Noname manuscript No.
(will be inserted by the editor)

Perspectives on engineering more usable context-aware systems

Unai Alegre-Ibarra · Juan Carlos Augusto · Carl Evans

Received: date / Accepted: date

Abstract The expectations of the abilities of context-aware systems (C-AS) often differ from reality. It becomes difficult to program contextual services that react adequately to the circumstantial needs of users as developers need to know, beforehand: the set of contextual states that may exist, what information could accurately determine a contextual state within that set, and what appropriate action should be taken in that particular state. Although there exist many frameworks and tools which support the design and implementation of C-AS, there is less conceptual help for developers to inform them of what contextual situations and services are appropriate (or feasible) to be implemented. This report reviews the state-of-the-art conceptualisation of context, which is more focused on the representational interpretation of the concept, to introduce a perspective that also acknowledges its interactional interpretation. A combination of revised and new definitions is introduced, which give key insights for the development of more useful C-AS. By acknowledging situations as a dynamic phenomenon that arises from action (interaction), and needs to be understood by the developers, it facilitates the analysis of these subjective interpretations into programming constructs (representation). The conceptualisation is also complemented with a set of guidelines for developers, an illustration of their usage, and a further discussion on the future directions for the engineering of more usable C-AS. The introduced conceptualisation is targeted towards the creation of an open-source tool supported framework for the engineering of C-AS.

Unai Alegre-Ibarra* · Juan Carlos Augusto · Carl Evans
E-mail: {U.Alegre,J.Augusto,C.Evans}@mdx.ac.uk
Research Group on Development of Intelligent Environments,
Middlesex University, The Burroughs London NW4 4BT, United Kingdom
*Tel.: +44(0)7871513023

1 Introduction

Approaching the first quarter of the XXI century, the information age has significantly materialised and the general public has access to information technologies which are no longer constrained to a desktop. Computation can occur in all sorts of devices that span from smartphones, tablets or laptops to everyday objects with embedded electronics, including cars and watches. There is a lot of information which is easily available for developers, such as that derived from social networks, or the sensors which are embedded in the many devices that the user already has, or normally interacts with. This scenario is ideal for context-aware computing, a field that has become popular in recent decades as it promises to significantly enhance the interaction between humans and computers. The idea is to use the available implicit information from the *situation* to provide services and information in a more natural way for stakeholders. Thus, the attention and effort required by the system is minimised, as it does not demand users to explicitly input all the information it needs to trigger or personalise a service. The interaction with computers would then be more natural, as it would resemble the way in which humans interact and communicate with each other.

The work presented in (Alegre-Ibarra et al. 2016) surveys existing frameworks, methodologies, and tools for the development of context-aware systems (C-AS), providing evidence that supports the need for a more unified approach to the creation of these types of systems. A key goal of this study, and the work reported here, is to improve the state-of-the-art with regard to techniques and methods to help establish the foundations of a uniform engineering process that covers the entire life-cycle of a C-AS. As an early step towards this goal, this report provides the conceptual layer that will enable the future creation of a more holistic approach, which is strongly related to the requirements elicitation stage. As acknowledged in (Alegre-Ibarra et al. 2016; Greenberg 2001), it becomes difficult for developers to program contextual services that react adequately to the circumstantial needs of the users, as they need to know beforehand: the set of contextual states that may exist, what information could accurately determine a contextual state within that set, and what appropriate action should be taken in that particular state (Greenberg 2001). Although many frameworks and tools which help support the design and implementation of C-AS exist, there is less conceptual help for developers to inform them with regard to what contextual situations are appropriate for them (Alegre-Ibarra et al. 2016; Greenberg 2001).

This report introduces a set of revised and new definitions which give key insights for the development of more useful C-AS. By acknowledging situations as a dynamic phenomenon that arises from action (interaction), and needs to be understood by the developers, it facilitates the analysis of these subjective interpretations into programming constructs (representation). The separation of concepts into *context* and *situation* allows distinction between the information that will be used to detect the proposed situation and trigger adequate services for the users, from the subjective phenomena that is part of the ex-

perience of the end-users, as understood by the developers of the system. It also enables, from early stages, an analysis of the implementation feasibility of the associated services of a situation, as well as the ability of the system to detect that particular situation. As context-awareness is essential for different related fields[1] of study that typically require a more humanised interaction with the users, the new, alternative definition has a user-centred perspective, which takes into account the needs, preferences and limitations of the end-user stakeholders from its conceptualisation.

The remainder of this report is structured as follows. Section 2 reviews previous works related to capturing the essence of context and its related concepts. Section 3 analyses the relevant philosophical challenges behind the creation of C-AS, identifying its current limitations with respect to the creation of usable systems. Section 4 introduces a set of revised and new definitions which is focused on providing the tools for engineering more usable C-AS by considering the existing limitations with regard to the engineering of context-awareness. Section 5 provides a set guidelines for developers, based on the insights gained during the requirements elicitation stage of the POSEIDON[2] EU funded project (Augusto et al. 2013), and illustrates the usage of the proposed definition. Finally, Section 6 presents the conclusions of this work.

2 Previous work

2.1 Context

Etymologically, "context" is an evolution of the Latin word "contextus", composed by the prefix "con" (together) and the root word "texere" (weave). In a more general sense, the meaning of this concept is used to broadly define the set of circumstances that frame an event or an object (Bazire and Brézillon 2005). Although the term can be easily understood, it is difficult to elucidate, leading to many attempts of defining this term throughout history. In regard to context-aware computing, the early definitions of context started in the form of synonymous (Brown et al. 1997), or example enumerations (Schilit and Theimer 1994; Brown 1995; Ryan et al. 1999; ISO 1999). Other conceptualisations encompass context as a broad representational concept (Dey 2001; Yau et al. 2003; Bazire and Brézillon 2005; Roto et al. 2006). The advantage of these types of definitions is that they give space for many relevant contexts to be captured. On the other hand, the disadvantage is that it becomes difficult to materialise broad definitions into adequate design principles (Greenberg 2001). The most acknowledged definition of context was introduced by Dey and Abowd (2001), which considered it as:

[1] Particularly referring to Pervasive and Ubiquitous Computing, Intelligent Environments, Ambient Intelligence, and Ambient Assisted Living.

[2] POSEIDON stands for PersOnalised Smart Environments to increase Inclusion of people with DOwn's syNdrome.

Any information that can be used to characterise the situation of an entity, where the entity is a person, place, or object that is considered relevant to the interaction between a user and its application, including the user and the application themselves.

However, their definition did not bring an absolute consensus about what context means. Some authors tried to extend the definition of context by providing more operational definitions and include other dimensions of context (Schmidt 2003; Zimmermann et al. 2007). Other researchers reported some drawbacks of this definition, such as being too broad, or fencing in the context of the system to an interaction with the user (Makris et al. 2013; Greenberg 2001). Other approaches to the conceptualisation of context have aimed to reduce the inherent broad sense of the definition by being more ad-hoc, such as for example, in (Bauer and Spiekermann 2011) or (Lamsfus et al. 2015). In the area of Intelligent Environments, Ambient Intelligence and Ambient assisted living, there is a considerable orientation towards a more user-centred perspective conceptualisation of the term (Shogren et al. 2014; Bauer et al. 2014; Bauer and Dey 2016).

2.1.1 Context definition through categorisation

In addition to those attempts at conceptualising context by giving a definition, another typical approach is that of dividing context into categories. Perera et al. (2014) review the different context categories, proposing a categorisation scheme that is based on (Dey and Abowd 1999). They divide the context conceptualisation into Operational and Conceptual. On the operational view, context can be either primary or secondary. The primary context is any information received without using existing context. Secondary context is any information which can be computed using existing context. On the conceptual side, they acknowledge the main (activity, time, identity and location) categories, from which other categories stem. Perera et al. acknowledge that from the analysed context categorisation schemes no single category can accommodate all the demands in the Internet of Things paradigm. Bauer and Novotny (2017) introduce a broader analysis on the different context categorisation schemes. They classify the context categories into three main groups: social, physical and technology context. The first group is divided into social environment, user and activity. The second group is divided into information technology and virtual environment. The third group is divided into physical deployment environment, location, movement and time. Finally, they also acknowledge an additional fourth group for domain-specific context.

2.1.2 Situation

Another term which can be found in the literature as having a strong link to context is that of *situation*. Dey and Abowd (2001) introduce the notion of a situation as the collection of particular states in which an entity exists.

In approaches which are more related to artificial intelligence, the notion of situation is more related to the dynamic nature of context, by relating it to actions (McCarthy and Hayes 1969), and considering the history of actions or states (Reiter 1997; Yau et al. 2003). Some other definitions of situation also acknowledge this changing nature of context by recognising that the activities happen in a specific time period (Anagnostopoulos and Hadjiefthymiades 2009), or by acknowledging the concept as some developing state which is characterised by its context. Other authors also consider the situation as the meaning given to sensor data (Ye et al. 2012).

2.2 Context-awareness

Schilit (1994) described a system as context-aware when it is able to "*adapt according to the location of use, the collection of nearby people, host and accessible devices, as well as such things over time*". Later on, the most acknowledged definition of context-awareness was introduced by Dey (2001), who considered a system as context-aware if *"it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"*. But there is no consensus on the definition of context-awareness. Although, generally, the term is used in the literature for describing any type of system that is able to use context, the systems that can be context-aware span many different fields, which understand the notion as the reflection of their own concerns (Alegre-Ibarra et al. 2016).

2.3 Features

Schilit et al. (1994) first identified different classes of context-aware applications. Later, Pascoe (1998) aimed to identify the core features of context-awareness. Dey and Abowd (1999) presented a categorisation for features of context-aware applications, based on the classification of Schilit and Pascoe, namely: 1) Presentation of information and services to the user. 2) Automatic execution of a service. 3) Tagging of context information for later retrieval. With regard to the first feature, the system decides which information and services are presented to the user, based on context. Nearby located objects might be emphasised or, for instance, a printer command might print to the nearest printer. The second feature refers to the automatic execution of a service. Finally, Dey and Abowd introduce the concept of "contextual augmentation", which extends the abilities of sensing, reacting and interacting with the environment by using additional information. This is achieved by associating digital data with a particular context. For example, a tour guide can augment reality by presenting information about the attractions that surround, or are approaching, the tour party (Pascoe 1998).

2.4 Interaction modalities

Barkhuus and Dey (2003) classified the possible interactions into three main modalities. The first is *personalisation*, in which the users are able to set their preferences, likes, and expectations of the system manually. The second interaction modality is *passive* context-awareness, whereby the system is constantly monitoring the environment and offers choices to the users in order to take actions. The third modality is *active* context-awareness, whereby the system is continuously monitoring the environment and acting autonomously.

3 Towards a better understanding of the challenges with regard to the conceptualisation of context

In order to better understand how to engineer more usable C-AS, there is a need for a broader understanding of the influencers and ideas that can serve as a source for inspiration for exploration and innovation that refocuses upon the first-person human experience of ubiquitous computing and C-AS (Takayama 2017). Dourish (2004) acknowledged that the drive to represent context is inspired by, and in some cases the direct response to, sociological investigations. Nevertheless, the philosophical tradition behind those investigations (phenomenology[3]) derives from a different tradition than that of computer science (positivism[4]). In the phenomenological perspective, context is understood as a continually evolving and highly situation-dependent construct (Greenberg 2001). Therefore, context is an issue that has a strong link with the concept of interaction, where:

1. Contextuality is a relational property that holds between objects or activities. It is not a matter of something being, or not being, context; rather it may or may not be contextually relevant to some particular activity.
2. The scope of contextual features is defined dynamically. Rather than being something that can be delineated and defined in advance.
3. Context is particular to each occasion of activity or action. Context is an occasioned property, relevant to particular: settings, instances of action and parties to that action.
4. Context arises from activity, being actively produced, maintained and enacted.

However, the representational nature of computing systems demands a different approach to the concept of context. After analysing the conceptual work of several definitions, Dourish extracted four assumptions that seem to underlie the notion of context as it operates in the view of computer science, where it is treated as a representational rather than an interactional problem. The assumptions are:

[3] A philosophical tradition related to the study of *phenomena*, or things, as they appear in a first-person experience, or consciousness.

[4] A philosophical system that recognises only that which can be scientifically verified.

1. Context is a form of information. Something that can be known, encoded and represented in the same way as other information is modelled in software systems.
2. Context is delineable. For some set of applications, one can define what counts as the context of activities that the application supports, and do so in advance.
3. Context is stable. Although the precise elements of a context representation may vary from application to application, they do not vary from instance to instance of an activity or an event. The determination of any contextual element can be made once and for all.

Dourish highlights an attempt to derive positivist responses from phenomenological arguments in regard to the development of C-AS. Computer science requires the creation of simplified models which are abstracted from the detail of particular occasions, so that developers can program them into computers. Even in computation structures that aspire to resemble the human brain, such as neural networks, the models need to be programmed and trained for obtaining the desired output. When the existing conceptual tools are obtained with the arguments of an antagonistic philosophical tradition to that of sociological investigations, turning social observation into technical design appears to be problematic. Based on Greenberg's (2001) reflections, the dual nature of context has the following implications[5]:

I_1 There is a need to conceptually support developers in: A) Enumerating the set of contextual states that may exist; B) Knowing what information could accurately determine a contextual state within that set; C) Stating what appropriate action should be taken in that particular state. Developers require a better understanding of the situations which are relevant to provide services according to the needs, limitations and preferences of the users. Even if frameworks and toolkits provide elegant ways to design and implement context-aware applications, they fall into a design trap if they do not provide any support for informing the developers of what contextual situations are appropriate to the system.

I_2 Although having adequate support, it is very difficult or even impossible to foresee all the situations in order to program them. Also, some situations that might seem similar a priori, can greatly differ from the actual instantiation of the situation.

- $I_{2.1}$ C-AS have a high chance of taking actions that might not be the most appropriate in certain situations. While the engineering of these systems matures, there is a need to mitigate the impact of context misinterpretation.
- $I_{2.2}$ There is a need to direct the research efforts towards the discovery and analysis techniques of the different situations in which the system can offer services, as they are key to the development of a C-AS.

[5] Note that the listed implications will be referenced as [I_1], [I_2.1], and [I_2.2] along the rest of the report. [I_1] and [I_2] appear in (Greenberg 2001), while [I_2.1], and [I_2.2] are reflections of the authors of this paper.

4 A usability oriented conceptualisation of context and context-awareness

4.1 Interacting with context-aware systems

While the challenges introduced in Section 3 remain unresolved, it becomes necessary to consider alternative ways of interacting with C-AS that go beyond computers being exclusively autonomous. As further explained in the Section 3, creating exclusively autonomous C-AS could make them, in all but simple cases, prone to take inappropriate actions [$I_{2.1}$]. The authors of this report introduced an extended approach to that of Barkhuus and Dey's (2003) in (Alegre-Ibarra et al. 2016), where two foundational dimensions were introduced: I) *Execution*: Referring to the actions or behaviours of the system when a specific situation arises; II) *Configuration*: Relating to the adjustment of actions that a system will exhibit and which takes place following implementation. Both execution and configuration dimensions are mutually exclusive, but both can be executed in two different modalities: A) *Active*, where the system changes its content autonomously; B) *Passive*, where the user has explicit involvement in the actions taken by the system. Therefore, each context-aware feature can be executed in four different interaction modalities:

1. *Active Execution*: The systems have autonomy to execute services, and self-adapt depending on the context. For example, the screen of a smartphone can switch from landscape to portrait automatically, when reaching certain accelerometer values.
2. *Passive execution*: The users are involved in the action-taking process of the system, where they specify if and how the application should change in a specific situation. The system can present services for that specific situation or ask permission from the user to take an action.
3. *Active Configuration*: The user is not directly involved in the evolution of the system after it is implemented. The system is able to discover and learn the user preferences, which are used (autonomously or through non-user human intervention) in order to maintain its rules. Data science techniques can also be considered to enhance this modality.
4. *Passive Configuration*: The user is involved in the manual personalisation of preferences, likes, and expectations of the system after its implementation. Overall programming complexity is reduced by introducing abstractions that enable users to act like software engineers to directly modify preferences or rules, in order to obtain the desired behaviour.

Taking into account the different interaction modalities, developers can mitigate the negative effects of not having a completely mature technology, in each particular context-aware feature. The determination of the most suitable interaction modality will depend on the particular situation and will have to be decided separately by the developers, taking into account their respective advantages and disadvantages (Alegre-Ibarra et al. 2016).

4.2 Features of a context-aware system

The authors of this work also introduced an enhancement of Dey and Abowd's (2001) context-aware feature classification in (Alegre-Ibarra et al. 2016), which accommodated the interaction modalities introduced above. Also, this conceptualisation of features, extended the information presentation to any system stakeholder, rather than limiting it to the users. The features are: 1) Presentation of information to the stakeholders; 2) Active or passive execution of a service; 3) Active or passive configuration of a service; 4) Tagging context to information.

4.3 Concepts

Sub-sections 4.1 and 4.2 introduce some revised work which was designed taking into account the implications explained in Section 3. This sub-section introduces a novel conceptualisation of context and context-awareness which is targeted at rethinking the way in which context-aware systems are engineered. A key concept introduced by this conceptualisation is that of *situation of interest*. This abstraction facilitates the separation of concerns. On one hand, it enables to isolate those concepts related to the detection of the situation of interest, as shown in Figure 1. On the other hand, it enables to treat separately those concepts related to the provision of context-aware features, as shown in Figure 2.

4.3.1 Context-awareness

The ability of a system to use context for exhibiting context-aware features which are useful to the stakeholders because they directly relate to their preferences and needs.

This updated definition is similar to that presented by Dey and Abowd (2001), but introduces a more user-centred perspective, connecting the definitions of context to the above-mentioned context-aware features and its usability. Therefore, context becomes that which makes the system run better for the stakeholders of the system. Also, it is important to mention the introduction of stakeholders as a concept with a broader scope than users. While *users* is a word more focused on those who have the most direct experience with the final product, *stakeholders* not only encompass them but also other people who might have an interest or a concern with the project. This ranges from companies with commercial interests, to governments that have interests in its implications. It is practical not only for commercial reasons but for other applications such as health-care, where the approval of medical staff is essential to certify the safety of end-users. Also, the enhanced definition of context-awareness above is linked to the provision of its features, as defined in Section 3, which relate to the different interaction modalities which aim to reduce the impact of misinterpreting the context [$I_{2.1}$].

4.3.2 Context

The information which is relevant for a computing system to characterise situations of interest.

The aim of this conceptualisation is to introduce a perspective that acknowledges the duality of context, that which brings closer the phenomenological (dynamic) and positivist approaches (static). This context definition acknowledges the positivist perspective, demanded by computerised systems, where context is necessarily a form of information. More precisely, context is considered as the sum of all the symbolic representations required by the computer to figure out when different situations of interest are happening in the real world. Notice that this definition of context depends on the concept of *situation of interest*, which, as further explained in the next subsection, is recognised as an observer-dependent and ontologically subjective phenomena. Although, in comparison, whilst the definition is similarly broad to that of Dey and Abowd's (2001), there are two main differences. The first is that the context information exists without requiring an interaction between the user and an application (Makris et al. 2013). Rather, it requires the existence of some added value for the stakeholders, provided in the form of a context-aware feature. The second difference, as further explained in the next subsections, is that it facilitates the application of development principles to identify the correct context $[I_1]$.

4.3.3 Situation of interest

The circumstance in which developers understand that the system can potentially exhibit a context-aware feature which is relevant for the intention, preferences, and needs of its stakeholders at that particular moment.

The power of the presented context definition resides in the role that the *situation of interest* (SOI) takes in the development of C-AS. Since the SOI is understood as an observer-dependent phenomenon, it is targeted to represent the interpretation that developers give to it. This implies that developers inherently need to engage in an understanding process for developing a C-AS. Particularly, they first have to understand how users give meaning to the actions they take in a SOI (semantics), and then find the best manner in which the computer can realise that situation (symbols) and to provide useful services that can help them accomplish their actions. Therefore, the SOI acts as a nexus between two key conceptual components in the development of C-AS, facilitating the application of the principles for getting the correct context $[I_1]$.

The first conceptual component that is related to the SOI is the provision of useful context-aware features, which are directly related to the needs and preferences of the stakeholders, and are relevant to their intention in that particular situation. The second conceptual component related to the SOI is the representation of the developers' plan to make the system realise that the particular SOI is happening. For the second component, the *context-attribute*

concept is used, as further explained in the next subsection. It needs to be mentioned that developers can consider more than one SOI detection plan, and evaluate which one is more suitable to be implemented, taking into account the particular restrictions of the project in which it is being developed. It could also happen that developers deem they do not have enough resources to make the system identify a particular SOI under their current project scope. Making such realisations and decisions at an early stage is fundamental to the creation of a successful C-AS. This conceptual tool enables the stakeholders to have more accurate expectations on the behaviours that the system will exhibit. It is also important to note that this relation between SOIs, context-aware features, and SOI detection plans, facilitates the analysis of C-AS not only during the requirements elicitation stage of the system, but also during maintenance stages, after the system has already been implemented and deployed. Additionally, since the definition of context-aware features considers all the interaction modalities, these are intrinsically included in the conceptualisation of what context and SOI are. Consequently, its own conceptualisation helps developers to foresee and reduce the potential misbehaviours of the system.

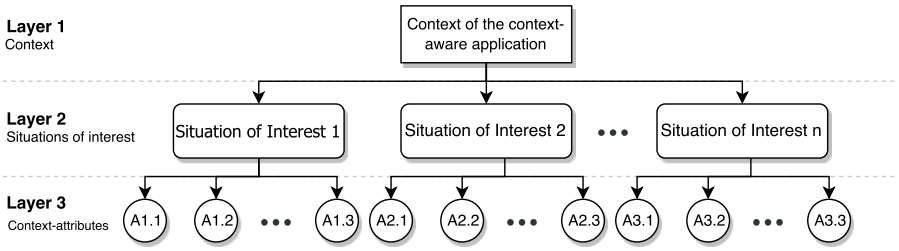

Fig. 1 Decomposition of the different concepts related to context.

4.3.4 Context-attribute

An observable property of a situation of interest which can be realistically attained from a sensor, application or stakeholder.

This definition enhances that of the context model introduced by Henricksen et al. (2003), and that of context-attribute provided by Ruiz-Lopez (2014; 2013). Although, typically, a sensor refers to a hardware sensor, in this case it has a broader sense, which includes physical, virtual and logical sensors (Perera et al. 2014; Indulska and Sutton 2003). Physical sensors refer to those tangible (hardware) sensors that provide data by themselves. Virtual sensors are those which are not tangible (software) and do not necessarily generate information by themselves. They can gather data from different sources and publish it as sensor data (e.g., twitter status, emails, contacts). Logical sensors combine physical and virtual sensors to provide more meaningful information. In this definition, the context-attribute is used by the application to characterise a SOI, and it is also an element of the context model, which describes

the whole context of the system. As acknowledged in (Ruiz-López 2014), a context-attribute can have a context value, which represents the particular value that a context-attribute can take, and can be a punctual value, an interval or a set. The context value is in a context-value domain, which indicates all the possible values that the context-value can take.

4.3.5 Personalisation Management

One important feature of the introduced conceptualisation is the end-user stakeholder centred perspective. For this reason, it is also important to introduce a way in which preferences can be handled using this conceptualisation. Section 4.1 has introduced an interaction modality that enables the configuration of the context-aware application, either in an active or a passive way. Particularly, the following subsection focuses on the modelling of subjective knowledge relative to the preferences of the end-user stakeholders, by introducing concepts to abstract the specific values of user preferences. This facilitates the treatment of generic knowledge about preferences, letting the users, after the system is implemented, define their own preferences as they use the application. For the passive configuration modality, the final values of preferences are meant to be determined by their own users. For the active configuration modality, instead, the final values of the preferences are meant to be determined by preference learning algorithms. The abstraction of particular preference values helps to handle subjective knowledge during the requirements, design and implementation stages. The preference configuration of the end-user stakeholders can be categorised in one of two ways, according to the introduced context conceptualisation, as follows:

A - Context-preference

A type of context-attribute whose particular values are to be personalised by a stakeholder or an agent after the system implementation.

The first configuration type is related to the detection of a situation of interest, via personalisation of certain context-attributes. For example, let the Reader imagine a scenario where it is required to know the temperature, and where a certain threshold temperature will detect a situation of interest where the associated context-aware feature is to automatically turn on the heating. In this hypothetical scenario there will be two context-attributes for detecting the situation of interest: temperature and temperature threshold. It is important to highlight that under the current context definition, both will be considered as being context. Particularly, the temperature threshold will be considered as a context-preference, which can be configured after the system is implemented.

B - Feature preference

A non-contextual software variable which is used to personalise the way in which a context-aware feature is provided.

On the other hand, the second configuration type consists of personalising the provision of context-aware features according to the user preferences. Under the current context conceptualisation, feature preferences are not used for identifying a situation of interest, but to to personalise the way in which a context-aware feature is provided. Therefore, the Reader should note that they can not be considered as context. For instance, let the Reader imagine that in the previous scenario, apart from triggering the heating on, there is another context-aware feature which is to prompt the user informing that the heater has been turned on. Some users with visual difficulties might prefer to receive this notification with a big font size. Other users might prefer to receive this notification with a normal font size. In this case, the feature preference associated to the provision of the context-aware feature will be the visual acuity of the user.

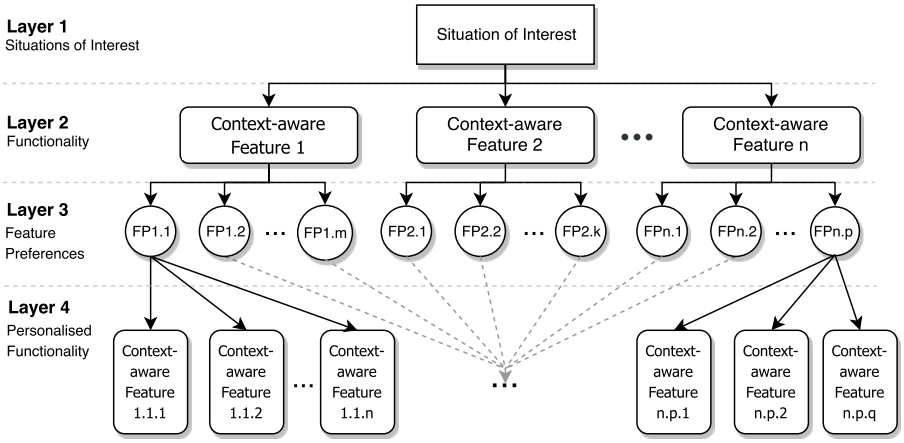

Fig. 2 Decomposition of the different concepts related to the functionality of the system and its personalisation. Note that additional layers for feature preferences and personalised functionality could be added, as context-aware features can be decomposed in other context-aware preferences depending on the feature preferences of the stakeholders.

5 General guidelines for context-aware systems engineering

This section introduces a set of guidelines for applying the above conceptualisation of context and context-awareness. These guidelines report insights gained during the process of conducting the requirements elicitation stage over the EU funded POSEIDON project (Augusto et al. 2013).

5.1 Analysing situations of interest

Each identified SOI requires a plan for the implementation of its detection and its associated services, for which a set of guidelines is introduced. These are to be applied for each SOI that the developers are able to identify, and for each stakeholder relevant to the SOI. The guidelines are composed of the following steps:

1. *Identify objectives, needs and preferences:* It consists of understanding the meaning of the action of the end-user stakeholders to identify the objectives of the different stakeholders in the situation. The objectives can help to unearth different needs and preferences of the stakeholders in that particular situation. A previous background analysis on the different stakeholders can facilitate the identification of these needs and preferences (Bryson 2004).
2. *Determine context-aware features:* Determine adequate context-aware features which are relevant to the intention of the users and help them achieve their goals according to their preferences and needs.
3. *Determine interaction modality:* The most appropriate interaction modality for the proposed service is determined, taking into account the feasibility evaluation conducted for the situation operationalisation.
4. *Propose situation detection plans:* Situation of interest detection plans are proposed. These consist of identifying the different context-attributes that can be used to detect a particular situation of interest. Note that there could be more than one plan for implementing the same situation of interest. This step can have as much detail as developers wish to consider, and it can be an informal definition or it can include the assignation of values or value domains to context attributes, or even the definition of particular rules or ontologies that will be used for inferring higher level context-attributes from lower-level context-attributes, or to trigger the associated services.
5. *Evaluate:* Developers evaluate different aspects which can help them determine if the situation of interest under analysis will be implemented or not. Additionally, the situation of interest is given a priority, which facilitates the selection process of situations of interest. These aspects are:
   (a) Situation of interest detection plan feasibility: Each proposed situation detection plan is analysed in more depth. For this analysis, two main features are taken into account. The first feature focuses on determining how likely it is for the current plan to misunderstand the occurrence of a particular situation of interest. For this, developers can check the accuracy of each of the proposed context-attributes. The second feature has to do with the impact, in terms of objectives of the system, that a failure in detecting a particular situation of interest can cause. A cost-benefit analysis using these features can help developers to determine the feasibility of implementing the detection plan under analysis. This can help to avoid the implementation of situation detection plans that have a high failure likelihood and a considerable situation detection failure impact.

(b) Evaluation of context-aware feature implementation feasibility: There are three main aspects that should be taken into account to conduct this cost-benefit analysis. The first aspect is the cost estimation, where developers gauge the cost of implementing that particular context-aware feature. The second aspect is the frequency with which the situation of interest related to the context-aware feature under analysis is expected to occur. Finally, the last aspect is the detection plan feasibility of the situation of interest related to the context aware feature under analysis. This analysis is explained in the previous bullet-point. This analysis can help to avoid implementing features that have a high cost and are not going to occur frequently, taking also into account the detection plan feasibility.

(c) Ethical concerns: Engineers evaluate if the implementation of the detection of the situation of interest or its associated context-aware features has a conflict with other stakeholders, or if it implies ethical or privacy concerns which are against those of the development team values. An ethical framework is recommended for this purpose (Jones et al. 2015). It is suggested to carefully evaluate and discuss with the end-user stakeholders the different ethical concerns that might arise, until there is an agreement between all parties. The discussions can be complemented by questionnaires and/or interviews.

(d) Validation: The situation of interest should pass through a selection process that helps to determine if the situation of interest is important. For this, a priority is assigned to the situation of interest. Engineers check if the situation is within the system scope or budget, and ask themselves if its associated context-aware features will truly help the user or not according to their preferences and needs. The proposed context-aware features can be validated with the end-user stakeholders in order to check whether or not the proposed features and their interaction modality are adequate for them. For this purpose, interviews or questionnaires can be used. During this stage, it is not the intention to run a questionnaire or interview for each of the identified situations of interest, but rather to add the corresponding questions to a general questionnaire or interview.

The proposed guidelines are not just envisaged for complementing the requirements elicitation stages of the system, but can also be applied at later stages of the life-cycle of a C-AS, such as maintenance. This conceptualisation enables the developer to have better control over the context-attributes and services associated to a particular SOI, facilitating the maintenance of systems. The dynamic nature of context demands that developers constantly exercise an understanding of the interaction between the user and the system in different situations. Such conceptual tools empower developers, as they guide them exactly to those elements that they need to alter in an already developed system. Developers can decide to remove or modify a particular SOI, controlling which context attributes and services will be affected. Also, they can have a better control over the system design when adding/removing associated services.

5.2 Example

This subsection will illustrate, with an example, how the proposed conceptualisation and guidelines can be applied during the development of a C-AS. For this purpose, a particular SOI relevant for the example is analysed in depth. To conclude, the result of applying the guidelines to other SOIs is also illustrated in Table 1.

Consider the example of a C-AS that is an outdoors navigation application which is bespoke to a particular disability. More specifically, a navigation system that has a higher order goal to aid the inclusion of people with Down's syndrome in society by fostering their independence through the use of assistive technologies (Kramer et al. 2014). Particularly, the development will focus on a mobile application that uses a real-world representation of maps along with location services to support outdoor journeys that might be walking or by bus. Due to space restrictions, this example will simply focus on bus displacements happening in London, United Kingdom. The application has customisable routes with tailored directions, notifications, reminders, and other services which will be triggered depending on the context.

This example was a specific scenario used in the requirements engineering phase of the POSEIDON project (Augusto et al. 2013). During this stage, several meetings, discussions, interviews, and questionnaires were conducted (POSEIDON 2015). Relevant to this research, was a questionnaire[6] to potential users of the application. Table 1 reflects several SOIs relevant to this example, and the results of their corresponding analysis. The table is divided into three main blocks. On the first block (A), the description for each SOI is included. Also, the mean (A-III) and standard deviation (A-IV) of the responses to the questionnaires regarding the perception of the primary users on the usability of the SOIs is provided. On the second block (B), each proposed context-aware feature is illustrated (B-III), for each of the end-user stakeholders of the system (B-I), as well as the proposed execution modality (B-IV). Additionally, mean (B-V) and standard deviation (B-VI) on the results of the questionnaire regarding the perceived usefulness of the features is included. Also, how much the primary users liked the feature (B-VII). Finally, the last block (C) represents the proposed operationalisation plan in which each context-attribute required is illustrated. It should be noted that these only represent a small example of all the possible SOIs that could arise during the development of such a system. Some other situations include when the primary user gets lost, someone tries to steal his/her phone, or the primary

[6] 130 British families were contacted for completing them, and these were composed of at least one family member with Down's syndrome. The respondent population was divided into people with Down's syndrome (primary users) and carers of people with Down's syndrome (secondary users). A total of 52 responses from potential secondary users and 29 from potential primary users were obtained. Each group had a different format of questionnaire. That prepared for primary users was an "easy-to-read" version, in which they were asked only about the different services. Also, this group was helped by their carers during the process. The questionnaire for the group of secondary users included questions about the situations and the services.

user falls asleep when travelling by bus. The secondary users that participated in the discussion provided information, using a scale from one to five, on the usefulness of particular SOIs (Table 1-A-III/IV) and its associated context-aware features (Table 1-B-V/VI). Primary users were asked, in a binary scale, if they would like to have the context-aware feature proposed (Table 1-B-VII). The interviews also prompted them about new SOIs and the best interaction modalities for the different features.

5.2.1 Situation of interest

The SOI to be analysed through this example is when the primary user is waiting for a bus that, due to unforeseen circumstances, will not arrive on time, or at all. In this particular SOI there are no more buses available within a given time-frame that will take the user to the destination. Another thing that needs to be taken into account for this SOI is whether or not the user is waiting under certain comfort standards. Some people with Down's syndrome might not be able to realise by themselves that a particular bus is taking more time than it should to arrive. So, particularly, if the user is standing outdoors for a long time, under bad whether conditions, it could imply certain risks for the user's health. Developers can also analyse other alternative variations or extension points on the target SOI. There could be other SOIs where more buses for the same line are available after the the one that is missing, which would arrive in a given time-frame, and would take the user to the destination. There could also be buses from another line available for an alternative route to the destination. The SOI analysed in this particular example is as follows:

The primary user is waiting outdoors, under bad weather conditions, for a bus that will not arrive due to unforeseen circumstances, and there are no more buses available in a given time-frame to the user's destination.

5.2.2 Feature proposal

Developers begin to consider meaningful actions that the user can take in that SOI, as well as the needs of the stakeholders. In this case, the requirements analysis puts the potential user into the situation and asks them for possible actions. Ideally, this process should also observe the users in such scenarios, to help get a better understanding of their actions. It is also worth remembering that the process applies to all the stakeholders, and has to be applied to each of the identified SOIs.

Primary Users In this situation, the primary user needs to know that the bus she/he is waiting for will not arrive, and what to do next in order to safely arrive at their destination. The following services are proposed:

- The primary user receives a notification communicating that the bus will not arrive, and a set of instructions to follow.
- The primary user is prompted to call a secondary user, and a list of carers appears in the screen as possible options.

Looking at the interaction modalities of the services, the notification service is chosen to be purely active at execution time. Nevertheless, the instructions received are passively configured by the secondary users, as further explained in the next paragraph. On the other hand, the call to the secondary user will be merely a prompt (passive execution). This second context-aware feature can be passively configured, enabling the users to set a list of preferred secondary users to call.

Secondary Users In the case of secondary users, they need to ensure the comfort and safety of the primary users during the process of finding another alternative and reaching their destination. The proposed service consists of sending a notification to the secondary users, prior to the notification for primary users, letting them know their current situation. Then, a set of options would be displayed, which include:

- Giving a call to the primary user.
- Requesting the location of the primary user. If the location is available:
  - Suggesting alternative places where the primary user can wait, according to her/his current location. After selecting them, the route to these places will be automatically added to the primary user instructions.
  - Automatically ordering a taxi. Once the user has arrived the suggested waiting destination, secondary users can select to order a taxi, by automatically sending the location to the taxi service, that will automatically charge them through their bank account.

Although most of the services, when selecting them, are automated, the secondary user is in control of the critical decisions the system will take (passive execution) while the system provides as much relevant information as possible to ease the decision taking. They can evaluate better than a machine whether or not it is the best option for the primary user to go to the nearby sheltered places in particular SOI instances. It might be that the place is inappropriate or that there is no information about opening/closing times, and it is risky to send the primary user there whilst it is raining. This subsection reflects on the result of the feature proposal process. Note that different developers may derive different services or execution modalities, according to their own criteria. The evaluation of the services may depend on different factors.

5.2.3 Situation Detection Operationalisation Plan

This particular SOI, can be split into: 1) The user waiting outdoors; 2) The user is standing still for a long time; 2) Bad weather conditions; 3) The bus will not arrive in a specified time-frame; 4) No more buses will arrive in a specified time-frame. For 1, the position can be operationalised just by using the signal-to-noise ratio from the phone of the primary user, and deducing if the person is indoors or outdoors. For 2, it can be estimated with the location history of the user device. In the case of 3, the weather conditions can be obtained via http, through the API of openweathermaps[7] and the location of the

[7] https://openweathermap.org/api

user. For 3 and 4, as the users will be located in London, the unified Transport for London (TfL) API[8] can be used. Also, for 3, the user location could be used for estimating how much time has elapsed since the user has been stationary. Note that this is just an example of one possible operationalisation, but developers can consider more than one. This particular operationalisation has the following context attributes: *inout*, *standingStill*, *weather*, *busNotArriving*, and *noMoreBusesArriving*. In the case of *inout*, it will indicate that the user is indoors when the signal noise-to-ratio value[9] is below 26, and outdoors otherwise. The value indicating if it is day or night will be taken from the date of the calendar. For the *busNotArriving* context-attribute, first, the location and time that the user has been standing in the stop will be taken into account. When the time waiting is higher than 10 minutes, the TfL API will be checked to see if the bus is delayed or not. The data for the origin and destination bus stops and current time will be needed for querying the API. The current bus stop can be deduced from the user location, and the time from that of the phone. For *noMoreBusesArriving*, other alternatives to reach the destination can be checked in the TfL API. Note that developers can analyse in greater depth and even start creating the corresponding rules or ontologies.

Finally, the context attributes for *standingstill* and *weather* will be enabled for personalisation as follows. The *standingstill* attribute, can be derived from two context-attributes, the *location* of the user, and a *timer* that counts how much time the user has been in the same position. Additionally, a context-preference will be used, which will let the users decide in how much time the application should consider her or him to be standing still. On the other hand, the *weather* context-attribute can be devided into *precipitation* and *temperature* context attributes. If the weather is cold or there are precipitations, the weather will be considered as bad weather. In order to calculate whether if the temperature is cold or not, a context-preference, *coldTemperature*, will be introduced. With this preference users can introduce to the system at what temperature they feel cold.

5.2.4 Evaluation

Finally, the four main aspects of the operationalisation plan are evaluated to determine if the situation of interest and its associated services should be implemented. For this purpose, the aspects introduced in 5.1.d are analysed. Particularly, the first two aspects and their sub-aspects are evaluated using the following metric: LOW, MEDIUM and HIGH.

1. Situation of interest detection plan feasibility: For estimating the detection plan feasibility, the failure likelihood and the failure impact are evaluated. For the failure likelihood, all the proposed context-aware features are evaluated individually.
   (a) *inout*: With an adequate implementation, it can detect open outdoors, semi-outdoors, light indoors and deep indoors environments with 100%

[8] https://api.tfl.gov.uk

[9] The suggested signal noise-to-ratio value is known to be accurate for this purpose.

A				B							C	
I	II	III	IV	I	II	III	IV	V	VI	VII	I	II
Activity	**Situation of interest**	**Mean**	**STD**	**Stakeholder**	**Situational Needs and Preferences**	**Context-aware Feature**	**Execution**	**Mean (PU)**	**STD (PU)**	**Likes C-A Feature (SU)**	**Operationalisation**	**Context Attributes**
Waiting for bus	PU arrives to the origin stop	3.9/5	1.1	PU	– Know waiting time – Remember bus line	Notify primary user the bus line and the time remaining	A	4.9/5	0.9	28/28	Navigating, the PU location is that of the bus stop	– Device location – Origin bus stop location
	A bus arrives to the bus stop	4.1/5	1.0	PU	– Identify whether is the bus they need to get on or not	Notify PU to get (or not) on the bus that has arrived	A	4.3/5	0.9	23/24	Current time is (approx.) the scheduled for bus arrival	– Current time – Scheduled bus arrival time – Margin time
Going by bus	The bus arrives to the destination stop	4.3/5	0.9	PU	– Remember that they need to get off the bus in this stop	Notification reminding that they need to get off the bus in this stop	A	4.5/5	0.7	22/26	The primary user's device location is in the destination bus stop location	– Device location – Destination bus stop location
	The primary user has to press the stop button	4.3/5	0.9	PU	– Remember that they need to press the stop button	Notification reminding that they need to press the stop button	A	4.5/5	0.7	23/26	The primary user's device location is one stop before the destination stop	– Device location – Location of the previous bus stop to the destination one
Getting off the bus	The primary user fails to get off at the correct stop	4.5/5	0.8	PU	– Realise situation – Know how to correct it	Notify situation and give instructions	A	4.5/5	0.9	17/29	PU speed is close to walking speed and is earlier than arrival time	– Device speed – Walking speed value – Current time – Bus arrival time
						Call secondary user	P	4.6/5	0.7	12/29		
				SU	– Realise situation – Safety and comfort of the PU – Know if the PU can resolve the siuation	Notify situation	A	4.8/5	0.4	16/29	The PU is in the bus route and the movement speed is (approx.) higher than walking speed	– Device movement speed – Walking speed – Device location – Bus route
						Request the location of the primary user	P	4.8/5	0.5	-		

Table 1 Situations of interest (A), context-aware features (B) and context attributes (C) of the illustration example for a navigation application to support people with Down's syndrome integrate in society. (PU = Primary User, SU = Secondary User, A = Active, P = Passive, STD = Standard Deviation)

accuracy (Wang et al. 2016). Therefore, the precision of this context-attribute can be estimated as HIGH.

(b) *standingstill*: This context-attribute directly depends on the accuracy of the location. Although it depends on the chipset of the phone of the user, and the number of satellites available in the zone, it is reasonable to estimate that in London, the accuracy will be around 10 meters. This can be used to detect if the primary user is standing still for a long time, in an area of 10 square meters. For this reason, the context-attribute is considered as having HIGH accuracy.

(c) *weather*: As explained in section 5.2.3, the use of open weather maps has been proposed. This service retrieves raw data from airport weather stations, radar stations, and other official weather stations. Additionally, it also involves weather station owners that can help to increase the weather data accuracy. Although this information could not be perfect, it is estimated that for the purpose of this application the weather accuracy is HIGH.

(d) *busNotArriving* & *noMoreBusesArriving*: For this particular case, it is known that many transport applications like Citymapper (Citymapper 2011) use real time information which is directly retrieved from the official London´s transport system, via the TfL API. Initial qualitative

analysis (as part of the POSEIDON project) has concluded that the accuracy is sufficiently HIGH.

Overall, the estimation for the failure likelihood is LOW, as the accuracy of all the analysed context-attributes has been estimated as HIGH. On the other hand, for the failure impact detection aspect, two cases can occur. The first case is when the system interprets the occurrence of the situation of interest, but this situation is not really occurring in the real world. In this case, primary users can simply ignore any prompts, and continue with what they were doing. It could happen that the secondary users get worried, but a phone call would take them out of doubts. For this reason, the failure impact in this case can be deemed as LOW. The second case is when the situation of interest is occurring, but the system fails to detect it. In this case, it might happen that the user waits for an undetermined time period for a bus that will never arrive without the secondary users knowing about it. This situation can entail some risks, but this risk can be estimated as MEDIUM, as the user life would not necessarily be in danger. Although this situation is undesirable, the failure likelihood has been estimated as LOW. For this reason, the result of the situation of interest detection feasibility is HIGH.

2. Feature Implementation Feasibility: For estimating the feasibility of implementing the proposed context-aware features three main dimensions are analysed: cost, frequency of occurrence and detection plan feasibility. For calculating the cost, each of the proposed context-aware features for the situation of interest are analysed individually against these criteria.
   (a) The primary user receives a notification: As it can be observed in Table 1, the system under development will use notifications in many occasions. Therefore, once a basic system for prompting notifications is created, the cost of implementation of this particular feature will be considered as LOW.
   (b) Give the primary user a choice to call secondary user: This feature is quite similar to the previous feature, in terms of implementation costs. It will be simply a prompt with different options. Since Android is an operating system that is focused on mobile devices, it also provides an easy framework from which programming a phone call is straightforward. Therefore, the implementation cost of this particular feature can also be considered as LOW.
   (c) Secondary users can call the primary users: As mentioned in the previous point, programming this feature in Android is straightforward. Therefore, the cost of implementing this feature can also be estimated as LOW.
   (d) Request location of the primary users: Although retrieving the location of the user can be readily achieved with the Android platform, this information needs to be transmitted to the phone of the secondary users. For this an additional secure connection needs to the POSEIDON server, where devices of both users would need to connect. For this reason, the implementation cost for this feature is estimated as MEDIUM,

as it requires of an additional server to work. The implementation cost is not estimated as HIGH, as the non-contextual requirements contemplate the creation of a server for storing the location of the primary users when required.

(e) Suggest nearby places: In order to retrieve nearby places using the location, the Google Places API can be used. This does not have any additional difficulty further than learning how to use the API, therefore it can be considered as having an estimated LOW cost.

(f) Order a taxi: For ordering a taxi, the Uber API (UberTechnologies 2009) for Android can be used. This does not have any additional difficulty further than learning how to use the API. Therefore, the cost can also be considered as LOW.

It is difficult to predict the exact frequency of occurrence of the situation of interest, as this would require a further analysis in the field, which can demand the observation of the users, and/or having some interviews with them about the matter. For the purpose of this example, a MEDIUM frequency of occurrence for the situation of interest will be assumed. As introduced in the previous point, the detection plan feasibility result has been HIGH. The feature implementation feasibility will be considered HIGH for all the features with a LOW COST, and MEDIUM for the location request.

3. Ethical concerns: For evaluating the ethical aspect of the situation of interest, the eFRIEND (Jones et al. 2015) ethical framework can be used. This enables the analysis of the situation of interest and its associated context-aware features, particularly with regard to the following principles:

   (a) Privacy: It is typically difficult to keep a balance between the privacy concerns and the design of context-aware systems. In different interviews and meetings (Augusto et al. 2017), the primary users expressed their concern for the secondary users knowing their location at all times. For this reason, an in order to maintain a balance between the design with the privacy concerns, an option to allow the primary users control when to share their location with the secondary users. Secondary users can request to obtain the location of the primary users, but it is the primary users who determine whether or not their location is shared.

   (b) User Autonomy: Analysing the proposed features, both primary and secondary users have control over the actions of the system. In the case of primary users, they receive indications and they decide if they want to call the secondary users or resolve the situation by themselves. In the case of secondary users, they can also select from a list of different services available.

   Other ethical aspects such as non-maleficence, user-centred perspective, security, transparency, equality, dignity and inclusiveness, are inherently included in the eFriend framework. In regard to data protection, the information about the users will be stored under the required conditions in the POSEIDON server.

4. Validation: Finally, the situation of interest and their corresponding services were brought up for discussion with the different stakeholders. The

secondary users gave 4.5 out of 5 in usefulness to this situation of interest, with a 0.6 standard variance. The context-aware features proposed for the primary users were both graded by the primary users in usefulness with a 4.5 out of 5 (0.5 standard deviation). Those services proposed for the secondary users were similarly graded with 4.7 out of 5 (0.3 standard deviation). Overall the validation gave positive results for the proposed situation of interest and associated services.

In this particular case, the whole analysis gives a positive result, and the situation is considered as implementable under the proposed situation detection plan. In this particular example, the situation of interest and its corresponding context-aware features are considered for implementation. Note that during the evaluation process, the situation of interest under analysis, or some of the associated context-aware features could be stopped to be considered as implementable for the system under development. In order to calculate the priority, the security of the user will be taken into account. Since its implementation might directly affect the health of the users, the priority of this situation of interest is considered as HIGH.

6 Discussion

The updated definition of context provided in this report encompasses a dual understanding of the concept. It has a strong link with the representational nature of context, acknowledging that, for C-AS development purposes, context is necessarily a piece of information coded in a computer. Albeit, the notion is subject to that of SOI, which is acknowledged as an observer-dependent phenomenon. Notice that computerised systems are not considered as observers. Such duality tries to bring closer an intermediate approach between two disparate philosophical paradigms. Although the SOI is a relational property that holds between objects and user activities, particular to each occasion, it is a task of the developers to exercise an understanding in order to program them into the computer. This comprehension exercise is not only about finding what users do, but rather to unearth how those actions accomplish meaningful events and are, in turn, accepted by others as meaningful (Dourish 2001). Such an understanding can lead to a better comprehension of user needs and preferences, and it eases the task of developers in finding adequate context-aware features to be displayed by the system, as well as its adequate interaction modality. Also, developers can create a context model which is exclusively based on the context that is required for making the context-awareness features happen. No more and no less. Therefore, developers have more guidance on what to do to translate the phenomenon into computer models. Nevertheless, it can occur that SOIs are not delineable into context, that is to say, they cannot be transformed into information which the computer is able to use to detect a situation. The presented conceptual tools and guidelines try to mitigate this impact by enabling developers to realise this and take adequate action at an early development stage. Note that the meaning which developers find arises

from the course of action itself, which keeps occurring after the implementation of the system. This implies that SOIs are defined dynamically, and therefore a C-AS will require continuous maintenance. Context is no longer considered as something static, but rather the approximation of the developers to the detection of the situation phenomena. A tool-supported framework that is able to trace the impact of changing a situation of interest in the context model, and that also speeds up its modelling, implementation and deployment can be key to the development of C-AS.

The approach presented here tries to work towards maximising the usability results of those situations that, to some extent, are possible to predict and be represented as computational models. The premise underlying this conceptualisation of context is that current C-AS can maximise their usability results when:

- Developers adequately identify a situation of interest.
- Developers understand the intention, meaning of the actions, preferences and needs of the users in a particular situation of interest.
- There exists a set of observable properties which can identify a situation of interest with enough accuracy to distinguish it from the maximum number of similar-looking siuations where the context-aware feature to be displayed is not appropriate.
- The properties that identify a situation of interest can realistically be attained from sensors.
- The developers can implement the situation of interest detection while it is still meaningful for the users.

More research is required in the direction of techniques for discovering adequate services for C-AS which can help the creation of more usable C-AS [$I_{2.2}$]. Particular approaches such as phenomenology (Winograd and Flores 1986; Dourish 2001), activity theory (Nardi 1996), ethomethodology (Suchman 1985), or Merleau-Ponty's phenomenology (Svanaes 2001) have been proposed. Other methods and perspectives should be considered too. For example, advanced computer technologies such as those provided by data science, machine learning or cognitive computing, could give developers a better understanding of the meaning of the actions of the users in particular situations. An analysis of the data that is already available in the sensors of the system can also be used for giving feedback to developers, and help them unearth new SOIs or enhance/modify the provision of existing context-aware features. There is a need for a theoretical foundation that allows developers to reach such understanding of user action through information technologies, rather than trying to make the machines do this process for them.

This conceptualisation has also provided conceptual tools to deal with the preferences of the users. Nevertheless, there are some issues that still need to be explored, such as the handling of incomplete or ambiguous preferences, and the addressing of conflicting knowledge. For example, in a smart-house scenario there might be a preference conflict if a user likes music with loud volume and another user prefers silence at the same time. When gathering the requirements

of a system, it can also be useful to have user profiling features (Evans et al. 2014; Sutcliffe et al. 2006) for addressing the different user preferences.

7 Conclusions

Context has been a topic of debate for a long time. This report is specifically concerned with the application of the notion of context with respect to the engineering of more usable C-AS. Particularly, a conceptualisation of context and context-awareness is provided, as a means to support the three principles to identify the correct context [I_1]. The work in (Alegre-Ibarra et al. 2016) concluded that none of the evaluated methodologies were offering enough support for these principles. The authors of this work are currently exploring alternative tools to assist developers with respect to the proposed conceptualisation of context and context-awareness. A key goal of this study is to improve the state-of-the-art with regard to techniques and methods to help establish the foundations of a uniform engineering process that covers the entire life-cycle of a C-AS. The work presented here aims to be used as the foundation of an open-source tool-supported requirements elicitation framework specialised for context-awareness. The fundamental feature of this model-based tool will be the traceability of SOIs with context-aware features and context-attributes. In this way, developers will have a very useful tool to evaluate the changes they can implement into the system. The definition proposed in this report uses the notion of SOI as a key nexus between the context information required and the services to be provided. When developing systems under this definition, the developer will have to deal with many SOIs, which can result in complex inter-relationships. This approach could greatly benefit from the traceability of SOIs and their relations with other requirements or design elements. Tools like the OMG Systems Modelling Language, SysML (OMG 2012), or the Unified Modelling Language, UML (OMG 2015), have the potential for creating personalised profiles and managing many relationships between different elements of requirements and design. Rather than starting a new framework from scratch, the aim is to integrate it with previous methodologies (Evans et al. 2014; Ruiz-López et al. 2013), diagrams (OMG 2012, 2015) and tools (Modeliosoft a). Its relation with (Ruiz-López et al. 2013) and the NFR Framework (Chung et al. 2012) can help the automation of the evaluation of the situation detection plans, and the satisfaction of situational goals with the proposed context-aware features. The tool prototype[10] developed as part of this research is open-source, and it can be downloaded from (Alegre-Ibarra 2016).

Acknowledgements To Tony Clark for initiating the discussion on how to distinguish simple data from context information and Dean Kramer for his useful ideas in the bus transportation examples. To Julian Hallett for his support running the main questionnaires

[10] The folder rcase/rcase/target/ of the rcase project contains a jmdac file, which is a module that can be installed in the Modelio tool v3.7.0 (Modeliosoft b) following the instructions in (Modeliosoft c).

of this report. To the anonymous reviewers of this paper for their insights. The research leading to these results has been partly supported by the POSEIDON project funded by the European Union (FP7/2007-2013) under grant agreement number 610840.

References

Alegre-Ibarra U (2016) Requirements for Context-Aware Systems Engineering (RCASE) Tool. https://github.com/ualegre/rcase. [Online; Last accessed 04-April-2018]

Alegre-Ibarra U, Augusto JC, Clark T (2016) Engineering context-aware systems and applications: A survey. Journal of Systems and Software vol. 117 pp. 55–83

Anagnostopoulos C, Hadjiefthymiades S (2009) Advanced inference in situation-aware computing. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans vol. 39, 5 pp. 1108–1115

Augusto JC, Grimstad T, Wichert R, Schulze E, Braun A, Rødevand GM, Ridley V (2013) Personalized smart environments to increase inclusion of people with down's syndrome. International Joint Conference on Ambient Intelligence. Springer, pp. 223–228

Augusto J, Kramer D, Alegre-Ibarra U, Covaci A, Santokhee A (2017) 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 pp. 1–16

Barkhuus L, Dey A (2003) Is context-aware computing taking control away from the user? Three levels of interactivity examined. UbiComp 2003: Ubiquitous Computing. Springer, pp. 149–156

Bauer C, Dey AK (2016) Considering context in the design of intelligent systems: Current practices and suggestions for improvement. Journal of Systems and Software vol. 112 pp. 26–47

Bauer JS, Newman MW, Kientz JA (2014) What designers talk about when they talk about context. Human–Computer Interaction vol. 29, 5-6 pp. 420–450

Bauer C, Novotny A (2017) A consolidated view of context for intelligent systems. Journal of Ambient Intelligence and Smart Environments vol. 9, 4 pp. 377–393

Bauer C, Spiekermann S (2011) Conceptualizing context for pervasive advertising. Pervasive Advertising, Springer. pp. 159–183

Bazire M, Brézillon P (2005) Understanding context before using it. Modeling and using context, Springer. pp. 29–40

Brown PJ (1995) The stick-e document: a framework for creating context-aware applications. Electronic Publishing-Chichester- vol. 8 pp. 259–272

Brown PJ, Bovey JD, Chen X (1997) Context-aware applications: from the laboratory to the marketplace. IEEE personal communications vol. 4, 5 pp. 58–64

Bryson JM (2004) What to do when stakeholders matter: stakeholder identification and analysis techniques. Public management review vol. 6, 1 pp. 21–53

Chung L, Nixon BA, Yu E, Mylopoulos J (2012) Non-functional requirements in software engineering, vol. 5. Springer Science & Business Media

Citymapper (2011) Citymapper Transport Application, Official Website. https://citymapper.com/. [Online; Last accessed 19-February-2018]

Dey AK (2001) Understanding and Using Context. Personal and Ubiquitous Computing vol. 5 pp. 4–7

Dey AK, Abowd GD (1999) Towards a better understanding of context and context-awareness. In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. Springer-Verlag, pp. 304–307

Dourish P (2001) Seeking a foundation for context-aware computing. Human–Computer Interaction vol. 16, 2-4 pp. 229–241

Dourish P (2004) What we talk about when we talk about context. Personal and ubiquitous computing vol. 8, 1 pp. 19–30

Evans C, Brodie L, Augusto JC (2014) Requirements Engineering for Intelligent Environments. Intelligent Environments (IE), 2014 International Conference on. IEEE, pp. 154–161

Greenberg S (2001) Context as a dynamic construct. Human-Computer Interaction vol. 16, 2 pp. 257–268

Henricksen K (2003) A framework for context-aware pervasive computing applications. Ph.D. thesis, Computer Science, School of Information Technology and Electrical Engineering, University of Queensland

Indulska J, Sutton P (2003) Location management in pervasive systems. Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21. Australian Computer Society, Inc., pp. 143–151

ISO (1999) ISO 13407: Human-centred design processes for interactive systems. Tech. Rep., International Standards Organization

Jones S, Hara S, Augusto J (2015) e-FRIEND: an Ethical Framework for Intelligent Environment Development. Ethics and Information Technology, vol. 17. Springer, vol. 17, pp. 11–25

Kramer D, Augusto JC, Clark T (2014) Context-Awareness to Increase Inclusion of People with DS in Society. Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence. pp. 27–31

Lamsfus C, Wang D, Alzua-Sorzabal A, Xiang Z (2015) Going mobile: Defining context for on-the-go travelers. Journal of Travel Research vol. 54, 6 pp. 691–701

Makris P, Skoutas DN, Skianis C (2013) A survey on context-aware mobile and wireless networking: On networking and computing environments' integration. IEEE communications surveys & tutorials vol. 15, 1 pp. 362–386

McCarthy J, Hayes PJ (1969) Some philosophical problems from the standpoint of artificial intelligence. Readings in artificial intelligence pp. 431–450

Modeliosoft (a) Modelio. https://www.modelio.org/. [Online; Last accessed 04-April-2018]

Modeliosoft (b) Modelio Module Installation Guide. https://www.modelio.org/downloads/download-modelio.html. [Online; Last accessed 04-April-2018]

Modeliosoft (c) Modelio Module Installation Guide. https://www.modelio.org/quick-start-pages/916-modelio/quick-start/24-working-with-modules.html. [Online; Last accessed 04-April-2018]

Nardi BA (1996) Context and consciousness: activity theory and human-computer interaction. Mit Press

OMG (2012) OMG Systems Modeling Language (OMG SysML), Version 1.3. http://www.omg.org/spec/SysML/1.3/

OMG (2015) OMG Universal Modeling Language (UML), Version 2.5. http://www.omg.org/spec/UML/About-UML/

Pascoe J (1998) Adding generic contextual capabilities to wearable computers. Wearable Computers, 1998. Digest of Papers. Second International Symposium on. IEEE, pp. 92–99

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

POSEIDON (2015) Poseidon Web Page. http://www.poseidon-project.org/research-scientists/questionnaires/. [Online; Last accessed 04-April-2018]

Reiter R (1997) The situation calculus ontology. Electronic News Journal on Reasoning about Actions and Changes

Roto V, et al. (2006) Web browsing on mobile phones: Characteristics of user experience. Ph.D. thesis, Helsinki University of Technology

Ruiz-López T (2014) Un enfoque dirigido por modelos para el desarrollo de servicios para sistemas ubicuos basado en propiedades de calidad. Ph.D. thesis, Universidad de Granada

Ruiz-López T, Noguera M, Rodríguez MJ, Garrido JL, Chung L (2013) REUBI: A requirements engineering method for ubiquitous systems. Science of Computer Programming vol. 78, 10 pp. 1895–1911

Ryan N, Pascoe J, Morse D (1999) Enhanced reality fieldwork: the context aware archaeological assistant. Bar International Series vol. 750 pp. 269–274

Schilit B, Adams N, Want R (1994) Context-aware computing applications. Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. IEEE, pp. 85–90

Schilit BN, Theimer MM (1994) Disseminating active map information to mobile hosts. Network, IEEE vol. 8, 5 pp. 22–32

Schmidt A (2003) Ubiquitous computing-computing in context. Ph.D. thesis, Lancaster University

Shogren KA, Luckasson R, Schalock RL (2014) The definition of context and its application in the field of intellectual disability. Journal of Policy and practice in Intellectual Disabilities vol. 11, 2 pp. 109–116

Suchman LA (1985) Plans and situated actions: the problem of human-machine communication. Xerox Corporation, Palo Alto Research Center

Sutcliffe A, Fickas S, Sohlberg MM (2006) PC-RE: a method for personal and contextual requirements engineering with some experience. Requirements Engineering vol. 11, 3 pp. 157–173

Svanaes D (2001) Context-aware technology: a phenomenological perspective. Human–Computer Interaction vol. 16, 2-4 pp. 379–400

Takayama L (2017) The motivations of ubiquitous computing: revisiting the ideas behind and beyond the prototypes. Personal and Ubiquitous Computing vol. 21, 3 pp. 557–569

UberTechnologies (2009) Uber API. https://developer.uber.com/. [Online; Last accessed 19-February-2018]

Wang W, Chang Q, Li Q, Shi Z, Chen W (2016) Indoor-Outdoor Detection Using a Smart Phone Sensor. Sensors vol. 16, 10 p. 1563

Winograd T, Flores F (1986) Understanding computers and cognition: A new foundation for design. Norword, NJ: Ablex Publishing Corporation

Yau SS, Liu H, Huang D, Yao Y (2003) Situation-aware personalized information retrieval for mobile internet. Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International. IEEE, pp. 639–644

Ye J, Dobson S, McKeever S (2012) Situation identification techniques in pervasive computing: A review. Pervasive and mobile computing vol. 8, 1 pp. 36–66

Zimmermann A, Lorenz A, Oppermann R (2007) An operational definition of context. Modeling and using context, Springer. pp. 558–571