there is no object pattern.

3) General Discussions

Sensor noise such as faulty sensors, communication and processing errors is inevitable in real use scenarios. In our experiments we simulate sensor noise in Type 3 activity specifications, leading to six occurrences of sensor noise for each activity among its eighteen activity occurrences, equivalent to 33.33% data accuracy. As can be seen from the results in Table 6 sensor noise does not have to affect activity recognition, i.e. generating a *UAT*. It will be up to the nature of sensor noise that determines whether or not an action trace with sensor noise could be recognised. The impact of sensor noise on recognition accuracy has been discussed in [31].

Sensor noise affects activity and profile learning. The analysis results in Table 7 show that the two unmodeled activities, *MakeChocolate* and *BrushTeeth* each have 18 *UATs* but only 12 of them are classified into one set due to sensor noise, equivalent to a 66.67% classification rate, which resulted from our simulation of sensor noise for exactly one third of activities in the experiments. Nevertheless, the extent to which the noise affects the classification rate is dependent on the similarity threshold which is used to decide whether or not two traces are deemed as similar. For example, our study only classifies absolutely similar traces, i.e. $sim_{uat}(UAT_i, UAT_j)=1$, into a set. If we reduce the similarity threshold, e.g. to 0.8, then any traces with $sim_{uat}(UAT_i, UAT_j)\geq 0.8$ will be classified to the same set. In this case the classification rate (66.67%) and the noise level (33.33%) will both be changed. This actually means that two activity traces with one of them having sensor noise such as a missing sensor observation or an incorrect object can still be classified to a set if the other objects are the same. Understandably, the lower the level of the similarity threshold, e.g. 0.65, the higher level of sensor noise which can be accommodated for. From this perspective, our approach to activity learning is resilient to a certain level of sensor noise.

Given that the threshold determines how much sensor noise can be assimilated by our learning approach, further investigation is required to decide upon an appropriate similarity threshold. Nevertheless, the current study has shown that our approach itself is conceptually and theoretically correct without specific limitations. The impact of sensor noise on profile learning as depicted in Table 8 can be discussed and explained in the same context as above. We shall not elaborate here due to limited space.

Computational performance: In the 3-phase iterative process of the hybrid approach to activity model learning, real-time continuous activity recognition requires high computational performance to ensure dynamic on-the-fly situation generation and reasoning against the activity models. The experiments and evaluation in [31] have shown the computational performance for real-time activity recognition is satisfactory. Given that activity and profile learning are intended to be performed periodically offline and most computation in these learning algorithms involve linear time complexity with regard to dataset volume, we believe that the technical correctness of these learning algorithms is more important than their computational performance. As such, our experiments and evaluation have focused on technical assessment.

Knowledge-driven versus data-driven: The presented hybrid approach combines knowledge-driven manual model specification with data-driven automatic model learning. One question arising from the study is to what extent models should be manually specified in advance. Should we specify as many models as possible with few to be learnt or the reverse? Relying on manual specification too much will have the disadvantages of the knowledge-driven approach. On the other hand, relying on automatic model learning too much will have the drawbacks of data-driven approaches. While the approach allows flexible specification of the initial seed activity models, it is an interesting research question to consider how to achieve the optimal balance between the two approaches to activity modeling.

Another question is which activity models should be specified and which should be learnt. As described in previous sections only high-level generic coarse-grained activity models, i.e. ontological activity classes are reusable to multiple users, low-level fine-grained activity models, i.e. instances of ontological activity classes, are user-specific. Though both models can be created by manual specification and/or learning, experiences and initial findings from our current studies suggest that we should specify as many generic coarse-grained activity models as possible as the models at this level of abstraction are generic and applicable to all users, thus insensitive to low-level special behavior of individual users. On the other hand, we should learn as many fine-grained activity models as possible as the models at this level of abstraction reflect the uniqueness and dynamics of an individual user's behavior. Data-driven activity learning plays a more important role in improving activity model accuracy and addressing the changing nature of activity models.

It is worth noting that the knowledge-driven approach essentially avoids the "cold-start" problem by using domain knowledge to create models through knowledge engineering techniques. As such, the approach could suffer from the generic problems of knowledge engineering such as incomplete and inaccurate knowledge. Nevertheless, such issues have been well addressed in the knowledge engineering research community and are beyond the scope of this paper.

VII. Conclusion

This paper introduced a hybrid approach to creating complete, accurate activity models through incremental activity discovery and profile learning. We have described a 3-phase iterative process and discussed the methodology of each phase of the lifecycle. While previous work [29] [31] [37] reported the details of ontological activity modeling and recognition, this paper has presented the details of activity and profile learning methods by which activity models can be expanded, personalized and adapted. The compelling feature of the approach is that it combines the strengths of traditional data mining based activity modeling with that of ontology based explicit activity modeling, making our approach flexible, applicable and scalable in terms of reusability, rapid system development and deployment.

We have implemented our approach in a feature-rich assistive system and conducted systematic controlled experiments in a number of well-designed activity scenarios. Initial results have demonstrated that the approach and algorithms are technically correct, viable and robust. Although the experimental dataset is not very large, it is representative and serves the purposes well. Our future work will focus on testing and evaluating our approach using publicly available activity datasets [43] [44] and also considering the exact impact of different noise levels on the performance of our approach.

References

[1] M. Chan, D. Estève, C. Escriba and E. Campo, "A review of smart homes—Present state and future challenges", *Computer Methods and Programs in Biomedicine*, vol.91, no.1, pp.55-81, 2008.

[2] C.D. Nugent, M. Mulvenna, X. Hong and S. Devlin, "Experiences in the Development of a Smart Lab", *Int. J. Biomedical Engineering and Technology*, vol.2, no.4, pp.319-331, 2009.

[3] Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hahnel, D.: Inferring activities from interactions with objects. *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 50-57, 2004.

[4] World Health Organization, International classification of functioning, disability and health (ICF) [Online]. Available: http://www.who.int/classifications/icf/en/.

[5] T.L.M. van Kasteren and Ben Krose, "Bayesian activity recognition in residence for elders", in *Proc. Int. Conf. Intelligent Environments*, 2007, pp. 209-212.

[6] D. Sanchez, M. Tentori, "Activity recognition for the smart hospital" *IEEE Intell. Syst.*, vol. 23, no. 2, pp. 50-57, 2008.

[7] K.P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning" PhD thesis, UC Berkeley, 2002.

[8] T.L.M. van Kasteren, G. Englebienne and B.J.A. Kröse, "Hierarchical Activity Recognition using Automatically Clustered Actions", in *Proc. Int. Joint Conf. Ambient Intelligence*, 2011, pp. 82-91.

[9] J. Hoey, T. Ploetz, D. Jackson, P. Olivier, A. Monk and C. Pham, "Rapid Specification and Automated Generation of Prompting Systems to Assist People with Dementia", *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 299-318, 2011.

[10] M. Brand, N. Oliver and A. Pentland, "Coupled hidden Markov models for complex action recognition", in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 1997, pp. 994–999.

[11] J.A. Quinn, C.K.I. Williams and N. McIntosh, "Factorial Switching Linear Dynamical Systems Applied to Physiological Condition Monitoring", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1537-1551, 2009.

[12] L. Bao and S. Intille, "Activity recognition from userannotated acceleration data", in *Proc. Pervasive, LNCS3001*, 2004, pp. 1–17.

[13] O. Brdiczka, J.L. Crowley and P. Reignier, "Learning situation models in a smart home". *IEEE Trans. Syst. Man Cybern. B., bern.*, vol. 39, no. 1, pp. 56-63, 2009.

[14] C. Sutton, A. McCallum and K. Rohanimanesh, "Dynamic Conditional Random Fields", *J. Machine Learning Research*, vol. 8, pp. 693-723, 2007.

[15] U. Maurer, A. Rowe, A. Smailagic and D. Siewiorek, "Location and Activity Recognition using eWatch: A wearable sensor platform", in *Ambient Intelligence in Everyday Life, LNCS Vol. 3864*, Springer-verlag Berlin, pp. 86-102, 2006.

[16] L. Liao, D. Fox and H. Kautz, "Hierarchical Conditional Random Fields for GPS-based activity recognition", in *Proc. Int. Symp. Robotics Research (ISRR)*, 2005, pp. 487-506.

[17] J. Lester, T. Choudhury, N. Kern, G. Borriello and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities", in *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2005, pp.766–772.

[18] T.L.M. van Kasteren, G. Englebienne and B.J.A. Kröse, "Transferring Knowledge of Activity Recognition across Sensor Networks", in *Proc. 8th Int. Conf. Pervasive Computing (Pervasive2010)*, pp283-300.

[19] P. Rashidi and D. Cook, "Activity knowledge transfer in smart environments", *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 331-343, 2011.

[20] H. Hu, Q. Yang, "Transfer learning for activity recognition via sensor mapping", in *Proc. 22nd Int. Joint Conf. Artificial Intelligence (IJCAI'11)*, pp. 1962-1967.

[21] D. Cook, K. Feuz, and N. Krishnan, "Transfer learning for activity recognition: A survey", *Knowledge and Information Systems*, vol. 36, no. 3, pp. 537-556, 2013.

[22] M. Perkowitz, M. Philipose, D. J. Patterson, K., "Mining models of human activities from the web", in *Proc. 13th Int. World Wide Web Conference (WWW 2004)*, pp.573-582.

[23] E. Munguia Tapia, T. Choudhury, and M. Philipose, "Building Reliable Activity Models Using Hierarchical Shrinkage and Mined Ontology," in *Proc. Pervasive2006*, pp.17-32.

[24] P. Palmes, H.K. Pung, T. Gu, W. Xue and S. Chen, "Object relevance weight pattern mining for activity recognition and segmentation", *Pervasive and Mobile Computing*, vol. 6, no. 1, pp. 43-57, 2010.

[25] H. Kautz, "A Formal Theory of Plan Recognition and its Implementation, Reasoning about Plans", Allen J., Pelavin R. and Tenenberg J. eds., Morgan Kaufmann, pp.69-125, 1991.

[26] W. Wobke, "Two Logical Theories of Plan Recognition", *J. Logic Computation*, vol. 12, no. 3, pp. 371-412, 2002.

[27] B. Bouchard, S. Giroux, "A Smart Home Agent for Plan Recognition of Cognitively-impaired Patients", *J. Comput.*, vol. 1, no. 5, pp. 53-62, 2006.

[28] L. Chen, C.D. Nugent, "A Logical Framework for Behaviour Reasoning and Assistance in a Smart Home", Int. *J. Assistive Robotics and Mechatronics*, vol. 9, no. 4, pp. 20-34, 2008.

[29] L. Chen, C.D. Nugent, "Semantic Data Management for Situation-aware Assistance in Ambient Assisted Living", in *Proc. 11th Int. Conf. Information Integration and Web-based Applications and Services (iiWAS2009)*, 2009, pp. 296-303.

[30] X. Hong and C.D. Nugent, "Segmenting sensor data for activity monitoring in smart environments", *Personal and Ubiquitous Computing*, vol. 17, no. 3, pp. 545-559, 2013.

[31] L. Chen, C.D. Nugent and H. Wang, "A Knowledge-Driven Approach to Activity Recognition in Smart Homes", *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 6, pp. 961-974, 2012.

[32] J. Ye, G. Stevenson and S. Dobson, "A top-level ontology for smart environments", *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 359-378, 2011.

[33] D. Riboni and C. Bettini, "OWL 2 Modeling and Reasoning with Complex Human Activities". *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 379-395, 2011.

[34] L. Chen, J. Hoey, C.D. Nugent, D. Cook and Z. Yu, "Sensor-Based Activity Recognition", *IEEE Trans. Syst. Man Cybern. C., pl. Rev.*, vol. 42, no. 6, pp. 790-808, 2012.

[35] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Englewood Cliffs, N.J.: Prentice Hall, ISBN:0-13-022278-X, 1988.

[36] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed., Elsevier, ISBN 978-0-12-

374856-0, 2011.
[37] G. Okeyo, L. Chen, H. Wang, R. Sterritt, "Dynamic Sensor Data Segmentation for Real time Activity Recognition", *Pervasive and Mobile Computing*, http://dx.doi.org/10.1016/j.pmcj.2012.11.004, in press, 2013.
[38] The Protégé framework [Online]. Available: http://protege.stanford.edu.
[39] Semantic Web RDF Library for C#.NET, [Online]. Available: http://razor.occams.info/code/semweb/.
[40] Euler proof mechanism [Online]. Available: www.agfa.com/w3c/euler/.
[41] Pellet: OWL 2 Reasoner for Java [Online]. Available: http://clarkparsia.com/pellet.
[42] F. Baader, D. Calvanese, D. L. McGuinness, "The Description Logic Handbook: Theory, Implementation, Applications", Cambridge University Press, 2003.
[43] TLM van Kasteren's dataset [Online]. Available: https://sites.google.com/site/tim0306/datasets.
[44] WSU CASAS dataset [Online]. Available: http://ailab.wsu.edu/casas/datasets/index.html.
[45] M.P. Lawton and E.M. Brody, "Assessment of older people: Self-maintaining and instrumental activities of daily living", *The Gerontologist*, vol. 9, no. 3, pp. 179-186, 1969.

Liming Chen is Professor of Computer Science at the School of Computer Science and Informatics, De Montfort University, UK. He received his B.Sc. degree and M.Eng. degree in Computer Engineering from Beijing Institute of Technology, China, and a Ph.D degree in Artificial Intelligence from De Montfort University, United Kingdom. His current research interests include ontology based data and knowledge modelling and reasoning, intelligent agents, activity recognition, personalization, assistive technologies and their applications in ambient assisted living, smart homes/cities and pervasive intelligent environments.

Chris Nugent (S'96–A'99–M'03) was born in 1973. He received the B.Eng. degree in electronic systems and the D.Phil. degree in Biomedical Engineering from the University of Ulster, Jordanstown, U.K., in 1995 and 1998, respectively. In 1998, he took the post of Research Fellow at the University of Ulster and now currently holds the post of Professor of Biomedical Engineering. His research interests include the design and evaluation of pervasive solutions in smart environments for ambient assisted living applications.

George Okeyo received the B.Sc. degree in mathematics and computer science and M.Sc. in information systems from the Jomo Kenyatta University of Agriculture and Technology (JKUAT) and the University of Nairobi, Kenya, respectively. He received the PhD degree in computer science from the University of Ulster, United Kingdom. He is a lecturer at the Department of Computing, JKUAT, Kenya. His current research interests include intelligent agents, smart homes, ambient assisted living, activity recognition, mobile and pervasive computing, data analytics, semantic technologies and knowledge representation and reasoning.