A Reactive and Proactive Approach for Ambient Intelligence

Alencar Machado[1,2], Daniel Lichtnow[2], Ana Marilza Pernas[3],
Leandro Krug Wives[1] and José Palazzo Moreira de Oliveira[1]
[1]*PPGC Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil*
[2]*Colégio Politécnico, Universidade Federal de Santa Maria (UFSM), Santa Maria, Brazil*
[3]*Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas (UFPEL), Pelotas, Brazil*

Keywords: Proactive, Reactive, Situation, Context, Event, Ambient Intelligence.

Abstract: Ambient Intelligence provides technology support and assistance to help people in their daily wellbeing. Equipped with ubiquitous technologies, Ambient Intelligence uses sensors to monitor the environment and to collect data continuously providing systems with updated information. Ideally, these computer-supported environments must detect relevant events to forecast future situations and to act proactively to mitigate or eliminate undesired situations while regarding user's specific needs. To build a system with reactive and proactive characteristics in Ambient Intelligence, it is important to allow it to be extensible, predictive and to incorporate decision-making capabilities. In this sense, the objective of this work is to propose an approach for providing reactive and proactive behavior in Ambient Intelligence systems. More specifically, we want to provide Situation as a Service in Ambient Assisted Living. In the present work, we illustrate practical aspects of the system's architecture by describing a home-care scenario in which the system is able to understand the behavior of the user, as the time goes by, and detect relevant (dangerous) situations in order to act reactively and proactively and help users manage their health condition.

1 INTRODUCTION

In the future it is expected that Ambient Intelligence (AmI) will enable environments to support people, being sensitive to their needs and capable of anticipating behaviors (Sadri, 2011). Fields such as Ambient Assisted Living (AAL) (Jara, Zamora and Skarmeta, 2011) and Smart Homes (Chan et al., 2008) are emerging as AmI focused on specific characteristics of the user.

Currently, there is a lack of support for extensible Ambient Intelligence systems to incorporate reactive and proactive behavior. In addition, these systems must (a) manage heterogeneous sources (sensors and appliances) to provide high level information such as situations; (b) process events for detecting situations in the environment; (c) make predictions of unwanted situations and to react in advance; (d) determine the policy of actions to consume appropriate services for adapting the environment ahead the situation envisaged; (e) have expansive capacities to manipulate different situations.

Among these challenges, this paper focusses on how to process events to detect and predict future situations. In this sense, we argue that for fulfilling user needs, AmI systems should be reactive and proactive. Thus, these systems must be aware of the user current situation and foresee future situations. The system must make decisions in advance, taking into account evidences that demonstrate the possibility of an unwanted situation happening in the future.

In our approach, the user and his actions are monitored through sensors that capture environmental data. This data is used to characterize the user context, using entities for obtaining a semantic characterization that determines the state of the environment. In the proposed approach, the state of the environment is called "situation".

Thus, when a situation is detected, if necessary, it is possible to act reactively and proactively on the environment, using capabilities (services) provided by electronically controlled devices, seeking to adapt automatically the environment according to the situation envisaged.

In our approach the actions of the system are achieved by using functionalities implemented by Web services embedded in physical objects such as mobile phones, televisions and radios.

Machado A., Lichtnow D., Marilza Pernas A., Krug Wives L. and Palazzo Moreira de Oliveira J..
A Reactive and Proactive Approach for Ambient Intelligence.
DOI: 10.5220/0004884205010512
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 501-512
ISBN: 978-989-758-028-4

This work focuses on unwanted situations involving the lives of elderly people at home. In this context, systems that foresee and handle unwanted situations proactively can assist caregivers of users who do not have physical or cognitive conditions for managing their own health, because it is necessary to act before an unwanted situation occurs.

Thus, the approach was applied to a specific case-based application for managing medicines. In this case, citizens self-manage their health, and an application assists this activity. Over time, the citizens are affected by cognitive decline, when they become unable to manager their medications and the system must adapt itself to assist caregivers. The aim is to analyze the user behavior for predicting the need of some early action to aid them to take their medications at the right time, preventing a decrease in their health condition.

This paper is organized as follows. Section 2 discusses background and related work. Section 3 presents the proposed reactive and proactive approach. In Section 4 and 5, presents the case study developed. Finally, in Section 6, we present and discuss our conclusions and future works.

2 BACKGROUND AND RELATED WORK

Ambient intelligence systems need to know the world around users they monitor and, in order to perform actions, they need to interact with users through the devices that surround them (Augusto, Nakashima and Aghajan, 2009). Therefore, intelligence systems must be context-aware and proactive, automatically adapting to changes in the environment and considering user needs, without requiring their personal attention.

Regarding to context-aware systems, the concept of *context* must be defined. We adopt the definition of Ye, Dobson and McKeever (2011), in which context is seen as "the environment in which the system operates". Dey and Abowd (1999), however, characterize *context* as the situation of an entity in an environment. In the present work, the context of an environment is thus represented by a set of entities that surround or interact with the user, and their semantic relations.

As the time goes by, different entities or interactions may be active. In this sense, we need to verify the current contextual state of the user and act upon it or on its changes. We thus need to define the concept of *situation*, since it is used by us to characterize the state of the user environment. For instance, Ye, Stevenson and Dobson (2011) define *situation* as the abstraction of the events occurring in the real world that are derived from the context and hypotheses about how the observed context relates to factors of interest.

In this sense, applications that deal with situations are called situation-aware. Awareness implies vigilance in observing or alertness in drawing inference from a previous experience, so something is aware only if it is able to observe some object and design conclusions through previous observations (Kokar, Matheus and Baclawski, 2009). Observations could be made by services provided through devices, such as sensors. Therefore, by these observations, it is possible to detected events that change the state of the environment, characterizing thus a situation.

Another important concept is the one of *event*. According to Etzion and Niblet (2010), an *event* is an occurrence within a particular system or domain, it is something that has happened, or is contemplated as having happened in that domain. Events can be modeled as raw and derived. Derived events are higher-level events in the semantic hierarchy. It normally corresponds to a pattern of observation. Raw events are produced by some entity of context (e.g., sensor). Events can change the state of the environment, therefore producing new situations.

Works such as SOPRANO (Sixsmith et al., 2009), PERSONA (Tazari et al., 2010), among others, aim at modeling context, events and situations in middleware systems to provide a platform of health services in AAL. They propose conceptual models to transform homes into AAL environments, modeling their context and services (Paganelli and Giuli, 2011). SOPRANO, for instance, has the intention of recognizing facts, objects, and people surrounding users allowing systems to act more appropriately and providing support to daily activities. However, in these works, the system is reactive, since it only acts after the evidence that an unwanted situation has occurred. They are not able to provide proactive actions to mitigate or eliminate undesired situations in advance.

The term proactive computing was first described by Tennehouse (2000), who proposed the following principles for proactive systems: they should be closely connected with their surrounding world; they should also deliver results to humans before the user action; and they must operate autonomously.

The characteristics proposed by Tennehouse turn systems essentially reactive. In this sense, proactive

computing overlaps with the term autonomic computing (Want, Pering and Tennenhouse, 2003). An autonomic system is one that reacts to events that already happened. Our approach follows the proactivity definition of Engel and Etzion (2011) who describe proactivity in computing systems as the ability to mitigate or eliminate undesired future situations, identifying and taking advantage of future opportunities by applying prediction and automated decision making technologies. Thus, the aim of the system actions is to prevent future unwanted situations. One example is the work of Fu and Xu (2010) where event correlations are used for predicting future failures in networked computing systems.

The current vision of proactive behavior is listed as the next phase in the evolution of complex event processing (Etzion and Niblett, 2010). Thus, Engel and Etzion (2011) and Engel, Etzion and Feldman (2012) present the Proactive Event-Driven Computing, proposing the extension of the event processing conceptual model and including more two types of agents to the architecture of proactive event-driven applications: predictive and proactive.

Proactive systems apply prediction methods for predicting future information and decision making. Boytsov and Zaslavsky (2011), for instance, analyzes and compares prediction methods in order to identify their benefits and shortcomings. Among these methods, they describe Bayesian networks as an appropriate approach for predictive models. Similarly, Nazerfard and Cook (2012) present a sequence-based activity prediction approach that uses Bayesian networks in a two-step process to predict both activities and their corresponding features.

Lotfi et al. (2012) seek to make prediction of abnormal behavior of elderly with partial dementia. They use sensor data for identifying anomalous sensor behaviors to predict the future values of the possible data for each sensor. The predicted values are used to inform a caregiver in the case of anomalous behavior of sensors in the near future.

To the best of our knowledge, works related to smart environments propose strictly reactive systems. We have seen some researches describe proactive behavior to anticipate user's actions, but reacting only after a situation has happened. For instance, a system for handling situations of agitations for elderly patients who take actions to anticipate actions of the caregivers. They, however, do not seek to identify this situation in advance to avoid it happening. Besides, in general these proposals do not address or include extensibility technologies, i.e., are not able to handle different situations in the course of time.

3 A REACTIVE AND PROACTIVE APPROACH

Our approach differs from other works because we present a new reactive and proactive approach that is more appropriate to attend the proper demands of AAL systems. Besides, it provides extensibility for residential smart environments. The approach is explained taking into account the recent history of self-management of a citizen's health, where the system triggers reactive and proactive actions.

The extensibility aspect of our approach is related to the concept of pervasive applications, and is based in a work named Situation as a Service (SIaaS) (Machado et al., 2013), which is described in the next section. In the present work, we have added temporal aspects, prediction and decision making techniques in order to prevent the existence of unwanted situations in future (Section 3.2).

3.1 Situation as a Service

In our approach, pervasive applications are installed in a middleware named SItuation as a Service (SIaaS). Pervasive applications (appPerv) are software applications developed by companies specialized in specific fields, like health, surveillance, energy. Designers of appPerv must implement in a conceptual model that corresponds to specific situations of the environment that are relevant to the appPerv. They also must inform the appPerv context of interest, generating instances of a particular type (e.g., Patient, Sensor) and make the linking semantics among them (as *hasSensor*).

Therefore, the pervasive application informs the SIaaS middleware about the situations that are important and should be managed for the detection and prediction of situations, as well as a set of contextual information necessary for decision making.

In this work, we are interested in a specific domain: home-care health support. Thus, user could extend the capabilities of the middleware buying a complete solution for managing chronic diseases or only one pervasive application for managing the schedules of their medicines. For example, the pervasive applications described in (Machado et al., 2013) performs reactive actions (consumption of services) when a patient's agitation situation becomes true.

The SIaaS manages the environment and provides the context of interest, as well as the situation of interest for pervasive applications, so it is possible decide the more appropriate action when a situation is detected. It is presented in Figure 1.

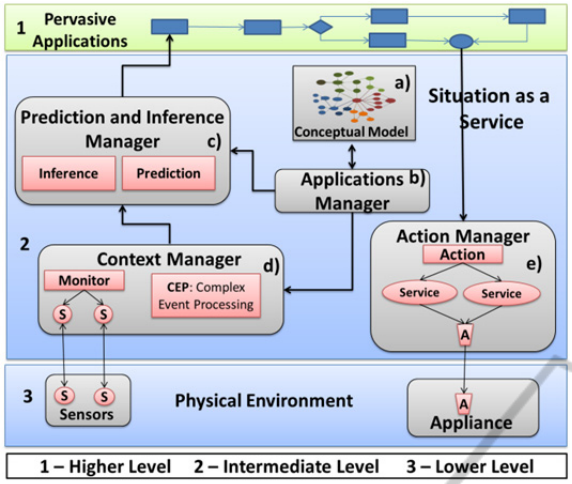

Figure 1: Levels of the architecture.

The SIaaS is a Service-oriented architecture with complex event processing that is implemented in three levels: Lower, Intermediate, and Upper. The Lower Level (3) corresponds to the physical environment (i.e., sensors and appliances).
The Intermediate Level (2) contains the middleware for providing a stable and secure environment for pervasive applications. It notifies any pervasive application whenever its situations of interest become true. It comprises a conceptual model, an application manager, a prediction and inference manager, a context manager, and a proactive actions manager. The Upper Level (1) offers a stable computing environment for pervasive applications.

3.2 Proactive Model

A pervasive application has interest in specific situations that involve users and what happens in their living environment. This pervasive application has the knowledge of what kind of reactive actions should be taken when an unwanted situation happens. The SIaaS middleware monitors the events that could generate a situation of interest for any pervasive application, and notifies it whenever such situation becomes true. These events occur in time windows and are related to situations. These situations are described by pervasive applications that specify their situations of interest.

Taking into account examples from the health domain, we consider a scenario where the aim is to monitor medicine administration. Thus, if the event "medicine X is not administrated" is detected during consecutive days, it may result in health problems for the patient, i.e., an unwanted situation.

The SIaaS must avoid such unwanted situations (e.g., the patient forgot to take his medicine and became sicker). For this purpose, the SIaaS initially learns a predictive model using data provided by the pervasive application. The pervasive application data consists on patient's behavior patterns that will be managed by the SIaaS. After the learning stage, the SIaaS can predict situations through events detected in real-time and is able to perform proactive actions, avoiding the occurrence of unwanted situation. Consequently, the reactive actions requested by the pervasive applications will not be executed, because the events (e.g., patient forgot to take his medicine) that would determine the situation will not occur. In this sense, the SIaaS acts proactively to prevent an unwanted situation.

As depicted in Figure 2, the environment may be in two states: controlled or uncontrolled. An event stream, predefined as normal, characterize a controlled environment where reactive actions are helpful. However, if the events are being detected outside this predefined state, it could characterize that the environment will become uncontrolled, thus increasing the dependency on proactive actions being performed by the SIaaS.

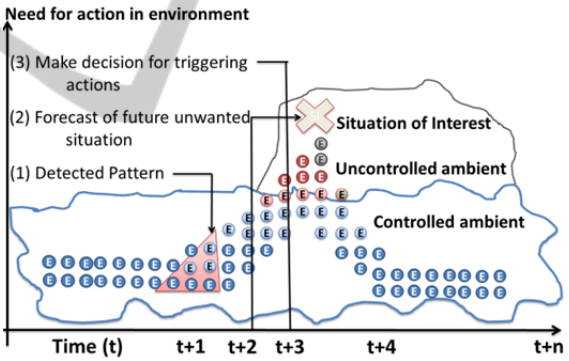

Figure 2: Environmental states.

In the environment, event streams are constantly monitored through data made available by sensors. Still, evaluating the events flow of Figure 2, at t+1 a pattern of events (provided by the pervasive application) is detected (1) by the Context Manager. This subsystem uses a prediction algorithm to determinate the probability of an unwanted situation becoming true in the user living environment. The Context Manager has t+2 times to make the prediction, having enough time to take corrective actions. After identifying the probability of the occurrence of a situation in t+2 (2), the Inference Manager process the respective rules to determinate if the rate of probability is relevant. If it is positive, the Proactive Actions Manager must be activated in t+3 (3) to

trigger proactive actions. This module is responsible for choosing the most appropriate policy to consume appropriate services corresponding to environmental devices and health care providers.

All these actions are taken in order to bring the environment to a normal state and to avoid future unwanted situations. The goal is to return to the initial streams of events, thereby characterizing the consistence of the environment.

3.3 Sliding Windows

In the present approach, we consider that it is necessary to analyze a sequence of events for learning and detecting patterns aiming to predict situations. Besides, we consider that this sequence of events occurs in a period of time, referenced as Sliding Window, i.e., a valid space of time where situations are predicted or detected and the relevant decisions are taken. The Sliding Windows model used in our approach is adapted from Salfner, Lenk and Malek (2010). In this model, a sliding can be sized or timed. A sized sliding window (SSW) has a specific size that corresponds to the number of events of a given pattern of interest. For instance, is possible to use a SSW with the last hundred events that match specific selection criteria. A timed sliding window (TSW) has a finite time frame where events of interest are monitored. In this work, only Δ***td*** is modeled as these two possibilities, the remainders are windows of time, because they are used to model the window in future for an unwanted situation.

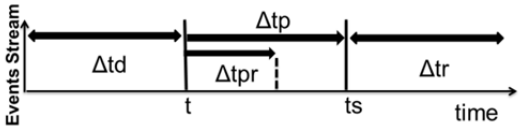

Figure 3: Sliding Windows.

Figure 3 presents a sliding window associated with a real time proactive behavior. At time *t*, the possibility of occurrence of a situation can be predicted with some time in advance. This period of time is called prediction time (Δ***tpr)*** and is based on the events currently detected in the environment. Situations are predicted in Δ***tpr,*** which uses a size or time sliding window (Δ***td***) that corresponds to the event stream monitored by the system. These timed or sized windows (Δ***td***) are used to perform prediction. We assume that proactive actions are valid for some period of time, named period of proactivity (Δ***tp***). In this time window, triggered actions can change predicted situations, which are expected to occur in the reaction time (Δ***tr)*****.** If Δ***tpr*** > Δ***tp*** then there will not be enough time for all proactive actions to be triggered before the predicted situation (i.e., an unwanted situation) becoming true. Thus, Δ***tr*** is the maximum time the system has to react, since Δ***tr*** is the time estimated to the situation to occur. Then, Δ***tr*** is the period where reactive actions, related to a specific unwanted situation, are triggered.

3.4 Event and Situation Model

In this work, the semantic relations {*R*} that form the context are represented by triples ‹*Es, p, Eo*› where the subject *Es* and the object *Eo* represent instances of entities of the environment, which could belong to the same domain or not. Similarly, as formalized by Ye, Stevenson and Dobson (2011), *p* represents a context predicate that encapsulates two entities of context in a relation. For instance, in the relation ‹John, *hasSensor*, RFID› John and RFID represent entities. Subjects and objects can also be represented by variables in reasoning rules. For example, in the following triple, ‹*x*, *hasSensor*, *y*›, *x* represents any entity instantiated in the user domain and *y* represents any entity instantiated in the sensor domain. In this example, any pair of values of User and Sensor, related by the relationship `hasSensor`, can validate this context predicate.

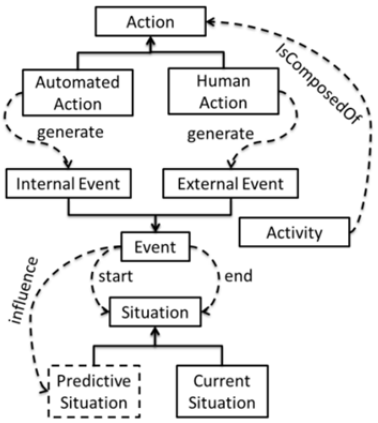

Figure 4: Conceptual model.

Figure 4 depicts the conceptual model of our approach. The Activity entity represents daily activities performed by the citizen in his home, like breakfasting, watching television, taking medicine or doing exercises. The activities are made up of human actions, for instance the activity to take medicine is composed by picking up a glass with water and taking the drug, represented by the semantic relations presented below (1).

```
‹John, catch, glass› ∧ ‹glass, has,
  water› ∧ ‹John, obtain, drug› ∧
       ‹John, take, drug› ⇒
        ‹John, took, medicine›
```
(1)

Besides human actions, we also take into account automated actions taken by the system. For instance, the system may notify the citizen through an audible warning using some device (we consider that automated actions are actions performed by devices). Thus, actions are carried out by an agent (human or device) in order to achieve a goal. When an action either achieves or not its goal, it can generate events.

In our approach, we take into account internal and external events, which are defined in the theory of Situation Calculus (Mccarthy, 2002). An external event is generated externally by human actions or by interacting with pervasive applications. In addition, external events can be generated by changing a user device or by changing the network connection used by a device. Internal events are generated internally by the system and are represented by assertions that can be given by an axiom. An internal event could be generated by an automated action, the detection of successive state variable changes, or by the modification of one specific state variable. Thus, as more human actions are transferred to automated actions, more extensible the system becomes. Events are represented by the following syntax (2).

$$\text{Event: } (name, type, time, \{R\}, p) \quad (2)$$

An event has a *name* and is characterized by a *type* (internal or external), a time (timestamp) within $\Delta\boldsymbol{td}$ windows and a set of contextual semantic relations $\{R\}$. When the event is not produced by a single entity (e.g., raw data sensor), it may also have a detection pattern (p). Events can be linked to one or more contexts; for instance, a pattern that defines that an event must be detected if a specific sequence of events happens within a given sliding window of time or size involving the "user" in his/her living room. In this work, events can determine the evidence of the beginning and the ending of a situation. Thus, events change the state of the environment and characterize a new situation. The current situation is represented by the following syntax (3).

$$Cs: (name, Ie, \{a\}, Fe) \quad (3)$$

As shown in (2), the current situation (Cs) has a *name* and a set of *events* that characterize its beginning (Ie) and ending (Fe), and the *time* attribute of these events that characterize the valid time window of this situation, which will always be in $\Delta\boldsymbol{tr}$. In addition, the current situation has a set of triggered reactive actions $\{a\}$ that were detected during a valid time for handling the current situation. For instance, below we present how to represent an event (Ie) that initiates the "unmedicated" situation (4), its corresponding final event (Fe) and the actions to be performed in this situation.

```
name   unmedicated                                          (4)
Ie     ‹John, shouldTake, Drug X› ∧
       ‹Medicine X, timeToAdminister, 10h›
       ∧ ‹currentTime, equals, 10:30›
       ⇒‹John, notTake, medicine›

       ⇒‹John,isSituationOf,{unmedicated}›
{a}    ‹System, trigger, audibleWarning›
       ‹System, trigger, visualWarning›
       ‹System, notify, Caregiver›

Fe     ‹John, catch, glass› ∧ ‹glass, has,
       water› ∧ ‹John, obtain, drug› ∧
       ‹John, take, drug› ∧ ‹John, on-
       Click, appPerv› ⇒ ‹John, take,
       medicine›
       ⇒‹John, isSituationOf, {medicated}›
```

The event evaluation can lead the system to find out that an unwanted situation has a probability of happening in the future. In (5) we show that a Predictive Situation (Ps) is characterized by a set of *events*; a set of patterns (p), which describes some form of correlation among events that shape this situation, the probability value (pr) of its occurrence in a context in the future; and a timestamp ($time$) during which it may occur within $\Delta\boldsymbol{tr}$.

$$Ps: (name, \{event\}, \{p\}, pr, time) \quad (5)$$

For example, if a sensor detects smoke in a house and another one detects a gas leak, then it will be characterized as a dangerous situation. This demonstrates how occurring events can influence the probability of a predicted situation (the prediction of a situation is addressed in the next section). Thus, proactive actions are needed to react to the possibility of a future unwanted situation. In the next section we show the model for performing reactive and proactive prediction for deciding what action must be triggered in such cases.

3.5 Prediction for Decision Making

Our predictive model is hybrid, since we intend to use an inference engine to process inference rules and infer that a current situation is occurring and a Bayesian network to determinate the probability of situations occurring in the near future.

In this sense, the Bayesian network is used in order to estimate the probability of a situation occurring in the future. To determine whether a probability is relevant, predetermined rules (which are not processed through inference), provided by pervasive

applications, are needed, thus determining whether some probability is relevant for activating the Proactive Actions Manager.

However, to build the Bayesian network, it is necessary the knowledge of an expert in the application domain to indicate which events can produce a situation. In this sense it is assumed that the pervasive application is provided with rules defined previously by an expert in the specific domain of health.

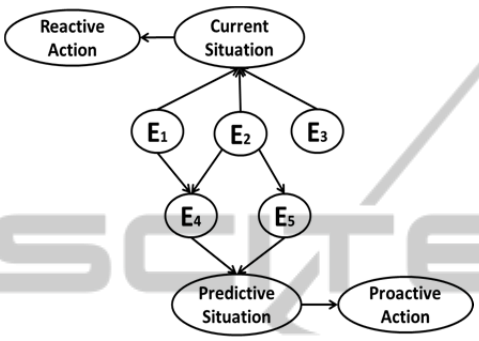

Figure 5: Proactive and Reactive Network.

Figure 5 shows the structure of the Bayesian network developed in our approach. In this model, we consider that the cause (event) precedes the effect (situation), the events are independent variables of each other, and a situation is represented in a leaf node.

This network has two facets, reactive and proactive. In the reactive approach, the presence of events is the cause for the evidence of the current situation. In this case, if all of the events that characterize a situation are detected in the environment, we can process inference rules to detect if the situation is ongoing, and thus requires reactive actions.

For instance, in (6), if events *e*1, *e*2 and *e*3 were modeled on the network as the precedents of the situation *s1* and these events where detected in this sequence, it would be possible to infer (reactive side) that *s1* is the current situation.

```
      IF sequence ‹e1, e2, e3›
⇒‹John, isSituationOf,{medicated}›        (6)
```

In the proactive approach, the presence of these events is used by the Bayesian network to calculate the probabilities of the predictive situation. Consequently, if this probability characterizes a future unwanted situation, proactive actions are needed. These values are calculated by $Pr(Ps|pa(Ps))$, which indicates the conditional probability of a predictive situation (Ps) occurring due to its relationship with the parent events $pa(Ps)$.

In our approach, every pervasive application installed on the SIaaS describe (through OWL files) a set of relevant events and how these events influence (qualitative part of the network Bayesian) each situation. Wok such as Lukasiewicz and Straccia (2008) detail how to model Bayesian networks through Semantic Web technologies. Therefore, the SIaaS builds a network to such pervasive application, isolating this network from others to be able to manage the specific network model that is of interest for a given application.

However, to avoid the system necessity of activating the reactive model some time before the network computes the probability of each event (quantitative part) for the predictive situation, we assume that the pervasive application provides data for the initialization of the network through a supervised learning process.

After the learning phase, the network enters into a production state and is dynamically updated with information about the events detected by the system. Two process are constantly running on the network: belief update and belief revision. The belief update is the upgrade of the network due to updated events, thus we update the Conditional Probability Table (CPT) of the network, whereas belief revision makes a belief assessment query, referring to updating the probability value within the predictive situation. Beyond that, the rules (not processed by inference) are constantly processed to identify if the probability value calculated by the network is relevant for a predictive situation. According to that identification, proactive actions are triggered. An example of a proactive action rule is presented in (7).

```
IF ‹predictiveSituation, greatherThan,
86%› ⇒ ‹emergencySituation, in, 10min›     (7)
```

Rules identify the specific time in which a predictive situation is expected to occur within Δtr. Such rules describe complex temporal predicates based on what the processing is carried out. These rules are associated with a set of actions. These actions, when consequently fired, result in new internal events (as described in Section 3.4), which feed into the network again and update the attribute probability of the predictive situation. Thus, through the set of actions, the SIaaS has the purpose of eliminating the occurrence of unwanted events (i.e., the cause of the situation) and consequently decreasing the probability value of the predictive situation.

The decision to perform an action (i.e., the actions policies) depends on the detected situation (current or future). The actions policies are provided together with their corresponding pervasive applications for SIaaS. These policies are defined by the developers of the pervasive applications with the

help of health experts. Experts help is necessary because these pervasive applications are related to the domain of health.

This is the technique used to generate the cost of taking the action changing over time. When unwanted events stop being detected, we characterize those actions as having the desired effect, thus acting as a reward value.

4 CASE STUDY SCENARIO

This case study aims to demonstrate the use of our approach in a scenario where the necessity of a mechanism that acts in proactive way is emphasized.

The scenario is related to the aforementioned self-administration of medicines by patients in their homes. The aim is to identify when patients are no longer able to control their own medication.

In this sense, we are considering that patients, in general, will take their medicine in a stipulated period of time or after an action system (e.g., using functionalities (services) of devices of the environment to remember the patient about the medication). However, some patients can have cognitive decline over time, compromising the self-management of their health condition, thus needing help to take the medication at the correct time**.** In this scenario, the situation known as "*unmedicated*" becomes habitual, and reactive/proactive actions are necessary to control this situation, considering the recent history of self-management of citizen's health, thus assisting the patient to take his medicine in the best possible way. We considered a pervasive application deployed in the SIaaS to help patients in the described scenario. For the implementation, we have used the monitoring component Esper (2010), the Bayesian tool Netica (2013), which provided an API that we used to create the Bayesian network that was inserted into the Context Manager. In addition Jess (Friedman-Hill, 2003) was used to build Java software process able to perform inference rules in the Inference Manager module.

For the case study, the following fictitious scenario was considered for describing the approach supported by the decision making process implemented by the pervasive application. Imagine 'Ms. Smith', a 70 years old citizen who has some aging associated diseases such as diabetes, hypertension and lightweight dementia. Ms. Smith's home is an intelligent environment managed by a SIaaS middleware, where a number of pervasive applications are installed. An example is the *pervasive application for managing medications* (*appPervMed*).

Ms. Smith initially controls the medication herself. However, as any ordinary person, sometimes, to be involved in some particular activity, she forgets to take or takes her medicines late, which puts her in an "*unmedicated*" situation. In these cases, the *appPervMed* requests to the SIaaS middleware to trigger an audible or visual warning through devices located near Ms. Smith. Whenever a warning reaches her attention, she can interact with the system through a smart phone or smart TV to report explicitly that she took her medication. After that interaction, the system will close (finish) the "*unmedicated"* situation.

After some stipulated time, if Ms. Smith does not take her medication, the system sends a warning to her caregiver. It warns her caregiver that Ms. Smith had not taken her medicine, thus placing the responsibility of interacting with the SIaaS on the caregiver. Once the caregiver gives her the medicine, and informs the system about that, the *appPervMed* will know that Ms. Smith took the medicine and will determine the end of the "*unmedicated*" situation.

Eventually, the caregiver himself may forget or may be not close to any device that could warn him about the moment that Ms. Smith must be medicated. Thus, the event "medicine X not taken" is detected, corresponding again to the beginning of the situation of "*Ms. Smith is unmedicated*". Audible and visual warnings are generated in different moments in the environment, and, after some parameterized time, the caregiver is warned. The *appPervMed* waits for a notification that Ms. Smith took the medication by the caregiver. If it is not notified in a specific period, the *appPervMed* triggers a warning directly to the healthcare provider (consuming a specific Web service), placing the responsibility on the healthcare provider to make Ms. Smith taking her medicine, and, once taken, it ends the situation.

As explained, alerting the caregiver is an exception. However, after some time, if Ms. Smith takes her medicine only after a system warning to her caregiver and if this behavior becomes more usual, this behavior may indicate a cognitive decline of Ms. Smith.

Thus, it is necessary to identify (in a proactive way) when the cognitive impairment happens, because if this identification does not happens fast, the treatment will be harmed by the administration of drugs in wrong times. This moment may characterize the end of the patient's ability to medicate herself, requiring the caregiver to this function. Therefore, the system must adapt itself and assist the caregiver in his task of assisting Ms. Smith.

5 APPLYING THE APPROACH TO THE CASE STUDY

The scenario described before shows that a pervasive application must react to an event (Ms. Smith did not take her medication) that characterize the unmedicated situation and also must forecast some situation (e.g., Ms. Smith will not take her medication without her caregiver help) and be proactive.

Next, following our approach, we show how a pervasive application should work either reactively and proactively.

5.1 Reactive Behavior

The events related to this case study that are relevant to the *appPervMed* are described here, as proposed in Section 3.4. For performing this task we used the Semantic Web Rule Language (SWRL), using the already defined semantic relations {*R*} and triplets in the form *‹Es, p, Eo›*, as presented in Section 3.4. The patterns to detect events were modeled as *Esper* statements. The variables were replaced to the values used in the scenario to provide an easier interpretation.

Event name: ea1
Description: Not took the medicine
Typed: Internal
{R}: ‹Ms. Smith, needToTake, Drug X› ∧ ‹Drug X, timeToAdminister, 10h› ∧ ‹currentTime, is, 10:15› ⇒‹Ms. Smith, notTaken, medicine›

Pattern
```
SELECT e FROM PATTERN[
every e=Event(name ='ea1') ->
(timer:at(*/15,10:00,*,*,*)
and not Event(name ='ea2'))]
```

In this rule, `timer:at` is an expression of a specific time that turns true. The syntax is timer:at (minutes, hours, days of month, months, days of week, seconds) (Esper, 2010).

Event name: ea2
Description: Took medicine
Typed: External
{R}: ‹Ms. Smith, medicationTime, Drug X› ∧ ‹doorMedicineChest, wasOpenedBy, Ms. Smith› ∧ ‹Ms. Smith, pressedOKButton, appPervMed›⇒‹ Ms. Smith, took, medicine›
Pattern: `SELECT e From e=Event(name='ea2')`

Event name: ea3
Description: Took medicine after some action
Typed: External
{R}: the same semantic relations of ea2

Pattern
```
SELECT e FROM PATTERN[
every e=Event(name ='ea3')->
(Event(name ='ea1') ->
 (Event(name = 'a1') or
  Event(name = 'a2') or
  Event(name = 'a3') or
  Event(name = 'a4')))]
```

Event name: a1
Description: Audible warning after 10 min of ea1
Typed: Internal
{R}: ‹appPervMed, trigger, audibleService›

Pattern
```
SELECT e FROM PATTERN[
every e=Event(name='a1')->
(timer:at(*/25,10:00,*,*,*)
and Event(name ='ea1'))]
```

Event name: a2
Description: Visual warning after 25 min of ea1
Typed: Internal
{R}: ‹appPervMed, trigger, visualService›

Pattern
```
SELECT e FROM PATTERN[
every e=Event(name ='a2') ->
(timer:at(*/40,10:00,*,*,*)
and Event(name ='ea1'))]
```

Event name: a3
Description: Notify caregiver after 45 min of ea1
Typed: Internal
{R}: ‹appPervMed, notify, Caregiver›

Pattern
```
SELECT e FROM PATTERN[
every e=Event(name ='a3') ->
(timer:at(*,11:00,*,*,*)
and Event(name ='ea1'))]
```

Event name: a4
Description: Notify health provider after 2h of ea1
Typed: Internal
{R}: ‹appPervMed, notify, HealthProvider›

Pattern
```
SELECT e FROM PATTERN[
every e=Event(name ='a4') ->
(timer:at(*/10,12:00,*,*,*)
and Event(name ='ea1'))]
```

As presented above, if event *ea2* is detected, Ms. Smith is in the situation of "*medicated*". Therefore, the *appPervMed* is not started because the situation "*unmedicated*" did not happen. The internal and external events and relevant actions that determine each situation are presented below:

Current Situation	
name = unmedicated;	Name = medicated;
Ie = ea1	Ie = ea2
{a} = a1, a2, a3, a4	{a} = -
Fe = ea3	Fe = ea1

If Ms. Smith does not take the medicine (event *ea1* was detected) the "*unmedicated*" situation is initiated and *appPervMed* chooses the reactive action to be triggered. After the detection of the event *ea1*, *appPervMed* waits for 10 minutes and, if an event *ea2* was not detected, it will trigger an audible warning (*a1*). After the audible warning, the application needs to wait for a feedback. If this feedback is not received, *appPervMed* triggers a2 to produce some visual warning. Thus, *appPervMed* terminate its execution when the resulting feedback is ea3 (i.e., took medicine after some action).

5.2 Proactive Behavior

This section presents the proactive behavior of the SIaaS, showing how it would preventing Ms. Smith from entering in an "*unmedicated*" situation. Initially, it makes a historical analysis of the situations generated by the events detected that are relevant for *appPervMed*.

Below, we present a historical situation (HS) that has happened and the reactive actions triggered for manipulating this situation.

$$HS(unmedicated)_{10:23}^{10:15} = \{a1\} \quad (1)$$

$$HS(\ medicated\ )_{10:00}^{10:00} = \{-\} \quad (2)$$

$$HS(unmedicated)_{10:33}^{10:15} = \{a1, a2\} \quad (3)$$

$$HS(unmedicated)_{10:33}^{10:15} = \{a1, a2\} \quad (4)$$

$$HS(unmedicated)_{11:03}^{10:15} = \{a1, a2, a3\} \quad (5)$$

$$HS(unmedicated)_{11:03}^{10:15} = \{a1, a2, a3\} \quad (6)$$

$$\ldots \quad (n)$$

$$HS(unmedicated)_{11:03}^{10:15} = \{a1, a2, a3\} \quad (n+1)$$

$$HS(unmedicated)_{12:10}^{10:15} = \{a1, a2, a3, a4\} \quad (n+2)$$

$$HS(\ medicated\ )_{10:00}^{10:00} = \{-\} \quad (n+3)$$

This representation is based in a notation proposed by Wasserkrug, Gal and Etzion (2005), which shows the initial and final time of the situation. In this case, curly brackets represent the actions triggered during the period of time when the situation was valid. For instance, the first event *ea1* was detected at 10:15 (1), and the situation was finalized when the event *ea3* was detected at 10:23 and the action *a1* was triggered. In (2), Ms. Smith took the medicine on the right time (10:00), so, no reactive action was triggered.

This historical data of behavioral management of medicines by Ms. Smith shows the sequence of actions that were needed to handle the unwanted "*unmedicated*" situation. This HS is used to generate the Conditional Probability Table (CPT) for each node event of the Bayesian network. In this case study, the generation of the CPT must be sensible to a cognitive decline, so the Bayesian network cannot be established with all the stored history of that situation. In order to identify a cognitive decline, the system needs to build a valid sliding window with an event stream of Δtd (which was described in Section 3.3) that corresponds to the current behavior of Ms. Smith. Therefore, the *appPervMed* will register in the Module Monitor (Esper) of the Context Manager only the patterns that correspond to the sliding window used to calculate the network CPT represented in (8).

```
select e from pattern                    (8)
[every e=Event].win:length(45)
where e.name='ea1' or e.name='a1' or
name='a2' or name='a3' or name='a4'
```

In the previous pattern, which was deployed in Esper, we have defined a sliding window of Δtd corresponding to the last 45 times in which Ms. Smith had not taken her medicine (event *ea1*), as well as the actions that were detected after that situation had happened. In this pattern, if Ms. Smith should take a medication once a day, this window would correspond to 45 days. Thus, the value of the probabilistic predictive situation always will be set to a percentage that corresponds to 45 days. In this sense, we avoid network scalability problems related to the excessive number of events (since they are modeled as nodes in the network), and detect cognitive declines with periods less than 45 days. This pattern generates the sequence of events that constantly updates the Bayesian network and the values of the probability of the event that determines the beginning of unwanted situations of this kind.

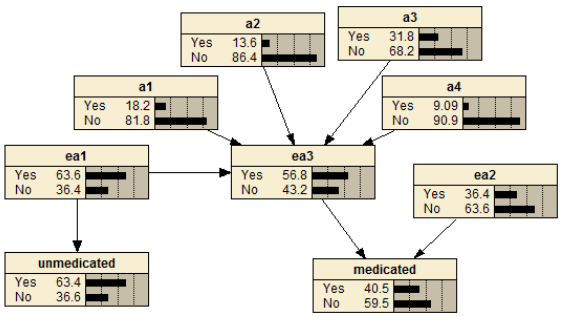

Figure 6: appPervMed Bayesian network.

As Figure 6 shows, event *ea1* is the cause of the "*unmedicated*" situation. This event, as well as events *a1*, *a2*, *a3*, *a4*, also influences event *ea3*. Besides, *ea2* and *ea3* influence the fact of Ms. Smith being medicated or not. The probability of Ms. Smith not taking her medicine at the correct time is 63.4%, according to the simulated behavior.

Based on this data, the SIaaS will update the probability attribute of the corresponding predictive situation for this case, as shown below:
Predictive Situation:

name = unmedicated;
{event} = {ea1, ea3, a1, a2, a3, a4};
{p} = ea1;
pr = 63.4%;
time = ea1.drug.timeTakeDrug;

The value of *time* is extracted by navigating through the event *ea1* and following to the entity of context drug and attribute time to take the medicine (timeTakeDrug) this entity of context. The *appPervMed* thus registers the following rule (9) that shows the relevance of the prediction value.

```
IF ‹unmedicated.pr, greather-
          Than,60%›
⇒‹Ms. Smith, isUnmedicatedIn,
  ea1.drug.timeTakeDrug›
```
(9)

This rule demonstrates that the "*unmedicated*" situation could happen the next time that Ms. Smith needs to take the medicine, activating the Proactive Actions Manager (PAM). The PAM uses the Bayesian network to identify, among the triggered actions, which one had the greatest influence for *ea3* to be detected and ending the "*unmedicated*" situation of Ms. Smith. It gives more relevance to the actions that were followed by *ea3* (i.e., took medicine after some action). If there is a sequence of actions triggered after the detection of an unwanted situation (corresponding by *ea1*), the PAM will chose the last action as being responsible for the ending of the unwanted situation (i.e., *ea3* detected).

Figure 6 shows that the action *a*3 (notify the caregiver) was the most successful action at this moment, thus the PAM changes the policy of triggering actions to a3 from "notify caregiver after *45 min of ea1*" to "*notify caregiver BEFORE 45 min of ea1*". Thus, the SIaaS will warn the caregiver that he/she has to ensure that Ms. Smith will take the medicine at the right time. In this case, the *appPervMed* will not be triggered to perform a reactive action, because event *ea*1 will not happen. This behavior causes more events of type *ea*2 to be detected since Ms. Smith starts taking her medicine at the right time, so consequently the situation "*unmedicated*" will not happen and the probability of the Bayesian Network for being medicated (*ea*2) increases.

We assume that there is a parameterized value with the criteria of policy selection actions to update the triggering order of proactive actions after this rule be updated. This will avoid that an action, after being identified as most effective, be always chosen as a proactive action that should be executed forever. This calibration is necessary because, in this case study, we are monitoring the behavior of a person, and this behavior may change over time. In the example given, Ms. Smith could not respond to the warnings of the SIaaS for some period of time because she was unmotivated with the treatment, so the warnings to the caregiver would effectively make her taking her medicine. However, if she did not show cognitive decline, she could return to her self-health management without requiring the notification of the caregiver. Therefore, there is a need for policies that trigger proactive actions to be updated. So, the SIaaS will again generate warnings to Ms. Smith.

6 CONCLUSIONS

Most of the research efforts in situation awareness for AAL are generally related to the detection of situations and the immediate reaction for these situations. In this sense, we have demonstrated the necessity of mechanisms to act proactivity in order to avoid unwanted situations in AAL.

In addition, we consider that for an Ambient Intelligence application to act proactively it must have learning capabilities. Therefore, these applications must understand and learn from the events that happened, predicting situations of interest and making decisions in advance related to the user needs. Thus, we consider the use of a Bayesian Network for identifying when it is necessary to act in a changed way.

In this paper, we presented an approach for enabling smart environments with reactive and proactive characteristics, more specifically in AAL. The main contributions of our approach are: (i) a method for supporting extensibility in systems to Ambient Assisted Living by including experts experience while modeling pervasive applications; (ii) an approach for handling reactive and proactive behaviors; and (iii) a model of sliding windows for modelling time in complex event processing.

The next steps of this research include (i) finishing an application prototype; (ii) testing the situation prediction over a real world automated environment; (iii) improving aspects related to the prediction model; and (iv) adapting the predictive model for taking decisions in a dynamic Bayesian network.

REFERENCES

Augusto, J. C., Nakashima, H., Aghajan, H., 2009. Ambient intelligence and smart environments: a state of the art. In: Handbook of ambient intelligence and smart environments: part 1. New York: Springer. pp. 3–31.

Boytsov, A., Zaslavsky, A., 2011. Context prediction in pervasive computing systems: Achievements and challenges. In: Supporting Real Time Decision Making, ser. Annals of Information Systems, Springer, vol. 13, pp. 35-63.

Dey, A., Abowd, G., 1999. The context toolkit: Aiding the development of context-enabled applications. In: Proc. of the SIGCHI conference on Human factors in computing systems, Pittsburgh, Pennsylvania, US, pp. 434-441.

Chan, M., Esteve, D., Escriba, C., Campo, E., 2008. A review of smart homes—present state and future challenges. Comp Computer Methods and Programs in Biomedicine, vol. 91, no. 1, pp.55-81.

Etzion, O., Niblett, P., 2010. Event Processing in Action. Manning Publications Co.

Engel, Y., Etzion, O., 2011. Towards proactive event-driven computing. In: Proceedings of the 5th ACM international conference on Distributed event-based system, pp. 125-136.

Engel, Y., Etzion, O., Feldman, Z., 2012. A basic model for proactive event-driven computing. In: Proc. 6th ACM International Conference on Distributed Event-Based Systems, pp.107-118.

Esper, 2010. Complex Event Processing. Espertech event stream intelligence, http://esper.codehaus.org/.

Friedman-Hill, E., 2003. *Jess in Action: Java Rule-based Systems*, Manning Publications Company, http://herzberg.ca.sandia.gov/jess/.

Fu, S., Xu, C., 2010. Quantifying event correlations for proactive failure management. In: Networked computing systems. Journal of Parallel and Distributed Computing, vol. 70, no. 11, pp. 1100–1109.

Jara, A. J., Zamora, M. A., Skarmeta, A. F. G., 2011. An internet of things based personal device for diabetes therapy management in Ambient Assisted Living (AAL). Pers Ubiquit Comput. vol. 15, no. 4, pp. 431–440.

Kokar, M. M., Matheus, C. J., Baclawski, K., 2009. Ontology-based situation awareness. In: Journal of Information Fusion. vol. 10, no. 1, pp. 83–98.

Lotfi, A., Langensiepen, C., Mahmoud, S. M., Akhlaghinia M. J., 2012. Smart homes for the elderly dementia sufferers: Identification and prediction of abnormal behavior. In: Journal of Ambient Intelligence and Humanized Computing, vol. 3, no. 3, pp. 205–218.

Lukasiewicz, T., Straccia, U., 2008. Managing uncertainty and vagueness in Description Logics for the Semantic Web. In: Journal of Web Semantics vol. 6, no. 4, pp. 291-308.

Machado, A., Pernas, A. M., Augustin, I., Thom, L. H., Krug, L., Palazzo, J., Oliveira, M. De, 2013. Situation-awareness as a Key for Proactive Actions in Ambient Assisted Living. In: Proc. of the 15th International Conference on Enterprise Information, pp. 418–426.

Mccarthy, J., 2002. Actions and Other Events in Situation Calculus. In: Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann Publishers, Toulouse, pp. 615– 628.

Nazerfard, E., Cook, DJ, 2012. Bayesian Networks Structure Learning for Activity Prediction in Smart Homes. In: 8th International Conference on Intelligent Environments (IE), pp.50-56.

Netica, accessed in 2013. https://www.norsys.com/netica.html.

Paganelli, F., Giuli, D., 2011. An ontology-based system for context-aware and configurable services to support home-based continuous care. In: IEEE Trans. Inf. Technol Biomed, vol. 15, no. 2, pp. 324-333.

Sadri, F., 2011. Ambient Intelligence: A survey. In: ACM Computer. vol. 43, no. 4, pp. 1-66.

Salfner, F., Lenk, M., Malek, M., 2010. A survey of online failure prediction methods. In: ACM Computing Surveys. vol. 42, no. 3, pp. 10–42.

Sixsmith, A., Meuller, S., Lull, F., Klein, M., Bierhoff, I., Delaney, S., Savage, R., 2009. SOPRANO: An Ambient Assisted Living System for Supporting Older People at Home. In: Lecture Notes in Computer Science. Springer Berlin: Heidelberg, vol. 5597, pp. 233–236.

Tazari, M.R.., Furfari, F., Lazaro, Ramos J.P., Ferro, E., 2010. The PERSONA Service Platform for AAL Spaces. In: Nakashima, H. et al. (eds.) Handbook of Ambient Intelligence and Smart Environments, Springer, Heidelberg, pp. 1171–1199.

Tennenhouse, D. L., 2000. Proactive computing. In: Communications of the ACM, vol. 43, no. 5, pp. 43–50.

Want, R., Pering, T., Tennenhouse, D. L., 2003. Comparing autonomic and proactive computing. In: IBM Systems Journal (IBMSJ), vol. 42, no. 1, pp. 129–135.

Wasserkrug, S., Gal, A., Etzion, O., 2005. A model for reasoning with uncertain rules in event composition systems. In: Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence, pp. 599-606.

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

Ye, J., Dobson, S., McKeever, S., 2011. Situation Identification techniques in pervasive computing: a review. In: Pervasive and Mobile Computing. vol. 8, no. 1, pp. 36-66.