Towards a novel analysis approach for collaborative ubiquitous systems

Nesrine Khabou, Ismael Bouassida Rodriguez

Towards a novel analysis approach for collaborative ubiquitous systems

Nesrine Khabou* and Ismail Bouassida Rodriguez†
*ReDCAD, University of Sfax, B.P. 1173, 3038 Sfax, Tunisia
Email: nesrine.khabou@redcad.org
†CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, LAAS, F-31400 Toulouse, France
ReDCAD, University of Sfax, B.P. 1173, 3038 Sfax, Tunisia
Email: bouassida@redcad.org

***Abstract*—In ubiquitous computing systems, applications must be able to respond to dynamic context changes in order to maintain collaboration between entities. A promising solution consists of developing context aware applications which automatically change their behavior according to the user needs, the available resources and the surrounding environment. Furthermore, a context aware application is characterized by a closed feed back loop with four phases: Monitoring, Analysis, Planning and Execution. In this paper, we focus on the second phase and we propose an analysis approach of collaborative ubiquitous systems which aims at analyzing context information and detecting significant abnormal changes. The proposed approach relies on thresholds in order to track down context changes. Once relevant context changes occur, the context aware application will be notified to trigger its suitable process dynamically in order to deal with the changes.**

***Keywords*-Ubiquitous computing, context awareness, adaptation, analysis techniques, tendency, peak.**

I. Introduction

Nowadays, the tremendous development of wireless communication technologies and the widespread of devices such as laptops, sensors, RFID tags, etc, have led to the appearance of ubiquitous computing that takes place as the successor of mobile computing systems. This new paradigm brings new challenges to the traditional applications. In fact, the deployment of the heterogeneous devices will certainly face an environment full of changes. These changes are related for example to the availability of mobile devices resources, the devices joining or leaving the network, the network topology varying, the changing execution environment (temperature, pressure, noise, etc) and the application execution context (user location, device screen size). Hence, the application needs to be able to detect these changes in order to adapt its behavior to these varieties according to the corresponding context information. This ability to sense and to detect the environment changes is called "context awareness". Moreover, context awareness [1] is considered to be the key issue for making devices aware of the situation of their users and their environment.

As a result, the design and the implementation of context aware applications on top of ubiquitous environments is a challenging task. First, these applications need to manage context continuously, including the collection of a multitude of context information from different sources such as sensors, operating systems, etc. Second, a classification of context information into different categories should be performed in order to facilitate the context use. Then, the context aware applications analyze and interpret context information so that being able to detect the conditions under which adaptation actions are required. Finally, the context aware application triggers adaptation actions when noticing context changes that can affect the application performance or functionalities.

In this paper, we propose a novel resource context classification taking into account the resource evolution behavior. Then, we introduce an approach that allows a collaborative application to detect the context changes, trigger notifications when necessary and adapt its functionalities to the current context. Our approach is based on thresholds in order to track down context changes. Moreover, thresholds are configured by domain experts or application designers and notifications are then raised by the corresponding entities in order to trigger appropriate reconfiguration actions.

The remainder of this paper is structured as follows: In section II, we introduce the context classification in ubiquitous environments followed by some research studies treating methods used for context changes detection in such environments. Section III presents the case study called "Entreprise hardware monitoring" that aims at monitoring resources state and raising appropriate notifications when resource violations are detected. In section IV, we introduce the proposed approach that allows for context classification and context changes analysis in collaborative ubiquitous systems using thresholds. In section V, we motivate and illustrate the feasibility of our approach through an illustrative scenario. The last section concludes the paper and gives some directions for future work.

II. Related Work

We begin this section by introducing some research studies dealing with context classification. Then, we give an overview of studies which propose techniques to detect context changes.

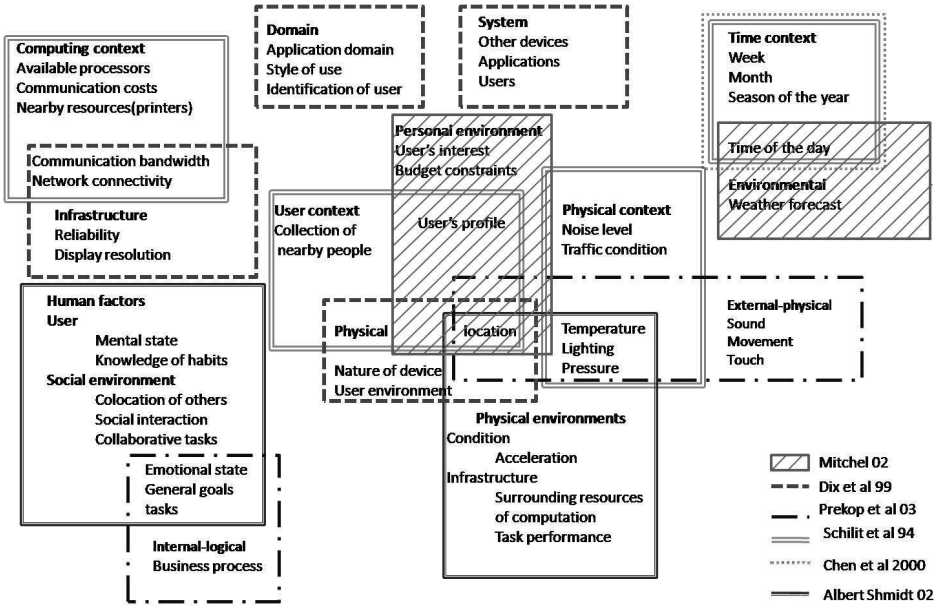


Figure 1. Context classification

A. Context and context classification

Dey et al [1] defined context as "any information that can be used to characterize the situation of an entity. An entity is a person, a place, or an object that is considered relevant to the interaction between a user and applications themselves". Due to the wide density of contextual data, many researchers have proposed several classifications of context into categories that are most suitable to their applications domain. These classifications are proposed in order to facilitate the context use. In fact, most of the studies made a classification into two categories [2],[3],[4]. On the one hand, Schmidt [3] partitions context into two classes, such as "Physical environment" and "Human factors" as depicted in the Figure 1. On the other hand, Mitchell [2] divides the context into two categories, a "personal environment" formed by the user's interest, the user's location and an "environmental context" formed by the weather forecast.

Other studies classify context to several categories like [5] and [6]. In fact, Schilit et al [6] define the context as the constantly changing execution environment. In fact, execution environment is the computing environment formed by the available processors, the network capacity, and the connectivity; the user environment which contains the user's location, its social situation. Finally, the physical environment. Brown et al [7] proposed to add a fourth context category as the time context defined by the time of a day, week, month and the season of the year.

B. Context changes detection techniques in ubiquitous environments

In the context changes detection research direction, several techniques and models are proposed in order to reveal the context changes. In fact, in [8], Cioara et al propose to use the context entropy concept for detecting the context changes and determining the degree of fulfilling a predefined set of policies. Moreover, context situation entropy defines the level of the system's self and execution envionment disorder which is measured by evaluating the degree of respecting a set of policies. Hence, once the context entropy exceeds a fixed threshold, then the system is in a critical state and it must execute adaptation actions. Although this approach allows self adaptation following context changes, it does not consider many context parameters to study as it is restricted to external parameters such as temperature, humidity and light etc.

In other studies, context changes are picked up by comparing a context value saved in a repository with a new context value. In fact, in [9], Zheng et al have addressed the issue of context change detection by proposing a context-aware middleware which conforms to the CORBA component model. The proposed middleware is composed of context aware services such as a context collector, a context interpreter, a context repository and a context analyzer. The latter is in charge of filtering and analyzing context information to determine relevant context changes and notifies the application afterwards. Furthermore, context filtering is based on a comparison of the context values saved in the context repository with the new context value in order to

detect context changes.

The proposed middleware enables to save the scare resources. In fact, the component deployment is performed "just-in-time". However, this middleware does not specify context information to take into account. Another approach for dynamic context management is proposed in [10]. Indeed, Taconet et al [10] present CA3M, a context aware middleware, which enables applications to adapt their behavior by dynamically taking into account context changes.

They model the application by "entities", which represent a physical or logical phenomenon (person, concept, etc.) and "observable", which defines something to observe. For instance, a mobile device state is an example of an observable which may take a finite number of values (e.g low battery, Almost low Battery or Normal Battery). They consider that the change of an "observable" state or even the observation goes past a given threshold from the last notified value leads to a different application behavior.

In [11], Bouassida et al proposed a model driven approach for collaborative ubiquitous systems. In order to detect context changes, they specify predefined thresholds. Then, once context values remain below/under the threshold values, a notification is raised. Although this approach enables to detect instantly context changes, it may cause false detections as well as missing alarms by using fixed thresholds.

III. Case study: Entreprise hardware monitoring

In order to illustrate the usability of our approach, we introduce an example of an M2M application named "Entreprise hardware monitoring" denoted in the Figure 2 which aims at enhancing security and detecting undesirable effects. In this case study, we focus on some context parameters such as battery level, available bandwidth and available memory such as the RAM.

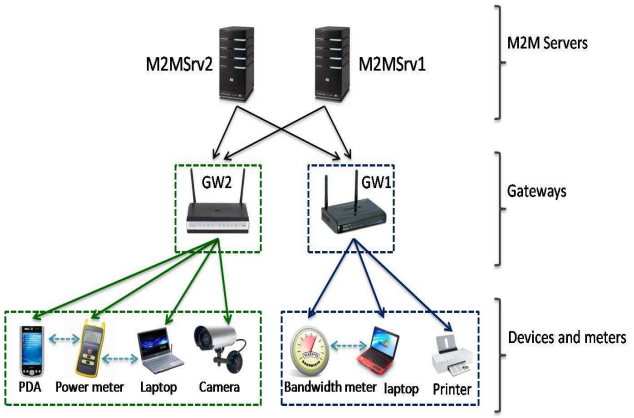


Figure 2. Entreprise monitoring use case

The application involves three kinds of participants as denoted in the Figure 2: Controlling M2M servers, M2MSrv1 and M2MSrv2, gateways such as GW1 and GW2, some meters used for monitoring and devices connected to the gateways. The M2M servers implement analysis algorithms, and process monitored data received from meters in order to analyze them. Once results are obtained, the corresponding M2M servers notify the device and/or act to reconfigure the architecture by switching or disabling the affected device(s).

For example, the M2M servers send requests to the power meter connected to the gateway GW1 in order to monitor the available battery level of some devices. Another bandwidth meter is used to calculate the connection speed between the different devices. In fact, the bandwidth meter is able to perform the bandwidth measurement. On the other hand, at the operating system level, a probe can measure the state of hardware resources such as the available RAM. Thus, the corresponding device can send a report to the appropriate M2M controlling server in order to notify it about its current state as well as its emergency degree.

IV. The proposed approach

We propose a distributed approach which aims at detecting context changes and raising notifications when context changes occur in order to adapt the application behavior accordingly.

Furthermore, in order to adapt an application to the changing context, the application should perform the following steps. It should collect information, analyzes contextual data and finally triggers the decided adaptation actions. Our approach depicted in Figure 3 is structured around this issue. In fact, it involves two main components: a component

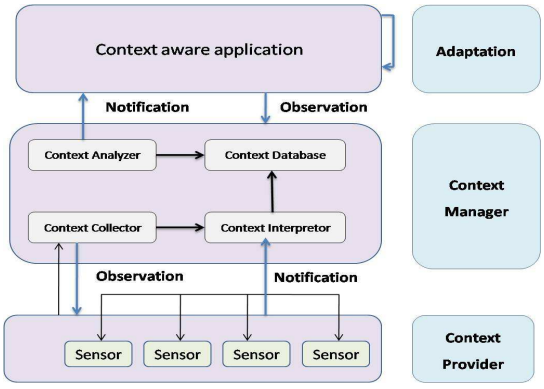


Figure 3. The proposed approach

named "Context Provider" responsible for providing context information. A second component, a "Context Manager" is in charge of managing context information. The activity of the "Context Manager" is divided into four large tasks. First, the context collecting from different sources. This task is performed by the "Context Collector". Second, the context interpretation which aims at processing context information and computing high level observations. Once achieved, processed context information are stored in a

"Context Database" which is used finally by the "Context Analyzer" in order to analyze stored context information and detect context changes.

In this paper, we focus on context information analysis performed by the "Context Analyzer". The "Context Analyzer" retrieves context information from the context database, analyzes it and tracks down the context changes using thresholds. Afterwards, the "Context Analyzer" notifies the application in order to execute the appropriate adaptation actions.

In our work, we consider computing context called resource context. Resource context constitutes the constraints imposed by the surrounding environment. For instance, the battery level, the available memory, the CPU load are examples of resource context.

A. Resource context classification

The diversity and the heterogeneity of devices and the network connectivity that ubiquitous environments provide, open the door to different resource context classification methods where context resources are combined in different ways to exploit.

A first classification divides resource context into two categories: Resource context related to the devices and resource context related to the network communication.

- ***Resource context related to the device***
  This category corresponds to the resources which characterize the device such as the available memory, the CPU frequency, the CPU load and the battery level.
- ***Resource context related to the network communication***
  This category deals with the resources that characterize the network communications as the network bandwidth, the network connectivity, the communication link load , the loss rate and the latency.

Despite the simplicity and the ease of this classification, it remains inappropriate to use. In fact, this classification doesn't take into account the resource behavior which is necessary especially in such systems that are characterized by an important evolution of the resource behavior. Second, we recall that our purpose is to define an approach that aims to detect context changes based on thresholds.

As a result, our idea consists on classifying resource context according to the resource evolution behavior.

This classification divides the resource context into two families: Resources whose behavior is characterized by a tendency and resources whose behavior is characterized by peaks. In the following, we present each family separately.

1) Resources with behavior characterized by a tendency: This family concerns resources whose model roughly coincides to a trendline as denoted in the Figure 4. In this class, we are interested to the tendency. This class includes resources whose behavior evolves (increase, decrease) in a constant way within time.

The battery level mentioned in the case study detailed previously is an example of this category. The available memory, the latency belong also to this class resource context classification.

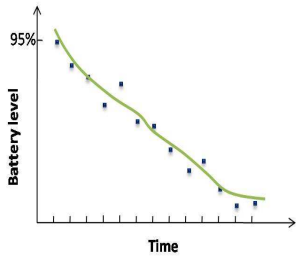


Figure 4. Battery level evolution within time

2) Resources with behavior characterized by peaks: In this category described by the Figure 5 the resource evolution behavior is characterized by abrupt changes called also peaks such as CPU load, link load, etc. The available bandwidth mentioned in the case study detailed before belongs to this family.

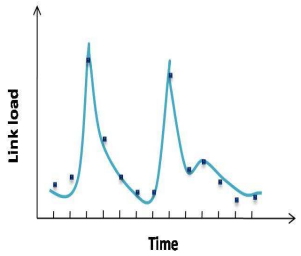


Figure 5. Latency evolution within time

B. Threshold calculation

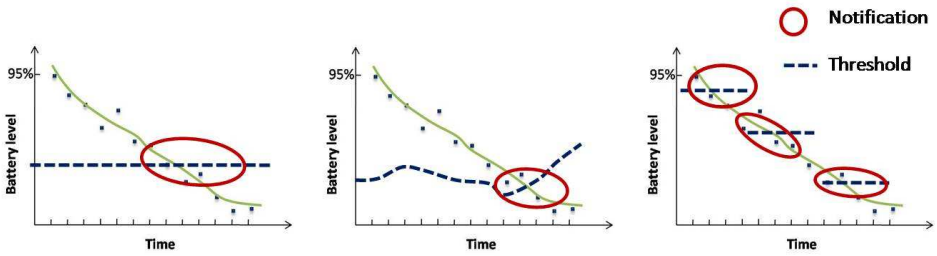
We have proposed a resource context classification which takes into account the resource evolution behavior.

Since we base our approach on thresholds for detecting context changes, we present the threshold calculation for each resource class.

We propose to attribute for each resource parameter belonging to a category, n thresholds and for each threshold, we assign a notification or a signal. This notification, defined by an expert, corresponds to an emergency degree or a need for adaptation that depends on the need of the expert or the application itself. The thresholds can be either predefined or adaptive ones.

Afterwards, we give some elements about threshold calculation for each category detailed previously.

1) Threshold calculation for the resources whose evolution behavior is characterized by a tendency: For this category, we remind that the resource evolution behavior is



(a) Fixed Threshold (b) Adaptive Threshold (c) Step Function Threshold

Figure 6. Threshold calculation for the resource whose evolution behavior is characterized by a tendency

described by a tendency. In order to avoid false detections as well as missing alarms, we need to define thresholds which are uncorrelated with the resource tendency evolution behavior. Thus, a notification is raised whenever the tendency curve crosses the threshold one.

Different kinds of thresholds can be applied for this class, such as fixed thresholds, adaptive thresholds and step function thresholds.

For instance, fixed thresholds may be defined by the application designer according to the resource characteristic. Then, a notification is raised once the resource crosses the threshold as depicted in the Figure 6(a).

For the adaptive threshold denoted in the Figure 6(b), mathematical methods can be applied in order to update threshold values at runtime such as the Exponential Weighted Moving Average technique used in [12]. However, for this kind of resource context characterized by a tendency, adaptive threshold must be uncorrelated with the resource evolution behavior in order to avoid false detections and missing alarms. Finally, for the step function threshold described in the Figure 6(c), thresholds are defined per period and notifications are raised when the resource behavior crosses the thresholds.

Let's consider the example of the battery level depicted in the previous section. In this example, whenever the battery level values remain under the threshold, a notification is raised. In the Figure 6(c), we use the thresholds modeled as a step function. Hence, we raise different level depending on the battery state. For example, in Figure 6(a), Figure 6(b), when the battery level values decrease until reaching a critical value under the threshold, an alarm is forwarded to the M2M server in order to switch the corresponding device.

2) Threshold calculation for the resources whose evolution behavior is characterized by peaks: In the second category of resource context whose evolution behavior is characterized by abrupt changes, specifying adaptive thresholds that are correlated with the resource evolution behavior tendency is an appropriate method for setting thresholds. Indeed, in this family, peaks characterize sudden changes form a normal behavior to an abnormal one. For that reason, adaptive thresholds correlated with the resource evolution behavior remain under the resource shape. Furthermore, a violation of the threshold reveals a context change.

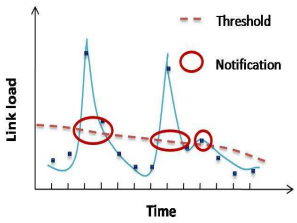


Figure 7. Threshold calculation for the resources whose evolution behavior is characterized by a peak

V. ILLUSTRATIVE SCENARIO

We illustrate the feasibility of our approach through a remote hardware maintenance application. This scenario shows how the application behavior is updated according to context changes and highlights the need for context modification at runtime. A computer company buys new devices such as laptops, desktop computers, PDAs, routers, etc. Furthermore, each new device embeds an analysis entity that implements our approach in order to detect resource context changes such as Battery level, available bandwidth, etc. Eric, the manager of the company decides to control daily the devices state as well as the communication links reliability. Bob is the employee selected to perform this work. To achieve this task, every day, Bob brings his PDA which contains a notebook application, equipped with a logical sensor, running a sender and a receiver program and connected to a power meter enabling him to monitor the battery state of his device as well as the other devices in the company. He starts then supervising the company hardware state. First of all, Bob broadcasts periodic messages through the network to monitor the devices state (ON, OFF), but also to check the network connectivity. Once a timestamp is elapsed without receiving a signal from the devices, in that case, Bob is notified that there is either lossy links or faulty devices or both. The bandwidth meter connected to Bob's PDA as depicted in the Figure 2 starts monitoring

the communication links. At each moment, Bob notes in its mobile device the message number through the corresponding link. By using the analysis approach, Bob decides that the link is overloaded, so that the bandwidth is in a critical state which engenders lossy links. Afterwards, Bob decides to monitor his mobile device state. So he starts by supervising the battery level and he initiates this task by applying threshold for this resource. In fact, since the battery level belongs to the first category detailed in section IV, Bob chooses three thresholds. Each one corresponds to an emergency degree: "Normal Battery", "Almost Low Battery" and "Low Battery". So when the power meter attached to his mobile device detects that the battery level of his mobile device reaches the "Low Battery" state, hence, the PDA switches to a poor mode omitting pictures and reducing contrast. After plugging his device, the PDA returns to its normal mode.

The scenario justifies the context awareness of the application in order to cope with different resources constraints and to adapt its behavior accordingly.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed an approach for the context changes detection in collaborative ubiquitous environments. Our approach is composed of two major components. A "Context Provider" and a "Context Manager". The "Context Provider" is out of the scope of this paper. In fact, we focus on the "Context Manager". The latter is divided into four components such as a "Context Collector", a "Context Interpreter", a "Context Database" and a "Context Analyzer". The "Context Analyzer" is the key component of our approach. Indeed, it is responsible for analyzing context information, detecting the context changes and forwarding notifications to the application when necessary. The "Context Analyzer" relies on thresholds in order to detect context changes. In our work, we have considered computing context to deal with as we have focused on resource context. Furthermore, we have presented a bi-classification of resource context according to the resource evolution behavior. A first category deals with resources whose behavior is modeled by a trendline. A second category concerns resources whose behavior is characterized by abrupt changes.

As future work, we plan to implement our analysis approach and to integrate it into the framework FACUS [13]. Then we intend to use ontologies in order to model context information.

ACKNOWLEDGMENT

This research is supported by the ITEA2's A2NETS (Autonomic Services in M2M Networks) project [1].

[1] https://a2nets.erve.vtt.fi/

REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, ser. HUC '99, 1999, pp. 304–307.

[2] K. Mitchell, "Supporting the development of mobile context-aware computing," Ph.D. dissertation, Lancaster University, 2002.

[3] A. Schmidt, "Ubiquitous computing - computing in context," PhD thesis, Doctor of Philosophy, Computing Department, Lancaster University,England, U.K., November, 2002.

[4] P. Prekop, "Activities, context and ubiquitous computing," *Computer Communications*, Mar. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0140-3664(02)00251-7

[5] T. Rodden, K. Chervest, N. Davies, and A. Dix, "Exploiting context in hci design for mobile systems," in *in Workshop on Human Computer Interaction with Mobile Devices*, 1998.

[6] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 85–90.

[7] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the marketplace," *Personal Communications, IEEE [see also IEEE Wireless Communications]*, no. 5, pp. 58–64.

[8] T. Cioara, I. Anghel, I. Salomie, M. Dinsoreanu, G. Copil, and D. Moldovan, "A self-adapting algorithm for context aware systems," in *Roedunet International Conference (RoEduNet), 2010 9th*, june 2010, pp. 374 –379.

[9] D. Zheng, J. Wang, W. Han, Y. Jia, and P. Zou, "Towards a context-aware middleware for deploying component-based applications in pervasive computing," in *Proceedings of the Fifth International Conference on Grid and Cooperative Computing*, ser. GCC '06, 2006, pp. 454–457.

[10] C. Taconet, Z. Kazi-Aoul, M. Zaier, and D. Conan, "Ca3m: A runtime model and a middleware for dynamic context management," in *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part I*, ser. OTM '09, 2009, pp. 513–530.

[11] I. Bouassida Rodriguez, G. Sancho, T. Villemur, S. Tazi, and K. Drira, "A model-driven adaptive approach for collaborative ubiquitous systems," in *Proceedings of the 3rd workshop on Agent-oriented software engineering challenges for ubiquitous and pervasive computing*, ser. AUPC 09, 2009, pp. 15–20.

[12] I. Lahyani, N. Khabou, and M. Jmaiel, "Qos monitoring and analysis approach for publish/subscribe systems deployed on manet," *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, vol. 0, pp. 120–124, 2012.

[13] G.SANCHO, "Adaptation d'architectures logicielles collaboratives dans les environnements ubiquitaires. contribution l'interoprabilit par la smantique," LAAS, 138p., LAAS Reports 10779, 2010-12-14.