Detecting Anomalous Elderly Behaviour in Ambient Assisted Living

Marco Manca, Parvaneh Parvin, Fabio Paternò, Carmen Santoro
CNR-ISTI, HIIS Laboratory
Pisa, Italy
{marco.manca, parvaneh.parvin, fabio.paterno, carmen.santoro}@isti.cnr.it

ABSTRACT
The increasing availability of sensors and intelligent objects enables new functionalities and services. In the Ambient Assisted Living domain such technologies can be used for monitoring the elderly behaviour, and reasoning about it to detect possible anomalous situations, which could be a sign of the next onset of chronic illness or initial physical and cognitive decline. In this paper we propose a solution that exploits task models describing *expected* user behaviour, and a context manager able to detect relevant contextual events and conditions determined by the *actual* elderly behaviour. Planned and actual behaviour are compared to detect if any deviation occurred. The resulting environment is able to generate multimodal actions such as reminders and alarms aiming to provide useful support when such anomalous behaviour is detected.

Author Keywords
Elderly Behaviour Analysis, Deviations in Task performance, Ambient Assisted Living.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI);

INTRODUCTION
Nowadays, ambient intelligence is increasingly used in order to support people in their everyday life by means of a plethora of heterogeneous sensors, actuators, appliances, devices, smart objects and services. The analysis of the data so gathered can support automatic monitoring of numerous physiological and environmental data useful for identifying the current state and activity of people. This is especially important in supporting solutions for ageing people in Ambient Assisted Living (AAL) scenarios. Indeed, with the rapid growth of the elderly population, there is an increasing need for supporting them in preserving an independent and healthy lifestyle in their homes rather than through more expensive hospitalization solutions. One important aspect in AAL is the monitoring of daily living activities [5], which is widely used in health care, and refers to daily routines on which the ability of a person to live independently is assessed (e.g. bathing/showering, dressing, food preparation, eating, personal hygiene, taking medicines, telephoning). However, while there are several contributions addressing the issue of activity or routine recognition in smart homes, relatively fewer solutions have addressed the challenging topic of detecting “unusual” behaviour on user’s side, i.e., possible deviations from expected users’ routines. This topic is highly critical for the deployment of remote monitoring systems for elderly, because changes in the elderly’s routines can be a potential sign of elderly’s changing health conditions in the next future due to e.g. initial signs of decline, beginning of unhealthy habits, modifications of user’s abilities. In addition, the identification of anomalous patterns can be highly valuable for experts to make decisions in emergency or risky situations or can be used to provide the elderly themselves with some relevant information and notifications helping them in properly carrying out their own activities. Anomalies could affect various routines aspects e.g. the order according to which activities are carried out, the place where the activity takes place (e.g. sleeping in the bed vs. on the sofa), the time (e.g. too early/too late) and duration (e.g. too long/too short) of the activity compared to the expected one. Furthermore, anomalies can have various degrees of severity (e.g. the consequence of not taking a pill could be more severe than just forgetting to take a shower). In addition, they can be detected by considering short timeframe monitoring periods (e.g. if detected data suddenly present values that are significantly different from the expected ones) or they could require a longer monitoring timespan in order to be identified as an actual risky situation (e.g. in case of more gradual changes in user behaviour). In this paper, we present an approach to detecting abnormal behaviour due to significant changes in user behaviour in AAL scenarios. Our solution exploits task model specifications in which the user's expected behaviour is represented, and compares such behaviour with sequences of events logged in the current context, which should provide information about the actual user behaviour. By using associations between

/EICS '17,/ June 26-29, 2017, Lisbon, Portugal
© 2017 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5083-9/17/06…$15.00
http://dx.doi.org/10.1145/3102113.3102128

elementary tasks defined in the considered task model and events gathered in the current context, the system is able to detect possible deviations and, depending on the type of deviation identified, trigger accordingly suitable actions addressing the concerned anomaly. In the paper, we first discuss related work, then provide a scenario to highlight the main issues addressed. We present the methodology as well as the architecture specification of the solution. Then, we describe and discuss an example application. Finally, we conclude with some remarks and plans for future research.

RELATED WORK

Previous work [6] has considered task models to compare expected user behaviour and actual user interactions with Web applications in order to identify possible usability issues. However, while in that work only Web interaction events are analysed, here we consider events that are triggered by user behaviour in daily life (e.g. associated with ubiquitous and physiological sensors), and whose anomalous appearance can have more severe consequences than the ones generally brought about by usability issues. Finally, while in the previous work the deviation analysis was part of an asynchronous, retrospective usability evaluation to identify parts of the Web application that need further improvements, here we also analyse the identification of suitable actions to perform in real time to cope with the identified deviation.

There have been several contributions dealing with the topic of detecting anomalous behaviour. A survey [1] about approaches used for activity and anomaly detection in smart homes basically identifies two ways to detect behavioural changes: *profiling* and *discriminating*. Profiling is modelling normal behaviour and considering values that do not comply with the model as anomalies. Discriminating is learning anomaly data from historical data and searching for a similar pattern from incoming data to consider it an anomaly. Profiling is more realistic as anomaly data is rarely seen in real life to provide the classifier with a sufficient number of sample instances to learn. We have developed a *profiling* strategy (in which task models specify the 'normal' behaviour), which can also work in case of rare anomaly data. In this area, Pollack et al. [7] describe Autominder, a system intended to help older adults adapt to cognitive decline and continue a satisfactory performance of routine activities, thereby potentially enabling them to remain in their own homes longer by providing adaptive, personalized reminders of (basic, instrumental, and extended) activities of daily living. Their solution uses a range of intelligent techniques to model an individual's daily plans, observe and reason about the execution of those plans, and make decisions about whether and when it is most appropriate to issue reminders. The effect of a deviation from the specified plan is the generation of a reminder, while here we present a system able to specify a wide range of actions depending on the detected anomaly. Jakkula et al. [2] describe a method to determine if anomalies can be effectively detected in smart home data using temporal data mining. They believe anomaly detection is most accurate when it is based on behaviours that are frequent and predictable. For this purpose, they mine the data for frequent sequential patterns using the Apriori algorithm through which they identify temporal relations (e.g. turn TV on before sitting on the coach) that occurred between events in these frequent sequences. The final step involves calculating the probability of a given event occurring (or not occurring), which forms the basis for anomaly detection. While they use machine learning in order to recognize frequent patterns, we exploit task models specified with the help of relevant stakeholders in order to describe the expected behaviour because it is not guaranteed that frequent user patterns represent the correct behaviour of elder people. Monekosso and Remagnino described a model-based behaviour analysis system for assisted living [3], in which the behaviour is defined as any pattern in a sequence of observations and the models are generated from sensed data. A model-based approach is employed for the detection of deviation from the expected behaviour: given a model of the system, the predicted output generated by the model is compared to the actual output, and any difference is a potential failure. While in that paper the behaviour is defined as an identifiable pattern in a sequence of observations, thus susceptible to errors, in our work caregivers contribute to define and specify elderly expected behaviour.

EXAMPLE SCENARIO

An example scenario considers the usual expected *morning routine* of an elderly (which in turn could be part of a more general daily routine). The task model of this scenario is shown in Figure 1. After waking up, and going to the bathroom, John should take a medicine (pill_1) without food. Then he prepares his breakfast and only after eating he should take another medicine (pill_2). Afterwards he relaxes in some ways (e.g. reading news, making a phone call to relatives, watching TV), then he should do some physical activity (at home or outside). Afterwards, he rests again and then he starts the lunch routine. From this scenario it is possible to derive various ways in which John's activities can be performed in a different manner from the expected one. One of the most critical deviations involves medicine intake: the ingestion of the two pills can be swapped in time (e.g. pill_1 is wrongly taken after breakfast and pill_2 is taken before breakfast), or the two pills can be wrongly taken together (e.g. both with food or both without food, so erroneously consuming one pill in any case). Alternatively, John could replace one of the two pills with a third one (e.g. pill_3) which should be taken later on in the day. In other cases, John could completely forget to take one or both drugs and thus he misses some medicines. On the contrary, in other situations he could repeat one or both intakes, thus assuming more medicines than prescribed. Deviations involving critical tasks, such as medicine intake, should receive quite prompt attention. In the scenario considered in Figure 1 it is possible to easily identify the deviations involving pill intake because they affect just the "Take medicines" subtask. Thus, the deviation remains quite 'localized' in the task model.

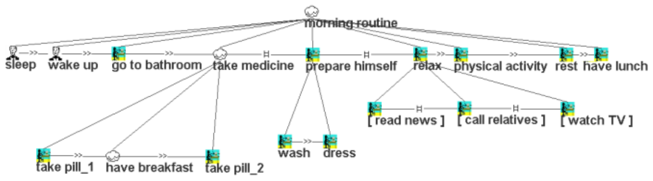


Figure 1: An Elderly morning routine specified in CTT

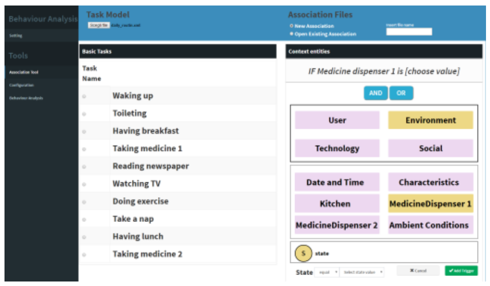
Other types of deviations can be detected only considering a longer (e.g. medium) time span. For instance, going to the toilet is not a deviation per-se, but it could be the sign of a next illness onset only if it occurs (too) frequently over a relevant period of time (e.g. during the whole day). Finally, other types of deviations can be uncritical in the short/medium term but they could become risky in a long-time span (e.g. the user starts to sleep longer and longer). In this case, since the involved changes in the routines are gradual, they need long timespans (e.g. multiple days or even multiple weeks) to be actually interpreted as a risky situation. As such, they are usually associated with a more 'global' deviation (i.e. affecting the whole task model over multiple iterations).

METHOD

Our method is composed of several steps, which are described in the following. The first steps are performed in the preparation phase in which all the necessary data and connections among them are created. Then, we have the steps performed while the elderly are actually monitored in their daily behaviour.

Set up a suitable task model for the analysis. In this step, a task model describing the planned activities from the user's viewpoint should be created with the help of relevant stakeholders (e.g. elderly's caregivers). The task models considered in this approach are specified according to the ConcurTaskTrees (CTT) language [4], through which hierarchical models of tasks connected by various temporal operators (e.g. choice, enabling disabling) can be expressed. However, to be exploited by the automatic behaviour analysis, the task model should be checked for its correctness by the relevant stakeholders. In addition, since a task model can include activities of various categories, another step consists of *pruning* the task model so that it will contain only the tasks relevant for the automatic analysis. Indeed, the CTT notation allows designers to indicate how the performance of the task should be allocated (to the user, to the system, to their interaction, or a combination of different types of allocation) according to these categories: application, interaction, user, abstraction. However, for the automatic analysis, only tasks that can be detected through sensors and/or user interactions are relevant, thus at the finest level of granularity we should limit the analysis only to *interaction* and *application* tasks. Consequently, user tasks, which are associated with only internal cognitive activities are pruned from the model. It is worth pointing out that in this work we consider not only common daily living activities but also user's access to services and user interfaces through digital devices and appliances available at home. In particular, *interaction* tasks refer to actions implying an on-purpose interaction with any physical or digital artefacts, while *application* tasks refer to either feedback from the application or to application actions that occur without being an effect of a specific user interaction.

Associate events occurred in the context with elementary tasks in the task model. In this step, we need to associate each elementary task in the pruned task model (see Figure 2) to one or more events gathered in the context. This association allows for using the semantic information contained in task models to analyse the sequence of corresponding associated events: when the event(s) associated with a task occurs, then the corresponding task is considered completed. However, it is worth noting that tasks could not have the same granularity of events. In some cases, there is a 1:1 mapping between tasks and events (e.g. the "user switches on the lights" task corresponds to the event: "light switched on"). In other cases, multiple events can be associated with a single elementary task (N:1), thus a complex event expression is needed. For instance, if we consider the "sleeping" task, then a corresponding event expression (depending on the available sensors) could be: (user posture= lying down AND user's heart rate= low AND user's bed pressure sensor=on AND user motion=none). In case of a complex event expression the task is interpreted as actually completed only when all the associated events or conditions have been detected in the current context.



Task Name
Waking up
Toileting
Having breakfast
Taking medicine 1
Reading newspaper
Watching TV
Doing exercise
Take a nap
Having lunch
Taking medicine 2

IF Medicine dispenser 1 is [choose value]

Figure 2: The association tool

Definition of the actions to perform when the deviation is detected

Different types of actions can be triggered depending on the deviation detected: *Modify the user interface* (presentation / content / navigation); *Activate some functionality*; *Generate alarms* (to highlight some potentially dangerous situations); *Send reminders* (to indicate a task that should have been accomplished); *Provide persuasive suggestions* (to encourage users to change their behaviour); *Send prompt* *messages*; *Provide explanation messages*; *Change the state of an appliance* (light, fridge, …)

Log relevant contextual events. The goal of this step is to log the events occurring in the context where the elderly actually live, also associating them with a timestamp needed for further analysis (e.g. precisely identifying the beginning of an activity, calculate how much time the user spent doing a specific task). As we will see in the next section describing the architecture of the system, there is a Deviation Analysis module, which subscribes to the Context Manager to receive such events, and then save them in a specific data structure for further analysis. It is worth pointing out that the events considered by the Deviation Analysis module will not be all the possible events that could be gathered in the current context, but only the ones which are related to the elementary tasks contained in the task model.

Compare the actual user behaviour with the expected behaviour specified in the task model and identify possible deviations.

This comparison consists of checking whether the sequence of basic tasks associated with the sensed events satisfies the temporal constraints specified in the task model. This means comparing from time to time, the *set of tasks* enabled in the task model (according to the current execution state of the model) and the task(s) associated with the real events just occurred in the context: if the latter one does not belong to the first one, then a deviation occurs. Several types of deviations can be identified as a result of this comparison. In particular, since user's activities can be described in terms of a set of task attributes (such as temporal relationships, location, time, resources/entities handled), it is possible to identify a number of corresponding types of task-related anomalies affecting elderly people. Some examples include i) unusually long inactivity: when a task has a temporal relationship attribute and no other tasks has been executed after that for a long period (e.g. because the elderly has fallen and has lost consciousness); ii) unusually short activity which equally refers to the temporal dimension (e.g. the lunch was too short); iii) unusually frequent activities when a task is not marked as iterative but it is performed repeatedly (e.g. too many visits to the toilet); iv) violations of task order relationship (taking medicine before having a meal instead after that); v) wrong object/resource handled (e.g. the elderly unintentionally took a medicine not prescribed instead of the right one); vi) missing task (the user goes to sleep without having dinner or the user does not do exercise). To sum up, anomalies occurring on tasks can be categorized according to the following types:

- *Less*: a task that was expected to be carried out, has not been performed;
- *More*: more tasks than expected have been performed;
- *Difference*: expected tasks have been performed but in a wrong manner (in terms of e.g. temporal order, time, location, objects manipulated) from the planned one;

Each type of deviation can be *critical* or *uncritical* depending on the attribute criticality level associated to the task(s) involved in the deviation.

Analyse the deviation and identify a suitable action for addressing it. In case a deviation is detected, in this step the system analyses the type of deviation (in terms of e.g. distance from the expected behaviour, criticality level of involved task) and identifies possible suitable remedial actions to take (e.g. generate alarms or reminders to the elderly) in order to cope with the identified anomaly, even by escalating them if no reaction from the elderly is obtained, e.g. sending an alarm message also to the caregiver if no action is obtained in reaction to an alarm sent to the elderly. Depending on the current context, the notification to be sent to the user can be adapted by considering various factors, for instance, how to render it (for instance, using a vocal message if the users are far from the device in order to get their attention).

ARCHITECTURE

Figure 3 shows the architecture of the system that we have designed and developed to support the proposed method. Its behaviour can be described according to two main phases: one is devoted to configuring the analysis (e.g. building/selecting the task model, building/selecting the association file), the other one is devoted to carrying out the automatic analysis and performing the consequent actions. In the first phase, the user has to select the task model to be used, which specifies how the various activities are expected to be carried out. In addition, using the Association Tool (Figure 2), the user has to build the concerned association file, containing the mappings between the elementary tasks of the task model that are of interest for the deviation analysis, and the associated events detected in the current context. The tasks that appear in such a list are basic tasks (see Figure 2 on the left side), tasks that are not further decomposed in the model. As said before, the idea underlying the associations is the fact that the occurrence of a specific event of interest in the current context (they can be interactively selected through the user interface shown in Figure 2 on the right side) can be translated into the completion of an associated elementary task in the task model. The second step more properly deals with the detection of possible anomalies in the sensed behaviour, and it is centred on the Behaviour Analysis module, which is composed of three sub-modules: the Simulator, the Deviation Analysis, and the Action Generator. The basic goal of the Simulator module is to provide the Deviation Analysis with the information needed to decide whether the sequence of events detected by the Context Manager can be translated in a path of elementary tasks that is compliant with the temporal relationships contained in the task model representing the expected behaviour. In order to do this, we implemented a Task Model Simulator which gets in input i) the specification of a complete and consistent CTT task model, and ii) the next task to perform in the simulation (corresponding to the relevant event occurred in the context of use). Depending on the task temporal operators, the Simulator calculates the updated list of the tasks currently enabled after the execution of the given task to perform on the current state of the simulation, and returns such updated list to the Deviation Analysis module. Before starting the Deviation Analysis, the user has to define the actions that should take place when a deviation occurs. These are actions that can be performed in the considered application or commands to change the state of some appliance (e.g. changing colour and intensity of lights).

The Deviation Analysis module first subscribes to the Context Manager in order to be notified when the events specified in the association file occur in the context. The Context Manager is a software module able to receive the events that occur in the sensors deployed in the context, abstracts out for the specificities of the different raw data formats coming from different sources, and then provide a uniform and homogenous representation of such context information. After having subscribed to the Context Manager, the Deviation Analysis receives relevant events from it and then it is able to check about any occurring deviation in such sequence of events. In order to do this, the Deviation Analysis module gets in input three elements: i) the association file; ii) the enabled task sets (from the simulator); iii) the event lastly occurred in the current context (from the Context Manager). The behaviour of the Deviation Analysis is the following one: it takes the event just occurred in the context (and provided by the Context Manager), retrieves in the association file the corresponding elementary task, and checks whether such task belongs to the set of enabled tasks (provided by the Simulator module). If it is the case, it means that the received event was an expected one (according to the task model), then the process iterates in a similar manner for the next received event. Otherwise the Deviation Analysis module signals to the Action Generator module the deviation that occurred in the detected behaviour. Depending on the type of deviation identified, the critical level of the involved task, and taking into account other contextual aspects, the Action Generator module is in charge of selecting the best action to do for addressing the identified anomaly. For instance, if the deviation is of type *less* and it involves a critical task, then the Action Generator could decide that an alarm should be sent to relevant stakeholders. In particular, it generates a high-level action (specified according to a specific XML-based language) to be sent to the application, which interprets it and performs accordingly the requested UI changes (e.g. modify the font size of some elements to make them more readable) or activate some functionality (e.g. generate an alarm) to deal with the identified deviation. The Action Generation module, in addition to the identified deviation, also receives in input some *rules*, which define the most suitable ways to render the actions identified for addressing the identified deviation by taking into account relevant contextual aspects. For example, if there is a rule such as "If the noise level is less than 50 dB, then sends reminders both textually and vocally", when the Action Generation module receives a notification of a deviation for which a reminder should be sent, then it asks the Context Manager the value of the noise level and depending on the current value, the Action Generator will require the application to render the identified action (e.g. send a reminder) in different modalities (both graphical and vocal if the noise level is low). The rules are specified in a Trigger-Action format (specified according to a XML-based language), and they are supposed to be built with the help of relevant stakeholders (e.g. caregivers). The application mentioned before can be any Web application for elderly remote assistance. When such application is loaded, it has to subscribe to the Behaviour Analysis in order to be notified when a deviation is detected. Moreover, the application must include suitable JavaScript scripts able to interpret the messages received by that module (e.g. the adaptation actions to perform) and apply the required actions on the application elements and structure.

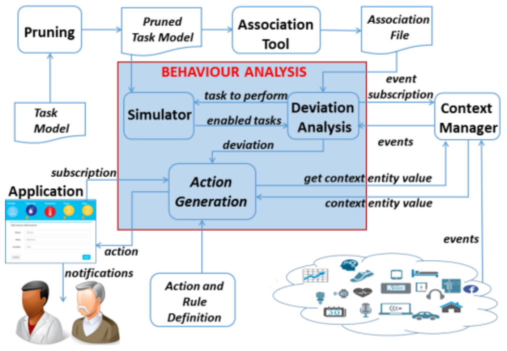


Figure 3: The Architecture of the Solution

EXAMPLE APPLICATION

In the example considered we more precisely focus on the task of medicine intake (see Figure 4). The user should take first the Spironolactone pill (used to treat fluid retention in people suffering from heart failures) without food (pill 1 in Figure 4), then he should have breakfast and only after that he should take the Diuretics pill (pill 2 in Figure 4). Before being used for the actual analysis, all the user tasks (e.g. the task "Plan what to have for breakfast") are filtered out. The tasks "Take pill 1" and "Take pill 2" are associated with the events generated by a Smart Medicine Dispenser which sends to the Context Manager an update every time the elderly takes a pill (we assume that the Smart Dispenser is able to detect the specific pill the user is currently taking). The task "Have breakfast" is split into two sub-tasks: "Prepare Breakfast" and "Eat breakfast". In turn, "Prepare Breakfast" is composed of three sub-tasks that can be executed iteratively in any order: 1) "Take food from refrigerator”, 2) “Cook food” and 3) “Use the sink”. The first subtask is associated to sensors installed in the refrigerator door, the second subtask can be related to a gas-sensor attached to the stove, while the third one is linked to a sensor connected to the sink tap. Then, the elderly has breakfast (which is detected according to the specific sensors available e.g. pressure on the chair is detected for a sufficient interval of time).

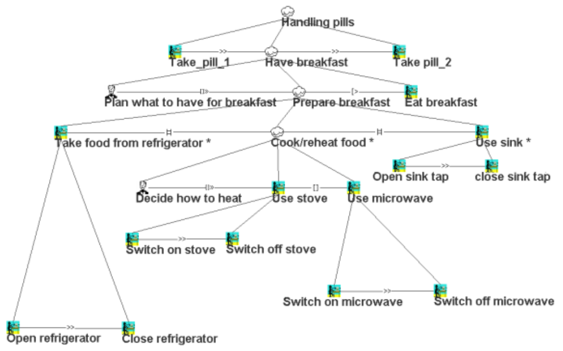
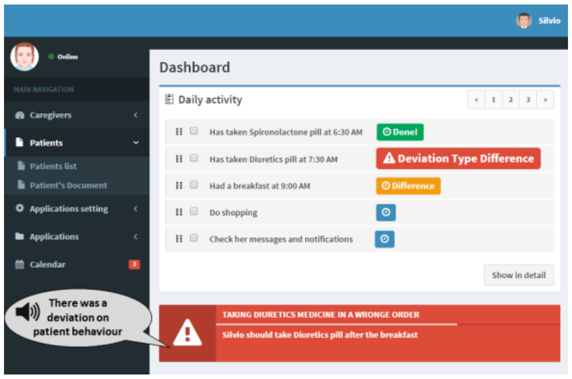


Figure 4: The task model of the example

We developed a Web application for caregivers who can monitor the activities of the patients in their daily life. As explained before, at the start up the application subscribes to the Behaviour Analysis module in order to be notified when a deviation occurs. If the senior takes the second medicine before having breakfast the Behaviour Analysis module identifies a deviation of type “Difference” and then it retrieves the action related to this deviation. The action has been defined by the caregiver and in this case is: Send an alarm with the text “The user should take the Diuretics pill after the breakfast”. The Action Generation module will also consider the rule which takes into account the current noise level in the environment where the caregiver accesses the application; in this case the noise level is quite low therefore the action will be applied in a multimodal way (see Figure 5) in order to better attract her attention.



Has taken Spironolactone pill at 6:30 AM — Done!
Has taken Diuretics pill at 7:30 AM — Deviation Type Difference
Had a breakfast at 9:00 AM — Difference
Do shopping
Check her messages and notifications

TAKING DIURETICS MEDICINE IN A WRONGE ORDER
Silvio should take Diuretics pill after the breakfast

Figure 5: The effect of the action sent to the application after the deviation

CONCLUSIONS

We have presented a method to detect abnormal elderly behaviour, which is important in AAL scenarios because these anomalies may represent early signs of the onset of health-related issues. Our solution exploits task model specifications and compares the expected behaviour expressed in such models with sequences of events generated by actual user behaviour. Thus, the proposed solution is able to detect significant deviations, and then derive accordingly suitable actions addressing the identified anomaly. The approach has been applied to a case study in the AAL domain of elderly independently living at home. However, it can also be applied to other situations in which deviations from the expected users’ behaviour in their daily activities can have potential risky consequences. For instance, this is the case of settings in which workers perform their job in hazardous environments, such as construction sites. In future work, we plan to extend the analysis by considering other task-related aspects/attributes too (e.g. task time and duration, and objects manipulated by the tasks), and to empirically validate the type of deviations detected and actions generated with this approach[1].

REFERENCES

1. U.A.B.U.A. Bakar, Hemant Ghayvat, S.F. Hasanm and S.C. Mukhopadhyay.2016. “Activity and Anomaly Detection in Smart Home: A Survey.” Springer International, in Next Generation Sensors and Systems, Smart Sensors, Measurement and Instrumentation
2. Vikramaditya Jakkula, Diane J. Cook. 2008. "Anomaly detection using temporal data mining in a smart home environment." Methods of information in medicine 47.1: 70-75.
3. Dorothy N. Monekosso, Paolo Remagnino. 2010. "Behaviour analysis for assisted living." IEEE Transactions on Automation science and Engineering 7.4: 879-886.
4. Giulio Mori, Fabio Paternò, Carmen Santoro . 2002. “CTTE: Support for Developing and Analyzing Task Models for Interactive System Design.” IEEE Trans. Software Eng. 28(8): 797-813
5. Qin Ni, Ana Bel n Garc a Hernando, Iv n Pau de la Cruz. 2015. “The Elderly’s Independent Living in Smart Homes: A Characterization of Activities and Sensing Infrastructure Survey to Facilitate Services Development.” Ed. Panicos Kyriacou. Sensors,15.5: 11312 11362. PMC.
6. Laila Paganelli, Fabio Paternò .2003. “Tools for remote usability evaluation of Web applications through browser logs and task models.” Behaviour Research Methods, Instruments, & Computers. August 2003, Volume 35, Issue 3, pp 369 378.
7. Martha E. Pollack, et al. 2003. "Autominder: An intelligent cognitive orthotic system for people with memory impairment." Robotics and Autonomous Systems 44.3: 273-282.

[1] This work has been partially funded by the AAL Project PersonAAL (http://www.personaal-project.eu)