Workshops at 18th International Conference on Intelligent Environments (IE2022)
H.H. Alvarez Valera and M. Luštrek (Eds.)

doi:10.3233/AISE220056

Engineering Context Updates

Juan Carlos AUGUSTO[a,1]
[a] *Research Group on Development of Intelligent Environments*
Department of Computer Science, Middlesex University London, U.K.

Abstract. Context has become a concept with practical applications in Computer Science. It is a complex concept which has been used often, however one which is not well understood, and its use is often superficial. This article highlights some considerations which may be important for our technical community to think more explicitly and to investigate in further depth. The contribution of this article is in highlighting the ramifications of system updates in the contexts being considered. The goal is to encourage future closer analysis and the development of much needed design and development tools that can provide support for developing systems which are more resilient to context updates.

Keywords. Contexts, Context-awareness, Requirements, User Preferences.

1. Introduction

Technical publications from a decade ago where highlighting the importance of context variability and their relationship with requirements (see for example: [1,2]), the state of the art is still very much the same. No much progress have been made in this direction. This work aims to contribute to that discussion. Some work has been done to help link some of the contexts in the system with change by forecasting context variability (e.g., [3]). We feel the problem is much wider and requiring a more holistic approach, and developers in this area ought to have a more comprehensive approach to identifying the inter-relations between contexts and of contexts with other important system concepts.

Here we take a wider view, where we consider context-context, contexts-requirements and context-user preferences interactions as areas to discuss and represent more explicitly during system development. We take as a departing point a recent theoretical redefinition of context and context-awareness which more explicitly take into account stakeholders and we use this framework to provide a first approach to context revision with a more identification of possible system concept interactions which we propose require more careful consideration.

We illustrate our considerations with the support of a scenario we borrow from [4]. This considers an Ambient Assisted Living system where we refer to as 'beneficiaries' the main recipients of the system services:

Beneficiary B lives in a smart home and has two main weekly routines from Monday to Friday and during weekends. Monday to Friday routines involve waking up at 7AM to be ready to go to work at 8AM. B expects some automation services. During the process of getting up in the morning B typically gets up from bed, goes to the bathroom, then to the kitchen, has breakfast, and goes out of the house to work. The pressure pad

[1] Corresponding Author, E-mail: j.augusto@mdx.ac.uk.

in the bed and motion sensors in the bedroom allows the system to understand when B is physically getting up and other motion sensors in the corridor, the bathroom and the kitchen allows tracking B's trajectory. The system turn on lights in the next relevant room and turns them off in rooms where they are perceived not to be useful anymore. When the user enters the kitchen the system turns on the radio. The sensors in the doors of the kitchen coverts and fridge as well as the devices used, such as the kettle or the microwave oven, provide clues of the user preparation of breakfast. Meanwhile the system can present information on weather and air quality air for the work area of the city, which helps B's decision making about transport choices to reach the workplace. It could also happen that B gets up during the night to go to the bathroom and the system is expected to understand this is not the same than the breakfast routine to go to work. One of B's elderly parents, PB, also lives in the same house and has been increasingly experiencing symptoms of senile dementia, with increased safety risks, so the system is expected to differentiate between different beneficiaries going to the bathroom and trying to go out of the home and at what times these events are expected. Guiding lights can reduce PB's risks of falling and also help with her orientation. Getting up in the middle of the night for the bathroom should not trigger actions in the kitchen. B's leaving going out of the house during dark hours is fine but it may be dangerous for PB given the spatial and temporal confusion experienced for that person. Table 1 summarizes this.

Table 1. Examples of context descriptions based on the smart home scenario..

	Context Concept: front door use	
Name	Front_door_use1	Front_door_use2
Benficiary	B	PB
Activation	Any time & at Front door & Open door & Leave house	8 AM–8 PM & at Front door & Open door & Leave house
Effect	Do nothing	Inform B
	Context Concept: going to bathroom	
Name	Going_to_bathroom1	Going_to_bathroom2
Benficiary	B	PB
Activation	7:00–7:15 & Gets up from bed & Bedroom movement	7:00–9:00 & Gets up from bed & Bedroom movement
Effect	Turn on lights in bedroom, corridor, and bathroom	Turn on lights in bedroom, corridor, and bathroom
	Context Concept: getting up routine in process	
Name	Getting_up_AM_routine1	Getting_up_AM_routine2
Benficiary	B	PB
Activation	7:00–7:15 Enter bathroom	7:00–9:00 Enter bathroom
Effect	Turn on radio and kettle	Notify B
	Context Concept: skipping lunch	
Name	Skipping_lunch1	Skipping_lunch2
Benficiary	B	PB
Activation	12:00–2:00	12:00–2:00
Effect	Do nothing	Remind PB
	Context Concept: being at kitchen	
Name	Being_at_Kitchen1	Being_at_Kitchen2
Benficiary	B	PB
Activation	Kitchen PIR activated	Kitchen PIR activated
Effect	Log activity	Log activity

The content of the table reflects some of the contexts in that described scenario and each context is defined using the format of the following four context "aspects": *[Name, Beneficiary, Activation, Effect]*. There are 8 different contexts defined, four for beneficiary B in the second column and four for beneficiary PB in the third column.

We keep the description of contexts in the table and in the rest of the paper at a high level to focus on the concepts that matter, and not get distracted with the lower level close to the technology details. For example, "Gets up from bed" can be interpreted as the occurrence of a set of lower level sensor activations such as the bed pressure sensor is on for some time then is off for a few seconds and overlapping with these changes the bedroom passive infrared sensor has been triggered for a while.

2. Conceptual Grounding

This paper forms part of a line of work on revisiting the notion of context and context-awareness and their link with system stakeholders, especially to developers [5,6,7,4]. The main aim of that line of work is to reassess these concepts from a point of view that is more useful for modern system developers interested in context-awareness and facilitates discussion from a more Software Engineering oriented point of view. Here we retake from previous publications and explore in more detail context change and context revision from the developers' point of view. First we start with our two stakeholder-centric definitions[2] of Context and of Context-awareness:

Context: the information which is directly relevant to characterize 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.

Where we emphasize that the usefulness of these key concepts in Intelligent Environments[3] is on how effectively it provides services to humans. This becomes explicitly built in the theory explained in the coming sections and becomes a core of the following discussions.

We assume a theory of Contexts for Intelligent Environments (CIEn), as recently presented and illustrated in [4], represented through a structure focusing on a set of contexts and a set of operators defined over them:

CIEn=<B, S, C, Ops, A, OW> where:

$B=\{B_1, B_2, \ldots, B_b\}$ is a finite set of beneficiaries,

$S=\{S_1, S_2, \ldots, S_s\}$ is a finite set of services,

$C=\{C_1, C_2, \ldots, C_c\}$ is a finite number of contexts,
with C_i=*[Name,Beneficiary,Activation,Effect],*

$Ops=\{Op_1, Op_2, \ldots, Op_o\}$ is a finite set of context operations,

$A=\{Al_1, Al_2, \ldots, Al_a\}$ is a finite set of algorithms to process context information,

$OW=\{\ldots OW_i \ldots\}$ is a finite set of instances OW_i of observations of the real world.

[2] Different to the popular Dey's definitions [8,9].

[3] By Intelligent Environments [10] here we mean sensing systems with intelligent software to provide context-aware services, and we consider the concepts discussed are very much applicable to closely related systems as those discussed in areas such as IoT, Ubicomp, Percomm, Ambient Intelligence, AAL, etc. [11].

Be L a language and $\wp(L)$ a set of valid sentences in that language. For $C_i \in \mathbb{C}$ the "Activation (Condition)" can be defined as $AC_i=\{w^1_{ci}, ..., w^n_{ci}\}$ and given $OW_j=\{w^1_j, ..., w^m_j\}$, where $OW_i \subset \wp(L)$, $OW_i \not\models \bot$, $AC_i \subset \wp(L)$, $AC_i \not\models \bot$ (i.e., OW_i and AC_i are assumed to be consistent well-formed formula in L), we can also define a context satisfaction function $\vee$: $\mathbb{C}$ x $\mathbb{OW}$ --> *Bool*, which decides whether the observable world of the system satisfies the context or not. This operator will basically check whether $OW_j \models AC_i$ and it can be defined as:

$\surd(C_i, OW_j)$=

"true" iff forall $w^x_{ci} \in AC_i$ there exists: $w^y_j \in OW_j$ such that $f(w^x_{ci}, F_c, w^y_j, F_0)$;

"false" otherwise

where $f(w^x_{ci}, F_c, w^y_j, F_0)$ is a fulfilment function checking that well-formed formula w^x_{ci} in the higher-level context language F_c is semantically fulfilled by the meaning of well-formed formula w^y_j in the lower level language of sensing in the observable world F_0.

The concepts above tie more closely with the notion of being stakeholders' centred as an IE system can be conceived as an optimization function which maximizes services performance in alignment to user expectations as follows [4]. Consider the following additional concepts:

Beneficiary Context Perception (BCP) is the context as perceived by the beneficiary, where Perception here is understood as measured with the available infrastructure.

Beneficiary Context Expectations (BCE) are the services the beneficiary expects in a given context.

If we use a function $BCE(p_i, s_j, c_k, b; t)$ which measures how a beneficiary (b) prefers (p_i) a contextualized (c_k) service (s_j) at a certain time (t), and a function $BCP(p_i, s_j, c_k, b; t)$ which measures how that beneficiary perceives the actual delivery of that service, then we can define the level of *Service Achievement Satisfaction of an IE system for a beneficiary b at time t* as:

$$SAS(IE, b, t) = \sum_{i=1..p; j=1..s; k=1..c} |BCE(p_i, s_j, c_k, b; t) - BCP(p_i, s_j, c_k, b; t)| = 0$$

That is the IE system should aim to achieve the best possible alignment of the user expectation with the user perception of systems across all services at all times. A generalization $SASm(IE, B, t)$ for multiple users was also given in [4].

3. Context Changes

Contexts in a system can change. So the originally envisaged contexts may need revision. This could be because at early stages of design developers are still evolving these contexts, because careful system testing and validation uncovered problems with the current contexts being considered, or because after the system has been working for some time it requires adjustments as part of the system maintenance. How should this be done?

Some basic operations defined over contexts we may like to consider for a start are the obvious (we will call them *consistency preserving updates*):

- Deletion(C_i, $\mathbb{C}$, $\mathbb{C}$'), the deletion of C_i from $\mathbb{C}$ with result $\mathbb{C}$': $\mathbb{C}' = \mathbb{C} - \{C_i\}$),
- Addition(C_i, $\mathbb{C}$, $\mathbb{C}$'), the addition of C_i to $\mathbb{C}$ with result $\mathbb{C}$': ($\mathbb{C}' = \mathbb{C} \cup \{C_i\}$), we assume C_i has a different name to those already in $\mathbb{C}$. Developers need to consider carefully that this may add a context $C_i=[Name1, Beneficiary, Activation, Effect1]$ and $\mathbb{C} \supseteq C_j=[Name2, Beneficiary, Activation, Effect2]$. If $\{Effect1, Effect2\} \models \bot$ it will require a system capable to handle inconsistency.

On another hand, if$\{Effect1, Effect2\} \not\models \perp$ it will require a system capable to handle non-determinism allowing the system to choose between two possible courses of actions effects in the same situation.

- Modification(C_i, $\mathbb{C}$, $\mathbb{C}$'), the modification of C_i from $\mathbb{C}$ with result $\mathbb{C}$'. Here several types of modifications may be distinguished. (1) A modification of name must use a name different to names of other existing contexts. Are contexts only distinguished by their name in the system? (2) A Beneficiary modification could lead to the potentially problematic situation of having $C_i=[Name1, Beneficiary, Activation, Effect]$ when $\mathbb{C} \supseteq C_j=[Name2, Beneficiary, Activation, Effect]$. (3) An Activation Condition modification could be made by eliminating part of the Activation Condition; or when the modification is by augmenting the conditions: where C_i has $AC_i=\{w^1_{ci}, ..., w^n_{ci}\}$ and C'_i gets Activation Condition $AC'_i=\{w^1_{ci}, ..., w^n_{ci}, w^{n+1}_{ci}\}$, provided AC'_i remains consistent; or when the modification is altering part of the condition: so that C_i has $AC_i=\{w^1_{ci}, ..., w^n_{ci}\}$ and C'_i gets Activation Condition $AC'_i=\{w^1_{ci}, ..., w^{n'}_{ci}\}$, provided AC'_i remains consistent. (4) Finally a modification of the Effect part could lead to the issues raised as for the Addition operation.

As examples of modifications, consider a context (using the format $C_i=[Name, Beneficiary, Activation, Effect]$):

> C_1={Going_to_bathroom1, B, 7:00–7:15 & Gets up from bed & Bedroom movement, Turn on lights in (bedroom, corridor, and bathroom)}

We can improve the name to make it more specific, and also add a condition that to request that is only triggered when movement in the bedroom is followed by movement in the corridor:

> C'_1={Going_to_bathroom_earlyAM, B, 7:00–7:15 & Gets up from bed & followed_by(Bedroom movement, Corridor movement), Turn on lights in (bedroom, corridor, and bathroom)}

Or we can also transform it into a more generic one by removing some conditions:

> C''_1={Going_to_bathroom, followed_by(Bedroom movement, Corridor movement) & bathroom dark, Turn on lights in bathroom}

From a higher perspective at an engineering level is less clear how change should be managed. There are publications which focus on an automation level. For example a system can learn statistically that the user is shifting some habits starting them earlier or later in the day or on different days, so rules can be modified or added or deleted, however these modifications can introduce conflicts with other rules, including contradictions, and although in theory some of these can be handled through automated formal verification (e.g., model checking) either on the at design time or at run-time, these are far from being a reality. Besides there are new concepts or modifications of contexts which can involve changes the system cannot collect statistically. Hence the engineering process of these systems is still one that requires human developers in the loop. Now, what developers should take into account when considering modifications to the contexts of a system?

One system concept that comes to mind is obviously Requirements. A change in a key system element should be reflected in a change of requirements. The problem is although requirements can be represented in formal languages; they are usually represented through sentences and descriptions written in one of the so-called 'natural languages' (e.g., English). Still there should be a synchrony between requirements and contexts and changes in contexts should most likely involve a revision of the

requirements. So we can supplement the Contexts for Intelligent Environments (CIEn) given above with some accessory concepts which are more related to the process of engineering such systems. For example, we can assume a finite set of requirements: $\mathbb{R}= \{R_1, R_2, \ldots, R_r\}$. Now the tricky part comes on the vagueness often associated with requirements specification, so it could be that a requirement R_i is actually realized through one or more contexts C_j. For example, "lights should be managed automatically" may lead to the consideration of a daylight context and a darkness context to be considered. It could also be a context is related to more than one requirement. For example, turning off lights in a room when is empty can relate to a requirement focused on money savings and to another one on reducing carbon footprint. This Requirements-Contexts relationship is likely not to be a one-to-one mapping. Some way of keeping track of this relation needs to be used so that modifications during system build up and later during system maintenance can be used to more safely handle change at context-awareness level. As an initial simplification we will assume here a relationship 'one to many' between requirements and contexts and we will assume there is a mapping as in Table 2 (let us call this an *R-C Mapping Table*).

Table 2. Requirements to Contexts mapping.

Requirements	Contexts
R_1	...
R_i	$C^i_1 \ldots C^i_m$
R_r	...

Let us assume we know which contexts we want to change. How do we change them? Consistently with the case we have been developing for a more explicitly stakeholders' centred approach to contexts engineering in Intelligent Environments here we propose that a highly important element to use as a guide is the so called user preferences [12]. So let us assume each beneficiary B_i has associated a preference structure, a partial order of preferences $\mathbb{P}= \{P_1, P_2, \ldots, P_p\}$. For example, in terms of the entertainment options at home, a younger member of the family may have preferences for certain T.V. programs and music whilst older adults in the house prefer a different set of T.V. programs and music and/or in a different order of priority. A context may relate to several user preferences and a preference may relate to several contexts. For example, a context of putting calming music when a member of the family is stressed can link to preferences of music genre, also music volume, whilst a preference on a music genre could relate to both a stressful context and a daily yoga practicing context. Again as a simplification here we will consider a 'one to many' relationship and a corresponding table reflecting that, and we will assume there is a mapping as in Table 3 (let us call this a *C-P Mapping Table*).

Table 3. Contexts to Preferences mapping.

Contexts	Preferences
C_1	...
C_j	$P^k_1 \ldots P^k_n$
C_c	...

All of these main ingredients are highlighted in Figure 1. Instead of the usual trial and error of 'tweaking the code and see whether it works' we advocate here for a more methodic approach where preferences, contexts and requirements are looked as a whole

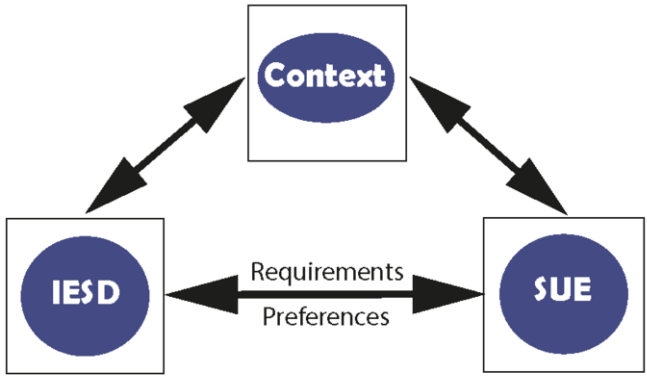

Fig. 1. To modify Contexts, IE System Developers (IESD) need to have into account System Users Experience (SUE) and, especially, their priorities and system requirements.

How change should proceed? We propose that if context C_j is being changed into C'_j then developers should:

1. Obtain the list of preferences $P^k_1 \ldots P^k_n$ associated with C_j from *C-P table*
2. If context change implies revising preferences then update *C-P table*
3. Make changes to C_j into C'_j using *consistency preserving updates* and also in a way it takes into account its effect on *SAS*
4. If C'_j implies revising requirements then update *R-C table*

For example, Say we have a context: C_{22}= {Getting_up_AM_routine2, PB, 7:00–8:00 & Enter bathroom, Notify B} and we need to modify it to account for the seasonal variations given beneficiary PB wakes up later in winter. The developers may be tempted to modify the Activation Condition so that time window considered is from 8:30-9:30. However consulting the list of preferences may highlight PB has a preference for waking up earlier and also there is a preference from a medical point of view for PB not to wake up too late so that it keeps a healthier and more active lifestyle. This discovery actually highlights and important requirement which is not explicit and could be considered for revision and addition to the existing ones. In the balance of services delivered by the system, the SAS function will improve the service value as, whilst lowering on the financial side because of higher consumption of energy, it will increase the service satisfaction on PB for satisfying the preference for an earlier start and the satisfaction of other stakeholders (family and doctor) by keeping PB more healthily active.

So far we have looked at how contexts fit in the 'bigger picture' of an IE system and how contexts relate to other concepts (requirements and user preferences). Let us look next at how contexts relate to each other.

4. Contexts Inter-relationships and Interactions

If we want to create systems which are perceived as 'effectively smart', we need to scale up from the typical knowing the location and time of the day, and more sophisticated combinations of contexts should be considered. In [4] a number of new possibilities were highlighted as different perspectives developers can consider when thinking about contexts in their system, these included more explicit considerations on how contexts may inter-relate to and interact with each other.

One perspective was looking at them from a hierarchical/organizational relationship:

- "C_1 is equal to C_2 on aspect A_i", represented by $(=)$ (C_1,C_2,A_i) when aspect A_i in both C_1 and C_2 have the same content;
- "C_1 overlaps with C_2 on aspect A_i", represented by $(><)$ (C_1,C_2,A_i) when aspect A_i in C_1 and C_2 have some common content;
- "C_1 subsumes C_2 on aspect A_i", represented by $(>)$ (C_1,C_2,A_i) when aspect A_i in C_1 contains all elements of the same Aspect in C_2 and more;

These relationships are clearly of relevance when modifying the definition of a context (say when an extra individual should be notified on emergency contexts) as explicitly knowing there is a relationship between two contexts will remind developers to revise the ones related to that one being modified or deleted. Equally when one is added, a relationship analysis can be conducted to understand the role of the new context in the "contextual ecosystem".

Another perspective considered was looking at how directly and strongly contexts influence each other:

- "C_1 *Directly Influence* C_2", $C_1 \triangleright C_2$: if we have C_1=[1, *, *, Effect] and C_2= [1, AC_2, *, Effect] then we can say C_1 directly influences C_2 when the effect of a context C_1 has a direct impact on another context C_2, that is, if we consider C_1(*Effect*) the symbolic representation of the services triggered then C_1(*Effect*) $\models S$ where $S \subseteq AC_2$.
- "C_1 *has a Ripple Effect on* C_2", represented as $C_1 \triangleright\triangleright C_2$: this can be seen as Indirect Influence, the most complex and most interesting case, where events occurring within one context, C_1, affect another context, C_2, but in a less obvious way. $C_1 \triangleright\triangleright C_2$ can be characterized as follows: (1) C_1 has to start at least no later than C_2 finished, and (2) there are C_1-related events which occur during the span of C_1 and affect the value of properties in AC_2

There is an advantage in the system if developers can somehow understand when there are context activations which tend to lead to other contexts (say the chain 'getting up', 'washing face', 'preparing breakfast', 'leaving home for work') because modifications in one likely will influence the others.

And a third perspective was looking into their interaction modalities:

- *Cooperative*: C_1 (+) C_2, both contexts can be combined to realize another one 3, that is given any $p_1 \in AC_1, p_2 \in AC_2, p_3 \in AC_3$ and $\{p_1, p_2\} \models p_3$, for example 'getting up', 'washing face', 'preparing breakfast', and 'leaving home for work' all together can form part of the higher level context 'weekday morning routine',
- *Competitive*: C_1 (-) C_2 means when either is detected the other one is not, they "turn off" each other by disabling some conditions in the context description, $p_1 \in AC_1$ and $p_2 \in AC_2$ and $\{\ldots, p_1,\ldots\} \not\models p_2$, for example context working will be associated with certain requirements on good lighting and non-disruptive levels of sound supporting concentration,
- *Incompatible*: C_1 (x) C_2 means they cannot coexist simultaneously at any time $p \in AC_1$ and $\neg p \in AC_1$, for example the context of a person being at home or not being at home.

Again, at the time of modifying contexts in the system developers can benefit from knowing which contexts collaborate, compete or are incompatible with each other, so that changes in ones may require revision of the others. In this category of inter-relations the most likely important to pay attention to are changes to the "effects" part of contexts.

5. Conclusions

We have argued that to create more useful and interesting systems in our area, a more sophisticated analysis of contexts and context-awareness is needed. We started revisiting concepts and theories which bring more explicitly to the forefront of the development process the alignment of the system behavior with the expectations from the stakeholders. We also highlighted the need to look in more detail the way contexts interact with other main concepts in a system. This is somehow done to some extent, however, there is not much in the literature guiding developers on that. We argued this is an important aspect which cannot be neglected as changes are unavoidable and they will bring the need to carefully consider the effects of those changes in the rest of the system. We also showed that systems have a lot of internal interconnections, and changes made to contexts, or other elements related to those contexts, have ramifications. With this we hope to make a case for future explorations and developments of more rigorous engineering of context-aware systems. This contribution does not pretend to be a solution in itself, more the starting of a, potentially long lasting, discussion on an important but so far neglected aspect of context-aware system engineering.
For now there is an abundance of systems created mostly in an ad-hoc manner. Our area needs to dig deeper into the concepts, methods and tools related to Context and Context-aware system engineering, so that as an industry we are better equipped to create more mature, useful and reliable systems in the future.

References

[1] Ali R, Yu Y, Chitchyan R, Nhlabatsi A, Giorgini P. Towards a unified framework for contextual variability in requirements. 2009 Third International Workshop on Software Product Management, 31-34.
[2] Ali R, Dalpiaz F, Giorgini P. A goal-based framework for contextual requirements modeling and analysis. 2010 .Requirements Engineering 15 (4), 439-458
[3] Rodrigues A, Rodrigues GN, Knauss A, Ali R, Andrade H. Enhancing context specifications for dependable adaptive systems: A data mining approach. 2019. Information and Software Technology 112, 115-131.
[4] Augusto JC. Contexts and Context-awareness Revisited from an Intelligent Environments Perspective. To appear in Applied Artificial Intelligence, Taylor and Francis. 2021. https://eprints.mdx.ac.uk/34176/
[5] Augusto JC. Reflections on Ambient Intelligence Systems Handling of User Preferences and Needs. In Proceedings The 10th International Conference on Intelligent Environments (IE'14), Shanghai. IEEE Press. 2014. pp. 369-371. https://eprints.mdx.ac.uk/14687/
[6] Evans C, Brodie L, Augusto JC. Requirements Engineering for Intelligent Environments. In Proceedings The 10th International Conference on Intelligent Environments (IE'14), Shanghai, IEEE Press. 2014. pp. 154-161.
[7] Augusto JC, Quinde MJ, Oguego CL, Gimenez Manuel JG. Context-aware Systems Architecture (CaSA). Cybernetics and Systems, Taylor and Francis. 2020. https://eprints.mdx.ac.uk/31198/
[8] Dey AK, Abowd G. Towards a better understanding of context and context-awareness. CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness.
[9] Dey AK. Understanding and using context. Personal and ubiquitous computing 5 (1), 4-7. 2001
[10] Augusto JC, Callaghan V, Kameas A, Cook D, Satoh I. Intelligent Environments: a manifesto. Human-centric Computing and Information Sciences, 3:12, 2013. Springer. DOI: 10.1186/2192-1962-3-12 URL: http://www.hcis-journal.com/content/3/1/12
[11] Augusto JC, Aztiria A, Kramer D, Alegre U. A survey on the Evolution of the Notion of Context-Awareness. Applied Artificial Intelligence, Volume 31 Issue 7-8. Taylor and Francis. 2017. https://eprints.mdx.ac.uk/23310/
[12] Augusto JC and Muñoz A. Managing Preference Profiles in Multi-user Intelligent Environments. Proceedings of Citizen-Centric Smart Cities Services Workshop. 20-21 July 2020, Madrid, Spain.