The 10h IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2019
5-7 June, 2019, Leeds, United Kingdom

Context-aware Systems Testing and Validation

Juan Carlos Augusto[1]
Department Computer Science
Middlesex University
London, UK
J.Augusto@mdx.ac.uk

Mario Jose Quinde[1, 2]
Department Computer Science
Middlesex University
London, UK
MQ093@live.mdx.ac.uk

Chimezie Leonard Oguego[1]
Department Computer Science
Middlesex University
London, UK
CO527@live.mdx.ac.uk

***Abstract*— Newer systems are still tested and validated following techniques which have been developed decades ago when systems were of a different nature. We report on an attempt to define a new method which is practical and focus on the concept of 'context' as a system aspect which have become more relevant in the development of the subsystem category called Intelligent Environments.**

Keywords—testing, validation, context, context-awareness, intelligent environments.

I. INTRODUCTION

Testing has been a core activity for computing systems since the very beginning of computing [1]. Initial focus of testing was on purely software based systems. However automated systems have grown in all directions and as such they keep offering new challenges to developers and to the testing community. Here we are considering an area of recent expansion and which represents a complex mix of other CS subareas. Many new areas of exploration have emerged around the century transition: Pervasive Computing and Communications (Percomm) and Ubiquitous Computing (Ubicomp) [2], then Internet of Things (IoT) [3], Ambient Intelligence (AmI) [4], and Intelligent Environments (IE) [5]. All these have in common the use of new sensing technology distributed in small devices which are used to design more context-sensitive services. Although the most abundant of such systems are 'apps' in mobile phones, there is a wide range of sensorized systems, smart homes, smart offices, smart farming, automated production plants, modern cars, and modern passenger planes are just some examples of the diversity and different scale of such systems.

These systems are made of a complex combination of infrastructure subsystems, typically, there are sensors, a network which links them, data bases, interfaces, human users and, of course, software at various levels performing low or high level functions, all of which hopefully leads to the expected services delivered in good time and form. Because of the diversity of applications, environments, infrastructure, and absence of standards, or at least accepted good practice principles, these systems are made in a somehow ad-hoc manner. Part of our contribution to the developing community is in the form of methodologies based on our experience of creating systems of this nature. One key common concept in all these systems is they more or less explicitly rely on the notion of contexts and context-awareness [6]. When we looked ourselves for community advice on how to test and validate the context core of systems in this area we found very little and not very practical or useful, hence this work is trying to improve that shortage.

Context-awareness has been used as a key concept to develop Ubiquitous Computing and Pervasive Computing. One definition often cited is that one in [7] however it is one that emphasizes too much the system and travels in the direction from the system to the user. We use a person-centric approach [8] which goes from the user to the system, and the user determines which are the relevant contexts, that is "*the information which is directly relevant to characterise a situation of interest to the stakeholders of a system*". Context-awareness is then defined as "*the ability of a system to use contextual information in order to tailor its services so that they are more useful to the stakeholders because they directly relate to their preferences and needs*". We believe this user-centred focus, the relevance of contexts and the heterogeneity of the systems where these concepts are being realized create conditions to revisit testing and to reimagine testing in a way which is of practical benefit to developers in this area, and ultimately lead to more dependable systems and services.

We noticed some difference with normal self-contained software-dominant systems. Whilst in self-contained software systems where software is most of it, models can be better justified. However in IEs the physical part with its collection of sensors, offers a bigger gap between the model and the real system as there are more things unpredictable to fully and faithfully reflect through a model, or collection of models, of the system. There are industries which handle similar technological cocktails and also focus on situations where the system is expected to reliably produce a specific outcome. For example automotive [9] and aviation [10]. However, in automotive and aviation there is more replication as they rely on fewer better studied machines they can rely more on and they can afford that because then they are sold in massive quantities offsetting the design and development costs. In some other IEs the physical realizations of the project are less numerous, and have more unique features, for example, it is unlikely two smart homes systems will be deployed in houses with the same shape and dimensions, same sensing kit and humans with same routines, needs, and preferences. Personalization is also a dimension of product design which is growing and adds to this differentiation of some systems. All our development support strategies are user-centred [8] as we understand these systems are primarily conceived to satisfy human requirements hence they are the main stakeholders and the technology being produce should reflect that subordination to human's needs and preferences (Figure 1).

[1] All authors are members of the Res. Group on Development of Intelligent Environments. http://ie.cs.mdx.ac.uk/

[2] Also member of: Dept. de Ingeniería Industrial y de Sistemas, Universidad de Piura, Perú. Mario.Quinde@udep.pe

978-1-7281-1733-1/19/$31.00 ©2019 IEEE

II. Related Work

There has been some more specific research on testing in relation to contexts. When we searched for methods in the literature this is a summary of what we found. Flores et al. [11] transfer manually a model of components structure and their interactions to automata and then uses TL to study the model properties. Tse et al. [12] Guides testing based on contexts by exploring different types of context variations. Jang et al. [13] Elements are represented in a simulated environment (CSF - Context Simulation Framework). Each one's capabilities (on-off, moving) define a different 'context' which can be assessed in the simulation or by communicating with the real environment. Faria et al. [14] Part of the AAL4ALL project, it provides a framework for products and services specifications (UML based) and how developers can extract test cases from them. Sernani et al. [15] It programs the logic of the AAL environment into an "carer" expert system type of BDI agent system whose input/output is reflected in a 3D simulation environment. The visualization facilitates understanding of correct system behaviour in specific circumstances. Matalonga et al. [16] surveys testing in C-A (software) systems and highlights the presences of a "Context generator" feeding the random "test items" generator and the "test oracle". Rodrigues et al. [17] address Context transitions (example of Android camera running out of battery during the context where is supposed to be used). Naranjo et al. [18] proposes a framework to gather, represent and validate system requirements. Augusto and Hornos [19] outlines a process to map main system elements and their interactions into a model which can be simulated and then formally verified (e.g. in SPIN), simulations and verification counter-examples offer testing opportunities [20, 21].

The only reports we have seen in the literature proposing a more systematic approach and considering the importance of contexts is the Context-aware Test Suit Design (CATS) by Rodriguez et al. [17] which divides the process in three main stages:

1. Context Variables Identification:
   Step 1: Identify context variables from the requirements
   Step 2: Identify additional context variables
   Step 3: Thresholds Identification
2: Create a conceptual model:
   Step 4: Find thresholds in the conceptual model
   Step 5: Create an analytical model
   Step 6: Find thresholds in the analytical model
3. Test Suite Generation:
   Step 7: List test oracles
   Step 8: Create test cases
   Step 9: Package the test suite

They illustrate how it works with a Smart Campus scenario.

However this system is mostly based on the modelling approach and we think in our area of Intelligent Environments the interaction with the physical part of the environments is extremely important both for testing and for validation (two activities which in our view should be more smoothly interlinked).

III. Context-Aware Systems Testing and Validation

Based on the above analysis we are revisiting Testing and Validation from a different perspective with more attention to the diversity of components very much on a System of Systems approach but with a stronger context-awareness and person-centric focus given the nature of the applications we focus on.

In order to differentiate our approach from previous ones we will refer to the approach here discussed as "COntext-Aware systems Testing and valIdation" (COATI). To explain succinctly the theoretical framework we operate on we start by assuming a set of contexts $C=\{C_1 \ldots C_n, ©\}$ which have been gathered from interaction with stakeholders as in Figure 1. Here $|C|$ is finite and practically "manageable" given they are only a number of specific situations we want to secure specific system behaviours for. We assume a default context © which encapsulates all other system situations which are outside the perceived more useful contexts $C_1 \ldots C_n$, that is, © is 'everything else' from a theoretical point of view. Notice each of these contexts can encapsulate a potentially infinite number of consequences, think for example about each tiny variation of a body which can trigger a sensor in a room. We assume also there is a number of requirements $R_{i1} \ldots R_{ir}$ associated with each context C_i. For practical reasons out of the crossing of contexts and requirements a discretization of possible input combinations of interest will arise, so that a body one millimetre to the left or to the right is not relevant as long as successfully triggers a PIR sensor indicating there is a person moving in that room. Also a Context C_i of interest to the final user can be in turn sub-organized in a sequence of formed by Context Features $CF_1 \ldots CF_z$ (each of which in turn for the developer or the system point of view can form a context on its own, forming a hierarchy of contexts at different levels of abstraction). Each context will require a number of resources from the infrastructure to be present for the context to happen, we call these elements Enablers.

Each Intelligent Environment is realized in a system and the resources of that system are used to make the services related to specific contexts to be delivered. In our case we based our system in our Smart Environments Architecture SEArch, [22] outlined in Figure 2. Their elements are used to focus the attention of the developer on how the context will be realized in practice and which system components and resources play a role on making the context work. Other teams with different resources can adapt the table to their infrastructure.

An example of table inspired in SEArch can be seen in the Appendix A. Different tables can emphasize on different aspects of the system architecture. For example Security can require its own ad-hoc table. The table can be used to highlight a minimum configuration required for a context to work. One element failing should lead to the context not giving the expected outcome(s). This can probably be automated in a spreadsheet to be used to generate test cases automatically in the future. For now we are discussing the context. There may be more than one configuration which makes the context work well (e.g., one can detect lights left on unnecessarily in different rooms). Expected failing tests can also be generated by removing some of the perceived necessary enablers or by forcing them to perform in an undesirable way.

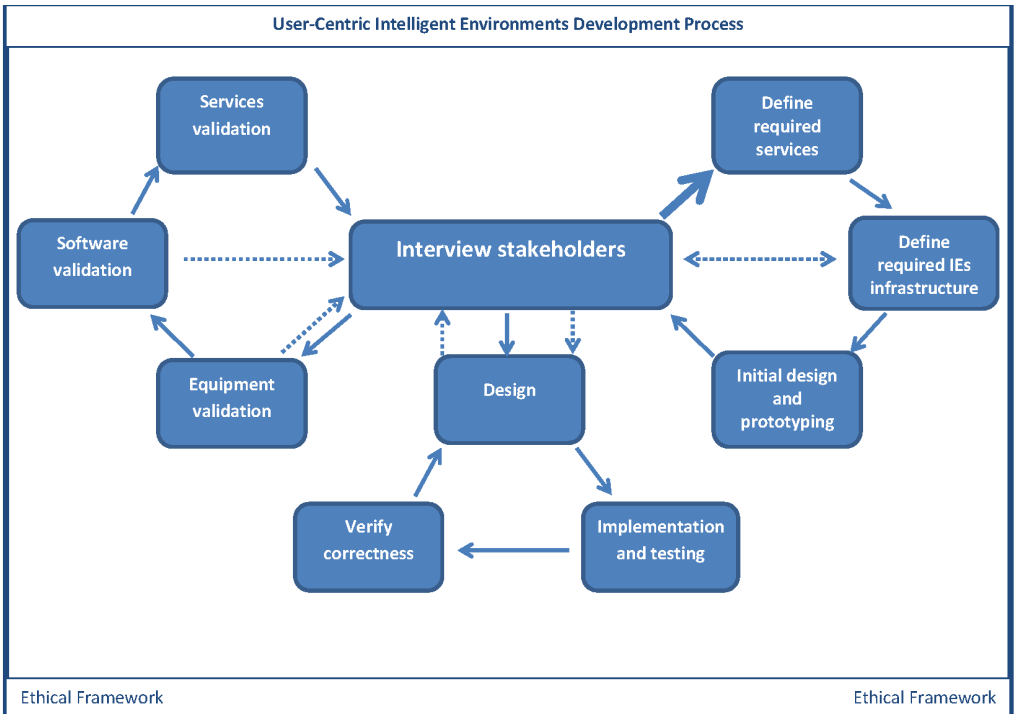

Fig 1. User-centred Intelligent Environments Development Process

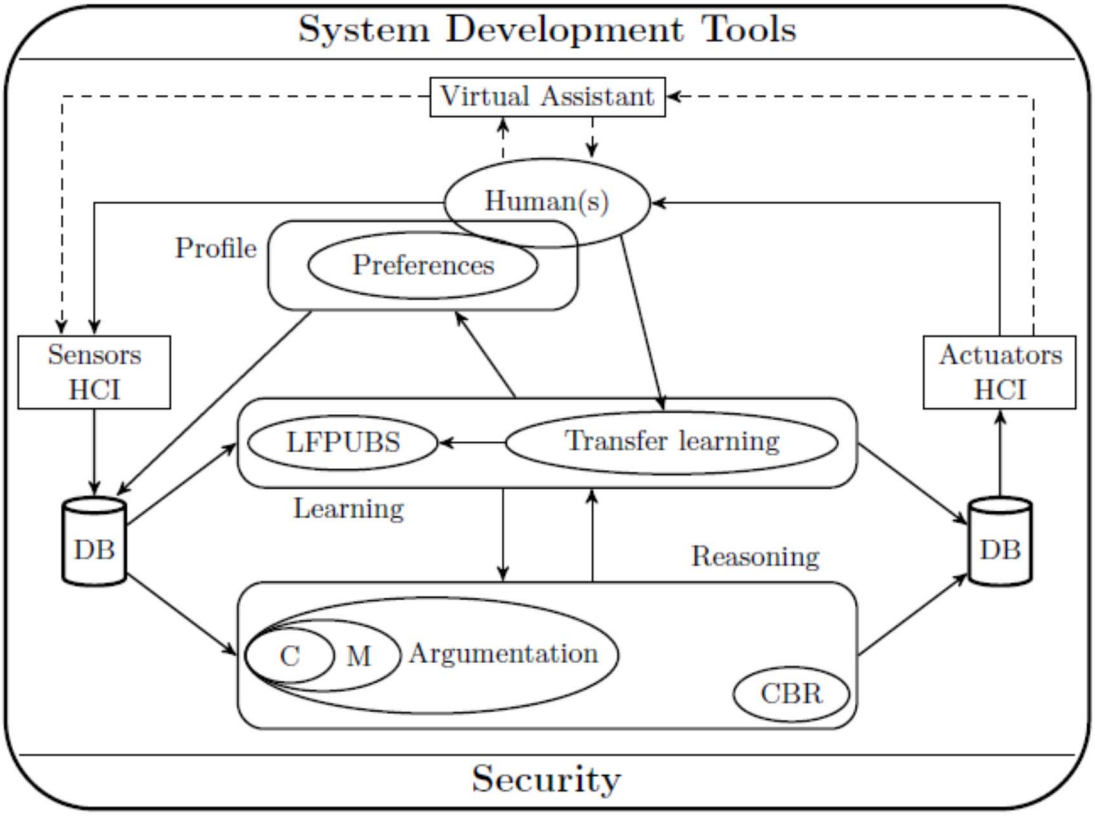

Fig 2. Smart Environments Architecture (SEArch)

IV. Illustration through Case Studies

Following we present two of the various case studies our research team has been working on validated in the Smart Spaces Lab at Hendon Campus[3]. For each of them we provide a description of the system being developed, some contexts under consideration and for one of them we use the method suggested above.

A. *Automated handling of lights*

This section is related to the system being developed and reported in [23]. This is a system to facilitate users to set higher order preferences in a Smart Home system. Some of those preferences have to do with lifestyle, comfort, financial or safety choices. Several planned scenarios were recorded to indicate how the house should react when user performs certain activities or routines and how the system will adjust to the changes of preferences as well as potential conflicts between preferences of different cohabiting users.

An interface was developed to collect and manage user's preferences (Light or Comfort for instance) and to study how the system should react in the smart home especially when there are several options of what the house could or should do for the user. There are several scenarios developed to test with the user to show how the house reacts, including for example:

- The user going to bed in the evening (probably after coming home from work) between 7 or 8pm
- The user waking up in the middle of the night to use the restroom anytime between (12am and 7am)
- The user waking up in the morning to leave home, probably for work.

Table I below includes a couple of such contexts. In this paper we will emphasize on the first scenario where the user is going to bed and should be asleep from 10pm to 7am the following morning. It is also known that the user lives alone, can decide to prefer the lights 'off' when he is asleep at night which provides more comfort compare to having the lights 'on' when he is asleep. So this way user will set his preference of Comfort over Light ("Comfort > Light").

TABLE I. Samples of Contexts for Lights Management

Context label	Context description	Context feature being tested
C1	User preference interface.	Various interface developed with various mobile software applications (ionic, android studio, etc.) were tested with the reasoning system.
C2	Bedroom lights management	Vera Control Box and Tthe reasoning system along with the movement sensor, light actuators and pressure pad.

B. *Assisting users with health conditions*

This section is related to the system being developed and reported in [24]. A person with asthma (PwA) knows that his triggers are low temperatures and high pollen level. He uses a mobile application allowing him to personalize the indicators to monitor. Thus, he configures the mobile application to show alerts when there could be a potential exposure to temperatures below 10C. He also wants to get alerts when the pollen level might be harmful. He chooses to monitor Particulate Matter 10 (PM10) as it is the indicator mostly used to monitor pollen level. He does not know the exact limits to monitor PM10 but the mobile application uses an air quality guideline suggesting PwA to avoid places where the PM10 level is above 50 ug m^{-3}.

TABLE II. Samples of Contexts for Asthma Management

Context label	Context description	Context feature being tested
C1	Potential environmental hazards affecting a person with asthma that knows their triggers.	Alerting users when the environmental indicators they chose to monitor may be hazardous for their asthma health status.
C2	Potential hazards affecting a person with asthma that does not know their triggers.	Alerting users indicators being monitored may be hazardous for their asthma, based on previous hazardous contexts experienced by that person.
C3	Choosing the people to contact by a person with asthma (stakeholders) in case of emergency.	Deploying a notification protocol configured by the user in such way that it involves the stakeholders of a person with asthma.

As in the system described in the previous section there are several contexts. Table II shows a few samples. We choose one of them to illustrate how the testing strategy is applied to it. Examples of how the template of Appendix A can be exercised by contexts in Table I and Table II are available in [25].

Noted benefits from using the table in this context:

- The linking of the higher level context and the lower level infrastructure elements involved in the context realization.
- The column 'Assumption Initial Values' helps to think about how the system should behave in case these conditions are not met. For instance, if the mobile application is installed but the personalization has not been done by the user yet, then it is convenient to push a notification asking the user to personalize it.
- It highlights issues related to the real infrastructure not obvious in a development laboratory environment. Hence, it eases the creation of testing cases that expose the solution to less controlled environments.

V. Validation

The interplay between testing and validation is one that has not been exploited well. Although there is a relation between both in IEs, given the physical presence of systems in the real world (imagine a smart home, a smart office, or an autonomous car), both emphasize different aspects of the interplay between software and hardware and give priorities to different stakeholders (Figure 3). We believe more can be done in Intelligent Environments to interlink both.

[3] http://ie.cs.mdx.ac.uk/smart-spaces-lab/

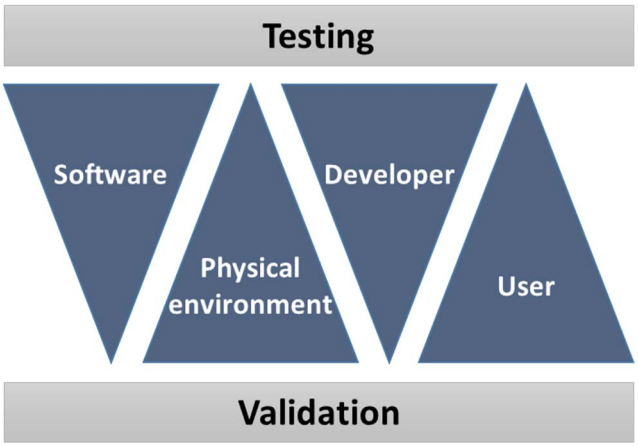

Fig 3. Interplay between testing and validation.

A failing validation experiment can point to the failure of a "Context Feature" CF_i, which focuses attention on a column and depending on the inability of the system observed, it may also focus the attention to those columns where the expected "Enablers" E_j did not perform as expected. This is typically noticed as a mismatch in the real environment between expected (B_i) and observed (O_i) system behaviours:

IF in context C_i *:* $O_i <> B_i$ *THEN*
trace back implementation of failing CF_i
to ill-performing E_j

VI. Conclusions

Our approach is inspired by our user-centric approach to develop Intelligent Environments, and focus on the notion of context which so far has not been given due attention within the developing community with regards to testing and validation.

Our proposed testing method is consciously simple so that any developer can follow it and obtain benefits with smaller time investment. The strategy is to focus developers' minds first on the contexts that matter and then on the infrastructure elements which act as enablers of those contexts. At this stage the contribution is not on a quantitatively more efficient method but one which emphasizes the role of "contexts" as a priority concept. Finally we do a preliminary discussion on the inter-relation between testing and validation.

We do not consider this a closed discussion, rather the beginning of it. Testing and validation are important aspects in the development of Intelligent Environments and there is a remarkable absence of methods and tools to assist developers. This needs improvement if we want IEs to become more dependable, necessary for their adoption.

References

[1] G. Myers, The Art of Software Testing, 3rd ed., Wiley, 2011.

[2] M. Weiser, "The computer for the 21st century," Scientific American, vol. 265, 1991, pp. 94-104.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," Comput. Netw., vol. 54, issue 15, 2010, pp. 2787-2805.

[4] E. Aarts and R. Roovers, "IC Design Challenges for Ambient Intelligence," The conference on Design, Automation and Test in Europe (DATE '03), vol. 1, 2003, IEEE Computer Society, pp. 10002-.

[5] J.C. Augusto, V. Callaghan, D. Cook, A. Kameas, I. Satoh, "Intelligent Environments: a manifesto," Human-centric Computing and Information Sciences, vol. 3, number 1, 2013.

[6] J. C. Augusto, A. Aztiria, D. Kramer, and U. Alegre, "A survey on the evolution of the notion of context-awareness," Applied Artificial Intelligence, vol. 31, number 7-8, 2017, pp. 613-642.

[7] A. K. Dey, "Understanding and using context," Personal and Ubiquitous Computing, vol. 5, 2001, pp. 4-7.

[8] J. C. Augusto, "User-Centric Software Development Process," 2014 International Conference on Intelligent Environments, Shanghai, 2014, pp. 252-255.

[9] H. Altinger, F. Wotawa, and M. Schurius, "Testing methods used in the automotive industry: results from a survey," The 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing, San Jose, 2014, pp. 1-6.

[10] Y. Li and W. Fan, "Analysis of software testing system in civil aviation field," Physics Procedia, vol. 33, 2012, pp. 470-475.

[11] A. Flores, J. C. Augusto, M. Polo and M. Varea, "Towards context-aware testing for semantic interoperability on PvC environments," IEEE International Conference on Systems, Man and Cybernetics, The Hague, 2004, pp. 1136-1141 vol.2.

[12] T. H. Tse and S. S. Yau, "Testing context-sensitive middleware-based software applications," Proc. of COMPSAC 2004., Hong Kong, 2004, pp. 458-466 vol.1.

[13] M. Jang, J. Kim, and J. Sohn, "Simulation framework for testing context-aware ubiquitous applications," Proc. ICACT 2005, Phoenix Park, pp. 1337-1340.

[14] J. P. Faria, B. Lima, T. B. Sousa and A. Martins, "A testing and certification methodology for an Ambient-Assisted Living ecosystem," The 15th International Conference on e-Health Networking, Applications and Services, Lisbon, 2013, pp. 585-589.

[15] P. Sernani, D. Calvaresi, P. Calvaresi, M. Pierdicca, E. Morbidelli, and A. Dragoni, "Testing Intelligent Solutions for the Ambient Assisted Living in a Simulator," The 9th ACM Conf. on Pervasive Technologies Related to Assistive Environments, Corfu, 2016.

[16] S. Matalonga, F. Rodrigues, and G. H. Travassos, "Characterizing testing methods for context-aware software systems: Results from a quasi-systematic literature review," Journal of Systems and Software, vol. 131, 2017, pp. 1-21.

[17] F. Rodrigues, S. Matalonga, and G. H. Travassos, "CATS Design: A Context-Aware Test Suite Design Process," The 1st Brazilian Symposium on Systematic and Automated Software Testing (SAST), Maringa, 2016, Article 9, 10 pages.

[18] J.C. Naranjo, C. Fernandez, P. Sala, M. Hellenschmidt, and F. Mercalli, "A modelling framework for ambient assisted living validation," in Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments. LNCS 5615, C, Stephanidis, Eds. Springer, Berlin, Heidelberg, 2009.

[19] J. C. Augusto and M. J. Hornos, "Software simulation and verification to increase the reliability of Intelligent Environments," Advances in Engineering Software, vol. 58, 2013, pp. 18-34.

[20] A. Gravell, Y. Howard, J. C. Augusto, C. Ferreira, and S. Gruner, "Concurrent development of model and implementation," The 16th International Conference on Software and Systems Engineering and their Applications, Paris, 2003.

[21] Y. M. Howard, S. Gruner, A. M. Gravell, C. Ferreira, and J. C. Augusto, "Model-based trace-checking," UK Software Testing Workshop, York, United Kingdom, 2003.

[22] J. Augusto, M. Quinde, J. Gimenez Manuel, M. Ali, C. Oguego, C. James-Reynolds. The SEArch Smart Environments Architecture. Proc. of IE'19. Morocco, June 2019.

[23] C.L. Oguego, J.C. Augusto, A. Muñoz, M. Springett, "Using argumentation to manage users' preferences," Future Generation Computer Systems, vol. 81, 2018, pp. 235-243.

[24] M. Quinde, N. Khan, J. C. Augusto, "Personalisation of context-aware solutions supporting asthma management," in Computers Helping People with Special Needs. LNCS 10897, K. Miesenberger, G. Kouroupetroglou, Eds., Springer, 2018, pp. 510-519.

[25] J.C. Augusto, M. Quinde, C. Oguego, "COntext-Aware systems Testing and validation" (COATI) Appendices. figshare. 2019. https://doi.org/10.22023/mdx.7970564

Appendix A

	Enablers	Assumptions Initial values	Context Feature 1	Context Feature 2	…	Context Feature z
Context Description						
Expected Outcome (s)						
Sensors	S_1					
	…					
	S_s					
Network	N_1					
	…					
	N_n					
Database	D_1					
	…					
	D_d					
Reasoners	R_1					
	…					
	R_r					
Learners	L_1					
	…					
	L_l					
HCI	H_1					
	…					
	H_h					
Preferences	P_1					
	…					
	P_p					
Users	U_1					
	…					
	U_u					