To appear in *Applied Artificial Intelligence*
Vol. 00, No. 00, Month 20XX, 1–27

Context-aware Systems Architecture (CaSA)

Juan C. Augusto[a*] and Mario J. Quinde[b] and Chimezie L. Oguego[a] and José Giménez Manuel[a]

[a] *Research Group on Development of Intelligent Environments, Department of Computer Science, Middlesex University, London, UK;* [b] *Faculty of Engineering, University of Piura, Perú*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Context-aware systems are becoming increasingly mainstream as more and more technology allows real-time collection of daily life data and it is more and more affordable to provide useful services to citizens in various situations of need. However, developers in this field are not well supported. Naturally we have inherited a number of methods and tools from past software engineering efforts to create previous computing systems. However the most recent generation of systems dominated by sensing supported context-awareness integrating a variety of data sources and with a higher expectation of personalized services delivered at the right time, place and in the right form, are not well supported. Developers need more guidance and support to pinpoint those valuable contexts and to work out ways of detecting them and activating the right services associated with these contexts. Our community has reported on various systems they created however not much is emerging in a way of a methodology, a standard, a transferable body of advice and guidance which can help teams next time they need to develop a new system. In this article we explain a couple of complementary methodologies which we have tried and tested through development of different context-aware projects. We argue these are of practical usefulness and provide an initial valid point of discussion for our community to create evolved versions of these which can be tested more widely to identify good practice in the area.

Keywords: intelligent environments, system architecture, contexts, context-awareness, context testing, context validation.

1. Introduction

Our society has recently witnessed a significant increase in technology supported tools in a diversity of areas with direct impact in our daily lives. Various areas of business, industry, technology and science have evolved enough to facilitate these developments. Amongst those areas we can highlight a more IT aware population, optimization of technology production, the exploration of sensing, and maturity of various computing related subfields. Many new areas appeared within Computer Science around and after the turn of the century: Pervasive Computing and Communications (Percomm) and Ubiquitous Computing (Ubicomp) Weiser (1991), Internet of Things (IoT) Atzori et al. (2010), Ambient Intelligence (AmI) Aarts and Roovers (2003), and Intelligent Environments (IE) Augusto et al. (2013). 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 larger scale 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

*Corresponding author. Email: j.augusto@mdx.ac.uk at various levels performing low or high level functions. All of these hopefully reasonably combined so that the expected services are 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. A key common concept in all these systems is they more or less explicitly rely on the notion of contexts and context-awareness Augusto et al. (2017). This reliance on the notion of context and on system context-aware driven reactions have constantly growing in relevance since the turn of this century. Systems are progressively becoming more personal and more personalized, expectations have changed, engineering techniques need to adapt to these changes so that development teams have better support to engineer these new generations of systems. Most of the 'tried and tested' engineering techniques can be reused to some extent. However we can build chips with 'hammer and chisel', we need to make the methods and tools more efficient for these new generation of products. But how? When we searched for help from our reference communities to adopt and reuse even something as basic as a context-aware architecture we found a shortage of guidance and help. Not only there is no widely accepted standard context-aware architecture, it is even hard to find one which can be easily adapted to the various similar yet different systems based on the notions of contexts and context-awareness. What we did, and report in this study, is to compile and investigate a number of previous suggestions from our scientific community in relation to context processing within our technical area and to select what we considered various useful features complemented with our long experience of building systems in this area. The result is an architecture which we feel should be relatable by teams working in the area and provides both a guidance for those starting as a safe departing point as well as for more experienced teams to be able to adapt it to different systems as it encapsulates a number of building blocks which are application independent.

In the next section we provide a survey covering various relevant systems and we outline our current Context-aware Systems Architecture (CaSA). We then relate how that conceptual tool is integrated to more general and other more specific methods we use to support our work on Context-aware Systems. One underlying principle is our emphasis on human-centric systems. The other important principle is that contexts are essential building blocks which differentiate systems in this area with those in other fields of Computer Science. So we show in Section 3 how the new proposed architecture links effectively with well established stages of traditional development processes such as requirements, testing and validation. We also want our methods and tools to be effective, and that includes to be easily understood and applied. Hence we include in Section four detailed descriptions of how the Architecture influenced Requirements, testing and validation helped us develop some of our deployed projects.

2. Context-aware Systems

Context-awareness is an essential aspect of Intelligent Environments. These environments are sensitive to specific circumstances occuring in in certain ways, in which case they are supposed to deliver certain services in certain manner. Contexts are the key to effective systems, their success is directly proportional to their 'context-awareness':

> Context: "the information which is directly relevant to characterise a situation of interest to the stakeholders of a system".
> Context-awareness: "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".

Consistently with our user-centred approach we differentiate our approach from other more system-oriented and data-oriented approaches Abowd et al. (1999) by emphasizing the humans which are supposed to benefit with the system.

Given the key role of contexts, one natural follow up question is: How to organize context processing within a system? Scientists have tried a number of approaches. Most of them ad-hoc to address the implementation of specific applications using the notion of context as their development strategy. However, the diversity of these proposals and the lack of generality in their approach meant we often found them unsatisfactory as a guide to develop other systems which depart from that specific example they provided. With the spirit of obtaining a more useful guide for ourselves and also to share a more useful recipe with our community we created a new approach as a symbiosis of several of these previous context-processing architectures. We do not claim perfection, however, we believe it provides a reasonable start to support more discussions on how context-aware architectures can work more effectively.

We reviewed all publications we could found by searching in journals and conference proceedings with the keywords, "context" AND "architecture" as well as "context-awareness" AND "architecture". This included several entire proceedings of several conferences in the area (CONTEXT, IE, AmI, PerComm, Ubicomp, ICCASA, etc.). We have been working in this area for more than a decade and we are familiar with the main publication venues. Context and context-awareness have attracted significant attention however we found very little in the way of system architectures for context-aware systems. Most of what we found, as we explain further up, were reports of system implementations and little in the way of architectures, the architectures reported were ad-hoc and hard to generalize and/or reuse. Below we summarize what we found. Coverage is not exhaustive, however this is a very representative selection of the 18 publications which we found more relevant, most of them were published in conferences.

Time wise we noticed there were more discussions about this a decade ago, however there was no standard or reference emerging from that process, and then fewer proposals more recently. Part of that more intensive discussion about context-awareness architectures was purely focused specifically on middleware for context-awareness Li et al. (2015), however here we are addressing the wider picture of context acquisition, detection and use.

Ayed et al. Ayed et al. (2007) provide a higher level view of the processing of contexts within a system. From here we liked the identification of characteristics which identified contexts of interest. Park et al. Park et al. (2007) presents a security focus, however, one generic feature we found useful is "context aggregation". Chaker et al. Chaker et al. (2011) highlighted the value of structuring rules into higher level concepts in the system representation they called 'situations'. Other approaches considered specific well-known conceptual tools pre-existing in Computer Science which they then extended with concept elements. For example, Zacarias et al. Zacarias et al. (2007) offered an architecture which is 'software-agent'-centric and placed context related elements within that. There has been also a modelling of the contextual processes through Petri Nets, see Neumann et al. (2011). A number of application dependent architectures have been also published whilst describing specific system implementations Allegre et al. (2013); Furno and Zimeo (2013); Papadopoulou et al. (2013); Peko et al. (2014); Souabni et al. (2014). There has been also some interesting work on so-called 'contextual graphs', Tahir and Brézillon (2013), and on hierarchy of ontologies, Kolbe et al. (2017), to structure context knowledge and its processing. These later work being more focused on knowledge organization rather than our focus on how the overall system infrastructure can be used to support and guide the consideration of contexts. Different specific architectures were proposed for specific applications and at a much lower level of task management, Hassani et al. (2017) and Hassani et al. (2017), more akin to middleware architectures.

A recently relevant publication by Perera et al. (2014) mentioned context categories, although the categorization is only binary and the context dimensions are four fixed ones and we believe this is an oversimplification. A circular process of context acquisition, modelling, reasoning and dissemination/distribution offers an overall context lifecycle, although, again may be an idealized way which does not take into account for example existing, mutated and newly acquired contexts, requiring different associated processes. These are then combined in a sort of declarative table of how contexts and associated processes are classified. A more procedural account could be a good complement. This was more the intention in the architecture presented at Kramer et al. (2014) which was generic and used to develop services in a specific area of Ambient Intelligence. This was then explained at a lower level in Kramer and Augusto (2017).

Out of the analysis of all those interesting developments we noticed not standard or architecture general enough to be easily adapted and reused came out so far, hence we set up to develop an architecture which generalize what we considered to be their best features and also complements previous efforts by highlighting certain dimensions we consider important in the context-awareness systems development. Taking into account all those previous useful exercises and our own continuous experience we arrived at an architecture which is consistent with our user-centred approach and is adaptable to systems of varying complexity.

One important departing point for our proposal is that 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. Basically, in IE we are mostly limited to the contexts we can detect. In our case we based our approach in our Smart Environments Architecture SEArch, see Augusto et al. (2020) for a complete description, outlined in Figure 5 which makes explicit the algorithms and other resources our system can use. In summary, the overall SEArch architecture highlights the input of the system on the left and middle top and the output on the righth hand side. At its core there are various resources which cooperate and can be used depending on the problem at hand. There is a Machine Learning cluster in the centre (LFPUBS and transfer learning). An Automated Reasoning cluster in the lower centre (includes real time monotonic languages such as C and M as well as an inconsistency handling Argumentation System). These resource clusters is also complemented by hybrid approaches such as Case-based Reasoning (CBR). The system is personalized and aligned with the users' priorities through user profiles, which is represented in the upper left side of the diagram. Most importantly for this article, acting as a first important nexus between the input (especially sensor data streams) and the learning and reasoning clusters is the context-awareness module which we describe below in more detail and allows a first identification understanding of the world and a decision on which learning and reasoning resources are required to act appropriately.

The elements of SEArch help focus the attention of our developers on how the context will be realized in practice: (a) which system components and resources are available to be used, (b) which of those are more relevant on making the envisioned context work. Of course other teams with different resources will have a different SEArch archtecture diagram and a different list of sensors and they can adapt this figure to their infrastructure as well as the other steps we explain further down. However, at least now we are converging into a process which teams can adopt as a general recipe and easily adapt to their specific infrastructure. Our Context-aware System Architecture (CaSA) is shown in Figure 6. The higher levels are focused on "context identification": at the top the relevant contexts are determined based on user feedback. Then developers need to create a strategy for context detection in real-time. Based on that the "collectors" are decided, i.e., which part of the infrastructure, will be required. In our area Sensing equipment will make a substantial part of the collectors. Initially a decision will be made about the collector type, then about how many, where to place them, etc. Once the collectors retrieve information they are used to detect contexts in real-time. This process sometimes requires some "intra-context" inferences, and sometimes contexts can also be aggregated to form higher level contexts and do "inter-contexts" reasoning. For example, intra-context activities may require to recognize some of the possible combinations of smaller micro activities which allow to have certainty that a person is geting up from bed, not only getting the body physically detached from the bed but the act of getting up in the morning to go to work. This activity put together with other typical actions such as going to the bathroom and going to the kitchen can form a higher level context of breakfast-time, this one with other signs of leaving the building, at a certain time of a certain day of the week, can match another associated context of "leaving for work'. Finally once meaningful contexts have been detected they are related to the rest of the knowledge management part of the system (more general reasoning and learning) to produce changes in the environment which are fed back to the users. For example, matching the breakfast routine could trigger the automated turning of the kettle on and bringing a report on city air quality for a user affected by asthma. The CaSA architecture description below is minimalistic as it is designed to capture the essential aspects of context handling at a level which is generic enough for other projects to adopt and refine ad 'libitum' according to their needs. It is also amenable of extensions with concepts deemed to be generic enough, now we have a tool that other projects can relate to, previous architectures would not have resisted specialization without significant alteration to the point of no recognition. Contexts are an important part of our system development as we know they really decide how responsive is the system to what the main beneficiaries care about and expect from the system, the behaviours which make users have the feeling the system is intelligently looking after their interests and improving their life experience. So we use contexts in a practical sense to guide our system testing and validation through our COATI methodology (Augusto et al., 2019).

To explain succinctly the theoretical framework we operate on we start by assuming a set of contexts $C = \{C_1 ... C_n, ©\}$ which have been gathered from interaction with stakeholders. Here $|C|$ is finite and practically "manageable" given there 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 $\{C_1 ... C_n\}$, that is, ©is 'everything else' from a theoretical stance. Notice each of these contexts are actually clusters of system situations. For example if we consider light automation in a room by detecting movement of a person, then all movements which come into the reaching zone of the sensor and are higher of the threshold of sensitivity of that sensor will be referred to by that context. The person can move in various different ways, at different speeds, in different places, for different reasons. Different contexts can be more specific, however usually contexts C_i are labels for a combination of conditions which can be satisfied in more than one way. If we consider examples such as the one mentioned above for light automation, and if we consider all combination of body positions within a granularity of say 1 cm in any ordinary room we are already considering an amazing number of small system variations. Also a Context C_i of interest to the final user can be in turn sub-organized in a sequence of, or formed by, *Context Features* $CF_1, ... CF_z$. Each of these in turn can form a context on its own for the developer or the system point of view, 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*.

3. Engineering Context-Aware Systems

Our vision of context-aware systems engineering from an ICT/CS perspective is user-centred, in the wider sense, as in system stakeholders. In the specific technical areas we are considering, contexts are considered the 'building bricks' of systems, and our emphasis and interest are in that these systems are developed by humans for humans. Even if the system is deployed in the middle of the wilderness, it is in the vast majority of cases to ultimately benefit humankind. Taking that as a fundamental point, we aligned our work to explicitly recognize this and to use it as an important guide and inspiration. After many years of work in this area a picture started to emerge in the process which not only took into consideration this user-centred aspect, also other distinctive features we put in our area such as the emphasis on the sensing enabling technology, its acceptability issues, and the real world physical environment validations. This led to the User-centred Intelligent Environments Development Process, Figure 7. It is a flexible iterative strategy which is centred on stakeholders involvement (starting with the stronger arrow in the upper right corner) and have smaller specialized loops which can be used with different level of intensity at different stages of the project. These smaller loops centre more on design (right hand side of figure), development (lower centre), and validation (left hand side). For more detailed explanation and extensive explanation of its use in projects see Augusto (2014) and Augusto et al. (2018).

There is a useful complementarity in between the three system views we have presented so far. Whilst Figure 5 shows the hybrid software-hardware infrastructure, and Figure 6 'zooms in' the context module of that infrastructure, Figure 7 shows a higher level strategy of how we use the resources available in the infrastructure through a series of iterations until a product is obtained. Now we illustrate with a number of projects which have been recently developed in our lab how some aspects of the context-aware features are tracked through the system. These examples, shows at the same time what is common process for all projects and at the same time how the methodology is both specific enough to provide practical usefulness and generic enough to accommodate a diversity of applications.

3.1. *Requirements*

We assume there is a number of requirements $R_{i1}...R_{ir}$ associated with each context C_i. As we explained before, each context defines actually a number of practical combinations. We leave to the developing team to decide the level of granularity of those contexts and associated sub-contexts. According to established traditions these will be described through a number of requirements. Again, for this process we have our own recipes which we use to secure alignment between requirements and user expectations Evans et al. (2014).

3.2. *Testing*

The knowledge of the real contexts leads to the understanding of the basic components of the system and the range of conditions they have to satisfy to enable detection of a context C_i. We choose to highlight these conditions in a table, given it summarizes in a concise way the key factors in the process of context identification. This process is dependent on the infrastructure being used, in our case we base our development in the SEArch system architecture hence our testing guiding tables are influenced by it as well as the hardware used in the application. There is a generic table offered as guidance to teams (Augusto et al., 2019) and we offer three examples of its use in the next section.

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 be used to both explore combinations which exemplifies different ways to satisfy or to fail to satisfy a context detection, or the explore optimizations of a context detection by removing components or changing conditions associated with them. Noted benefits from using the template table are:

- 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.

3.3. *Validation*

The interplay between testing and validation is one that has not been exploited well yet in our area. 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. For example, a failing validation experiment can point to the failure of a "Context Feature" CF_i, which focuses attention on a column of the table for that context. Depending on the inability of the system observed, it may also focus the attention to those columns where the associated "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 behaviors:

```
If in context $C_i$: $O_i <> B_i$
        Then trace back implementation of failing $CF_i$ to ill-performing $E_j$
EndIf
```

So far we have presented the need for a generic context-awareness system architecture and we created one inspired in the best elements of previous project specific attempts plus other concepts based on our experience and on human centric priorities we suscribe too. Also we have explained associated important concepts (UC-IEDP and SEArch) which help the reader to understand the architecture we created and used was not an isolated tool. In addition to having a deeper understanding on how these type of systems work, we also mentioned we were interested in practical benefits of having clarified these concepts and having built strategies based on them. One of these benefits we had during development is that it helped guiding the connection between requirements, testing and validation. We illustrate this in the next section with three projects which were developed concurrently and allowed us to test and correct initial approaches of the concepts described above.

4. Illustration Through Case Studies

The previous section explained the conceptual tools and strategies we follow to put special attention on how we address contexts and context-awareness. Whilst SEArch tells us what infrastructure to use, CaSA tell us how these resources cooperate and COATI tables highlight the critical conditions which are required for context detection. All this as part of a bigger project development life cycle which is user-centred and the segment of that process we are focusing on here is the connection between design and testing (also with an impact on guiding validation). To illustrate how this works at a practical level we present some of the projects our research team has developed and validated with support of the Smart Spaces Lab at Hendon Campus[1]. For each of them we provide a description of the system being developed, some of the contexts which were considered and how the concepts and methods described above where applied. Figure 8 below shows the technology installed in the Smart Home environment which was used as support for the development of the scenarios described further down.

4.1. *ADL coaching*

Ambient Assisted Living offers many solutions addressed to People living with Dementia (PwD). Many approaches are based on enhancing their autonomy by detecting and supporting the Activities Daily Living (ADL) such as (Lazarou, et al., 2016) or (Stucki, et al., 2014) show. These works use different sort of sensors for activity recognition (AR). These solutions within dementia provide a

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

track of user's activities, which is valuable to understand and diagnose the impairment based on abnormal users patterns. That also helps health professionals to adjust the user's treatments or interventions for the user. Although these solutions claim enhancing the users' autonomy, they still omit PwD as the main beneficiary and show a strong dependence on secondary users.

The AnAbEL system approach focuses on primary users. It is integrated within the SEArch architecture (Augusto et al, 2019). The system detects user's actions in the house through non-intrusive sensors and infers the activities using MReasoner. Once it detects an activity, AnAbEL assesses it as usual/unusual or healthy/unhealthy regarding the context defined by user parameters in the interface. If the activity is unusual the system warns users about their behaviour trying to entice the behaviour. In case the user goes on with the behaviour since the system still is checking the activities the system alert caregivers allowing them to take a proper action.

Among the ADLs, AnAbEL focuses on sleeping, eating, wandering and elopement (leaving the house by no reason). The elopement is a common problem in PwD. The person feels confusion or anxiety and tries to go out of the house. The next testing case focuses on this behaviour.

Requirements that led to the context description table: Determining activities or behaviours using non-intrusive sensors in a complicate task to test, since these techniques do not use cameras that allow checking what is happening certainly. The activities recognition using non-intrusive devices implies an IoT deployment around the house such as motion, reed, pressure or energy sensors.

The functional requirements met for all activities and behaviours related to the context are:

- Sending a warning to the user's mobile when the activity is assessed as unusual after the threshold time set in the interface to warn the user has elapsed.
- Sending an alert to caregivers to inform about the situation after the threshold time set in the interface to alert the caregiver has elapsed.
- 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.

Since this context explanation includes abbreviation of some parameters from user's interface, next they are explained:

- T_a,T_b: Represents the time of an schedule. (T_a,T_b) describes the period between an initial time T_a and final T_b request from the interface. For example, $T_a = 12:00$ and $T_b = 19:00$ represent the period between 12 PM and 7 PM when usually the user leaves the house. Out of this time, the activity could be considered unusual and therefore risky.
- T_{au}: This time represents the elapse of time the system waits once detect the behaviour until it takes action and alerts the user.
- T_{ac}: Time the system wait to alert a caregiver once the behaviour is detected. If the user does not amend the behaviour.

The main door reed sensor and corridor motion sensor close to the main door trigger the "goOut" state. We also define the action "goIn" to determine when the user is coming back into the house. Since both actions use the same sensors, the order when one is detected regards the other is important. We define "goOut": "first the movement is detected inside the house and later the door is opened". On the contrary, we define "goIn" as: "first the door is opened and later movement is detected inside the house". Initially, we estimated 2 seconds was enough between actions. Now, the system detects the activity and assesses with the timetables (T_a,T_b). As the user is leaving the house at improper time, the user's mobile receives the warning about that after the time set in the interface has elapsed (T_{au}).

On the contrary, we define "goIn" as: "first the door is opened and later movement is detected inside the house". Initially, we estimated 2 seconds was enough between actions. Now, the system

Table 1.: Context Samples for ADLs Scenario

Context Label	**Requirements**	**Context description**	**Context feature being tested**
EloCF1	System notifies the user when he leaves the house at unusual time.	The user leaves the house during a time out of the timetables configured in the interface. Possible elopement detected.	Whether the action occurs during the time considered unusual the users mobile app receive an alert asking about the situation along the list of response to select. The system saves the action “goOut” in the server.
EloCF2	System notifies the caregivers whether the user does not come back.	The user does not get back home after receive the alert and after the time set in interface to alert caregiver passes.	According to the time to alert the caregivers after elopement is detected whether the user is not coming back to home, the caregiver receive an alert in the APP.
EloCF3	The user amends the behaviour and comes back into the house. No action occurs.	The user goes back home after receive the alert and before the selected time to alert caregiver passes.	The system detects the user is going in the house. It does not sent any messages. The system save the action “goin” in the server.

detects the activity and assesses with the timetables. As the user is leaving the house at improper time, the user's mobile receive the warning about that after the time T_{au} set in the interface has elapsed.

Testing elopement context feature 1 (EloCF1): “If the user leaves the house at a time (T_d) during a period [T_a-T_b] configured in the user's interface, then notifying the user at T_{au} (seconds to alert the user once this behaviour is detected) after the ‘goOut’ action is detected”. In case the user stays out, the system does not detect “goIn” action and then sends a message to user's mobile application asking about the situation.

Testing elopement context feature 2 (EloCF2): “Whether user has leaved the house at a time (T_d) during a period [T_a-T_b] and is not coming back then notify caregiver T_{ac} seconds after the ‘goOut’ action is detected”. On the other hand, whether the user is persuaded by the message and goes back, the caregiver is not alerted. However the action “goIn” is save in the server.

Testing elopement context feature 3 (EloCF3): “The user has gone back into the house ‘goIn’ before T_{ac} seconds elapse after the ‘goOut’ action is detected”. These contexts are listed in Table 1 associated with the requirements and context features to be tested in each context.

Test 1: Trying the initial configuration for elopement EloCF1, as Table 4 shows in column Test1, gives an undesirable outcome compared to the expected one. Checking the enablers that also are not having the expected outcomes, we conclude that the problem is the action “goOut” that is not detected since the value of sensor was not triggering as we expected. That fact helps to discover that the movement is detected earlier than we though. Whether motion sensor reset to 0 (by default 5 seconds later) when the user opens the doors and the motion sensor has changed the state at least one second ago, there is not action detected. From here, we concluded that the definition of rule “goOut” was not precise, so we modified “goOut” rule by defining as “if during at least the last 3 seconds the motion sensor has been activated" as the code previously explained to detect elopement behaviour. Also, the time operators used in the rule were changed to adapt the rule to the new suggestion, although here this is not explained, the above explanation summarise well the process.

Test 2: showing the expected outcome in several attempts as the table shows, so that the new configuration was adopted.

4.2. *Incorporating User Preferences*

Ambient Assisted Living (AAL) according to Ref. McNaull et al. (2014) provides subjects with required assistance, which includes monitoring their activities using AAL systems that will allow detection of undesired circumstances they might suffer. Such system can be created with current technology based on wireless sensors Martínez Madrid et al. (2013). For instance, movement within a room can be perceived by the system using Passive Infrared Sensors (PIR).

Some PIR sensor system, for example, can respond to human movement Martínez Madrid et al. (2013), as they are typically used at offices which will turn lights 'off' when there is no movement after some time. However, this is unhelpful when one stays still absorbed in reading and suddenly the lights go off, breaking the concentration and forcing the wave of arm(s) to turn lights back on again. Conversely, as soon as movement is detected the system brings the lights back on. This is fine for an office though not for a bedroom, as moving during the night will cause the lights to go 'on' and 'off' intermittently several times.

There are two problems with the above type of systems which the present work has aimed to address Oguego et al. (2018). One is that those office systems are set in such a way that (whilst not impossible to change), modifying the waiting time is usually beyond most typical users' capabilities. The other problem is that the system's notion of context is very limited. The only context they recognize is time without movement.

The research into these systems aims at providing ways for users to easily personalize the behaviour of the system through parameters which represent their preferences. The parameters which can facilitate this personalization depend on the technology available in a given environment. The aim was to keep the system functionality, the technology and the type of personalization simple, and demonstrate that the system is intelligent and capable enough to detect whether the person is sleeping or not and whether lights should be turned 'on' or 'off' in a sensible and flexible way. Providing such system involved series of testing along the way, and some of the test carried out are discussed here.

Requirements that led to the context description table: some requirements were crucial in testing the personalised system, so as to ensure it has the ability to interact among user, system and the real world.

- The User's preference interface: This the user uses to prioritise their preference ranking. The preference ranking order goes from 1-10, with 1 being the lowest and 10 being the highest.
- Informal Scenarios: Various informal scenarios were generated to test with the system and see how the home will react when the user performs certain activities. For instance is asleep during night and the system keeps the lights on or turn them off depending on the user preference priority.
- Reasoning System: A reasoning system capable of detecting conflicting situations (Hybrid System) and a smart hub (Vera) that accepts the request to query to either check or modify the state of the sensors connected to it.

The requirements in the previous section have been used to describe the contexts in Table 2 below. These contexts features were tested as part of the process, and the table shows the details of each context, illustrating problem description mentioned earlier.

Table 2.: Context Samples for User Preferences Management

Context Label	**Requirements**	**Context description**	**Context feature being tested**
C1	The system notifies the user on the content of a product, and advice accordingly based on preference priority set by the user.	The reasoning system uses preferences managed by the user to inform them if they should consume their favourite cake or not.	When the system is not sure if it should advise a user to buy his/her favourite chocolate cake and the user is also known to be diabetic, but base on the priority assigned by the user to their preferences (Health over. Pleasure), the system advises the user accordingly not buy the product.
C2	Automating the lighting aspect of the house or the user based on preferences set by the user.	Activity of the user going to bed and wants the house to keep the bedroom light 'on' when they are asleep.	The bedroom light can be "on" or "off" when user is sleeping during the night. However, in the case, the user prefers the light to be 'off' when he is asleep during the night. Then he assigns higher priority to "Comfort" over "Light", which allows the system to turn the light "off" after detecting that the user has gone to bed.
C3	The system turns off the lights on behalf of the user after sometime of not detecting any movement in the house.	When user is not at home, and there is not activity around, the house assumes there is no one in the house.	When no movement is detected around the house for certain period of time, and there is not pressure on the bed pad, the Hybrid System believes no one is the house, and if the lights were "on", the system turns all the lights "off".

Explanation of Contexts detection shown in Table 2:

(1) The context label C1 was detected based on the user configuration in prioritizing their preferences. For instance if a user wants the system to alert them about the sugar content of their favourite chocolate cake and wants the system to advise them to purchase the cake or not, they will have to rank their preferences between "Health" and "Pleasure". If the user is diabetic and wants to avoid sugar intake, he/she has to assign a higher priority to their "health" preference over "pleasure". The reasoning system will use this preference ranking to advise the user not to purchase a cake if it contains sugar.

(2) Context label C2 is also based on the user's preference ranking, but within a smart home, as the user needs to prioritize their preferences between "Light" and "Comfort". In this case, the user wants the comfort of keeping the lights 'off' when he/she is asleep during the night, they have now rank their preference of "Comfort", higher than "Light". When the user is on bed and there is no movement detected in bedroom, the system automatically switches 'off' the bedroom light after sometime. This context has been further emphasised in Table 5.

(3) The same way no movement has been detected in the bedroom over a period of time in C2, context label C3 has that similar attribute as the system does the same detection check to know if there is movement in the entire house. But in this situation, there are no conflicting preferences, as the users has configured their preferences to inform the system to switch 'off' all the lights in the house, when no movement is detected around the house.

Table 5 depicts the testing of one of the context in relation to C2, which indicate how the house should react when user performs certain activities of going to bed. An interface was also developed to collect and manage user's preferences ('Light' or 'Comfort' for instance) to measure how the system should react in the smart home especially when there are complexities of what the house should do for the user.

There are three contexts in Table 2, however, the test table emphasized on one of the contexts (C2), Here, the user is known to be in bed and possibly be asleep from 10 PM to 5:30 AM. It is also known that the user lives alone, can decide to prefer the lights 'off' when he is asleep during these times to provide more comfort compare to having the lights 'on' when he is asleep. So the user prioritize his preference of Comfort over Light, if for any reason he wants the bedroom light 'on" while he is asleep, he will have to set the preference of "Light" to be higher than "Comfort".

Test 1: First test was not successful. This failed because there were too many request to Vera, and this caused some delays, which made the bedroom light not to react instantly. An investigation on the increasing load and delay led to uncovering a problem with the way that Vera was contacted by the main Java thread running the MReasoner. The main Java thread running the MReasoner was updated, Vera system restarted and then the system was working effectively.

Test 2: It was successful as the smart hub (Vera) was fresh in accepting to query and modifying the state of the sensors/actuators connected to it via its own Wi-Fi network.

4.3. *Assisting users with respiratory health conditions*

A recent literature survey on context-aware solutions for asthma management by Quinde et al. (2018) shows, among other outcomes, a lack of context-aware solutions supporting the personalisation of asthma management. This is important given the fact that personalisation is the key to address the heterogeneity of asthma, which makes people suffering from this condition to have different triggers provoking their attacks, and to experience different respiratory symptoms when these attack occur.

A system allowing users to personalise their asthma management process has been under development, see Quinde et al. (2018),Quinde et al. (2019), and Quinde et al. (2019), and provides assistance as a context-aware mobile application. The system allows users to personalise the features of the prototype considering the specific characteristics of their asthma management process and these are then closely linked to contexts in the system. Some of these are:

(1) Sending alerts to people with asthma when the system finds a hazardous situations based on the configuration made by the user. For instance, if the user knows that they are affected by low temperature they can set up the system for it to alert them when the outdoor temperature is below 15° C.
(2) Sending alerts to the people chosen by the user when the user confirms that is having an emergency. For instance, if the user chooses to contact their partner in case of emergency, then the system will notify this person when the user activates the emergency notification protocol, which is activated by the user there is an emergency.
(3) Detecting potential hazards that may affect a person with asthma that does not know their triggers properly. In this case, the person cannot not set the monitoring control limits of the system because they does not know what to monitor. Thus, the system can provide support by using previous health deterioration experiences of the person with asthma in order to predict situations that may affect them.

It is important to highlight that the system not only supports people with asthma that know what to monitor, but also those who do not know what affect them yet. This uncertainty stage is common in, for example, recently-diagnosed people with asthma. More information about these and more requirements of the system, and how the U-CIEDP Augusto et al. (2018) was used to

Table 3.: Context Samples for Asthma Management

Context Label	**Requirements**	**Context description**	**Context feature being tested**
C1	Sending alerts to the PwA when the system finds a hazard based on the configuration made by the user.	Potential environmental hazards affecting a person with asthma that knows their triggers.	Notifying the PwA when the conditions may provoke an asthma attack on them.
C2	Sending alerts to the people chosen by the user when they confirm that are having an emergency.	The people to contact by a person with asthma in case of emergency.	Notifying people chosen to be contacted by the person with asthma in case of emergency (stakeholders).
C3	Detecting potential hazards that may affect a PwA that does not know their triggers properly.	Potential hazards affecting a person with asthma that does not know their triggers.	Notifying a PwA not knowing their triggers that there is a potential hazard considering previous hazardous experiences.

define these requirements can be found in: see Quinde et al. (2018), Quinde et al. (2018), Quinde et al. (2019), and Quinde et al. (2019),.

Requirements that led to the context description table: The requirements shown above were used as the foundation to describe the contexts and context features that needed to be tested as part of the development process. Table 3 shows the contexts and context features that were chosen to illustrate this case study.

The contexts described in Table 3 were obtained from the requirements of the system. Because of this, it is important to explain how these contexts are detected in order to understand the features of the system better and how they should be tested.

The detection of context C1 is based on the configuration made by a user that knows what their triggers and the monitoring indicators (e.g. temperature, humidity, PM10, etc.) that are linked to their triggers. The user sets up the system to alert them when the indicators are below or above their control limits at some specific places of interest that may be indoors or outdoors (e.g. home, nearby work). The system uses this configuration in order to monitor the environment and detect hazards.

Context C2 is also defined by a previous configuration done by the user who chooses the people that they wish to contact in case of emergency. The system allows the user to enter the details of the people to contact and they preferred way of contacting each of them (SMS, notification on their phones or email). Thus, when the user activates the emergency notification protocol, which is activated manually and not automatically in order to avoid false positives, the system notify the chosen people considering the configuration done by the user.

The least conventional way of detecting the contexts shown in Table 3 is the one for C3. In this case, the system uses case-based reasoning in order to collect information about previous experiences of the user that refer to normal or deteriorated health status. As the user, in this case, does not know what to monitor, the system creates cases representing the environment the user has been exposed to and the asthma health status (normal or deteriorated) associated with each case. Thus, the system uses previous experiences (previous cases) to assess whether the current environmental exposure of the user (new case) may be considered a hazard or not.

Table 6 is an application example of how the test table was used in this project to test the context feature related to Context C1 that is needed to deliver one of the required services in this project. Following CaSA to describe this application example, the context that matters is defined by the personalisation made by the user. The system then automatically defines the collectors needed to achieve the monitoring process requested by the user. These can be information gathered directly from devices or through APIs providing the required information. The automation in selecting the collectors and defining the intra/inter-context processing that is needed to achieve the monitoring process required by the user, can be considered as the biggest context-aware reasoning challenge of this project.

Table 6 illustrates three tests that were run until the expected outcome was achieved. The details of the test that were not successful when the Context Feature referred to the Context C1 was tested are shown below:

Test 1: Unsuccessful. The system crashed when the user did not choose the indicators to be monitored nor the control limits for the monitoring process. This failed test is reflected in the HCI row of Table 6. This was fixed by showing a message to the user indicating the monitoring process has to be configured before it could run. This message is shown when they open the personalisation feature of the app. Changes were made in the Preferences row of Table 6.

Test 2: Unsuccessful. The system assessed an old value for Outdoor Temperature as the Logical sensor (API) used to gather this value malfunctioned. This failed test is reflected in the HCI row of Table 6. This was fixed by making the system flexible enough to show details to the user explaining that it is not capable of gathering and up-to-date value for any indicator related to a sensor (physical or logical) that is malfunctioning. Changes were made in the HCI row of Table 6.

Test 3: Successful. The expected outcomes of this test are considered satisfactory.

5. Discussion

The context-aware system architecture has of course all the core benefits of all system architectures as a mental guide for how a system is composed and connected. It helps bring to a more conscious level the understanding of a specific type of system. Our state of the art analysis, brought to our attention all the previous work which seems to bear some resemblance to a context-aware oriented architecture were actually quite partial, in terms of being ad hoc for a specific application, or focusing on specific aspects such as security. Therefore one gain of CaSA is that now we have a more generic and up to date conceptual tool to guide our understanding of systems in this area. We do not claim perfection and we imagine it can be specialized and extended in different directions to suit specific needs in different projects. The other important question arising from this work is how CaSA helps at a practical level developers of context-aware applications? Here there are some lessons learnt from our use.

One way the system helped was unforeseen. It helped with personalization. All systems can enjoy with some attention to personalization, but for some this is really key for its success. Consider for example the system which helped users with asthma. There is a high range of factors (usually identified with places where these manifest more strongly) which can trigger symptoms. The level of CaSA which highlights the system as a convergence of collectors and aggregators led naturally to the consideration of sources of information which can be given different weights in the prediction and diagnosis tasks based on the significance for different individuals.

In a similar way the CaSA architecture bring to the forefront of developers the holistic picture of context processing the COATI tables highlight the infrastructure available and their enabling capabilities in relation to the lower levels of the contexts the developers are trying to implement. Some of these contexts require a trial and error process with a few iterations until context detection is achieved. The combination of these two views at a higher and lower level of context processing complement each other to provide a more realistic thinking about how to develop contexts. The CaSA architecture is more about the software processing steps, the COATI table links the infrastructure on one side with the goals through logical conditions.

The user-centred focus we put consistently in all our methods was also naturally beneficial, as CaSA is focused on stakeholders it helped to connect naturally and from the very beginning the needs and preferences of the stakeholders with the elements of the system feeding the decision making, for example in the asthma project connecting the numerical answer for each of the five questions of the Asthma Control Questionnaire (ACQ) with the classification process which guided advice and warnings.

The intra-context reasoning section preceding the inter-context reasoning was also found to be a useful guide in the organization of decision making processes from lower levels of data and into higher reasoning levels of the system.

6. Conclusions

This article considered the concept of context and context-awareness from two complementary perspectives. On one side we looked at high level system architectures for context-aware systems and on another side we looked at the lower level process of finding logical connections between the infrastructure available in a Smart Environment and the context trigger conditions.

We described how the CaSA architecture provides a unifying view to the current state of the art systems and how the COATI methodology provides a link between testing and validation to connect different processes often disconnected. These two methodologies can be used in isolation, we also described they are complementary to an ecosystem of methodologies and tools we developed to assist developers of smart environments, such as the SEArch architecture and the U-CIEDP process.

One challenge when trying new concepts at a systems abstract level is how to test the adequacy of the idea. Sometimes it takes years to have a sequence of projects where to try and test whether the concepts work in real systems. We were fortunate to have a number of ongoing projects developing concurrently in the last few years which allow refinement and maturity of the concepts. Although we do not claim three projects is a safety threshold to confirm adequacy of a system architecture it certainly provides a reasonable number of contexts and diversity of situations with three different set of overlapping but different infrastructures.

So far one of the hindering aspects for these combination of methodologies to be more practical and widely used is the lack of automated support. Part of our team is working now on automating translations from one step to the next one and to facilitate traceability of contexts. We believe it is now relevant that this initial approach is communicated and offered to our community for a more extensive assessment.

We use tables as an easy and effective way to summarize the essential features which support context detection, however this relationship may be better served with a more sophisticated graphical interface which is more expressive, more visually appealing and perhaps identify aspects of the process we cannot capture with the simple tables. This we think could be an interesting issue for our community to explore.

Acknowledgement

Research reported in this article benefited enormously by the continuous feedback from various stakeholders in various projects. These include Asthma UK, the London Housing Network, the Dementia Services of the London Boroughs of Enfield and Croydon, and the EU/UK/CHISTERA funded project "Secure Accessibility for the Internet of Things (SUCCESS)".

Table 4.: Table Sample Tests for ADLs Scenario in Context EloCF1

	Enablers	Assumptions Initial values	Test 1	Test 2
Context Description	The user leaves the house during a time out of the timetables configured in the interface. Possible elopement detected.			
Expected Outcome (s)			The user receives an alert in the mobile app asking about the activity/behaviour.	
Real Outcome (s)			The user does not receive an alert	The user receive an alert with the proper message related to the activity detected.
Sensors	Main Door Motion sensor (MDM)	No movement. MDM=0	MDM=1	MDM=1
	Main Door Reed sensor (MDR)	Door is closed. MDR=0	MDR=1 two seconds after MDM changed to one.	MDM=1 when in the last 3 seconds MDM=1 has been one.
Network	Z-wave (Vera hub)	Vera has connection with the sensors involved.	There is a connection with MDM and MDR, and updating their values.	There is an appropriate connection.
Database	Preferences database	Timetable wherein the user is usual leaves the house.	Preferences has been stored correctly.	The times have been stored correctly.
	Monitoring database	Stores the context being monitored.	Preferences has been loaded correctly.	The times has been loaded correctly.
Reasoners	Connection with Sensors and Server.	The tool connects with Vera and MReasoner.		
	Context-Aware Reasoner	The tool is running with all states loaded. The action goOut, goIn, elopementSchedule and elopementAlertUser are false detected and goIn is not detected before time alert user span (Tau)	The action goOut is false (it is not detected). The action goIn is false (it is not detected). elopementAlertUser is false. elopementSchedule is false.	The action goOut is detected. The action goIn is false (it is not detected). elopementAlertUser is true. elopementSchedule is false.
HCI	User mobile app GUI	The APP is running without errors.	User has not received an alert in app GUI.	User has received an alert in app GUI.
Preferences	User web settings GUI	The schedule was setting from initial time Ta to final time Tb.		
Users	Person with Dementia (PwD)/ Caregiver		The user leaves the house at a time between Ta and Tb and remains outside.	The user leaves the house at a time between Ta and Tb and remains outside.

Table 5.: Sample Tests for User Preferences Scenario in Context C2

	Enablers	Assumptions Initial values	Test 1	Test 2	Test 3	Test 4	Test 5
Context Description	Activity of the user going to bed and wants the house to keep the bedroom light 'on' when they are asleep.						
Expected Outcome (s)			Movement sensor is expected to be active, when movement is detected around the bedroom.	Movement sensors become inactive when no movement is detected in the bedroom.	Pressure pad should not be idle when user is on the bed.	The light should go 'off' after detecting that the user is on bed after 30 s of being and no movement detected.	The light should go 'off' after Vera is restarted and remain off even if movement is detected unless user gets out of bed
Real Outcome (s)			The movement sensor is active.	The movement sensor becomes inactive.	Pressure pad becomes active.	The light remains 'on' after 30 s, due to delay.	Light goes 'off' after 30 s of no movement detected in the room.
Sensors	Light Actuator	Light is 'on'	Light Actuator=1	Light Actuator=1	Light Actuator=1	Light Actuator=1	Light Actuator=0
	PIR Sensor	PIR, movement detected	PIR Sensor=1	PIRSensor=0	PIRSensor=0	PIRSensor=0	PIRSensor=0 or PIRSensor=1
	Pressure pad	Pressure pad idle	Pressure pad idle=1	Pressure pad idle=1	Pressure pad idle=1	Pressure pad idle=0	Pressure pad idle=0
Network	Z-wave (Vera hub)	Vera is connected to sensors and actuators.	The connection is active, as long as the Vera box is 'on'.	Vera accepts request to query or modify sensor/actuators state.	Vera keeps updating change of values for the sensors/actuators.	Vera gets restarted to be refreshed due to too many requests.	Vera keeps updated the values for the light actuators instantly.
Database	Database records	Comfort =8 Light=4					Preferences are saved in the database for use.
Reasoners	Connected to sensors and database servers	ssr(([-][30s.] #PadIdle ˆ #Movement ˆ prefComfort) ->#Light);	The MReasoner continue to check if movement is triggered which continues to keep the light 'on'.	Once movement (#Movement) is no longer detected the MReasoner indicates that movement is no longer active.	The MReasoner also indicates of the state of the pressure pad has changed to pad not idle (#PadIdle) once the user is on bed.	MReasoner advice turns light off (#Light) when no movement (#Movement) is detected and the user is on bed.	The MReasoner checks the preference database. If the case he ranks comfort more (prefComfort), then the light goes 'off' (#Light).
Learners	N/A	N/A	N/A	N/A	N/A	N/A	N/A
HCI	Desktop interface						Comfort higher than Light.
Preferences	Preferences ranking	'Comfort' over 'Light'					
Users	Home user=Bob	Sets priority to prefer 'Comfort' over 'Light'	Bob moving around the bedroom.	Bob is on bed.	Bob is on bed.	Bob is falling asleep.	Bob sleeps in comfort.

Table 6.: Sample Tests for Asthma Management in Context C1

	Enablers	Assumptions Initial values	Test 1	Test 2	Test 3
Context Description			Potential environmental hazards affecting a person with asthma that knows their triggers.		
Expected Outcome (s)			Delivering a notification to the PwA when any of the indicators are out of the control limits.		
Real Outcome (s)			The movement sensor is active.	The movement sensor becomes inactive.	Pressure pad becomes active.
Sensors	Indoor temperature	19C is the lower control limit.	T <19C	T <19C	Light Actuator=1
	Indoor humidity	60% is the upper control limit.	H >60%	H >60%	H >60%
	Outdoor temperature	12C is the lower control limit.	T <12C	T <12C	T <12C
	Outdoor humidity	80% is the upper control limit.	H >80%	H >80%	H >80%
Network	Internet	Permanent connection			
	Zwave (Vera)	Permanent connection			
Database	Preferences DB	User set up the control limits.	Provides the monitoring control limits.	Provides the monitoring control limits.	Provides the monitoring control limits.
	Monitoring DB		Stores contextual data.	Stores contextual data.	Stores contextual data.
	Cases DB	N/A	N/A	N/A	N/A
Reasoners	Context-aware Reasoner		Monitors the indicators based on users preferences.	Monitors the indicators based on users preferences.	Monitors the indicators based on users preferences.
Learners	N/A	N/A	N/A	N/A	N/A
HCI	Mobile application GUI		HCI component does not show the notification.	Shows a notification alerting about a hazard but does not inform that it is based on an old value of Temperature.	Notification alerts about the hazard and shows the latest reading times of the indicators.
Preferences	Mobile application		Allows user to personalise the control limits for the indicators.	- Shows a message indicating that they have to configure the monitoring process before it can run. - Allows user to personalise the control limits for the indicators.	- Shows a message indicating that they have to configure the monitoring process before it can run. - Allows user to personalise the control limits for the indicators.
Users	Person with asthma		Confirms health status.	Confirms health status.	Confirms health status.
	Stakeholders	N/A	N/A	N/A	N/A

References

Aarts, E. and R. Roovers (2003). Ic design challenges for ambient intelligence. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE 03, USA, pp. 10002. IEEE Computer Society.

Abowd, G. D., A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles (1999). Towards a better understanding of context and context-awareness. In H.-W. Gellersen (Ed.), *Handheld and Ubiquitous Computing*, Berlin, Heidelberg, pp. 304–307. Springer Berlin Heidelberg.

Allegre, W., T. Burger, J.-Y. Antoine, P. Berruet, and J.-P. Departe (2013). A non-intrusive context-aware system for ambient assisted living in smart home. *Health and Technology 3*, 129–138.

Atzori, L., A. Iera, and G. Morabito (2010). The internet of things: A survey. *Computer Networks 54*(15), 2787 – 2805.

Augusto, J., A. Aztiria, D. Kramer, and U. Alegre (2017). A survey on the evolution of the notion of context-awareness. *Applied Artificial Intelligence 31*(7-8), 613–642.

Augusto, J., J. Gimnez-Manuel, M. Quinde, C. Oguego, M. Ali, and C. James-Reynolds (2020). A smart environments architecture (search). *Applied Artificial Intelligence 34*(2), 155–186.

Augusto, J., D. Kramer, U. Alegre, A. Covaci, and A. Santokhee (2018). 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 17*(1), 115–130.

Augusto, J. C. (2014, June). User-centric software development process. In *2014 International Conference on Intelligent Environments*, pp. 252–255.

Augusto, J. C., V. Callaghan, D. Cook, A. Kameas, and I. Satoh (2013). Intelligent environments: a manifesto. *Human-centric Computing and Information Sciences 3*(1), 12.

Ayed, D., D. Delanote, and Y. Berbers (2007). Mdd approach for the development of context-aware applications. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, and L. Vieu (Eds.), *Modeling and Using Context*, Berlin, Heidelberg, pp. 15–28. Springer Berlin Heidelberg.

Chaker, H., M. Chevalier, C. Soulé-Dupuy, and A. Tricot (2011). Business context information manager: An approach to improve information systems. In M. Beigl, H. Christiansen, T. R. Roth-Berghofer, A. Kofod-Petersen, K. R. Coventry, and H. R. Schmidtke (Eds.), *Modeling and Using Context*, Berlin, Heidelberg, pp. 67–70. Springer Berlin Heidelberg.

Evans, C., L. Brodie, and J. C. Augusto (2014, June). Requirements engineering for intelligent environments. In *2014 International Conference on Intelligent Environments*, pp. 154–161.

Furno, A. and E. Zimeo (2013). Context-aware design of semantic web services to improve the precision of compositions. In P. C. Vinh, N. M. Hung, N. T. Tung, and J. Suzuki (Eds.), *Context-Aware Systems and Applications*, Berlin, Heidelberg, pp. 97–107. Springer Berlin Heidelberg.

Hassani, A., P. D. Haghighi, F. Burstein, and S. Davey (2017). Context aware data synchronisation during emergencies. In P. Brézillon, R. Turner, and C. Penco (Eds.), *Modeling and Using Context*, Cham, pp. 406–417. Springer International Publishing.

Hassani, A., P. D. Haghighi, P. P. Jayaraman, and A. Zaslavsky (2017). A context aware framework for mobile crowd-sensing. In P. Brézillon, R. Turner, and C. Penco (Eds.), *Modeling and Using Context*, Cham, pp. 557–568. Springer International Publishing.

Kolbe, N., A. Zaslavsky, S. Kubler, J. Robert, and Y. Le Traon (2017). Enriching a situation awareness framework for iot with knowledge base and reasoning components. In P. Brézillon, R. Turner, and C. Penco (Eds.), *Modeling and Using Context*, Cham, pp. 41–54. Springer International Publishing.

Kramer, D. and J. C. Augusto (2017). Supporting context-aware engineering based on stream reasoning. In P. Brézillon, R. Turner, and C. Penco (Eds.), *Modeling and Using Context*, Cham, pp. 440–453. Springer International Publishing.

Kramer, D., J. C. Augusto, and T. Clark (2014). Context-awareness to increase inclusion of people with ds in society. In *AAI-14 Workshop Artificial Intelligence Applied to Assistive Technologies and Smart Environments*, Quebec, Canada.

Li, X., M. Eckert, J.-F. Martinez, and G. Rubio (2015, Aug). Context aware middleware architectures: Survey and challenges. *Sensors 15*(8), 2057020607.

Martínez Madrid, N., J. M. Fernández, R. Seepold, and J. C. Augusto (2013). *Sensors for Ambient Assisted Living and Smart Homes*, pp. 39–71. Berlin, Heidelberg: Springer Berlin Heidelberg.

McNaull, J., J. C. Augusto, M. Mulvenna, and P. McCullagh (2014). Flexible context aware interface for ambient assisted living. *Human-centric Computing and Information Sciences 4*(1).

Neumann, M. A., T. Riedel, P. Taylor, H. R. Schmidtke, and M. Beigl (2011). Monitoring for digital preservation of processes. In M. Beigl, H. Christiansen, T. R. Roth-Berghofer, A. Kofod-Petersen, K. R. Coventry, and H. R. Schmidtke (Eds.), *Modeling and Using Context*, Berlin, Heidelberg, pp. 214–220. Springer Berlin Heidelberg.

Oguego, C., J. Augusto, A. Muoz, and M. Springett (2018). Using argumentation to manage users preferences. *Future Generation Computer Systems 81*, 235 – 243.

Papadopoulou, E., S. Gallacher, N. K. Taylor, M. Howard Williams, and F. Blackmun (2013). Context-aware user preferences in systems for pervasive computing and social networking. In P. C. Vinh, N. M. Hung, N. T. Tung, and J. Suzuki (Eds.), *Context-Aware Systems and Applications*, Berlin, Heidelberg, pp. 10–17. Springer Berlin Heidelberg.

Park, S.-H., Y.-J. Han, and T.-M. Chung (2007). Context-aware security management system for pervasive computing environment. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, and L. Vieu (Eds.), *Modeling and Using Context*, Berlin, Heidelberg, pp. 384–396. Springer Berlin Heidelberg.

Peko, G., C.-S. Dong, and D. Sundaram (2014). Adaptive sustainable enterprises: A framework, architecture and implementation. In P. C. Vinh, V. Alagar, E. Vassev, and A. Khare (Eds.), *Context-Aware Systems and Applications*, Cham, pp. 293–303. Springer International Publishing.

Perera, C., A. Zaslavsky, P. Christen, and D. Georgakopoulos (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials 16*(1), 414–454.

Quinde, M., N. Khan, and J. C. Augusto (2018). Personalisation of context-aware solutions supporting asthma management. In K. Miesenberger and G. Kouroupetroglou (Eds.), *Computers Helping People with Special Needs*, pp. 510–519. Springer.

Quinde, M., N. Khan, and J. C. Augusto (2019). Case-based reasoning for context-aware solutions supporting personalised asthma management. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada (Eds.), *Artificial Intelligence and Soft Computing*, Cham, pp. 260–270. Springer International Publishing.

Quinde, M., N. Khan, J. C. Augusto, and A. van Wyk (2019). A human-in-the-loop context-aware system allowing the application of case-based reasoning for asthma management. In V. G. Duffy (Ed.), *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Healthcare Applications*, Cham, pp. 125–140. Springer.

Quinde, M., N. Khan, J. C. Augusto, A. van Wyk, and J. Stewart (2018). Context-aware solutions for asthma condition management: a survey. *Universal Access in the Information Society*.

Souabni, R., I. Bayoudh Saadi, Kinshuk, and H. Ben Ghezala (2014). A comprehensive view of ubiquitous learning context usage in context-aware learning system. In P. C. Vinh, V. Alagar, E. Vassev, and A. Khare (Eds.), *Context-Aware Systems and Applications*, Cham, pp. 316–326. Springer International Publishing.

Tahir, H. and P. Brézillon (2013). Contextual graphs platform as a basis for designing a context-based intelligent assistant system. In P. Brézillon, P. Blackburn, and R. Dapoigny (Eds.), *Modeling and Using Context*, Berlin, Heidelberg, pp. 259–273. Springer Berlin Heidelberg.

Weiser, M. (1991). The computer for the 21st century. *Scientific American 265*, 94104.

Zacarias, M., H. S. Pinto, and J. Tribolet (2007). Integrating engineering, cognitive and social approaches for a comprehensive modeling of organizational agents and their contexts. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, and L. Vieu (Eds.), *Modeling and Using Context*, Berlin, Heidelberg, pp. 517–530. Springer Berlin Heidelberg.

List of Figures

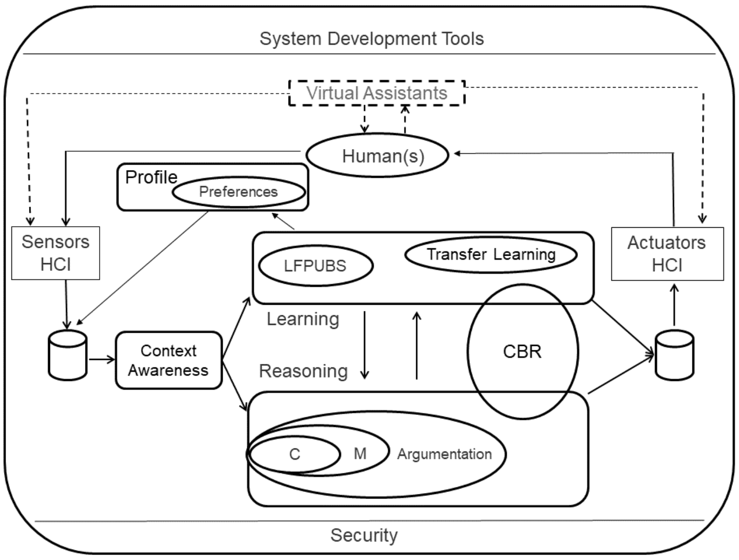

Figure 1.: Smart Environment Architecture (SEArch)

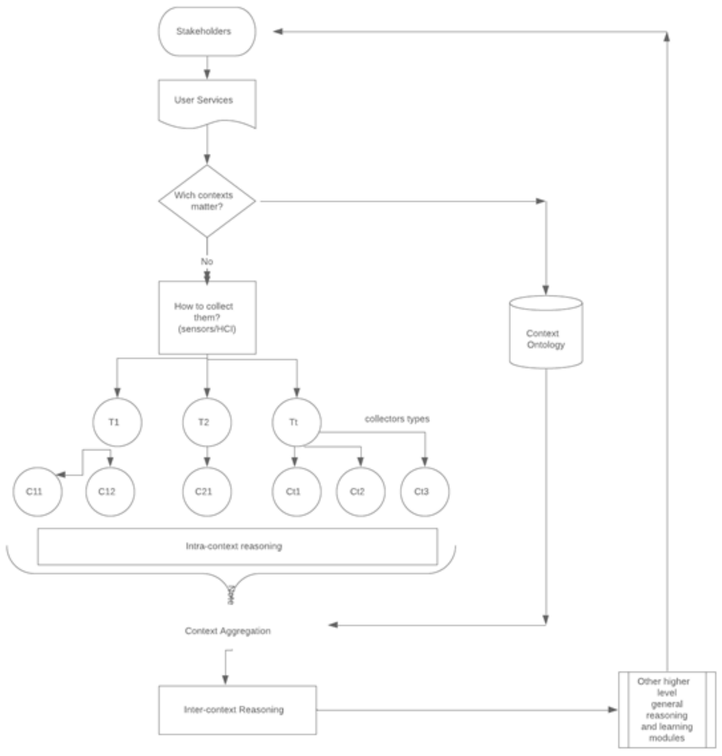

Figure 2.: Context-aware System Architecture (CaSA)

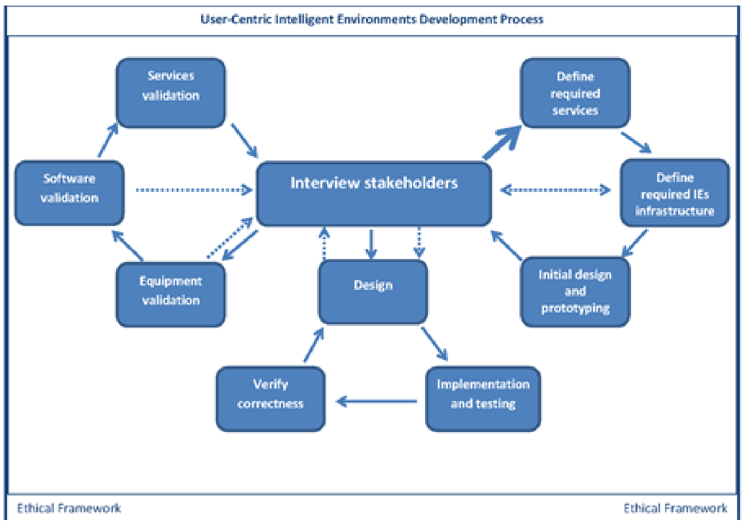

Figure 3.: User-centred Intelligent Environments Development Process (U-CIEDP)

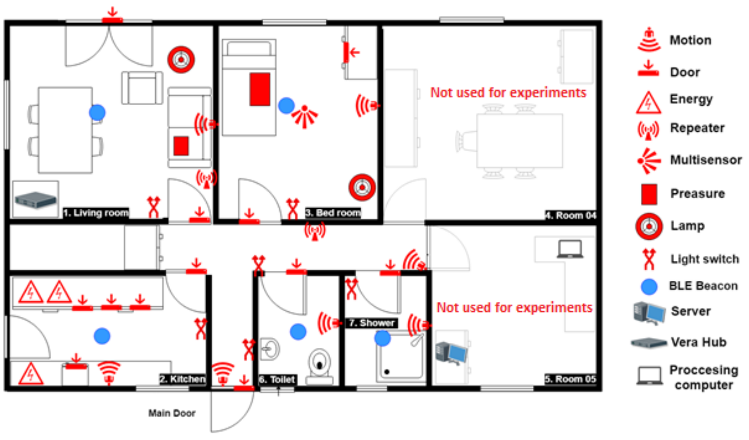

Figure 4.: Distribution of technology in the Smart Home environment.