Quality Traceability for User-centric Context-aware Systems in Intelligent Environments

Nawa Sakanga
Dep. of Computer Science
Middlesex University
London, UK
ns1314@live.mdx.ac.uk

Juan C. Augusto
Dep. of Computer Science
Middlesex University
London, UK
J.Augusto@mdx.ac.uk

Lindsey Brodie
Dep. of Computer Science
Middlesex University
London, UK
L.Brodie@mdx.ac.uk

Lisa Marzano
Dep. of Psychology
Middlesex University
London, UK
L.Marzano@mdx.ac.uk

***Abstract*—Context-awareness is an important component of modern software systems. For example, in Ambient Assisted Living (AAL), the concept of context-awareness empowers users by reducing their dependence on others. Due to this role in healthcare, such systems need to be reliable and usable by their intended users. Our research addresses the development, testing and validation of context-aware systems in an emerging field which currently lacks sufficient systems engineering processes and disciplines. One specific issue being that developers often focus on delivering a system that works at some level, rather than engineering a system that meets a specified set of system requirements and their corresponding qualities. Our research aims to contribute towards improving the delivery of system quality by tracing, developing and linking systems development data for requirements, contexts including sensors, test cases and their results, and user validation tests and their results. We refer to this approach as the "quality traceability of context-aware systems". In order to support the developer, the quality traceability of context-aware systems introduces a systems development approach tailored to context-aware systems in intelligent environments, an automated system testing tool and system validation process. We have implemented a case study to inform the research. The case study is in healthcare and based on an AAL system used to remotely monitor and manage, in real time, an individual prone to depressive symptoms.**

Keywords—IoT; context-awareness; intelligent environment; testing; validation; quality traceability; smart home; user-centric systems

I. Introduction

In recent years, computing has seen a significant increase in the rise of context-aware systems stemming from technologies such as the Internet of Things (IoT) [1], Intelligent Environments (IE) [2], Ubiquitous Computing [3] and Ambient Intelligence, AmI, [4]. The concept of context-awareness has enhanced user experience and enabled new directions such as Ambient Assisted Living, AAL, [5]. AAL provides systems comprising sensors, actuators, networks, computers and software applications for healthcare. They can be used for preventing or monitoring or improving wellness and managing health conditions in affected individuals [6]. Such systems empower users and reduce their dependence on others. However, this does imply a need for system quality: such systems have to provide usability and be reliable [7]. Appropriate system development and testing underpins the delivery of such systems.

Due to the uniqueness of context-aware systems, we propose a system development approach tailored to context-aware systems in intelligent environments (IEs), an automated testing tool and validation process. The proposed system development approach is a user-centred approach that guides the developer to the specification of requirements and contexts and definition of context testing tables. The automated testing support tool has the capability to generate test cases and enables adequate regression testing. The testing tool assists the developer to explore test cases more relevant to the context under consideration, thereby focusing more on the quality of the system under development rather than test coverage and has been seen to handle the issue of potential sensor combinatorial explosion. The proposed validation process is capable of identifying and tracking the quality of system responses to contexts from identification of requirements, through to the development, testing and validation phases. The system development approach, testing tool and validation process link requirements with testing, testing with validation and validation with requirements. We refer to this as *quality traceability of context-aware systems*.

The aims are to guide developers through the system development cycle, provide improved testing (increased test case generation productivity with focus on requirement and context pairs of interest and provide better test coverage) and validation process for better system quality. Consequently, this will have a bearing on the overall quality of the resulting system.

To support this research, we have implemented a case study based on an AAL system that aims to remotely monitor and manage depressive symptoms, in real time for an individual living in a smart home. We call this system the Depression Monitoring and Management System (DMMS). The DMMS has been developed, tested and validated using our proposed system development approach, testing support tool and validation process respectively.

A collaboration with the Middlesex University Department of Psychology has benefited the DMMS by allowing the selection of more meaningful evidence-based requirements and contexts of interest. The defined system requirements and contexts both have a bearing on the development, testing and validation of the context-aware system. The DMMS is used as a test bed in both simulation and real smart home environment. Results of the study show a sustainable approach that not only guides the development process for context-aware systems, but also brings about increased efficiency associated with developing, testing and validating context-aware systems.

Note the definitions used in this current paper are that testing is conducted in simulation mode to inspect the correctness of the system specification at programming level whereas validation happens in a real smart environment.

II. Problem Statement

This section highlights relevant work reported in the literature connected with our topic of research.

A. System Development Approaches

The IE community has commenced an ongoing discussion regarding the adoption of best practices that have evolved over the years from within the software engineering domain [8]. The movement is still in its infancy and the early focus has been on application of formal methods and model checking in particular, in order to increase the reliability and robustness of software systems [9].

Context-aware systems use IoT devices that add additional complexity into their development and implementation. Current system development approaches do not adequately and explicitly address the uniqueness of context-aware systems in intelligent environments [10]. Our proposed approach is tailored to such context-aware systems, given that it considers the inclusion of context features, and provides a system development approach, an automated system testing tool and validation process to support the developer.

B. *Testing and Validation of Context-aware Systems*

Testing and validation of systems help developers to verify and validate that they have built the required system. It is expected that the implemented systems satisfy context-aware requirements taking into consideration system qualities such as ease of installation, reliability, performance, security, compatibility and recovery after a failure. The integration of context-awareness capabilities into applications allows them to leverage contextual information to provide additional services while maintaining an acceptable quality of service. Several approaches to achieve this are now being adopted.

Heng et al. [11] proposed a novel family of testing criteria to measure the comprehensiveness of context-aware tests. Almeida et al. [12] discuss the relationship between requirements and elements of a solution and call it 'requirements traceability'. This is used during acceptance testing to assess the extent to which a system adheres to its requirements.

Wang et al. [13] introduced an approach to improve the test suite for context-aware applications that identifies key context-aware program points (capps) where context information can effectively affect the application's behaviour.

Mirza and Khan [14] proposed an approach for behaviour modelling of context-aware applications by extending Unified Modeling Language (UML) activity diagrams.

Augusto et al. [15] reported on an attempt to define a new method of context-aware systems testing and validation. They made a first attempt at linking contexts with test through a table.

From the reviewed literature, we identified the need to consider the uniqueness of context-aware systems concerning their development, testing and validation approaches. Current proposals in the literature are seen to aim to link contexts with testing but they do not go sufficiently far in helping developers and specifically do not support these concepts with tools, thereby not being practically useful. Our overall research aims at addressing both testing and validation and completing the full circle from requirements to contexts, contexts to tests, tests to validation and validation back to requirements. At this stage the research focuses on the first half of the loop with an emphasis on testing.

C. *Application of context-aware systems in healthcare*

There is a staffing crisis in healthcare today due to at least three major issues: doctor shortages worldwide, the aging and burnout of physicians and a higher demand for chronic care [16].With the increase of life expectancy, the population over 65 years is expected to double by 2030, and the number of chronic illnesses, the demand towards the healthcare system is also constantly growing. As a result, the lack of access to care and the differing quality are general worldwide[17]. Therefore, we recognise the need to provide sustainable solutions to the health sector addressing system quality (*usability, availability, accessibility and acceptability among others)* as a priority.

III. Proposed Method

A. *Proposed Quality Traceability*

In order to record the desired requirements and contexts of interest for a context-aware system, we adopted the User-Centric Intelligent Environment development process (UCIEDP) approach [18] and propose a quality traceability. The proposed process is user-centric and presents a complete cycle that allows the system development process to be carried out in a loop until the desired outcome is accomplished. Figure 1 shows how the Quality Traceability (QT) fits into the UCIEDP. The approach complements the UCIEDP by providing a step by step approach to the development of context-aware systems. QT shows stakeholders at the centre of the development process and the three main development phases of the approach. The key stakeholders play a vital role in the successful definition of functional requirements and contexts of interest. Emphasis of the system development process is placed on the interaction between the developer(s), and users and the other key stakeholders so as to specify a system that the users need. The first phase, initial scoping, starts with engaging key stakeholders to define and record requirements and contexts. The initial phase also includes prioritising requirements and contexts and thereafter creating requirement and context pairs. Successful specification of requirements and contexts is achieved through several iterations. Once the requirements and contexts have been agreed upon, the system design is developed and the developer then writes the system pseudocode.

The second phase is the main development phase when context testing tables (CTTs) are developed. The CTTs form the basis on which test cases shall be generated. Thereafter, using an activity recognition application called MReasoner [19], the pseudocode is translated into a meaningful MReasoner specification. This is followed by system testing which starts with the automatic generation of test cases. These test cases are written to the MReasoner specification file and a copy kept in a text file. The third phase is the IE Installation phase where validation takes place. Validation is done in a real smart home environment. The approach is iterative in nature, thereby enabling requirements and contexts to be revisited as need arises.

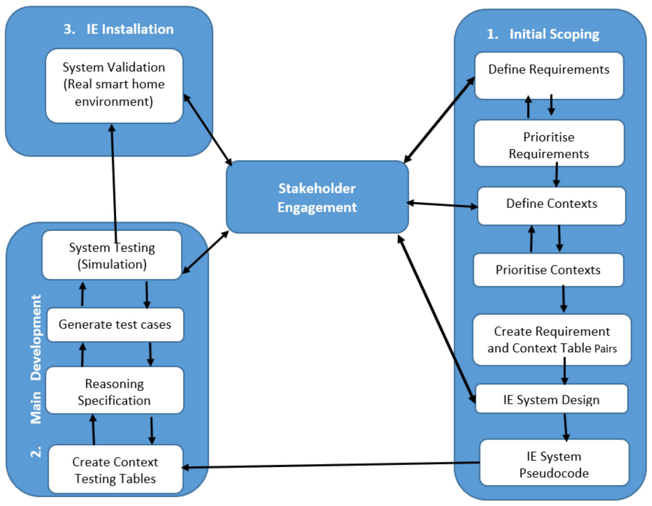


Fig. 1. Combined Quality Traceability (QT) in UCIEDP

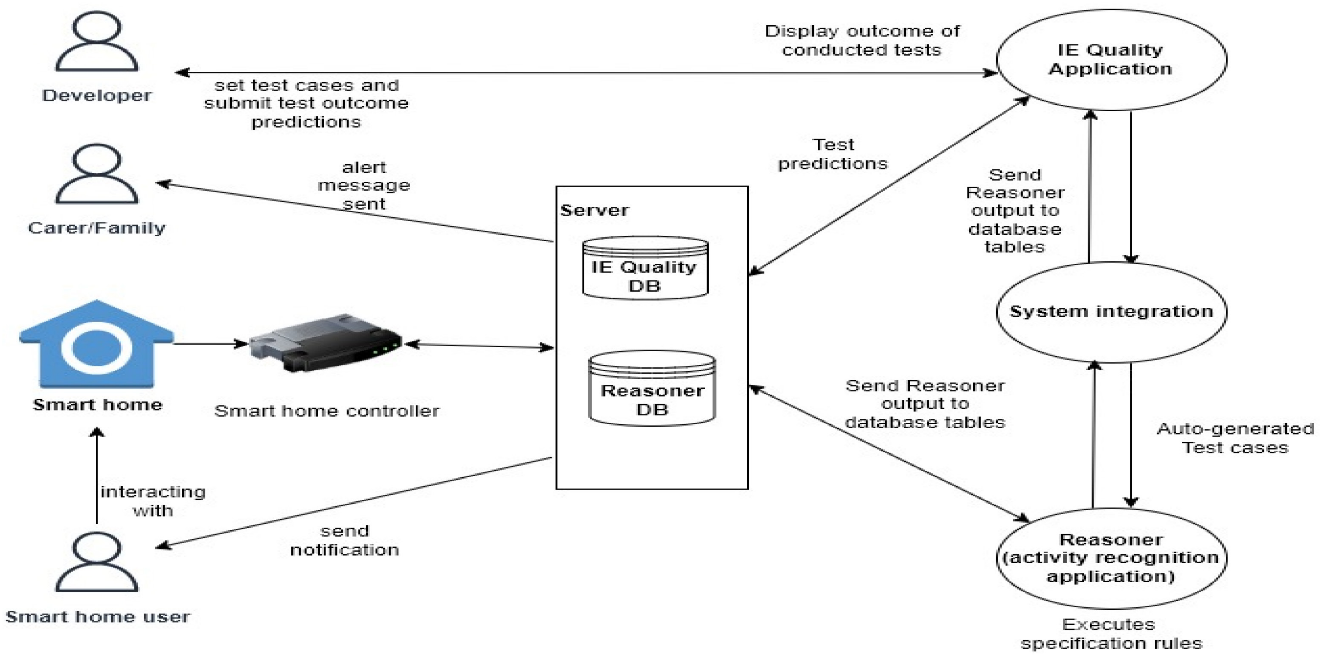

Developer
Carer/Family
Smart home
Smart home controller
Smart home user
set test cases and submit test outcome predictions
Display outcome of conducted tests
alert message sent
interacting with
send notification
Server
IE Quality DB
Reasoner DB
Test predictions
IE Quality Application
Send Reasoner output to database tables
System integration
Send Reasoner output to database tables
Auto-generated Test cases
Reasoner (activity recognition application)
Executes specification rules

Fig. 2. IE Quality Traceability System Overview

B. *Overview of IE Quality Traceability*

Figure 2 shows the overview of the IE Quality Traceability System (IEQTS). The IEQTS aims to develop, trace and link systems development data for requirements, contexts including sensors, test cases and their results, and user validation tests and their results, thereby improving the overall system quality. To achieve this, we developed the IE Quality Traceability Application as a testing tool to support the developer. The application facilitates automated testing in context-aware systems. Additionally, it contributes to the overall aim of linking requirements to testing, testing to validation and validation to requirements, thereby closing the circle as depicted.

The IEQTS comprises of a smart home controller, IE Quality Application, Reasoner and a server. The server houses the IE Quality database and reasoner database. The IE Quality database stores all tables related to the testing tool. The smart home controller records activities in the smart home as they are being generated. This information is stored in the Reasoner database. The two databases are linked. This configuration assists and assures the developer of the performance, stability and correctness of the developing system and contributes to improving the quality of the emerging system.

The figure shows how the developer interacts with a smart home via the IE Quality application (testing tool) which integrates with the system under development. The IEQTS allows the developer to develop, trace, link systems development data and record the context-aware system information (that is, the requirements and the context information) and develop test and validation cases and their results.

IV. Case Study – Depression Monitoring System

A. *Depression Monitoring and Management System*

AAL systems have the potential to ease staffing shortages in healthcare by facilitating diagnostics, decision making and administration whilst handling technological, ethical and legal obstacles [16]. To examine our proposed IEQTS method, we developed a test bed, the Depression Monitoring and Management System (DMMS), as a case study. DMMS is a user-centric context-aware system implemented in a smart home in our Smart Spaces laboratory. It is used to detect, monitor and manage the behaviour of an individual at risk of depression and has the ability to send alert messages and notification to both the smart home user and the carer(s) as need arises. In consultations with the Department of Psychology, the following behaviours were selected for monitoring depressive symptoms: sleeping patterns; eating patterns and overall activity levels [20].

The case study as described in this paper focuses only on monitoring of sleep patterns and disturbances. In order to achieve this context monitoring, the three sensors considered are *bed pressure pad, Passive Infrared Sensor (PIR)* and *light actuator*. The *bed pressure pad* is a sensor pad that is placed on the bed and it is activated when it detects pressure on it. The *PIR sensor* detects motion in the bedroom. Finally, the *light actuator* is responsible for turning the light *on/off.* The sensors that define eating and overall activity patterns are not included in this paper.

B. *Smart home technology*

For validation purposes, we use a real smart home. This smart home has several hardware elements installed including the required sensors for this research, a server and smart home controller (Vera hub).
The communication protocol used to manage the sensing environment is Zwave. Note additionally, the DMMS has a database that stores information (sensor states) recorded by the hub. The smart home infrastructure shown in figure 3.

C. *Activity Recognition Application - MReasoner*

The DMMS system was developed using an activity recognition application called MReasoner [19] which is developed in Java. MReasoner offers the mechanism to retrieve and gather data from a smart environment in real time, and then based on contextual reasoning instructions set up within it, to trigger appropriate actuation in the smart environment. The MReasoner specification is available on https://figshare.com/s/379ff6b786bcec324d98. Note that other applications could be used instead of MReasoner.

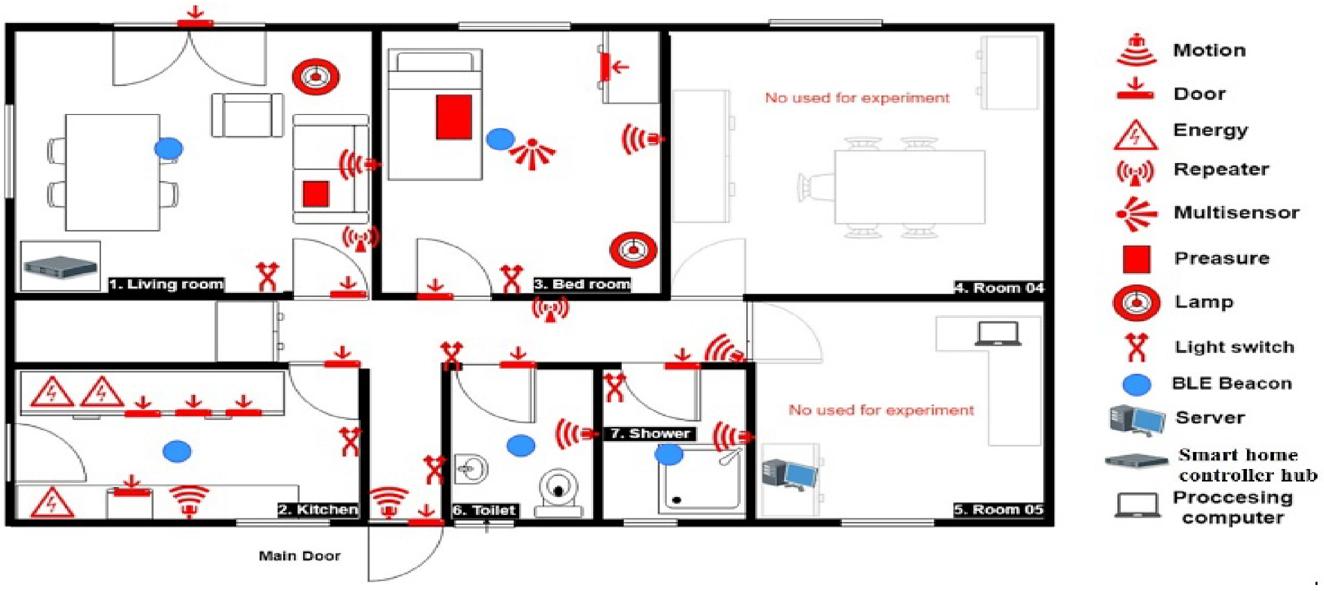


Fig. 3. Smart home infrastructure

D. *Description of the IEQTS – Testing Tool and Validation*

Within context-aware systems, it was identified that there was a need to establish: the data required to specify requirements adequately; the data to link requirements to contexts; the data to link requirements to test cases and the data to help ensure test coverage. Tables were devised to record this information.

E. *Recording Requirements and Contexts*

The DMMS system requirements were specified, with particular attention being given to identifying the contextual information. To record these context-aware requirements, two tables were created. Table I, the requirements table records the requirements using requirements label (RL) to uniquely identify the requirement, requirements name, requirements description and priority. Table II, the contexts table records the contexts, using context label (CL) to uniquely identify a specific context situation, context name to identify the context, context description and context priority. The priorities for both the requirements and the contexts range from 1 to 4 and refer to mandatory, high, medium and low respectively.

TABLE I. REQUIREMENTS TABLE

RL	Requirement	Requirement description	Priority
FR1	System shall monitor sleeping pattern	The system saves "interrupted sleep" action in database The system saves "prolonged sleep" action in database The system saves "normal sleep" action in database	1
FR2	System shall notify the user of prolonged inactivity	The user could be sitting in the living room watching TV for a prolonged period	2
FR3	System to monitor the eating pattern	Notify user if meals are skipped	1

Once the requirements and contexts of interest have been recorded, the developer pairs them as shown in Table III, the requirement and context pairs table. This table links requirements to contexts and also includes the context feature (CF) being tested, the situational parameter (SP) which is the situation being monitored, and the corresponding sensor (S) responsible for tracking the context.

TABLE II. CONTEXT TABLE

CL	Context	Context description	Priority
CL1	sleep	User has slept for prolonged period	1
CL2	sleep	User experienced interrupted sleep	1
CL3	watchTV	User sitting in living room watching TV for a prolonged period of time	2
CL4	eat	User eating pattern	1

TABLE III. REQUIREMENT AND CONTEXT PAIRS

RL	CL	CF	SP	S
FR1	CL1	When agreed pattern of prolonged sleep is observed, a message is sent to the caregiver	User in Bedroom Lying on bed	PIR BedPad Clock Light
FR1	CL2	When agreed pattern of interrupted sleep is observed, a message is sent to the caregiver	User in Bedroom	PIR BedPad Clock

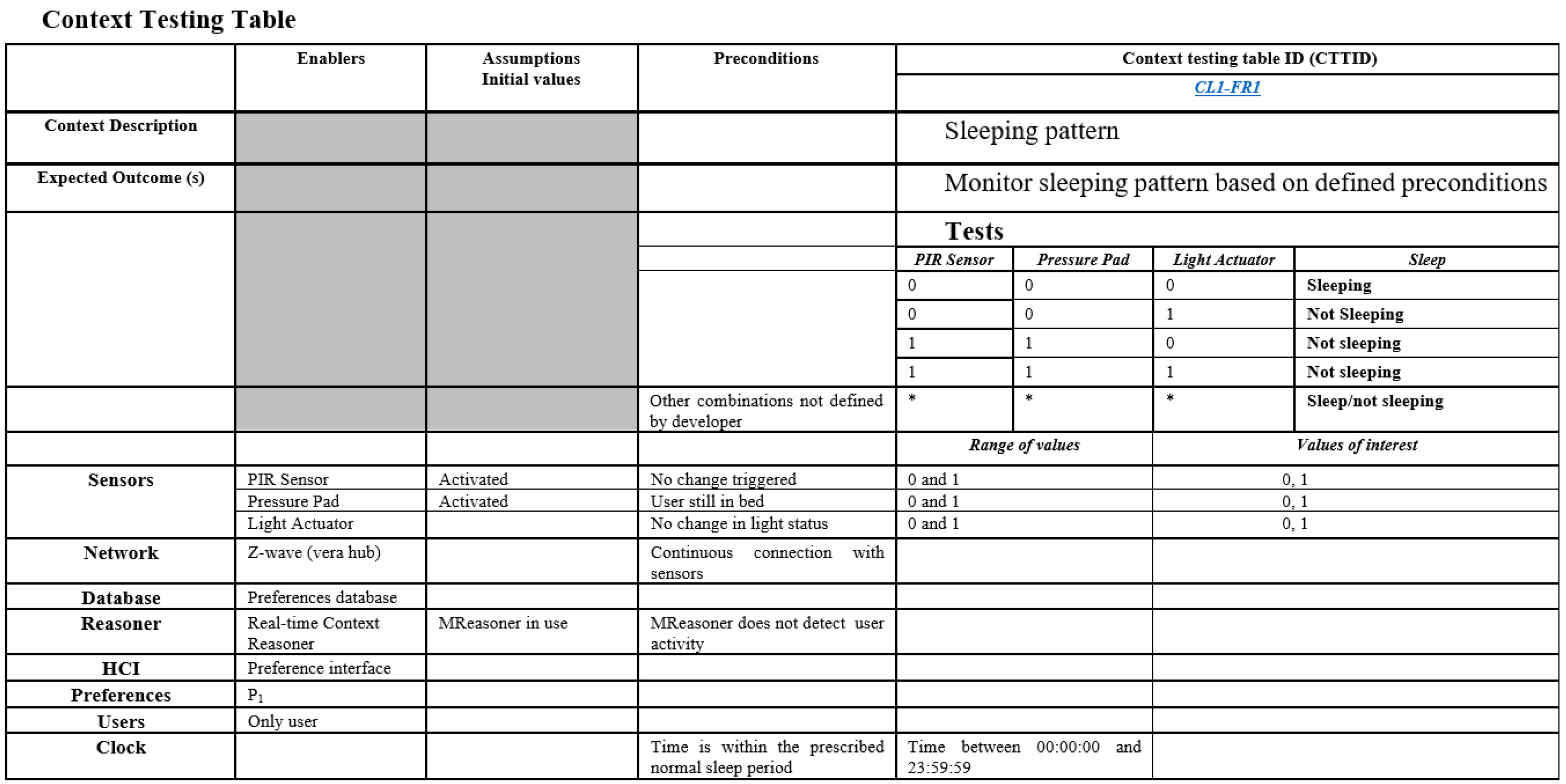
F. *Development of Test Cases and Recording of Their Result*

After requirement and context pairs have been recorded, once again, a table was devised to highlight interesting potential test cases, this table is called a context testing table (CTT).

Context testing tables (CTT) are seen as an important part of the support for context-aware system testing and validation. CTTs are completed by developers. Each CTT has a unique context testing table identifier (CTT ID), that helps identify the table. The information recorded in the CTT is based on agreed system requirements and contexts. Rather than aiming only for test coverage, the CTT aims to record cases of interest and they also focus on more critical combinations.

Table IV shows an example of a CTT. This CTT provides combinations of the sensors required to monitor a context and has valuable information about the context-aware system infrastructure.

TABLE IV. CONTEXT TESTING TABLE



Context Testing Table

	Enablers	Assumptions Initial values	Preconditions	Context testing table ID (CTTID)			
				CL1-FR1			
Context Description				Sleeping pattern			
Expected Outcome (s)				Monitor sleeping pattern based on defined preconditions			
				Tests			
				PIR Sensor	***Pressure Pad***	***Light Actuator***	***Sleep***
				0	0	0	**Sleeping**
				0	0	1	**Not Sleeping**
				1	1	0	**Not sleeping**
				1	1	1	**Not sleeping**
			Other combinations not defined by developer	*	*	*	**Sleep/not sleeping**
				Range of values		***Values of interest***	
Sensors	PIR Sensor	Activated	No change triggered	0 and 1		0, 1	
	Pressure Pad	Activated	User still in bed	0 and 1		0, 1	
	Light Actuator		No change in light status	0 and 1		0, 1	
Network	Z-wave (vera hub)		Continuous connection with sensors				
Database	Preferences database						
Reasoner	Real-time Context Reasoner	MReasoner in use	MReasoner does not detect user activity				
HCI	Preference interface						
Preferences	P_1						
Users	Only user						
Clock			Time is within the prescribed normal sleep period	Time between 00:00:00 and 23:59:59			

Note: * refers to any boolean number (allowing for other possible combinations not shown in the table)

The CTT provide a major resource for test cases (which also cover validation tests) and for the MReasoner. MReasoner implements part of the DMMS System prescribed in Table IV, in this case providing the rules that track the sleeping pattern.

V. Testing And Validation

The development team is in charge of the testing and validation and always has an option to revisit the defined requirements and contexts. The output of testing and validation are test and validation reports, respectively.

A. *Testing Tool*

The testing tool has an interface that enables automatic generation of test cases. Test cases can be generated by the developer or the system. The developer generates test cases by highlighting, to the random test generator, focal points of interest. We approach that from the position that not all values need to be explored and the developer will know which values are reasonable to consider for that specific context being investigated. Another option is to let the system generate test cases. In this case, possible sensor values are inputted and the system generates test cases based on the supplied information. See figure 4.

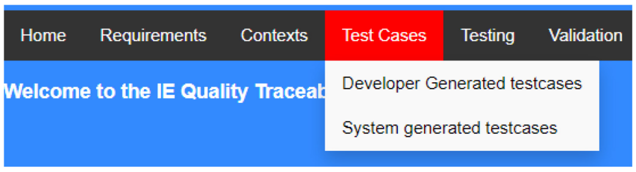

Home
Requirements
Contexts
Test Cases
Testing
Validation
Welcome to the IE Quality Traceab
Developer Generated testcases
System generated testcases

Fig. 4. Menu for test case generation in IE Quality Traceability Software

The general test cycle is depicted in figure 5. The test cycle shows that once requirements and contexts have been identified and implemented in the context-aware system, the IE Quality Traceability software generates test cases. Thereafter, these test cases are written to the MReasoner specification file and executed accordingly. Failed tests are identified and re-run after taking the necessary corrective measures.

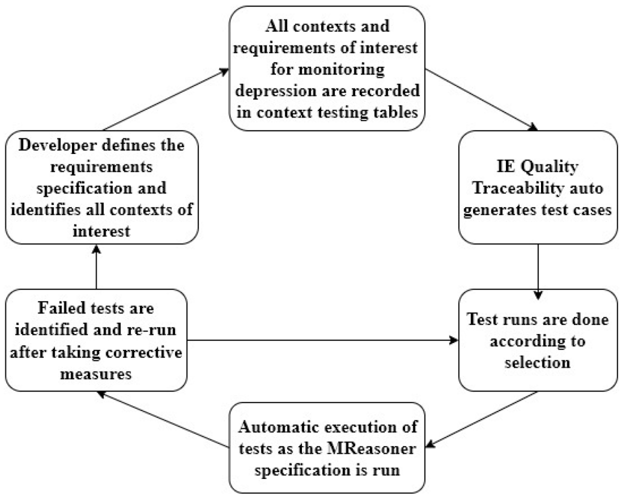

All contexts and requirements of interest for monitoring depression are recorded in context testing tables
IE Quality Traceability auto generates test cases
Test runs are done according to selection
Automatic execution of tests as the MReasoner specification is run
Failed tests are identified and re-run after taking corrective measures
Developer defines the requirements specification and identifies all contexts of interest

Fig. 5. IEQT Test cycle

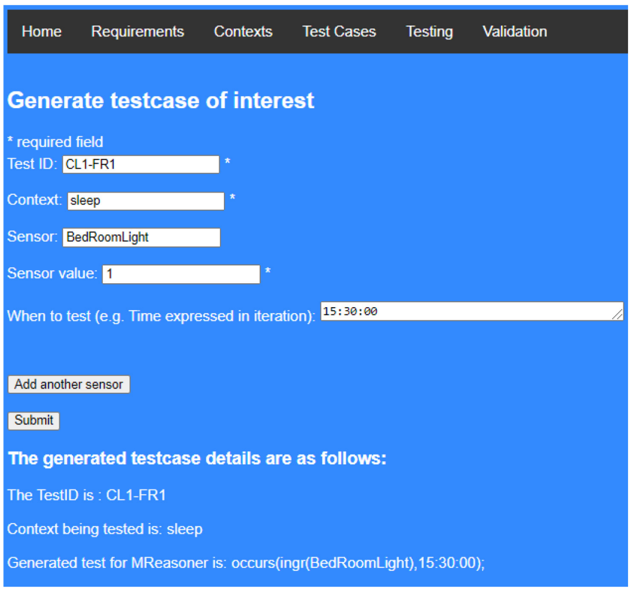
B. *Generating Test Cases*

An important concept in our test generation strategy is the human machine cooperation where a developer identifies the required testing either by specifying test cases or by instructing a random test generator. Generally, the testing process always involves choices over the test cases executed. For context-aware system testing, such choice becomes even more necessary due to the nature of the sensors involved. Some sensors (e.g. PIRs, light switches, device operation sensor and pressure sensors) have boolean values whereas other sensors have a richer range of values. For example, temperature, luminosity, noise levels, and other non-boolean values, usually a set of real numbers. This last group presents a potential challenge in terms of combinatorial explosion. We approach that from the point that not all values need to be explored and the developer will know which values are reasonable to consider for the specific context being investigated. For example, while temperature sensors may have a range of values associated to them, developers can choose to test the system with values '8.0', '16.0' and '24.0' as representatives of situations when users may perceive the environment to be cold, mild or hot respectively.

The overall process of producing context-related tests is as follows:

1) *Developer generated test cases*

One option of generating the test cases is by the developer. Using the drop down menu from the Test Cases menu, the developer clicks on *developer generated test cases*. This menu allows the developer to define the values of interest to create the specific tests. These are first recorded in the CTT and later recorded into the testing tool using an interface. This is done for all the sensors associated with a test. The generated tests are then written to the MReasoner specification file and a copy is also written to a text file. This text file keeps a copy of the generated test cases in order that they can be reused. Figure 6 shows the interface used by the developer to generate test cases of interest.



Generate testcase of interest

* required field

Test ID: CL1-FR1 *

Context: sleep *

Sensor: BedRoomLight

Sensor value: 1 *

When to test (e.g. Time expressed in iteration): 15:30:00

The generated testcase details are as follows:

The TestID is : CL1-FR1

Context being tested is: sleep

Generated test for MReasoner is: occurs(ingr(BedRoomLight),15:30:00);

Fig. 6. Developer generated test cases

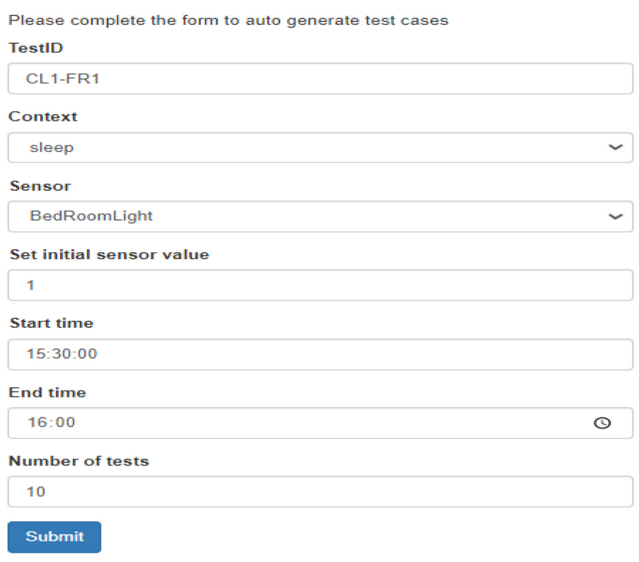
2) *System generated test cases – Simulation mode*

The second option of generating test cases is to set the system to randomly create them. This is depicted in figures 7 and 8. This process is as follows:

Step 1: Input test details: the developer selects the CTT ID, context and sets initial sensor values. Then they specify the test start time, end time and number of test sequences required.

Step 2: Set time frequency boundary: the developer sets the frequency boundary for test generation e.g. generate test cases within a frequency boundary of 30 seconds.

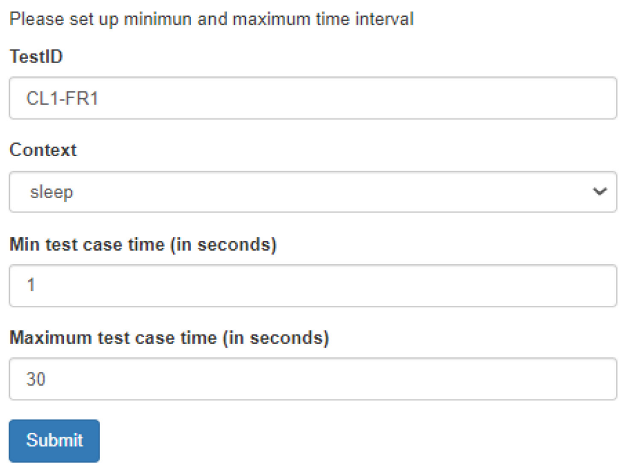
Step 3: Randomly generate the test cases: the developer clicks the button for random test generation. The algorithm decides which states change and at what time they change.



Please complete the form to auto generate test cases

TestID	CL1-FR1
Context	sleep
Sensor	BedRoomLight
Set initial sensor value	1
Start time	15:30:00
End time	16:00
Number of tests	10

Fig. 7. Step 1: Setting up system generated test cases



Please set up minimun and maximum time interval

TestID	CL1-FR1
Context	sleep
Min test case time (in seconds)	1
Maximum test case time (in seconds)	30

Fig. 8. Step 2: Time frequency boundary between test case generation

Once the test cases are generated, they are written to the Reasoner specification file, a copy is written to the text file and the system displays test cases to the developer, figure 9. At this point, the developer is now ready to start testing.

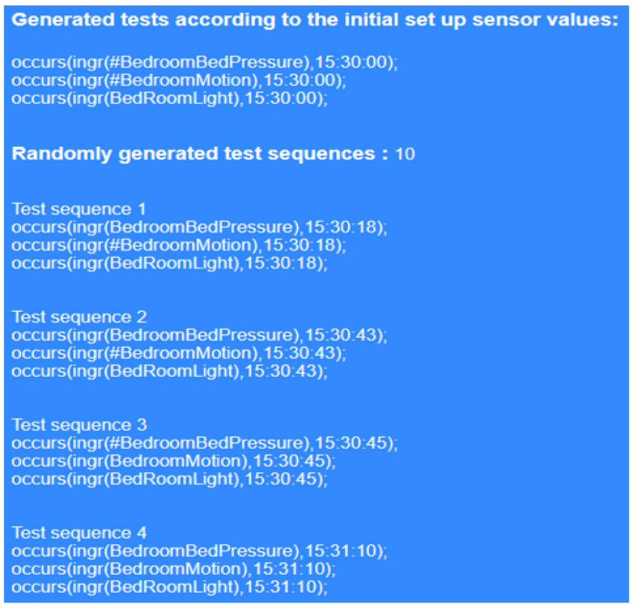
C. *Example of Testing - Simulation*

To give a sample of testing that was carried out:

Test 1: Using the simulation mode test cases were automatically generated and the MReasoner specification was executed. The result of the test showed a mismatch between the actual and predicted outcomes. This entailed the developer tracing the source of the defect via output from the IEQTS.

Test 2: Once the error in the code was fixed by the developer, the test suite was run again. The system still did not give the desired output. However, this time around, the error pointed to a faulty motion sensor. The developer then fixed the faulty motion sensor and re-ran the test suite.

Test 3: Using the same test suite, the system produced desired outcomes. Test results showed that predicted and expected outcomes matched, confirming the correctness of the specification file as intended by the developer.



```
Generated tests according to the initial set up sensor values:

occurs(ingr(#BedroomBedPressure),15:30:00);
occurs(ingr(#BedroomMotion),15:30:00);
occurs(ingr(BedRoomLight),15:30:00);

Randomly generated test sequences : 10

Test sequence 1
occurs(ingr(BedroomBedPressure),15:30:18);
occurs(ingr(#BedroomMotion),15:30:18);
occurs(ingr(BedRoomLight),15:30:18);

Test sequence 2
occurs(ingr(BedroomBedPressure),15:30:43);
occurs(ingr(#BedroomMotion),15:30:43);
occurs(ingr(BedRoomLight),15:30:43);

Test sequence 3
occurs(ingr(#BedroomBedPressure),15:30:45);
occurs(ingr(BedroomMotion),15:30:45);
occurs(ingr(BedRoomLight),15:30:45);

Test sequence 4
occurs(ingr(BedroomBedPressure),15:31:10);
occurs(ingr(BedroomMotion),15:31:10);
occurs(ingr(BedRoomLight),15:31:10);
```

Fig. 9. Display of test cases to developer

D. *Test Results*

Test results are produced from the IE Quality application. A test result that has a mismatch between the predicted and actual outcome notifies the developer of a defect in the specification. When this happens, the developer traces back to the source of the defect. Once the defect is identified, the developer fixes the error and re-runs the test. This is done several times until a desired and expected test result is obtained thereby showing due diligence shown from tests conducted. Test results show a context-aware system that would benefit the healthcare sector in that it is at least accessible, available, usable and acceptable.

E. *Validation Overview*

The scope of our validation considers the identified functional requirements and some quality attributes stipulated in the ISO25000 standard. Our approach does not see validation as independent but connected to testing. We therefore consider the system functionality, system qualities such as response time (speed) and assess user experience. All this is done in order to uncover and fix system lapses.

F. *Example of Validation in the smart environment*

System validation was carried out in a real smart environment accessible on https://ie.cs.mdx.ac.uk/smart-spaces-lab/. During validation the minimum set of requirements were identified and ethical consideration taken into account by not using cameras. Later when the MReasoner specification was executed, the system provided services to the smart home user. For this case study we considered the *sleep scenario*. For this scenario, we considered the three sensors of interest: pressure pad, light actuator and PIR motion sensors. According to the requirement specification for monitoring the sleeping patterns, the user sleeps with the light off and when the user

gets up from the bed in the night, the light is supposed to turn on automatically. This is meant to allow the user to safely navigate their way out of the bedroom and back. Once the user is back in bed and no motion is sensed in the bedroom, the light is expected to turn back off. System validation scenarios 1, 2 and 3 have been shared on the figshare website and can be accessed using the following URLs:
Scenario 1: https://figshare.com/s/61ee984dc5b280615ae4
Scenario 2: https://figshare.com/s/2761acf5edc4542c3d74
Scenario 3: https://figshare.com/s/aeaaf190f3ddff669411

VI. CONCLUSIONS

The uniqueness of context-aware systems entails tailored system development, testing and validation approaches. We have provided a systematic approach to develop context-aware systems, guiding the developer regarding specification of requirements and contexts, and providing a support testing tool that can automatically generate test cases.

Testing context-aware systems is challenging. This is because such systems usually require a combination of sensors to track a context. The sensors with boolean values are easier to manage but it becomes challenging to manage sensors with a range of values that are not boolean (e.g. temperature, luminosity or noise). The latter group presents a challenge in terms of combinatorial explosion. This has resulted in inadequate testing of such systems. Such systems would benefit from our approach because the approach allows the developer to highlight valuable tests of interest and not necessarily focus on exhaustive testing. Consequently, we have brought about structure and efficiency which aids sustainability for developers.

Contexts play a pivotal role in context-aware systems. Our traceability method allows developers to track context development through the system development phases.

In the research undertaken so far, we have achieved a strategy and provided a testing tool that enables *adequate regression testing*. We have provided a testing support tool that assists the developer to explore test cases more relevant to the context under consideration, thereby focusing more on the quality of the system under development rather than test coverage. We offer developer generated test cases based on values of interest. This has been approached from the view that not all values need to be explored and the developer knows and understands which values are reasonable to consider for that specific context being investigated. We also offer random system generated test cases. All generated test cases contain time references to test real life scenarios. Our testing tool has lessened the developer effort during testing, thus bringing in efficiency. The overall research aims at addressing both testing and validation and completing the full circle from requirements to contexts, contexts to tests, tests to validation and validation to requirements.

The next part of our research will focus on the *Validation to Requirements* part of the loop.

REFERENCES

[1] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey", Computer Networks 54 (15), 2010.

[2] J. Augusto, V. Callaghan, D. Cook, A. Kameas and I. Satoh, "Intelligent Environments: a manifesto", *Human-centric Computing and Information Sciences*, vol. 3, no. 1, 2013. Available: 10.1186/2192-1962-3-12.

[3] M. Weiser, "The computer for the 21st Century," in *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 19-25, Jan.-March 2002, doi: 10.1109/MPRV.2002.993141.

[4] E. Aarts and R. Roovers, "IC design challenges for ambient intelligence," *2003 Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 2-7, doi: 10.1109/DATE.2003.1253578.

[5] J. Augusto, M. Huch M., A. Kameas., J. Maitland, P. McCullagh, J. Roberts, A. Sixsmith and R. Wichert, *Handbook of Ambient Assisted Living*. Amsterdam: IOS Press, 2012.

[6] P. Prociow, K. Wac and J. Crowe, "Mobile psychiatry: towards improving the care for bipolar disorder", *International Journal of Mental Health Systems*, vol. 6, no. 1, p. 5, 2012. Available: 10.1186/1752-4458-6-5.

[7] J. Hoyos, "Quality parameters as modeling language abstractions for context-aware applications: an AAL case study | Research Portal", *Researchportal.be*, 2017.

[8] D. Preuveneers and P. Novais, "A survey of software engineering best practices for the development of smart applications in Ambient Intelligence", *Journal of Ambient Intelligence and Smart Environments*, vol. 4, no. 3, pp. 149-162, 2012. Available: 10.3233/ais-2012-0150.

[9] J. Augusto and M. Hornos, "Software simulation and verification to increase the reliability of Intelligent Environments", *Advances in Engineering Software*, vol. 58, pp. 18-34, 2013. Available: 10.1016/j.advengsoft.2012.12.004.

[10] S. van Engelenburg, M. Janssen and B. Klievink, "Designing context-aware systems: A method for understanding and analysing context in practice", *Journal of Logical and Algebraic Methods in Programming*, vol. 103, pp. 79-104, 2019. Available: 10.1016/j.jlamp.2018.11.003.

[11] L Heng, W. Chan and T. Tse, "Testing Context-Aware Middleware-Centric Programs: a Data Flow Approach and an RFID-Based Experimentation", In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering* (*SIGSOFT '06/FSE-14*). ACM, New York, USA, 242–252, 2006. DOI:https://doi.org/10.1145/1181775.1181805.

[12] J. P. Almeida, P. van Eck and M. Iacob, "Requirements Traceability and Transformation Conformance in Model-Driven Development," 2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 2006, pp. 355-366, doi: 10.1109/EDOC.2006.45.

[13] Z. Wang, S. Elbaum and D. S. Rosenblum, "Automated Generation of Context-Aware Tests," 29th International Conference on Software Engineering (ICSE'07), 2007, pp. 406-415, doi: 10.1109/ICSE.2007.18.

[14] A. M. Mirza and M. N. A. Khan, "An Automated Functional Testing Framework for Context-Aware Applications," in IEEE Access, vol. 6, pp. 46568-46583, 2018, doi: 10.1109/ACCESS.2018.2865213.

[15] J. C. Augusto, M. Jose Quinde and C. L. Oguego, "Context-aware Systems Testing and Validation," 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2019, doi: 10.1109/DESSERT.2019.8770048.

[16] B. Meskó, G. Hetenyi and Z Gyorffy, "Will artificial intelligence solve the human resource crisis in healthcare," BMC Health Services Research, 2018, https://doi.org/10.1186/s12913-018-3359-4.

[17] J. G. Meara, A. J. M. Leather and L. Hagander, "Global surgery 2030: evidence and solutions for achieving health, welfare, and economicdevelopment. Vol.386. Lancet. 2015:569–62

[18] J. Augusto, D. Kramer, U. Alegre, A. Covaci and A. Santokhee, "The user-centred intelligent environments development process as a guide to co-create smart technology for people with special needs", *Universal Access in the Information Society*, vol. 17, no. 1, pp. 115-130, 2018. Available: 10.1007/s10209-016-0514-8.

[19] U. A. Ibarra, J. C. Augusto and A. Aztiria Goenaga, "Temporal Reasoning for Intuitive Specification of Context-Awareness," International Conference on Intelligent Environments, 2014, pp. 234-241, doi: 10.1109/IE.2014.44.

[20] A. Lopresti, S. Hood and P. Drummond, "A review of lifestyle factors that contribute to important pathways associated with major depression: Diet, sleep and exercise", *Journal of Affective Disorders*, vol. 148, no. 1, pp. 12-27, 2013. Available: 10.1016/j.jad.2013.01.014.