# Activities of Daily Life Recognition using Process Representation Modelling to Support Intention Analysis

**Abstract**

Activity of Daily Life (ADL) recognition plays an important role in tracking functional decline among elderly people who suffer from Alzheimer's disease. Accurate recognition enables smart environments to support and assist the elderly to lead an independent life for as long as possible. Current work has generally focused on applying a range of traditional classification and semantic reasoning based techniques in order to recognise ADLs. However, the ability to represent the complex structure of an ADL in a flexible manner remains a challenge. In this paper, we present an ADL recognition approach, which uses a hierarchal structure for the representation and modelling of the activities, its associated tasks and their relationships. We describe an approach in constructing ADLs based on a task-specific and intention-oriented plan representation language called Asbru. The proposed method is particularly flexible and adaptable for caregivers to be able to model daily schedules for Alzheimer's patients. A proof of concept prototype evaluation has been conducted for the validation of the proposed ADL recognition engine, which has comparable recognition results with existing ADL recognition approaches.

***Keywords:*** Activity Recognition; Alzheimer's disease; Assisted Living; Elderly Monitoring; Activities of Daily Life; Activity Modelling, Inference Engine, Unobtrusive

## 1 INTRODUCTION

Alzheimer's disease (AD) is the most common form of dementia and contributes 62% in comparison to other forms of dementia in the UK. This progressive disease of the brain is a fatal neurodegenerative disorder that is visible via cognitive and memory deterioration of elderly people as they try to carry out activities of daily living (Jeffery and Cummings, 2004).

Elderly people should be able to perform daily tasks such as cooking, dressing and other Activities of Daily Living (ADL), such as personal hygiene, eating, and functional movements. The ability to monitor ADLs in a ubiquitous environment (Aztiria et al., 2012), (Doctor et al., 2005) is seen as a key for tracking functional decline among elderly people (Fleury et al., 2010).

In the USA, caregivers prescribe a set of ADLs to elderly patients with dementia, which they are expected to conduct during the day; information is then collected on each regular visit from the caregiver, via interaction with the elderly person to see if they have been successfully carrying out the ADLs. Collected information by the caregivers is considered vital as medicines prescribed depends on it. However, collecting information in this manner can often lead to inaccurate data (McDonald and Curtis, 2001). Another drawback of this approach is the window used for collecting information, in comparison to the period being evaluated. Therefore, manual data collection regarding ADLs can be long and tedious which imposes further workload on caregivers.

In addition to information about the safety and wellbeing of an elderly person, recognition of activities can support providing assistance given a particular scenario.

For recognising activities of Alzheimer's patients, this research involves an ADL recognition approach. We introduce a novel concept of modelling hierarchal ADLs based on task-specific and intention-oriented plan representation language called Asbru (Fuchsberger et al., 2005). While prior work has focused on the lower tier of task recognition and the interaction with the higher tier of the framework (Naeem and Bigham, 2007a), (Naeem and Bigham, 2009), this paper focuses primarily on the higher tier ADL recognition that is based on understanding the constituent set of lower tier ADLs, and the novel approach for modelling these ADLs. We describe the ADL recognition engine with a worked example that is based on a hierarchal structure, which has the capability to generate a range of possible task sequences from a stream of sensor data in order to determine the ADL being conducted. The remainder of the paper is organised as follows. Section 2 provides an overview of the related literature, while Section 3 describes the hierarchal framework for ADLs. Section 4 describes the novel approach of modelling ADLs using Asbru, followed by the implementation overview of the ADL recognition engine in Section 5. Finally, experimental results are presented in Section 6.

## 2 RELATED WORK

Feature detection plays an important role in carrying out robust activity recognition. These data can be captured using visual surveillance equipment, which can be intrusive and computationally expensive when analysing video footage.

A popular alternative to vision systems is to capture object usage data which is made possible by 'Dense Sensing' (Buettner et al., 2009), (Philipose et al., 2004). This feature detection approach is based around numerous individual objects such as toasters and kettles being tagged with wireless battery-free transponders that transmit information to a computer via a Radio Frequency Identification (RFID) reader (Kalimeri et al., 2010), (Philipose et al., 2005) when the object is used or touched. Wearable sensors such as accelerometers can be seen as more intrusive than RFID tags; however, they are practical for capturing human body movements (Wang et al., 2007).

ADL recognition frameworks (Fleury et al., 2010), (Medjahed et al., 2009), (Ros et al., 2013) can be divided into two main categories, inductive and deductive. Inductive frameworks such as machine learning have the potential to learn and generalize by example (Delgado et al., 2009), (Ros et al., 2011), while deductive methods can provide powerful means to encode semantic process knowledge (Suzuki et al., 1999). The proposed hierarchal approach leverages both, as the lower task recognition tier is based on an inductive framework, while the higher tier ADL recognition is based on a deductive framework.

One of the favoured approaches in inductive frameworks is Hidden Markov Models (HMM) (Kim et al., 2010) used for probabilistic state transition models. Wilson et al. (Wilson et al., 2005) integrated HMMs and Viterbi algorithm for an activity recognition process. Unfortunately such approaches cannot recognise activities when they are carried out in a random order, typical in normal daily life activities. Sanchez et al. (2008) developed an approach for automatically estimating hospital-staff activities by training discrete HMMs for mapping contextual information to user activities. This approach suffers from 'cold start', as large datasets are required. Also, if there are large numbers of users with different ADLs and with a diversity of ways an ADL can be performed, this approach will be very slow and it will be difficult to learn each and every activity model for all users.

Novak et al. (2012) presented an approach for anomaly detection in users' activities by utilising data from unobtrusive sensors. The data utilised was from activities that had a typical duration of around 15 minutes or more. This particular approach was not able to detect any anomalies when dealing with interweaving activities, as the anomaly detection was based on the presence of a user at certain places which can not specifically distinguish between normal or abnormal activities.

Knowledge-driven techniques for ADL recognition have mainly focused on the use of ontologies to specify the semantics of activities, and semantic reasoning to recognise ADLs based on contextual information. Preliminary results suggest that existing ontological techniques underperform data-driven ones, mainly because they lack support for reasoning with temporal information. However, Riboni et al. (2011) conclude that when ontological techniques are extended with even simple forms of temporal reasoning, their effectiveness becomes comparable to one of a state-of-the-art technique based on HMM. We believe that hybrid statistical/probabilistic and semantic reasoning approaches have great potential for ADL recognition applications.

The most common ontology-based approaches to ADL recognition (Chen et al., 2012), (Riboni and Bettini, 2011) consists of specifying the semantics of ADLs based on the observation of a user's current context such as current location, current time, and objects that individual users are using. Chen et al. (2012) proposed a knowledge-driven approach to real-time and continuous ADL recognition in smart home environments that is based on ontological modelling and semantic reasoning. Their approach uses domain knowledge in the life cycle of activity recognition and exploits semantic reasoning and classification for inferring activities, enabling both coarser and finer-grained ADL recognition. Their work presents a generic system architecture based on their proposed knowledge-driven approach and describes the associate semantic and assumption reasoning algorithms for activity recognition. Riboni et al. (2011) defined an architecture for a mobile context-aware activity recognition system (COSAR) that is capable of detecting information about simple activities, which are recognised by a hybrid ontological and statistical reasoning approach. The architecture includes an ontology of human activities and reasoners, which executes on users' personal mobile devices. At run time, context information coming from distributed sources in an intelligent environment is retrieved and aggregated by the middleware. Context data are mapped to ontological classes and properties are added as individual instances belonging to theses classes. Ontological reasoning to recognise ADL activities is performed by the reasoning engine, either periodically or on the occurrence of specific events.

Scalability issues are a major challenge faced by activity recognition approaches. One such top-down, goal driven approach addressed this by hierarchally structuring activities, which is made up of execution conditions and abstract sensor mappings (Rafferty et al., 2013). The work proposed in this paper carries out a similar function, as it also structures ADLs as a hierarchal entity.

This work proposes a novel approach for activity modelling and recognition based on exploiting a process representation language called Asbru (Fuchsberger et al., 2005). Asbru is a task-specific and intention-oriented plan representation language for defining clinical guidelines, and protocols in XML. It represents clinical protocols as skeletal plans, which can be instantiated for each patient's specific treatment. These skeletal plans are useful guides for physicians when monitoring patients on a treatment protocol (Kosara et al., 1998). Asbru allows each skeletal plan to be flexible and to work with multiple skeletal strategies. Each ADL can be represented as a skeletal plan, which can then be instantiated with the specific ordering and temporal intervals based on the characteristics for each user. In comparison to other languages and methodologies, the ability to manage execution orderings of nested elements within skeletal plans and temporal phases are specific to Asbru for representing hierarchal ADLs. The flexibility and scalability of the proposed ADL modelling approach can also be used to compliment existing work (Okeyo et al., 2012), (Meditskos et al., 2013) and (Bouchard et al., 2007), which are based on different hybrid frameworks for complex activity recognition.

In addition, a recognition engine has been developed that exploits the plans that have been modelled as ADLs.

## 3 HIERARCHALLY STRUCTURED ACTIVITIES OF DAILY LIFE

ADLs have been modelled in a hierarchal structure, which enables scalable modeling of ADLs (Rafferty et al., 2013), (Lazaridis et al., 1994), (Kempen et al., 1995). The hierarchal structure has the capability to represent simple tasks such as "turn on tap", to more complex activities such as "making lunch". In order to accommodate the different degrees of complexity, ADLs have been modelled as *plans*, which can contain *sub-plans*. A *plan* that cannot be decomposed further is known as a *task*. This type of modelling requires two phases of recognition: task recognition phase and ADL recognition phase. When an ADL is performed, a task generates sensor events that are based on the objects (e.g. kettle) that are used to perform the ADL; hence, task recognition is solely based on recognising tasks from the captured sequence of sensor events. ADL recognition is responsible for recognising ADLs from the tasks identified in the task recognition phase.

Figure 1 gives a schematic representation of the Hierarchal ADL 'Make Breakfast', which shows a breakdown of the recognition phases. Starting from the bottom, a (potentially variable) number of sensor readings correspond to a particular task, which could be currently active. A number of tasks determine an ADL or set of ADLs that could be active. ADL *"Make Breakfast"* contains a simple sequence of tasks, *Make Tea*, *Make Toast*. Figure 1 illustrates a very simple sequence of sensor events that could be triggered whilst *Making Tea*. However the task recognition phase is also responsible for ensuring that it is able to deal with more complex sequence of sensor events. For example, the sensor event 'Kettle' could be triggered twice or more because the user might be *'filling the kettle'*, *'boiling* *kettle'* or *'pouring water from kettle'*. Therefore the RFID reader could capture the following sequence, *'kettle' 'tap' 'kettle' 'teabag', 'cup', 'kettle'*. This sequence of events would then be used by the lower tier task recognition to determine the task being conducted by the user.

For the lower tier task recognition phase, three different approaches have been developed. One is based on Multiple Behavioural Hidden Markov Models (MBHMMs), which accommodates different possible task orderings with different models (Naeem and Bigham, 2007a), while the second technique is based on an approach inspired by a text segmentation technique, namely Task Associated Sensor Events (TASE) segmentation (Naeem and Bigham, 2007b). This approach has the ability to segment tasks given a series of captured sensor event sequences. The third approach is an extension of the TASE approach, which generates a set of different task sequences from a stream of objects usage data that is based on the conjunction of the disjunction of tasks possibilities for each sensor event. This approach is called Generating Alternative Task Sequences (GATS) (Naeem and Bigham, 2009). The recognition engine presented in this paper employs this approach for the lower tier task recognition described next.

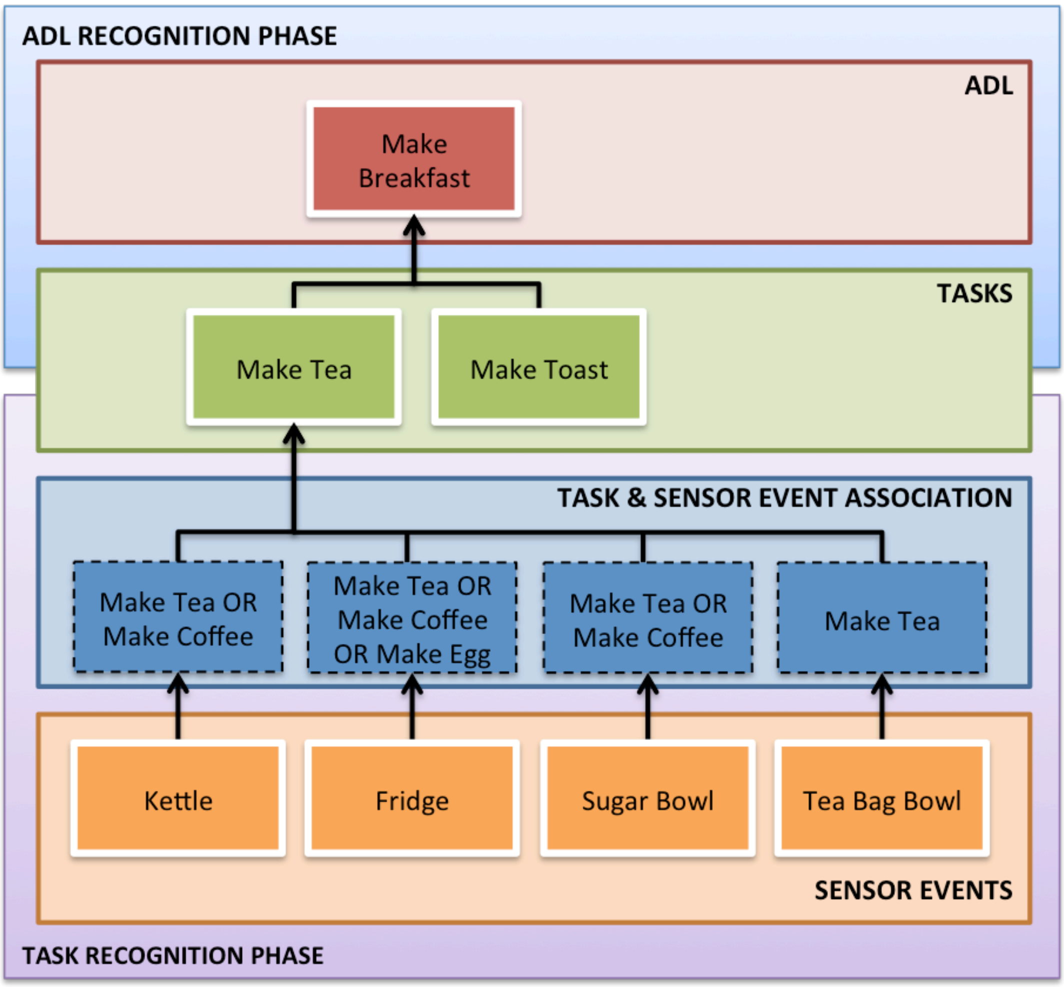


**Figure 1 - Hierarchal Activities of Daily Life Example**

### *3.1 ADL Recognition Phase*

The objective of the higher tier ADL recognition phase is to determine which ADL is being conducted based on identified tasks. In contrast to the approaches used in the lower tier task recognition phase, the ADL recognition phase required an approach that gave an overview of all possible ADLs that could occur within a given time. The approach had to consider any overlapping ADLs and also to distinguish which ADL is currently active by the tasks that are discovered at the lower level.

The elements of an ADL are made of behavioural patterns and the ADL itself can be classified as a type of behaviour. One way of representing and modelling high tier behaviour could have been by using workflows, such as using an augmented Petri Net (Zurawski and Zhou, 1994).

Within a workflow system, a process is used to represent a set of tasks that are required to occur in an agreed sequence in order to achieve an outcome (Browne et al., 2004). The goals of a person typically require particular constituent activities (tasks or sub-activities) to be ordered sequentially or in parallel. A majority of ADLs that the elderly people carry out are process oriented and so workflow systems are potentially a good modelling tool.

One approach for dynamic workflow processes to enable ad-hoc and evolutionary changes is in (Van der Aalst, 1999). Ad-hoc changes are usually caused by rare events occurring, while evolutionary changes often arise in order to make the workflow more efficient. An example of the latter could be removing unused nodes in the Petri net.

However, workflows are too prescriptive in their ordering. If workflows are applied in dynamically changing environments, they require a large number of permutations to be explicitly enumerated. Workflows can scale badly to cases where there are many possibilities and this is often the case for goals performed by people (Van der Aalst, 1999). In addition to scalability issues, it can be very difficult to manage the representation of priorities and ordering. Thus, more flexibility is required when modelling hierarchal ADLs.

## 4 MODELLING WITH ASBRU

The Asbru language is a process representation language which has similarities to workflow modelling, but has been designed to provide more flexibility than workflows. Its roots are in the modelling of medical protocols and monitoring the application of such protocols (Kosara et al., 1998). Asbru allows flexibility in how it can represent temporal events, namely their duration and sequence. ADLs have a number of attributes and characteristics which make them difficult to represent in a logical framework. These characteristics include variable duration, variable ordering of the tasks, and overlap with other ADLs. Techniques which attempt to map these as a flat structure are problematic because they are unable to model flexible scenarios, such as interweaving ADLs. The ability to monitor interweaving ADLs is a key advantage of the proposed approach over existing ADL modeling approaches (Chen and Nugent, 2009) (Aztiria et al., 2012). This is because the proposed approach has the ability to model a variety of temporal phases and execution orderings of sub-activities and tasks within ADLs, which provides the capability of representing and managing the execution of multiple ADLs.

In relation to the high tier modelling, Asbru is being used as a representation language to model ADLs. The skeletal plans in Asbru represent ADLs and sub-activities within an ADL. Like workflows, in Asbru when a goal is reached, it is represented as a plan being executed. In the case of the high tier modelling of an ADL, when all of the *phases* and *conditions* of an ADL have been met, the ADL can be classified as being executed. An ADL will only be classified as executed once all its mandatory sub-activities have been executed. For example, if a *Prepare Breakfast* ADL has a mandatory sub-activity called *Make Tea*, this sub-activity needs to be executed in order for the *Prepare Breakfast* ADL to be classified as successfully completed. The inclusion of a learning mechanism and context awareness data from smart devices (e.g. weather in local region) can also make it possible to adjust the mandatory options for a sub-activity. For example, sub-activity "put the coat on" could be changed from mandatory to optional during the summer or when the weather above a certain temperature.

In a real life scenario, the instantiation of the ADLs will be different depending on the individual who is being monitored; therefore in order to achieve reliable modelling the ADLs modelled in this paper are based on planned activity examples constructed by the Alzheimer's Association for people with dementia (Table 1). The person with dementia has an organised day consisting of activities to meet each individual's preference, enhance the individual's self esteem and improve quality of life (The Alzheimer's Association, 2012).

| Morning | Afternoon | Evening |
|---|---|---|
| - Wash, brush teeth, get dressed<br>- Prepare and eat breakfast<br>- Discuss the newspaper or reminisce about old photos<br>- Take a break, have some quiet time | - Prepare and eat lunch, read mail, wash dishes<br>- Listen to music or do a crossword puzzle<br>- Take a walk | - Prepare and eat dinner<br>- Play cards, watch a movie or give a massage<br>- Take a bath, get ready for bed |

**Table 1 - Daily Activity Plans constructed by Alzheimer's Association**

The proposed method may seem very prescriptive, as it relies on a crisp set of modelled ADLs; however we feel the use of this method is justified as patients with Alzheimer's require prescriptive assistance in the form of ADL schedules. While many common activities can be modelled within a library of ADLs, it is impossible for a library to contain plans for every possible ADL.

### *4.1 Phases and Conditions in ADL Execution*

With Asbru, each ADL can have 7 possible phases in its execution. The plan phase model is called the ADL phase model and shows a possible sequence of ADL phases. As shown in Figure 2, the first three phases (*considered, possible,* and *ready*) constitute the *preselection phase*, while the latter four (*activated, suspended, aborted,* and *completed*) forms the *execution phases*. For an *activated* ADL, the *suspended* phase, *completed* phase or *aborted* phase are optional.

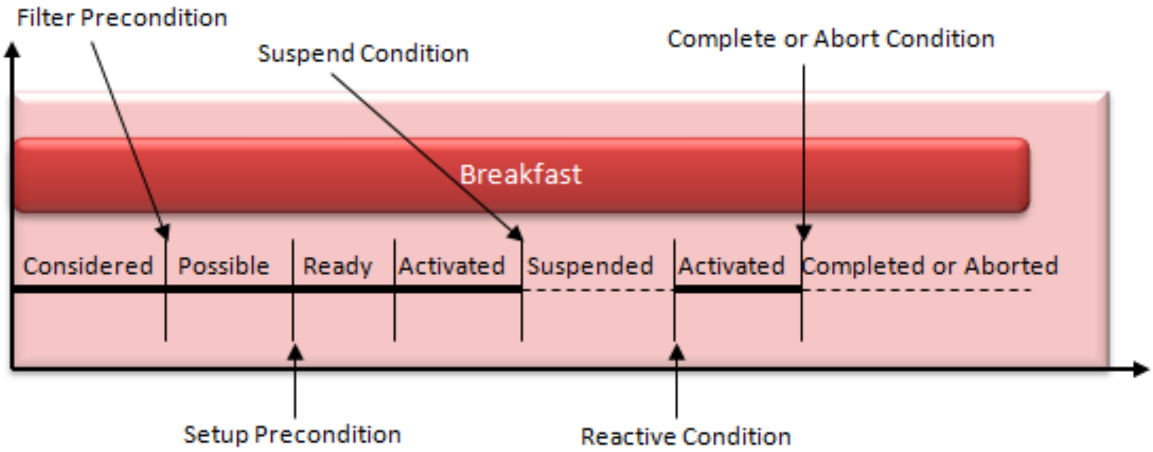


**Figure 2 - ADL phase model representation in Asbru**

#### *4.1.1 Preselection Phases*

*Considered:* This first phase of an ADL considers any filters to be fulfilled. If the filter preconditions are fulfilled then the ADL moves onto the possible phase. If the filter conditions are not fulfilled then the ADL does not execute any further. For example: The ADL "*Breakfast*" would only be considered if the person has been awake for '10' minutes or more. Figure 3 shows an XML representation of this filter precondition.

*Possible:* This pre-selection phase of the ADL checks whether all the setup preconditions of the main ADL have been fulfilled. Setup preconditions are imposed when the filter precondition cannot be achieved. These setup preconditions need to be fulfilled in order for the ADL to be in the ready phase. The difference between filter preconditions and setup preconditions is that filter preconditions consider whether it is possible for an ADL to be carried out, while setup preconditions must hold before an ADL can be executed. Setup conditions can also have a dependency on time. For example, if a setup condition is not fulfilled during a particular time frame that has been defined by an optional waiting period then the ADL is not executed. However, if there is no time frame assigned for the ADL, then the ADL stays in the possible phase until all preconditions have been fulfilled.

```xml
<filter-precondition>
    <parameter-proposition parameter-name="Person Awake">
        <is-known-parameter/>
        <context>
            <any/>
        </context>
        <time-annotation>
            <time-range>
                <starting-shift>
                    <latest>
                        <numerical-constant unit="min" value="-10"/>
                    </latest>
                    </starting-shift>
                    <finishing-shift>
                        <earliest>
                            <numerical-constant unit="s" value="0"/>
                        </earliest>
                        </finishing-shift>
                </time-range>
                <reference-point>
                    <now/>
                    </reference-point>
            </time-annotation>
        </parameter-proposition>
</filter-precondition>
```

**Figure 3 - Filter precondition in XML**

*Ready:* Once the setup conditions have been fulfilled then the ADL is ready for the activation phase. Depending on the type of ADL or sub-activities within the ADL, an ADL may not move to the activation stage straight away. If an ADL or a sub activity has to be executed in a parallel order then the ADL that is in the ready phase must wait for the ADL that is in the activated phase to be completed, aborted or suspended.

### *4.1.2 Execution Phases*

*Activated:* Before an ADL is activated it takes into consideration the activation condition. This condition is a token that determines if an ADL is manually or automatically. This is specified by using the attributes *overridable* and *confirmation*. These attributes are generally used for plans that have been modelled for clinical procedures and not used for ADLs. However, once an ADL is in the activated phase it will then either move on to any one of these three phases: *suspended*, *aborted* or *completed*. An example of an ADL being activated is when a task (i.e. *Make Tea*) has occurred that is a part of an ADL activity such as *Make Breakfast*.

*Suspended:* An ADL in an activated phase will only move on to the suspended phase if the conditions for suspension have been fulfilled. The only way an ADL can move out of the suspension phase and back into the activated is if the reactivate conditions have been fulfilled.

*Aborted:* An ADL in an activated phase will move to the aborted phase if the conditions for aborting the ADL have been fulfilled.

*Completed:* When an ADL is in the completed phase, all sub-activities (consisting of tasks from the lower tier recognition phase) and actions (tasks in the low tier) have been completed, thus allowing for the next ADL in the ready phase to be activated.

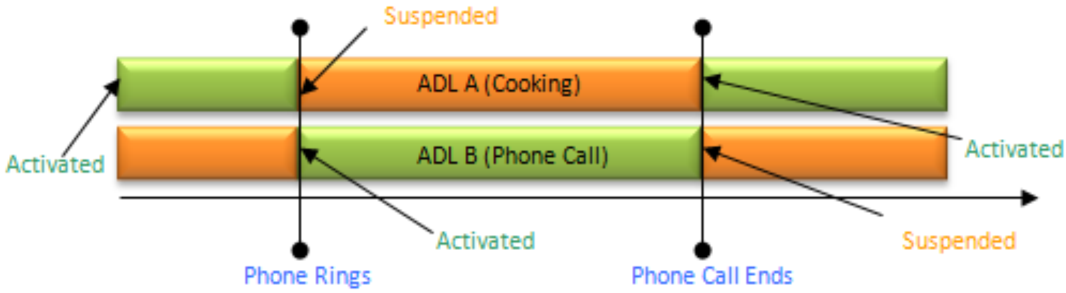


**Figure 4 - Using conditions to suspend tasks**

Some ADLs modelled with Asbru have preconditions that can only be started if a certain action (task) that satisfies the ADL's precondition *has been* executed. For example, a precondition for an ADL "*washing face*" may be to apply soap. Another feature of the condition element is that it allows ADLs to suspend and restart if another ADL is going to become active. For example (Figure 4), if an elderly person is *cooking* (ADL A) and a *phone call* occurs (ADL B) then the elderly person picks the phone up; with the aid of the conditions Asbru can suspend ADL A and start ADL B. Once an elderly person is off the phone then ADL A will be reactivated and ADL B will be suspended. The recognition of whether a particular phone call was completed successfully would be captured by the object usage data generated by picking up and putting down the handset. The time interval between these two actions enables the possibility of determining the call duration and different instances of phone calls.

When a suspension occurs it is important that certain conditions are satisfied (like the gas cooker is turned down) or certain monitors to check that certain conditions (such as the food on the hob is not boiling over) are setup.

The example in Figure 4 demonstrates the suspension and activation of two ADLs, and shows how an ADL resumes after being interrupted. However, this does not mean that another ADL could not be activated before the initial ADL resumes. This is important, as there might be situations where the elderly person with Alzheimer's disease conducts an initial activity and after an interruption forgets to resume that activity and starts executing another ADL. In this situation, the ADL that has been suspended may have a condition triggering abortion if the ADL has not been reactivated within a few hours.

### *4.2 ADL Execution Synchronisation*

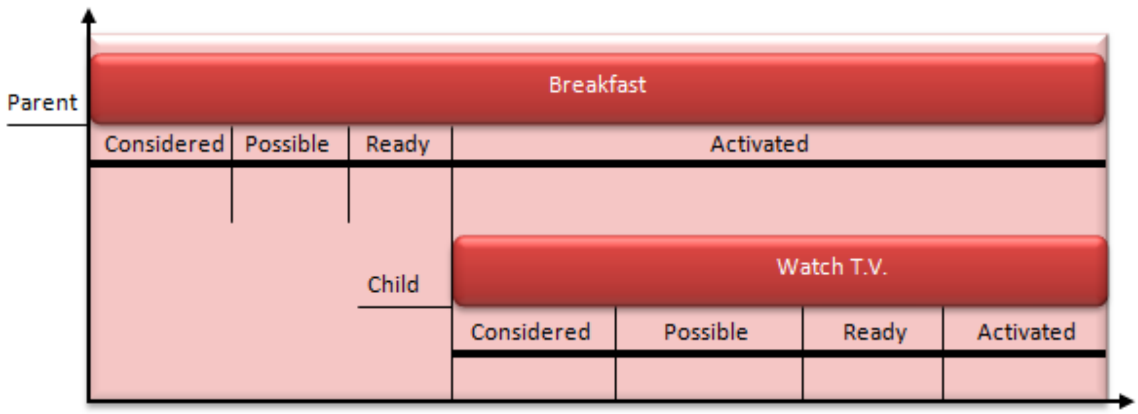


**Figure 5 - Parent-Child synchronisation between ADLs**

Asbru has the capability of representing and managing the execution of more than one ADL at a given time. In Figure 5, the child is an ADL (sub-activity "*Watch T.V.*") invoked by another parent ADL (ADL "*Breakfast*"). The child's preselection phase starts only after the parent's preselection phase terminates. In other words, the sub-activity's filter condition is not checked until the ADL is activated. Thus, an ADL is executed once the complete condition of the ADL has been fulfilled and all of its mandatory sub-activities have been completed.

Another aspect of Asbru is that it allows different ADLs to have different execution orders. The execution orders of an ADL have been represented as Sequential, Parallel, Any-order and Unordered execution order.

*Sequential Execution Order*
For an ADL that has a sequential execution order, its children execute in the prescribed sequence. The second ADL's pre-selection phase cannot begin until the first ADL completes or aborts (Figure 6). This is the same for any sub-activities that are sequential within an ADL that might not have a sequential execution order.

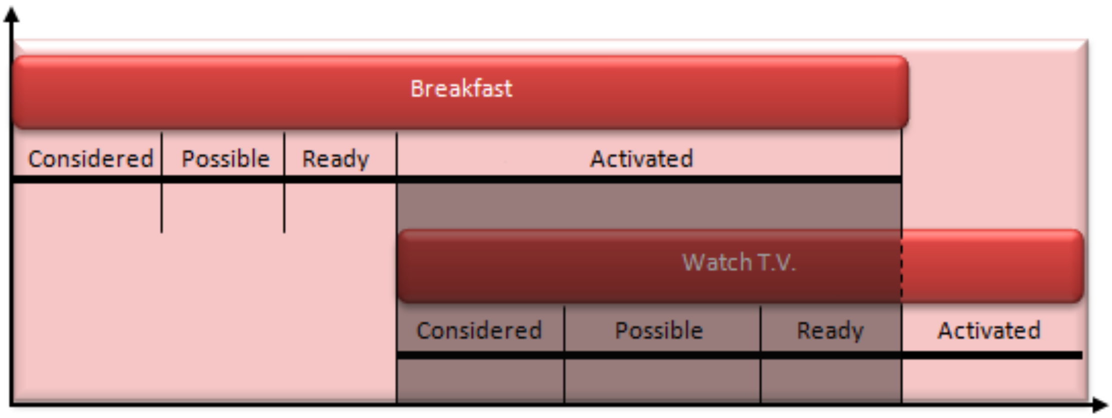


**Figure 6 - Sequentially ordered ADL**

*Parallel Execution Order*
All sub-activities with a parallel execution order are executed so that they are synchronised together. If the conditions or filters in the pre-selection phase of ADL 1 are not fulfilled, then ADL 2 has to wait until ADL 1 has fulfilled its conditions. If ADL 1 is aborted, then ADL 2 cannot be executed, which leads to ADLs not being executed (Figure 7).

For instance, an ADL "*Make Breakfast*" may have two parallel sub-activities, which are "*Make Tea*" and "*Make Coffee*", as the person being monitored may make tea for himself and coffee for someone else. In Figure 7, the pre-selection phase could be that the kettle has not reached the boiling point; hence "*Make Coffee*" has not been activated. Until the kettle reaches the boiling point, none of these sub-activities can be executed.


**Figure 7 - Parallel ordered ADL**

*Any-Order Execution*
With this type of execution the pre-selection phase is done in parallel to the other ADLs; however the execution is one at a time. The other ADLs remain idle when the execution of an ADL is taking place (Figure 8).


**Figure 8 - Any order ADL execution**

*Unordered Execution Order*
An ADL with an unordered execution order is able to execute all phases of an ADL together (in parallel) or in any order, which means that ADLs can stay idle throughout the pre-selection and execution phases (Figure 9).

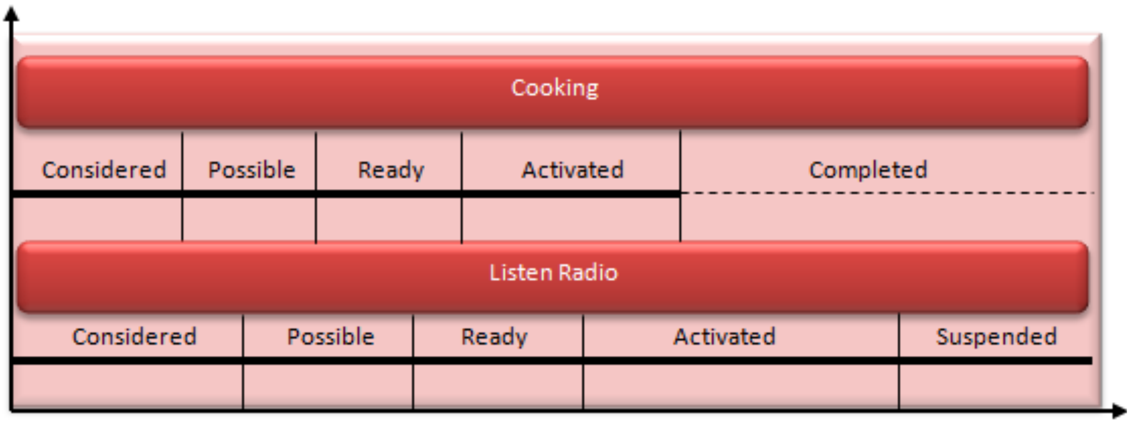


**Figure 9 - Unordered ADL execution**

## 5 ADL RECOGNITION ENGINE

The Java-based ADL recognition engine includes an ADL recogniser that reads the features captured, a sequence of sensor events (e.g. objects triggered during activity). These are then used to generate a sequence of tasks (e.g. *Make Tea* + *Make Coffee*) given the associated sensor event (e.g. *Kettle*). The ADL recogniser then reads the generated stream of tasks and calculates the possibility of each ADL being an active ADL using the discrepancies with each ADL and sub-activities that could currently be active. A discrepancy is the count of observed tasks that are inconsistent with a particular ADL. The system has also incorporated surprise indexes for each ADL, to reflect that some tasks are more likely than others.

The system reads in the ADL descriptors and stores them as a Document Object Model (DOM) tree. The ADL descriptors are constructed in XML as each ADL descriptor has the relevant sub-activities and tasks nested within them. The XML files are created either by hand as a source XML document or by a graphical tool called AsbruView [26].

Once the ADL descriptors have been loaded into memory, the ADL recogniser acts as a server which listens for incoming task notifications. These task notifications are the tasks that have been determined from the lower tier task recognition phase. After each task is read, the estimator outputs the names of the ADLs and sub-activities that may be currently active. Depending on how many ADLs the task belongs to, the ADL recogniser provides a list of the most probable ADLs that may be currently active. The output of the most probable ADL is determined by a dynamic weight tuning, which is based on the level of discrepancies and surprise indices that are calculated by the ADL recogniser. The following illustrates the behaviour of the ADL recognition engine with a sample sensor event stream.

### *5.1 Lower Tier Task Inference*

The lower tier recognition component computes a list of all possible task sequences given a sequence of sensor events (object usage data), which is based on the conjunction of the disjunction of tasks possibilities for each sensor event. For example, the object *'Kettle'* $(x)$ can be associated with the tasks *'Making Tea'* $(A)$ or *'Making Coffee'* $(B)$. Hence the sensor event *'Kettle Sensed'* is replaced by the disjunction Making Tea | Make Coffee, which is represented as $x = A + B$, where $+$ is used to represent the disjunction.

Each task sequence has a generated cost, where the highest task sequence is considered to be the most likely task sequence as the cost function reflects the compliance of the task sequence with the captured sensor event sequences and the relative frequencies of ADLs. The function of the lower tier task recognition component can be represented as:

$$e1, e2, \ldots en \ \rightarrow \ \{< TS1, c1 > + < TS2, C2 > + < TSm, cm >\} \qquad (1)$$

In (1), $TS$ represents a task sequence, where $m$ is an upper limit chosen when the task recogniser is asked for its set of task sequences that match the events. This also ensures that if there are fewer than $m$ possibilities, then only the actual possibilities are generated. For example, the sensor events sequence will generate the following task sequences and associated costs, where A, B, C and D are tasks

$$e1, e2, e3 \ \rightarrow \ \{< ABC, c1 > + < ABD, c2 >\} \qquad (2)$$

In the following example the captured sensor event sequence $(e1, e2, e3, e4)$ contains the task *'Make Tea'* being carried out. However there are also some partial tasks that could potentially be carried out given the sensor events such as *'Make Coffee'* and *'Make Toast'*. The notation below maps the objects as sensor events, and tasks as letters:

$e1 = kettle$
$e2 = sugar\ bowl$
$e3 = refrigerator$
$e4 = tea\ bag\ bowl$
$e5 = coffee\ bowl$
$e6 = toaster$

$A = Make\ Tea$
$B = Make\ Coffee$
$C = Make\ Toast$

These are then represented as task associated sensor events, with their associated prior conditional probability values. As there are no training data, these values are based on the number of associations each task has with the sensor event.

Sensor event sequence = $e1, e2, e3, e4$

$e1 = A\ (0.5) + B(\mathbf{0.5})$
$e2 = A\ (0.5) + B(0.5)$
$e3 = A\ (0.33) + B(0.33) + C(\mathbf{0.33})$
$e4 = A\ (\mathbf{1})$
∴
Maximum Conditional Probability Values equal the following:
A = 1
B = 0.5
C = 0.33

Given the sensor event stream $e1, e2, e3, e4$ the disjunction will be as follows:
$(A + B)(A + B)(A + B + C)\ A$
$= \ (AA + AB + BA + BB)(A + B + C)\ A$
AA can be reduced and the task sequences are be generated by computing the function in the following way:
$= \ (A + AB + BA + B)(A + B + C)\ A$
$= \ (A + ABA + BA + AB + BABA + B + AC + ABC + BAC + BC)\ A$
$= \ A + ABA + ACA + ABCA + BA + BABA + CA$

In order to generate the associated cost for each task sequence, the maximum of the conditional probability values assigned to each task above is used to find the cost for each task sequence; for example: $ABA = (A)1 * (B)0.5 * (A)1 = 0.5$. The associated costs for each of the task sequence are as follows:

$$e1, e2, e3, e4 \ \rightarrow \ \{< A, 1.0 > + < ABA, 0.5 > + < ACA, 0.33 > + < ABCA, 0.165 > + < BA, 0.5 > + < BABA, 0.25 > + < CA, 0.33 >\}$$

This approach mitigates the chances of not being able to infer tasks that have been performed using different orderings of objects, as this approach considers all the possible types of task sequences given the task associated sensor events. Also this approach considers the conjunction of the disjunction of tasks possibilities for each sensor event, including noise. For example, if a sensor event sequence is made up of noise then this will be reflected in the generated cost, as this task sequence will have a lower cost than other tasks sequences. This means that the captured sensor events do not comply with the task sequence.

### *5.2 Dynamic Weight Tuning by Computing Discrepancies & Surprise Indices for ADL Inference*

Recognition of ADLs is dependent on the tasks that have been inferred in the lower tier; hence the recognised task is used as input in order to determine the ADL that it belongs to.

When constructing an ADL descriptor it is possible to construct one ADL per XML file, or several ADLs can be constructed into one larger XML file. Both of these options are likely to lead to a situation where one XML file will contain the same tasks.

When an ADL has been detected by the ADL recogniser this is represented by the absolute pathname of the nested elements within ADL XML file that has been detected, for example:

- ADL: Make Breakfast
    - Sub-Activity: Prepare Food
        - Task: Make Tea

*Make Breakfast → Prepare Food → Make Tea*

In an XML file, a discrepancy is a task (i.e. single step plan) that has not been detected or that should have been detected if the ADL were executed. The overall discrepancy of an ADL is computed by summing the discrepancies of its sub-activities.

To compute the overall discrepancy, two discrepancy counts for each ADL are calculated, namely the completed and incomplete discrepancy counts. If the sub-activity is known to be complete then the completed discrepancy of the sub-activity is used when computing the sum; otherwise the incomplete discrepancy is used.

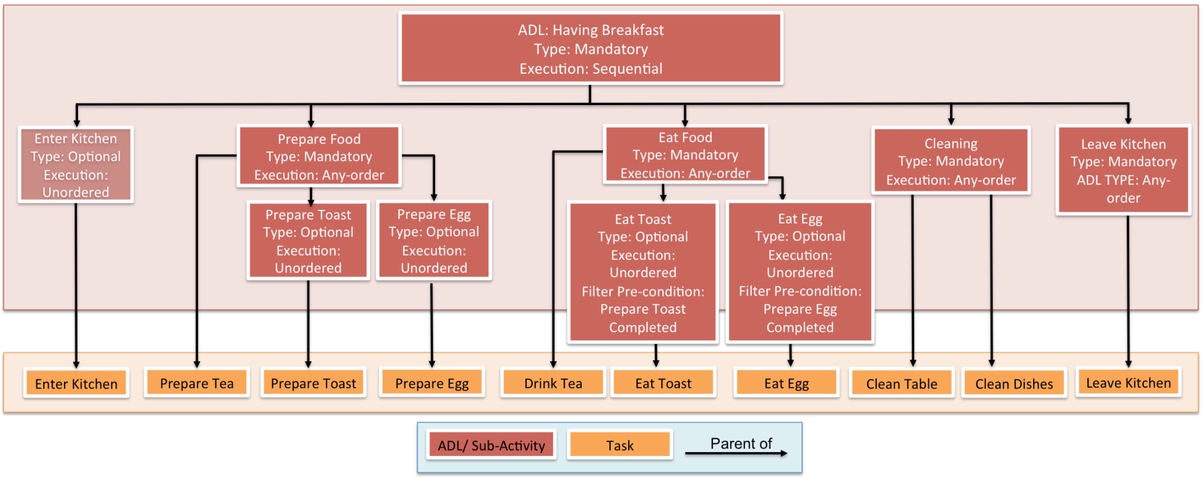


**Figure 10 - Modelled ADL example of 'Having Breakfast'**

Whether an ADL has been logically completed or not it is represented by true or false of its completed labels. The completed labels have a default false value. All labels in the absolute path name of an XML file are set recursively to true once a new task is detected in the XML file. When the completed label is set to true, the ADL can be idle as tasks within this ADL might be detected later.

The mechanism to mark labels as complete is based on:

1. The execution order - sequential, parallel, any-order, or unordered.
2. The continuation condition – whether a sub-activity is optional or mandatory for its parent ADL's continued execution.
3. The filter pre-condition – the compulsory conditions for an ADL to be activated.

Once a new task is detected or otherwise known as completed in the higher tier component, the following discrepancy counting processes occur:

<u>Process 1:</u> If the parent ADL has filter preconditions, then all other ADLs that are compulsory to fulfill the pre-conditions should have been completed. Hence these ADLs are set as being completed.

<u>Process 2:</u> If all tasks and mandatory sub-activities of an ADL have been set to completed, then this ADL is set as being completed.

<u>Process 3:</u> An ADL is only set as completed, once it has been completed, according to the assigned order of execution. For example, if a parent ADL is sequential, then all its preceding mandatory child ADLs should have been completed in the sequential order. This is also true for ADLs that have parent plans that are either parallel, any-order, or unordered, as the child ADLs will only be set as completed once they have been executed in a particular order.

<u>Process 4:</u> If an ADL has been set as completed then all mandatory children should have been completed; hence these mandatory children are set to complete. This process traverses down the ADL to the sub-activities that are nested within it.

Process 5: The process continues in a depth first search like manner - traversing from the current ADL to its siblings, then parents, repeating process 1-4 until no parent ADL is available. The completed discrepancy and incomplete discrepancy of each ADL are updated if any changes take place.

### *5.2.1 Working Example*

A working example has been modelled to illustrate how discrepancies are computed for a simple "*Having Breakfast*" ADL (Figure 10): It is supposed that the following tasks are detected in the low level modelling – "*Enter Kitchen*", "*Prepare Toast*", "*Drink Tea*", "*Eat Egg*", "*Clean Dishes*", and "*Leave Kitchen*" - in this order. At the detection of each task in the higher tier (e.g. output task from lower tier), the above recognition processes (1 to 5) will take place. By convention, in Asbru all single task plans are mandatory. If a single task needs to be optional it has to be embedded in another optional plan, which can contain the single activity.

1. Enter Kitchen is detected

*Enter Kitchen* is the only task in sub-activity *Enter Kitchen*; hence Process 2 will occur here and the single step plan (task) *Enter Kitchen* is set to completed. The update process continues and stops when reaching the sequential root plan *Have Breakfast*, since *Enter Kitchen* has no preceding plans.

2. Prepare Toast is detected

Similar to the case when *Enter Kitchen* was detected, Process 2 also occurs here and the single step plan (task) *Prepare Toast* is set to completed. The discrepancy counting algorithm goes to the sequential sub-activity *Prepare Food*, and Process 3 occurs because the single step plan (task) *Prepare Tea*, which as a preceding mandatory child, should have been completed. However, it has not been detected and is calculated as a discrepancy. The update process continues until the root ADL is reached.

3. Drink Tea is detected

Process 2 occurs as sub-activity *Eat Food* has been set as completed since the only mandatory child single step plan (task) *Drink Tea* is completed. Also the sub-activity *Prepare Food* is set to complete, as it is a preceding sub-activity to *Eat Food*. *Prepare Food* is set to complete because there is a possibility that the task recognition component may have not discovered the task *Prepare Tea*. Even though the sub-activity *Prepare Food* has now been set to complete, the discrepancy count remains the same. An ADL plan that has a high discrepancy count is less likely to be the ADL that is being conducted.

4. Eat Egg is detected

Process 1 occurs as in order to fulfill the filter condition "*egg is cooked*", the single step plan (task) *Prepare Egg* should have been completed. Like the previous *Prepare Tea* situation, *Prepare Egg* is also set to complete, where the discrepancy count for the sub-activity *Prepare Food* remains the same and does not decrement.

5. Clean Dishes is detected

Any-order sub-activity *Cleaning* is not set to completed because only the task clean dishes was detected and both of the tasks (*clean dishes* and *clean table*) were required in order for the sub-activity to be set to complete, as the sub-activities were mandatory.

6. Leave Kitchen is detected

Process 2 occurs as the ADL *Leave Kitchen* is set to completed; also as this is the last task of the task sequence the overall discrepancy of the ADL can be calculated.

The completed discrepancy and incomplete discrepancy counts of each ADL, sub-activity and single step plan (tasks) are updated if any changes take place. The overall discrepancy is calculated as the sum of the chosen completed or incomplete discrepancies of each ADL and sub-activity.

In this example, the modelled ADL's final matching result is shown in Table 2. The overall discrepancy of "*Having Breakfast*" is 3; if other ADLs have a higher overall discrepancy than 3 then "*Having Breakfast*" is the ADL rated as being conducted. The recognition process does not necessarily just rely on the overall discrepancy, as at each step when a task is discovered the individual discrepancies and complete labels can be used to assist the recognition process, meaning there is no need to wait for a complete stream of task sequences before determining the activity.

The surprise index is used to account for the absence of some sensor events being more unusual than others, and quantifies this by accruing a measure of how likely a sensor event is when a task is being executed.

| ADL/Sub Activities/Task | Execution Order | Mandatory or Optional | Complete Label | Complete Discrepancy Count | Incomplete Discrepancy Count |
|---|---|---|---|---|---|
| Having Breakfast | Sequential | Root Plan | False | 0 | 0 |
| Enter Kitchen | Unordered | Optional | True | 0 | 0 |
| Leave Kitchen | Unordered | Optional | True | 0 | 0 |
| Prepare Food | Any-order | Mandatory | True | 1 | 0 |
| Prepare Toast | Unordered | Optional | True | 0 | 0 |
| Prepare Egg | Unordered | Optional | True | 1 | 0 |
| Eat Food | Any-order | Mandatory | True | 0 | 0 |
| Eat Egg | Unordered | Optional | True | 0 | 0 |
| Eat Toast | Unordered | Optional | False | 0 | 0 |
| Cleaning | Any-order | Optional | False | 0 | 1 |
| Overall discrepancy of plan Having Breakfast is 3 | | | | | |

**Table 2 - ADL Discrepancies for Having Breakfast**

While the discrepancy is computed whenever there is any missing mandatory task, such as "*Make Tea*" for the ADL "*Having Breakfast*", the surprise index of a missing sub-activity is the maximum of the conditional probabilities $P[a_i|b]$ of its missing sub-activities tasks occurring $[a_i s]$ given that the ADL $[b]$ is being conducted. A mandatory task will have probability of 1. The maximum is taken over the immediate sub-activities or tasks, i.e. children. This approach is cautious and ad hoc, but the information required to use a more sophisticated approach, would need significant knowledge collection. Equation (3) is used to compute the maximum likelihood (ML) probability estimates, $N(o_i, a)$ represents the number of times object $o_i$ occurs in activity $a$, while $|o|$ represents the cardinality of the set of all objects (Tapia et al., 2006).

$$P(o_i|a_j) = \frac{N(o_i, a_j)}{\sum_{s=1}^{|o|} N\,(o_s, a_j)} \tag{3}$$

## 6 PROOF OF CONCEPT PROTOTYPE EVALUATION

The objective of this evaluation was to investigate the performance of the ADL recognition engine, which utilises the hierarchal ADLs modelled with Asbru. The effectiveness of the ADL modeling and performance of the proposed recognition engine has been measured by calculating the precision and recall rates of each ADL given its constituent tasks that were performed.

The precision $(P)$ and recall $(R)$ are calculated as follows:

$$\begin{aligned} P &= \frac{True\ Positive}{True\ Positive + False\ Positive} \\ R &= \frac{True\ Positive}{True\ Positive + False\ Negative} \end{aligned} \tag{4}$$

The feature detection technique employed was based on the collection of object usage data, where an RFID reader was used to collect data from transponders that had been installed on household objects (such as cup, kettle and utensils) in the kitchen. The RFID reader was the size of a matchbox and was worn on the finger of the subject conducting the experiment. There were occasions where the RFID reader captured objects that were not part of the ADLs being conducted. Hence it was imperative that the proposed approach was able to address this problem.

| ADL | Sub Activities | Tasks |
|---|---|---|
| Breakfast | Enter Kitchen (*) → | Enter Kitchen |
| | Prepare Food → | Make Tea, Make Coffee, Make Toast |
| | Eat Food → | Drink Tea, Drink Coffee, Eat Toast |
| | Cleaning → | Clean Table, Clean Dishes |
| | Exit Kitchen (*) → | Exit Kitchen |
| Prepare Lunch | Enter Kitchen (*) → | Enter Kitchen |
| | Prepare Food → | Make Sandwich, Make Wrap |
| | Cleaning → | Clean Table, Clean Dishes |
| | Exit Kitchen (*) → | Exit Kitchen |
| Put Shopping Away | Enter Kitchen (*) → | Enter Kitchen |
| | Groceries Away → | Fridge Shopping Away<br>Cupboard Shopping Away |
| | Exit Kitchen (*) → | Exit Kitchen |
| Prepare Snack | Enter Kitchen (*) → | Enter Kitchen |
| | Prepare Food → | Make Tea, Make Coffee, Get Biscuits |
| | Eat Food → | Drink Tea, Drink Coffee, Eat Biscuits |
| | Exit Kitchen (*) → | Exit Kitchen |
| Clean Kitchen | Enter Kitchen (*) → | Enter Kitchen |
| | Clean Floor → | Sweep Floor |
| | Clean Worktop → | Wipe Countertop with Wipes |
| | Exit Kitchen (*) → | Exit Kitchen |
| Laundry | Enter Kitchen (*) → | Enter Kitchen |
| | Wash Clothes → | Wash Clothes - Washing Machine<br>Dry Clothes using Tumble Dryer |
| | Exit Kitchen (*) → | Exit Kitchen |
| Prepare Ready Meal | Enter Kitchen (*) → | Enter Kitchen |
| | Warm up Meal → | Heat up food in Microwave |
| | Exit Kitchen (*) → | Exit Kitchen |
| Clean Up (Post Lunch) | Enter Kitchen (*) → | Enter Kitchen |
| | Cleaning → | Clean Table, Clean Dishes |
| | Clean Floor → | Sweep Floor |
| | Exit Kitchen (*) → | Exit Kitchen |

*Optional

**Table 3 - ADL, Sub Activities, Tasks Conducted for Experiments**

Ten volunteers were recruited from the community to carry out these experiments. The volunteers were asked to perform each ADL three times by changing the ordering of objects used, which also increased the degree of variation within the data collected. The subjects reported the ADLs they had conducted within a given time frame. This reported information was then used as ground truth.

The experiments were based around 8 ADLs, which were made up of a series of sub-activities and tasks that belonged to more than one ADL (see Table 3). This was done intentionally to see how the ADL recogniser would deal with tasks that belong to more than one ADL.

Figure 11 shows that the precision rates ranged from 93% to 86%, which is based on the instances of the ADLs recognised being relevant to the actual ADL being performed. The recall rates ranged from 96% to 90%, indicating that the hierarchal modelling enables the ability to consider all possible relevant tasks when performing ADL recognition. This is very important, as a task could belong to more than one ADL; hence it is important the recognition process does not rule out task sequences during the recognition process. The recall and precision rates suggest that the proposed hierarchal modelling in Asbru with the ADL recognition engine was able to return more relevant recognition instances of an ADL as opposed to irrelevant instances.

This experiment has also shown that the performance depends very much on the degree of overlap between tasks in different ADLs. Because of this, different scenarios with different degrees of overlap are currently being considered.

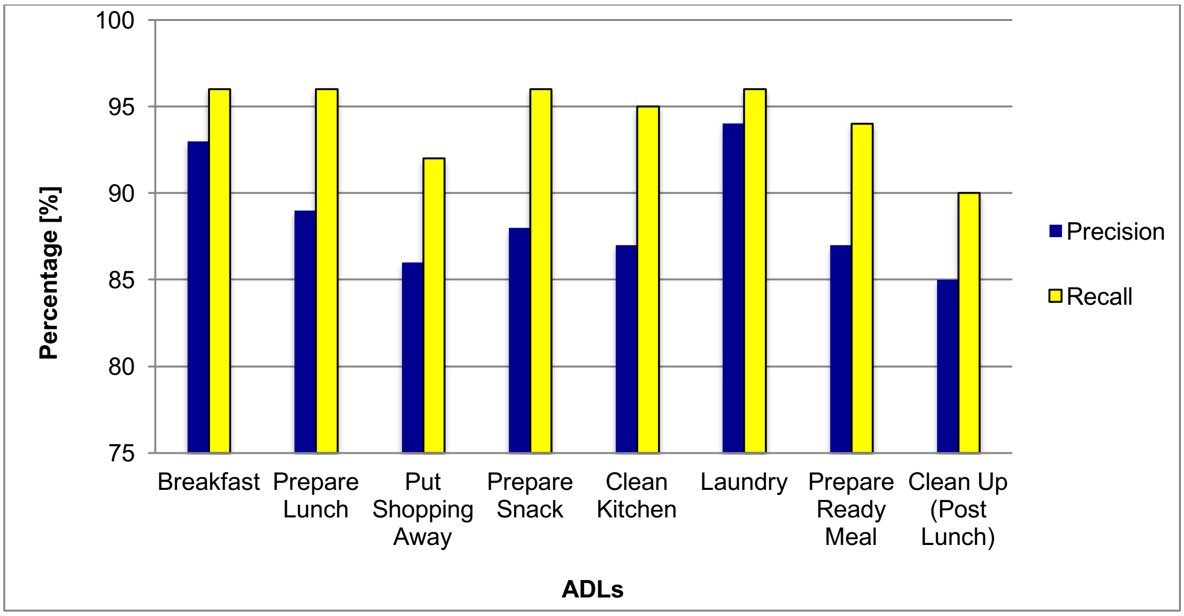


**Figure 11 - ADL Precision and Recall Rates**

However, even for such a scenario, the ADL recogniser still needs to be improved. This of course, depends on the nature of the ADL. The more optional sub-activities and the more sharing of sub-activities the more difficult it is to be absolutely certain. However, even if ADLs are not identified uniquely, the set of possible ADLs may be enough to a) give feedback to the task identification system and b) support context sensitive help - as the ADLs may be related.

These experiment results are comparable in terms of the recognition rates with existing ADL recognition approaches (Rashidi et al., 2011),(Baños et al., 2013). However, it is important to highlight that the other approaches have deployed feature detection techniques that capture richer data (e.g. acceleration data for movement, ambient temperature readings, analogue sensors for hot & cold water, phone usage data and pressure sensors). The approach presented in this paper is primarily based on object usage data collected by RFID transponders, hence if the proposed approach employed a similar feature detection technique then this would have further improved the recognition results.

## 7 CONCLUSION

This research focused on how ADLs could be structured in a hierarchal structure based on the fundamentals of the task-specific and intention-oriented plan representation language called Asbru. The experiment conducted indicates that the hierarchal structure of ADLs makes it possible to recognise an ADL even though all of the tasks within the ADL may not have been fully completed or correctly recognised by the lower tier recognition. This is made possible by the flexible nature of how the ADLs have been modelled in a hierarchal structure.

This work is a very important step towards carrying out intention analysis. As the representation of prescribed ADLs for Alzheimer's patients can enable recognition systems to be preemptive when trying to determine the future actions of a person. This can enable the possibility of safeguards being initiated before a certain activity takes place. However, the proposed approach does not currently run in real-time, which is a major limitation. In order to make this possible a series of bottlenecks will need to be addressed. Firstly, we will need to address issue of deploying a learning mechanism, which will support the knowledge base. Secondly, a challenge that needs to be addressed is determining the length of the captured sensor events stream that will be useful for real-time inference. One option could be to carryout inference at regular timing intervals, however this could be very inefficient as only the most recent events could be of interest. These challenges will be carried out as part of future work.

Additionally, the proposed ADL recognition modelling and inference approach could be applied to the lower tier task recognition level, as tasks could also be modelled as an activity that is composed of the tasks corresponding to sensor events that need to be executed within a certain order and within certain temporal constraints.

## REFERENCES


The Alzheimer's Association. (2012) "Activities at Home, Planning the day for a person with dementia", available at: http://www.alz.org/national/documents/brochure_activities.pdf (accessed 20 October 2014)

Aztiria, A., Augusto, J., Basagoiti, A., and Cook, D. (2012), "Discovering frequent user-environment interactions in intelligent environments", *Personal Ubiquitous Computing*, vol. 16, no. 1, pp. 91-103.

Baños, O., Damas, M., Pomares, H., Rojas, F., Delgado-Márquez, B.,and Valenzuela, O. (2013), "Human activity recognition based on a sensor weighting hierarchical classifier", *Soft Computing*, vol.17, no.2, pp.333-343.

Bouchard, B., Giroux, S. and Bouzouane, A. (2007),"A Keyhole Plan Recognition Model for Alzheimer's Patients: First Results", *Applied Artificial Intelligence*, Vol.21, No.7, pp. 623-658.

Browne, E.D. , Schrefl, M. and Warren, J.R. (2004), "Goal-focused Self-Modifying Workflow in the Healthcare Domain", in *Proceedings of the 37th IEEE International Conference on Systems Sciences, Hawaii.*

Buettner, M., Prasad, R., Philipose, M. and Wetherall, D. (2009), "Recognizing Daily Activities with RFID-Based Sensors", in *Proceedings of the 11th International Conference on Ubiqitous Computing*, *Orlando,* pp.51-60.

Chen, L., and Nugent, C. (2009), "Ontology-based activity recognition in intelligent pervasive environments", *International Journal of Web Information Systems*, vol. 5, no. 4, pp. 410-430.

Chen, I., Nugent, C.D. and Wang, H. (2012), "A knowledge-driven approach to activity recognition in smart homes", *IEEE Transactions on Knowledge and Data Engineering*, vol.24, pp.961-974.

Delgado, M., Ros, M., and Vila, A. (2009), "Correct behaviour identification system in a tagged world", *Expert Systems with Applications*, vol.36, no.6, pp.9899-9906.

Doctor, F., Hagras, H., and Callaghan, V. (2005), "A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol.35, no.1, pp.55-65.

Fleury, A., Vacher, M. and Noury, N. (2010), "SVM-Based Multimodal Classification of Activities of Daily Living in Health Smart Homes: Sensors, Algorithms, and First Experimental Results", *IEEE Transactions on Information Technology in Biomedicine*, vol.14, no. 2. pp.274-283.

Fuchsberger, C., Hunter, J. and McCue, P. (2005), "Testing Asbru Guidelines and Protocols for Neonatal Intensive Care", in *Proceedings of the 10th Conference on Artificial Intelligence in Medicine, Aberdeen, UK*, pp.101-110.

Jeffery, L., and Cummings, M.D. (2004), "Alzheimer's disease", *The New England Journal of Medicine, Drug Therapy*, vol.351, pp.56-67.

Kalimeri, K., Matic, A., and Cappelletti, A. (2010), "RFID: Recognizing failures in dressing activity", in *Proceedings of the 4th International Conference on Pervasive Computing Technlogies for Healthcare (PervasiveHealth),* pp.1-4.

Kempen, G.I.J.M., Myers, A.M., and Powell, L.E. (1995), "Hierarchical structure in ADL and IADL: Analytical assumptions and applications for clinicians and researchers", *Journal of Clinical Epidemiology*, vol.48, no.11, pp.1299-1305.

Kim, E., Helal, S., and Cook, D. (2010), "Human Activity Recognition and Pattern Discovery", *IEEE Pervasive Computing*, vol.9, no.1, pp.48-53.

Kosara, R., Miksch, S., Shahar, Y., and Johnson, P. (1998), "AsbruView: Capturing Complex, Time-oriented Plans - Beyond Flow-Charts," in *Second Workshop on Thinking with Diagrams, Aberystwyth, UK*, pp.119-126.

Lazaridis, EN., Rudberg, MA., Furner, SE., and Cassel, CK. (1994), "Do activities of daily living have a hierarchical structure? An analysis using the longitudinal study of aging", *Journal of Gerontol*, vol. 49, no.2, pp.47-51.

McDonald, A and Curtis, J. (2001), "Not Alone: A Good Practice Guide to Working with People with Dementia in Sheltered Housing", *Anchor Trust*.

Meditskos, G., Dasiopoulou, S., Efstathiou, V. and Kompatsiaris, I. (2013), "SP-ACT: A Hybrid Framework for Complex Activity Recognition Combining OWL and SPARQL Rules", in *Proceedings of the International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 25-30.

Medjahed, H., Istrate, D., Boudy, J., and Dorizzi, B. (2009), "Human activities of daily living recognition using fuzzy logic for elderly home monitoring", in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp.2001.

Naeem, U. and Bigham, J. (2007a), "A Comparison of Two Hidden Markov Approaches to Task Identification in the Home Environment," in *Proceedings of the 2nd International Conference on Pervasive Computing and Applications, Birmingham, UK*, pp.383-388.

Naeem, U. and Bigham, J. (2007b), "Recognising Activities of Daily Life Using Hierarchical Plans," in *Proceedings of the 2nd European Conference on Smart Sensing and Context, LNCS 4793, Lake District, UK*, pp.175-189.

Naeem, U. and Bigham, J. (2009), "Activity Recognition in the Home using a Hierarchal Framework with Object Usage Data", *Journal of Ambient Intelligence and Smart Environments*, vol.1, no.4, pp. 335-350.

Novak, M., BInas, M. and Jakab, F. (2012), "Unobtrosive anomaly detection in presence of elderly in a smart home environment" in *Proceedings of ELEKTRO*, pp.341-344.

Okeyo, G., Chen, L., Wang, H. and Sterritt, R. (2012), "A Hybrid Ontological and Temporal Approach for Composite Activity Modelling", in *Proceedings of the International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp.1763-1770.

Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J. , Kautz, H. and Hahnel, D. (2004), "Inferring Activities from Interactions with Objects", *IEEE pervasive Computing Magazine*, vol.3, no.4, pp.50-57.

Philipose, M., Smith, J.R., Jiang, B., Mamishev, A., Roy, S. and Sundara-Rajan, K. (2005), "Battery-Free Wireless Identification and Sensing," Pervasive Computing IEEE Journal, vol. 4, no.1, pp.33-45.

Rafferty, J., Chen, L., and Nugent, C. (2013) "Ontological Goal Modeling for Proactive Assistive Living in Smart Environments", in *Proceedings of the 7th International Conference, UCAmI 2013, Carrillo, Costa Rica, December 2-6, 2013*, pp.262-269.

Rashidi, P., Cook, D., Holder, L., and M. Schmitter-Edgecombe. (2011), "Discovering activities to recognize and track in a smart environment", *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no.4, pp.527-539.

Riboni, D. and Bettini, C. (2011), "COSAR: hybrid reasoning for context-aware activity recognition", *Personal and Ubiquitous Computing*, vol.15, issue.3, pp.271-289.

Riboni, D., Pareschi, L., Radaelli, L. and Bettini, C. (2011), "Is ontology-based activity recognition really effective?", in *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp.427-431.

Ros, M., Delgado, M. and Vila, A. (2011), "Fuzzy method to disclose behaviour patterns in a tagged world", *Expert Systems with Applications*, vol.38, no.4, pp.3600-3612.

Ros, M., CuéLlar, M.P., Delgado, M., and Vila, A. (2013), "Online recognition of human activities and adaptation to habit changes by means of learning automata and fuzzy temporal windows", *International Journal Information Sciences*, vol.220, pp.86-101.

Sanchez, D., Tentori, M. and Favela, J. (2008), "Activity Recognition for the Smart Hospital", *IEEE Intelligent Systems*, vol. 23, no. 2, pp.50-57.

Suzuki, M., Matsushima, T. and Hirasawa, S. (1999), "On a deductive reasoning model and method for uncertainty", in *Proceeding of the IEEE Conference on Tools with Artificial Intelligence*, pp.161-164.

Tapia, E. M., Choudhury, T. and Philipose, M. (2006), "Building Reliable Activity Models Using Hierarchical Shrinkage and Mined Ontology", in *Proceedings of the 4th International Conference on Pervasive, Dublin, Ireland*, pp. 17-32.

Van der Aalst, W.M. (1999), “How to Handle Dynamic Change and Capture Management Information? An Approach Based on Generic Workflow Models,” *Technical Report, University of Georgia, Department of Computer Science, Athens*.

Wang, S., Petney, W., Popescu, A., Choudhury, T. and Philipose, M. (2007), “Common Sense Based Joint Training of Human Activity Recognizers” in *Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, pp.2237-2243.

Wilson, D.H., Wyaat, D. and Philipose, M. (2005), “Using Context History for Data Collection in the Home”, in *Proceedings of Pervasive 2005: Workshops - ECHISE. Munich, Germany*.

Zurawski, R. and Zhou, M. (1994), “Petri Nets and Industrial Applications: A Tutorial in Industrial Electronics,” *IEEE Journal* , vol.41, no.6, pp.567-583.