Testing Ambient Assisted Living Solutions with Simulations

Marlon Cárdenas(✉), Jorge Gómez Sanz(✉), and Juan Pavón(✉)

Department of Software Engineering and Artificial Intelligence,
Complutense University of Madrid, Madrid, Spain
{marlonca,jjgomez,jpavon}@ucm.es

Abstract. The paper introduces a testing solution for evaluating Ambient Assisted Living systems by means of 3D simulations generated with a game engine, complex event processing, and classifiers. The solution aims to ensure that: (1) some key features of the problem appear in the simulation, and (2) the assistive solution interacts with the persons in the right way. A specific testing solution is needed because of the evolving nature of the simulation (each iteration of the requirements specification involves changes in the activities within the simulation) and the assistive solution (it operates in real time and evaluating its performance may require manually inspecting hours of simulation). The approach is illustrated with a proof-of-concept experiment.

Keywords: 3D simulation · Ambient Assisted Living (AAL)
Real-time simulation · Requirements gathering

1 Introduction

One of the main areas of application of Ambient Intelligence (AmI) is the support for improving the autonomy and quality of life of ageing population with Ambient Assisted Living (AAL) solutions [8]. They cover many needs of the daily life, such as facilitating the administration of medication, fall detection, and monitoring of chronic diseases or common activities, among others.

Although these tasks are usually performed by care-givers, AAL technologies are contributing to the assistance of older persons at home with new services, which can work 24 h a day and reach a larger population. The development of an assistive solution must take into account the participation of the end-users in its process, which has to be user-centered and co-creative [6], to prevent technology rejection situations. Co-creation happens usually in the context of expensive facilities, such as living labs.

Some recent tools [7] are translating the co-creation effort to the computer, addressing both the modeling of the problem and its corresponding assistive solution through computer 3D simulations [3]. 3D simulations are easier to understand than technical specifications, and do not require special skills by the users.

I. Medina-Bulo et al. (Eds.): ICTSS 2018, LNCS 11146, pp. 56–61, 2018.
https://doi.org/10.1007/978-3-319-99927-2_5

This facilitates communication in multidisciplinary teams. Though this is an advance, they have led to new problems: *how to ensure that each new iteration of the 3D simulations is consistent with previous iterations, and how to verify that the assistive solution and the simulated characters within are behaving as they should*. Since simulations are run in real time, validation requires some sort of time accounting while analyzing the events that are produced by the simulation. A way to do this is by using Complex Event Processing (CEP). This technique analyses incoming events from some user defined sources and applies user-defined rules to find patterns in them. We apply this feature to arrange unit tests to support the analysis of the 3D simulation progress.

The contributions of this work are an informal formulation of a unit test and the identification of the required elements for the validation of 3D simulations of AAL solutions. Section 2 presents an informal unit test definition. This is followed by a description of the deployment structure of the solution in Sect. 3. Section 4 presents a proof of concept, an example of memory loss scenario, which has been developed using simulations generated with the AIDE (*Ambient Intelligence Development Environment*) software framework [5], which is based on the use of model driven development techniques [4], and has been applied in different domains [2]. Finally, Sect. 5 summarizes the findings.

2 A Unit Test Definition

The unit test considers either the validation of simulation components (be it the behavior of its constituents, including the assistive solution) or the simulation as a whole. If the assistive solution is conceived within a use case of a certain technology in one or many scenarios, a single simulation represents one (preferably) of those scenarios. The developer expects to model a scenario that reflects some problem of a person (for instance, a person at home forgets to close a cupboard door because she initiates another activity), and an assistive solution (for instance, something detects the situation and reminds her that the door should be closed).

The testing goal is to generate validation instructions for each of these two issues (the description of the problem and the assistive solution), and both are assessed according to the progress of the simulation. The simulation consists of several elements, which imply specific validation issues:

Avatar. A character that performs actions within the simulation and interacts with the assistive solution. **Avatar validation** implies that (1) the character is in certain locations and that (2) it initiates, and (3) completes certain activities, successfully or not. The outcome of the activity, since it may involve an interplay with the assistive solution, needs to account the sensory input of the avatar (e.g., determining if anyone has talked to the character) or the status of some elements in the environment.

Activities. Activities in the simulation are presented as graphical animations of the gestures the character should make, such as running, walking, falling, water

tap open, switch on light, to cite some. **Activities validation** implies that (1) they occur along the simulation in a particular sequence, and that (2), despite the animation being used, the character is showing certain gestures, such as bending the arm, even-though there is no explicit isolation animation for that and it is just a part of a bigger animation. The outcome of the activity is evaluated at the avatar level.

Environment. The place where activities take place: a house, a mall, an university building, etc. This includes *Actuators*, objects that allow avatars in the simulation to interact with the environment, such as power switches, TV remote control, water taps, etc. **Environment validation** implies that (1) the objects in the environment produce the expected stimulus at the expected order, such as a TV showing the expected TV program, and that (2) they perceive the avatar's or assistive solution's actions, such as a fridge door being closed by the avatar or a house alarm being activated by the assistive solution.

Sensors. They provide information on what is happening in the simulation, from an avatar, the environment, or the interaction among them. They intend to reproduce the expected output of a real sensor. **Sensors validation** implies (1) determining that the expected sensory input is present at the necessary moments as a consequence of some actions performed along the simulation by the character, other objects, or the assistive solution itself. Also, (2) that the sensor output matches in frequency and data quality those values obtained by real sensors. For instance, some sensors need to be modeled with noisy signals because of limitations of the current technology.

2.1 Validation Success Criteria

Evaluating the success of an assistive solution-simulation interplay requires taking into account two issues. First, that the simulation continuously produces **streams of events** that have to be processed on-line (if the simulation is a long one and the developer wants to identify failures sooner) or off-line (if the simulation is a short one). Second, that the interaction of the assistive solution and the simulation may be intermittently successful along the simulation.

The idea of **time window** is used to partially deal with these two issues. A **time window** classifies the events from the simulation into groups corresponding to those produced within two instants of time. The size of the time window will be defined by the developer depending on the domain.

The success of the simulation-solution will then be defined in terms of: *what situations should occur, or should not, within the time window; and whether what happened in some window, should or should not happen in some/any/all existing time windows along the execution.*

3 Testing Infrastructure

The software components that are necessary for this validation are run within three separated nodes, as shown in Fig. 1. **Simulations** are created and run with

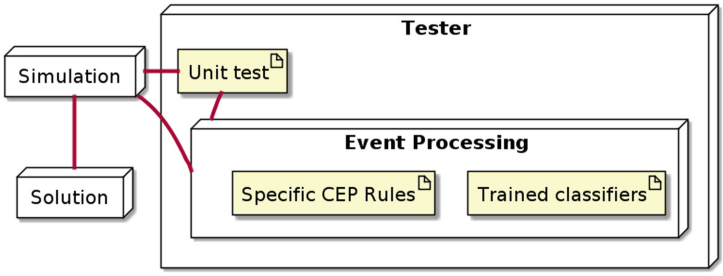

Fig. 1. UML deployment diagram showing nodes and artifacts involved in the testing

the AIDE tool. AIDE also enables communication between the **Solution** and the **Simulation**. The **Tester** is the one being introduced in this paper.

The **Solution** runs outside the **Simulation**, so as to reproduce real situations where delays in the decision making may lead to failures in assisting people. It can be made of emulators (e.g., Android emulators or IoT emulators) or plain processes enabled with communication facilities towards the simulation. It interacts with the **Simulation** through AIDE [5] middleware, obtaining information from the simulated sensors deployed within the simulation (e.g., movements of some limbs of the avatar) and triggering actions (e.g., speaking to the avatar). The **Simulation** uses the open source 3D game engine JMonkey to run the actions as defined in a custom visual language [1]. The simulation runs in real time, though it can be accelerated to some extent.

The **Tester** is the node performing the validation of the simulation and the interplay of simulation and the solution. It initializes and launches a Complex Event Processing (CEP) instance with the necessary files containing the **Specific CEP rules**, the **Trained classifiers**, and a particular **time window** (CEP is configured with one). The **Specific CEP rules** analyze the stream of events as produced by the simulation and generates success/failure states within each time window. The rules identify patterns in the generated events along the **time window** and generates new labeled events, such as *the avatar is in the living room* or *the avatar is running*. Other CEP rules combine the new labeled events with other pieces of information to implement the success criteria within a time window and across time windows.

Some low level information cannot be informed directly by the simulation, such as whether a door is open or closed, if there are people speaking, or if the character is raising a hand. Focusing on the later, if the situation to be reproduced is a character that cannot perform movements with an arm, this cannot be identified by just checking what animations are being run. The only way is by checking position and movement data associated to the hand component in the simulation. **Trained classifiers** allow to deal with such cases. They are connected directly with the **Simulation** and classify low level events, e.g., position of elements, into some predetermined categories, such as *the arm is being moved*. The classifiers have to be trained to the situations to be recognized, though, which may not be trivial for all developers.

4 Proof of Concept

This framework is illustrated with an assistive system that should detect when doors are left open. This case is relevant in cases of memory loss, which are frequent in patients with Alzheimer. Table 1 describes the sequence of validations to be performed and that affects both objects in the environment (the fridge door, the glass), the location of the avatar (kitchen and living room), and the activities (walking, drinking, opening doors, sitting down). Some low level evaluation is required to decide.

Table 1. Definition of the activities of the simulation and the corresponding validation steps

Order - Patient Activities	Test Case	sensors	Expected output
1-patient is in the living room of the house watching television	checkStop	s_1, s_2, s_3	true
2-patient walks to the refrigerator	checkMovements	s_2, s_3	'get up/get down'
3-patient opens the refrigerator door	checkMovement	s_2	'get up hand'
4-patient drinks water for 30 sec	checkMovement	s_2	'get glass'
5-patient closes the refrigerator door	checkMovement	s_2	'get down hand'
6-patient returns to the sofa	checkMovements	s_2, s_3	'get up/get down'
7-end simulation	checkStop	s_1, s_2, s_3	true

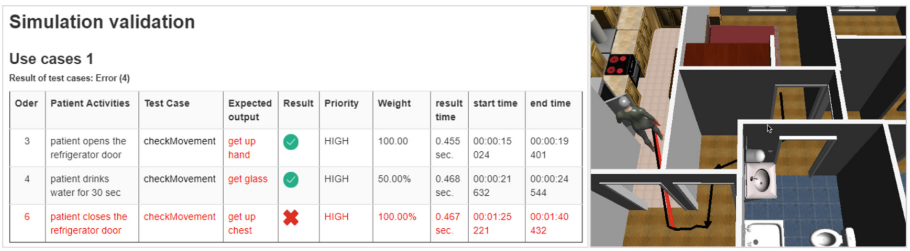

Fig. 2. Result of all the tests after running the 3D simulation of the scenario

It is expected that step number 5 fails because the patient forgets to close the door. Only an assistive solution will change the situation and make the test complete successfully. The assistance could be as simple as reminding the avatar that the door was left open. To achieve so, the challenge is to determine the minimal amount of affordable sensors to be deployed in the simulation. The use case assumes the sensors are located over the avatar (chest - s_1, left hand - s_2, right hand - s_3). The generated sensor feed is processed by the assistive solution, or the validation component, the same way as a real sensor feed. With this information, it is possible to detect gestures with specific **trained classifiers**, such as *get up/get down hand*. The execution of the simulation with the analysis of the outputs made with the CEP infrastructure is presented in Fig. 2. The streams of

events are analyzed and the success of each step decided. Each validation step is associated with a time window at the right hand part of the figure. To the left, an actual 3D simulation snapshot is presented. To the right, a sample report generated by the unit test is presented where step 5 fails. Now, it would be up to the co-creation team to determine what assistive solution can solve this issue.

5 Conclusion and Future Work

This paper has introduced elements necessary to validate 3D simulations that capture a daily living issue and permit to address the features required from an assistive solution. The paper has presented the elements to be validated and illustrated the general approach using complex event processing technology combined with classifiers to analyze low level raw data. The result is validated in a 3D scenario where one of the validation steps fails and can only succeed when the assistive solution is attached. This necessary failure could be maintained along the development to ensure the assistive solution is really making a difference.

Acknowledgments. We acknowledge support from the project "Collaborative Ambient Assisted Living Design (ColoSAAL)" (TIN2014-57028-R) funded by Spanish Ministry for Economy and Competitiveness; and MOSI-AGIL-CM (S2013/ICE-3019) co-funded by the Region of Madrid Government, EU Structural Funds FSE, and FEDER.

References

1. Campillo-Sanchez, P., Gomez-Sanz, J.J.: A framework for developing multi-agent systems in ambient intelligence scenarios. In: Proceedings of the 2015 AAMAS Conference, AAMAS 2015, pp. 1949–1950 (2015)
2. Fernández-Isabel, A., Fuentes-Fernández, R.: Analysis of intelligent transportation systems using model-driven simulations. Sensors **15**(6), 14116–14141 (2015)
3. Gomez-Sanz, J.J., Campillo-Sánchez, P.: Domain independent regulative norms for evaluating performance of assistive solutions. Pervasive Mob. Comput. **34**, 79–90 (2017)
4. Gómez-Sanz, J.J., Pavón, J.: Meta-modelling in agent oriented software engineering. In: Garijo, F.J., Riquelme, J.C., Toro, M. (eds.) IBERAMIA 2002. LNCS (LNAI), vol. 2527, pp. 606–615. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36131-6_62
5. GRASIA: AIDE (2018). http://grasia.fdi.ucm.es/aide
6. Pallot, M., Trousse, B., Senach, B., Scapin, D.: Living lab research landscape: from user centred design and user experience towards user cocreation. In: First European Summer School "Living Labs" (2010)
7. Pax, R., Cárdenas Bonett, M., Gómez-Sanz, J.J., Pavón, J.: Virtual development of a presence sensor network using 3D simulations. In: Alba, E., Chicano, F., Luque, G. (eds.) Smart-CT 2017. LNCS, vol. 10268, pp. 154–163. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59513-9_16
8. Rashidi, P., Mihailidis, A.: A survey on ambient-assisted living tools for older adults. IEEE J. Biomed. Health Inf. **17**(3), 579–590 (2013)