A Flexible Context Stabilization Approach for Self-Adaptive Application

Russel Nzekwa, Romain Rouvoy, Lionel Seinturier

A Flexible Context Stabilization Approach for Self-Adaptive Application

Russel Nzekwa
University of Lille 1, LIFL
INRIA Lille – Nord Europe
ADAM Project-Team
F-59650 Villeneuve d'Ascq
russel.nzekwa@inria.fr

Romain Rouvoy
University of Lille 1, LIFL
INRIA Lille – Nord Europe
ADAM Project-Team
F-59650 Villeneuve d'Ascq
romain.rouvoy@inria.fr

Lionel Seinturier
University of Lille 1, LIFL
INRIA Lille – Nord Europe
ADAM Project-Team
F-59650 Villeneuve d'Ascq
lionel.seinturier@inria.fr

***Abstract*—Pervasive applications are characterized by variations in their context of execution. Their correct behavior requires continuous adaptations, accordingly to changes observed in their environment. Some of the existing approaches tackle this problem by adding stabilization mechanisms on the decision making layer. In most cases it remains costly for applications to trigger decision making procedures, specially when application changes frequency is potentially high. We believe that, to provide more flexible and efficient context-aware applications, stabilization issues should be addressed separately from decision making issues. In this paper, we present a flexible stabilization approach as an elegant solution to that problem.**

I. Introduction

Pervasive applications aim to seamlessly provide services to their users. To achieve this task, they need to adapt according to changes observed in their environment. From one application to another, the nature [1] (nominal, ordinal, numeric) of processed data varies in number and complexity. In order to optimize decision making processes for reconfiguration or adaptation, stabilization mechanisms are required. Stabilization is then a common problem for pervasive systems. Context region [2] is an example of a stabilization mechanism which advocates a strict partition of context space, in oder to reduce the adaptation side effects. However, existing stabilization mechanisms, only partially solve the problem. In particular context regions, does not handle properly application behavior during the transition from one state to another. More powerful and complex stabilization mechanisms based on learning algorithms are suggested in many works like [3], [4] for example. But, despite some good results in predicting application behavior, they still suffer from weak reactivity.

In this paper we propose a smart approach for composing stabilization algorithms in order to increase their efficiency and accuracy. First, we present the conceptual architecture and feature diagrams that define the rules governing relationships between algorithms. Then, we explain the composition modalities for stabilization algorithms.

The rest of the paper is organized as follows. In section II, we discuss some related work. In section III, a conceptual architecture and features diagrams are introduced. We suggest a flexible context stabilization approach in section IV. We show some simulation results in section V. Finally, we conclude this paper in section VI.

II. Related Work

Although only few works in the state of the art directly address stabilization issues for context-aware applications, for example in [5], [6] , the issue of stabilization is implicitly omnipresent in most of the works that are related to context reasoning or context fusion of information. Most of the proposals concerning stabilization algorithms or techniques for context-aware applications can be categorized into five groups. The first group is based on *Filtering techniques*. It focuses on data filtration using statistic or parametric based techniques. The second group is composed of algorithms based on *threshold techniques*. Stabilization is done by checking the system state regarding the threshold values. The third group of algorithms is based on *refresh techniques*. The fourth group is based on *probabilistic schemas*. Stabilization is done by inferring the system's state on the base of probability and previous states of the system. The fifth group, contains all the uncategorized algorithms that use specific methods to handle the stabilization of the system. The works of Folliot et al. [7] and Da Rocha et al. [8] belong to the last group. The work of Temal et al. [6] is an example of the second group. The works of Dargie [3] and Sekkas et al. [9] is an illustration of the fourth group. Stabilizations techniques used in the project TEA [10] could be categorized in the fifth and third group according to the previous classification. One of the project goal was to evaluate the impact of using many sensors instead of a single one, more powerful, in a mobile environment. In order to stabilize data collected from different data sources, several stabilizations techniques like stacking layer, or refresh techniques were implemented. The principal limitation of this project is that, implemented stabilization techniques are tightly coupled to application architecture and could not easily be replaced or modified. [7], presents the PHOENIX project which targets the management and monitoring of distributed

system (clusters) processing many heterogeneous data. To improve the performance of the system monitoring, and detect emergency situation just when they happen, PHOENIX introduces DELTA OPERATOR (DO) with other first order logic operators. DO's allow to evaluate the variation of signal amplitude, so the data flow is relayed only when it has satisfied the required threshold value. Despite good results recorded by the method, one of the main drawbacks is that, the PHOENIX platform do not support extension of new operators, and that the stabilization method is targeted for quantitative numerical data. In [5], Padovitz suggested an architecture to handle stabilization issue in pervasive systems on the base of kalman's filter learning algorithm, which refers to the fifth group of the classification presented above. The main limitation of that approach was the weak reactivity of the system compared to other approaches. Indeed, a survey of stabilization techniques done at earlier stage of our work shows that a unique stabilization algorithm that could meet all requirements of pervasive applications does not exist. In fact, for pervasive applications the environment keeps evolving, and to be efficient stabilization strategy must be flexible enough in order to be adaptable depending on changes in the environment. We believe that, in the perspective of improving the management of stabilization process for pervasive systems, a solution can be found in the flexible combination of several existing approaches.

III. Conceptual architecture and Features diagrams

If we take a look at the general organization of a context-aware application, we can identify, independently of the implementation, four main entities that characterize the execution process of such an application: DATA COLLECTORS, DECISION MAKING, ACTUATORS, and the ENVIRONMENT. In this architecture, the DECISION MAKING entity is the heart of the application, and the place where all decisions about system adaptation or reconfiguration are made. This traditional organization of context-aware applications, is not efficient enough for systems where data frequently vary, suddenly and unpredictably. This is justified by the fact that, the DECISION MAKING entity has to handle at the same time system reconfiguration and data stabilization issues. On that specific point, we believe that we can improve stabilization efficiency if we can address stabilization separately from decision making issues. By applying this principle, it may result a flexible and highly reusable stabilization strategies for pervasive systems. In the following paragraphs, we present a conceptual architecture and feature diagrams as a base of our flexible stabilization approach presented in section IV.

A. Conceptual architecture

The main principle underlying this architecture consists in using first, during the stabilization process, algorithms with a relative low execution cost on the lower layer, which means close to context sources, and those with relative high cost would be bounded on the upper layers. Next, we define a set of principles to discriminate algorithms in our architecture. Criteria used are: The algorithm complexity, and the nature [1] of manipulated data.

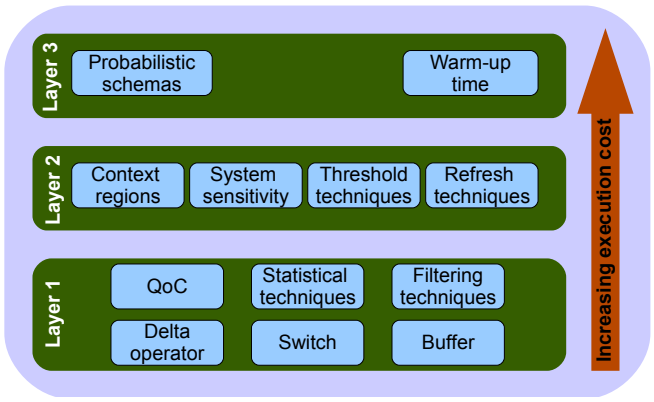

Figure 1. Conceptual architecture

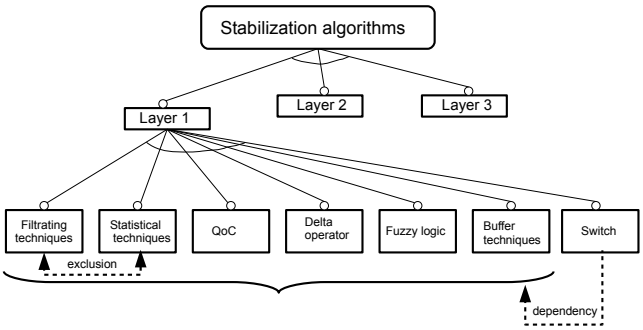

Figure 2. Feature diagram: Layer 1

Based on these criteria, we propose a classification for the most commonly used stabilization techniques. We propose a multi-layer architecture that shows the relationship between stabilizations algorithms depending on their execution cost. Figure 1 gives an illustration of this architecture.

In order to apply the composition model that we describe in section IV-A it is important to understand the constraints and dependencies between the algorithms in the same layer. The next Section III-B explains it in detail using feature diagrams.

B. Features diagrams

In this section we present existing relationships between algorithms that belong to the same layer of the conceptual architecture that we have presented above. Features diagrams corresponding to each layer of the conceptual architecture are depicted in Figures 2 ,3 and 4.

Figure 2 shows the relationships between algorithms in the first layer of the conceptual architecture. Typically,

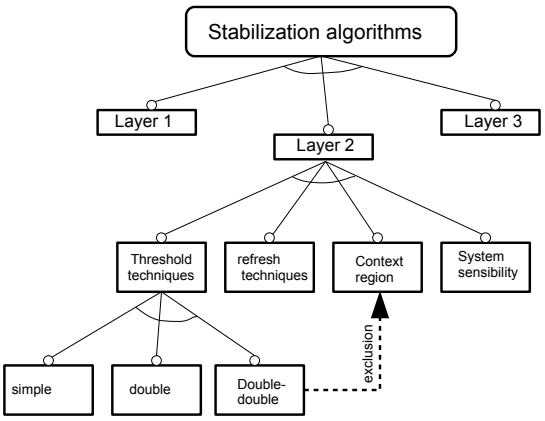

Figure 3. Feature diagram: Layer 2

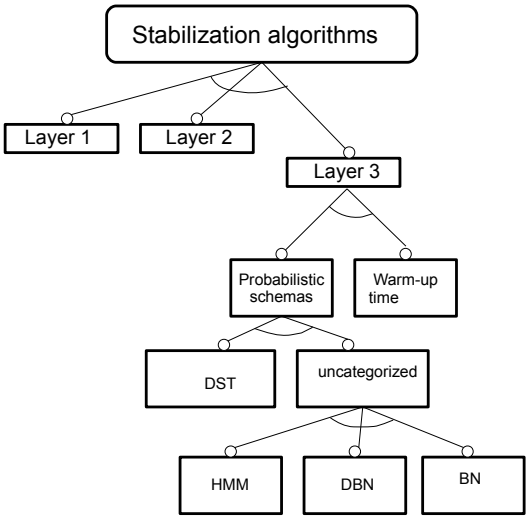

Figure 4. Feature diagram: Layer 3

the diagram points out that the implementation of "switch technique" depends on the other algorithms of the same layer. Indeed, there is a need of at least two other algorithms in order to use this stabilization technique. Moreover, we can notice that implementation of "filtering techniques" excludes, the use of "statistical techniques" in the list of algorithms that are candidates for an association. This is necessary to reduce redundancy, since even if these algorithms in our categorization fall into different groups, they implement quite similar methods for data processing. Some similar kind of relationships between algorithms can be observed for the two others features diagrams.

IV. Flexible Context-aware Architecture

The main limitations of existing stabilizations approaches that we point out in section III were amongst other things,

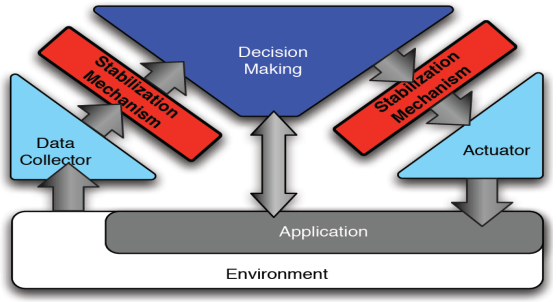

Figure 5. Flexible context-aware architecture

the lack of flexibility in terms of reuse of stabilization techniques, or in terms of management of the data stabilization process. The main challenge when trying to stabilize a context-aware application is to find the right level of stabilization, in order to keep the application reactive enough to be aware of important changes that might occur in the surrounding environment. We present here, our flexible stabilization approach for pervasive systems. We consider that in order to boost flexibility and efficiency stabilization issues should be handled separately from decision making basics concerns. In that way, we suggest to integrate stabilization blocks at two levels of the chain processing information data: Firstly, after acquisition of raw data from different probes or sensors, and secondly, after the decision making block. Figure 5 depicts this architecture. The purpose of the first stabilization block, located between data collectors block and decision making block, is to filter data coming from collectors and to forward to decision making structures only significants values that could lead to system reconfigurations. This considerably reduces access to decision making module, resulting in a gain of performance for the system, since running decision making processes is quite often a costly procedure for the application. The purpose of the second stabilization block, in our architecture located between decision making block and actuator, is to optimize the work of actuators by finding the suitable moment when reconfigurations of the system can be executed. This issue has a significant impact when dealing with distributed applications, where the decision making structure and the application are not hosted on the same machine. In order to realize an effective stabilization, stabilization mechanisms should take into account the following characteristics of contextual information mentioned in [11]: (i) Inaccuracy- This property expresses the fact that contextual information give the measure of observed phenomena with some errors. (ii) Incoherence- This property denotes that informations

collected from different sensors can be contradictory. (iii) Incompleteness- This property expresses the fact that some aspects of the observed phenomena can be absent from contextual informations. In order to meet flexibility property for stabilization, we need a generic approach to handle this issue, namely, a flexible model to implement stabilization mechanisms in a context-aware architecture. For that purpose, we define a composition model that we are going to presented below .

A. Composition Model

Our composition model defines how to combine different techniques or stabilization mechanisms in order to meet flexibility in the application. The model consists of two modalities of composition: *Horizontal composition* and *Vertical composition*.

1) Horizontal Composition: Some context-aware application implement learning based stabilizing algorithms like Dempster-Shaffer (DST) [4] or Bayesian Network (BN), in order to improve the knowledge about their environment. This set of algorithms are known both for their efficiency in predicting contextual changes, and their higher cost of execution. Besides, these algorithms introduce a certain latency in detecting changes in the environment. To increase system's reactivity for detecting changes in the environment, several methods have been proposed. Some methods suggest to model context-state of a pervasive systems, it is the case of Padovitz et al. [5]. The main limitations for the proposed approach are the difficulty to model context-aware application in a context-state systems, and the complexity to evaluate the mathematical expression determining instability of the system, when the number of arguments increases.

Horizontal composition consists in executing concurrently several stabilization algorithms. The idea behind this concept is to benefit from passive and reactive stabilization techniques by combining them adequately. The detection of irregular application's behavior can be done by combining a reactive algorithm like Delta Operator (DO) , and less reactive algorithm like Dempster-Shaffer (DST) algorithm, through a composition rule (CR). A CR can be some simple rule like " $if \quad v_n \geq \lambda \Rightarrow \quad proceed \quad value$", where v_n is a context value and λ a threshold value, or a more complex rule involving Quality of Context (QoC) of the processed data. Hence, using this composition model can help to improve accuracy of the stabilization process while keeping a reasonable level for the reactivity of the system. We will further detail to this example in section V.

2) Vertical Composition: Vertical composition in our approach consists in applying sequentially two or more stabilization algorithms on the same data set. Some works [9], suggest that it can be interesting in terms of performance for stabilization of context information to apply successively several algorithms of stabilization on the same sample of data. A good illustration of that idea is the work of Sekkas et al. [9] where, the authors compare the efficiency of using Bayesian Network (BN), Dynamic Bayesian Network (DBN), and Fuzzy Logic (FL) alone or in a combined way. In our approach we believe that, the combination of algorithms using *vertical composition* can increase efficiency of the stabilization . In order to limit the overhead introduced by the use of several algorithms, less costly (execution) algorithms are found at the beginning (bottom) of the stabilization chain while costly algorithms are on the top of the architecture. This association rule is mostly justified by the fact that, the amount of processed data decreases from the bottom to the top, thus more costly algorithms at the top of the architecture would have to process less data, decreasing by the same way the overall cost of the stabilization process, which is tightly bound to the amount of context information processed.

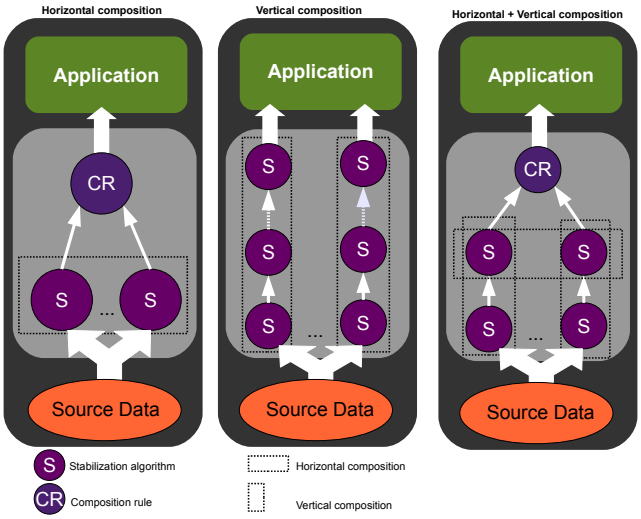

Figure 6. Composition Models

Figure 6 gives an illustration of the composition model. From left to right, we have horizontal composition, then vertical and finally the combination of the primitive types of composition. Combination of both primitive composition models in our approach can be benefit for the stabilization process in order to meet accuracy and efficiency properties.

V. Evaluation and Simulations

In the previous sections, we presented a novel approach in order to meet flexibility when dealing with stabilizations issues in pervasive applications. We suggested to use composition of algorithms based on the conceptual architecture that we have presented in section III. Composition of algorithms offers the possibility to maximize the efficiency of the stabilization process at a relative low cost for the application. While in most of the existing systems, the decision making module applies a generic algorithm to processed data, our approach offers the possibility to adapt the stabilization

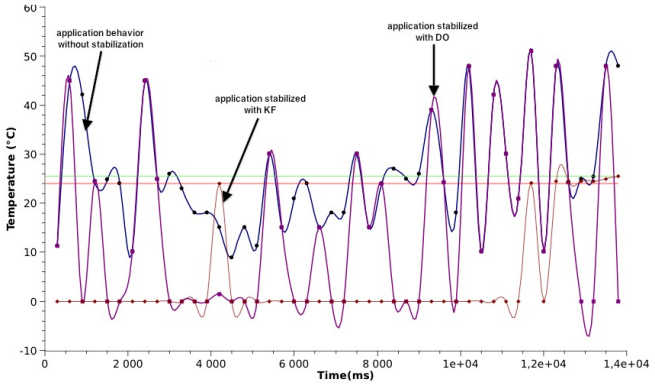

Figure 7. Stabilized application behavior - Kalman filter and Delta operator

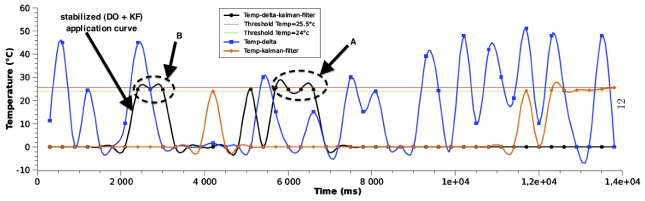

Figure 8. Comparatives curves of stabilized application behavior

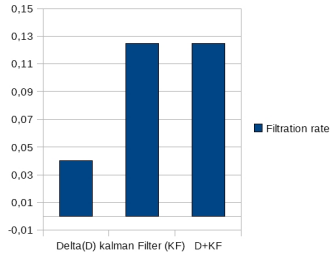

Figure 9. Filtration rates measurements

mechanism to the data being processed.

To evaluate our proposition we have made some experimental simulations. Our simulations are based on COSMOS [12](Context Entities Composition and Sharing), which is a framework for context management that uses the FRACTAL [13] component model. We also use the simulation engine SIAFU [14] to generate contextual informations.

The scenario that we used is the one presented in [5] and that we briefly detailed in section IV. We have made some changes in order to meet requirements of our experimental platform. In the current scenario, we have temperature sensors simulated by the simulator engine that send data concerning the temperature of the environment. Time elapsed between two consecutive's readings from sensors is 0.3 second. Collected values have an accuracy close to 0.1. To be able to evaluate the impact of the stabilization on events detection we will focus our observations of the application behavior around $24°\mathrm{C}$ and $25.5°\mathrm{C}$.

The first implemented algorithm, is "Delta operator" (DO) [7]. The choice of the threshold value is motivated by the data sample's variance, in order to choose the most appropriate value. The second implemented algorithm is kalman filter. The central point for kalman's filter algorithm is the determination of the matrix of transition A, in this case we defined the transition matrix as following: Given A the transition matrix, given x_{k-1}, z_{k-1}, R the prediction, the value of measured variable at $k-1$ step and the variance respectively. We have defined ε, Δ then A as follows: $\Delta = x_{k-1} - z_{k-1}, \varepsilon = R * z_{k-1}$

$$A_k = \begin{cases} A_{k-1} & for \quad |\Delta| \leq \varepsilon \\ \frac{z_k - \varepsilon}{x_{k-1}} & for \quad |\Delta| > \varepsilon \quad and \quad \Delta < 0 \\ \frac{z_k + \varepsilon}{x_{k-1}} & for \quad |\Delta| > \varepsilon \quad and \quad \Delta > 0 \end{cases}$$

Figure 7 shows the curve of the application behavior without stabilization, stabilized with KF, and stabilized with DO. As expected, application stabilized with DO is very reactive, we can notice that around $t = 2s$ for example, the curves of the application without stabilization and stabilized with DO are almost indistinguishable. Figure 8 shows application's behavior curves when combining the both algorithms using our model, in this case an horizontal composition. On Figure 8, the curve of the behavior of the application stabilized with both KF and DO algorithms is represented. On that graph, we can notice two things. Firstly, the detection of the first event for the application occurs earlier (zone B) than for an application stabilized by KF only. That is due to the fact that the application benefit from the reactivity property of DO algorithm. Secondly, we can notice that when an event is detected the behavior of the system does not change suddenly (zone A or zone B) even when the next raw value process by the system does not belong to the target area. That last point is a property that the system inherited from KF algorithm. Obviously, the composition model offers a good compromise between a reactive and less reactive system. Finally, to characterize our system we have measured the filtration rate, the results of the measurement are depicted in 9. We can notice that resulting application (DO+KF) obtains a good filtrating rate which is equivalent to the one for and application stabilized only with KF.

VI. Conclusion

In this paper, we have proposed a flexible approach for context stabilization. For that purpose, we suggest two modalities of composing stabilization algorithms in order to increase accuracy and efficiency of the stabilization. We also present a conceptual architecture and feature diagrams that describe relationships between algorithms for the given approach. Finally, we showed simulation examples that validate our approach. In the future, we plan to study how this approach can be applied in the field of Wireless

Sensors Network (WSN) where stabilization issue is still a crucial challenge, and in large scale infrastructures where the flow of information is critical.

Acknowledgments. The work reported in this paper is partly funded by the FEDER and the ANR ARPEGE SALTY project.

REFERENCES

[1] R. Mayrhofer, H. Radiand, and A. Ferscha, "Recognising and Predicting Context By Learning From User Behavior," *The International Conference On Advances in Mobile Multimedia (MoMM2003)*, vol. 171, pp. 25–35, September 2003, austrian Computer Society (OCG).

[2] "Quality objects (quo)," http://quo.bbn.com.

[3] W. Dargie, "The role of probabilistic schemes in multisensor context-awareness," *5th IEEE international conference on pervasive computing and communication workshops (PerCom'07)*, 2007.

[4] H. Wu, "Sensor data fusion for context-aware computing using dempster-shafer theory," Ph.D. dissertation, Carnergie Mellon University, pittsburgh, 2003, phd thesis dissertation.

[5] A. Padovitz, A. Zaslavsky, S. W. Loke, and B. Burg, "Maintaining Continuous Dependability in Sensor-Based Context-Aware Pervasive Computing Systems," *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, vol. 9, 2005.

[6] L. Temal and D. Conan, "Failure, connectivity and disconnection detectors," *In Proc. of the 1st French-speaking conference on Mobility and ubiquity computing*, 2004.

[7] C. B. Saab, X. Bonnaire, and B. Folliot, "PHOENIX: A Self Adaptable Monitoring Platform for Cluster Management ," *Cluster Computing*, vol. 5, no. 1, january 2002.

[8] C. A. da Rocha, Ricardo, and E. Markus, "Middleware: Context Management in Heterogeneous, Evolving Ubiquitous Environments," *IEEE Distributed Systems Online*, vol. 7, no. 4, 2006.

[9] O. Sekkas, C. B. Anagnostopoulos, and S. Hadjiefthymiades, "Context fusion through imprecise reasoning," *International Conference on Pervasive Services*, pp. 88–91, 2007.

[10] H. W. Gellersen, A. Schmidt, and M. Beigl, "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts," *Mobile Networks and Applications*, vol. 7, no. 5, pp. 341–351, September 2002.

[11] K. Henricksen, J. Indulska, and A. Rakotonirainy, *Modeling Context Information in Pervasive Computing Systems*. First International conference on pervasive computing, August 2002, vol. 2414.

[12] R. Rouvoy, D. Conan, and L. Seinturier, "Software Architecture Patterns for a Context-Processing Middleware Framework," *In IEEE Distributed Systems Online (DSO)*, vol. 9, no. 6, June 2008.

[13] E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, and J.-B. Stefani, "The Fractal Component Model and Its Support in Java," *Software Practice and Experience, special issue on Experiences with Auto-adaptive and Reconfigurable Systems*, 2006.

[14] M. Martin and P. Nurmi, "A Generic Large Scale Simulator for Ubiquitous Computing," *In Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, July 2006.