# Supporting context on software applications: a survey on context engineering

## Le support applicatif à la notion de contexte : revue de la littérature en ingénierie de contexte

Manuele Kirsch Pinheiro[1], Carine Souveyet[2]

[1] Centre de Recherche en Informatique, Université Paris 1 Panthéon Sorbonne, France, mkirschpin@univ-paris1.fr
[2] Centre de Recherche en Informatique, Université Paris 1 Panthéon Sorbonne, France, souveyet@univ-paris1.fr

**ABSTRACT.** Engineering context-aware applications, i.e. applications that are able to adapt their behavior according to context information, is a complex task. Not only is context a large and complex notion, but its support on software applications involves tackling multiple challenges and issues. These challenges involve not only technical challenges, but also quality concerns. Indeed, with the growing development of context-aware applications, it is becoming essential to start considering the quality of context on every step of the application development. The goal of this paper is to provoke discussion on the issues related to the support of the notion of context and its quality concerns on software applications. We present here a roadmap on context management considering different dimensions of supporting context and quality of context (QoC) on software applications, and a literature review of solutions and issues related to these dimensions. Through these, we aim at sharing with non-expert designers the necessary expertise on context management allowing them to better understand the notion of context and QoC and their challenges.

**RÉSUMÉ.** La conception d'applications sensibles au contexte, i.e. applications capables d'adapter leur comportement au contexte d'exécution, est une tâche complexe. Non seulement la notion de contexte correspond à un concept large et complexe, mais également son support au sein d'un logiciel implique la prise en compte de plusieurs challenges. Ceux-ci ne se limitent pas aux challenges techniques, incluant aussi le support à la qualité de contexte (QoC). Avec le développement croissant de ces applications, il devient essentiel de considérer la notion de qualité à chaque étape de leur développement. L'objectif ici est ainsi d'inciter la discussion et la prise de conscience sur ces différents aspects liés à la gestion de contexte et de ses paramètres de qualité. Nous présentons une roadmap tenant compte des différentes dimensions nécessaires à la gestion de contexte, ainsi qu'une révision de littérature discutant les solutions et les problèmes liés à ces dimensions. A travers ces éléments, nous voulons partager une connaissance nécessaire à la compréhension de la notion de contexte et de QoC, et à la conception d'applications sensibles au contexte par de concepteurs non-experts.

**KEYWORDS.** Context-aware computing, context engineering, Quality of Context.
**MOTS-CLÉS.** Informatique sensible au contexte, ingénierie de contexte, qualité de contexte.

### 1. Introduction

Observing the environment using software applications is now possible. The development of low cost sensors, actuators, nano-computers and other IoT-related technologies is allowing software developers to easily propose applications that observe and interact with the physical environment. Applications may now observe the execution context and integrate such information into their own behavior. The growing interest for IoT applications demonstrates this tendency quite well.

The capability of sensing enables the design of new intelligent systems that are aware of their context and able to adapt their behavior accordingly [2]. In other words, thanks to this growing development of technology, one may also expect to observe in the next few years a growing development of context-aware applications, i.e. applications that are able to adapt their own behavior according their execution context [1][21]. We are already seeing this phenomenon, with an increasing number of applications that are able to observe elements from the environment (e.g. the user's location, physical activity, etc.) and to adapt the content proposed to the user accordingly. This kind of application is already part of our everyday life. However, in most of cases, its development is still

 

performed in an ad-hoc way, despite all the research that has been done about context-aware applications. Undeniably, supporting context information on software applications involves several technical challenges, with multiple impacts on the application architecture. Currently, the main challenge is no longer on the development itself, but mainly on understanding the issues involved and on exploring the opportunities that arise through these new technologies. Indeed, in order to go further with the technology itself and the development of simple ad-hoc solutions, it is necessary to better understand the notion of context and its challenges, since this notion is central for the design and conception of new solutions.

Understanding the notion of context and its support is a complex but necessary task. It is complex because the notion of context is itself a complex and ambiguous notion, whose support on software systems involves several technical issues. It is necessary because it is only by understanding this notion and its support that we will be able to explore the full potential of it and all the opportunities it opens for business models and Information Systems. It is only through a better understanding of this notion that a real "context engineering" process can be achieved, allowing the production of new complex and extensible context-aware applications. More than ever, it is becoming necessary to form a new generation of software engineers capable of "thinking" about context in the same way they are able to think about components and about object-oriented solutions.

In our opinion, it is important to supply non-expert designers with the necessary knowledge about the issues and challenges related to context support and management on software systems. It is only through this knowledge that the above-mentioned understanding will be developed. In the past, we have identified a set of dimensions, which we consider to be necessary for such support [31] and analyzed the impact of quality considerations on such dimensions [32]. These dimensions act as guidelines in a requirement analysis process, helping non-expert users to identify necessary issues on context support for new software applications. This support can be greatly affected by quality concerns (for instance, precision and uncertainty issues affecting the information reliability), making the quality support a transversal concern affecting all dimensions of context support.

Together, all these aspects, analyzed separately in [31][32], offer a global view of the challenges involved with software support for the notion of context. In this paper, we discuss this global view, thanks to a literature review pointing out existing solutions and open issues related to context support and management. The goal of this paper is to build a survey on context engineering for non-expert software developers and designers. This survey is intended as a basis for training new "context engineers", capable of understanding and building new context-aware applications for tomorrow's Information Systems.

The paper is organized as follows: Section 2 introduces our motivations and illustrative scenarios; Section 3 discusses the context engineering dimensions and their quality support, extending what we have proposed in [31] and [32]; Section 4 introduces a literature review, pointing out challenges and solutions for each context support dimension; finally, Section 5 discusses conclusions and future work.

## 2. Motivation & illustrative examples

### 2.1. *Illustrative scenarios*

Today, it is undeniable that computation is embedded into our everyday life; we continually use computational devices without thinking of them as computational in any way [4]. Indeed, we live surrounded by multiple computing devices, forming a truly pervasive environment, as envisioned by [63]. The dynamic and ad-hoc nature of such environments leads intrinsically to important adaptation needs: the environment has to adapt to changing operating conditions and changing user preferences and behaviors in order to enable more efficient and effective operation, while avoiding system failure [26]. The user acceptation of such environments depends on these adaptation capabilities.

 

Context-aware systems can be seen as applications that are able to respond to these changes. They are defined as applications capable of observing context changes and adapting their behavior accordingly [1] [21]. Compared with traditional software applications, context-aware applications can be considered as more complex since they must cope with heterogeneous and dynamic environments. They have to run, often continuously, under changing conditions. They must observe different elements from the environment and react to their changes accordingly, often using very constrained computing platforms (for instance, nano-computers or smartphones with battery and connectivity limitations). Such a dynamic and constrained execution environment has a significant impact on the software architecture and development, notably in terms of modularity, integration, interoperability and increasing number of non-functional constraints (e.g., robustness, scalability). Under these conditions, traditional software qualities, such as flexibility, dynamicity, modularity and extensibility, become difficult to satisfy, notably with ad-hoc development processes, which are still frequently adopted when developing context-aware applications, as observed in [2][3].

A central aspect that makes developing context-aware applications complex is the notion of context itself. The notion of context corresponds to a large and ambiguous concept that has been analyzed several different ways in Computer Science and other domains [6] [7] [8] [42]. Supporting this notion on a software application involves different challenges, from identifying relevant context information, acquiring and modeling it up to its interpretation and exploitation for different purposes [1] [31] [32]. It quickly becomes arduous for non-expert designers to design and build new applications using this notion.

Before considering the challenges of developing new context-aware applications, let us consider some scenarios for such applications. Several application domains may benefit from context-aware systems, including the smart cities and smart agriculture domains. In order to demonstrate the interest of such applications, let us consider three illustrative scenarios.

The first scenario we would like to mention is a flood warning scenario, proposed by [28] [55]. This scenario, called GridStix, considers a Wireless Sensor Network (WSN) deployed on the Rivers Ribble and Dee in England and Wales. Each GridStix node (illustrated in Figure 2.1a) consists of depth and flow sensors in which power is supplied by batteries, replenished by solar panels. Nodes are equipped with both 802.11b (Wi-Fi) and Bluetooth communications for inter-node data transmission and with a GSM uplink node. In addition to the different transmission and data collecting modes, each node can be activated or deactivated according to the power level of the corresponding battery and the state of the river. Based on the information collected from GridStix nodes, a flood warning application considers a stochastic model for predicting flooding situations. It may also perform adaptation actions, such as activating or deactivating nodes for battery saving according the node's power level and neighborhood nodes' status (preventing low quality observation on some portions of the river). In order to support such a dynamic adaptation of the configuration of the WSNs, the system has to be aware of changes in the nodes' context and to respond in a reactive and proactive manner. This implies not only data acquisition, by collecting data from GridStix sensors, but also transferring this data and making it available for processing stochastic models. It also implies specifying adaptation policies that state the actions required to adapt the running system to a configuration that better fits its current context (e.g. change a node with a low battery to a neighboring node with a full of battery and turning off a node to save battery power).

Another scenario that we would highlight is the one considered by the project CC-Sem[1], whose goal is to develop an integrated platform for smart monitoring, controlling, and planning of the energy consumption and generation in urban scenarios. This project considers that the capabilities of monitoring/controlling/managing the energy consumption and generation are very important when

[1] https://www.fing.edu.uy/inco/grupos/cecal/hpc/cc-sem/

 

implementing the smart city paradigm. In this scenario, one may consider the use of smart electricity meters for collecting consumption information from homes, as well as other sensors for collecting information such as temperature, humidity and weather conditions. Collected data could then be analyzed in order to identify patterns of energy consumption. Such patterns may be used as the basis for recommendation purposes on smart home controllers, suggesting economy actions for final users (e.g. reducing air conditioning or heating intensity, based on weather information), but also for preventive actions such as turning off water heaters and air conditioning systems in case of system overload. They could also be used by electric grid administrators and energy providers in order to better predict consumption and anticipate preventive actions for preventing problems due to over consumption. Carrying out such a scenario demands not only the deployment of intelligent energy meters and temperature/humidity sensors, but also the deployment of an appropriate infrastructure. Such an infrastructure is necessary for collecting and transferring raw data, as well as for analyzing it using Big Data techniques. It also implies dealing with privacy and security issues necessary for keeping personal consumption data safe and secure.

(a)  (b) 

**Figure 2.1.** *(a) A GridStix node taken from [55]. (b) A prototype of hydric stress monitoring system.*

Finally, a third scenario we would like to highlight is the application of IoT to the smart agriculture domain. Indeed, the use of sensors and actuators opens new perspectives for the agriculture. By using different sensors, such as humidity, hydric stress, luminosity and temperature sensors, it is possible to better monitor the overall state of health of plants and production. Such monitoring activity can be used as the basis not only for decision-making actions, but it can also trigger preventive actions automatically. Small cultures, like flowers, tomatoes, strawberries or spices, often deployed over greenhouse structures, may benefit from a constant surveillance of temperature and hydric conditions. Data observed from sensors directly located in the field can be used for decision making: producers may receive daily reports about their crops and decide preventive actions for saving or improving production. Such data may also be used for taking actions automatically, controlling, for instance, the water supply of a given position of the crops according to the plants' hydric stress. Figure 2.1b shows a very small prototype of a hydric stress monitoring system in which an Arduino nano-computer is used to monitor hydric stress, thanks to a soil moisture sensor, and to control a water pump accordingly. Commercial systems similar to this prototype are already available for domestic users, like the Daisy system[2]. Nevertheless, deploying this kind of equipment for a professional crop demands considering not only the infrastructure and energy supply, but also tackling challenges such as choosing the most suitable sensors and data to be monitored, choosing the necessary frequency among data collection

[2] http://daisy.si/

 

according the crop and environment needs, choosing the appropriate thresholds for triggering actions, or choosing how to represent/store collected data for better reporting and analyzing. The quality of decision making (automatic or not) depends on these issues, from the selection of data to be observed up to analysis.

This scenario also illustrates the ambiguity that may characterize context-aware applications. One may easily wonder if this scenario corresponds to a context-aware application or to a self-adapting one [16]. What is the difference, if there is a difference, between those? Should all the collected data be considered to be context data or just application data? For non-expert designers these questions appear to be hard to answer and even to understood, notably because the notion of context and context-awareness are complex and hard to understand without the necessary knowledge. Promoting this knowledge to non-expert designers is necessary if we want to stimulate the development of such new context-aware applications.

### 2.2. *Illustrative target group*

The number of applications exploring the notion of context is constantly increasing thanks to the growing development of sensor technologies now often embedded into smartphones, tablets or associated with IoT devices. Before being considered as context-aware, applications such as those in Section 2.1 can be seen by consumers and non-expert designers as “smart” applications, since they propose what can be perceived as an “intelligent” behavior with data collected from the environment. Indeed, being able to observe the physical environment, to dynamically adapt its behavior without any human intervention are behaviors commonly perceived as “intelligent” (or “smart”) by consumers, and consequently, more and more developers are considering integrating such behavior into their new applications.

Even if these applications are becoming more and more popular, their design and implementation is often poorly understood or has not been mastered sufficiently by non-expert designers. In order to illustrate this situation, we have invited two groups of master’s degree students in computer science to participate in a survey. In this survey, students are invited to answer a set of 50 questions about their practices on designing/engineering “intelligent” software applications. These questions consider different aspects concerning the project, the design and the development of such smart applications. Both groups, containing about 20 students each, were composed, in their majority, by students in apprenticeship, with 1 to 3 years of experience, and for the others about 4-6 months of internship, both on software (mainly Web) development or Information Systems.

Before submitting the students to the survey, we gave them a small experiment using a RaspberryPi nano-computer (see Figure 2.2). Students had to build a small application that observed temperature in the room and reacted to the observed temperature through a LED (red LED if temperature is greater than a threshold, yellow if it is lower, and green otherwise). Students were organized into small groups. Each group received a kit containing one RaspberryPi ZeroW, a temperature sensor I2C BMP 280 and a triple LED (see Figure 2.2), as well as a SD card containing the OS Raspbian Jessie and two examples of code for handling the sensor and the LED. It is only after connecting the sensors and building the application that students received the survey for completion.

Although all the students concerned had a computer science background and an academic background in software development, only half of them (about 48.83%) had declared having already participated in a project developing a smart application. Among those that had not yet participated, about 63.6% were planning (or would have liked) to participate in such kind of project. Among those with previous experience, mobile and Web applications appeared to be the most commonly concerned, counting for respectively about 66.67% and 42.85% of those students, followed by location-aware applications and smart home applications (concerning each 28,57%)[3]. Independently of their previous

[3] Students could select more than one category of application.

 

experience, almost all students declared to know platforms commonly used for context-aware applications: 100% declared to know RaspberryPi and Android Phone, 86% for iPhone, 83% and 81% for iPad and iPad Pro, 81% for Android tablets and 48.83% for Arduino. A similar tendency appeared when considering platforms they had already used or wanted to use: 93% declared to use/want to use the Android phone as a target platform, 72% for RaspberryPi, 67.44% for Android tablets and iPhone, 30.23% for Arduino and even 11.62% for SmartTV platforms. These numbers illustrate quite well the growing interest these young developers have for creating applications for mobile and IoT platforms, and for exploring the possibilities of these platforms.

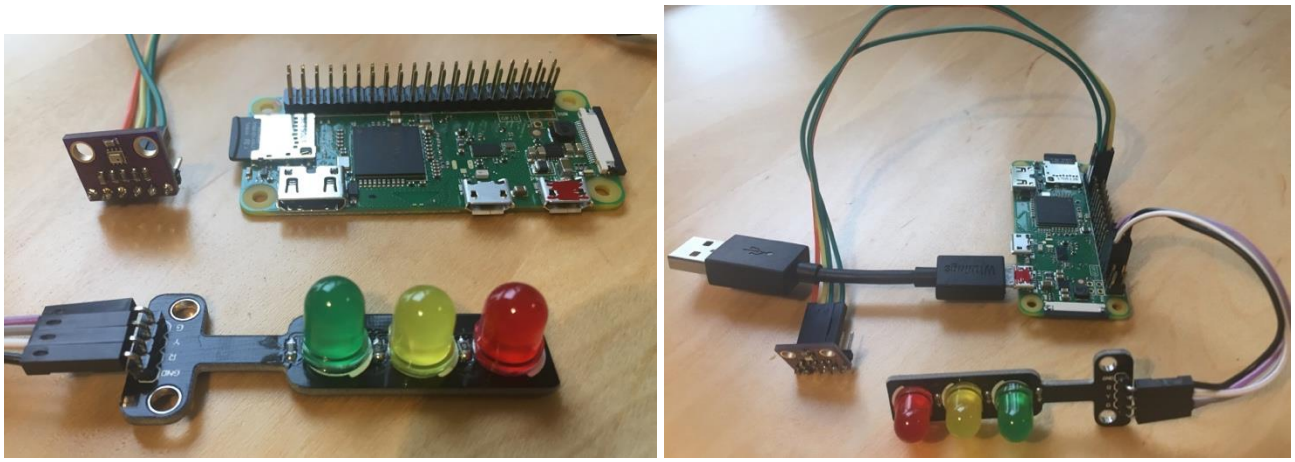

**Figure 2.2.** *Kit distributed to the students containing a RaspberryPi ZeroW, a I2C BMP 280 temperature sensor and a triple LED.*

It is worth noting that all these students had already been briefly introduced to the notion of context and to Pervasive Computing during their scholarship. Although the notion of context was not totally unknown to these students, they still could not be considered as "experts" in the domain, being mostly novices in this area. When asked about programming platforms (libraries, APIs, framework or middleware), almost no particular technology for context-aware and IoT application was cited. Only 4 students cited Pi4J, the library proposed in the examples used in the experiment they performed with the RaspberryPi. Among those with previous development experience, 36.36% declared using an ad-hoc direct access to the devices and 31.81% declared using the OS/programming language calls/API. The multiple platforms and technologies proposed in the literature (e.g. [18] [13] [22] [53] [61]) seem to remain unknown and unexplored for these students.

Similarly, when we asked students with previous development experience in "smart applications" when, during the previous project(s), the technologies were chosen, about 22.72% couldn't answer ("I don't know") as much as "at the very beginning, it was predefined in the project specifications". When asked about the difficulty of using these technologies, about 31.81% of the students evaluated the difficulty as "medium", pointing out that some elements were unknown and that they had to learn how to handle these technologies. It is interesting to note that no student evaluated the difficulty as "easy" and only about 9% declared it as "hard", pointing out that they had never used such technologies before. In any case, about 90% of students with previous experience declared that having previous knowledge about these technologies is necessary (50% considered that "it helps a lot" and 40% chose "it could help, but it is not mandatory"). Better dissemination of the knowledge about these technologies seems to be an interesting means for the progress of context-aware applications.

Finally, to the question "have you already heard about context-aware computing?", almost 52.27% of the students answered "yes, I have some notions on it" and 22.72% answered "yes, very slightly", which corresponded to the expected values since both groups of students already had some academic background in this topic. Still, when asked about whether the application they had built or wished to build was "context-aware", about 36.36% said "I don't know". Similarly, when asked if they were

familiar with the notion of “context”, about 43.18% answered “yes, slightly”, 25% declared to overcome the concept and 25% assumed to have some difficulty with it. Nevertheless, when asked how this notion could correspond to their past/future application(s), only 34.09% of the students could answer this question. Through these elements we may observe that the understanding of the notion of context and its use on software applications seems to remain a challenge for these computer science students, even for those with some previous experience on smart applications.

Students like those who participated in this survey illustrate the public we are focusing on here: software designers who, despite some experience, are not necessarily experts on the design and on the development of context-aware applications, but who could be led to participate in this kind of project in the near future.

## 3. Context engineering roadmap

As one may observe from the scenarios in Section 2.1, engineering context-aware applications can become a difficult task due to the complex nature of the notion of context as well as the different aspects that should be taken into account for considering this notion. Engineering such applications implies taking into account different aspects involving the notion of context and its support, which include collecting, transferring and analyzing context data in dynamic environments. Non-expert designers, such as the students cited in Section 2.2, were left alone to understand and identify the necessary concepts and components for building such applications. Acquiring the necessary knowledge for developing software applications exploring this notion thus represents a challenge for non-expert designers.

In [31], we tackled this question by considering multiple dimensions necessary for supporting context on software applications in a context engineering roadmap. This roadmap represents, for us, a first step towards a global approach, allowing us to better grasp the different aspects involved in the management of context information. It considers different challenges related to context management, organized along multiple dimensions. Each dimension focuses on different aspects and tackles different issues necessary to context management.

Additionally, in [32] we extended this roadmap in order to take into account quality aspects on context management. Indeed, the roadmap proposed in [31] does not consider the influence of quality, limiting its analysis to a few dimensions, such as the acquisition of context data. Still, managing the quality of context information demands a deeper reflection about the consequences of quality on the application behavior, and its influence over every aspect of context management. With the growing development of context-aware applications in multiple domains (healthcare, smart homes, transport, etc.), the importance of managing Quality of Context (QoC) is also growing, since the consequences of low-quality observations on the application behavior might be dramatic, or at least, may seriously affect the application reliably. These consequences can be illustrated when considering the smart agriculture scenario (cf. Section 2.1): low quality observations from malfunctioning or non-functioning soil moisture sensors may lead to erroneous decisions considering irrigation, exposing the production to the consequences of a too abundant or too sparse irrigation. According to [58], context information is naturally dynamic and uncertain: it may contain errors, be out-of-date or even incomplete. The quality of the information collected by a given sensor may vary according to several different and possible unpredictable factors (e.g. weather and wind conditions, failures of energy supply, communication interferences, etc.), leading to erroneous, incomplete or missing information. Uncertainty being indissociable from context information, handling quality of context becomes a central concern for reaching the reliability that is mandatory for the development of context-aware applications in the near future.

Since the quality of context information may potentially affect all dimensions represented by the roadmap, it should be considered as a transversal concern affecting all aspects of context management.

 

Again, the aforementioned scenario on smart agriculture illustrates this point quite well: errors in the information collected, as well as in the threshold definition may affect the water supply to crops and lead to an excessive or an insufficient water provision, which will in its turn influence the production.

In this paper, we propose a unified view of [31] and [32], which is enriched with a literature review (cf. Section 4), covering challenges and solutions related to every dimension pointed out by the roadmap. In the next few sections, we describe the context engineering roadmap, with its proposed dimensions, as well as the quality dimension, proposed as a transversal plan affecting all previous dimensions.

### 3.1. *Context support dimensions*

As briefly illustrated by the scenarios in Section 2, several kinds of software application may use the notion of context. Most known applications are probably context-aware applications, which use this notion for adaptation purposes, adapting the behavior of the application accordingly. Nevertheless, adaptation is not the only possible purpose of using context information on a software application. This notion may be explored in several ways, with different implications on the application design and behavior. Whatever the purpose of using context information, handling such information means dealing with different aspects related to its management, from its observation up to its use on a software application. For instance, deciding modalities and means for data collection and transfer on Grid Stix and CC-Sem scenarios (cf. Section 2.1), choosing threshold on smart agriculture one, defining adaptation mechanisms on GridStix or data analysis methods for CC-Sem, are just a few examples of issues that should be considered when developing these scenarios. Identifying these challenges and issues related to the management of the notion of context on software applications becomes mandatory for supporting the growing development of new intelligent applications using this notion. We believe that considering context information through its multiples challenges may contribute to a global approach for designing such applications, by allowing a better understanding of the different aspects involved in the management of context information.

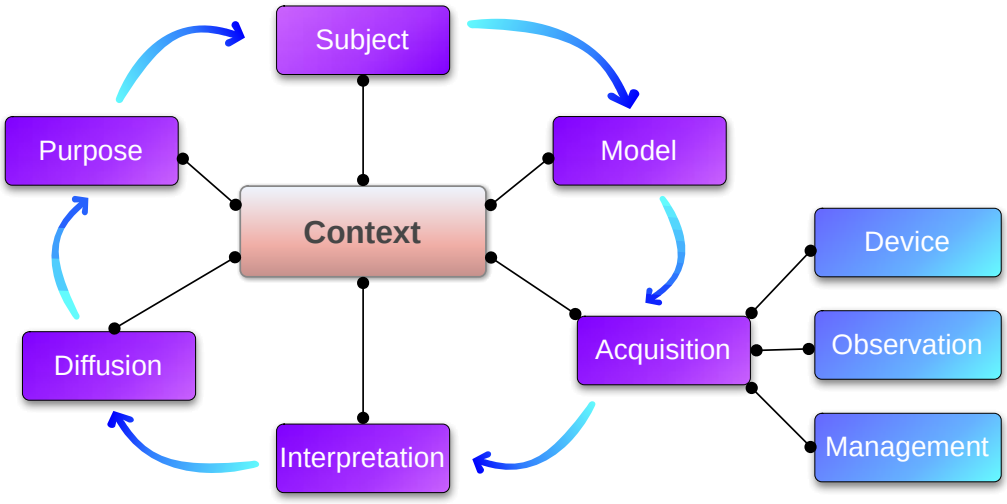


**Figure 3.1.** *Context engineering roadmap [31].*

Indeed, engineering context-aware applications involves tackling several challenges involving the notion of context and its support. By analyzing several existing works applying this notion on software applications, we could identify in [31] the most relevant characteristics of context management required by context-aware systems and organize them according to six dimensions, represented in Figure 3.1: purpose, subject, model, acquisition, interpretation and diffusion. Each dimension identifies challenges and issues, leading to the identification of functional and non-functional goals that should be considered and satisfied (at least partially) by these applications. These dimensions do not necessarily follow a particular order. As demonstrated by [2], projects developing intelligent software

 

applications using the notion of context do not follow a singular process; the adopted process may change according to the team and the project itself. Nonetheless, in order to simplify the presentation of the roadmap, we will discuss the proposed dimensions respecting the order represented in Figure 3.1.

The first dimension we consider is the Purpose dimension, which focuses on the purpose of using context information in a given application, and on the meanings and mechanisms for reaching it. This dimension considers why a given application needs context information. Such a purpose has a significant impact on how this information is exploited and consequently on what information will be considered, how it will be acquired, represented and analyzed, which are issues considered by subsequent dimensions of the roadmap. For instance, considering the scenarios presented in Section 2, two main purposes can be highlighted: the adaptation purpose, such as in the third scenario, in which the water supply is automatically adapted according to humidity conditions; but also, decision making, like in CC-Sem scenario, in which information collected about energy consumption can be used by power grid administrators for decision making.

The second dimension in this roadmap is the Subject dimension. It focuses on what information could be considered as context and how to identify relevant elements. This is not a trivial question, since the notion of context corresponds to a large and often ambiguous concept [42][7][25], potentially referring to very different elements, whose relevance depends on the use we will make of it. The subject dimension is directly connected to the first dimension, since the purpose of using context information in an application will influence the relevance (or not) of a given information for that application.

Multiple definitions have been proposed for the notion of context [7] [42] [25]. One of the most commonly accepted considers context as any information that can be used to characterize the situation of an entity (a person, place, or object) that is considered relevant to the interaction between a user and a system [21]. This definition points out both an observed entity (e.g. the user) and a piece of information that is observed about this entity. The entity delimitates the observation: the aim is to observe a given entity, but when looking at this entity, different elements can be observed. For instance, when considering a user (i.e. an entity), it is possible to observe his location, his mood, his level of expertise, etc.; when considering a device (i.e. another entity), it is possible to observe its available memory, network connection, etc. Thus, the entity corresponds to the subject of the observation. It plays a central role in context modeling, as pointed out by [17] [9], since it is precisely the context of this subject that is currently been observed. Everything we observe is related to this subject [34]. He we call the information observed (location, memory, etc.) about a given entity/subject the context element. For [34], when observing such context elements, we obtain *values* corresponding to their current status that will probably evolve over the time. For instance, by observing the *context element* ‘location’ for a *subject* ‘user’, we may obtain *values* for latitude and longitude, corresponding to the current user’s location. Similarly, in the smart agriculture scenario, the *context element* ‘temperature’ of a land parcel, representing our *entity*, can be estimated through multiple *values* obtained from a temperature sensor. Figure 3.2 represents a meta-model proposed by [34] in which context information corresponds to a set of context elements that are observed for a given subject (entity) and for which multiple values can be dynamically observed.

 

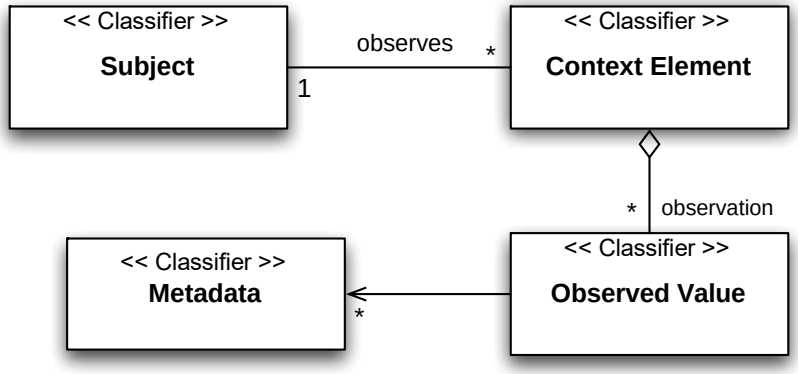


**Figure 3.2.** *Context meta-model from [34].*

Identifying what information should be considered as a context element in a given application means identifying what entities and context elements should be considered as relevant. This relevance depends on the identified purpose. Context information is observed in order to satisfy this purpose. Its relevance depends on its contribution to this purpose. For instance, in GridStix scenario (cf. Section 2), considering that the goal of this system is the flooding prediction, the main entity to be considered is the river and more precisely its state. The latter is determined by observing the depth and the flow rate of the river at different points. Nevertheless, the GridStix nodes themselves can also be considered as a possible entity since it is necessary to observe their battery level and communication capabilities (Bluetooth and Wi-Fi connections) for self-adapting the overall infrastructure in order to reduce the energy consumption.

This identification process remains a significant challenge, as identified by [21] [25] [2], notably because any information that can be used to characterize something (an entity or the subject in the metamodel in Figure 3.2) can be potentially considered as context. The main aspect to be taken into account remains its relevance, if it is relevant for characterizing a given entity, considering the system purpose. For instance, information related to the battery level can be considered as relevant in the GridStix scenario because it supposes adapting node behavior according to this. In the CC-Sem scenario the same information can be ignored since its domestic use allows us to suppose a continuous power supply. According to [2], since context is a crucial element that defines the functionality of a context-aware system and shapes its behavior, context selection becomes a significant task in the design of these systems. For these authors, system designers need to anticipate the relevant combinations and characteristics of context before implementing the system, and to decide which context to include in their design.

However, just identifying relevant context information is not enough, it is also necessary to consider how to represent this information in an appropriate manner. The Model dimension focuses precisely on context modeling issues. Its main goal is to determine the most appropriate representation for context information on a given application according to its identified purpose. Context information must indeed be represented, internally, in a software application, in such a way that it becomes practical and possible for this application to explore it and to realize its purpose. As pointed out by [34], a context model ensures the definition of independent adaptation processes and isolates this process from context acquiring techniques. An inappropriate model may compromise, or at least make more complex, the implementation of a given application. For instance, in the GridStix scenario, the adoption of an appropriate context model (e.g. an object-oriented model as suggested in [31]) allows, on the one hand, the definition of adaptation rules (notably for turning on and off a given node) independently of precise sensors or APIs used for obtaining the data. On the other hand, an inappropriate model (e.g. a too complex model) may negatively impact the performance of the flooding prediction system since data analysis by stochastic models may demand extra processing of the available data. Several approaches of context modeling exist, from key-value sets and object-oriented models up to complex ontologies

 

[42] [6] [12] [33] [54] [24]. Simple models, such as key-value ones, will be easy to implement but will offer no particular reasoning mechanism. On the contrary, ontology-based models will be more complex to implement, but they will allow complex reasoning mechanisms.

Representing context information is a challenging issue due to the nature of this information. Firstly, context information can be heterogeneous. Since different kinds of context element can be observed, the information obtained may vary from numeric information, like GPS coordinates or a percentage (e.g. CPU load), to symbolic values (e.g. the role of a user in a group). Such heterogeneity can be observed on both content and data structure. For instance, in the GridStix scenario, multiple elements can be considered (as pointed out in [31]), such as the flow rate and depth of the river (the observed entity), and the battery level and Bluetooth state from the nodes. Most of these consider numeric values, followed by a timestamp (meta-data describing the observation), except the Bluetooth state, which can be represented by a symbolic value (‘on’ or ‘off’). However, in other scenarios, context elements with a more complex structure can be observed. For instance, when considering location information, multiple representations are possible, such as GPS coordinates or postal address. Both representations are composed of multiple values (at least latitude and longitude for the former, street name and number, locality, zip code, country, etc. for the latter). It is the context model that organizes and structures data obtained from sensors and other data sources into valuable information that can be explored by the application.

Furthermore, context information is naturally dynamic, varying among observations. For instance, in our hydric stress scenario (cf. Section 2), observed values for humidity levels will vary between observations according the plants’ consumption and weather conditions. This dynamicity must be supported by the context model, which should keep values associated with context elements and assume that these values will vary over time. Indeed, we may consider that, by definition, context is about characterizing the situation of an entity that is (or may) be constantly evolving.

Finally, context information is also uncertain and often incomplete or presenting errors and imprecisions [58] [14] [13] [37], mainly due to problems during the acquisition of data (connection problems, interferences, etc.), resulting in erroneous or missing data. For instance, in the CC-Sem scenario, the weather conditions and notably exposing temperature sensors to direct sunlight may adulterate the quality of the measurements. Similarly, in the GridStix scenario, the river conditions (e.g. an important flow in a flood period) may damage GridStix nodes, causing missing data on portions of the river. This uncertainty represents an important issue since this data influences the behavior of a context-aware application, making the quality of context a very important issue when considering context support on software applications. The influence of quality concerns on the context management is not limited to only context modeling, representing a transversal concern. We will discuss this concern and its influence on all dimensions of the roadmap in Section 3.2.

It is worth noting that these three aspects (heterogeneity, dynamicity and uncertainty) profoundly characterize context information. Handling these aspects is a key factor for successfully exploiting context information in a software application. More than in traditional applications, managing context data implies handling these aspects as a priority.

Context information should be acquired by observing the environment around a given entity. The Acquisition dimension highlights the challenges of acquiring context information from the environment, which implies considering the capture devices, the observation process and the management of context sources. Indeed, in order to correctly capture context information from the environment, one has to observe this environment, using an appropriate acquisition device. The main challenges here are the heterogeneity and the interoperability of these devices since their nature can be quite variable. For instance, a user’s location can be acquired using a GPS device or calculated from a Wi-Fi connection; temperature data, necessary for CC-Sem and smart agriculture scenarios, can be observed using very different kinds of sensors, which are not necessarily interoperable. Such

 

heterogeneity makes it more complex to support different context elements in a given application, as well as the evolution of existing applications (e.g. observing new context elements or modifying the acquisition device), since changing the observation device may seriously affect the application code or design. This is particularly true in large deployment scenarios, such as CC-Sem and smart agriculture ones. In these cases, in which a large number of sensors are to be deployed, it is particularly important to support multiple models of sensors (e.g. temperature sensors on CC-Sem scenario) and to easily replace one sensor with another similar one (e.g. a soil moisture sensor by another humidity sensor). Thus, when considering the acquisition dimension, it becomes imperative to consider how to isolate the software application and its behavior from the precise technology used for acquiring context information, as underlined by authors such as in [21] [6].

Moreover, using a given acquisition device for observing the environment means fulfilling the context model with observed values. The Observation process should consider not only the device used for this, but also the observation frequency, according to the expected dynamicity of the observed information. For instance, location information will demand an active observation in order to guarantee some accuracy, while the user's role can be acquired on-demand. Once observed, this information also has to be kept updated in order to represent the current context of the observed entities. For instance, in the smart agriculture scenario (cf. Section 2), the information obtained from a soil moisture sensor must be regularly updated in order to keep track of changes on the plants' hydric stress level. These updates should be frequent enough to satisfy plants' hydric requirements, but too frequent observations will probably be inefficient or even useless since the plants' current hydric situation will not change drastically over a very short period of time (e.g. several seconds). The acquisition dimension also involves managing the environment. The environment itself being dynamic, the availability of devices used for observation is not guaranteed. Some devices may disappear (e.g. being switched off) and others may join the environment, becoming available for capturing the context of a given entity. The management of this dynamic environment is also a challenge, considering the evolution of the environment and the availability of the acquisition devices in it. The GridStix scenario is a good example of this management, since GridStix nodes can be switched on and off, coming in and out the system, according to battery conditions.

Data collected during the acquisition process corresponds to raw data that often have to be aggregated or interpreted in order to be better exploited by context-aware applications. The Interpretation dimension focuses on this issue, considering the challenges related to the interpretation of context information in its different forms (interpretation rules, context mining, etc.). It considers how to transform raw context data on useful knowledge for a given application. For instance, when considering the smart agriculture scenario, a soil moisture sensor has been used in the prototype illustrated in Figure 2.1b for evaluating the humidity level. Raw data offered by this sensor corresponds in fact to impedance values observed on the soil parcel around the sensor. This raw data is compared to predefined thresholds in order to deduce the parcel humidity level. The goal of this dimension is then to specify appropriate interpretation mechanisms and to consider necessary reasoning and aggregation mechanisms that can be applied according to the capabilities of the context model. Different interpretation mechanisms can be considered, from ad-hoc reasoning up to complex rule-based systems [54][23]. Those mechanisms cannot be dissociated from the context model. Not only will the context model limit the possibilities of interpretation (i.e. a key-value structure will offer fewer reasoning opportunities than an object-oriented or an ontology-based model), but the information that will also be deduced from the interpretation mechanism will also feed the context model, similar to an acquisition mechanism, building for instance high level information from raw data.

 

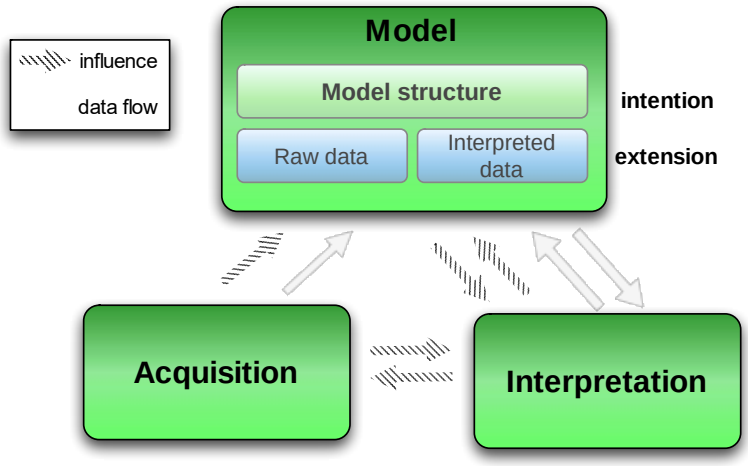


**Figure 3.3.** *Influence and data flow among Model, Acquisition and Interpretation dimensions.*

The three dimensions, *Model*, *Acquisition* and *Interpretation*, are then intrinsically connected and cannot be dissociated, as illustrated by Figure 3.3. Firstly, they are connected because the mechanisms on these dimensions exchange data, represented by the data flow in Figure 3.3: acquisition mechanisms feed context model with raw data, which is also consumed by the interpretation mechanism, whose results will again feed the model extension (i.e. instances). Secondly, these three dimensions influence choices about each other (as illustrated by the influence arrow in Figure 3.3): acquisition devices may influence the interpretation mechanisms that can be applied (for example, gyroscope and accelerometer data that can be interpreted on a user's movement information thanks to statistical methods such as in [51]); conversely, the interpretation mechanism can influence the selection of acquisition methods (for instance, triangulation methods can be used to deduce a location from GSM-based data instead of GPS, as in [48]). Similarly, decisions about interpretation and acquisition may influence the model, both its intention (i.e. structure) and extension (i.e. instances), and the model will guide and limit interpretation possibilities (for instance, rule-based mechanisms, such as [23], will be hard to apply without an ontology-based model, and statistical methods such as Bayesian networks applied on [51], often require a numeric representation of data).

Finally, the Diffusion dimension explores the issues related to the transmission of context information among multiple nodes. Indeed, context-aware applications often behave as distributed applications, in which multiple nodes communicate and exchange information about their current state. In some cases, the context information should be distributed from the node in which it is observed to a different node, in which it will be processed, interpreted or stored. For instance, in the GridStix scenario (cf. Section 2), context information about the river and the nodes' conditions is transmitted for remote processing, for a flooding warning application. Similarly, in the CC-Sem scenario (cf. Section 2), context information should be transferred from acquiring devices to an appropriate computational infrastructure allowing the data analysis of the energy consumption. Several challenges arise from this distribution, above all, the stability of the context information (how long does a given piece of information remain valid and useful after being transferred from a different node?) and the coherence of the collected data, since contradictory data can be reported from multiple nodes observing a given entity (e.g. multiple temperature sensors observing different values for a given room according to external influences such as sunlight or heating).

In addition to the dimensions considered by the roadmap in Figure 3.1, another concern must be considered: the quality of context (QoC). As one may observe, the roadmap presented here does not consider the influence of quality in depth. In our opinion, managing the quality of context is not only a matter of correctly representing the context information or its meta-data. It demands a deeper reflection on the influence of QoC on all aspects of context management. For us, quality should be a transversal

 

concern affecting all dimensions of the context management. In the next section, we discuss this influence, extending what we introduced in [32].

### 3.2. *Quality on context support*

As introduced earlier, Quality of Context (QoC) is a transversal concern that influences all aspects of the context management. According to [5], Quality of Context is usually defined as the set of parameters that express quality requirements and properties for context data (e.g., precision, freshness, trustworthiness...). Being able to observe and handle these properties requires the consideration of their influence on every aspect of context management, transforming QoC concerns on a transversal plan, affecting all other dimensions of the context engineering roadmap, as represented in Figure 3.4. In every dimension, particular challenges are considered and somehow influenced by the notion of quality. When considering each dimension, we should consider the influence of taking QoC into account on supporting the dimension challenges. This influence is materialized in Figure 3.4 by the verbs attached to each dimension of the roadmap, representing a guideline when considering QoC on the dimension. Thus, for each dimension, we tried to identify in [32] questions, represented through the verbs in Figure 3.4, that should be considered when thinking about the influence of QoC. Similar to the context engineering roadmap presented earlier (cf. Section 3.1), the main goal of this proposal is not necessarily to give solutions to these questions, but mainly to raise discussion on the impact of QoC on context management.

The first dimension, Purpose, considers the purposes for which the notion of context is used in an application. According its purpose (e.g. adaption, decision making, etc.), an application can be more or less sensitive to errors or low-quality context information, since errors on context information may lead to erroneous decisions from the application, which may be more or less important, according to the application domain. For example, let us consider a healthcare application that proposes to automatically adapt insulin levels according to a patient's blood sugar levels or to call the emergency services if a patient falls over. Reliability of this kind of application depends on the quality of context observed since erroneous information may lead to a wrong decision with significant consequences for the patient's health. The same can be assumed in the smart agriculture scenario (cf. Section 2), in which errors on context information could directly affect water supply and consequently production.

The question raised by the Purpose dimension is essentially whether to and how to *follow* QoC in a given application. The management of QoC should consider the consequences of a poor-quality context information and the consequences of having no information about it. These consequences should be considered but also the costs of managing QoC. For example, still considering the smart agriculture scenario, errors on humidity data may lead to an excessive water provision. Similarly, a problem affecting the humidity data provisioning (i.e. missing context information) may lead to an insufficient water provision. Both cases can be very harmful for the plants and affect productivity. Furthermore, reaching application purposes implies several mechanisms that are potentially affected by QoC. Including QoC in these may represent a cost that should be considered. Will the cost of observing QoC be more or less important than the risk of not observing it? For instance, in a healthcare application, considering QoC in the adaptation process implies using different algorithms for detecting and eliminating suspicious measures. Such algorithms will consume processing and battery power in the hosting device. These represent an execution cost, in addition to design and development costs. Even if these can be significant compared to overall application costs, the risks and the possible consequences of not considering QoC justify these costs. The application developer should then consider the risks and the costs that will follow QoC observation on the application purpose. However, one question arises from this dimension: how can we measure such risks and costs? This point remains an open question.

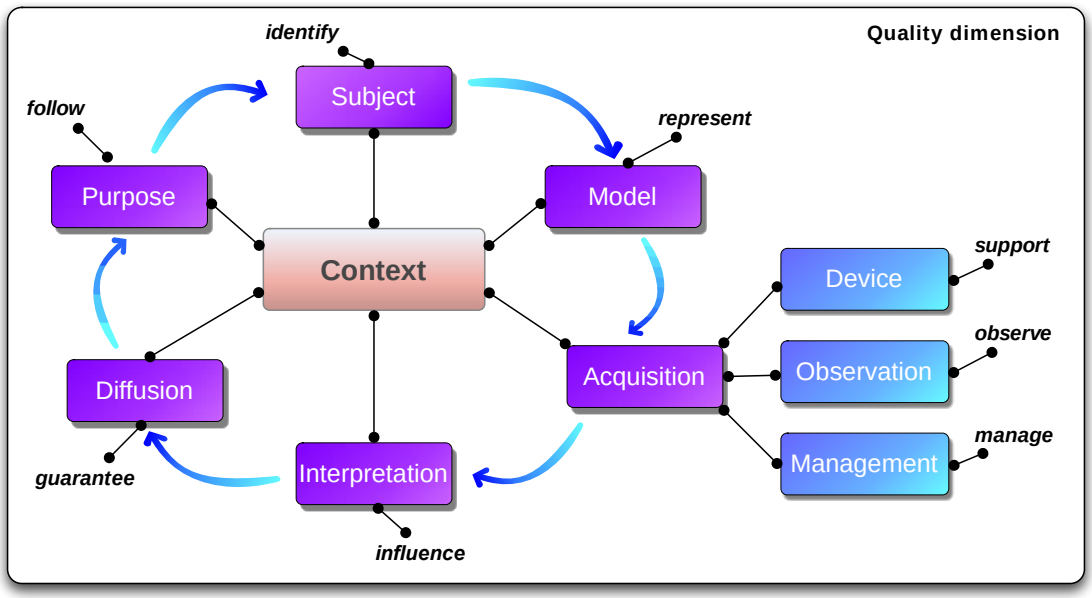


**Figure 3.4.** *Quality plan on the context engineering roadmap [32].*

The remaining dimensions may give us some insights about the costs and risks of observing (or not) QoC. The Subject dimension considers what kind of information can be observed as context information. When identifying relevant context information, one should also *identify* possible QoC indicators that can be associated with it. Often, QoC consists of several elements such as precision, up-to-dateness, freshness or probability of correctness [37][5]. Identifying what context information will be observed allows application developers to identify either what quality information is relevant to be associated with it for reaching the application's purpose. For instance, when considering location information, different quality indicators can be considered, such as estimated error or precision, freshness (which can be obtained regarding production time) or even the number of available satellites, when considering GPS data. Indeed, several QoC criteria are possible, as illustrated by [37]. These authors have identified and compared different QoC indicators proposed in the literature, highlighting the variation in terms and in meaning of these criteria. Similar to context information itself, the relevance of a given indicator often depends on the use that will be made of it. Again, the purpose of a system highly influences the relevance of context and QoC information. Both are observed considering this purpose, their contribution with its satisfaction determines their relevance. For instance, considering the CC-Sem and hydric stress scenarios presented in Section 2, both may consider accuracy as a QoC indicator, but their needs considering this indicator will not be the same, errors would be better tolerated on the first scenario than on the latter.

All identified information should be represented in an appropriate context model. Considering QoC on the Model dimension implies considering how to *represent* QoC information, how it will be associated with observed context. Several research works have been carried out on context models [6][7][8] [42], and multiple proposes have already considered the question of QoC [14] [13] [37] [27], often through meta-data representing QoC indicators. As summarized by [6], a good context modeling approach must include modeling of context information quality to support reasoning about context.

It is impossible to consider context information without considering its acquisition. The same can be said about QoC information. The Acquisition dimension considers this question through three different points of view (or sub-dimensions): device, observation and management. QoC will influence the analysis of each of these sub-dimensions. First, when considering the necessary devices for acquiring context information, it is also important to consider if these devices are able to *support* acquiring QoC indicators. Similarly to context information, QoC indicators are calculated based on information supplied by acquiring devices. The possibility of obtaining a given QoC indicator depends on the capability of these devices offering basis information. For example, when considering GPS data, the

 

number of satellites can be used for supposing data precision. If this information cannot be obtained for any reason, this criterion will be unavailable. In the hydric stress scenario presented in Section 2, one may consider using precision as a valuable QoC indicator. However, it is also necessary to consider how to obtain such an indicator, since most soil moisture sensors are unable to calculate this by themselves. A calibration phase is necessary, which may offer only an estimation of the sensor precision.

Similarly, data confidence can also be considered as a possible quality indicator. Considering, for example, a team application that constantly informs users about project context (and progress), information about task progression might heavily depend on the information supplied by the users themselves. In this case, it seems difficult to estimate data confidence and thus trustworthiness of this information. Furthermore, assuring quality of context information leads to choosing appropriate acquisition devices, and consequently, to the costs associated with such devices. For instance, on a smart home application, one may consider using ground sensors for detecting a resident's fall, since these devices may offer a better accuracy for fall detection than simple accelerometers. The costs associated with these devices are not the same, but they can be justified according to the application purposes (e.g. if the application is designed for supporting medical care or special needs residents). Software developers must be aware of these issues when considering their QoC indicators and the devices used to obtain them.

Furthermore, context observing policies are directly influenced by QoC considerations. For instance, considering if a given context information needs very frequent observation probably implies that freshness is a relevant QoC indicator for this information, and vice versa: high levels of freshness demand very frequent observations. This is often the case of location information on transport applications: when considering moving vehicles, the freshness of location information will indicate if this information can still be used or if new measures are necessary. Application developers must then consider how to *observe* QoC indicators during context observation process and how often this observation process should be realized. Finally, the management of acquiring infrastructure is also influenced by QoC. Observed environment being more and more dynamic, it is necessary to *manage* acquiring devices on this environment. This management can be influenced by QoC indicators (for example, deactivating a given device if precision offered by it is too low or reactivating it in order to increase overall system precision). This example is visible on GridStix scenario mentioned in Section 2, in which GridStix nodes containing flood and depth sensors are turned off in order to save battery and on again in order to guarantee that every portion of the river has enough sensors observing it, improving the quality of overall observation system. It is then important to consider not only how to *manage* QoC information, but also how to manage acquisition environment according to QoC information.

Similar to previous dimensions, the Interpretation dimension is also influenced by QoC considerations. Quality of context information may affect interpretation and reasoning mechanisms and influence the reliability of the target application. An illustration of this influence is given by [60]. In this work, the authors discuss a set of metrics evaluating QoC and propose using such metrics on context prediction, in order to prevent low quality information affecting prediction mechanism. Works such as [60] demonstrate the importance of considering how QoC *influences* context reasoning and interpretation, and how these reasoning mechanisms can explore QoC information for better results. Indeed, it is worth noting that interpretation mechanisms may consume both context data and QoC data. Context being dynamic and uncertain, context data and their quality indicators will evolve over time, making possible different historical and statistical analysis (and then interpretation). For instance, the analysis of outliers and weak signals on context values and on QoC indicators (e.g. means, accuracy, precision, etc.) may contribute to predicting new tendencies and anticipating actions. Weak signals are usually seen as abnormal values or "information on potential change of a system to an unknown direction" [39]. The analysis of such signals can be particularly interesting in some scenarios, revealing undiscovered events or phenomena. In the CC-Sem scenario (cf. Section 2), the analysis of

 

weak signals and outliers may contribute to the energy consumption prevision, revealing new tendencies on energy consumption (for example the presence of a new device) or forecasting variation due to seasonal climate changes. Similarly, in the smart agriculture scenario, the analysis of weak signals on mean values of temperature and soil moisture sensors may help in identifying possible misfunctioning sensors.

Finally, with the growing distribution of software applications, the distribution of context information over multiple nodes is becoming a necessary. For instance, in IoT scenarios, context information is often transferred to distant servers or cloud platforms for data analysis. In these cases, quality indicators such as latency or packet loss can significantly affect the information reliability. The Diffusion dimension considers the challenges related to the distribution of the context information. When considering QoC on such distributed environments it is important to consider whether this transmission may affect the quality of transmitted context information. According to [5], it is necessary to consider the quality of both the exchanged context data and the distribution process to ensure user satisfaction. Indeed, if the context data distribution is not aware of the data quality, possible service reconfigurations could be misled by low quality data. For instance, real time context information may be affected by network latency and become out-of-date. When considering, for example, an application such as [18] that deploys its components on remote nodes according to available resources, if information about these resources is outdated because of network latency, deployment decision may lead to user dissatisfaction and performance loss. It is then necessary for application developers to consider not only how to guarantee QoC information transfer, but also how to *guarantee* that this diffusion of context information will not affect QoC?

As illustrated in Figure 3.4, quality concerns affect all aspects of context management and consequently all kinds of software applications using this notion. Both the context engineering roadmap represented in Figure 3.1 (cf. Section 3.1) and its quality concerns illustrated in Figure 3.4 offer a multi-dimensional view of the multiple aspects of context management in software applications. It is worth noting that, as the variety of examples given in this section leads us to suppose, every dimension proposed in the roadmap in Figures 3.1 and 3.4 will not equally influence all kinds of applications. The relevance of each dimension depends on the purpose of the application itself and on the considered context information. It is then essential to consider and discuss each dimension, considering its possible relevance for an application and the influence of the quality concern on it. In this section, we raise questions about the support of context information on software applications and about the influence of quality concerns on it. More than solutions, the main goal here is to initiate discussion and to point out challenges and issues of context management and QoC on software applications.

## 4. Literature review: challenges and solutions

The main goal of the context engineering roadmap presented in Section 3 is to help non-expert designers to better understand challenges and issues related to the support of the notion of context on software applications. Some of the challenges suggested here have already be highlighted by previous works in the literature. A good example of this is the context lifecycle proposed by [46] (see Figure 4.1), which considers the movement of context data on software applications, focusing on context modeling, acquisition, reasoning and dissemination. All these steps are considered in the roadmap as dimensions (respectively modeling, acquisition, interpretation and diffusion), to which we have added the purpose and the subject dimensions. Indeed, the context engineering roadmap intends to represent a summary of the main challenges highlighted in the literature, such as [1][42][7][6][25][14][5][37][46]. In this section, we introduce a brief summary of the literature review in which the roadmap is based. The main goal here is to discuss solutions and open issues proposed in the literature for each dimension of the roadmap. This summary is a necessary complement to the roadmap for non-expert designers, giving concrete examples of the challenges considered on each dimension.

 

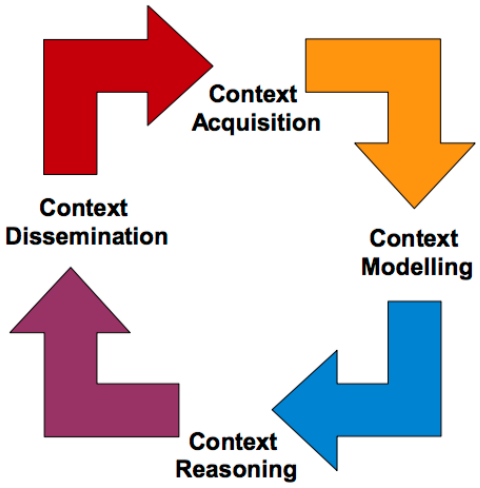


**Figure 4.1.** *Context life cycle proposed by [46]*

When analyzing the literature, a first class of applications advocating using the notion of context is represented by context-aware applications, which use context information for adaptation purposes. This adaptation may affect different aspects of the application behavior: one may adapt the content supplied to the user [56][15], the services offered by the application [58] [12], or even the application composition itself [49][18][22] according to the execution and the user's context. Still, as mentioned earlier, adaption is not the only purpose for using context information on software applications. In [14], the authors underline six common uses for context information: (i) *context display* (i.e. presenting context information to the user); (ii) *contextual augmentation* (i.e. annotating data with context information); (iii) *context-aware configuration* (i.e. configuring a service using context information); (iv) *context triggered actions* (i.e. triggering actions according to context information); (v) *contextual mediation* (i.e. modifying services and content to best match context of use); and (vi) *context-aware presentation* (i.e. adapting user interface or content presentation).

As one may observe, most of these uses refer to adapting application behavior (content, actions, services or interface) according to the context of use, but for [14], context information can also be used for annotation or simply displayed. Indeed, annotating data or objects using context information allows applications, services or even users to better characterize information or data, while displaying context may contribute to decision-making processes. For instance, [35] uses context information for characterizing fragments of methods on Method Engineering, while [47] [54] use context information for characterizing tasks on workflow models, and [33] associates context and group awareness information on Groupware Systems for helping users to better coordinate their actions. More recently, IoT applications are being considered for massively observing information from the environment, opening new perspectives for the use of this context information on many different applications. Works on IoT and smart cities [46][40][50][64], as well as projects such as CC-Sem mentioned in Section 2, illustrate this tendency quite well. They demonstrate the interest of using context information as a support for decision making as well as for adaptation.

When considering subject dimension, the literature illustrates quite well the variability of the notion of context. From initial works on context-aware computing, which mostly consider the user's location and device as the main context elements [56] [10], up to works considering complex situations on their behavior [58] [18], multiple visions of what can be considered as context are given. First of all, multiple definitions can be found in the literature [42]. For instance, [56] defines context as the location, the identity of nearby people and objects, and changes in those objects. Through this definition, these authors focus on three main questions: where, who is around and the surrounding resources. For [41], context refers to physical and social situation in which computational devices are embedded. This definition focuses on computational devices, delimitating this notion to its software perception, without considering precisely possible elements, keeping yet its generality. Currently, one

 

of the most accepted definitions is given by [21], which considers context as “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. Similar to [41], [21] examines the notion of context from a software background, by focusing on the interaction between a user and a system.

As one may observe from these definitions, the notion of context may cover several different elements. For instance, in [56][15][10], location is mainly considered, while in [22] [24] information about the execution environment, such as available memoire and network connection, is considered. In [49] [18], information about the executing device but also about surrounding devices is considered as context and used for better identifying the user’s situation and deploying application over these devices consequently. On [51], data obtained from a smartphone accelerometer is used for activity detection, while in [48], location detected using GPS and GSM data, as well as connectivity information, is used for adapting content delivery on a health record application.

A similar question arises when considering QoC. Considering quality of context on a software application means considering what indicators or properties to use for measuring such quality. Several authors have highlighted different indicators for context information, as demonstrated by [37]. Among common indicators, we may cite precision, up-dateness, accuracy or freshness.

There is a consensus in the literature about the arduousness of defining what context elements or QoC indicators can be considered as relevant for a given application (e.g. [21][2][25][37]). This question of choosing elements remains an open issue. Even if cited works do not tackle this issue of selecting context information and QoC indicators directly, some of them [24][22][27][13][37] propose interesting solutions for supporting these elements during the design phases. For instance, [24] and [13] propose a MDE (Model Driven Engineering) approach for developing context-aware applications. Both properties corresponding to the observed context elements (for the first) or to the QoC indicators (for the later) are modeled using UML and other high-level model representations at the very beginning of the design process. By focusing on this modeling, these approaches help designers to better conceptualize the necessary elements for their software applications. On the one hand, freeing designers from implementation details at the very beginning of the project allows those to focus on concepts and then to consider more easily the necessary information for their applications. On the other hand, by producing code from these model specifications, such approaches help designers in the development task itself.

Furthermore, the literature also reveals other issues that follow from identifying relevant context information. The first one concerns the relationship among the observed elements. Context elements are not necessarily independent, and their relationship can also be relevant. This is notably the case of cooperative applications, in which group members, tasks and objects can be related in different ways. For instance, [33] considers that knowing that an activity is related to a group can be as important as knowing the group itself. These authors explore information about these relationships in order to characterize the relevance of given information to a user. In [23], these relationships are explored through rules, allowing reasoning about access rights on shared resources according to the user’s context. Another issue is the granularity of the observed information. Some context elements can be broken down into lower-level elements or regrouped forming higher-level elements. Managing different levels of abstraction can be required by complex applications. For instance, in [18], the authors propose to aggregate low level information in order to describe a complex user’s situations. Similarly, in [45], the authors propose a pluggable architecture that allows new context information to be composed based on lower level data. In both cases, this composition of more complex information based on lower-level context data can also be assimilated into the interpretation of raw data for producing new context information.

 

Another common topic in the literature concerns context modeling. Several works have tackled this question, proposing many different context models [42][6][7]. All these works highlight several challenges related to context modeling, and notably how to deal with the heterogeneity and dynamicity of context information, as well as with the uncertainty that characterizes context information. Indeed, QoC concerns strongly affect context modeling, since those models should include quality information in order to handle QoC concerns on application behavior. As suggested by [14] or [27], quality information should be part of the context model and cannot be dissociated from it, in a holistic view of context modeling. Furthermore, context modeling plays an important role in the application extensibility and evolution. Context models contribute to isolating application behavior from the acquisition technologies. By doing so, these models contribute to the possibility of evolution, allowing new context information to be easily considered in the application without demanding large recoding efforts from the developers.

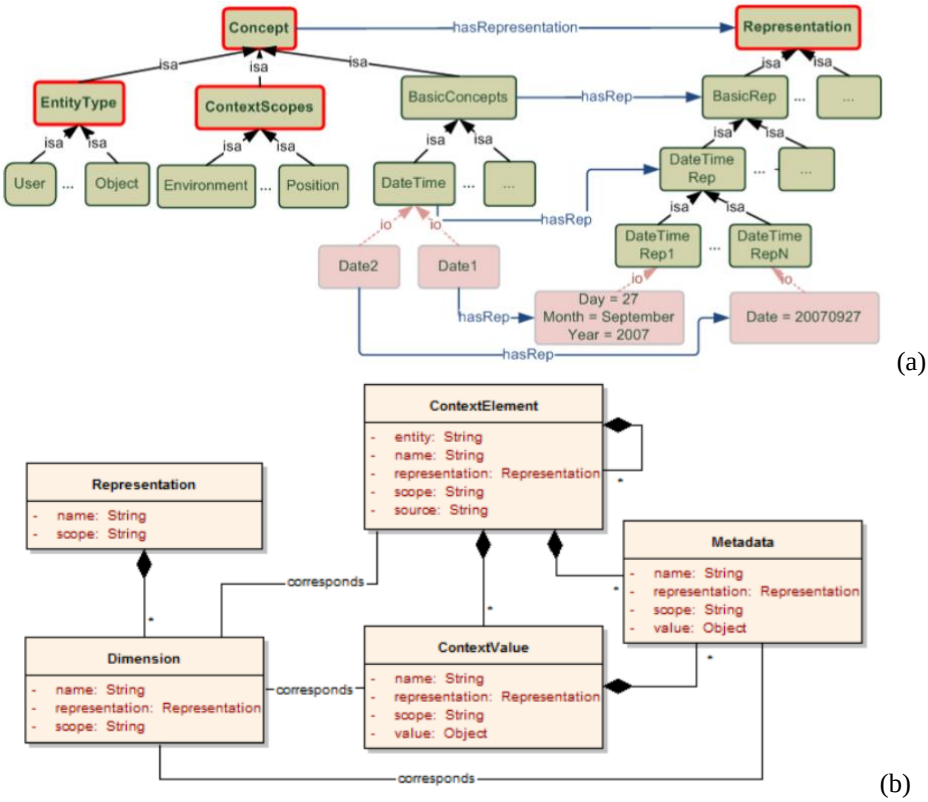


**Figure 4.2.** *Object oriented context meta-model (a) and context ontology (b) proposed on [62].*

As underlined by [42][6], different kinds of context models have been proposed in the literature. They vary from simple "key-value" models up to complex ontologies (e.g. [54]), passing by structured models (XML or RDF based, for instance [36]) and object-oriented models (e.g. [33]). These multiple modeling paradigms involve different degrees of complexity, both in implementation and execution. They also offer different reasoning capabilities, which may include ad-hoc processing, statistical and data analysis techniques or complex rule-based reasoning. For instance, in [33] an object-oriented model allows the representation of different information from a user's physical and organizational context. Organizational elements are also considered by [54], which use an ontology-based model for representing context information (time, location, availability, etc.) of entities related to a business process (actors, resources, etc.). In [62], different paradigms are combined. First, a context ontology, illustrated in Figure 4.2a, allows significant concepts related to context information to be described, including context entities (the user, an object, etc.), information elements about it (e.g. location) and their representation. Then, an object-oriented meta-model, resuming the same concepts, is used mainly for implementation purposes (see Figure 4.2b). Finally, an XML schema representing these concepts is used for exchanging context information among different computing nodes.

 

Modeling is also an important concern when considering QoC. In [14], a context model is proposed considering in particular the uncertainty of context information. This is performed notably through the definition of relations allowing context information to be compared under uncertainty. In [27], authors analyze the effects of quality on a context model. They consider a MDE approach for context modeling, proposing a DSL (Domain Specific Language) for creating context models. Through this language, it is possible to describe different context elements and its data sources, as well as more complex situations combining different context elements. Similarly, [13] [37] have also considered a MDE approach, focusing particularly on the modeling and support of QoC indicators. In [37], the authors have proposed a QoC meta-model in which quality indicators are associated with context information. Each QoC indicator has a set of associated values and it is defined by a QoC criterion containing a set of defined metrics, allowing a full definition of each QoC indicator, from the concept definition to its metrics.

As mentioned before, context models play an important role in isolating the application's behavior from the technology used for acquiring context information. By doing so, context models also contribute to hiding the heterogeneity of the acquisition devices. Indeed, many different devices can be used for acquiring a single context element. Maybe the most common example of this is location information, which can be observed using different devices and methods (GPS, GSM-based estimation, Wi-Fi triangulation, etc.), but the same can be said about other context elements. For instance, temperature can be observed using several different models of sensors. Not all these sensors necessarily offer the same support according to the programming language, libraries and platforms. As an illustration, the website HomeAutomation.org[4] compares about 5 different temperature sensors for Arduino, with different C programming codes for accessing each one. Code and process necessary to capture data from sensors is not necessarily the same according to the sensor model. This heterogeneity makes the acquisition process more complex and can seriously compromise the interoperability among acquisition devices.

Several research works have considered this issue, proposing mechanisms for isolating applications from the heterogeneity of the environment and consequently allowing a better interoperability among different acquisition devices. Among these, we may cite the Context Toolkit proposed by [21]. This toolkit isolates the application itself from the acquiring technology through different abstractions, and notably the notion of the context widget, which encapsulates the access to the physical device. The application only has to handle these abstractions, no direct knowledge about the physical device is necessary. This knowledge is concentrated inside each context widget, which offers a standard interface for the application, improving interoperability, from the application point of view. A similar approach is assumed in [45], which proposes a pluggable architecture in which context plugins are dynamically loaded according to the application needs. Again, applications are not directly faced with the real physical environment, keeping contact only with their context plugins, which provide a standard interface for accessing context information. All the interaction with the real environment is confined in the context plugin, reducing the application's complexity and improving interoperability and possibilities of code reuse.

Less discussed in the literature but still important, the acquisition of quality information suffers from a similar issue. Indeed, not all sensors offer the necessary information or metrics for quality indicators, and often software designers have to consider other means to calculate or estimate such indicators. For instance, considering temperature sensors like those mentioned by HomeAutomation.org[3], these sensors may easily be influenced by external factors (e.g. exposition to sunlight or to heating/cold source), which may lead to significant errors in acquired data. In the experiment we performed with the students (cf. Section 2.2), we used five temperature sensors of the model I2C BMP 280 in a single room (about 35 $m^2$) and obtained up to 2~3°C of difference in the perceived temperature among the

[4] http://www.homautomation.org/2014/02/18/arduino-temperature-sensor-comparison/

 

sensors. Unfortunately, sensors like BMP 280 do not offer natively meta-data or any quality information allowing an application to automatically calculate QoC indicators. Calculating QoC indicators such as precision demands, in this case, extra equipment or processes for calibration. This issue of how to obtain or estimate QoC indicators remains until now an open question left in charge of the application designer.

The same can be affirmed about the observation process. This process as well as the QoC considerations concerning it have not received the same attention in the literature as other dimensions considered in the roadmap. One of the reasons motivating this could be the dependency between this process and the application itself. Indeed, this process greatly depends on the application needs considering the context information and its freshness or up-dateness. Nonetheless, most of the middleware solutions for context management, such as [22][45][24], offer push (a.k.a. publish-subscribe) and pull mechanisms for capturing context data from the environment. Such mechanisms represent the basis for successful observation policies. Nevertheless, the definition of these policies is left under the responsibility of the application designers, according to the application needs.

Furthermore, one of the main challenges of context-aware applications is the dynamicity of pervasive environments in which these applications are supposed to execute. This is the main aspect analyzed by the management dimension. By its own nature, context data is in constant evolution. Context is a dynamic construct as viewed over a period of time, episodes of use, social interaction, internal goals, and local influences [25]. Context is not simply the state of a predefined environment with a fixed set of interaction resources. It is part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources [17]. Dynamicity is intrinsic to the notion of context and it should be taken into account properly when considering it on a software application. It should be considered through appropriate models and acquisition mechanisms, but also in the management of the surrounding environment. Indeed, this environment cannot be supposed to be static. By definition, context-aware applications must consider a dynamic environment, in which execution conditions may vary, users and devices may move, resources may come in or disappear at any moment. Such a dynamic environment leads to multiple challenges and notably service and resource discovery. As resources move or change their current state, the composition of the surrounding environment and its available resources also change, making the ability of discovering and managing such surrounding resources a necessity. Multiple works have considered this issue, such as [52] which proposes a dynamic binding and component discovery mechanism for service component architectures. Moreover, this question also has be considered on other domains, such as the grid systems [43], proposing interesting solutions that could also be applied for handling the dynamicity of pervasive environments.

The dynamicity of pervasive environments also impacts the diffusion of context information. Diffusion of context information to other nodes is often necessary mainly when considering adapting applications to the surrounding available resources, such as [18], in which context information about surrounding nodes is used for adapting application deployment. As underlined by [5], multiple approaches for distributing context information have been considered, including centralized approaches, peer-to-peer and hybrid or hierarchical ones. We may cite, for instance, [59] and [19]. The former proposes to organize the distribution of context information on dynamic groups, which regroup nodes presenting a similar situation. Groups are established dynamically based on distribution policies, which indicate context information that can be shared among group members and the common context elements that define group membership. The latter propose a hierarchical solution based on a SIP communication protocol for sharing context information among members of a community.

Most of these context distribution approaches are now faced with an IoT environment, which imposes an important requirement: scalability. Indeed, in an IoT environment, the number of pieces of equipment may grow exponentially [44], demanding distribution mechanisms to be able to scale up. Many of the approaches underlined by [5] are unable to scale up and potentially handle hundreds of

 

nodes, such as considered in IoT scenarios. Hybrid architectures, using hierarchical approaches, such as [53] or architectures combining IoT and cloud infrastructures such as [40] [61], are increasingly being considered.

The use of cloud computing infrastructures leads us to consider the persistence of context information and its access control policies. Unfortunately, these questions remain marginal in the literature compared with their importance for a user's privacy. One possible reason explaining this is the fact that context information is often supposed to be consumed in "real time": it is the context of a given entity (user, object, service, etc.) at this particular moment. In this case, storage and historical analysis are not necessarily taken into account. However, this is about to change with the massive adoption of cloud-based solutions for storage and the application of data mining techniques for context analysis, such as in [51] [29]. The question of security and access control for context information will also become more and more relevant in the next few years. Nowadays the number of works considering these issues is not proportional to their relevance. Only a few works have, for instance, focused on how to control the access to context information produced by a given node. For instance, [20] proposes access policies based on the XACML standard using a FOAF (Friend Of A Friend) approach. We hope that, with the development of IoT technologies, these questions will in the near future receive the attention they need in the literature.

Finally, the interpretation dimension has been considered in the literature through several different approaches, according to the application purposes. Indeed, as mentioned in Section 3, context data is often acquired as raw data that must be interpreted to become information or knowledge. Different ways to proceed to such an interpretation are possible, such as the possibility to aggregate low level data on more complex context information. The previously mentioned context plugins proposed by [45] represent an interesting mechanism for aggregating context data in a transparent way: the application may handle aggregated context information in the same way as it handles raw data through these context plugins.

More sophisticated interpretation mechanisms are also possible. For instance, [33] and [58] consider similarity measures for analyzing and comparing context information, while [23] considers rule-based systems for deducing new information from context information. In all these cases we may observe the relationship between interpretation mechanisms and context models: none of these proposals would be applied without an appropriate context model. For instance, rules proposed by [23] are possible thanks to the ontology-based model adopted by these authors. Other complex mechanisms, not necessarily based on ontology-based models, can also be cited. In [18] a workflow mechanism is used in order to deduce complex situations from the context data, while in [55] context data is used with constraint programming in order to control application self-adaptation according to environment changes.

Furthermore, a new tendency towards interpretation can be observed: the mining of context information. The idea is to apply data mining techniques on context information for different purposes: to discover missing information [51][57], to anticipate context evolution [38], or to determine the relevance of a context element on a given system [29]. For instance, [51] considers different statistical methods, and notably Bayesian networks, for analyzing accelerometer and gyroscope data and identifying a user's movement related situation (e.g. walking, running, etc.). In [38], the authors propose using Markov chains in order to deduce the next context information and thus to anticipate a user's possible situation. Finally, [29] analyzes the relevance of context elements in the use of a given application by crossing context and application use data using Formal Concept Analysis (FCA). All these mining techniques may contribute to context-aware applications by allowing them to assume a more predictive behavior, anticipating pervasive environment evolution.

All the works cited in this section contribute somehow to the support and management of context information in software applications. They illustrate the challenges and possible solutions for different issues considered by the roadmap dimensions. Through this literature review we intend to contribute to

 

a better understanding of these challenges, complementing the roadmap dimensions by concrete examples of these dimensions in action. Table 4.1 summarizes what has been discussed here, associating dimensions identified in the context roadmap with their key concepts and the main questions that raise these dimensions, as well as examples from the literature review.

| Dimension | Key concepts | Questions | Examples |
|---|---|---|---|
| **Purpose** | Adaptation<br>Quality of Context (QoC) | Why use context information? What is the purpose of using this information in the application?<br>How will this information be explored?<br>Why should the application follow QoC indicators?<br>Is QoC relevant enough for the application? | [11] [12] [15] [18] [22] [49] [56] [58]<br>[33] [50]<br>[35] [47] [54] |
| **Subject** | Context information<br>QoC indicator | What information is considered context?<br>How can we identify it?<br>What information is needed for the application?<br>What quality indicators can be used? | [2] [3]<br>[10] [48] [51] [56] [37] |
| **Model** | Context model<br>QoC metric & models | How can we internally represent context and QoC data?<br>How can we structure this data?<br>How can we handle heterogeneity, dynamicity and uncertainty in this representation? | [6] [24] [33] [36] [54] [62]<br>[13] [14] [27] [37] |
| **Acquisition** | Sensor devices<br>Acquiring platform<br>QoC measurement | How can we acquire context and QoC data?<br>What acquisition devices can be used?<br>How and how often should data be collected?<br>How can we manage the environment and its devices?<br>How can we support devices' heterogeneity?<br>Are QoC indicators supported by the devices? | [21] [18] [22] [45] [52] |
| **Interpretation** | Reasoning<br>Context mining | How can we interpret context and QoC data?<br>How can we produce new context and QoC data from low level data?<br>How can we apply a reasoning/analysis method on available data?<br>How can we take into account QoC during interpretation? | [14] [18] [23] [29] [38] [51] [55] [57] |
| **Diffusion** | Context distribution<br>Scalability | How can we transfer context and QoC data among nodes?<br>May this transfer affect QoC?<br>How can we guarantee the reliability of these data during transfer? How can we ensure data validity and coherence?<br>How can we manage scalability when the number of nodes grow?<br>How can we ensure privacy and data access policies? | [5] [19] [20] [44] [53] [59] |

**Table 4.1.** *Context management dimensions with their main questions and some related examples.*

## 5. Discussion & conclusions

The notion of context has been widely explored in different ways through software applications. This use is likely to grow in the next few years with the development of IoT technologies, which allow

applications to observe the physical environment using low cost devices. Nevertheless, the notion of context remains an obscure and ambiguous concept. What information can be considered as context, what information cannot, is a justified question for software developers. Information such as available memory, battery level or user's role can be considered as context for some [45][24][33], and as a simple application parameter for others [28][55]. This is also perceptible through some answers in the survey we performed (cf. Section 2.2). For example, a student that has declared having previous experience with "smart" applications and "mastering" the topic of context-aware computing assumed context in his/her application as "keywords" spoken by the user to "activate a feature". Clearly, an element that can be perceived as a simple input parameter and not necessarily as "context" by context-aware computing literature, is being considered as such in this case. Some authors, such as [11], tried to bring a distinction between context data and application data. For these authors, context data corresponds to a set of parameters, which are external to the application and that influence the behavior of the application. Despite the efforts to clarify this distinction, the boundary between context and application data remains blurred, as well as the notion of context itself, which remains often unclear for software developers.

There is an ambiguity not only on what can be considered as context, but also on what is a context-aware application and the possible distinctions between those and self-adapting applications. The smart agriculture and GridStix scenarios illustrate these ambiguities. In the former, context information can be easily seen as application data, while the latter was considered in [55] as a self-adapting application. According to [30], the concepts of context awareness and self-adaptation are often sources of confusion because self-adaptive applications often adapt their behavior based on the context stimuli and therefore it is often difficult to make a clear distinction between these two concepts. Indeed, both can be considered as adaptive systems, which, according to [16], aim to achieve a certain *goal* through the definition of some form of loop whereby the environment and/or the system itself is *monitored*, the information gathered is *analyzed*, a *decision* is taken as to what change is needed in response, and these changes are then *enacted* in the system. For these authors, 'self-awareness' means that changes can often be handled automatically compared with conventional systems that require off-line re-design, implementation and redeployment, which is also true for context-aware systems since they automatically adapt their behavior according to context changes without any particular human intervention. Some authors have tried to establish some distinction between these concepts [16] [30]. Nevertheless, the most important question here is not actually these potential differences (if they really exist), but the support of these systems. Both are very complex and poorly known concepts for non-expert designers. Both base their behavior on a very dynamic and complex kind of data. Whenever we call it context or not, the management of such dynamic data raise several challenges that should be considered when developing such a system. While these challenges remain misunderstood by non-expert designers, the possible distinctions between context-awareness and self-adapting, between context data and application data will remain irrelevant faced with these challenges.

The main question is not whether or not a given piece of information can be considered as context, but how to support and manage it in a software application. As pointed out by [17], it is commonly agreed that context is about evolving structured and shared information spaces, and that such spaces are designed to serve a particular purpose. Whatever information is considered as context profoundly depends on the application and on its purposes, and whatever this information could be, it is necessary to support it appropriately. This support requires understanding the challenges that it implies and the main characteristics of context information, such as its heterogeneity, its dynamicity and its uncertainty. The main goal of the roadmap discussed in this paper is precisely to contribute to this understanding.

The roadmap presented in this paper analyzed these different challenges of context management, organized through six dimensions. Each dimension has considered a precise aspect of context management and its quality concerns. Through this roadmap, we hope helping non-expert designers to better understand the challenges of supporting context information in software application. Indeed, in

 

our opinion, the main challenge now may not only be dealing with the remaining unsolved issues in this support, but perhaps it is about acquiring the necessary knowledge for developing new applications. Non-expert software developers, when developing context-aware applications, are faced with a very complex concept, the understanding and management of which is far from simple, as the multiple dimensions of the roadmap and the literature review we presented demonstrate. With the development of IoT and connected devices, and their integration into our everyday life, it is becoming essential to form a new generation of software developers that are able to reason about context support and its challenges, including quality concerns. More than just technical solutions (which are still necessary), we also need to go further towards context engineering approaches, offering a global approach to understand the context notion in software development. Raising questions and discussion about context management and the impact of Quality of Context on it is, for us, an important step towards a true context engineering approach.

Finally, we are strongly convinced that, as underlined by [17], context is key in the development of new services that will impact social inclusion for the emerging information society. More than ever it is important to encourage a better understanding of the notion of context and its support in young non-expert software designers in order to incite the development of new services and applications using this notion in a responsible way.

## Acknowledgments

The authors would like to thank the students from the MIAGE Sorbonne master's degree program for accepting to participate to our survey. We would like to thank Dr. Raul Mazzo, from Université Paris 1 Panthéon Sorbonne, for his inestimable help with the GridStix scenario and testing the roadmap, as well as Dr. Luiz Angelo Steffenel, from Université Reims Champagne-Ardenne, for his also inestimable help on the CC-Sem project and on the setup of the students' experiment.

## Bibliography

[1] BALDAULF M., DUSTDAR, S., ROSENBERG, F., « A survey on context-aware systems », *Int. J. of Ad Hoc and Ubiquitous Comp.*, vol. 2-4, 91-S46, p. 263-277, 2007.

[2] BAUER C., DEY A., "Considering context in the design of intelligent systems: Current practices and suggestions for improvement", *J. of Systems and Software*, vol. 112, p. 26-47, 2016, Elsevier.

[3] BAUER, J. S., NEWMAN, M. W., KIENTZ, J. A., "Thinking About Context: Design Practices for Information Architecture with Context-Aware Systems", *iConference 2014 Proceedings*, p. 398–411, 2014, doi:10.9776/14116.

[4] BELL G., DOURISH P., "Yesterday's tomorrows: notes on ubiquitous computing's dominant vision", *Personal and Ubiquitous Computing*, vol. 11, p. 133-143, 2007.

[5] BELLAVISTA, P., CORRADI, A., FANELLI, M., FOSCHINI, L., "A survey of context data distribution for mobile ubiquitous systems". *ACM Comput. Surv.*, vol. 45, p. 1–49, 2013.

[6] BETTINI, C., BRDICZKA, O., HENRICKSEN, K., INDULSKA, J., NICKLAS, D., RANGANATHAN, A., RI-BONI, D., "A survey of context modelling and reasoning techniques", *Pervasive and Mobile Computing*, vol. 6 issue 2, p. 161-180, 2010.

[7] BRÉZILLON, J., BRÉZILLON, P., "Context Modeling: Context as a Dressing of a Focus". In: Kokinov, B.; Richardson, D.; Roth-Berghofer, T. & Vieu, L. (Eds.), *Modeling and Using Context*, Springer, LNCS 4635, p. 136-149, 2007.

[8] BRÉZILLON, P., "Context-Based Development of Experience Bases". In: Brézillon, P.; Blackburn, P. & Dapoigny, R. (Eds.), *8th Int. and Interdisciplinary Conf. on Modeling and Using Context (CONTEXT)*, p. 87-100, 2013.

[9] BREZILLON, P., "Expliciter le contexte dans les objets communicants". In: Kintzig, C., Poulain, G., Privat, G., Favennec, P.-N. (Eds.), *Objets Communicants*, Hermes Science Publications, p. 293–303, 2002.

[10] BROWN, P.; BOVEY, J. & CHEN, X., "Context-aware applications: from the laboratory to the marketplace", *IEEE Personal Communications*, vol. 4 issue 5, p. 58-64, 1997.

 

[11] CHAARI, T., LAFOREST, F., CELENTANO, A., "Adaptation in context-aware pervasive information systems: the SECAS project", *Journal of Pervasive Computing and Communications*, vol. 3-4, 2007.

[12] CHAARI, T.; DEJENE, E.; LAFOREST, F.; SCUTURICI, V.-M. "Modeling and Using Context in Adapting Applications to Pervasive Environments". *IEEE Int. Conf. on Pervasive Services (ICPS'06)*, p. 111-120, 2006.

[13] CHABRIDON, S.; CONAN, D.; ABID, Z.; TACONET, C. "Building ubiquitous QoC-aware applications through model-driven software engineering". *Sci. Comput. Program.*, vol. 78, p. 1912–1929, 2013.

[14] CHALMERS, D.; DULAY, N.; SLOMAN, M. "Towards Reasoning About Context in the Presence of Uncertainty". *1st Int. workshop on advanced context modelling, reasoning and management*. Nottingham, UK, 2004.

[15] CHEVEREST, K.; MITCHELL, K.; DAVIES, N. "The role of adaptive hypermedia in a context-aware tourist guide". *Communication of ACM*, vol. 45, p.47–51, 2002.

[16] COLMAN, A., HUSSEIN, M., HAN J., KAPURUGE, M., "Context Aware and Adaptive Systems". In: Brézillon, P., Gonzalez, A. J. (Eds.), *Context in Computing*, Springer, p. 63-82, 2014.

[17] COUTAZ, J.; CROWLEY, J.; DOBSON, S., GARLAN, D., "Context is the key", *Communications of the ACM*, vol. 48 n° 3, p. 49-53, 2005.

[18] DA, K.; ROOSE, P.; DALMAU, M.; NEVADO, J.; KARCHOUD, R. "Kali2Much: a context middleware for autonomic adaptation-driven platform". *Proceedings of the 1st Workshop on Middleware for Context-Aware Applications in the IoT (M4IoT@Middleware 2014)*, p. 25–30, 2014.

[19] DEVLIC, A., & KLINTSKOG, E., "Context retrieval and distribution in a mobile distributed environment". In Proceedings of the *Third Workshop on Context Awareness for Proactive Systems (CAPS 2007)*. Guildford, UK. 2007.

[20] DEVLIC, A.; REICHLE, R.; WAGNER, M.; KIRSCH PINHEIRO, M.; VANROMPAY, Y.; BERBERS, Y., VALLA, M., "Context inference of users' social relationships and distributed policy management", *IEEE Int. Conf. on Pervasive Computing and Communications (PerCom 2009)*, p. 1-8, 2009.

[21] DEY, A., « Understanding and using context », *Personal and Ubiquitous Computing*, vol. 5 issue 1, p. 4-7, 2001.

[22] FLOCH, J., FRÀ, C., FRICKE, R., GEIHS, K., WAGNER, M., LORENZO, J., SOLADANA, E., MEHLHASE, S., PASPALLIS, N., RAHNAMA, H., RUIZ, P.A., SCHOLZ, U., "Playing MUSIC: building context-aware and self-adaptive mobile applications". *Softw.: Pract. Exp.*, vol. 43, p.359-388, 2013.

[23] GARCÍA, K.; KIRSCH-PINHEIRO, M.; MENDOZA, S.; DECOUCHANT, D. Ontology-Based Resource Discovery in Pervasive Collaborative Environments. In: Antunes, P., Gerosa, M.A., Sylvester, A., Vassileva, J., and de Vreede, G.-J. (eds.), *19th Int. Conf. on Collaboration and Technology (CRIWG 2013), LNCS 8224*. Springer, p. 233–240, 2013.

[24] GEIHS, K., REICHLE, R., WAGNER, M., KHAN, M.U., "Modeling of Context-Aware Self-Adaptive Applications in Ubiquitous and Service-Oriented Environments", In: B.H.C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee (Eds.), *Software Engineering for Self-Adaptive Systems, Lecture Notes in Computer Science, 5525*, p. 146-163, 2009.

[25] GREENBERG, S. "Context as a Dynamic Construct". *Human-Computer Interact.*, vol. 16 issue 2, p. 257–268, 2001.

[26] HAGRAS H., "Intelligent pervasive adaptation in shared spaces", In A. Ferscha (Ed.), *Pervasive Adaptation: Next generation pervasive computing research agenda*, p. 16-17, 2011.

[27] HOYOS J.R., PREUVENEERS D., GARCÍA-MOLINA J.J., "Quality Parameters as Modeling Language Abstractions for Context-Aware Applications: An AAL Case Study", In: Brézillon P., Turner R., Penco C. (eds), *Modeling and Using Context. CONTEXT 2017. Lecture Notes in Computer Science, vol. 10257*, p. 569-581, 2017.

[28] HUGHES, D.; GREENWOOD, P.; BLAIR, G.; COULSON, G.; GRACE, P.; PAPPENBERGER, F.; SMITH, P. & BEVEN, K., "An Experiment with Reflective Middleware to Support Grid-based Flood Monitoring", *Concurr. Comput.: Pract. Exper.*, John Wiley and Sons Ltd., vol. 20 issue 11, p. 1303-1316, 2008.

[29] JAFFAL, A., LE GRAND, B., KIRSCH PINHEIRO, M., "Refinement Strategies for Correlating Context and User Behavior in Pervasive Information Systems", *Int. Workshop on Big Data and Data Mining Challenges on IoT and Pervasive Systems (BigD2M 2015), 6th Int. Conf. on Ambient Systems, Networks and Technologies (ANT-2015)*, Procedia Computer Science, vol. 52, p.1040-1046, 2015.

[30] KHAN, M.U., "Unanticipated Dynamic Adaptation of Mobile Applications", PhD thesis (Doktor der Ingenieurwissenschaften), University of Kassel, German, 31 March 2010.

 

[31] KIRSCH-PINHEIRO, M., MAZO, R., SOUVEYET, C., SPROVIERI, D., "Requirements Analysis for Context-oriented Systems", *7th Int. Conf.on Ambient Systems, Networks and Technologies (ANT 2016), Procedia Computer Science*, vol. 83, p. 253-261, 2016.

[32] KIRSCH-PINHEIRO, M., SOUVEYET, C., "Quality on Context Engineering". In: Brézillon P., Turner R., Penco C. (Eds.) *Modeling and Using Context. CONTEXT 2017. Lecture Notes in Computer Science*, vol 10257, p. 432-439, 2017.

[33] KIRSCH-PINHEIRO, M., GENSEL, J., MARTIN, H. "Representing Context for an Adaptative Awareness Mechanism", In: de Vreede G.-J.; Guerrero L.A.; Raventos G.M. (Eds.), *X Int. Workshop on Groupware (CRIWG 2004), LNCS 3198*. Springer-Verlag, p. 339-34, 2004.

[34] KIRSCH-PINHEIRO, M., RYCHKOVA, I., "Dynamic Context Modeling for Agile Case Management", In: Y.T. Demey and H. Panetto (Eds.), *OTM 2013 Workshops, LNCS 8186*. Springer-Verlag, p. 144–154, 2013.

[35] KORNYSHOVA, E.; DENECKÈRE, R.; CLAUDEPIERRE, B. "Towards Method Component Contextualization". *IJISMD*, 2, p. 49–81, 2011.

[36] LEMLOUMA, T., LAYAÏDA, N., "Context-Aware Adaptation for Mobile Devices". *5th IEEE International Conference on Mobile Data Management (MDM 2004)*, p. 106-111, 2004.

[37] MARIE, P. DESPRATS, T., CHABRIDON, S., SIBILLA, M., "The QoCIM Framework: Concepts and Tools for Quality of Context Management". In: Brézillon, P. & Gonzalez, A. J. (Eds.), *Context in Computing: A Cross-Disciplinary Approach for Modeling the Real World*, Springer New York, p.155-172, 2014.

[38] MAYRHOFER, R., HARALD, R., & ALOIS, F., "Recognizing and predicting context by learning from user behavior". In. W. Schreiner, G. Kotsis, A. Ferscha, & K. Ibrahim (Ed.), *Int. Conf. on Advances in Mobile Multimedia (MoMM2003)*, p. 25–35, 2003.

[39] MENDONÇA, S., PINA E CUNHA, M., KAIVO-OJA, J., RUFF, F., "Wild Cards, Weak Signals and Organizational Improvisation". FEUNL Working Paper No. 432, 2003. Available at SSRN: https://ssrn.com/abstract=882123

[40] MIORANDI, D.; SICARI, S.; PELLEGRINI, F. D. & CHLAMTAC, I., "Internet of things: Vision, applications and research challenges", *Ad Hoc Networks*, vol. 10 issue 7, p. 1497-1516, 2012.

[41] MORAN, T., DOURISH, P., "Introduction to this special issue on context-aware computing", *Human-Computer Interaction*, vol. 16 issue 2-3, p. 87-95, 2001.

[42] NAJAR, S., SAIDANI, O., KIRSCH-PINHEIRO, M., SOUVEYET, C., NURCAN, S. "Semantic representation of context models: a framework for analyzing and understanding" In: J. M. Gomez-Perez, P. Haase, M. Tilly, and P. Warren (Eds), *Proceedings of the 1st Workshop on Context, information and ontologies (CIAO 09), European Semantic Web Conference (ESWC'2009)*, p. 1-10, 2009.

[43] NAVIMIPOUR, N.J., RAHMANI, A.M., NAVIN, A.H., HOSSEINZADEH, M. "Resource discovery mechanisms in grid systems: A survey". *J. Netw. Comput. Appl.*, vol. 41, p. 389–410, 2014.

[44] PARIDEL, K., YASAR, A., VANROMPAY, Y., PREUVENEERS, D., & BERBERS, Y., "Teamwork on the road: Efficient collaboration in VANETs with context-based grouping". *Proceedings of The Int. Conf. on Ambient Systems, Networks and Technologies (ANT-2011)*, Procedia Computer Science, vol. 5, p. 48-57, Elsevier, 2011.

[45] PASPALLIS, N., ROUVOY, R., BARONE, P., PAPADOPOULOS, G.A., ELIASSEN, F., MAMELLI, A. A Pluggable and Reconfigurable Architecture for a Context-Aware Enabling Middleware System. In: Meersman, R. and Tari, Z. (eds.) *On the Move to Meaningful Internet Systems: Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, LNCS 5331*. Springer, p. 553–570, 2008

[46] PERERA, C., ZASLAVSKY, A.B., CHRISTEN, P., GEORGAKOPOULOS, D., "Context Aware Computing for The Internet of Things: A Survey". *IEEE Communications Surveys & Tutorials*, vol. 16 issue 1, p. 414-454, 2014.

[47] PLOESSER, K., RECKER, J., ROSEMANN, M., "Supporting Context-Aware Process Design: Learnings from a Design Science Study", In: zur Muehlen, M. & Su, J. (Eds.), *Business Process Management Workshops, BPM 2010. Lecture Notes in Business Information Processing, vol 66.* Springer, Berlin, Heidelberg, p. 97-104, 2010.

[48] PREUVENEERS, D., NAQVI, N.Z., RAMAKRISHNAN, A., BERBERS,Y., JOOSEN, W., "Adaptive Dissemination for Mobile Electronic Health Record Applications with Proactive Situational Awareness". *49th Hawaii Int. Conf. on System Sciences* (HICSS 2016), p. 3229-3238, 2016.

[49] PREUVENEERS, D.; BERBERS, Y. "Context-driven migration and diffusion of pervasive services on the OSGi framework". *Int. J. Auton. Adapt. Commun. Syst.*, vol. 3, p. 3–22, 2010.

 

[50] PUGLISI, S., MOREIRA, Á. T., TORREGROSA, G. M., IGARTUA, MÓ. A., FORNÉ, J., " MobilitApp: Analysing Mobility Data of Citizens in the Metropolitan Area of Barcelona", In: Mandler, B.; Marquez-Barja, J.; Mitre Campista, M. E.; Cagáňová, D.; Chaouchi, H.; Zeadally, S.; Badra, M.; Giordano, S.; Fazio, M.; Somov, A. & Vieriu, R.-L. (Eds.), *Internet of Things: IoT Infrastructures, 2nd Int. Summit, IoT 360° 2015, Part I*, Springer, p.245-250, 2016.

[51] RAMAKRISHNAN, A.K., "Support for Data-driven Context Awareness in Smart Mobile and IoT Applications: Resource Efficient Probabilistic Models and a Quality-aware Middleware Architecture" (Ondersteuning voor data-gedreven context-bewustzijn in intelligente mobiele en IoT applicaties: Hulpbronnenefficiënte probabilistische modellen en een kwaliteit-aware middleware architectuur), PhD thesis, Katholieke Universiteit Leuven, Belgium 2016.

[52] ROMERO, D., ROUVOY, R., SEINTURIER, L., CARTON, P., "Service Discovery in Ubiquitous Feedback Control Loops". In: Eliassen, F., Kapitza, R. (Eds.), *10th IFIP WG 6.1 Int. Conf. on Distributed Applications and Interoperable Systems (DAIS) / Int. Federated Conf. on Distributed Computing Techniques (DisCoTec)*, Lecture Notes in Computer Science, vol. 6115, p.112-125, Springer, 2010.

[53] ROTTENBERG, S., LERICHE, S., TACONET, C., LECOCQ, C., DESPRATS, T., "MuSCa: A multiscale characterization framework for complex distributed systems", *2014 Federated Conference on Computer Science and Information Systems*, Warsaw, p. 1657-1665, 2014.

[54] SAIDANI, O., ROLLAND, C., NURCAN, S., "Towards a Generic Context Model for BPM". *48th Annual Hawaii International Conference on System Sciences (HICSS'2015)*, Jan 2015.

[55] SAWYER, P., MAZO, R., DIAZ, D., SALINESI, C., HUGHES, D., "Using Constraint Programming to Manage Configurations in Self-Adaptive Systems". *Computer*, vol. 45 issue 10, p. 56-63, 2012.

[56] SCHILIT, B.N., THEIMER, M.M. "Disseminating Active Map Information to Mobile Hosts", *Network*, vol. 8, IEEE, p. 22-32, 1994

[57] SIGG, S., HASELOFF, S., DAVID, K., "An alignment approach for context prediction tasks in ubicomp environments". *IEEE Pervasive Computing*, vol. 9, issue 4, p. 90–97, 2010.

[58] VANROMPAY, Y., KIRSCH-PINHEIRO, M., BERBERS, Y., "Service Selection with Uncertain Context Information". Reiff-Marganiec, S. & Tilly, M. (Eds.), *Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions*, p. 192-215, 2011.

[59] VANROMPAY, Y., KIRSCH PINHEIRO, M., BEN MUSTAPHA, N., AUFAURE, M.-A., "Context-Based Grouping and Recommendation in MANETs", In: Kolomvatsos, K.; Anagnostopoulos, C. & Hadjiefthymiades, S. (Eds.), *Intelligent Technologies and Techniques for Pervasive Computing*, Chapter 8, IGI Global, p. 157-178, 2013.

[60] VANROMPAY, Y., "Efficient Prediction of Future Context for Proactive Smart Systems", PhD thesis, Katholieke Universiteit Leuven, Belgium, 2011.

[61] VILLALBA, L., PREZ, J.L., CARRERA, D., PEDRINACI, C., PANZIERA, L., "Servioticy and iserve: A scalable platform for mining the IoT". *6th Int. Conf. on Ambient Systems, Networks and Technologies (ANT-2015)*, Procedia Computer Science, vol. 52, p.1022-1027, 2015.

[62] WAGNER, M., REICHLE, R., KHAN, M.U., GEIHS, K., LORENZO, J., VALLA, M., FRÀ, C., PASPALLIS, N., PAPADOPOULOS, G.A., "A Comprehensive Context Modeling Framework for Pervasive Computing Systems", *8th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2008), Lecture Notes on Computer Science, 5053*, p. 281-295, 2008.

[63] WEISER M., "The computer for the 21st century", *Scientific American*, vol. 265 issue 3, p. 66-75, 1991.

[64] ZAPPATORE, M., LONGO, A., BOCHICCHIO, M. A., ZAPPATORE, D., MORRONE, A. A., DE MITRI, G., "Towards Urban Mobile Sensing as a Service: An Experience from Southern Italy", In: Mandler, B.; Marquez-Barja, J.; Mitre Campista, M. E.; Cagáňová, D.; Chaouchi, H.; Zeadally, S.; Badra, M.; Giordano, S.; Fazio, M.; Somov, A. & Vieriu, R.-L. (Eds.), *Internet of Things: IoT Infrastructures, 2nd Int. Summit, IoT 360° 2015, Part I*, Springer, p. 377-387, 2016.

 