Using Agent-Based Modeling and Simulation for Quality in Use Evaluation of Ambient Assisted Living Applications

Maria Paula Corrêa Angeloni[1][0000−0002−1129−5073], Emmanuelle Grislin-Le Strugeon[1,2][0000−0002−8429−4012], Káthia Marçal de Oliveira[1][0000−0001−8146−5966], Cristina Tirnauca[3][0000−0002−7129−2237], and Rafael Duque[3][0000−0001−8636−3213]

[1] UPHF, CNRS, UMR 8201 - LAMIH, Valenciennes, France
[2] INSA Hauts-de-France, Valenciennes, France
[3] Depart. de Matemáticas, Estadística y Computación, Universidad de Cantabria, Santander, Spain

Abstract. The Ambient Assisted Living (AAL) paradigm promotes ubiquitous systems that are intended to facilitate different activities for the end user (resident of that environment). This is even more impactful for elderly citizens or people with mobility issues, who tend to face more challenges to perform tasks than someone with no movement problems, as it allows them to age in their own personal space with technological aid. Evaluating the Quality in Use (QinU) of these applications is, therefore, highly necessary. Doing that in a classical way requires involving potential users actively testing the applications in place, which can be difficult for those with some sort of motor impairment. To answer this problem, we propose to evaluate the QinU through simulations using software agents to replace the user's explicit and implicit interactions with a software application in a simulated AAL. These simulations generate the necessary synthetic data to make a QinU evaluation possible by applying quality measures, therefore providing supporting data for a quality analyst to play their role. A proof of concept is presented as the feasibility study of this proposal.

Keywords: Ambient assisted living · Agent-based modeling and simulation · Quality in use.

1 Introduction

When evaluating the quality of applications used in smart contexts, such as Ambient Assisted Living (AAL), specific difficulties come from the interaction with the connected environment. The applications that are designed for these spaces usually take into account not only explicit interactions through the user interface (UI), but also implicit interactions using sensors data [11]. According to the ISO/IEC 25019:2023 [8] standard, Quality in Use (QinU) is defined as the "extent to which the system, when it is used in a specified context of use, satisfies or exceeds the stakeholders needs to achieve specified beneficial goals or outcomes".

In a previous study [2], we identified that the QinU evaluation of smart environment applications (such as the ones found in AAL spaces) usually utilize conventional procedures. These procedures consist in asking users to interact with the system while being placed in the environment (if necessary) in order to evaluate in-vivo their engagement with the surroundings, and then asking them to answer some usability questionnaire. The person in charge of the evaluation supervises the users throughout the activities that collect data and analyses the results of the questionnaire. However, this kind of evaluation may be difficult or even unsafe for the elderly and people who have reduced mobility. For this reason, we argue that it is essential to ensure the QinU as early as possible, before involving potential end users in the evaluation. Our proposal applies Agent-Based Modeling and Simulation (ABMS) to replace the "human" part in the human-computer interactions and generate synthetic data to measure QinU properties and, therefore, obtain a QinU evaluation of a software.

Our main research question in this paper is: can we measure the QinU of applications found in AAL without involving any real user? To that end, we define an approach and perform a proof of concept applied to a subset of human behaviors while interacting with an application in a simulated smart environment.

This paper is organized as follows: the next section presents a background on the subject, while Section 3 presents the outlines of the approach that is detailed in Section 4. Section 5 discusses the encountered difficulties and possible improvements, and Section 6 presents the conclusion and future work.

2 Background

2.1 QinU evaluation of smart environment applications

According to the ISO/IEC 25010:2011 standard [7], software quality can be measured with three different approaches: by internal properties, which typically refer to static characteristics of intermediate products; by external properties, which typically refer to the behavior of the application's code when it is executed; or by QinU properties, whether it is a real or a simulated use of the software system in a specific context of use. In this study, we are interested in measuring software quality by QinU properties in a simulated use.

The *context of use* is an essential element for the QinU evaluation. It is defined as a combination of users, goals, tasks, resources and environment. Different contexts of use take into consideration different users (hence, distinct behaviors) in the same environment, as well as the same user in different environments. For instance, if the user is performing the actions alone at home or performing them at a doctor's office accompanied by someone, these are different contexts of use, which impacts the QinU.

QinU is evaluated by measuring several characteristics [8], such as usability (which consists of effectiveness, efficiency and satisfaction), obtained as the outcome of the system usage under an identified context of use. The ISO/IEC 25022:2016 standard [9] provides a set of measures to evaluate these characteristics, as presented in Table 1.

Table 1. QinU measurements.

Characteristic	Measure	Description	Measurement function
Effectiveness	Completed tasks	The proportion of tasks that are completed	X = A/B; A = no. of subtasks that are completed without assistance; B = total no. of subtasks
Efficiency	Task time	The time taken to successfully complete all subtasks	X = T; T = task time

From the human-computer interaction (HCI) perspective, smart environments should consider implicit and explicit interaction. The explicit interaction refers to events purposefully triggered by the user, as for instance, when they are interacting with the system through the UI and it requires some sort of dialog between the user and the computer. Implicit interaction refers to actions from humans that are not seen as an interaction with a computer, but are still "captured, recognized and interpreted" regardless [11], and commonly happen in smart environments with events triggered by their sensors.

2.2 ABMS to simulate human behavior

Agent-Based Modeling and Simulation (ABMS) is well-adapted to represent phenomena that come from interaction behaviors at a micro-level, allowing "to take into account heterogeneity and complexity both in the individual layer and in the environment layer" [1].

Among the benefits of using ABMS when applied to systems that include humans [4], two established situations reflect our case: i) "when space is crucial and the agents' positions are not fixed", since we are interested in the person moving in a smart home; ii) "when the population is heterogeneous, when each individual is (potentially) different", since one of the objectives is to reflect different people, each with their own aspects and (possible) disabilities.

In a previous systematic study about QinU evaluation for smart environments [3], from 846 papers, we selected 15 for analysis and only two of them mentioned the use of agent-based simulations. The first one [5] created an agent-based simulation for an approach that evaluates Ambient Intelligence, where the agents represent different people performing various tasks at an airport. The second one [13] presented a simulator capable of reproducing the behavior of a person in terms of activities of daily living to create synthetic data from sensors deployed in a virtual environment. However, neither of them use ABMS to directly evaluate the software quality of applications present in smart environments.

3 The ABMS approach for QinU evaluation

Our approach to evaluate the QinU by simulation consists in applying ABMS to reproduce human actions interacting with software systems, which would eliminate the need to include people in a first evaluation of the QinU. Our goal is to simulate a person living alone in an AAL. This approach is divided into two steps, as seen in Fig. 1 and described in the following paragraphs.

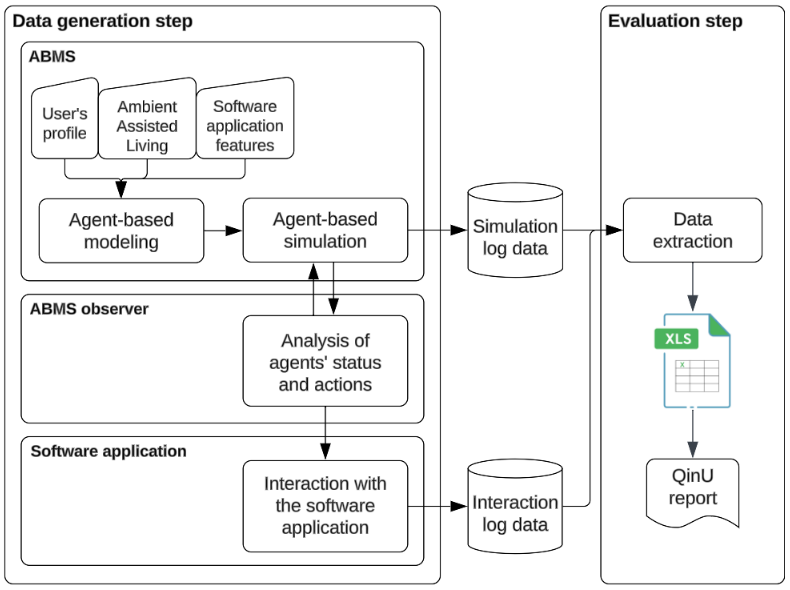


Fig. 1. The main components of the QinU evaluation by simulation.

Data generation step This step includes setting up the *agent-based modeling* before *simulating* possible implicit and explicit interactions with the software application. For the *agent-based modeling*, we need to consider the information of the *context of use*, which means the user's profile, the environment around them (i.e. the AAL), and the software application itself. The profile of the person usually found in an AAL can be made of a senior citizen with some sort of motor impairment, therefore, they will have specific needs that make them need a special support provided by software applications. These features of the profile are considered as parameters of the agents and should be set by the quality analyst (person responsible to conduct software quality assurance evaluations). For the *agent-based simulation*, the goal is to generate synthetic log data from the interactions between the agent and the software application to be evaluated. The content of the output data must be adapted to the purpose of the simulation.

The agent-based model includes elements representing the potential users' states and behaviors that are relevant regarding the interaction with the application that is being evaluated, as well as the relevant aspects of the environment in which they will use such software. However, the model will not recreate the entire targeted application to prevent errors in replicating its functioning. Our approach consists in simulating the agent's behavior while interacting with the software and its surroundings. The aim is to produce synthetic interactions of a user with the application along with the corresponding contextual data. These interaction events are then reproduced on the real application's interface to create a contextualized *interaction log data*, which stores information regarding the session identifier, the IP address, the timestamp for when the application is accessed and what actions are performed. On the other hand, the *simulation log data* stores information exclusively about what happens in the ABMS, such as the agent's location throughout the simulation, their age and gender, the time – in the simulation – when events happen, and "actions" that take place – such as falling down to the ground or using the application.

To create the connection between the simulation and the real application, an additional component is required, which we call *ABMS observer*. The goal of this mediator is to collect data from the simulation and send the relevant events to the *software application*. This key component combines the calls to the simulator and to the application, according to the functionalities and the simulated context, generating the *simulation* and the *interaction log data*.

Evaluation step This step consists of evaluating different QinU characteristics by using their respective measures from the ISO/IEC 25022:2016 standard [9]. The necessary data to calculate the measures are extracted from both the *simulation* and the *interaction log data*, producing a spreadsheet as an outcome that allows to produce the QinU report of the evaluated software application further used by the quality analyst.

4 Evaluation of a simple software application as proof of concept

To assess the feasibility of our approach, we designed and implemented a simple software application called HelpMe, recreating a feature from an application for people with Parkinson's disease[4].We implemented a webpage designed for mobile devices with the functionality of calling for help, as seen on the original application. This action is supported either by an explicit request of the user by clicking a button or when the user falls down (implicit interaction).

For the first situation (explicit interaction), the user presses a button to ask for help with the possibility of including a message, and the system sends out an email for the emergency registered contact. The second situation (implicit interaction) occurs when the gyroscope sensor of the mobile device detects that

[4] https://www.parkinsoncom.eu

the user has fallen down, so an automatic email is also sent to the registered contact. In both cases, the location of the user when asking for help is sent in the email thanks to the sensors found on-site.

4.1 The ABMS

The purpose of the ABMS is to simulate a person living alone – moving around and performing different actions – in an AAL and using the HelpMe application presented previously. Using NetLogo [14]we defined one agent that plays the role of a person in their home. The first step is to set the parameters of the context of use (user, environment and task).

Regarding the user, the agent's behavior is guided by a set of fundamental needs that determines the tasks to be performed, which includes eating, going to the toilet, taking a shower, and going to sleep. Once the agent has fulfilled its most urgent need, it moves forward to accomplish the next task depending on its other needs. The parameters applied to these needs, e.g. the meal duration, are present as constant values in the model.

Moreover, to define diverse users' profile, apart from age and gender, we also considered issues related to Parkinson's disease to reproduce the expected behavior. For that, we selected some of the symptoms described in the Parkinson's Well-Being Map [12] and added a set of parameters to modulate the agent's behavior as follows:

- "I feel the urge to pass urine": a switch, when turned on, will make the agent go to the toilet with a higher frequency;
- "I have slowness of movement (Bradykinesia)": a slider to determine the speed with which the agent should move around the house and perform actions, which will have an impact on the navigation speed when they are interacting with the software application;
- "I fall over": a slider to set the frequency with which the agent may fall down to the ground;
- "I have shaking (tremor)": a slider to set the frequency of the tremors that the user may have: the stronger they are, the more likely it is for the user to not be able to properly use the application.

Regarding the environment, the following aspects are modeled: the configuration of the house, which includes the structure, such as walls and doors, as well as the objects found in it, such as beds and drawers. For the active agents, there is the person that lives in the space, and 15 sensors around the house that receive information about where the person is at the moment. The design of the apartment was inspired by the real apartment and its sensors from the Aruba dataset in the CASAS data repository [6].

Regarding the task performed, its purpose is to use the application to call someone for help. For the explicit interaction, we broke it down into four subtasks so that we could get the proper data to apply in the QinU measurements: (i) accessing the application, (ii) writing a message, (iii) pressing a button to call out for the help of the primary contact, and (iv) pressing another button to call out for a secondary contact. Each one of these actions are saved in the *interaction log data* with timestamps for when they occur.

The agent's movement speed as well as the frequencies of their possible falls and tremors can be changed directly on NetLogo's UI. Moreover, the decision of using or not the application happens according to the outcome of a procedure that generates a random number, for which there is also a slider to determine the time period – in the simulation – for when this procedure should run. Since they are parameters, all these agent's characteristics can be modified according to the quality analyst's goals of evaluation.

4.2 The ABMS observer

The tools we used to develop the mediation between the ABMS and the application are based on Python. To create the link between the webpage and NetLogo, we used the pyNetLogo[5] library to enable the control of NetLogo from Python, which is "often used as a 'glue' language, meaning that it connects pieces of software written in different languages together" [10]. Hence, pyNetLogo was used to transmit commands from the observer to NetLogo and read information about the agent's status and actions inside the ABMS. Then, to access the webpage, we used the Selenium[6] library that allows performing tasks in different web browsers with Python.

As shown in Fig. 2, the observer sets up and starts the NetLogo model and also launches the software application that is going to have its QinU evaluated (in this case, our webpage). The simulation and the application become thus both active. The events occurring in the simulation cause changes in the variable states that are read by the observer, such as the agent's coordinates, the agent's state (in this specific case, whether they have fallen or not) and actions (in this case, if they have decided to interact with the software application). Based on this information, the observer can launch potential related actions towards the application. The decision that the user can make related to the software application is choosing to use the software application to call someone.

Regarding the explicit interaction, if the agent decides to use the webpage to call someone, they will take more or less time to complete this task according to the speed that they have been assigned in NetLogo's graphical UI: the slower the agent is in their movements, the longer it takes for them to open the web browser, access the web application, and click on the button to call for help. The more frequent their tremors are, the more likely they are to not complete the action, as they face more mobility difficulties. Besides, when performing this action, the agent can include a message in the text field found in the webpage (which is not required to be filled) before clicking on the button. When actual end users access the application, they can write whatever they wish; however, to include this possibility in the simulation, we prepared three different sentences, each one brought up for a different scenario. Therefore, there are four possible

[5] https://pynetlogo.readthedocs.io

[6] https://selenium-python.readthedocs.io/

outcomes for the explicit interaction, which happen in a random way once the agent decides to use the application: no message, "I've got a headache", "I broke something", and "I need help with my medication". The first one sends a default email letting the contact know that the user needs help, while the other three provide the inserted text in the webpage. These variants show the possibility to adapt to different inputs depending on the functionalities of the software application that is being evaluated.

Regarding the implicit interaction, if the agent falls down, the observer detects the change in the agent's status, similar to how a gyroscope sensor would work, and triggers the call. The higher the frequency the agent falls down, the more likely it is for an implicit interaction to be triggered. To represent this type of interaction where smart devices can trigger actions without explicit actions from the user, we have implemented a distinct and hidden button that is pressed on the webpage. For reasons of simplicity, we assumed the sensors placed in the environment do not present errors, as this is not part of the scope of our study.

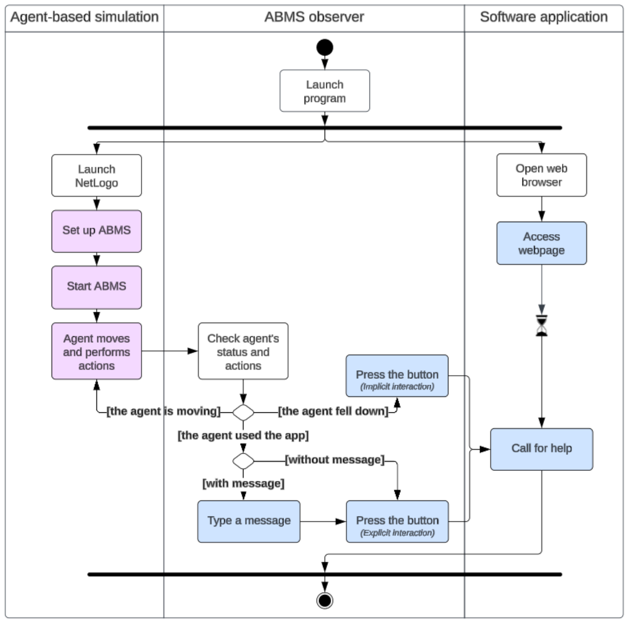


Fig. 2. Activity diagram of the steps taken to integrate the different applications.

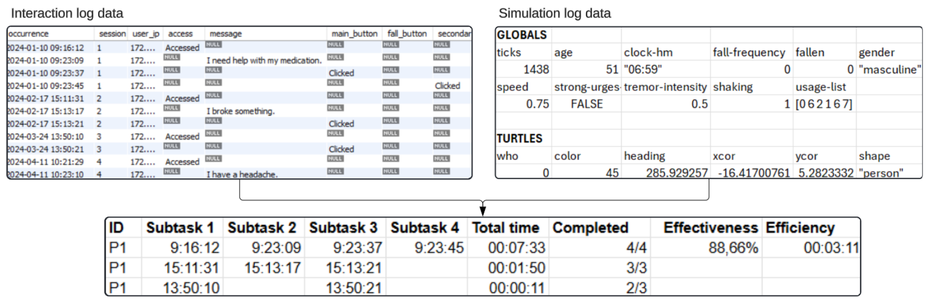
We have set up a MySQL database in a Raspberry Pi so that all the interactions are saved and can be checked afterwards. In Fig. 2, the activities in pink are saved in the *Simulation log data* from Fig. 1 and the ones in blue correspond to the *Interaction log data*. For the former, the saved data includes all the agent's characteristics, as their age, gender, and frequency of activities. For the latter, the recorded information includes timestamps for when the user accesses the webpage and for when each interaction happens, as well as its content – which buttons are clicked and if a message is included.

4.3 Calculating the QinU

For the proof of concept of this first study, we chose the two QinU measurements presented in Table 1 for effectiveness and efficiency. To collect the data for the measures, we used both log data to generate a spreadsheet file (see Fig. 3) including:

- *ID* is the agent's identifier – even though every simulation we ran presents only one agent that represents a person, each one of them presented a different combination of parameters to represent different people. Each one of them has its own set of characteristics, which can be found in the *simulation log data* such as age, gender, as well as the frequency of falling or shaking. Each one of these is stored as a global variable in NetLogo's log data;
- *Subtasks 1 to 4* are all part of the "asking for help" task. Their corresponding timestamps are extracted from the *interaction log data* by accessing the fields corresponding to the events (access, message, main_button, fail_button and secondary_button) that have non-empty values: Subtask 1 gets the timestamp of the "access" column, Subtask 2 from the "message" column, and so on;
- the *Total time* of the completed task is obtained by subtracting the initial time for when the first subtask took place from the time when the last subtask happened;
- *Completed* provides the number of subtasks that were successfully accomplished – getting that information from how many of the subtask columns have information – divided by the total number of subtasks. Writing a message and calling a secondary contact are not required to successfully use the application, therefore, the number of total subtasks varies;
- *Efficiency* provides the QinU measure by calculating the average time from all the values in the "total time" column performed by this person (according to their identifier);
- *Effectiveness* provides the QinU measure by calculating the average percentage of success that the user has accomplished when using the application (in this case, in the period of 24 hours in the simulation) according to the "completed" column.

Based on the data collected, the QinU report can be visualized with charts and explanations. For our simulation, we based the agents' characteristics in eleven real people interviewed for the ParkinsonCom project. Fig. 4 shows an example of chart generated in the QinU report for the *explicit interaction.* The agents are represented by their identifiers, P1, P2, and so on. They are ordered (X axis) according to their tremor frequency, which can be seen below their identifiers. By analysing this figure, it is possible to notice that:


Interaction log data

occurrence	session	user_ip	access	message	main_button	fall_button	secondar
2024-01-10 09:16:12	1	172....	Accessed	NULL	NULL	NULL	NULL
2024-01-10 09:23:09	1	172....	NULL	I need help with my medication.	NULL	NULL	NULL
2024-01-10 09:23:37	1	172....	NULL	NULL	Clicked	NULL	NULL
2024-01-10 09:23:45	1	172....	NULL	NULL	NULL	NULL	Clicked
2024-02-17 15:11:31	2	172....	Accessed	NULL	NULL	NULL	NULL
2024-02-17 15:13:17	2	172....	NULL	I broke something.	NULL	NULL	NULL
2024-02-17 15:13:21	2	172....	NULL	NULL	Clicked	NULL	NULL
2024-03-24 13:50:10	3	172....	Accessed	NULL	NULL	NULL	NULL
2024-03-24 13:50:21	3	172....	NULL	NULL	Clicked	NULL	NULL
2024-04-11 10:21:29	4	172....	Accessed	NULL	NULL	NULL	NULL
2024-04-11 10:23:10	4	172....	NULL	I have a headache.	NULL	NULL	NULL

Simulation log data

GLOBALS					
ticks	age	clock-hm	fall-frequency	fallen	gender
1438	51	"06:59"	0	0	"masculine"
speed	strong-urges	tremor-intensity	shaking	usage-list	
0.75	FALSE	0.5	1	[0 6 2 1 6 7]	
TURTLES					
who	color	heading	xcor	ycor	shape
0	45	285.929257	-16.41700761	5.2823332	"person"

ID	Subtask 1	Subtask 2	Subtask 3	Subtask 4	Total time	Completed	Effectiveness	Efficiency
P1	9:16:12	9:23:09	9:23:37	9:23:45	00:07:33	4/4	88,66%	00:03:11
P1	15:11:31	15:13:17	15:13:21		00:01:50	3/3		
P1	13:50:10		13:50:21		00:00:11	2/3		

Fig. 3. Interaction (top left) and simulation (top right) log data and the spreadsheet (bottom center) calculating the QinU measure with data that comes from them.

- the effectiveness is higher for people that do not present tremors. Consequently, the quality analyst can learn that the application is not very well adapted for people who suffer with this;
- about the efficiency, the measures confirm that the slower the agent is in their movements, the longer it takes for them to perform each subtask while using the software application (see, for instance, P7 and P8's orange bar);
- P8 and P9 have same effectiveness, but since P8 has slower movements, they got low efficiency. The measures provide an average number that try to mirror real life; even though these two agents present a slight difference in their tremor frequency, both were able to achieve the same measure for effectiveness in the application (with different efficiency), just as different people might conquer the same tasks with different degrees of disabilities.

With the QinU report in hand, the quality analyst can decide if the application should be improved or not.

For the *implicit interaction*, we assumed perfect sensors with no noise, which means that every time the agent falls, the call is triggered. As this is executed according to parameters, it can be adapted. However, because of that, this present example calculates the effectiveness for the explicit interactions only.

5 Discussion

Contrary to what happens in the case of digital twins or models, the purpose here was not to simulate a physical system, but an environment and a contextual use for a software system. Moreover,we chose to seek for a method that would avoid any replication of the tested application inside the simulation. Consequently, our approach had to detect from outside of the simulator any possible changes in the components' states of the simulation. The techniques we used for the proof of concept, based on NetLogo associated with Python libraries, provide a solution to transmit inputs to the software application. However, this solution is not prepared to include responses from the software back to the simulation: even

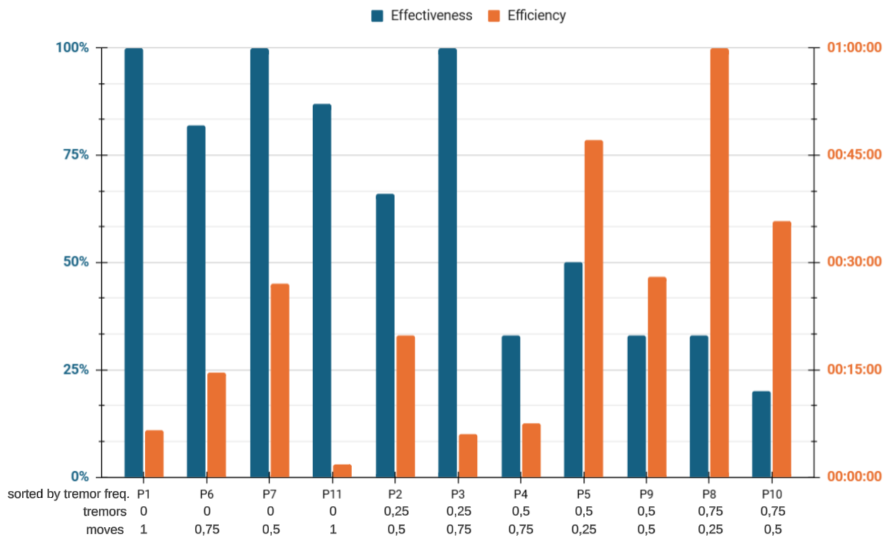


Fig. 4. Chart presenting both the effectiveness and efficiency of the application for each user, as well as their tremor frequency and movement speed.

though it is possible for Selenium with Python to notice changes in the webpage, an investigation is necessary for the interference with a running simulation.

Regarding specifically the ABMS, the main issue was to properly represent people's characteristics and determine how often the actions should happen. Another point of difficulty that has been encountered and will have to be dealt with in the future is the time difference between the ABMS and the real time, which is the time that the third-party software applications run at.

Regarding the plausibility of the model, the agent's actions occur randomly in this first study. Machine learning techniques can be used to ground our work in real patterns, providing a file to be read by the ABMS and determining how the agents act.

Finally, we set the variations in the context of use by defining the parameters in the ABMS. All the user profile information and the frequency of use of the application by the agent can be set generating different executions, and therefore, different results to be provided for the quality analyst to evaluate the QinU.

6 Conclusion

In this paper, we have proposed an approach for evaluating the QinU using ABMS. For that, agents simulate human actions in an AAL simulated environment and interact with the software application. Data is collected and quality measures are calculated. By replacing people with agents in this approach, it is possible to create multiple contexts of use, representing different people in different places. This promotes various scenarios for the QinU to be evaluated as accurately as possible, trying to predict many combinations for the use of the software system that needs to have its quality measured. We are currently working with two fronts: i) moving the agent around in the environment according to patterns derived from real data [6] using machine learning techniques and ii) calculating other measures of QinU.

Acknowledgements This study is partially supported by the Hauts-de-France Region, by the REUNICE project funded by the European Union and the CNRS, by the Gov. of Cantabria through projects SUBVTC-2023-0013 and VP84, and by the project PID2022-139237NB-I00 financed by MICIU and FEDER, UE.

References

1. Adam, C., Gaudou, B.: Bdi agents in social simulations: a survey. Knowledge Engineering Review **31**(3), 207–238 (2016)
2. Angeloni, M.P.C., Duque, R., Oliveira, K.M.d., Grislin-Le Strugeon, E., Tirnauca, C.: A tertiary study on quality in use evaluation of smart environment applications. In: Research Challenges in Information Science. pp. 115–130. Springer (2024)
3. Angeloni, M.P.C., Oliveira, K.M.d., Grislin-Le Strugeon, E., Duque, R., Tirnauca, C.: A review on quality in use evaluation of smart environment applications: What's next? In: Proc. ACM SIGCHI Symp. on Engineering Interactive Computing Systems. p. 9–15 (2023)
4. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. Proc. of the National Academy of Sciences of the USA **99**(Suppl.3), 7280–7287 (2002)
5. Carbo, J., Sanchez-Pi, N., Molina, J.M.: Agent-based simulation with netlogo to evaluate ambient intelligence scenarios. Journal of Simulation **12**(1), 42–52 (2018)
6. Casas: Center of advanced studies in adaptive system, www.casas.wsu.edu/datasets/, accessed: 21/02/2024
7. ISO/IEC: 25010:2011. systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models (2011)
8. ISO/IEC: 25019:2023. systems and software engineering — systems and software quality requirements and evaluation (square) — quality-in-use model (2011)
9. ISO/IEC: 25022:2016. Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — Measurement of quality in use (2016)
10. Jaxa-Rozen, M., Kwakkel, J.H.: PyNetLogo: Linking NetLogo with Python. J. of Artificial Societies and Social Simulation **21**(2), 4 (2018)
11. Schmidt, A.: Interactive context-aware systems interacting with ambient intelligence. Ambient intelligence pp. 159–179 (2005)

12. UCB: Parkinson's well-being map (2016), www.ucb.com/_up/ucb_com_patients/documents/NeuproPSP_Parkinsons_Wellbeing_Map_Update_L2.pdf
13. Veronese, F., Masciadri, A., Trofimova, A.A., Matteucci, M., Salice, F.: Realistic human behaviour simulation for quantitative ambient intelligence studies. Technology and Disability **28**(4), 159–177 (2017)
14. Wilensky, U.: NetLogo. Center for connected learning and computer-based modeling, northwestern university. Evanston, IL. http://ccl.northwestern.edu/netlogo/ (1999)