Why Context Matters in a Service-based Architecture for Mobile Devices

Andreas Zeidler

Databases and Distributed Systems Group
Department of Computer Science, Darmstadt University of Technology
E-mail: az@informatik.tu-darmstadt.de

Abstract

In order to interact with its mobile clients in an ubiquitous and smart fashion, a "mobility-supporting infrastructure" has to use context information *about its own and the user's state. Based on context information, actions have to take place which services are offered to mobile clients. In this paper we discuss the various level of abstraction where* context *plays an important role for designing smart applications as well as smart infrastructures which benefit from using context information. We will show that* context *affects all parts of a system, from the link layer to the application layer. Therefore,* context *is ubiquitous in ubiquitous computing.*

Keywords: *Context-Awareness, Ubiquitous Computing, Smart Infrastructures, Mobile Computing.*

1 Introduction

What would you expect when you are using an application on a mobile device? The application should perform its task as reliably and trustworthy as the same application designed for the use on a desktop PC. There should be no need to reconfigure and adapt the device and the applications whenever the situation the device is used in, changes. But nowadays, there are only a few mechanisms providing for automated adaption to the network, finding services in the surrounding infrastructure, and re-configuration of the device and its applications for using the "right" services. Currently used operating systems and middleware are far from being easily usable in changing environments. To fill this gap we propose a layer of software between the operating system and the user helping to *adapt the device to the infrastructure and the infrastructure to the device* by finding surrounding services in an intelligent way, *configuring the device* to use the services found according to predefined *policies-of-use* and save users from the ever-recurring task of reconfiguration, both, on the network layer and the application layer.

Further down the road we assume that network technologies will be ubiquitous [5]. The dream or nightmare of constantly being "online" will come true and the cost of communication will be low or even drop to zero. New business models will be centered around the notion of *services.* Usage of services will generate a revenue for the provider. From the user's point of view a new problem arises: How does a user deal with the vast variety of services popping out of the woodwork? More technical questions are: How to decide which service to use? How to make an appropriate re-binding of services every time a user changes the context? How can context be uniformly accessed across different context-aware systems?

In our research project, we are building an architecture for mobile devices providing for automated adaption of the device to the infrastructure and *vice versa*, based on using context information provided by the infrastructure, the device, and the user. Currently, the enabling middleware for this project is Jini from Sun Microsystems [7].

2 Research Topics

Motivated by the introduction, the research topics presented in this paper can be grouped around a single question: *"How can someone use a mobile system without being annoyed by adapting it to changing environments?"*

Keeping the question in mind, several tasks around "context" can be identified:

Link Layer Technology. In mobile environments "connectivity" is very important. Nowadays technology supports two major communication schemas: 1:1, 1:N. Some mobile devices use infrared as link layer technology which is well suited for 1:1-communication, in the near future devices might be using Bluetooth [3], which supports a 1:N-communication pattern. Dependent on the situation communication takes place in, one schema is preferable over the other. When exchanging private data, 1:1-communication is desirable, but not when announcing the agenda of today's meeting. Two points are

important to remark: (a) The communication schema *needed* is dependent on the situation or context, (b) most devices provide only one schema on the link layer. To get around this drawback imposed by the actual communication hardware, a Jini-service is under development mapping among different schemas, based on and extending the approach presented in [1]. After establishing a private communication link between the service and the device, the service transparently maps the 1:1-communication to different schemas. By now, reasonable experiences have been gathered based on establishing communication via infrared between small mobile devices, such as a 3COM Palm Pilot and the Jini infrastructure. How this applies to wireless broadcasting technology is up to be explored in the near future.

Network Transparency. For operating systems, network transparency is partly achievable by using configuration protocols, such as DHCP [4] or TFTP [6]. They provide for the assignment of network parameters, like IP addresses or the name of the Domain Name Server. On the application level more effort has to be invested for the support of roaming users, as proposed in [2], where an architecture for the employment of so called "Application Level Gateways" is presented, providing for the transparent re-binding to connectionless services, such as Mail-servers or HTTP-proxies. In short, a user can configure all network-clients, such as Web-browsers, to "localhost:port" and the application level gateway will take care of submitting the data to the appropriate server in the surrounding infrastructure. Gaining the same level of transparency for services that rely on constantly accessible network connections, like the "Network File System" (NFS), is an ongoing research topic.

Intelligent Service Re-binding. Often, the use of predefined service instances is pointless for mobile devices, e.g., all services using some sort of hardware to fulfil their tasks, such as printers or storage services needed only temporarily. In this case *context* plays a central role. When someone is scheduled to visit a customer to sign a contract and changes to it are made, he or she wishes to printout the modified contract. Using the printer at home is pointless here. More appropriate is the use of a printer near the conference room where the contract will be signed. In this example the notion of *context-awareness* is narrowed to *location-awareness* and the *finding (lookup)* of services of the required *types*. In the more general case, several *contextual parameters* can be identified necessitating a decision which service (out of a set) is to be used in the context a person is in. Our thesis is that transforming these parameters into a context-free description language, which can be used as query-language in the surrounding infrastructure, is a suitable approach to locate services automatically. Examples might be XML or SQL. Currently we are implementing a trader architecture for Jini which allows for submitting arbitrary queries to the Jini infrastructure in order to evaluate different approaches to how to find the *right* service. Often the resolution of a printer is not as important as the physical distance to the printer. Leaving open the rule for the resolution in an unexplored location (like in the example above) might be the correct query; Leaving it open when printing out holiday pictures might produce unacceptable results.

Not all decisions can be made by a subsystem of the mobile device. Some decisions are inherently "human related", e.g., which kind of food is favoured on Fridays. So, additional tools are needed to formulate *policies-of-use*, describing which *selection of services* is actually presented to the user, when asked to do so. Here we propose a pragmatic approach comparable to the WYSIWYG approach found in many HTML editors: The user should not be bothered with details of the underlying description-language, but should formulate the preferences based on templates and symbolic choices. The outcome might not be perfect, but a sufficient one. Currently, a working prototype is under development.

Application- and Presentation-Layer. In different environments, it is not guaranteed that all environments offer the same services to the user. This is a major difference to "classical" mostly *static* networked environments with desktop PCs. In mobile scenarios there must be no need to install any special-purpose "drivers" or software, as these pieces of software are only valid in one particular environment. Furthermore, services with the same name do not always follow identical semantics. As we mentioned above, the overall goal is to guarantee that the device is functioning as good as possible independent of the situation the device is used in. Hence, finding, evaluating, and presenting information about services and how to use them is of fundamental importance. Here we use Jini as it provides *mobile code* (state and implementation of an object are moved to a client and executed there). The effect of this is that no special purpose drivers are needed as the needed functionality can be "plugged in" easily on the mobile device. Furthermore, services can bring along their own GUIs when moved to a client. So, a user can use a

service without prior knowledge about it. This concept and its consequences were tested in a laboratory course given recently. The outcome was promising (as successful) but more difficult as expected. Parts of the system have to be assessed automatically using *policies-of-use* and therefore *conventions and guidelines* are needed of how to describe services uniformly.

Interoperability Mobility imposes special requirements on interoperability. In ubiquitous computing, it can be assumed that different context-aware infrastructures have to co-exist and, given this, interoperability is of supreme importance. Mobile devices must work in each of this smart infrastructures equally well. To support interoperability, we introduce the model of *Context-Cells* providing for a general model of how context can be *wrapped and accessed* on different level of abstraction and detailedness. Moreover, CCs are *active*, hence well suited for mobile user (and their devices). Just to give an example: A user can have a *personal CC* wrapping all services, their description, and some other information about the user (as XML document). When entering the scope of some infrastructure the user's CC is subsumed under a super-CC offered by the infrastructure (e.g., a car of a train), which itself is subsumed under some other CC dynamically. So, a context-federation is built out of CCs dynamically and the *context dependencies* are obvious. The major advantage of this approach is that it is fully distributed and almost no central configuration and control is needed. Moreover, as personal CCs are mobile and integrating theirselves into surrounding infrastructures, the user is of full control over the information and data leaving or entering his/her personal CC. Situation-dependant *roles* and privacy requirements seem to be realistic using this general model for federating, finding, and accessing different contexts in different infrastructures.

Security. Security is inherent to all issues mentioned above and has to be enforced on all layers of an infrastructure supporting mobile systems. How this can be done is an open problem, due to the fact that a suitable *security model* for spontaneous collections of entities, as found in mobile systems, still has to be developed. To adapt known security models to infrastructures for mobile systems does not seem to be suitable. The main reasons are that they require undesirable structures like "globally unique identifiers" and globally federated key-authorities making public-keys of everyone accessible everywhere by everyone else. As a consequence, a security model must be developed based on a more locally and anonymous notion of "trust". How this applies to public networks (e.g., on trains) is questionable and beyond the main scope of this research group. Nevertheless, using "local trust models" can significantly increase security on a workgroup-level. Here, security is much easier to enforce.

3 Summary

We have given a short description of our basic assumptions and main research topics. Starting from the user's point of view, we analysed the basic requirements for an infrastructure supporting mobility in order to give a user a good starting point to feel comfortable while using mobile devices in different contexts. We hope we made clear why and where we are using information provided by the context the mobile device is used in. The overall impression of using a reliable and trustworthy environment cannot be achieved when context is left out of consideration when designing and implementing infrastructures for the future of ubiquitous and mobile computing.

References

[1] Gerd Aschemann, Svetlana Domnitcheva, Peer Hasselmeyer, Roger Kehr, and Andreas Zeidler. A Framework for the Integration of Legacy Devices into a Jini Management Federation. In *Proceedings of the Distributed Systems: Operations and Management (DSOM99)* , pages 257–268. Springer-Verlag Berlin Heidelberg, 1999.

[2] Gerd Aschemann, Roger Kehr, and Andreas Zeidler. A Jini-based Gateway Architecture for Mobile Devices. In *Proceedings of the Java-Informations-Tage 1999 (JIT99)*, pages 203–212. Springer-Verlag Berlin Heidelberg, 1999.

[3] Bluetooth Consortium. The Bluetooth Project. `http://www.bluetooth.com/`, 1999.

[4] R. Droms. Dynamic Host Configuration Protocol (DHCP). Internet RFC 2131, March 1997.

[5] Mark Weiser. The Computer for the Twenty-First Century. `http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html`.

[6] K. R. Sollins. The TFTP Protocol (Revision 2). Internet RFC 783, June 1981.

[7] Sun Microsystems Inc. *Jini Architecure Specification – Revision 1.0*, January 1999.