Dependability Analysis in the Ambient Assisted Living Domain: an Exploratory Case Study

Genaína Nunes Rodrigues, Vander Alves and Renato Franklin
Departamento de Ciência da Computação
Universidade de Brasília, Brasília, DF - Brazil
Email: genaina@cic.unb.br, valves@unb.br, renflanklin@gmail.com

Luiz Laranjeira
Faculdade Gama
Universidade de Brasília, Brasília, DF - Brazil
Email: luiz.laranjeira@gmail.com

Abstract—Ambient Assisted Living (AAL) investigates the development of systems involving the use of different types of sensors, which monitor activities and vital signs of lonely elderly people in order to detect emergency situations or deviations from desirable medical patterns. Differently from the state-of-the-art of ‘push-buttons’, AAL solutions need to provide high accuracy and proactive responses, ‘perceiving’ lonely elderly people in their household environment through various sensors and carrying out appropriate actions under the control of the underlying software. Dependability in the AAL domain is a critical requirement, since poor system availability, reliability, safety, or integrity may cause inappropriate emergency assistance to potentially have fatal consequences. Nevertheless, contemporary research has not focused on assessing dependability in this domain. Therefore, this work aims at exploring the application of modern quantitative and qualitative dependability analysis techniques based on software architecture. The benefits of using these techniques are twofold. Firstly, they allow us to seamlessly integrate the analysis during subsequent software lifecycle stages in critical scenarios. Secondly, we aim to identify the components which have the highest impact on software system dependability, and therefore, be able to address software architecture and individual software component problems before they are implemented and critical errors occur.

I. INTRODUCTION

Ambient Assisted Living (AAL) [1] investigates the development of systems that monitor activities and vital signs of lonely elderly people in order to detect emergency situations or deviations from desirable medical patterns. Differently from the state-of-the-art of push-buttons, AAL solutions need to provide high accuracy and proactive responses, perceiving lonely elderly people in their household environment through various sensors and carrying out appropriate actions under the control of the underlying software.

A key issue in the AAL domain is dependability. Dependability of a system “is the ability to deliver service that can justifiably be trusted” or simply “the dependence being placed on that system” [2]. Also, dependability is an integrating concept that encompasses the following attributes: availability—readiness for correct service; reliability—continuity of correct service; safety—absence of catastrophic consequences to the user and the environment; integrity—absence of improper system alteration; and maintainability—ability to undergo modifications and repair [2].

The need to provide ways to analyze dependability in the AAL domain is crucial as the assisted person relies on the AAL solution when an emergency occurs: the system detects the emergency situation and calls the emergency service. The design of AAL systems, being still in its infancy, poses significant challenges. In particular, it is not completely clear how to address dependability of such systems. Therefore, the importance of conducting a dependability analysis in the early stages of the software development cycle becomes even more evident [3]. Such analysis would provide early identification of system problems which would avoid unforeseen expenditures of cost, time, and effort.

In the AAL domain, there are few case studies and experiment reports [1], [4]. It is still difficult to do verification and validation of AAL systems in real scenarios due to their complexity and the unavailability of reference implementations. So we need first to find ways to quantitatively assess the architecture of AAL systems for dependability. To accomplish that we must express dependability in terms of relevant domain properties and identify critical components that may require special design attention and project resource allocation.

Accordingly, in this paper we perform dependability analysis in an AAL architecture [5]. The analysis method consists in conversion patterns from UML behavior models (Activity and Sequence diagrams) of the AAL software architecture into a formal model, based on a probabilistic process algebra description language, in order to enable sound quantitative and qualitative analysis prior to software implementation. The UML models specify the components interactions in the system architecture and are annotated with component failure probabilities and system usage profile information. Furthermore, from the formal model, we identify a set of domain-specific dependability properties expressed declaratively in Probabilistic Computational Tree Logic (PCTL) [6]. PCTL is used to query the aforementioned process algebra model to enable dependability and sensitivity analysis. As a result, we are able to identify critical components requiring special design attention and thus to optimally allocate project resources early in the AAL development lifecycle.

The remainder of this paper is structured as follows: Section II covers background on AAL, on the formal language, and on the environment employed in the dependability and sensitivity analysis. Next, Section III presents the employed method. Section IV then describes the case study employing the method in the AAL domain. Section VI conducts evalua-
tion and discusses lessons learned. Related work is discussed in Section VII, and Section VIII offers concluding remarks.

II. BACKGROUND

A. AAL

AAL systems are by definition assistance systems. The assistance functionality comprises two aspects: (i) a user-initiated access to services, e.g., home control, social interaction, etc., and (ii) a system-initiated (proactive) provision of services, e.g., emergency support, home automation, activity coaching. In order to support the anticipatory assistance, the system must include some kind of closed loop controller that senses its environment (especially the persons living therein) and influences the environment with its actuators. The rendered functionality can be decomposed into two functional blocks: Awareness and Presence (Figure 1) [1].

![Diagram](image.png)

Fig. 1. Ambient Assisted Living Reference Architecture [1].

Awareness denotes the sound understanding of the assisted person’s current situation in terms of physical and mental health conditions, of the person’s context and goals, as well as of the environmental situation. Awareness can be decomposed in three functional blocks: (i) Sensing measures parameters in the environment (especially those related to the persons living therein) and provides results as numeric values. (ii) Perception transforms sensed numerical values into symbolic ones, e.g., the location of an object, and detects relevant situations. While doing so, the signal quality is improved by means of sensor fusion. If required, it also tracks the activities and states in form of traces over time and aggregates them into traces on different abstraction levels. (iii) Identification analyzes the perceived situations, e.g., behavioral deviation, and assesses them. It also recognizes recurring patterns as well as deviations from them and thus identifies higher level situations, e.g., emergency situation. Additionally, the system must be able to predict the response of the persons and their environment to possible actions and must be able to align predicted and actual responses. AAL systems have an awareness of the assisted person’s situation, that of the respective environment, and its own situation. To operationalize this awareness, the system must comprise a model of the user (referred to as patient profile), of the environment, and of the system itself.

The Presence part of the system comprises those functionalities that directly affect the assisted persons or their environment (e.g., a medication reminder, automatic control of the lights during the night). This functionality can again be decomposed into three functional blocks: (i) Planning assesses the current situations and goals provided by the Awareness block to decide if assistance is required and of what kind. It plans the assistance, and issues the appropriate high level commands, e.g., a medication reminder. (ii) Controlling executes commands firstly by decomposing them into single actions and secondly by translating them into numerical values. (iii) Acting influences the environment according to those numerical values. Besides Awareness and Presence, User Interaction supports the explicit interaction with the different end users.

B. The PRISM Model Checker

The dependability analysis of the AAL system in this work is accomplished through a probabilistic model checking tool called PRISM [7]. The reason for choosing PRISM as the probabilistic state-based model checker in this case study was twofold: (1) tool maturity, considering the number of successful case studies that have used the tool [8]; and (2) the richness of the tool environment, which is able to represent various kinds of probabilistic models and their evaluations, as we briefly explain below.

PRISM is a tool for formal modeling and analysis of systems which exhibit random or probabilistic behavior. It supports three types of probabilistic models: Discrete-Time Markov chains (DTMCs), Continuous-Time Markov Chains (CTMCs) and Markov Decision Processes (MDPs), plus extensions of these models such as the ability to specify costs and rewards. The tool has three environments: (1) one for system modeling in PRISM language, a state-based language derived from the Reactive Modules formalism; (2) one for model simulation; and (3) one for property specification, which uses temporal logics such as the Probabilistic Computational Tree Logic (PCTL) [6], [9] and includes extensions for quantitative specifications and expressions of costs and rewards.

In the modeling environment we model processes, which in PRISM are called modules. A model in PRISM is composed of a number of modules. Each module has a set of finite-ranged variables, which define the possible states of that module. The final model is the synthesis of all modules through parallel composition. Each module is composed of a set of guarded commands. For example, a DTMC command in PRISM takes the form:

\[
[action] < guard >\rightarrow< probability >::< update >;
\]

The guard is a predicate over all variables in the model, and once it is satisfied, the module will make a transition with a certain probability to the update state, where \(0 \leq
probability ≤ 1. The action can be used in order to tag a command that is synchronized with other commands in the same or in a different module. When there is no action label the command will run asynchronously. An example of a simple PRISM command is the following:

$$[notify] s = 0 \rightarrow \text{Rel} : (s' = 1) + (1 - \text{Rel}) : (s' = 2);$$

which states that, if \( s \) is 0, then it makes the transition either to state 1, with probability \( \text{Rel} \), or to state 2, with probability \( 1 - \text{Rel} \). Also, note that we use the action \( \text{notify} \) in order to synchronize with other commands labeled with the same action in accordance to Communicating Sequential Process (CSP) rules [10], which synchronizes all commands with the same action label, once their guard conditions are satisfied.

Once system modeling is completed using the PRISM specification language, PRISM reads and parses the language statements and constructs the corresponding probabilistic model, in this case a DTMC (although it can also be used for CTMC and MDPs as well). PRISM computes the set of all states reachable from the initial state and checks the model for deadlocks. In the simulation environment PRISM allows the visualization of possible execution traces of the synthesized model.

Another feature in PRISM is the ability to specify properties of the probabilistic model. Properties in PRISM are expressed using temporal logic (e.g., in PCTL). PRISM also performs model checking, determining the quantitative value of each specified property and whether the model satisfies it. For example, in dependability analysis a very useful property is the reachability property expressed as: \( P = \sum [ F (\Phi) ] \), which computes the probability that the system will eventually reach a state that satisfies \( \Phi \). In dependability analysis reachability is an important property to satisfy. It guarantees that the final successful state of the system will be reached, in this case, regardless of the time elapsed to reach it from the initial state.

### III. Method

Our method for dependability and sensitivity analysis consists of firstly identifying ways to convert the UML models (particularly the Activity and Sequence Diagrams) into the modeling language used in PRISM, while preserving the semantic expressed in the UML models. The purpose of modeling in PRISM as a probabilistic model checking tool is to make a qualitative and quantitative dependability analysis of the model using sound techniques before implementation takes place. Secondly, we annotate the PRISM model with variables denoting component reliability and transition probabilities in order to quantify its dependability through PCTL properties. This whole process is represented in Figure 2. The conversion process from UML to PRISM models is described in Section III-A; the analysis is domain-specific and is described in the context of the case study and is presented in Section V. Section IV illustrates the process in a case study.

#### A. Conversion from UML to PRISM Models

This section presents how we map UML models, particularly Activity and Sequence diagrams, to PRISM. Initially, each action node becomes a module in PRISM. The conversion process consists in first building a state machine model that represents each node from the Activity Diagram (AD). The AD is composed of two types of nodes: decision and action. Action nodes represent execution scenarios, each represented as a sequence diagram (SD). Each message in the SD is represented as a transition between states, annotated with the probability of the component processing that message successfully.

Specifically, the conversion from the action node to a PRISM module is accomplished as follows. For each component service execution in the SD that models the action node, there is a corresponding state in the state machine, while the messages exchanged between components in the SD are represented as labeled transitions between states. Therefore, for each state of the state machine model there is a component \( C \) processing a service. For this reason, each state is associated with the component reliability \( R_c \), i.e., the probability of successful execution of the service, while the probability of the service failure \( (1 - R_c) \) is represented by a transition to state \( E \). By failure we simply mean that an error is propagated to the service interface causing deviation from correct to incorrect service [2]. This failure can be classified in various modes: domain, detectability, consistency and severity. Our major property of interest in our modeling is the probability of reaching the end state of the system successfully, i.e. reachability.

Differently, decision nodes represent choices and each of their outgoing transitions is represented as a transition between states in the state machine model, annotated with the probability of transitions between scenarios. Precisely, the conversion from decision node to a PRISM module consists of representing the probabilities of transition \( PT_{ij} \) from a decision node \( i \) to an action node \( j \). This information would be normally derived from a system usage profile [11]. Therefore, each decision node in the AD is represented as a state and each outgoing transition in the state machine is labeled with \( PT_{ij} \). Thus, from decision node \( i \), the sum of the probabilities \( PT_{ij} \) for all successor action nodes \( j \) is equal to one.

Once the PRISM model is finished, thus consisting of a set of modules representing state machines, the synthesis of the final stochastic model follows the CSP synchronization rules and the probabilistic compositions for the corresponding Markov model of choice. In our case, we focus on DTMC class as it describes a more direct association with state machine models as each state and their respective sets of transitions can be consistently represented as a command line in DTMC, which is not usually true for other kinds of markov models. Therefore, we annotate the models with discrete values of their corresponding probabilities of both component reliability and transition probabilities to the states.

### IV. Case Study

#### A. Context

The goal of this case study is to perform dependability and sensitivity analysis in the AAL domain. The scope of
the domain consists of a specific software architecture in this domain (described in Section IV-B), and the methodology is the one described in Section III.

B. AAL System

Figure 3 depicts the high-level static view of an instance of the AAL reference architecture presented previously. The remainder of this work refers to this concrete architecture, which is a view of the EMERGE system architecture defined elsewhere [5]; the view is sufficient for the scope of this paper, which is on detection and handling of acute medical emergency of an assisted person at home. Figure 3 shows the relationships among the main subsystems involved. Assistants are components that render the higher level system functionality and define the overall workflow. There are two kinds of assistants: Emergency and Personal Assistants. The former are responsible for planning actions to react upon a given medical emergency, whereas the latter are responsible for interacting with the assisted person (e.g., in confirming emergency situations). Perception components perceive different situations based on the low level sensor data, e.g., a complex situation. Dispatch represents the dispatch center subsystem, which is contacted by the AAL system within the home (inside large square) once an emergency occurs, and EDOC is the emergency doctor subsystem contacted by the dispatcher and thus assigned to handle the emergency at the home hosting the AAL system. The following subsections present and relate different views of this architecture, highlighting its functional and stochastic behavior.

C. UML Modeling

We first present an Activity diagram (Figure 4), which is based on a view of [12] by focusing only on the emergency scenarios, representing all the possible flows, starting with the occurrence of the emergency situation until its cancelation or its final state of success when all possible actions have been performed to handle the emergency.

Next, we map each action node of this Activity diagram to a Sequence Diagram that illustrates the interaction between the components of the system in the proposed action. The sequence diagrams are our starting point to the state machines that will be represented by the PRISM models, as presented Section IV-D.

Accordingly, the sequence diagram Emergency detection scenario (Figure 5) corresponds to the actions Sensors detected emergency and shows a scenario in which the components and roles involved are the following: the Sensor Node—representing ambient sensors that regularly send data to the Perception component—and the perception component—responsible for receiving the data from the Sensor Node, treating it properly, and detecting the possibility of an emergency. Sequence diagram Emergency confirmation request, shown in Figure 6 and related to the action System notifies assisted person about detected emergency the components and roles involved are the following: the Perception component—responsible for the detection of the emergency after receiving the raw data from the Sensor Node and for notifying the detected emergency to the Emergency Assistant. This latter component will manage the information about the emergency and coordinate other components (Personal Assistant and UI) for handling this situation. The User Interface and the Personal Assistant are involved with the assisted person in the system. The Personal Assistant acts on behalf of the assisted person retrieving the user model and interacting with him/her via the UI when necessary, e.g., asking for a confirmation or issuing a warning.

The diagram Alarm manual activation scenario, in Figure 7, corresponding to action Assisted person raises the alarm...
depicts the interaction of the assisted person with the User Interface, the Personal Assistant and the Emergency Assistant. In this scenario the Perception component is not involved because the emergency is not detected by the sensors, but rather manually raised by the assisted person.

Three sequence diagrams (not shown here for brevity) correspond to the activities on the right-lower part of the Activity diagram and represent the steps of the system after confirmation of the alarm: retrieving personal and medical information about the assisted person from the user model, contacting the emergency service and notifying the assisted person that the emergency support is on the way. In the left-lower part of the Activity diagram stands the action

**Assisted person cancels the alarm**: The sequence diagram that illustrates this action (not shown here for brevity either) represents the necessary steps to cancel an emergency, either an automatically detected or a manually raised one.

To model communication with other subsystems, like the Dispatch Center or the EDOC service (Figure 3), we have introduced a new component that represents the communication channel between the subsystems. The reliability of this connection will be assigned to this channel in the PRISM model. With this level of details we were able to develop the state machines of the system that will be represented by the PRISM modules.
D. Conversion from UML to Prism Models

The overall procedure of the conversion process for our AAL case study is as follows. There are nine action nodes in the Activity Diagram (AD), out of which, seven are represented by a Sequence Diagram each and two are symbolic representations to a decision node. In this section we illustrate how the conversion to an action and to a decision node are accomplished, one for each case. The nodes chosen are the 'Emergency situation occurs' and 'Sensors detected emergency' in Figure 4, which represent a decision and an action respectively.

Following the conversion process described in Section III-A, we represent in Figure 8 the state machine that models the decision between manual or automatic emergency detection. Figure 9 represents the state machine model for the Emergency Detection SD.

![State Machine Representation of Manual or Automatic Emergency Detection](image)

**Fig. 8.** State Machine Representation of Manual or Automatic Emergency Detection

![Emergency Detection State Machine](image)

**Fig. 9.** Emergency Detection State Machine

From the state machine model, the representation in PRISM can be directly derived as illustrated in Figure 2. The resulting PRISM module of the decision node modeled as the state machine in Figure 8 is presented in Listing 1, whereas the Emergency Detection state machine in Figure 9 is presented in Listing 2.

**Listing 1. Decision Module for Manual or Automatic Emergency Activation**

```prism
1 dtmc
2 3 // Trans. probability to Alarm Manual Activation
4 const double PT_Manual;
5 6 const double PT_Automatic;

7 8 module Initial_Decision
9 10 s0 : [0..2] init 0;
11 12 [] s0=0 -> PT_Manual:(s0' = 1) + PT_Automatic:(s0' = 2); [manual]
13 [ auto] s0=2 -> (s0' = 2); // Synch. with automatic
14 15 endmodule
```

**Listing 2. Emergency Detection module in PRISM**

```prism
1 // Sensor node reliability
2 const double R_SN;
3 // Perception reliability
4 const double R_Prcp;

5 6 module Emergency_Detection
7 8 s1 : [0..3] init 0;
9 10 [ auto] s1=0 -> R_SN:(s1' = 1) + (1 - R_SN):(s1' = 3);
11 [ auto] s1=1 -> R_Prcp:(s1' = 2) + (1 - R_Prcp):(s1' = 3);
12 [ emerg] s1=2 -> (s1' = 2); // End state
13 [ fail ] s1=3 -> (s1' = 3); // Failure state
14 15 endmodule
```

**Listing 3. Decision Module for Alarm Cancellation in PRISM**

```prism
1 // Trans. probability to Cancel
2 const double PT_Cancel;
3 // Trans. probability to Notification
4 const double PT_Notification;

5 6 module Cancel_Alarm
7 8 s : [0..2] init 0;
9 10 [ cancel] s=0 & s2_rel -> PT_Cancel:(s' = 1) +
11 (PT_Notification):(s' = 2);
12 [ cancel] s=1 -> (s' = 1);
13 [ confirm] s=2 -> (s' = 2);
14 15 endmodule
```

In Listing 1, we show the declaration of the model as a DTMC in line 1. In lines 4 and 6 we declare the constants for the transition probabilities $PT_{Automatic}$ and $PT_{Manual}$. The module name is specified in line 8 and the set of states in line 10. The remainder of the specification (lines 12-14) present state transitions, a line specifying each of which and where the left-hand side of the arrow shows previous state (denoted by $s$) and right-hand side of the arrow shows the next state (denoted by $s'$). In Listing 2, we declare the constants for the component reliabilities in the Emergency Detection state machine in lines 2 and 4 representing the reliability of Sensor Node and Perception, respectively $R_{SN}$ and $R_{Prcp}$. Note that the failure state modeled in Figure 9 as state $E$ is equivalent to state 3 (line 13) in module $Emergency\_Detection$.

Two points of note are as follows. Firstly, lines 10 to 12 in Listing 2 are labelled with $auto$ and $emerg$ actions. This is used for synchronizing actions between modules, allowing the modules to make transitions simultaneously according to their respective guard conditions. In this case, the action $auto$ is used by $Initial\_Decision$ module in Listing 1, line 14, to indicate the occurrence of an automatic emergency and therefore the Emergency\_Detection module should be triggered.
Secondly, guards of these commands may refer to variables from other modules, used to indicate that a module starts only once the boolean condition has been satisfied. Listing 3 illustrates this situation, where the boolean variable $s_2\text{\_rel}$ in line 10 is used to indicate that module Cancel\_Alarm starts only once that variable is true. The variable $s_2\text{\_rel}$ is declared in module Emergency\_Confirmation, and it is used to indicate when that module reached the end state successfully.

The final model synthesized in PRISM contains 69 states and 121 transitions. In PRISM simulation environment, we make extensive runs of the synthesized model in order to check if it consistently expresses the behavior modeled in the UML specification of the AAL system.

V. ANALYZING THE DEPENDABILITY OF THE AAL SYSTEM

We now make a qualitative and quantitative analysis of AAL system modeled in PRISM. We have used PRISM in order to build a DTMC model of the system, as described in Section IV-D, and to analyze its dependability properties of interest. In the core of this analysis are the PCTL properties. We first consider the qualitative properties; next, we reason about the quantitative ones, and then we conduct a sensitivity analysis in order to filter out those components that have a major impact on the dependability of the system. For the qualitative analysis we consider the properties liveness and safety, whereas for the quantitative analysis, we consider the property of reachability of the successful final state, as presented in Section II.

A. Qualitative Analysis

The qualitative analysis of the AAL system consists in reasoning about safety and liveness property of the model, i.e., the possibility that eventually something good (the messages delivery) will happen. In this case, we are not worried if the message will fail, but if our model guarantees fairness in the execution of the PRISM modules preventing the messages to be delivered. In temporal logic, the concept of liveness is expressed as it is the case that something good “eventually” happens, using the F operator [6], [9]. In other words, this property is used to analyze if it is true that, from the initial state of the system, it will “eventually” reach the final system output, either failure or correct end. In our PRISM model, this property is evaluated through the following PCTL property:

- \( \text{“init”} \Rightarrow P \Rightarrow 1 [ F (execute) ] \),

where the atomic proposition execute is defined as follows: execute := end\(_1\) | end\(_2\) | end\(_3\) | end\(_4\) | end\(_5\) | end\(_6\) | end\(_7\), such that each end\(_i\) represents the final state of each PRISM module \( m_i \) and \( 1 \leq i \leq 7 \), where each \( i \) represents a SD in the UML model. Once executed in PRISM, this property returns true and thus liveness is assured.

Regarding safety, we intend to check if our model is deadlock free. In PCTL, instead of using the F operator (for “eventually” or “future”), we can use the G operator (for “all paths” or “globally”) [6], [9]. Therefore the PCTL property for verifying the model for safety is:

- \( \text{“init”} \Rightarrow G (execute) \),

Once executed in PRISM, this property returns true as well, meaning that there is no implicit end that is not part of the execute proposition. Therefore, the system will end up in either the failure state or in the successful end state, and will be deadlock free. The probability of reaching either state will depend on the reliability of the individual components, which we analyze next.

B. Quantitative Analysis

The quantitative analysis of the PRISM model consists of obtaining the system reliability as the probability of it reaching the terminal state from the initial state [13]. In the AAL system, the terminal state is either the canceling of the emergency by the user or the alarm notification reaching the caregiver at the final end. We express these states as cancel and notification, respectively. In PRISM, we use the following PCTL formula to obtain the AAL system’s reliability:

- \( P =? [ F (\text{cancel} \lor \text{notification}) ] \)

The PCTL statement queries to our DTMC model what the probability of reaching cancel or notification is, defined as follows:

- cancel := end\(_4\), where the alarm cancelation scenario, represented by index 4, reaches its successful end state.
- notification := end\(_7\), where the alarm notification scenario, represented by index 7, reaches its successful end state.

Considering that we do not have the initial value of the reliability of each component, obtaining a single reliability value for the AAL system is not meaningful in our analysis at this stage. However, for a preliminary quantification of the availability of the AAL system, we assume that the downtime of each component is no more than one minute per week which equates to 52 minutes per year. A quick arithmetic shows that each component steady state availability is 99.99%. We run the PCTL statement for the PRISM model of the AAL system and obtained approximately 99.81% for the system steady state availability, showing that the system would be unavailable around 8 hours per year.

Once the system is implemented and testing is underway, we can update the reliability values of the components and use them as input for our model. Another option is to obtain reliability values from reused components in case they are available. On the other hand, the most useful way to use this PCTL query is to compute and plot the value of system reliability obtained from the query by treating each individual component reliability value as an independent variable and varying this variable in order to identify those components that have a higher impact on the overall system reliability. This is accomplished in the following section.

1) The Sensitivity Analysis: The most useful way to analyze the model and to gain insights into its dependability, mainly reliability, is to compute and plot the values as some parameters are varied, i.e. to perform a sensitivity analysis. Plotting
the value of system reliability while varying the reliability of the components will reveal which components have higher impact on the overall system reliability. Therefore, focusing on enhancing the reliability of those components with proper allocation of project resources will be paramount to enhance the reliability of the overall system.

We structure the sensitivity analysis in two parts. Firstly, we consider the general scenario without taking into account any particular system usage profile. Secondly, we conduct sensitivity analysis based on the most frequent AAL system usage profile. For the quantification of the analysis we use the same PCTL reachability statement considered earlier in this section:

- \( P = \{ F ((\text{cancel}) \mid (\text{notification})) \} \)

According to this statement, we want to obtain the probability to reaching the cancel or notification state without failing. We have the reliability of the components as input parameters and vary the reliability one component at a time, from 80 to 100%. Note that, for this analysis, we only consider the long-run system reliability, i.e. we do not take into account the amount of time it takes for the system to reach its successful end. The outcome of this first part of the sensitivity analysis is depicted in Figure 10.

The Y-Axis in Figure 10 represents the reliability of the system, while the X-axis represents the reliability of a component. The analysis includes ten components of the AAL system. We order the legend from top to bottom, from the least to the most significant component. According to the plotted results, the steeper the slope for a component, the more significant is its impact on system reliability. This way, the most critical components are the Personal Assistant and the Emergency Assistant, while the Sensor Node is the least critical. Indeed, the Emergency Assistant and the Personal Assistant are in the core of the AAL system, their major function being to handle the emergency detection procedure. Therefore, this design decision is consistent with the outcome of this first sensitivity analysis.

2) Analyzing the Most Frequent Scenario: The second part of our sensitivity analysis aims at focusing the emergency detection scenario which occurs more often, according to system usage profiles reported on the AAL literature [14]. Our purpose in performing this analysis is to investigate if we can obtain useful information for this specific scenario, which could not be observed in the general sensitivity analysis conducted previously.

In this scenario we have to consider that the system will automatically detect the emergency using the data received from the set of sensors. It is important to consider that the assisted person may not be able to confirm the alarm when the system asks for this confirmation. In this case the alarm will not be canceled but the flow of system emergency actions must continue.

The mapping of the information of the most frequent scenario is modeled as the transition probabilities annotated in the outgoing transitions in the decision nodes of the Activity Diagram presented in Figure 4. Therefore, there are two decision nodes to consider: one to represent the alarm triggered either automatically (\( PT_{\text{automatic}} \)) or manually (\( PT_{\text{manual}} \)); and another to represent either the alarm cancelation (\( PT_{\text{cancel}} \)) or the notification of the emergency to the remote caregiver (\( PT_{\text{notification}} \)). In the most frequent scenario, an emergency signal is notified to the remote caregiver after the sensors automatically detect an emergency situation, such as the assisted person fainting. As a result, there is no transition to the manually triggered alarm state or to the alarm signal cancelation state, i.e. both \( PT_{\text{manual}} \) and \( PT_{\text{cancel}} \) shall be zero.

The outcome of this second part of the sensitivity analysis is depicted in Figure 11. The result of this analysis shows that Emergency Assistant and Personal Assistant are still the most critical components. However, it reveals that the AAL system’s reliability became more sensitive to the reliability of the Sensor Node and the Perception components. Compared to the previous results shown in Figure 10 the curves for these components have now been shifted vertically, implying a higher impact on overall system reliability. In fact, the occurrence of an automatic trigger of the emergency alarm raises the importance on the awareness operationalization for the assisted person. This functionality is accomplished through the Awareness block shown in Figure 1, that is exactly where
the Sensor Node and Perception modules are modeled in the refined concrete architecture (Figure 3). This outcome is, therefore, consistent with the results obtained through the sensitivity analysis.

VI. LESSONS LEARNED AND EVALUATION

A. Method Applicability

The methodology presented is intended to augment design activities with an assessment of dependability and sensitivity analysis prior to implementation. This is relevant to assess system’s architecture and adjust it, if necessary, to comply with desired dependability levels, thus avoiding deep restructuring after implementation begins (or becomes more intense). Furthermore, such analysis becomes crucial in domains with critical levels of dependability requirements, such as AAL. The analysis and modeling approach we described was conducted in the context of this domain. More precisely, the final results, regarding dependability and sensitivity, are specific to a specific architecture in this domain. Nevertheless, the conversion phase (Figure 2) could be applied to other domains as well. To the best of our knowledge, this conversion process from UML Activity and Sequence Diagrams (and their annotations) to PRISM language is a novel contribution. The PCTL logic could be applied to perform safety and liveness checks in other domains as well.

B. Usefulness of the results

The evaluation of properties (expressed in PCTL) in the quantitative analysis in Section V does not use real individual component failure rates, which may compromise the results of the assessed system’s reliability. Indeed, obtaining such failure rates is an issue in this domain, given component heterogeneity and availability. Nevertheless, the results of the qualitative and the sensitivity analysis in Section V remain valid and useful, despite the lack of such concrete data. Furthermore, the variation of the components sensitivity to the system usage is consistent with the architecture reference model. A system of the magnitude of AAL, once implemented, may be too complex to have its reliability validated and its levels of safety assured, based on the behavior of individual components. However, the analysis discussed in this paper has been shown to provide tangible support for assuring system dependability in its forthcoming realization state.

C. Threats to Validity

1) Construct Validity: Construct validity concerns establishing correct operational measures for the concepts being studied. In this work, we perform a preliminary dependability analysis for an AAL subsystem. As input data, we have the component reliabilities and the scenario transition probabilities. We assume relatively high uptime reliability for the components, considering that a failure in the system component could incur fatal consequences. The transition probabilities values were based on most frequent scenarios of the literature, but which could also stem from system profile usage.

2) Internal Validity: Internal validity concerns establishing a causal relationship, whereby certain conditions are shown to lead to other conditions. At first we applied a conversion process, during which we check safety and liveness properties of the model. At second we noticed that the variation of the reliability of individual components and their probabilities of transition do impact the final results computed using PRISM. In particular, the sensitivity analysis well illustrates this impact. In the analysis of the most frequent scenario we have shown that the sensitivity of the overall system reliability with respect to variations in the reliability of individual components depend on the particular scenario considered. However, since we carried out an exploratory case study, further empirical assessment is necessary for validating the proposed method. enough for validating the proposed method.

3) External Validity: External validity concerns establishing the domain to which a study’s findings can be generalized. As discussed previously, the conversion phase could be applied to other domains, whereas the analysis phase is domain specific. From a formal model perspective, we have used DTMC for system modeling. Although the model assumes discrete-time probability input values, we noticed that the model parametrization in PRISM and the action annotations used to synchronize PRISM modules may be applicable to various other types of software systems. At the moment we are carrying out other case studies to investigate the generalization of this approach.

4) Repeatability: Repeatability concerns demonstrating that the operations conducted in a study can be repeated with the same results. We expect that replications of our study should offer results similar to ours. Indeed, the characteristics of the specific system architecture may differ from the one used in the current study, but the underlying reference architecture should remain unchanged.

VII. RELATED WORK

Research on AAL is still in its early stages. Current efforts focus mainly on specification and design with few actual systems built [1], [4]. Within specification, dependability has been promptly identified as a key system quality and as an architecture driver [12]. Nevertheless, there is a lack of work assessing dependability in this domain which use sound qualitative and quantitative methods.

System architectures for Ambient Intelligence in general and Ambient Assisted Living in particular have been addressed in a number of prominent projects as SOPRANO [15], and OASIS [16], just to name a few. All of them focus on different applications, with different quality requirements. Unfortunately, there is currently only a dim understanding of how characteristics such as dependability are assessed and actually contribute to the overall achievement of user goals in specific scenarios, and of how the inherent tradeoffs can be handled in an optimal way. This considerably complicates the development of appropriate solutions, often leading to suboptimal solutions. A sound AAL quality model and quality service profiles that point to desirable system qualities and
their interdependencies would be helpful, but is not yet available today. To achieve this intensive interdisciplinary research will be required in the coming years.

Another thread of work focuses on implementing specific layers of the AAL reference architecture. In particular, Nunes [17] implements the Identification and Planning layer of the reference architecture in Figure 1, corresponding to the assistants in the concrete architecture in this work, using agents following the Gaia methodology [18]. Such methodology was used to improve traceability of stakeholder goals (assisted person’s well being) to system implementation, thus helping to guarantee desirable system properties.

In a previous work, Rodrigues et.al. [19] proposed a Message Sequence Chart (MSC) based reliability prediction technique and had the technique automated by using the Label Transition System Analyser Tool (LTSA) [20]. Though it was a sound work, the LTSA tool was not originally designed for probabilistic model checking, such as PRISM. On the other hand, PRISM does not provide the scenario-based modeling facilities such as the MSC plugin provided by LTSA [21]. This case study explored the potential of the PRISM tool using a model-driven approach to conduct a component-based dependability analysis in the early stages of the software development of a system that had dependability as a critical requirement.

VIII. CONCLUSION

Ambient Assisted Living investigates the development of systems involving the use of different types of sensors, which monitor activities and vital signs of lonely elderly people in order to detect emergency situations or deviations of desirable medical patterns. Dependability is a crucial quality in this domain which has not been adequately modeled and evaluated. The contribution in this work was to use modern quantitative and qualitative dependability analysis techniques based on software architecture and behavioral UML models to assess the dependability of a particular AAL system, showing that such analysis can bring significant gains early in the software development cycle. We carried out a sensitivity analysis in this case study to identify system components that have the highest impact on system software dependability. It was verified that this kind of analysis can uncover software architecture and individual module problems before they are implemented and critical errors occur. As future work, we plan to obtain concrete data on individual components failure rates to be able to finalize the quantitative analysis, to expand the set of PCTL properties, to apply the method to other domains that have dependability as a critical requirement, and to conduct further empirical assessment in order to validate the use of UML diagrams as a front-end to the probabilistic dependability model.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for invaluable feedback and the Fraunhofer Institute for Experimental Software Engineering for granting access to EMERGE’s documents. This work has been partially supported by CNPq, under Edital MCT/CNPq 14/2009 - Universal, grant number 482481/2009-9.

REFERENCES