Threshold-based context change detection for ubiquitous environments

Nesrine Khabou*
ReDCAD Research laboratory
Sfax University, Sfax
3078, Tunisia
nesrine.khabou@redcad.org

Ismael Bouassida Rodriguez†
ReDCAD Research laboratory
Sfax University, Sfax
Univ de Toulouse, LAAS,
F-31400 Toulouse, France
bouassida@laas.fr

Mohamed Jmaiel‡
ReDCAD Research laboratory
Sfax University, Sfax, 3078,
Tunisia
Research Center for Computer
Science & Multimedia of Sfax
Technopark of Sfax, B.P.275,
Sakiet Ezzit, 3021 Sfax,
Tunisia
mohamed.jmaiel@enis.rnu.tn

ABSTRACT

The increase of mobile and interconnected devices leads to the growth of demands for pervasive and mobile applications. Due to the heterogeneity of devices in terms of resources, capabilities, etc., these applications have to adapt themselves to the context changes and the execution environment. This requires the development of context aware applications that must perform four phases. Collecting and monitoring context, analyzing context, deciding adaptation actions and finally executing the planned adaptation actions to respond to the context changes. We focus on the second phase (analyzing context). It's aim is to analyze context to detect changes. In this paper, we propose a context analysis approach that relies on different thresholds defined according to the user needs to detect changes and raise notifications when changes occur. The analysis approach is divided into three steps. A context storage step, a context classification step, and a threshold calculation step.

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems—*Distributed applications*; C.4 [**Performance of Systems**]: [Measurement techniques, performance attributes]

General Terms

Measurement

*PhD student
†Associate professor
‡Professor

Keywords

Ubiquitous computing, context awareness, adaptation, analysis techniques, threshold calculation, mathematical models.

1. INTRODUCTION

Recently, we have been witnessing how various applications are being integrated more deeply into the life of everydays users [4]. These technological advances are providing the hardware infrastructure necessary to achieve the ubiquitous computing paradigm [9]. Ubiquitous computing is defined by Mark Weiser [11] as: *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it"*. One of the important properties of ubiquitous computing is context awareness that refers to the ability of an application to acquire and reason about context and adapt its behavior accordingly.

Therefore, applications in ubiquitous environments need to be context aware so that they can adapt themselves to the changing context. These applications that feature this property should perform four phases starting by collecting and monitoring context. These monitored information are exploited and analyzed during the second phase (Analyze) to detect changes and raise notifications. These notifications are used as a basis for the third phase (Decide) to decide the adaptation actions. In the last phase (Act), the adaptation actions are executed to react to the changes and adapt the application behavior accordingly. Our interest is focused on the second phase (Analyze). In fact, detecting changes in a timely manner is crucial to avoid indesirable situations such as device crashes, etc. In this paper, we detail the context analysis approach in ubiquitous environments. First, context parameters are stored for further use. Second, the stored context parameters are classified to facilitate the context use. Finally, the context parameters are analyzed and changes are detected by comparing context parameter values and computed threshold values. Notifications are triggered whenever the context values cross the threshold value.

The rest of the paper is organized as follows. In section 2, we present some studies related to context change detection (Context analysis). In section 3, we present a case study named "Smart Campus System". Its purpose is to provide suitable services to users according to context. The analysis

ICCASA 2014, October 15-16, Dubai, United Arab Emirates

DOI 10.4108/icst.iccasa.2014.258607

approach is detailed in section 4. First, we define a context parameter classification step which takes into account the context parameter evolution. Second, we detail the threshold calculation step that allows to analyze context and detect changes. In section 5, we illustrate the usefulness of our approach through an illustrative scenario. The last section concludes the paper and gives some directions for future work.

2. RELATED WORK

In the context changes detection research direction, several techniques are proposed in order to detect the context changes.

Cioara et al. [3] 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. 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 a self adaptation, it is restricted to external parameters such as temperature, humidity and light etc.

In other studies, context changes are detected by comparing a context value saved in a repository with a new context value. In fact, Zheng et al. [13] 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 parameters to determine relevant context changes and notify the application afterwards. Context filtering is based on a comparison of the context values stored 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 parameters to take into account. Another approach for dynamic context management is proposed by Taconet et al. in [10]. They present CA3M, a context aware middleware, which enables applications to adapt their behavior by dynamically taking into account context changes. The authors model the application by "entities", which represent a physical or a 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 leads to a different application behavior. Other approaches are used to analyze context. In fact, Bouassida et al. [2] proposed a model driven approach for collaborative ubiquitous systems. In order to detect context changes, they specify fixed thresholds. Then, once context values remain below/under the threshold values, a notification is raised. Although this approach enables to detect context changes, it may cause false detections by using fixed thresholds.

In the work of Hussein et al., [5], the authors proposed an architecture based approach for developping context aware adaptive systems. The proposed architecture has three layers that partition the system in a way that allows the context aware adaptive system requirements to be handled using the different layers. The context is first modelled at layer two (The system and its context representation), processsed and managed at layer three (The change management). This layer is responsible for adapting the system to cope with context. It is based on adaptation rules that rely on fixed thresholds to trigger adaptation actions. The above layered approach allows for context aware system development. The used technique for context change detection is based on fixed threshold which can lead to false detections or missing alarms.

Birje et al. [1] propose a multiagent model to monitor the resource availability and control the device state. The mobile agent models aims to provide not only a resource monitoring scheme to keep track of all devices and their resource utilization at any instant but also a device state control such that avoiding an overloaded state. For that reason, they consider three resource states: overload/poor state, an underload/excellent state and a normal state. To evaluate the resource state, the authors define two thresholds. A minimum threshold $threshold_{min}$ and a maximum threshold $threshold_{max}$. The proposed multi agent model monitors and controls the resource utilization, the resource availability, the device mobility and the device state by using the two thresholds. Although the proposed approach takes into account various parameters, the use of fixed thresholds can lead to false or missing detections.

3. CASE STUDY: SMART CAMPUS SYSTEM

To motivate the use of our approach, we introduce in the following an example of a smart campus system illustrated in Fig. 1. Smart campuses, with the ability to collect and analyze data, are built in order to benefit the institutions, the actors -the students and the teachers- by providing services which facilitate interaction between them. For instance, every actor is equipped with a personal device such as a Personal Digital Assistant (PDA), smart phone, tablet, etc. and the smart campus system provides different services to actors based on their current situations. To maintain the collaboration between students and teachers, the actors' devices need to be aware of their environment and their execution context, etc. Consequently, smart campuses contain an infrastructure that allows devices and systems to be monitored and adapted autonomously according to the changing context. Our work focuses on several context parameters such as temperature, pressure, position and light which need to be monitored. Furthermore, a multitude of resources such as the memory consumption, the energy, the CPU load and the available bandwidth should be monitored to assess both the devices and the communication state. The campus architecture depicted in the Fig. 1 involves different kinds of participants. On the one side, two controlling servers called Smart Campus Servers (SCS) namely SCS_1 and SCS_2 which are Ethernet-connected and equipped with important storage and high computational capabilities. On the other side, fixed (Presence sensors, cameras, lamps, air conditioners) and mobile (laptop, PDA, phones) devices are placed. The mobile devices are usually resource constrained in terms of memory, bandwidth and energy for example. Hence, a periodic monitoring by the controlling servers is needed in order to check their state which changes according to context. Two gateways called Smart Campus Gateways (SCG), SCG_1 and SCG_2 implementing software interfaces

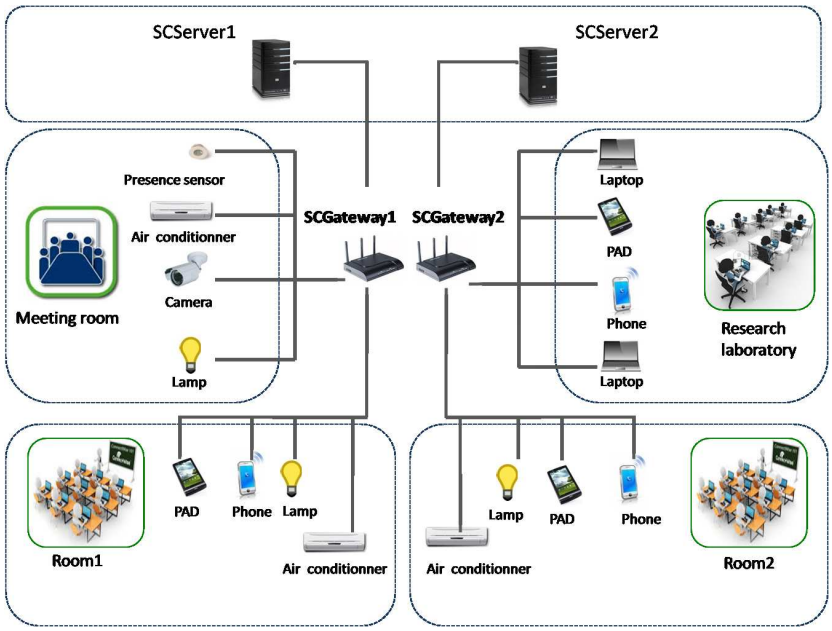


Figure 1: Case study: Smart campus

are used to connect devices to the corresponding controlling servers to exchange information. The campus space is composed of separate rooms (Research laboratory, Meeting room, Classroom1 and Classroom2). Each room is equipped with fixed devices as well as mobile devices that are carried by the different actors. Because of the complexity of the interaction between the different entities, we propose to focus our study on a part of the Smart Campus. It consists in the gateway SCG_2 connected to both the SCS_2 and the devices located in the Classroom2.

After presenting the case study, we detail in the following the proposed approach.

4. THE PROPOSED APPROACH

Our proposed approach is divided into three steps: A context storage step, a context processing and classification step and a threshold calculation step. The context storage step is out of the scope of this paper. In the following we present respectively the context classification step and the threshold calculation step.

4.1 Context parameter classification step

With a wide range of context parameters, context parameters should be classified into categories to use context easily. Since context parameters evolution is dynamic especially in ubiquitous environments, we propose to divide the context parameters into three categories according to the context parameter evolution. Three categories are identified. Parameters whose evolution is characterized by a trend, parameters whose evolution is characterized by peaks and parameters whose evolution is characterized by bursts. Therefore, this classification covers almost all context parameter types. Indeed, each context parameter evolves over time, so, it belongs inevitably to one of the defined context categories.

4.1.1 The trend category

A trend is defined as a line in a graph which shows the general direction that a set of points seem to be heading as illustrated in Fig. 2(a). For example, each mobile device such as PDA, mobile phone, etc. mentioned in section 3 periodically monitors its resource state namely the battery level, the memory consumption that belong to the trend category.

4.1.2 The peak category

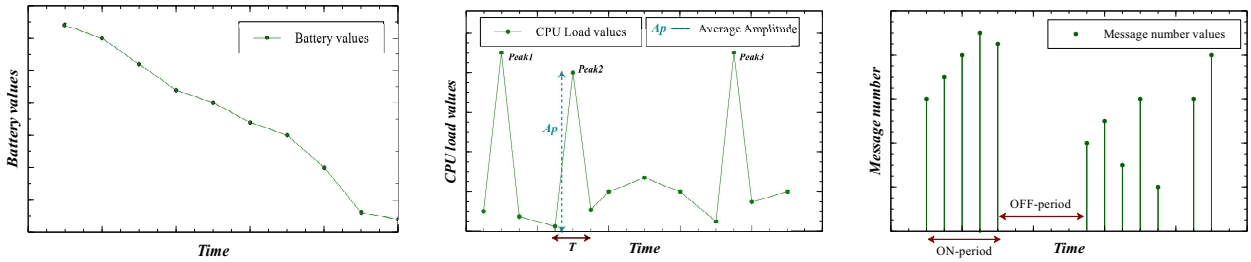
Peaks are defined as high values with sharp rise followed quickly by sharp fall implying a narrow period width [8]. We define a peak "*Peak*" by its amplitude "A_p" and its period "T" as depicted in Fig. 2(b). In fact, the peak is a very narrow period of high values- That is, its amplitude "A_p" exceeds for n times the average amplitude "A_{av}" of the time series formed by the context parameter evolution. We define a peak by the following formulas:

$$\begin{cases} A_p & = & n \times A_{av} \\ T & \leq & m \times TimeUnit \end{cases}$$

Where A_p defines the peak's amplitude, A_{av} defines the average amplitude of the context parameter evolution, n and m are constants fixed by the application designers. The SCSs receive the monitored data from the mobile devices in order to analyze them. For that reason, the SCSs CPU load can rise suddenly reaching high values after a high rate of requests. Hence, the CPU load especially of the SCSs can be modeled by a peaked function. The link load and the available bandwidth belong also to this context category.

4.1.3 The burst category

A burst consists on a relatively wide contiguous region of values. Otherwise, a burst is defined as a large number of occurring events [6]. As depicted in the Fig. 2(c), we model the wide region by an ON period and the other by an OFF behavior [12]. The ON-period models a single flow such



(a) Trend category (b) Peak category (c) Burst category

Figure 2: The proposed context categories

as the transfer of a single web page, and the OFF-period models the user's thinking time. The ON-period and the OFF-period are strictly alternating. The message number received by the SCSs during an ON-period is modeled as a burst.

4.2 Threshold calculation step

In the following, we detail some elements about threshold calculation for each context category defined previously.

4.2.1 *Threshold calculation for the trend category*

For this category, the context parameter evolution is described by a trend. In order to avoid false detections as well as missing alarms, we need to define thresholds which are uncorrelated with the context parameter evolution. The notification raised when the context parameter behavior crosses the threshold is illustrated in Fig. 3.

Uncorrelated thresholds can be fixed thresholds, uncorrelated adaptive thresholds and step function thresholds. For instance, fixed thresholds are defined by the application designers according to their needs in terms of Quality Of Service (QoS).

For the adaptive threshold denoted in the Fig. 3(b), mathematical methods can be applied in order to update threshold values at runtime. However, for this kind of context parameter characterized by a trend, adaptive threshold must be uncorrelated with the context parameter evolution in order to avoid false detections and missing alarms. For the step function threshold described in the Fig. 3(c), thresholds are defined per period.

4.2.2 *Threshold calculation for the peak category*

In this context category, the idea consists in specifying adaptive thresholds that are correlated with the context parameter evolution. Adaptive thresholds are calculated via diffrent mathematical models such as Exponential Weighted Moving Average (EWMA) technique used by Lahyani et al. [7].

4.2.3 *Threshold calculation for the burst category*

In this context category, our idea consists on transforming a bursty model into a trend/peak model. So we propose to apply an aggregate function G in each ON-period. The obtained model coincides with a trend function as illustrated in Fig. 5(b). A notification is raised when the context parameter crosses the threshold calculated in an ON-period.

For the smart campus case study (section 3), the gateway

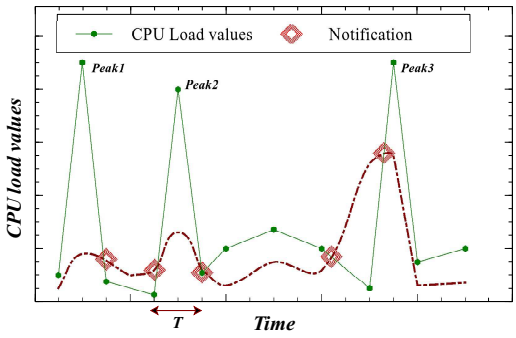


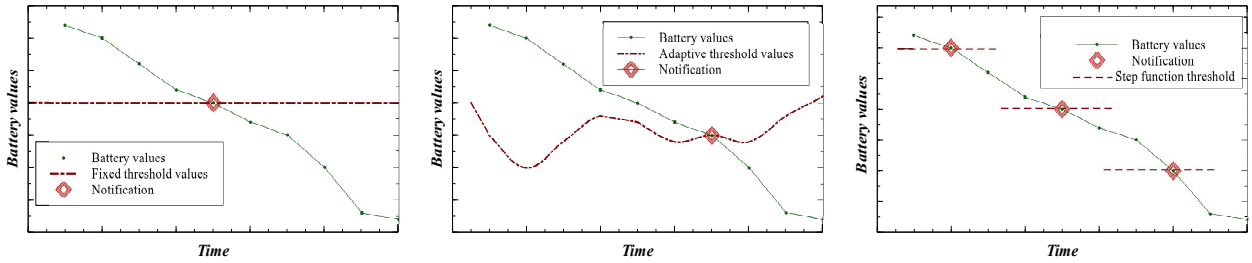
Figure 4: Threshold calculation for the peak category

SCG_2, analyzes the received data from the devices. The maximum queue size for this device is set by the application designer. So that, if the traffic received by this gateway in a period T_i exceeds a maximum, then a burst is identified in T_i. To detect bursty periods, applying the aggregate function G consists in calculating the slope of the scatter diagram obtained in each ON-period. Second, in each ON-period, we compute the intensity of each slope formed in each ON-period. We obtain the Fig. 3(b). Consequently, if the slope intensity exceeds the threshold, then a burst is detected and appropriate adaptation actions are triggered.

5. ILLUSTRATIVE SCENARIO

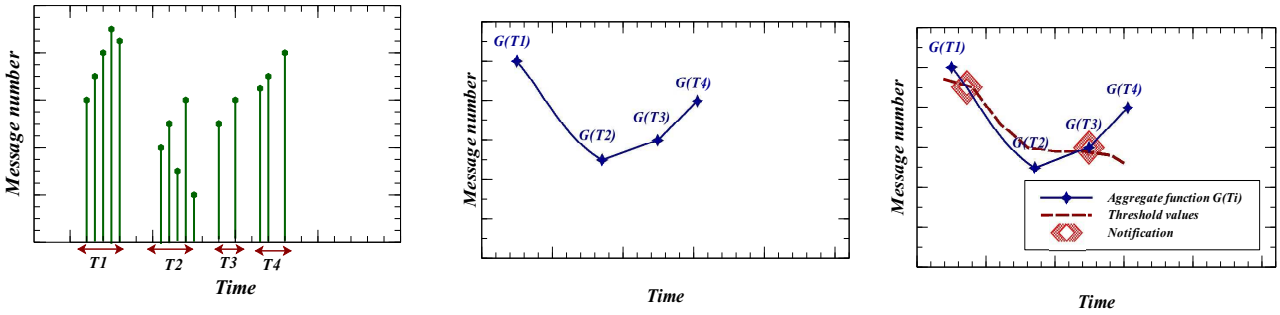
To illustrate the usefulness of our approach, we elaborate the following scenario which is conducted in the smart campus system presented in Fig. 1. It highlights the ability of an application to react accordingly to the changing context. We focus on the interaction of the actors of the Classroom2 and SCS_2 through SCG_2. Classroom2 is used by the researchers, the teachers and the students. Different context parameters are considered in this scenario.

- The actors' position: It belongs to the peak category. Adaptive thresholds are applied to this context parameter.
- The memory consumption: It belongs to the trend category. Fixed, adaptive or step function thresholds are applied to this context parameter.



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

Figure 3: Threshold calculation for the trend category



(a) Burst model (b) Aggregation function application (c) Threshold calculation

Figure 5: Threshold calculation for the burst category

The scenario starts as follows: At the beginning of each course session, the presence sensor captures and localizes the mobile actors. Their positions are then forwarded to SCS_1. SCS_1 runs the analysis algorithm based on thresholds on the context parameter (mobile actors' position) and takes the appropriate desicion. Each student participating to a course uses a tablet device to which the course will be dispatched through bluetooth. The tablet device holds the analysis algorithm in order to detect context changes. During the course, the students' tablet display appropriate slides and they follow their courses. Furthermore, the students can write annotations on their tablets and publish their comments to share knowledge between all the group members to enrich the course and enhance the collaboration. A student holding a tablet is participating to the course by exchanging information and slides.

For each amount of data received, the tablet device retrieves the memory consumption from the operating system using probes. Since the memory consumption belongs to the trend category, fixed threshold(Threshold3), step function threshold (Threshold1) and uncorrelated adaptive threshold (Threshold2) are applied for this context parameter as illustrated in the Fig. 6.

- $Threshold_3$: A fixed threshold (value=90). This threshold is specified by the application designer. It corresponds to a critical memory. For instance, if the memory values crosses this fixed threshold, the analysis module detects a device crash.
- $Threshold_2$: An adaptive threshold. This threshold is updated using the mathematical formula described as follows:

$$Threshold_t = \lambda \cdot t + \beta$$

This threshold level corresponds to a normal memory state. As shown in Fig. 6, we do not need to trigger adaptation actions when memory values oscillate around the $Threshold_2$. However, we must react when the memory not only goes past the $Threshold_2$ but comes closer to the critical threshold ($Threshold_3$).

- $Threshold_1$: A step function threshold defined as follows.
  - For the first step (threshold value=30), we can consider that the course to which the student is participating does not require exchanging information.
  - For the second step (threshold value=50.5), we can consider that the student is participating to a course that requires interaction with the teacher, displaying and exchanging information.
  - For the third and the fourth step (threshold value=20, threshold value=20.5) respectively, we can consider that the student is in a pause time or he is not doing any activity.
  - For the last step (threshold value=70.5), it means that we must react once the memory values crosse this threshold and increase rapidly.

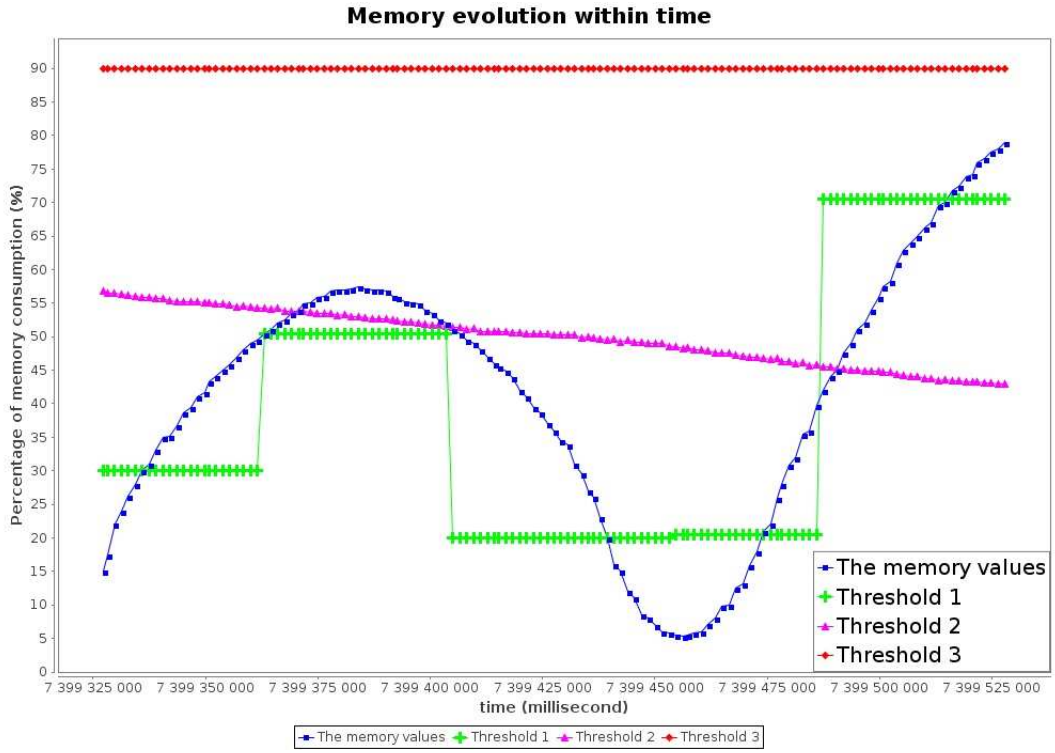


Figure 6: Application of threshold on the memory context parameter

6. CONCLUSION

The challenges for context aware applications design and implementation are to perform four phases.Collecting context from different sources such as sensors, widgets, etc., analyzing context to detect changes, deciding adaptation actions and executing the planned action to react dynamically to context changes. In this paper, we have considered context analysis. we have proposed an analysis approach for context change detection in ubiquitous environments. The proposed analysis approach is divided into three steps. A context storage step, a context classification step and a threshold calculation step. The context storage step is out of the scope of this paper. We have proposed a context classification based on context parameter evolution. This context classification takes as input the collected context parameter and attributes for each context parameter one or many categories. Three categories have been identified. Second, we have detailed the threshold calculation step that aims at analyzing context and identifying context changes. Threshold calculation allows to analyse context parameters, identify context changes and notify the context aware application. The thresholds can be fixed, step function and adaptive ones. for the adaptive thresholds, mathematical models are applied in order to update the threshold used to detect context changes. Further, when the context parameter crosses the threshold, a notification is triggered to react to the context changes. As future work, we plan first to stretch the context parameter classification. Second, our aim consists not only in detecting context changes but also predicting context behavior using mathematical models such as autoregression.

7. REFERENCES

[1] M. Birje and S. Manvi. Multiagent model for device state control in the wireless grid. In *Electronics Computer Technology (ICECT), 3rd International Conference on*, volume 3, pages 456 –460, Apr 2011.

[2] 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*, AUPC 09, pages 15–20, New York, NY, USA, 2009. ACM.

[3] 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*, pages 374 –379, Jun 2010.

[4] E. Gilman, O. Davidyuk, X. Su, and J. Riekki. Towards interactive smart spaces. *JAISE*, 5(1):5–22, 2013.

[5] M. Hussein, J. Han, A. Colman, and J. Yu. An architecture-based approach to developing context-aware adaptive systems. In *Engineering of Computer Based Systems (ECBS), 2012 IEEE 19th International Conference and Workshops on*, pages 154–163, 2012.

[6] D. Klan, M. Karnstedt, C. Politz, and K. Sattler. Towards burst detection for non-stationary stream data. In *KDML*, pages 57–60, 2008.

[7] I. Lahyani, N. Khabou, and M. Jmaiel. Qos monitoring and analysis approach for publish/subscribe systems

deployed on manet. In *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, pages 120–124, Los Alamitos, CA, USA, Feb 2012. IEEE Computer Society.

[8] G. K. Palshikar. Simple Algorithms for Peak Detection in Time-Series. In *Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence*, 2009.

[9] L. A. San Martín, V. M. Peláez, R. González, A. Campos, and V. Lobato. Environmental user-preference learning for smart homes: An autonomous approach. *J. Ambient Intell. Smart Environ.*, 2(3):327–342, Aug. 2010.

[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*, OTM '09, pages 513–530, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] M. Weiser. Human-computer interaction. chapter The Computer for the 21st Century, pages 933–940. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[12] X. Yang. Designing traffic profiles for bursty internet traffic. In *In Proceedings of IEEE Global Internet*, 2002.

[13] 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*, GCC '06, pages 454–457, Washington, DC, USA, 2006. IEEE Computer Society.