Heterogeneity in mobile computing environmens

Robert Schmohl, Uwe Baumgarten
Technische Universität München, Department of Informatics, 85748 Germany
schmohl@in.tum.de, baumgaru@in.tum.de

Abstract ***The rapid evolution of mobile computing has spawned a very heterogeneous spectrum of technologies affecting both devices and system in the mobile computing domain. Since mobile computing emphasizes the ubiquitous use of interconnected systems the enabling of interoperability remains a significant requirement, which is considerably complicated by the impact of heterogeneity. In response, research has quickly addressed the issue and elaborated diverse solutions to the problem. This paper surveys the different approaches and extracts the key concepts in handling different heterogeneity aspects. We summarize those concepts by abstracting a general approach on heterogeneity-handling.***

Keywords: mobile computing, heterogeneity, middleware

1 Introduction

Mobile computing is characterized by a high level of heterogeneity, since there are only a few standards, that are commonly obeyed by the device manufacturers, software developers and network providers [1]. This aspect is especially reflected by heterogeneous characteristics of mobile devices, operating system, resources and network capabilities [2].

The goal of mobile computing suggests including devices spanning the entire hardware spectrum [3]. This argumentation includes the appliance of various use cases, including both the pervasive access to mobile services and ubiquitous communication between mobile hosts [1, 4]. Hence, those application cases can be reduced to the basic demand of *communication among heterogeneous devices in heterogeneous environment*. This statement can be refined as follows:

- *Communication among mobile devices*: This basic use case depicts the communication of at least 2 mobile hosts, both capable of roaming in correspondent networks and enabling the mobile devices' users to exploit this communication link.
- *Communication with back-end systems:* In this case, the user employs his mobile device to connect to a network's back-end system. This activity usually triggers a specific workflow on the back-end, e.g. a mobile service delivering a specific response to the requesting user.

Many research groups agree in handling heterogeneity by employing middleware solutions [2, 3, 5, 4]. The common idea behind those approaches is to position the middleware layer between the application layer at the top and the heterogeneous environment at the bottom as displayed in figure 1. Hence, transparent access is provided to the heterogeneous environment by masking the underlaying heterogeneity.

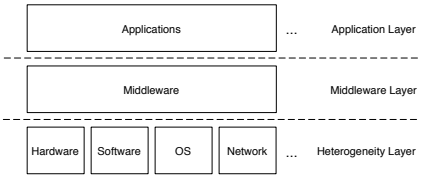

Figure 1: Middleware approaches

The rest of the paper is structured as follows: section 2 decomposes the heterogeneity issue. Section 3 introduces the key concepts to address the heterogeneity aspects abstracted in section 2. Subsequently, architectural realization issues are discussed in section 4 before we conclude the paper in section 5 presenting an abstraction of general approach for handling heterogeneity in mobile computing.

2 Heterogeneity Abstraction

In mobile environments, the problem of heterogeneity concerns a wide range of architectural domains. A simple cut allows the abstraction of heterogeneity into 3 different views [2]:

- *Hardware heterogeneity*: Hardware heterogeneity reflects the presence of different devices with different capabilities, as well as different network technologies integrating those devices.
- *Software heterogeneity*: Software heterogeneity is characterized by the presence of different applications and operating systems.
- *Architectural heterogeneity*: This heterogeneity aspect illustrates environments where network interconnections do not share any common architectural characteristics.

All of those heterogeneity aspects address the problems arising from the endeavor to achieve interoperability among different devices and systems. Interoperability

may be decomposed into the following communication models:

- *Direct communication*: Mobile devices communicate directly with each other. Heterogeneity has a direct impact here, since communication among mobile devices must be based on commonly shared standards.
- *Brokered communication*: The communication link between devices is established by some sort of centralized instance. The challenge here is to make the server being able to address a heterogeneous mass of devices.
- *Unidirectional workflow activation*: A mobile user may trigger a specific workflow on a server by contacting it unidirectionally. In this case, the demand for support of heterogeneous callers applies again.
- *Service provision*: A mobile user employs his device to use services provided by a server by requesting the service and getting an appropriate response subsequently. Technically speaking, this application case describes an extension of the previously described universal workflow activation adding a response after completion of the workflow, so that the service is described by the delivery of responses following client requests. Again, the server must be able to deal with heterogeneous sets of devices.

2.1 Hardware Heterogeneity

Hardware heterogeneity simply reflects different *physical* implementations of the mobile devices' underlaying technologies. Based on that, different capabilities of the particular devices can be derived. Those capabilities allow the heterogeneous set of devices to be structured accordingly.

Those capabilities may be categorized under consideration of the following aspects of hardware heterogeneity, which are of particular interest in mobile environments:

- *Communication interfaces:* Those are the physical components, that connect the device with its surrounding, i.e. enabling access to the network [6, 7, 3]. E.g., prominent network technologies are GSM, 3G-networks, satellite phone networks, 802.11,Bluetooth, etc.
- *UI capabilities:* User interfaces differ widely among mobile devices concerning both input (keypad, microphones, etc.) and output (displays, speakers, etc.) capabilities. Since this aspect concerns the interaction with the user, it aims at providing content to the user and gathering his commands, which can both be device-specific [8].
- *Performance:* Mobile devices are equipped with performance-constrained hardware due to limitations in the power supply and the available space. Although those restrictions apply to most of the mobile device spectrum, the level of restriction differs significantly. The reduction of performance especially concerns capabilities of computation, communication and storage [9].

2.2 Software Heterogeneity

Software heterogeneity concerns the set of *software payloads* on mobile devices. It can be decomposed into 4 categories:

- *Operating systems:* There are numerous operating systems running on mobile devices. They all handle their particular systems tasks differently creating a high degree of heterogeneity on the level of OS-internal architectures. Furthermore, some operating systems provide common native APIs for third-party applications to run on them. Prominent representatives of such systems are SymbianOS [10] and Windows Mobile [11].
- *Middleware APIs:* Some operating systems provide common non-operating-system APIs for third-party applications. Those APIs are designed to supplement the operating system, hence they are shielded from the operating system's core functions making them less powerful than an operating system's native API (if existent). A prominent example for such a middleware API is J2ME [12].
- *Applications:* This abstraction criterion describes the mass of applications available for APIs provided by operating systems and middlwares as stated above.
- *Application domains:* Applications can further be categorized by application domains (e.g. context awareness, mobile billing, etc.). Even though application domains differ among each other, software belonging to a particular domain often shares common principles.

2.3 Architectural Heterogeneity

Architectural heterogeneity addresses differences in any aspects focussing the architectural design of mobile computing systems, of which the following seem to matter most:

- *Network topology:* A network's static and dynamic settings can both differ greatly. Although the static architecture can only differ among complete networks, this aspect is relevant when mobile devices roam *between* such different networks. Dynamically changing network topologies even concern devices roaming *across* one network. Those issues particularly raise availability issues and imply accordingly working hand-over techniques [13].

- *Services*: Heterogeneity of services is reflected by the existence of a large spectrum of diverse services, which can be utilized by mobile devices. It especially concerns the services' protocols specifying access and result delivery [1].

3 Heterogeneity Handling

This section focusses on how to overcome the heterogeneity issues as decomposed in section 2 in order to maximize interoperability. As stated earlier, a key concept in approaching the problem is the design of middleware solutions, which are intended to make heterogeneity transparent. The following subsection discuss various concepts addressing this issue. First, techniques of abstracting heterogeneous data into proper representations are discussed. Subsequently, this information is employed to evaluate diverse operational techniques of overcoming the problems caused by heterogeneity. It is further to be noted, that all of these concepts presented in this section are *not* necessarily disjoint.

3.1 Representation of Heterogeneous Data

Capabilities Heterogeneity defines itself by the presence of different capabilities in the same domains. In order to accomplish interoperability among heterogeneous devices, the first step is often to capture all of their capabilities and structure them into suitable representations. Those representations serve as a data basis for solutions adopting to the device heterogeneity.

There are plenty of representation techniques available, which are suitable for implementing such a representations. 2 examples are given below:

- *Device capability databases:* A DCDB stores the devices' capabilities and provides them to the system, which is communicating with the devices [8, 4]. It especially applies for understanding the devices' requests and providing according content to them. DCDBs are normally implemented as relational databases or using XML. A very neat example for the latter case is the open source project WURFL [14].

- *Ontologies:* A more sophisticated way of storing data is the employment of ontologies. Ontologies allow the representation of data as an interrelated set of concepts [15]. E.g., Christopoulou et al. [6] employ ontologies to describe the characteristics of heterogeneous devices. Just like the DCDBs, those ontologies solely focus on characteristics, which differ among the devices in question.

DCDBs and ontologies serve the same purpose, whereas ontologies may be regarded as a sophisticated refinement of DCDBs. Since both are storing information about devices, they will both be referred as *device databases* from now on.

Intermediary Representations The capability representation techniques stated previously have shown an approach how to reduce a heterogeneous spectrum of data to uniform representations, hence allowing uniform *storage* of heterogeneous data. Analogously, this principle can be applied to facilitate *workflows* of middlewares dealing with heterogeneous environments. In that case, the middleware uses uniform representations of data to allow communication between heterogeneous mobile devices among each other and any back-end system behind the middleware. Such representation represents a generalization of all supported communication semantics from the heterogeneous spectrum. It may be called *intermediary* since it is not feasible for neither any of the devices nor the middlware's back-end services. When communicating with either of those the data is dynamically adapted to the target's specification at runtime using DCDBs as described previously. Concluding, intermediary representations introduce an interoperability layer in the middleware enabling heterogeneous communication through a generalizing approach, such as illustrated in figure 2 (it is to be noted, that the middleware is not necessarily physically decoupled from the mobile device; possible deployment option middlewares are discussed later on).

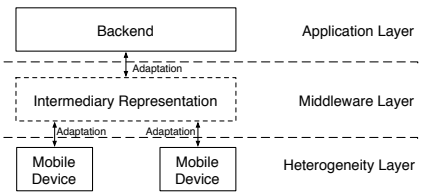

Figure 2: Intermediary Layer

In summary, an intermediary representation allows the middleware to utilize heterogeneity aspects uniformly at runtime. This concept can be particularly exploited for the following application domains:

- *Inter-device communication*: Intermediary representation can be used to represent all aspects relevant for the communication between devices, which differ in characteristics, protocols, etc. [4].

- *Content provision*: The utilization of intermediary representations can facilitate the provision of content to a heterogeneous spectrum of devices by first generating device-independent intermediary content, which is subsequently adapted to the receiving device [8]. This approach emphasizes the entire spectrum of defining content (structure, content, style, etc.).

3.2 Exploiting common Interfaces

One of the most basic approaches in enabling interoperability among mobile hosts is to identify their common interfaces and exploit them accordingly. Those in-

terfaces allow the devices to be very different by hiding their individual heterogeneous characteristics behind their commonly shared interface specification and thus making themselves transparent to each other [7].

This thought is projectable on both heterogeneity domains of hardware and software, as depicted in section 2. Regarding heterogeneous hardware, this simply means that all devices communicate via the same technology and/or via the same protocols. E.g., devices from a large spectrum reaching from desktop computers to small handhelds may communicate wirelessly via 802.11 although being very different individually. The same principle applies for software heterogeneity, where the utilization of common APIs marks the correspondent approach. E.g., J2ME [12] or the Series60 API [16] provide uniform and widely spread interfaces for applications.

However, refinement of the common interface principle allows yet more sophisticated solutions. Common APIs on mobile devices provide a sort of *gateway* to the devices' heterogeneous subsystems to deploy more functionality. In this case, the common API is used to deploy and utilize device-specific implementations on any target device and to provide universal communication to the network [5].

In particular, an installation package, which compliant to the common API is prepared. It consists of a device independent part bound to the API and the device-specific implementations of *all* target devices. Once the installation package is deployed on the target device and executed on the upon the common API the installation routine detects the device and subsequently selects and installs the appropriate device-specific implementation. The resultant operation consists of accessing the device's core functions through the device-specific part and enabling communication to the environment through the device-independent routine.

3.3 Individual Adaptation

Direct communication among two heterogeneous partners requires, that at least one of the communication partners adapts to the other's communication technique. This may include direct inter-device communication as well as communication between a device and a server. Extending this approach on a setting with devices implementing numerous different communication techniques and hence enabling uniform communication results in the demand of general interoperability [4].

Apart from the necessity of using commonly shared hardware technology, which comes along with the direct communication approach, the concept of adapting communication to the current partner dynamically may handle heterogeneity issues in both hardware and software, as depicted in section 2. The following workflow enables this principle [8]:

1. As soon as the communication partner is identified, it is evident which communication standard is needed. Identification is conducted using a device detection mechanism exploiting a device database, which also contains the capabilities of the identified device. Section 3.1 has discussed such data storages.

2. Since the communication partner supports only one of many communication standards, an intermediary layer is inserted between the communicating entities for reasons explained in section 3.1. Incoming and outgoing communication is translated to the intermediary representation first before being committed to the communication partner or the adapting device (its back-end). The intermediary layer allows swift and scalable adaptation to any communication technique necessary.

The application cases of this principle, which is illustrated in figure 3, are various. The approach can be projected onto the client-server paradigm, so that a server adapts to a heterogeneous set of mobile devices, that implement heterogeneous communication techniques [8, 4]. An implementation based on a P2P architecture is imaginable, too, but raises question about facing the peers' low capabilities concerning computation and storage, with are both required for the adaptation process.

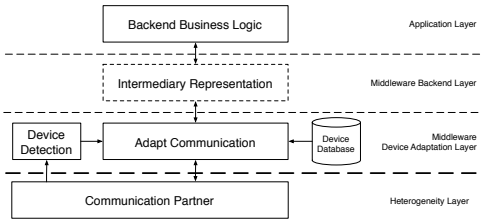

Figure 3: Individual Adaptation to Communication Partner

3.4 Bridges

The basic thought in the approach discussed in this section is the insertion of an entity between at least 2 heterogeneous systems or devices enabling the communication between those. This concept may be allegorically related to *bridges*, since they are intended to connect possibly totally different domains. Since the concept is very basic in nature, numerous instantiations are imaginable. The following discussion categorizes the general application cases into hardware and software bridges.

Hardware bridges: Those are dedicated stand-alone hardware devices negotiating the communication between heterogeneous communication partners. Those bridging devices provide interfaces for each of the supported heterogeneous domains. As an example, consider a laptop as a bridging device for heterogeneous sensors [7, 17]. It is to be assumed, that those sensors may be connected to the laptop through its various interfaces

(e.g. USB, PCMCIA, serial port, etc.). The laptop captures the sensory data, refines them into an appropriate format and commits them to the system's back-end wirelessly via 802.11, thus bridging data transmission from heterogeneous sources to a commonly communicating consumers.

Software bridges: Such as hardware bridges provide interoperability between different hardware domains, software bridges enable the communication between different systems on the software level analogously. For this purpose, consider a *bridging framework*, as described by Nakazawa et al. [4]. Several levels are important on the abstraction layer dealing with software bridging: *Transport-level bridging* involving the translation of protocols and data types inherent in the systems to be bridged, *service-level bridging* including the discovery of new devices in bridged systems and offering appropriate services to them (out of heterogeneous services), and *device-level bridging* handling and translating different device semantics (such as roles and compatibility).

Since all of the bridging levels discussed here are characterized by heterogeneous communication they are eligible for the adaptation mechanism discussed in section 3.3. They need information about how to adapt the communication to the correspondent device, requiring some sort of capability databases. Also, the translation of protocols, data types, services and device semantics requires individual handling for each bridged system. In order not implement a translation technique for each pair of systems, an intermediary representation of information on each bridging level may greatly reduce the translation complexity.

3.5 Mobile Agents

The discussion so far has focussed on the heterogeneity aspects concerning mobile devices' hardware and software (as depicted in section 2). A suitable way of addressing the remaining aspect of architectural heterogeneity is the employment of *mobile agents*. Those autonomous programs usually run on back-end systems (not on mobile devices) and may be used to individually address any arising heterogeneity issue. For example, consider a set of heterogeneous services, which are accessible by mobile devices. Mobile agents can be employed to handle service request for such devices by implementing the following workflow [18, 19]: The mobile agent provides a uniform interface for mobile devices and accepts the their service requests. Subsequently, the agent queries the appropriate service and waits for its result delivery. This is the phase where the adaptation to the heterogeneous mass of service is applied. Finally, the results are returned to the requesting device.

Since the adaptation to the heterogeneous service interfaces is handled by the mobile agent, mobile devices are provided transparent access to a set of heterogeneous services, thus successfully hiding the service heterogeneity from them.

4 Architectural Issues

After having decomposed the heterogeneity issue in mobile computing and having presented approaches to address those in the precedent sections 2 and 3 this section deals with aspects of realizing those approaches.

4.1 Communication Models and Middleware Application

The actual instantiation of the heterogeneity-adressing concepts are dependent on each individual application case. However, most application cases can be projected onto the 4 communication models depicted in section 2 (direct communication, brokered communication, unidirectional workflow activation and service provision). The realization approaches of heterogeneity-handling middleware can roughly be mapped onto those 4 communication principles. They are outlined consecutively grouping the communication models into the generic communication paradigms of *client-server* and *peer-to-peer*.

Client-server Model Communication takes place between a mobile device denoting the client, and a server in the back-end. The heterogeneity-aware middleware may be deployed on the client [7], the server [8] or both [5] depending on the particular use cases. Those can be grouped by the communication models as follows:

- *Unidirectional workflow activation*: Since the communication flow is unidirectional in this case, the server simply needs to offer a set of common interfaces, which clients access to trigger workflows on the server. Individual adaptation to the clients' protocols is rudimentary and is limited to understanding the clients' invocations.

- *Service provision*: Since the communication is bidirectional, device detection occurs so that the server is able to respond accordingly to the client. Subsequently, the requested service is performed by the server's application back-end and the results are committed to the middleware where they are stored in an intermediary format. Since the device is known, the middleware now adapts the service's results to the client's specification and returns them to the requesting device. Hence, individual adaptation must not only be used to understand the client's requests, but also to tailor device-specific responses, as illustrated earlier in figure 3.

The communication models discussed here emphasize the deployment of the heterogeneity-aware middleware on the server only. However, special use cases may require to deploy parts of the middlewre onto the clients as well. Regarding this context it is to be noted that decoupling the middleware from the client potentially increases the range of the correspondent application since there is no need to deploy any middleware components on the devices [8].

Peer-to-Peer Model This communication paradigm emphasizes inter-device communication without strong centralized instances.

- *Direct communication*: Pure P2P networks are characterized by the absence of centralized components. Middleware enabling interoperability among heterogeneous peers is hence deployed on the peers themselves. Basically, this implies that communication partners always have to agree on common protocols prior to communicating with each other. With the exception of the underlaying hardware requirments, the communication protocols are usually provided or complemented by the middleware. Hence individual adaptation and thus the use of intermediate representations are unnecessary since the middleware is the same on each client. Furthermore, middleware on mobile peers has a strong emphasis on the presence of common interfaces and on the adherence of hardware restrictions, which are normally existent on mobile peers (e.g. computation, communication, storage) [9].

- *Brokered communication*: Those brokers allow inter-device communication enabled by a centralized component, usually seen in hybrid P2P networks. The server-resident middleware negotiates the communication between heterogeneous partners [4] as it has been described in the software-bridging concept in section 3.4. Hence, device detection, common interfaces and indivdual adaptation are of significant importance.

4.2 Components

The discussion about approaches to face the heterogeneity issues allows the identification of several components relevant to each heterogeneity-aware middleware [8, 4]:

- *Device database*: A database storing all available information about devices, especially including device capabilities and semantics.

- *Device detector*: A component dedicated to the identification of devices communicating with the middleware. It usually uses the device database for this purpose by matching the identification attribute with the corresponding entry in the database.

- *Intermediary storage*: The intermediate space is used for the device-independent representation of information.

- *Translation adapters*: Those components translate device-specific information to the device-independent intermediary representation and vice versa. Abstractly speaking, there is an adapter for each device group sharing common capabilities.

- *Common interfaces*: Middlewares offer a set of interfaces, which are commonly accessed by the heterogeneous communication partners. They are a simple approach to leverage transparent access on heterogeneous technology.

4.3 Workflow

From the discussion of the heterogeneity handling strategies in section 2 and the derived generic components from the previous section 4.2, the following generic workflow can be conceptualized for a heterogeneity-aware system:

1. *Device detection*: The first step consist of identifying the heterogeneous communication partners, i.e. detecting the devices used. This is done by matching the devices' identification attribute with the corresponding entry in the device database.

2. *Interface selection*: With the device capabilities known, its interfaces are known as well. Hence, the next step consists of selecting the proper interface for the initiation of the logical communication link.

3. *Device-specific translation*: The device-specific output is translated to the system's device-independent intermediary representation and vice versa by using the corresponding translation adapters device-specifically. This translation occurs either unidirectionally or in both directions depending on the communication models depicted in section 2. E.g., for service provision only intermediary content is adapted to the target's specifications whereas direct and brokered communication require translations both to and from intermediary representations.

5 Conclusion

In this paper, we have presented and evaluated techniques how to address heterogeneity aspects, which are present in mobile computing environments. From those approaches, we can derive a general architecture implementing those approaches as illustrated in figure 4.

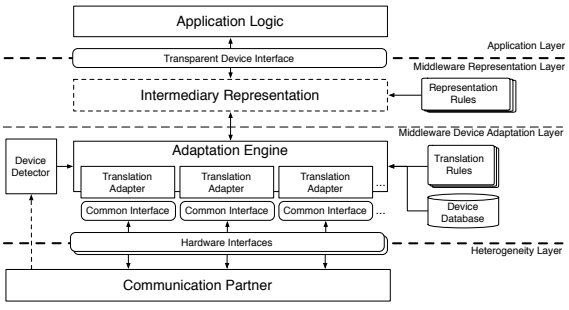

Figure 4: General Architecture for heterogeneity-aware Middleware

The workflow to enable interoperability among heterogeneous devices and systems is basically inspired by the steps depicted earlier in section 4.3. First, the *device*

detector identifies the partner device by matching the device's identification attribute with the corresponding entry in the *device database*. With this information available the middleware is able to negotiate communication between the *heterogeneous device* on the heterogeneity layer and the *parent application* on the application layer. On the top, it provides a *transparent device interface* to applications, hiding all heterogeneity issues from them. On the bottom, it utilizes native hardware interfaces to connect to the communication partner. The middleware decomposes into 2 layers:

- *Device adaptation:* We have labeled the core component on this level the *adaptation engine*, which adapts all communication to the heterogeneous device. Each of its translation adapters is bound to a common interface present on the deployment system. It utilizes the device information from the device database to query information on how to actually adapt the communication to the partner device. For reasons of modularity, the translation rules are kept separately and interpreted at runtime. This approach allows the adaptation engine to handle heterogeneous devices through the common interfaces (bottom) and to provide a device-independent interface to the representation layer (top).
- *Data representation:* This layer's purpose is to make all information send through the middleware uniform and device-independent. It therefore uses an intermediate representation format.

The architecture presented here allows communication with heterogeneous devices. The application logic on top of the transparent interface may be instantiated as applications in many domains, including communication, services and interoperability. All of the application cases may profit from neglecting heterogeneity issues addressed by the middleware.

References

[1] Robert Schmohl and Uwe Baumgarten. Mobile services based on client-server or p2p architectures facing issues of context-awareness and heterogeneous environments. In *PDPTA '07*, pages 578–584. CSREA Press, 2007.

[2] Ricardo Couto A. da Rocha and Markus Endler. Evolutionary and efficient context management in heterogeneous environments. In *MPAC '05*, pages 1–7, New York, NY, USA, 2005. ACM Press.

[3] Martin Modahl, Ilya Bagrak, Matthew Wolenetz, Phillip Hutto, and Umakishore Ramachandran. Mediabroker: An architecture for pervasive computing. *percom*, 00:253, 2004.

[4] Jin Nakazawa, H. Tokuda, W.K. Edwards, and U. Ramachandran. A bridging framework for universal interoperability in pervasive systems. In *ICDCS 2006*, pages 3–3, 2006.

[5] Ellick Chan, Jim Bresler, Jalal Al-Muhtadi, and Roy Campbell. Gaia microserver: An extendable mobile middleware platform. *percom*, 00:309–313, 2005.

[6] Eleni Christopoulou and Achilles Kameas. Gas ontology: An ontology for collaboration among ubiquitous computing devices. *International Journal of Human-Computer Studies*, 62(5):664–685, May 2005.

[7] Christian Bartelt, Thomas Fischer, Dirk Niebuhr, Andreas Rausch, Franz Seidl, and Marcus Trapp. Dynamic integration of heterogeneous mobile devices. In *DEAS '05*, pages 1–7, New York, NY, USA, 2005. ACM.

[8] Robert Schmohl, Uwe Baumgarten, and Lars Koethner. Content adaptation for heterogeneous mobile devices using web-based mobile services. In *MoMM2007*, pages 77–86. Oesterreichische Computergesellschaft, 2007.

[9] Roy Want and Trevor Pering. System challenges for ubiquitous & pervasive computing. In *ICSE '05*, pages 9–14, 2005.

[10] Symbian developer network. http://developer.symbian.com/.

[11] Microsoft windows mobile. http://www.microsoft.com/windowsmobile/.

[12] The java me platform. http://java.sun.com/javame/.

[13] Marcello Cinque, Domenico Cotroneo, and Stefano Russo. Achieving all the time, everywhere access in next-generation mobile networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):29–39, 2005.

[14] Luca Passani and Andrea Trasatti. Wireless universal ressource file. http://wurfl.sourceforge.net.

[15] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *UbiComp 2004*, 2004.

[16] Series 60 platform for symbianos. http://www.forum.nokia.com/s60.

[17] Olga Volgin, Wanda Hung, Chris Vakili, Jason Flinn, and Kang G. Shin. Context-aware metadata creation in a heterogeneous mobile environment. In *NOSSDAV '05*, pages 75–80, New York, NY, USA, 2005. ACM.

[18] V. Baousis, E. Zavitsanos, V. Spiliopoulos, S. Hadjiefthymiades, L. Merakos, and G. Veronis. Wireless web services using mobile agents and ontologies. In *2006 ACS/IEEE International Conference on Pervasive Services*, pages 69–77, 2006.

[19] Iulian Radu and Son T. Vuong. Nemos: Mobile-agent based service architecture for lightweight devices. In *SWWS '07*, 2007.