Object Technology for Ambient Intelligence and Pervasive Computing

Report on the OT4AmI-Workshop at ECOOP 2007

Jessie Dedecker[1,2,⋆], Éric Tanter[2,⋆⋆], Holger Mügge[3], Cristina Videira Lopes[4], and Pascal Cherrier[5]

[1] PROG Lab, Vrije Universiteit Brussel, Belgium
[2] PLEIAD Lab, Computer Science Dept (DCC), University of Chile, Chile
[3] University of Bonn, Germany
[4] University of California, USA
[5] France Telecom, France

Abstract. This report summarizes the main activities held during the third workshop on object-technology for Ambient Intelligence and Pervasive Computing. The goals of this workshop series are to identify and discuss the impact of Ambient Intelligence on object-oriented technologies and vice versa. This report summarizes the scope of the workshop as well as the contents of the presented position papers, the discussions provoked by these papers and the brainstorm sessions that followed the presentations. In particular, groups of participants actively discussed issues such as context volatility, development process, and user involvement in adaptable applications.

1 Introduction

Computing technology is no longer uniquely associated with mainframes or desktop computers. Due to the technological advances computing technology has become so small and cheap that it can be embedded in everyday devices such as cars, toys, furniture and even clothes. This integration of technology with everyday devices, which is also known as "ubiquitous computing", "pervasive computing" and "ambient intelligence", enables new types of applications and requires a model of interaction that is less intrusive than the traditional desktop model of computing. The idea is that everybody will be surrounded by a dynamically-defined processor cloud, of which the applications are expected to cooperate smoothly. This technological setting puts new challenges on the software. Software should take into account the context in which it operates and adjust its behavior as its ambient context changes over time. Context changes are provoked due to the mobility of the users and the devices. Devices form a large-scale distributed system that communicates using wireless technology.

⋆ J. Dedecker is partially funded by FONDECYT project 3080020.
⋆⋆ É. Tanter is partially financed by the Millennium Nucleus Center for Web Research, Grant P04-067-F, Mideplan, Chile, and FONDECYT Project 11060493.

M. Cebulla (Ed.): ECOOP 2007 Workshop Reader, LNCS 4906, pp. 171–181, 2008.

The use of wireless technology and limited energy resources implies that the distributed software system will be subject to a higher rate of failures as compared to a traditional distributed system. Other issues such as security, privacy, new interaction models, limited resources and others also have a high impact on the design and implementation of software.

2 Scope of the Workshop

2.1 Goals

Important goals of the workshop were to identify and discuss the impact of Ambient Intelligence on object-oriented technologies and vice versa, and to outline fruitful paths for future research concerning the connection between Ambient Intelligence and object-oriented programming languages and systems. In this context, we understand the term "object technology" to cover the whole range of topics that have evolved around the notion of object-orientation in the past decades, starting from programming language design and implementation, ranging over software architectures, frameworks and components, up to design approaches and software development processes.

2.2 Topics

In the call for participation, the following non-exhaustive list of potential topics was included: programming models, reflection, security, software adaptation, context modeling, engineering of autonomous systems, biologically-inspired concepts, human-device and device-device interaction. We accepted eight workshop papers discussing various topics such as distributed memory management, mobile agents, and rule-based systems to support the development of ambient intelligent systems.

2.3 Workshop Organization

The workshop format was chosen to promote discussions on the topics set for the workshop. In order to accommodate this format we asked the authors to prepare short (with a maximum of ten minutes) presentations where they would defend the position of their paper. Hence, the goal was not to elaborate on the technical details of their work but rather defend their vision on how software for ambient intelligence should be developed. In preparation of the workshop we asked all participants to read the accepted papers. The positions of the accepted papers were presented in the morning. The afternoon session was used to brainstorm on emerging topics resulting from the paper discussions.

3 Summary of Position Paper Discussions

In the paper session each principal author defended the position of his paper in a short (under ten minutes) presentation. After the presentation there were two discussion rounds. In the first round of discussions two participants were selected:

one participant had to argue in favor of the position whereas the other participant had to argue against the position. In this process the presenter was not allowed to intervene and could only act as an observer. The second discussion round enabled participants to ask questions to the presenter and gave the presenter the opportunity to react to remarks made in the first discussion round. This section presents the abstracts of the submitted position papers and summarizes the discussions that followed the presentations. These papers are included in the workshop proceedings [4], and can be downloaded from the workshop's home page at `http://sam.iai.uni-bonn.de/ot4ami2007/`

3.1 Introducing Context-Awareness in Applications by Transforming High-Level Rules [5]

In the last years, we have witnessed the increase in the popularity and capabilities of mobile technologies. This evolution has enforced the idea of smart environments, in which devices are aware and able to react to changes in their environment. In this position paper we describe a specific approach for the development of context-aware software. We propose to make existing applications context-aware by means of three main components: context models, high-level rules and code-generation processors. We present each component and analyze the issues related to the development of context-aware software following this strategy.

Arguments Pro. A generational approach has advantages because it enables separation of concerns: the context specific behavior is generated and compiled into the application. Also, the rules that determine when the context specific behavior should be enabled are separated from the base application. The rules encapsulate the adaptations to the base applications.

Arguments Contra. Is it possible to use a generational approach without the requirement that the application has to be refactored? The quality of the base application code affects the efforts that will be required to adapt the code with context information. The proposed use of annotations seems to conflict with the base application developer being unaware of the context adaptations that will be required. Conflicts can arise when different context adaptations affect the same application behavior.

3.2 Reasoning about Past Events in Context-Aware Middleware [6]

The miniaturization of computational devices like for instance mobile phones have caused a revolution in every day life. With the use of a variety of standard technologies like infrared, bluetooth and wifi, we are able to interconnect these devices in a mobile ad hoc network. These technologies bring us closer to the vision of Weiser where persons are surrounded by a cloud of small devices cooperating with each-other and adapting themselves to their context. As these small devices can go out of earshot at any moment in time, these disconnections

cause important changes in the perceived contextual information. The Fact Space Model is a coordination model which gives the user fine-grained control over the effects of disconnection and thus over changes in the perceived context. However, this model does not incorporate the loss of useful information from the past. For example when we have used a service in the past, this information might even be relevant when this service is currently not available. In order to overcome the loss of useful information we have extended our Fact Space Model with temporal operators capturing exactly this relevant information an application might need in the future. In order to let applications better adapt their behaviour to the current context, we advocate the use of this logic coordination language incorporated with temporal operators.

Arguments Pro. Looking back at history is fundamentally a good idea. Looking back at what one has done is important to understand what is relevant for taking the decision of what to do now. Problems such as circular triggering of rules can be easily addressed by preprocessing/analyzing the temporal logic statements.

Arguments Contra. Using logic programming is confusing for end users because they must be able to understand how to describe the behavior of their application. The current approach generates too much mental overhead on the user that defines the rules. Finding a manner to express the rules in a more convenient way is needed. Especially the temporal operators seem unintuitive to interpret.

3.3 Context-Aware Leasing for Mobile Ad Hoc Networks [3]

Distributed memory management is substantially complicated in mobile ad hoc networks due to the fact that nodes in the network only have intermittent connectivity and often lack any kind of centralized coordination facility. Leasing provides a robust mechanism to manage reclamation of remote objects in mobile ad hoc networks. However, leasing techniques limits the lifetime of remote objects based on timeouts. In mobile networks, we also observe that devices need to continuously adapt to changes in their context. In this position paper, we argue that changes in context not only require adaptation in the behaviour of the application but also permeate to distributed memory management, leading to the concept of context-aware leasing.

Arguments Pro. The presented approach contributes to a field that has been mainly focussing on the use of timeouts. Involving context information in the memory reclamation process is interesting because it enables memory management to not only be guided by non-functional concerns but also application-specific details. Another interesting approach is to involve the user in the process of determining which remote objects may be reclaimed.

Arguments Contra. The expressiveness of the presented approach seems limited. Is it possible to use more expressive formalisms to determine whether objects can be reclaimed? The problem is that time is not an expressive means to

define memory reclamation rules. Also, manual memory management strategies have been abandoned in many popular programming languages today. In this regard it seems odd that one is interested to introduce it again because it's an extra burden for the developer.

3.4 AmI: The Future Is Now [2]

Because of the unique nature of the AmI domain, specifically the high amount of industrial involvement in this area, we fear that a classical long-term scenario for the use of academic research is no longer valid. In this paper we argue that the AmI research community should adapt to this context. To do this, we consider a short-term approach, and raise some points for discussion.

Arguments Pro. It is true that in the domain of Ambient Intelligence we are not fast enough to do relevant contributions. The available manpower in academic institutes is limited compared to the available manpower in industrial projects. Another advantage found in industry is that they are focussed on "real world" problems whereas academics sometimes do not have such a focus.

Arguments Contra. The presented perspective of what happens in industrial context is too pragmatic. There are no criteria for evaluating the manner in which problems are solved. It is difficult to compare industrial to academic approaches because in an industrial approach the main goal is make things work whereas the main focus in academic research is to find better methods to make things work.

3.5 Ambient-Oriented Programming in Fractal [7]

Ambient-Oriented Programming (AmOP) comprises a suite of challenges that are hard to meet by current software development techniques. Although Component-Oriented Programming (COP) represents promising approach, the state-of-the-art component models do not provide sufficient adaptability towards specific constraints of the Ambient field. In this position paper we argue that merging AmOP and COP can be achieved by introducing the Fractal component model and its new feature: Component-Based Controlling Membranes. The proposed solution allows dynamical adaptation of component systems towards the challenges of the Ambient world.

Arguments Pro. The approach promotes separation of concerns: adaptation of the component is separated from the business code. As a consequence both the adaptation and the business code can evolve independently from one another. The adaptations are encapsulated at the membrane level. The manner in which the membrane controls the business code and how it can hook into the business code can be based on quantitative approaches found in AOP.

Arguments Contra. It is hard to support independent evolution of the business code and the adaptations because there can be composition conflicts. Furthermore, easy composition of the adaptations with the business code depends

on the quality of the code. Code that has been well structured will be more easy to compose vs. code that lacks the right structures. Another issue is where to put the code that deals with multiple conflicting adaptations. This is a hard problem because of the separation of the business code and the adaptations.

3.6 Dealing with Ambient Intelligence Requirements [8]

Ambient Intelligence is characterized by a heterogeneous and highly dynamic infrastructure. In this paper we present requirements that we identified for developing applications for Ambient Intelligence scenarios. We sketch our own approach based on self-adaptive mobile processes that makes application development a manageable task and fulfills parts of these requirements.

Arguments Pro. The presented work is pragmatic because it combines several established methods to solve problems in a new field. Furthermore, the solution proposes to automate many of the individual steps. For example, the BPEL code that is being generated. Another advantage of this approach is that the code generator can be "trusted" as opposed to individual developers deploying their component in the system.

Arguments Contra. It is not clear whether it is possible to really address the intricacies of such applications at a higher level. For example, where is the development taking place? It is impossible to automate all steps in the development process. Furthermore, it is unclear how the presented model supports evolution of the code. In order to support such a system it is important to have a good mapping between high-level (models) and low-level (code) artifacts. This is especially important in the debugging phase of the application. In the debugging process it needs to be clear whether the cause of the problem is related to the code or the models.

3.7 Proximity Is in the Eye of the Beholder [1]

The notion of proximity is a key to scalable interactions in distributed systems of any kind, both natural and artificial, and in particular in pervasive computing environments. However, proximity as such is a vague notion that can be considered both in a very factual manner (spatial distance) and in a very subjective manner (user affinity). We claim that an adequate system or programming language for ambient intelligence applications ought to support an open notion of proximity, making it possible to rely on different, possibly subjective, understandings of proximity, as well as their combinations.

Arguments Pro. It is good to have a more general notion of proximity. Perhaps physical proximity can be used as a starting point for the objective dimension because physical proximity can rely on a metric system that is widely accepted. Starting from this definition the middleware or language can then support further customization and enable the introduction of other such notions.

Arguments Contra. The dimension of physical proximity is merely one case. One could argue that physical proximity is encompassed in the abstract proximity. The distinction objective vs. subjective definitions of proximity seems good. However, it is unclear whether it is possible to integrate subjective definitions and it seems much more convenient to choose one shared definition.

4 Discussion on Emerging Topics

During the afternoon brainstorm sessions three subjects were selected based on an iterative agreement process. Each participant proposed three subjects. After that three groups of three participants were formed and within these groups the participants had to agree on a single topic. The selected topics and the following breakout discussions are briefly summarized below:

4.1 Context Volatility

Group: Elisa Gonzalez Boix, Christophe Scholliers, Eline Philips, Peter Barron, Jessie Dedecker

The goal of this brainstorm session was to identify the different types of context volatility and to discuss how the software system should deal with its consequences.

Types of Context Volatility

Accuracy. Every sensor has a fault margin and can provide inaccurate readings. Ideally, readings with an unacceptable error margin should be filtered or weighed with the other readings.

Timeliness and Freshness. The frequency by which a sensor is read is important w.r.t. the freshness of the data. Some sensors require a substantial amount of time to be read and can require a substantial amount of energy. As a consequence, continuously reading the sensor is not always possible such that a tradeoff has to be made between the frequency at which a sensor is being read and the timeliness of the sensor data.

Source of Information/Trust. Sensor data delivered from third party components in a distributed context brings the risk that the data can be forged.

Concurrent Transactions in Context. Determining context based on multiple sensor sources that are not necessarily read at the same point in time implies that the determined context can be inaccurate.

Impact of Context Volatility on Software

Where is the volatility exposed? An important consideration is at what level to expose the volatility of context. A first option is to expose it at the lowest level,

where the context is being derived. At this level inaccuracies and faulty readings could be filtered away by weighing the results. However, there is probably no common strategy that fits all application requirements. Another manner to deal with context volatility is to introduce context as a first-class concept in the middleware and provide hooks such that an application is exposed to the context volatility. Regarding context volatility as a first-class concept would involve exposing details such as the sensor specifications w.r.t. accuracy, the sources that were used to derive the detected context information and the sensor and the freshness of the sensor data.

How to deal with concurrent transitions in context? While context-dependent code is executing it is possible that a change in context is detected. Dealing with such concurrent adaptations is not trivial. One option is to introduce the concept of atomic adaptations and process context-dependent actions as non-interruptible events. Another option is to consider context-dependent actions from a transactional perspective. Whenever the context changes during a context-dependent action a roll back could undo previous computations before adapting to the new context.

Approaches to deal with Volatility. When dealing with context volatility at a lower level the history of sensor data becomes an important resource. The history is important because it is useful to determine what sensor data should be considered as faulty and also as a way to improve sensor readings (i.e. by weighing them against previous results). Dealing with context volatility at a higher level could be done with meta -or aspect architectures to separate the context-specific code from the base level. Other options are to use rule-based systems such as logic programming to consider multiple possible context derivations. Fuzzy logic programming could be used to explicitly consider uncertainty in the context rules.

4.2 Responsibilities for AmI Concerns in Development Chain

Group: Carlos Parra, Ales Plsek, Guillaume Dufrene, Philip Mayer

The goal of this brainstorm session was to identify the layers found in AmI infrastructure and to determine the responsibilities of each layer. For the brainstorm session a number of assumptions were made: the AmI system is a distributed system that is based on an event-messaging model and there is an established ontology for context information. The group identified five responsibility layers, which are listed below in a bottom-up order:

Communication in Ambient Environments. At the lowest level, devices are communicating via events. These events disperse observed information about the ambient environment to other devices in proximity of one another. These low-level events contain information that is retrieved from sensors deployed in the ambient environment.

Context Construction. At the next level the events containing low-level sensor information are picked up and combined into context events that are more meaningful and better match the domain of the application. At this level sensor resolution techniques can also be employed to deal with inaccurate sensor readings.

Context Reason. After the establishment of meaningful context events it becomes possible to further reason about this context information and derive the context semantics of the ambient environment.

Application (business logic). The application layer expresses the core semantics of the AmI systems and is influenced by the context reasoning engine that feeds facts about the physical environment to the application.

User. The user layer is responsible for tracking the user's preferences. This information can be either explicit or implicit. Explicit preferences are unambiguously chosen by the user whereas implicit preferences can be derived by observing a user's behavior.

4.3 On User Involvement in Adaptable Applications

Group: Carlos Noguera, Guido Sölduer, Holger Schmidt, Éric Tanter, Ellen Van Paesschen

This group discussed the issue of user involvement in adaptable applications. A lot of work is dedicated to build adaptable programs, but most of the time, this adaptation is foreseen and/or specified by the programmers. How can actual end users be involved in the adaptation process? What consequences does this have on the techniques to adopt for developing the system?

Spectrum of Involvement. User involvement in adaptation can be seen as a spectrum, with fully transparent and oblivious adaptation on one extreme, and at the other extreme, the user has to configure every adaptation explicitly. The first extreme has the disadvantage of being *obscure* for the user, while the latter is *obstrusive*. What is needed is to address a middle ground in this spectrum.

Learning. The group discussed the idea of learning in the adaptation process. First of all, when does learning actually start? It seems that just inference is not enough. From a user perspective, it is interesting that a system can learn by examples and counter-examples. This means using feedback on decisions, in order to determine whether a particular adaptation was well received by the end user. To this end, it is necessary to be able to support explicit declarations by the user. Of course, there is a tradeoff in determining what parts of adaptations can be scripted by the user, and which are not accessible.

Profiles. The group agreed that the profiles, i.e. archetypes, are a good way to abstract away minor variability scenarios by providing a set of possible coarse-grained alternatives. It was recognized that it is important to support both *user*

profiles and *application profiles*. The adaptation process then consists in matching a user profile to an application profile. Problems include: who is the best indicated to find the relevant profiles of real users of a system? how can we define archetypes in complex environments? Also, since all this is about ambient intelligence, there is a need for group-wide, collective, adaptations; in other words, *ambiental profiles*, describing typical ambiental scenarios.

Engineering. Creating adaptable systems therefore requires to consider profiles from the start. Profiles are (or are associated to) sets of transformations or extensions to apply to the system. This clearly refers to variability management, such as product line architectures, with a particular focus on runtime variability. But there are also additional engineering challenges if one wants to deliver system that are capable of "offshore learning", that is, learn new adaptations (and discover new profiles?) once they have been delivered and are used by clients in unforeseen ways and contexts.

5 Conclusion

This third edition of the OT4AmI workshop was particularly lively due to the format including group work sessions. We can see that beyond the particular details of each technical contribution discussed in the presented articles, there is a growing concern for engineering and usability issues, such as how to distribute responsibilities in the process of developing ambient applications, and how to help the user drive context adaptation. As a matter of fact, users are at the center of the ambient intelligence vision, so it is time to develop software that lets the knowledge of end users pervade through the whole system, driving adaptations in a consistent manner. This undoubtedly raises a huge number of challenges, of which only a few have been briefly touched upon during the workshop.

References

1. Barron, P., Dedecker, J., Tanter, É.: Proximity is in the eye of the beholder. In: Mügge, et al. (eds.) [4], pp. 1–6
2. Fabry, J., Noguera, C.: AmI: The future is now – a position paper. In: Mügge, et al. (eds.) [4], pp. 13–18
3. Boix, E.G., Vallejos, J., Von Cutsem, T., Dedecker, J., De Meuter, W.: Context-aware leasing for mobile ad hoc networks. In: Mügge, et al. (eds.) [4], pp. 7–12.
4. Mügge, H., Tanter, É., Cherrier, P., Dedecker, J., Lopes, C., Cebulla, M. (eds.): Proceedings of the 3rd ECOOP workshop on Object Technology for Ambient Intelligence and Pervasive Computing (OT4AmI 2007), Berlin, Germany, Technical Report 2007-12, Technische Universität Berlin (July 2007)
5. Parra, C.A., D'Hondt, M., Noguera, C., Paesschen, E.V.: Introducing context-awareness in applications by transforming high-level rules. In: Mügge, et al. (eds.) [4], pp. 19–25

6. Philips, E., Scholliers, C., Herzeel, C., Mostinckx, S.: Reasoning about past events in context-aware middleware. In: Mügge, et al. (eds.) [4], pp. 27–32
7. Plsek, A., Merle, P., Seinturier, L.: Ambient-oriented programming in Fractal. In: Mügge, et al. (eds.) [4], pp. 33–38
8. Schmidt, F., Kapitza, R., Franz, J.H.: Dealing with ambient intelligence requirements – are self-adaptive mobile processes a feasible approach? In: Mügge, et al. (eds.) [4], pp. 39–44