Software Engineering Support for the Development of Context-aware, Adaptive Applications for Mobile and Ubiquitous Computing Environments

PhD Thesis Proposal

Nearchos Paspallis

Department of Computer Science, University of Cyprus
P.O. Box 20537, Postal Code 1678, Nicosia, Cyprus
Tel: +357-2289 2684, Fax: +357-2289 2701
nearchos@cs.ucy.ac.cy

Abstract. The main motivation for this research is the increasing popularity received by mobile and pervasive computing applications in combination with the increased complexity which is involved in the development of software systems targeting such environments. In this respect, two basic dimensions are detected: environment *sensing* and environment *shaping*. The former refers to the ability of systems to monitor their environment with the purpose of collecting information which is relevant to their operation. The latter, refers to the ability of systems to adapt themselves and their environment by means of adapting their own operation, or instructing changes in other devices they control (e.g. actuators embedded in the environment). Evidently, a critical requirement for facing and tackling these challenges is to glue the *sensing* and *shaping* parts together, by means of appropriate algorithms for reasoning on the environment situation and deciding on the appropriate adaptations. While these subjects are already quite complex on an individual level, developing suitable adaptive applications tackling all three of them increases the complexity substantially. The primary objective of my research is to encompass the complexity involved in the development of context-aware, adaptive systems targeting mobile and pervasive computing environments by introducing and describing appropriate software engineering techniques and algorithms.

1 Introduction

Today, one can observe an ever increasing trend in the use and proliferation of mobile systems. Furthermore, there is a tremendous shift towards designing and building ubiquitous computing systems, which aim at seamlessly improving the *Quality of Service* (QoS) observed by the end-users. This has inevitably affected the design and implementation of modern software systems targeting such environments. However, it is argued that the complexity involved in these systems renders the effort required for designing and implementing such context-aware, adaptive systems prohibitively high. It is also argued that as the development complexity grows even further, the boundary

between cyber-space and real world becomes increasingly obscured. Evidently, new development approaches and tools are needed to mitigate the increased complexity. As *Paul Horn* [1] elegantly expresses it in IBM´s autonomic computing manifesto, "*The obstacle* [for developing software systems] *is complexity. Dealing with it is the single, most important challenge facing the I/T industry. It is our next Grand Challenge*".

For instance, additional functionality in terms of context awareness and adaptive behavior is now a feature commonly desired (but less frequently found) in modern computing systems. While the adaptive-behavior implies the capability of a system to run in a number of different configurations or modes, context-awareness refers to its ability to dynamically perceive the characteristics of its surrounding environment. The ultimate benefit is provided by mobile systems which are capable of monitoring and exploiting the contextual information, to automatically infer decisions on evaluating and choosing the optimal adaptation. This process is typically guided by the aim for maximizing the quality of service which is perceived by the end-users.

To better facilitate the understanding of this idea, consider the following example. Assume you use a smart-phone, which has access to your daily agenda. Naturally, many people would like their phones to automatically switch to the most appropriate profile based on their status (i.e. according to their agenda). Designing such an application though, implies that you need to design an appropriate context monitoring and management system (i.e. to monitor the agenda entries), and an appropriate adaptation mechanism (i.e. to apply the selected profile). Of course these two must be complemented by an appropriate logic module which implements the algorithm which makes the evaluation of context and selects the right profile. As there is an easily observed pattern in such context-aware, adaptive systems, one can safely assume that software engineering models and tools can greatly facilitate the development of such adaptive applications.

As a second example, consider the case where one's living room is engineered according to the ubiquitous computing paradigm. For instance a set of devices and appliances are embedded in the living room. Furthermore, assume these devices are equipped with the required means for communicating with each other and forming arbitrary synergies (e.g. the TV's sound can be directed to any combination of the TV set's own speakers, the stereo system's speakers, or the user's wireless headset). Designing the software for a newly added device, such as an adaptive digital picture frame, which can truly mesh into the environment, becomes quite challenging. Such an endeavor requires not only a set of globally accepted protocols and standards, but also sufficient means of software engineering support, such as predefined models, software libraries and middleware tools.

2 Examining the problem

As a means of better understanding the requirements imposed by the challenge of designing and building context-aware, adaptive software systems for mobile and ubiquitous computing environments, we introduce a model which aims at capturing all the important aspects of it.

Viewing the problem from a *context-causes-adaptation* perspective (see Figure 1), for each adaptive system there are three main factors of interest: the internal and external conditions of the system (also simply referred to as *context*), the adaptation points in the system and the algorithms used to decide the optimal adaptations. The context is monitored by a process which is generally called *environment sensing*. The situation is continuously evaluated and whenever an improving setting is detected, a process is initiated which affects a set of predefined adaptation points. This process is depicted in this figure as the *environment shaping*.

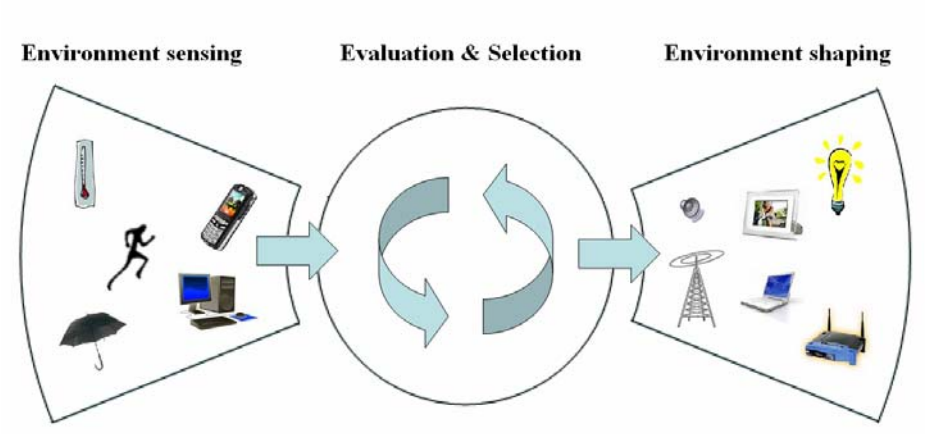

Figure 1: Viewing the problem from a context-causes-adaptation perspective

The environment sensing is today a hot research topic in the academic community. Many researchers are investigating approaches for enabling context awareness properties in their software. *Anind Dey*, one of the most influential contributors to this area, defined context as "[context is] *any information that can be used to characterize the situation of an entity; an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*" [2].

However, the main potential of context awareness does not lie simply in the detection of context, but in the exploitation of the detected information in forms which adapt the extra-functional characteristics of the application, thus improving the quality of the offered service as it is experienced by the end user. This capability was originally detected by *Mark Weiser* in his seminal work at Xerox PARC [3], where he introduced the concept of *Ubiquitous Computing*. The basic idea behind ubiquitous computing is the fact that the ratio of computers (and processor-equipped devices) which can potentially offer services to a single user is constantly increasing (from values a fraction of the unit, to multiples of it), and as a result we are close to the point where the users will be simply overwhelmed by the demand for interaction. In this respect, future systems are expected to be designed so that they monitor as much information about the environment as possible, and automatically and autonomously respond to the stimuli in a manner which optimizes the utility offered to the users while at the same time minimizing the required interaction.

In order to enable such a behavior though, and in addition to enabling *environment sensing*, the developers are also expected to enable *environment shaping* and provide appropriate algorithms for implementing the dynamic *evaluation and selection* of the most suitable configuration. Typically, the environment shaping is achieved through adaptations, either of computing systems, or of devices which are controlled by embedded processors. The two basic possibilities for adaptation are the *parameter* and the *compositional* adaptations. These, along with answers to *where*, *when* and *how* adaptations occur are very thoroughly examined by *McKinley et al* [4].

Finally, having established the mechanisms which enable environment sensing and shaping, the last requirement is to provide a mechanism and the required algorithms for evaluating the sensed context in order to evaluate the possible options and select the one which optimizes the offered utility. Although this is a very popular topic for disciplines such as control theory and artificial intelligence, from a software engineering perspective there are primarily three basic approaches available, namely: *action-based*, *goal-based* and *utility function-based* [4, 5].

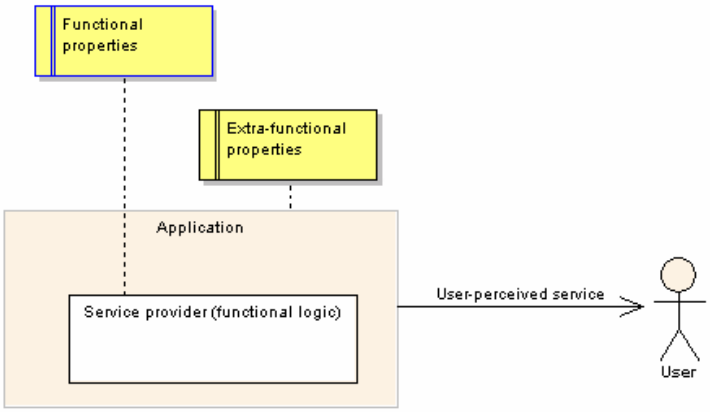

Figure 2: Experiencing the functional and extra-functional properties of a service from a single-user's perspective

An interesting question raised when examining such problems, is *what does QoS imply in this case?* Or even more important, *who is the beneficiary of the QoS optimization?* As ubiquitous computing environments can be extremely complex systems involving many more than a single user, an answer to this question can be quite difficult to be provided. However, in this research we adopt a relatively simple assumption: Each adaptive system (both software and hardware) is designed so that it optimizes the QoS offered to a single user whenever possible (i.e. its owner if it is a smart-phone or a home appliance such as a TV). When a system is designed to offer services to more than a single user, then the adaptation algorithms aim to optimize the service to a wider-audience (e.g. the employees of a company, or the public in general) while at the same time trying to optimize their own resource consumption.

Moving one step back from the perspective depicted in Figure 1, one can argue that from a single user's point of view, a service is perceived as the concurrent perception of both the *functional* and *extra-functional* properties of the service offered by a system or application. For example, consider a scenario where a user uses a smart-

phone to perform some typical tasks such as synchronizing her or his agenda. In the background, the device could automatically adjust the brightness of the screen according to the environmental light conditions, as they are perceived by an installed light sensor. Additionally, the device could automatically decide whether to connect over a WiFi hotspot which is available in the area or via the more widespread but costly GPRS. These decisions and the resulting adaptations are all taken automatically and autonomously, thus minimizing the interaction required with the user. At the same time, the user continues to experience the same functional properties in the service (i.e. view the updated agenda items), while at the same time experiencing the improvements in its extra functional properties (e.g. faster synchronization over WiFi compared to GPRS, and brighter display which better matches the sunlit environment she or he is in at the time).

Of course, in real world scenarios, things get much more complex. Most notably, there is rarely a single user only, so that the whole environment sensing and shaping can be centralized around her or him. For example, the air-conditioning of an office which is shared by many colleagues cannot be tuned according to the preferences of a single occupant only. Additionally, many resources can not always be dedicated to the benefit of a single user only, but rather they must be simultaneously distributed over a number of users. This renders the algorithms which are required for deciding the desired adaptations and consequently the distribution of resources, significantly more complex. For example, an operating system shares the CPU-time between the running applications, but the exact scheduling algorithm can depend on many extra-functional requirements such as the importance of each application. Considering the case of ubiquitous computing environments, where interoperability between multiple systems is typically needed and the scenarios are much more complex, the task of developing appropriate software systems is undoubtedly rendered significantly harder.

3 Software engineering challenges

The principle idea of this thesis proposal is to show that software engineering approaches, in the form of models, middleware tools and development methodologies can greatly facilitate the development of complex, context-aware and adaptive applications and software systems in general. In this respect, these individual aspects of software engineering are further examined.

In our setting, we consider the two most widespread used technologies for software abstraction and encapsulation: *components* [6] and *services* [7]. Component orientation views software systems as consisting of modules which can be independently deployed and which interact with each other through well defined interfaces (ports). The second refers to deployed systems, which offer services over well defined interfaces, and which can be discovered, composed and used.

In the literature, the component-based design approach is frequently described as one of three main technologies supporting compositional adaptation (the other two being computational reflection [8] and *Separation-of-Concerns* [9]). Additionally, the widespread use of middleware technology is also typically attributed as one of the major catalysts for enabling compositional adaptations [4].

3.1 Models

Referring back to Figure 1, one can easily notice a pattern which is common in most context-aware, adaptive systems. This pattern implies an adaptation logic-centered view, where an appropriate mechanism exploits the provided context information with the purpose of evaluating and selecting the most suitable adaptation. Evidently, in order to enable reuse, the sensed information and the adaptation points must be both modeled in an appropriate manner. The context, for instance, must be modeled so that it enables easy and consistent evaluation by any software system. Naturally, consistent models across multiple platforms, is a crucial requirement for enabling interoperability. The latter is commonly accepted as a mandatory requirement for enabling the vision of ubiquitous computing.

Moving on to the adaptation points, further modeling approaches are also required for enabling the dynamic evaluation of the adaptive system. For instance, reconsider the example with the agenda synchronization on a PDA. This example demonstrated two adaptation (or variation) points: the selection of the network adapter to be used (WiFi versus GPRS) and the selection of the actual brightness level for the device's display. Naturally, more variation points will exist either related to this or other applications. In either case, software developers expect support in modeling these variation points so that they can model arbitrary systems in an intuitive and effective manner.

Once the relevant context and the variation points of a software system are detected and modeled, then appropriate algorithms and mechanisms are required for implementing the continuous monitoring, evaluation and selection of the optimal state for the system. Naturally, many such solutions exist and many more can be suggested. Furthermore, assuming consistent modeling over the context and the variation point dimensions, one can easily conclude that these algorithms and mechanisms should be reusable.

3.2 Middleware tools

The next challenge in providing software engineering support for the development of context-aware, adaptive systems is the design and provision of appropriate tools in the form of middleware. Middleware refers to a layer of reusable components in a system which can facilitate the development and deployment of specialized software by abstracting away from the developers many common tasks and functionalities such as distribution, transaction support, directory services, *etc* [10].

In the scope of context-aware and adaptive systems, middleware architectures can play a crucial role. For instance, considering again Figure 2, middleware architectures can provide significant services in the form of reusable mechanisms in all three aspects of context-aware, adaptive applications. For instance, many reusable context-management architectures have been proposed in the literature, thus rendering the design and the implementation of a new context system unnecessary. The same applies with the evaluation and selection algorithms and mechanisms, as well as with mechanisms which enable adaptivity.

3.3 Development methodologies

The last challenge that is examined with relation to software engineering support for the development of context-aware, adaptive systems is the use of development methodologies. It is argued that structured, well defined methodologies, in combination with appropriate models and middleware support, can significantly ease the task of developing such complex software.

One of the most promising directions towards the development of context-aware, adaptive systems is that of the *separation of concerns*. This approach was conceived as early as in the beginning of the seventies, with *David Parnas*' seminal paper on decomposing systems into modules [9]. However, this approach is contemporarily important even to our days, as it provides an undisputable benefit: it enables the mitigation of complexity in composite systems, by enabling the developers to tackle one aspect of the problem at a time.

As it was already argued, from the user's point of view the service is experienced as a combination of both the functional and extra-functional properties of the system (refer back to Figure 2). Naturally, this provides a great opportunity for relaxing the effort involved in the development of complex, context-aware and adaptive systems, by providing the developers with the appropriate means for tackling one aspect at a time (i.e. handle the development of the functional aspects of the system independently of the specification of its adaptive behavior).

4 Current results and future work

In this section, we discuss the current results that have been achieved already, as well as pointers to the future work plans. While some of the existing results have been published in relevant conferences or workshops already, some others are in the process of being submitted or prepared for submission.

With respect to modeling, some initial attempts at proposing a simple, yet functional context model were documented in [17] and [18]. This work is currently reconsidered, with the goal of introducing a three-layer model: providing a conceptual model (UML-based), a context-exchange model (XML-based) and a functional model (Object-based). It is argued that such a model will facilitate the development of context-aware applications (conceptual model), improve on the interoperability (context-exchange model) and improve the functional aspects of automated context management (functional modeling).

With respect to developing variability models, an initial approach was documented in [15], where an annotation-based approach was defined, allowing the specification *roles* offered by components, and of *variation points* which can utilize alternative roles. Future work includes an extension and improvement of this initial model, as well as its evaluation through a comparison with other variability modeling approaches such as *plan-based* approaches [19].

With regards to adaptation reasoning (evaluation and selection of adaptation), current work has attempted to provide models which can abstract the extra-functional characteristics of the computing resources, and based on that and on context

information provide algorithms for performing the evaluation. These algorithms are primarily utility-based. The results of this work are documented in [11] and [13].

Moving on to the middleware challenge, an extensive research was undertaken in the frame of distributed context management. This work proposed a distributed context management mechanism, which provided application developers with the capability of automatically exploiting distributed context data in their applications, in a seamless manner (i.e. without any differences as compared to utilizing local context information). The results of this work are documented in [12]. As this approach involved an ad-hoc communication approach, further research is underway at extending it to utilize peer-to-peer communication, using JXTA as an underlying middleware.

Furthermore, at the middleware layer, an approach was examined which aims at optimizing the exchange of context-exchange messages across collaborating, adaptive systems. To this end, systems are provided with additional information which enables them to assess whether particular context change events would be of interest to a collaborating node or not. In this way, the communication of messages can be optimized by avoiding communicating every context change, which results in saving valuable resources such as battery and CPU time. Initial results from this research were published in [14], and at the moment further research is underway, primarily in the terms of experimental results and evaluation.

Finally, moving on to the methodology challenge, an initial attempt was published in [15], where an approach for developing adaptive mobile applications with separation of concerns was proposed. This approach has already been extended, featuring context-awareness, and a more fine grained description, and has been submitted to an invited book chapter call. The latter describes a methodology for developing context-aware, adaptive applications with separation of concerns, and discusses detailed steps involved in the methodology, such as defining *where*, *when* and *how* adaptations are applied. However, this methodology is currently mostly oriented towards component-based systems. Extending this approach to include planning and decision making for Service-oriented architectures, as well as incorporating these results in the existing methodology is also planned as future work.

5 Conclusions

This document proposes a research path for fulfilling the requirements of my PhD thesis. In this respect, it introduces the problem which is incurred in the development of complex context-aware, adaptive systems, targeting deployment in mobile and ubiquitous computing environments. To this extend, it is argued that a significant challenge for software engineers is to propose appropriate solutions, supporting and easing the development of such systems.

The proposed research path suggests work in three main areas. First, *modeling support* for abstracting and encoding the contextual parameters of a system, as well as its variability characteristics. Second, provide *middleware support* for enabling reusability of methods and mechanisms, which can further facilitate the development and deployment of complex adaptive systems. Finally, it proposes the specification

and analysis of appropriate development methodologies which can provide structured and well-defined guidelines for software developers, thus mitigating the complexity of their task.

Finally, this document discusses the results that have been already achieved in each of these areas, and also provides information about the current and future work planned. Finally, it is argued that these three topics are complementary and, consequently, providing concrete and sound results in all three of them can significantly facilitate the development of context-aware, adaptive mobile and ubiquitous computing systems, and trigger the research, extension and improvement of these results even further.

References

1. Paul Horn, "Autonomic computing: IBM's Perspective on the State of Information Technology", International Business Machines (IBM) Corporation, New Orchard Road, Armonk, NY 10504, October 2001, http://www.ibm.com/research/autonomic.
2. Anind K. Dey, "Providing architectural support for building context-aware applications", PhD Thesis, College of Computing, Georgia Institute of Technology, December 2000.
3. Mark Weiser, "Hot Topics: Ubiquitous Computing," IEEE Computer, Oct. 1993, pp. 71-72.
4. Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, Betty H. C. Cheng, "Composing adaptive software", IEEE Computer, Jul. 2004, Vol. 37, No. 7, pp. 56- 64.
5. William E. Walsh, Gerald Tesauro, Jeffrey O. Kephart, Rajarshi Das, "Utility Functions in Autonomic Systems", International Conference on Autonomic Computing (ICAC), New York, NY, USA, 17-18 May 2004, pp.70-77.
6. C. Szyperski, "Component software: beyond object-oriented programming", ACM Press/Addison-Wesley Publishing Co., 1998.
7. Mike P. Papazoglou, Dimitrios Georgakopoulos, "Introduction to Service Oriented Computing", Communication of the ACM, Vol. 46, No. 10, Oct. 2003, pp. 24-28.
8. Pattie Maes, "Concepts and Experiments in Computational Reflection," Object-Oriented Programming Systems, Languages and Applications (OOPSLA), Orlando, Florida, United States, October 04-08, 1987, ACM Press, New York, NY, pp. 147-155.
9. David L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," Communications of the ACM, Vol. 15, No. 12, Dec. 1972, pp. 1053-1058.
10. Cecilia Mascolo, Licia Capra and Wolfgang Emmerich, "Mobile Computing Middleware", Advanced lectures in networking (NETWORKING 2002), Pisa, Italy, 2002, Vol. 2497, Springer-Verlag, pp. 20-58.
11. Mourad Alia, Svein Hallsteinsen, Nearchos Paspallis, and Frank Eliassen, "Managing Distributed Adaptation of Mobile Applications", 7th IFIP

International Conference on Distributed Applications and Interoperable Systems (DAIS), Paphos, Cyprus, June 5-8, 2007, Springer Verlag (to appear).
12. Nearchos Paspallis, Avraam Chimaris, George A. Papadopoulos, "Experiences from Developing a Context Management System for an Adaptation-enabling Middleware", 7th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS), Paphos, Cyprus, June 5-8, 2007, Springer Verlag (to appear).
13. Mourad Alia, Viktor S. W. Eide, Nearchos Paspallis, Frank Eliassen, Svein Hallsteinsen, George A. Papadopoulos, "A Utility-based Adaptivity Model for Mobile Applications", IEEE International Symposium on Ubisafe Computing (UbiSafe), Niagara Falls, Ontario, Canada, May 21-23, 2007, IEEE Computer Society Press (to appear).
14. Nearchos Paspallis, George A. Papadopoulos, "Distributed Adaptation Reasoning for a Mobility and Adaptation Enabling Middleware", 8th International Symposium on Distributed Objects and Applications (DOA), Montpellier, France, Oct 30-Nov 1, 2006, Springer LNCS 4277, pp. 17-18.
15. Nearchos Paspallis, George A. Papadopoulos, "An Approach for Developing Adaptive Mobile Applications with Separation of Concerns", 30th Annual International Computer Software and Applications Conference (COMPSAC), Chicago, IL, USA, September 18-21, 2006, IEEE, pp. 299-306.
16. Nearchos Paspallis, George A. Papadopoulos, "An Architecture for Highly Available and Dynamically Upgradeable Web Services", 15th International Conference on Information Systems Development (ISD), Budapest, Hungary, August 31–September 2, 2006, Springer, pp. 147-160.
17. Marius Mikalsen, Nearchos Paspallis, Jacqueline Floch, Erlend Stav, George A. Papadopoulos and Pedro A. Ruiz, "Putting Context in Context: The Role and Design of Context Management in a Mobility and Adaptation Enabling Middleware", International Workshop on Managing Context Information and Semantics in Mobile Environments (MCISME) in conjunction with the 7th International Conference on Mobile Data Management (MDM), Nara, Japan, May 9-12, 2006, IEEE Digital Library, pp. 76-83.
18. Marius Mikalsen, Nearchos Paspallis, Jacqueline Floch, Erlend Stav, Avraam Chimaris, George A. Papadopoulos, "Distributed Context Management in a Mobility and Adaptation Enabling Middleware", 21st Annual ACM Symposium of Applied Computing, Track of Dependable and Adaptive Systems (SAC), Dijon, France, April 23 -27, 2006, ACM, pp. 733-734.
19. Kurt Geihs, Mohammad U. Khan, Roland Reichle, Arnor Solberg, Svein Hallsteinsen, Simon Merral, "Modeling of Component-based Adaptive Distributed Applications", Dependable and Adaptive Distributed Systems (DADS Track) of the 21st ACM Symposium on Applied Computing (SAC), Bourgogne University, Dijon, France, Apr. 23-27, 2006, pp. 718-722.