Requirements Analysis for Context-oriented Systems

Manuele Kirsch-Pinheiro, Raul Mazo, Carine Souveyet, Danillo Sprovieri

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2015) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Requirements Analysis for Context-oriented Systems

Manuele Kirsch-Pinheiro*, Raúl Mazo, Carine Souveyet, Danillo Sprovieri

Centre de Recherche en Informatique, Université Paris 1 Panthéon Sorbonne, 90 rue Tolbiac, Paris 75013, France

Abstract

Context-oriented systems are systems that observe and handle context information from the environment to guide their own behavior. Engineering such systems represents a complex task not only due to their complexity, but also due to the notion of context. Handling this notion involves tackling several challenges, demanding to system designers a certain knowledge and expertise about this notion. In order to help designers on this engineering process, we propose in this paper a roadmap on context management and a requirements elicitation process. This roadmap aims at sharing with non-expert designers the necessary expertise on context management allowing them to better understand the notion of context and its challenges. The elicitation process aims at guiding these non-expert designers across the roadmap, supporting them in their requirements elicitation process concerning context management. The proposal is presented on a running example that illustrates the approach.

Keywords: Pervasive computing, context-aware computing, Information Systems, context modelling

1. Introduction

The notion of context corresponds to a large concept that has been explored in many different ways and systems[1, 2,3,4]. These systems, which we can call "context-oriented systems", have reached different levels of maturity in the way they consider the notion of context. Context-oriented systems are often distributed and composed of embedded systems, control systems, real-time systems, physical systems, and network and communication systems. They are seen in various application domains including irrigation, telecommunication (networks and communication devices like sensors and smartphones) and online-shopping. These systems have in common the need to be capable to run continuously under changing conditions. For instance, the price/furnisher of components can change, the requirements and deployment conditions (temperature, bandwidth, etc.) permanently change, and partial failures of

* Corresponding author. Tel.: +33 1 44 07 86 34.
E-mail address: Manuele.Kirsch-Pinheiro@univ-paris1.fr

subsystems can arise. To compare with traditional systems, systems that are able to respond to context changes are more complex in terms of increasing modularity, functionality, integration and interoperability, growing importance and reliance on software, and increasing number of non-functional constraints (e.g., robustness, scalability).

The ability to manage the execution context of an application is largely required in various application domains. Qualities of systems such as *flexibility, dynamicity, modularity and extensibility,* are difficult to satisfy with *ad-hoc development* processes. Various research works exist for handling context and for separating the infrastructure from the application with context-oriented middleware. However, it is difficult to have a *global vision or a large expertise on context management* because these works come from various research domains, each of them promotes a specific view and treatment of the notion of context. It is then particularly difficult to identify the relevant requirements about context management for a specific system in various application domains by a non-expert.

The purpose of this paper are (1) to propose a roadmap on context management in order to share expertise on this topic and (2) to exploit it during the requirements elicitation process in order to support the identification of the relevant requirements on context management for a given system.

This paper is organised as follows: Section 2 introduces a running example that illustrates the complexity of engineering context-oriented systems. Section 3 exposes the roadmap. Section 4 shows its exploitation during the requirements elicitation process applied on the running example. Section 5 discusses results and conclusions.

2. Motivating example

The concepts presented in this paper are considered within the flood-warning scenario proposed by Hughes et al.[5]. That scenario, called GridStix, considers a Wireless Sensor Network (WSN) deployed on the Rivers Ribble and Dee in England and Wales, respectively, as shown in Fig. 1. The WSN is used for collecting data from the physical environment through a network of sensors that communicate by means of a wired telecommunication infrastructure and without access to fixed power supplier. Each GridStix node consists of depth and flow sensors where power for these devices 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, with a single GSM uplink node.

A stochastic model predicts flooding according to the information sensed by each active node. In addition to the different transmission and data collecting modes of each node, they can be activated or deactivated according to the power level of the corresponding battery and the state of the river. The model's accuracy is a function of the number and distribution of nodes contributing sensor data and of the resources committed to processing the data. WSN designers need to make assumptions about the QoS that will be achieved by deployed components. For example, "the terrain, weather and other factors affect radio signal propagation, which in turn can affect QoS properties such as resilience. This is a key point; that systems and their environment do not always behave as expected and self-adaptation is a means to tolerate the unexpected"[6].

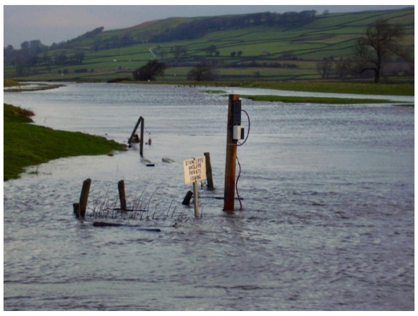

Fig. 1. A GridStix node next to the River Ribble, taken from Sawyer et al.[6]

For supporting dynamic adaptation of the configuration of the WSNs, each system should be aware of changes in its context and responds in a reactive and proactive manner. For engineering these kinds of systems it is necessary to specify adaptation policies that state the actions required to adapt the running system to a configuration that better fits not just its current context but sometimes the future context for the proactive adaptations. In order to do so, the occurrence of context changes (e.g. the low level of a battery) should be predicted to allow the system to change (e.g. change a node with a low battery by a neighboring node with a high level of battery) itself to better answer to the new context and prevent the occurrence of a predicted anomaly. Engineering systems like GridMix is then a challenge that heavily depends on context management. Different aspects should be considered, which represent a hard task for non-expert designers. The next section intends to tackle this challenge.

3. Contribution

The notion of context corresponds to a large notion explored by different kinds of systems. These systems, called here "*context-oriented systems*", are not limited to context-aware systems, which are systems capable of adapting their behavior to any change in their execution context[2,4]. We consider as "context-oriented systems" all kinds of systems that explore this notion in their execution behavior for different purposes. Due to the complex nature of the notion of context, engineering such systems can easily become a hard task since different aspects should be taken into account in that task. Although several surveys on context-aware computing exist[1,3,4,7,8,9], they focus often on this community and its experts, and not on non-experts designers from other communities. These are left alone for understanding and identifying the necessary concepts and components for building context-oriented applications. In order to help such non-expert designers to better understand challenges that raise from the notion of context, we propose in this paper a roadmap, which summarizes results from the literature (*e.g.*[1,4,7,8,9,10]) and from our own experience (*e.g.*[3,6,11,12,13,14,15]), and a requirements analysis process, which guides non-expert users on identifying requirements of these systems during the engineering process. Based on an extended review of literature, we could identify several dimensions that characterize context-oriented systems and the use they made of the notion of context. Each dimension reveals particular challenges and issues that should be considered when engineering such systems. The roadmap presented in Section 3.1 summarizes these dimensions in order to bring to a non-expert the necessary knowledge for this engineering process, while Section 3.2 describes a process for exploring this roadmap.

3.1. The context management roadmap

Engineering context-oriented systems implies to take into account different aspects involving the notion of context and its support. The roadmap we propose below aims at identifying such aspects and guiding the requirements elicitation process of such systems. By analyzing several existent works from different research areas, we could identify most relevant characteristics of the context management required by context-oriented systems and organize them according to six dimensions (Fig. 2): *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 systems.

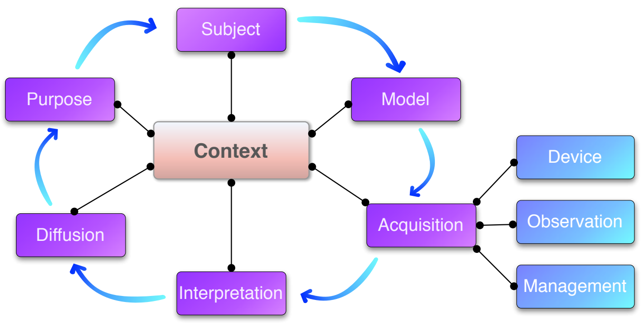

Fig. 2. The context management roadmap

The first dimension of the roadmap is the "*purpose*" dimension. It considers why a given system needs context information. A context-oriented system uses the notion of context for a precise purpose, which determines how this information is explored. Context-aware systems, for instance, use context information for adaptation purposes: adapting content supplied to the user[16,17]; adapting services[11,18] or the system composition[19,20] according to the execution conditions. Still, adaption is not the only purpose of context-oriented systems. Context information may also be used to characterize other information or for decision-making. For instance, Kornyshova et al.[21] use context information for characterizing fragments of methods on Method Engineering, and Kirsch-Pinheiro *et al.* [12] associate context and group awareness information on Groupware Systems for helping users to better coordinate their actions.

Once the "*purpose"* dimension is clearly identified, it is possible to consider what information precisely will be considered as a context. That is not a trivial question, since the notion of context corresponds to a large and often ambiguous concept[3,10]. The dimension "*subject*" aims at tackling this question.

Multiple definitions have been proposed for the notion of context[1,3,10]. The most accepted one 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 context-oriented system*[2]. This definition points out both an observed entity (*e.g.* the user) and information that is observed about this entity. The entity acts as a focus guiding the observation: the focus is the 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. We call here context element the information observed (location, memory, etc.) about a given entity.

What information about an entity could be considered as context remains an open question and it depends on the application[2,10]. The purpose of a context-oriented system determines what information will be observed. Different entities and context elements can be considered as relevant: information about the user (*e.g.* location), about a device (battery level, network connection…), etc. In addition to these commonly observed elements, other context elements can be considered, such as organizational elements characterizing an entity inside an organization or a group[21,12].

Identifying relevant context information raises other related issues. The first one concerns the relationship among the observed elements. Context elements are not necessarily independent and their relationship can be also relevant. It is notably the case of cooperative applications (*e.g.* knowing that an activity is related to a group can be as important as knowing the group itself[12]). Another issue is the granularity of the observed information. Some context elements can be decomposed on lower-levels elements or regrouped forming higher-level elements. Managing different levels of abstraction can be required by complex context-oriented systems (*e.g.* Da et al.[20]).

The "subject" dimension leads to a new question: how to represent observed context elements and their relationships? The dimension "*model*" intends to analyze this question. Its main goal is to determine the most appropriate representation for context information on a given system according to its identified purpose. An inappropriate model may compromise the implementation of the system. Several approaches of context modeling exist, from key-value sets and object oriented models up to complex ontologies[3,8]. Simple models, such as key-value ones, will be easy to implement but will offer no reasoning mechanism. In the opposite, 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. First of all, context information can be heterogeneous. Since different kinds of context elements can be observed, the information obtained may vary from numeric information, like GPS coordinates or a percentage (*e.g.* CPU load), up to symbolic values (*e.g.* the role of a user in a group). Context information is also uncertain and often incomplete[11,22], mainly due to problems during the acquisition of data (connection problems, interferences…), resulting on erroneous or missing data. This uncertainty represents an important issue since these data influence the behavior of a context-oriented system. Different approaches on quality of context[23] exist. They underline the importance of this issue when modeling context information. Finally, context information is naturally dynamic, and therefore varying among observations. This dynamicity must be supported by the context model, which should assume that values associated to context elements will vary on time.

The context model should be feed with values corresponding to the current state of observed entities and context elements. The dimension "*acquisition*" considers this question. The process of acquiring context information from entails three sub-questions, thus, three sub-dimensions: (i) how to observe the environment? (*device*); (ii) how to feed context model? (*observation*); (iii) how to manage dynamicity of the environment? (*management*).

The "*device*" dimension considers the devices used for observation purposes on a given system. The main challenge here concerns the heterogeneity, since the nature of the acquisition device can be quite variable. For instance, user's location can be acquired using a GPS device or calculated from a Wi-Fi connection; the information about the user's role on a team can be acquired from the Information System. These examples illustrate different natures of acquisition devices. Context-oriented systems must handle this heterogeneity and the interoperability among devices. Several researches[2,24] consider this issue, proposing mechanisms for isolating these systems from the heterogeneity of the environment and allowing a better interoperability among different observation devices.

The "*observation*" dimension considers how these devices will feed the context model. Different observation modalities can be considered according the observed information and its dynamicity. 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 will feed the context model, which needs to be kept updated in order to represent the current context of the observed entities. It is during the observation that all information needed for handling quality of context should be acquired and associated with observed values.

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 context-oriented system. The management of this dynamic environment is the goal of the "*management*" dimension, which considers the evolution of the environment and the availability of the acquisition devices on it. Several approaches on resource discovery exist[25] and can be considered on such environments.

Data collected during acquisition process correspond often to a raw data that should be aggregated or interpreted in order to be better exploited by context-oriented systems. The "*interpretation*" dimension focuses on this challenge, on how to transform raw context data on useful knowledge for context-oriented systems. Different interpretation mechanisms exist in the literature, such as rule-based reasoning based on ontologies[20,26]. The goal of the interpretation dimension is then to specify appropriate interpretation mechanisms and to consider necessary reasoning and aggregation mechanisms according to the capabilities of the context model. The interpretation cannot be dissociated from the context model. Not only the context model will constraint 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 also the information that will be deduced from the interpretation mechanism will also feed the context model, similar to an acquisition mechanism. For instance, the context plugins proposed by Paspallis *et al.*[24] allow deducing new information that is integrated to the context model as new observed value. The three dimensions, *model*, *acquisition* and *interpretation* are then intrinsically connected and cannot be dissociated.

Besides, a new tendency of 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[27], to anticipate context evolution[28], or to determine the relevance of a context element on a given system[13]. These mining techniques allow context-oriented systems to assume a predictive behavior, anticipating environment evolution.

Last but not least, the "*diffusion*" dimension considers the distribution of context information over a distributed set of nodes. Context-oriented systems are often distributed, 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, Da *et al.* [20] consider distributing context information about the nodes in order to better adapt application deployment to available resources. Different approaches exist for assuring the diffusion of context information: by proximity of nodes, by grouping nodes, etc.[9,14]. Several challenges arise from this distribution, above all, the stability of the context information (how long a given information remains valid and useful after being transferred from a different node?) and the coherence of the collected data are originated, since contradictory data can be reported from multiple nodes observing a given entity.

All these dimensions and their challenges should be considered when engineering a context-oriented system. The next section presents a process for guiding this engineering based on the proposed road map.

3.2. Engineering context-oriented system

On a context-oriented system, designers must also consider requirements related to context management issues. The process below uses the proposed roadmap for guiding the elicitation of these requirements. Starting from functional (FR) and non-functional (NFR) requirements of a system, this process guides non-expert designers across the roadmap for better understanding the system needs concerning context management.

The first step of this process (*Step 1* in Fig. 3) leads system designers to consider the purpose dimension. As explained in Section 3.1, a context-oriented system may explore the notion of context for different purposes. The first step for engineering this kind of system is then to clearly identify this purpose. After defining the purpose, *Step*

2 considers the *subject* dimension; designers must determine the entities and context elements that will be observed (*Step 2*). After identifying the entities and the relevant context elements to observe, a context representation model must be chosen to implement these concepts and their corresponding data (*Step 3- model* dimension).

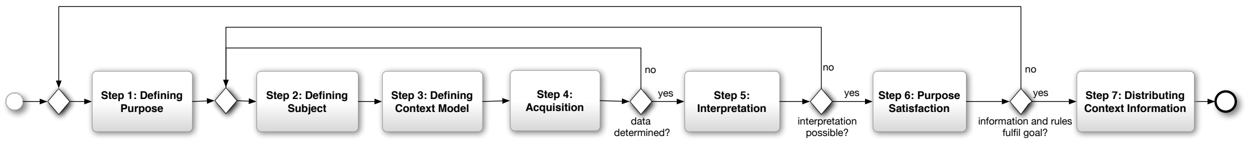

Fig. 3. Requirements analysis process of a context-oriented system.

Step 4 considers the *acquisition* dimension. Three aspects should be considered: the acquisition devices (*Step 4.1*), the observation (*Step 4.2*) and the device management (*Step 4.3*). In the first one, designers should define the devices that will be used for acquiring context information, while the second considers the observation frequency and quality of context criteria. In the third step, designers should consider the management of the observation devices and the dynamicity of the environment. Analyzing requirements on context acquisition may reveal new requirements on context modeling and subject. For instance, when considering quality of context (Step 4.2) or device management (Step 4.3), other context information may appear as necessary, forcing to reconsider Step 2.

Once context model and acquisition are fully defined, *Step 5* considers the *interpretation* dimension. Not all context-oriented systems will need interpretation capabilities. System designers must consider whether interpretation is needed and the different interpretation mechanism that could be used (aggregation of data, transformation, if-then rules, reasoning, etc.). According to requirements on interpretation, the context model can be eventually affected, since the chosen model should allow these interpretation requirements, which may lead back to Step 2.

Steps 1 up to 5 reveal the basic requirements on context management. These should contribute to the *purpose satisfaction*. *Step 6* considers whether the identified requirements permit to fully accomplish system purposes or if new elements are necessary. As shown in Fig. 3, the proposed elicitation process is an iterative process, involving multiple cycles. These cycles are due to the interdependency among the roadmap dimensions. The analysis of a given dimension may lead to new requirements on previously analyzed dimensions.

Finally, the last step (*Step 7*) considers the distribution of context information between different nodes (diffusion dimension). This step is particularly important on distributed systems in which context information acquired in a node should be made available. Next section illustrates the application of the proposed process on the running example introduced in Section 2.

4. Preliminary results

To show the relevance of our contribution, the process below is applied on our running example system (cf. Section 2). The main goal of this system is to *predict the flooding* of the river in order to help users to make decision and trigger actions according to the river state (***FR1***). The goal is combined with a non-functional requirement expressing that *the system must be energy-efficient* (***NFR1***). This flood monitoring system can be seen as a context-oriented system because it should observe in real-time the *depth* and the *flood rate* of river Ribble and according to their level it should alert the users and support them in their decision making. Following the process in Fig. 3, ***Step 1 defining the purpose*** leads to choose *decision making as purpose of the system.*

Step 2 defining subject considers the "*subject"* dimension, identifying the entities to observe and all information related to them. Our main entity is the river and more precisely its state. In order to determine the state of the river, we need to observe the *depth* and *flow rate* of the river at different points. The state of the river is classified in three categories: *normal*, *alert and emergency*. It is calculated by assembling depth and flow rate from ordered relevant points. Besides, the analysis of the subject allows determining that two levels of granularity must be managed in the context: the river level and the point level.

Step 3 defining the context model. According to the "model" dimension, several paradigms exist to implement the context model[3,8] (*key-value, semi-structured model, object orientation, graph orientation or ontology*). In order to choose the appropriate one, we have to consider several criteria such as: *dynamism, heterogeneity, generality or extensibility, quality and semantic richness*. Inspired by the framework proposed by Najar *et al.*[3] and by the analysis performed on multiple surveys[1,4,7,8], we could classify on Table 1 different context model paradigms considering these criteria, themselves resulting from the roadmap and its sources. According to the importance of these criteria for the system, Table 1helps to select the more appropriate paradigm to implement the context model. In the GridStix case, *dynamism* and *quality* are two main criteria and *generality & extensibility* are also useful if the control of the river intended to be extended to the *water quality***,** for instance. According to these criteria, the *object paradigm* seems appropriate to implement the context model for the GridStix case, combining the necessary criteria and the easiness of use and implementation.

Table 1. Classification matrix for context model paradigms.

Paradigm	Key Value	Semi-structured	Object-	Graph-	Ontology
Criteria		languages (e.g. xml)	Oriented	based	
Dynamism	+ +	- -	+ +	- -	- -
Heterogeneity		+ +	+ +		+ +
Generality & Extensibility	+	-	+	+	+ +
Quality	- -	+	+	-	+
Semantic Richness	- -	+	+ +	+	+ +

Step 4 acquisition analyzes the requirements about the infrastructure and the process of context acquisition. Step 4.1 deals with the device required to observe the *river* entity. In our case, GridStix sensor/activator is situated at each point of a river to capture the *flow rate* and *depth* parameters. In addition, these sensors will be organized through a network and communicate data among them by means of Bluetooth or Wi-Fi communication modes. *Step 4.2* leads to the identification of the observation frequency and the quality of context related data. Each observation of *flow rate* and *depth* follows a preconfigured frequency of time between two observations. This frequency is parameterized at the deployment according to each point of the river. Since the quality of the context is an important criterion for us, the model context needs to register especially the timestamp associated with *flow rate* and *depth* data. The support for decisions will be more accurate if the system keeps track of all data observed and calculated in the context management (*flow rate*, *depth* and *state*). This modification brings us back to *Step 2* to change the context model by adding the timestamp to each context information of a point of the river: *flow rate*, *depth* and *state*.

The last step of the "acquisition" dimension deals with the *device management* (*Step 4.3*). The availability and the dynamic of the observation infrastructure are important aspects of the GridStix case, since this infrastructure must be energy efficient (NFR1). Each sensor becomes an observed entity with relevant information such as: *battery level, Bluetooth status (on/off), Wi-Fi status (on/off) and general status (on/off)*. It is then necessary to return to Step 2 to extend the subject and the model with this new context information.

Step 5 interpretation aims at defining the interpretation rules. These rules will enable the derivation of new context information that is not observed by the acquisition infrastructure. This step leads to determine the computations of the energy state of each sensor (*low or high*), the state at each point and the whole river (*normal, alert* or *emergency*). ***Step 6 purpose satisfaction*** intends to verify that context information and rules are enough to fulfill the goals. The functional goal (FR1: predict flooding) can be satisfied with the current context and the interpretation rules. However, the non-functional requirement "*energy efficient*" (NFR1) cannot be satisfied because the rules for adapting the information transmission and the energy consumption of each sensor have not yet been defined. Therefore, ***Step1*** must be performed again to refine the purpose of our context-oriented system. The energy efficiency of the acquisition infrastructure leads to define a new purpose: self-adapting the acquisition infrastructure according to the energy consumption. It means a two-level architecture: one dedicated to the management of the acquisition infrastructure, and a higher level one dedicated to the flooding monitoring system. The first one manages the network of sensors and context information related to them, and the second one considers the context of the river including its overall level and the level of its observed points. Thus, returning to ***Step 2*** leads us to revise the context model and to verify that all information about each sensor and the network of sensors is enough to perform the adaptation rules required for monitoring the energy consumption and for the data transmission through the network. ***Step 4*** allows checking if this information about the sensors can be acquired. Besides, the sensor is also an activator on which an API is defined to get information but also to react by changing its status (general, Wi-Fi, Bluetooth). This API will be used on ***Step 5*** and on ***Step 6***, which determine the *interpretation rules* for the acquisition of the sensor context and *rules for adapting* each sensor according to the energy consumption.

Finally, ***Step** 7* considers distribution of context information in our system. The diffusion rule is then determined to adapt the transmission of context information between nodes in order to satisfy the *flooding prediction* (FR1), *and to adapt the acquisition* infrastructure to the energy consumption (NFR1).

The Fig. 4 resumes the architecture resulting from this elicitation process. In this architecture multiple elements of context management could be identified during the different iterations of the process (elements in green were identified in the first cycle, while elements in orange were defined during subsequent interactions). We can also observe two different levels of architecture using these context management features: one level handling the flooding prediction system (in green in Fig. 4), corresponding to the first requirements FR1, and a second level (in orange in Fig. 4) in charge of self-adapting the sensor network for energy saving (requirement NFR1).

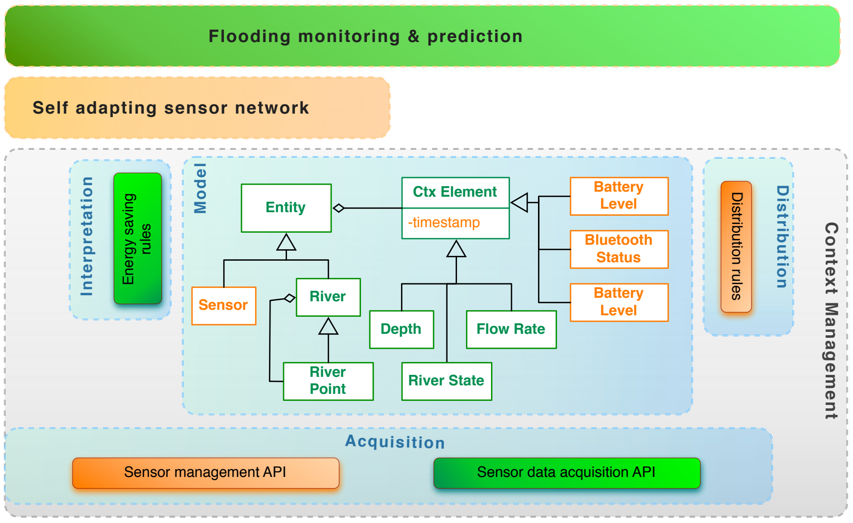

Fig. 4. Conceptual architecture resulting from the elicitation process.

5. Conclusion

This paper proposes a roadmap of context management challenges and issues for helping non-expert designers to better understand and elicit requirements of context-oriented systems. We present some advantages for supporting non-experts to define the architecture of context-oriented systems. By using the process of Fig. 3 for engineering GridStix, we obtained different results than those obtained in other studies. For example, the architectures proposed by Sawyer *et al.*[6] and Muñoz *et al.*[15] for the GridStix case are built in one block of requirements and restrictions that belongs to all different levels of the system. That vision contrasts to the one presented in this paper, which leads to the definition of a two-level architecture where each level has its own requirements and associated restrictions. Each level should consider their own context and the requirements about the expected dynamicity, modularity and extensibility that each level should guarantee in order to support the achievement of the requirements associated to the higher-level systems. Moreover, the separation between the concepts associated to the systems and the implementations of these concepts permit the definition of architectures that can easily evolve; while the physical structure of the sub-systems and the technologies used to implement them keep changing. The approach presented in this paper is a first step in that direction and we intend to continue its validation as part of our future work.

References

1. Brézillon, J.; Brézillon, P. Context Modeling: Context as a Dressing of a Focus. In: Kokinov, B., Richardson, D., Roth-Berghofer, T. & Vieu, L. (eds.). *8th Int. and Interdisciplinary Conference on Modeling and Using Context. LNCS 8175.* Springer Berlin / Heidelberg, 2007;136–149.
2. Dey, A. Understanding and using context. *Personal and Ubiquitous Computing,* 2001*;* **5**(1): 4-7.
3. Najar, S.; Saidani, O.; Kirsch-Pinheiro, M.; Souveyet, C. & Nurcan, S. Semantic representation of context models: a framework for analyzing and understanding. In: Gomez-Perez, J. M.; Haase, P.; Tilly, M. & Warren, P. (eds). *Proceedings of the 1st Workshop on Context, information and ontologies (CIAO 09), European Semantic Web Conference (ESWC'2009).* ACM; 2009. p. 1-10.

4. Baldauf, M.; Dustdar, S. & Rosenberg, F. A survey on context-aware systems. *Int. J. of Ad Hoc and Ubiquitous Comp.,* 2007; **2**(4):263-277
5. 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.,* 2008*;* **20**(11):1303-1316
6. Sawyer, P.; Mazo, R.; Diaz, D.; Salinesi, C. & Hughes, D. Using Constraint Programming to Manage Configurations in Self-Adaptive Systems. *Computer,* 2012*;* **45**(10): 56-63
7. Raychoudhury V, Cao J, Kumar M, Zhang D. Middleware for Pervasive Computing: A Survey. *Pervasive Mob. Comput.* Elsevier Science Publishers*,* 2013*;* **9**(2):177-200
8. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput*., 2010; **6**: 161–180.
9. Bellavista, P.; Corradi, A.; Fanelli, M.; Foschini, L. A survey of context data distribution for mobile ubiquitous systems. *ACM Comput. Surv*., 2013; **45**: 1–49.
10. S. Greenberg. Context as a Dynamic Construct. *Human-Computer Interact*., 2001; **16**(2): 257–268
11. Vanrompay, Y.; Kirsch-Pinheiro, M.; Berbers, Y.: Service Selection with Uncertain Context Information, In: Stephan Reiff-Marganiec and Marcel Tilly (eds.), Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions. IGI Global, 2011. p. 192-215.
12. 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.), *LNCS 3198 - X International Workshop on Groupware (CRIWG 2004)*. Springer-Verlag, 2004; p. 339--34
13. Jaffal, A.; Kirsch-Pinheiro, M.; Le Grand, B. Unified and Conceptual Context Analysis in Ubiquitous Environments. *The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2014*), 2014; pp. 48–55.
14. Kirsch-Pinheiro, M.; Vanrompay, Y.; Victor, K.; Berbers, Y.; Valla, M.; Frà, C.; Mamelli, A.; Barone, P.; Hu, X.; Devlic, A.; Panagiotou, G. Context Grouping Mechanism for Context Distribution in Ubiquitous Environments. In: Meersman, R. and Tari, Z. (eds.) *On the Move to Meaningful Internet Systems (OTM 2008), CoopIS, DOA, GADA, IS, and ODBASE 2008, LNCS 5331. Springer,* 2008; p. 571–588.
15. Muñoz, J. ; Tamura, G. ; Mazo, R. ; Salinesi, C. Towards a Requirements Specification Multi-View Framework for Self-Adaptive Systems. *CLEI electronic journal*, August 2015, **18**(2), Paper 5
16. Schilit, B.N.; Theimer, M.M. Disseminating Active Map Information to Mobile Hosts. *Network, IEEE*, 1994; **8**: 22–32.
17. Cheverest, K.; Mitchell, K.; Davies, N. The role of adaptive hypermedia in a context-aware tourist guide. *Commun. ACM*, 2002; **45**: 47–51.
18. Chaari, T.; Dejene, E.; Laforest, F.; Scuturici, V.-M. Modeling and Using Context in Adapting Applications to Pervasive Environments. *IEEE International Conference on Pervasive Services (ICPS'06*), 2006. p. 111-120.
19. Preuveneers, D.; Berbers, Y. Context-driven migration and diffusion of pervasive services on the OSGi framework. *Int. J. Auton. Adapt. Commun. Syst*., 2010; **3**: 3–22.
20. 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*), 2014; p. 25–30.
21. Kornyshova, E.; Deneckère, R.; Claudepierre, B. Towards Method Component Contextualization. *IJISMD*, 2011; **2**: 49–81.
22. Chalmers, D.; Dulay, N.; Sloman, M.: Towards Reasoning About Context in the Presence of Uncertainty. *1st international workshop on advanced context modelling, reasoning and management*, 2004. Nottingham, UK.
23. Chabridon, S.; Conan, D.; Abid, Z.; Taconet, C. Building ubiquitous QoC-aware applications through model-driven software engineering. *Sci. Comput. Program*., 2013; **78**: 1912–1929.
24. 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,* 2008; p. 553–570.
25. Navimipour, N.J.; Rahmani, A.M.; Navin, A.H.; Hosseinzadeh, M. Resource discovery mechanisms in grid systems: A survey. *J. Netw. Comput. Appl*., 2014; 41: 389–410.
26. 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, 2013; p. 233–240.
27. Ramakrishnan, A.; Preuveneers, D.; Berbers, Y. Enabling self-learning in dynamic and open IoT environments. In: Shakshuki, E. and Yasar, A. (eds.) *The 5th Int. Conf. on Ambient Systems, Networks and Technologies (ANT-2014), Procedia Computer Science*, 2014; **32**: 207–214.
28. Mayrhofer R. An Architecture for Context Prediction. PhD thesis, Johannes Kepler University of Linz. 2014.