Experiences from Ramping Up an Environment for Mobile Information Access

Thomas Strang, Jens Kammann, Patrick Robertson, Michael Angermann, Thaddaeus Dorsch, Christian Wasel and Kai Wendlandt
German Aerospace Center (DLR), Oberpfaffenhofen, Germany
{firstname.lastname}@dlr.de

ABSTRACT

This paper presents a comprehensive summary of the insights and experiences which a team of computer scientists and communications engineers gained during the ambitious three-year project *Heywow* in the field of mobile and ubiquitous computing. The project had started with a purely architecture-driven approach and with the goal to demonstrate the benefits of a mobile computing infrastructure applied to real-world showcases. The paper describes why, during the course of the project, the architecture-driven approach sometimes proved to be unsuitable to tackle real-world constraints and requirements. Several reasons, e.g. frequently changing device road-maps, made architectural adaptations necessary. This resulted in a trade-off between desirable architectural design iterations and inevitable design circumventions, which finally led to an architecture and software components that proved to be flexible, robust and fit to meet the relevant real-world requirements.

1. INTRODUCTION

Our project started in 1999 with the idea of designing and building a platform in order to showcase the services for the needs of people on the move [1]. The objective was to set up an infrastructure which provides location and context aware services for mobile devices, efficiently combining novel navigation and communications technologies.

At that time, projects such as *Cooltown* [2], *EasyLiving* [3], *Centaurus* [4], *Ninja* [5] or *MOCA* [6] started with similar goals. Unlike some of these, Heywow began with "real world requirements" meaning we wanted to demonstrate the feasibility of an implementation with standard off-the-shelf components. The main focus was on the mobile user's device (which we call WID - Wireless Information Device). Of course, laptops or to some degree personal digital assistants (PDAs) with reduced form factor would have provided the greatest flexibility. Questioning ourselves whether it would be likely to see people walking around with large devices we decided not to rely on the annually shrinking form factors of these devices. Instead, we focussed on upcoming smart phones, an evolution of mobile phones towards more flexible and programmable devices. This coincided with the requirement to be able to use commercially available devices, which are easy to configure and useable by non-experts by the time of the showcases towards the end of the project.

Although our first ideas were born while investigating the Jini specifications [7], it turned out, that it is not just about bringing navigation capabilities into an existing service framework like Jini. Most service frameworks, including Jini, were designed with the characteristics of classical distributed computing environments in mind, i.e. they implicitly assume the existence of a reliable, wired network which interconnects all hosts involved in distributed service interactions and largely neglect the issues arising from mobility, adaptation, context-awareness, zero-configuration, local intelligence, and others [8]. Thus, one of our project goals was to reflect the requirements resulting from the evolution from distributed computing via mobile computing towards ubiquitous computing (see figure 1).

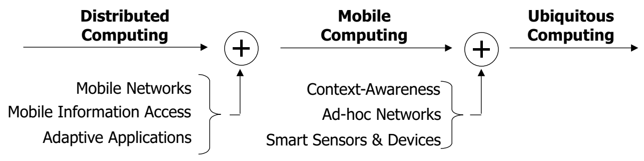

Figure 1: Ubiquitous Computing Evolution Chain

This paper is organized as follows: In the remaining part of this section we will give an introduction of our main design goals, followed by some considerations of the impact of external effects while "ramping up" an operational system. In section 2 we give some deeper insight into important system components and how we organized them in the architecture. The overall information distribution model is heavily influenced by the characteristics of mobile computing environments as we show in section 3. Section 4 presents our approach of displaying building or area maps on mobile devices with limited screen sizes, before we summarize our paper with a conclusion in section 5. Throughout all sections we emphasise the differences between the design and the lessons learned during the implementation.

1.1 Design Criteria

The penetration of smart mobile devices like mobile phones or PDAs is growing rapidly. They typically provide circuit and packet switched access to the Internet via public mobile networks. By using methods implemented today, mobile devices enable the user to access Web-based data from almost anywhere in the world while online by using a Web- or WAP-browser on their device. From a network operator centric viewpoint, this would have been sufficient for providing access to location-based services: GPRS provides Internet connectivity and the mobile networks facilitate subscriber localization. However, this would have caused tremendous operational costs which end users would not have been willing to accept. Even support through advertising revenues would not have lead to a viable business model for a successful commercial introduction. In contrast, when offline (e.g. in a plane, where the usage of wireless connections is currently prohibited), the user is unable to access online services and he or she is restricted to execute only a small set of offline applications stored and run on the device itself. Note that our definition of a service is not limited to information services. We define a service as a nameable entity being responsible for providing information or performing actions with specific characteristics [9], which requires for instance some kind of lifecycle-management on the device [10]. Thus, one of the challenges was to design an architecture which provides service discovery and execution mechanisms optimized for online *and* offline conditions in a best effort sense using on device processing and storage.

The upcoming smart phones added IrDA and Bluetooth for short range connectivity, providing no-cost data transfer and inherent location information. Wireless LAN was reserved to more powerful devices - both in terms of processing power and power supply. Also, infrared placed a too high challenge on making a full duplex data link work without modification to consumer devices (e.g. by adding lenses). Bluetooth promised to be the more flexible solution even if this meant developing our own access point infrastructure due to a mismatch of Bluetooth profiles between the mobiles and commercially available access points.

Realizing that smart phones usually have more than one interface for network access, it became obvious that we would like to use any of them where appropriate. Appropriateness is determined on several dimensions such as availability and coverage, transmission bandwidth, transmission costs, importance of the data to transmit and so on. In doing so, the heterogeneity of the different networks (Bluetooth, GSM-CSD, GPRS or even a docking cradle) turned out to be an advantage. That means, if for instance a high bandwidth at low transmission cost connection like Bluetooth is available, it is used for “data refueling”. If only a low bandwidth sz high transmission-cost connection like GSM-CSD is available, this connection is used for high priority requests only. And even the no-network situation should be covered in our system in terms of a combination of caching and smart prefetching algorithms as well as partial autonomy of the on-device services.

In our system location-awareness is seen as part of context-awareness. Following a sensing metaphor, positioning information is sensed as any other kind of context information using either a physical or a virtual sensor. In the case of positioning information, a physical positioning sensor may be an integrated or attached GPS receiver providing WGS84 coordinates or the mobile phone itself reporting Cell-IDs or Gauß-Krüger-coordinates. A virtual positioning sensor can be a software superposing and refining values of subsidiary sensors into adequate context information similar to the soft-location concept proposed in [11]. Context-awareness should be an inherent part of the service platform. This requires a well defined but yet extensible common model of what context is and how it can be used. The base model is designed to cover any dimension of the situation space (we call them *aspects*) such as the current position, the current network situation, specific device characteristics, user preferences, the history etc. The platform should provide access methods to any kind of context information through an API as well as necessary further infrastructure elements such as a monitoring and event generation component, enabling context-awareness during service discovery and service execution.

To achieve good end user acceptance during all showcases, we decided early on to provide not only some selected services to demonstrate the platform’s features. Whenever possible we approached content providers to get real world content for resource limited devices. To obtain high quality content, we opted to contract third parties whenever this kind of content could not be produced by any of the participating project partners. In this sense non system experts were enabled to provide content for our system.

1.2 Ramping up a Real System

While looking for adequate scenarios and locations to ramp up a demonstrator which is optimal in the sense of maximizing the demonstration effect by using already existing infrastructure and content, our initial intention was to approach an airport or individual airline operators, the administration of a capital city or even a trade fair - anything that is available in the vicinity of Munich close to our research lab. Obviously, a demonstration in collaboration with one or more of the parties mentioned above would have been beneficial in terms of visibility, tough and thus authentic requirements on the performance and stability of the products to show, and a reasonable number of users.

But due to their attractiveness, the parties and locations mentioned above are “saturated” by the amount of cooperation-requests to demonstrate something on the one hand, and allocate a huge amount of time for administration and negotiation on the other hand.

Thus, we decided to search for an attractive location with similar characteristics as mentioned above, with less adminstrative drawbacks. We found this location in a small Bavarian town named Landsberg am Lech (population approx. 27.000 inhabitants, city area 58 km^2). This town has numerous historic sites which are concentrated mainly in the old town area, which covers about 0.35 km^2. Furthermore, it has a well settled retail and tourism industry, which is open minded towards new ideas to make their city even more attractive for both tourists and local citizens. Importantly, the city administration supported our concern in various ways, either in terms of manpower, by establishing contacts or by providing access to valuable content like maps from the land registry office or access to the databases of the tourism office. Moreover, one should not underestimate the advantage of having the city administration as a committed partner when entering public discussions regarding electro-magnetic emission concerns due to modern communication techniques.

Another very important issue of our real world requirement was the device roadmap of the eligible vendors. Not every device that is announced will be available to the market or even for the developer community, more than ever not in time. The set of features of devices on the market is severely reduced compared to the ones of the developer version. The documentation is – if available at all – incomplete, deceptive and simply wrong, making the development process everything else but effective. Promising technologies such as access to the Bluetooth stack out of Java in the mobile phone as specified in JSR-82 [12] are postponed again and again, and once available it turns out to be useless due to some major design drawbacks.

2. ARCHITECTURE AND SYSTEM COMPONENTS

2.1 Devices

The devices deployed in the system may be categorized into three groups: end user devices, service points and the backbone infrastructure. Following our real world requirement, most hardware components are common off-the-shelf components. To cover the heterogeneity of the devices in use, an early design decision was made to use Java as a common programming language for all of the devices, hoping to increase the re-usability of any written code within in each of the three groups and optionally between the groups. In the remaining part of this section we will introduce the main components of the architecture on a more detailed level and discuss some of the lessons learned during the process from designing the system until the ready-to-use system in Landsberg had been set up.

2.1.1 End User Devices

We employed commercially available mobile end user devices (Wireless Information Device, WID) such as smart phones and PDAs from different vendors. Smart phones are standard mobile phones which additionally provide limited processing and storage capabilities for third party software. Most phones provide different levels of accessibility to the underlying software stack, ranging from full access to the operating system, such as Symbian OS, up to some scripting of GUI widgets only. Industry has approached this problem by creating standards for the application environment, e.g. by implementing functionality profiles on similar devices.

A typical representative of such a profile is the *Mobile Information Device Profile (MIDP)* based on the *Connected Limited Device Configuration (CLDC)* of the respective Java edition for smart phones. In theory, any phone implementing a profile such as CLDC/ MIDP should provide the same functionality to a third party software, which makes this software device and vendor independent. In practice, the heterogeneity has been shifted to a software level due to the disadvantageous ratio of mandatory and optional elements in the profile specification.

(Pure) PDAs differ from smart phones in terms of the user interface and connectivity, but are affected by the same problems w.r.t. third party software. Some provide direct access to the operating system such as Palm OS or Pocket PC, whereas others limit access to the API specified by a profile such as the *Personal Profile (PP, the successor of pJava)* on top of the *Connected Device Configuration (CDC)* of the respective Java edition. An upcoming trend is the convergence of smart phones with PDAs.

Following our design decisions, we implemented a life-cycle-management application as well as selected services to be run on the devices in Java. At this point we were already affected by two different profiles and dozens of special cases to cover the mandatory/optional mismatches. This resulted in a stack of interacting software modules covering the Java and networking capabilities of the different end user devices which is exemplarily outlined for the CityWalk tourguide application later in figure 5.

2.1.2 Service Points

Service points are network access devices which additionally provide area related services. We distinguish between *Local Service Points (LSP)* and *Global Service Points (GSP)*, depending whether they are equipped with short range communication facilities such as Bluetooth or with wide area communication facilities, particularly public access networks, to give end user devices access to the Heywow infrastructure. The LSPs are further distinguished by the type of connectivity towards the Heywow backbone:

- LSP Class A for stand-alone operation. Only initial or rare content updates, which may be uploaded using WIDs.
- LSP Class B with temporary and/or low-speed backbone connectivity such as GSM-CSD for scheduled content updates.
- LSP Class C with high speed backbone connectivity such as DSL or even LAN for spontaneous content updates. LSPs of Class C may act as full Internet access points (Hot-spot).

The development basis for our LSP was a robust small-sized, fully-integrated single board computer running Linux. To prevent failures to due mechanical wear, a fan-less version was chosen and the hard disk was replaced by a compact flash module of 256 MB or 512 MB. In future versions, a notebook power supply will replace the current one to allow quiet operation. Mostly standard components are used for the composition of the LSP (wireless LAN cards, common interfaces) to remain flexible to possible hardware changes. The most important requirement was to have access to nearly all parts of the software components inside. Because of this we did not choose any of the commercially available Bluetooth access points, as they often use special hardware or own protocol stacks with no or only sparse documentation. We decided very early to use the BlueZ Bluetooth protocol stack for Linux, which – although still under development – has achieved a very stable and comprehensive status during 2003. The BlueZ Bluetooth stack interconnects the Java Native Interface (JNI) of the Java Virtual Machine with the Host Controller Interface (HCI) of the Bluetooth hardware. We decided not to use the also available JBlueZ JNI classes because of a limited set of implemented HCI functions. This way we could define our own JNI classes according to our requirements. This protocol sits on top of the Bluetooth L2CAP layer. The LSP periodically scans for available mobile devices (*inquiry*) and checks for a custom Bluetooth device class. Once successfully identified, the LSP initiates a L2CAP connection to the WID and exchanges configuration data. The MP on the WID and the RP on the LSP are now connected via an error corrected L2CAP link - ready to transfer data. As the power consumption of a Bluetooth device in periodic *inquiry mode* is much higher than in *discoverable mode*, we decided to let the LSP look for available mobile devices and not vice versa. This mode significantly extends the standby time of the WID.

Measurements of the LSPs deployed in Landsberg showed some surprising results. We used power class 1 Bluetooth transceivers at the LSPs to provide access to WIDs which are typically equipped with power class 3 transceivers.[1] Although out of the WID's nominal range, we were able to establish and operate links between WID and LSP on a distance up to 40 m. Since we currently lack a more plausible explanation for this observation, we ascribe this increased range to a potentially higher receiver sensitivity of the power class 1 devices used in the experiment. Further measurements concerning

[1] power class 1: up to 100 mW (20dBm),
nominal range up to 100 m
power class 3: up to 1 mW (0dBm),
nominal range up to 10 m

the throughput were also remarkable. As seen from figure 2, GPRS transfer rates of up to 42 kbit/s were achieved (2 proxies involved), while transfer rates via Bluetooth of approximately 300 kbit/s were achieved (2 or 3 proxies involved).

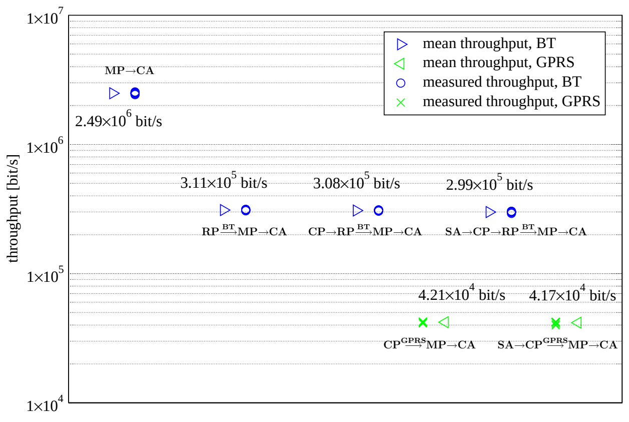

Figure 2: Measured Throughput via Multiple Proxies

However, as figure 3 shows, multiple connections at one LSP share this total bandwidth without major loss – therefore the total throughput per Bluetooth device remains the same (To increase throughput and the number of simultaneous supported users, additional USB Bluetooth devices can simply be plugged into the LSP). For a more comprehensive analysis of the potential impact of this observation see [13].

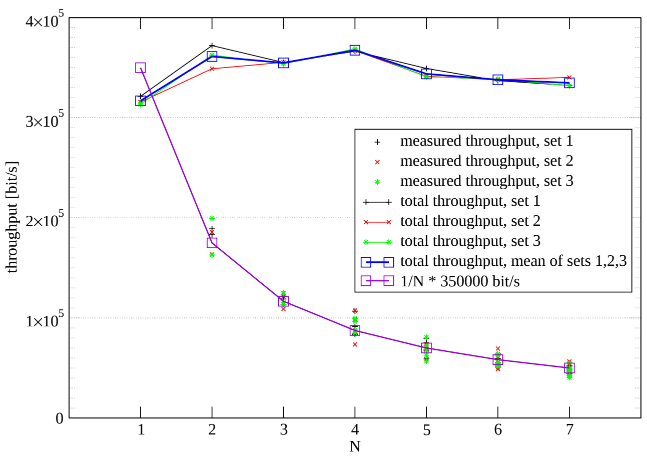

Figure 3: Bluetooth Throughput

2.1.3 *Backbone Infrastructure*

The backbone is essentially a virtual private network (VPN) spanning across public mobile networks, corporate networks and the Internet. WIDs access this network either using packet based public mobile networks, dialing into corporate networks (circuit switched) or using Bluetooth to communicate with a LSP which themselves access the network using fixed or wireless links.

Figure 4 shows the network as it is currently deployed for the Landsberg showcase. Within this figure, $\textcircled{1}$ shows WIDs communicating with LSPs of Class A, $\textcircled{2a}$ refers to fully connected LSPs of Class C in Landsberg with mirrors $\textcircled{2b}$ within our research lab for testing purpose. A W-LAN infrastructure covering the city centre was set up to serve both as backbone for LSPs indicated by $\textcircled{3}$ and to provide Internet access to laptops and PDAs with W-LAN option. Directional antennas on the LSPs allow for extended coverage.

Finally, few LSPs of Class B $\textcircled{4}$ with GPRS uplinks at remote locations are deployed. GPRS is also used directly by the WIDs $\textcircled{5}$ while outside of the Bluetooth/W-LAN coverage. In this case they communicate with a GSP $\textcircled{5'}$ located within our company's LAN.

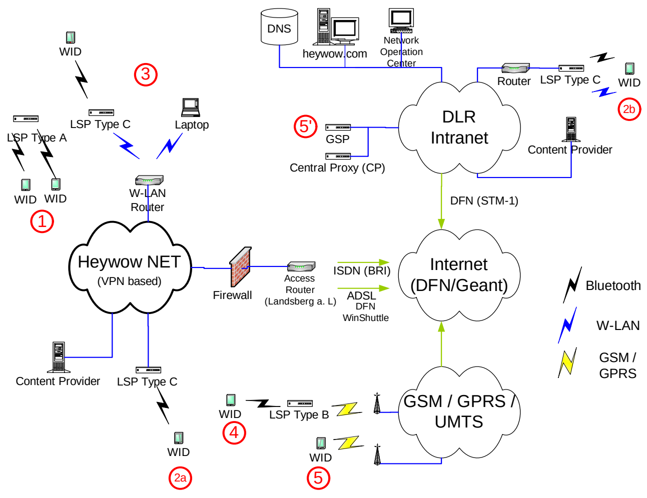

Figure 4: Heywow Network Topology

2.2 Software Components

This section will show our hybrid "medium client" approach employing standard components and dedicated applications, particularly on the WIDs. We rely on standard WAP/HTML browser and media player applications to render much of the application content and necessary user interaction – only if time or GUI constraints are severe, we rely on a dedicated application (such as a map viewer or autonomous tour guide application, see figure 5). By introducing the concept of application layer HTTP proxies we can still delegate most of the rendering and user interaction to the (usually built in) browser and players, but let the actual application logic and content reside as much on the device as necessary.

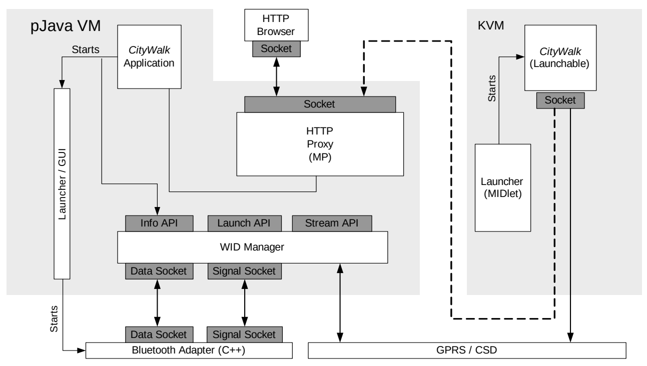

Figure 5: Interconnected Software Components

In order to provide a transparent and device independent solution, an application layer approach was chosen: Today a Web browser is the typical application for accessing content. Web browsers communicate using the HTTP [14] protocol to retrieve data from Web servers in the Internet. In order to control and modify the information flow, HTTP proxies are put in between the communication flow, which is illustrated in figure 6. In general, a client application (CA) retrieves data from server applications (SA). In order to support multiple communication links for cost effective switching [15] between WID and any LSP/GSP and for statistical analysis for optimizing data retrieval (e.g. pre-fetching [16]), a Central Proxy (CP) is placed within the infrastructure. Details of the so-called *Split Proxy Concept* can be found in [17].

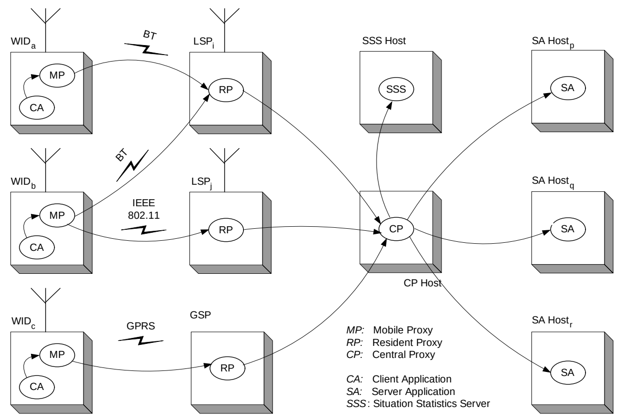

Figure 6: Communication Links

This chosen approach provides support for

- retrieving content located at the mobile device, the LSP or the Internet
- provisioning of the MIDP application to the mobile device (download of .jad/.jar files)
- communication of the MIDP and pJava application *CityWalk* with the back-end application
- communication link selection based on decisions derived from user preferences, data statistics and device feedback on a cost controller
- pre-fetching in order to optimize document retrieval time

3. INFORMATION DISTRIBUTION

As already mentioned in section 1, we followed our “good end user acceptance rate” design criteria by approaching and interconnecting to real world content providers whenever possible. Thus, the first dimension of the information retrieval design which results in our data flow model is determined by a selection of appropriate content providers. A second dimension is determined by the infrastructure, both set up by and within the administration domain of Heywow (see section 2) as well as given by the content providers and outside the administration domain of Heywow. A third dimension is determined by the intended use of the data, e.g. a specific service or application.

The overall data flow model is heavily influenced by the characteristics of ubiquitous computing environments (see figure 1). The most important ones associated with the design of the data flow model are:

- *Mobile Networks & Mobile Information Access:* In mobile networks spectral band is a limited and thus expensive resource. The amount of data exchanged should adapt to this fact in terms of transmitting only if necessary and according to the user’s (or the entity who pays for it) policies. This includes refraining from transmitting even during coverage unless a cheaper option is available [15]. This results in a store-and-forward like transmission pattern, whereas direct transmission patterns cannot be guaranteed for any point in time. Note that store-and-forward patterns are prone to data inconsistencies. The access to the data should be independent from the kind of access network as much as possible.
- *Adaptive & Context-Aware Applications:* Any application involved in the data flow may modify the data to adapt to user preferences, device capabilities, network situation or in general the context of the interaction. This may include omitting or transforming part of the data during transmission, as well as caching or hoarding and pre-fetching data based on usage and access statistics.
- *Smart Devices:* The processing and storage capabilities of the devices involved in the data flow can be used to shift computation as closed to the end user as possible which enables at least partial autonomy. This allows particularly for a medium client approach as described in section 2.2.

Figure 7 gives an overview for the data flow model from different content sources towards a mobile end-user device, which we will describe here as an example with the CityWalk tour guide application in the background. In this example, the CityWalk tour guide application is designed and implemented for CLDC/MIDP Java devices. Because of the *closed late binding* paradigm [9] of this Java derivative, it is not possible to add or change the class files after installing the package called *MIDlet Suite* onto a device. Thus, in contrast to other Java versions it is not possible to extend the functionality of the software at runtime by adding some new bytecode to an already existing package. Unlike the code, the data may be modified anytime.

The requirement for a maximum support of adaptation the need to maintain independence from the access network and the demand to do inconsistency checking resulted in a XML based approach. An XML schema[2] consists of CityWalk specific data structures as well as action (insert, update, delete etc.) and addressing elements. The schema specification has significant advantages for versioning and partial validation purposes, whereas the action elements may be used for database-like data merging operations. XML instance documents based on this XML schema can be used to be either displayed in a browser (e.g. as HTML representation as output of an XSLT processor) or incorporated in an application such as the pJava or CLDC/MIDP based CityWalk application. The markup information contained in the instance documents is essential for adaptation purposes.

[2]http://demo.heywow.com/schema/citywalk

A first rudimentary amount of data for an application is added to the Java archive during the creation of the MIDlet suite. This suite creation process itself is used to compose, personalize and device-optimize a MIDlet suite by merging a life-cycle management application, service code and profile-adapted data using a web portal[3]. Thus, the suite creation process takes care of user- and device-adaptation in the sense of the requirement described above. The life-cycle management application enables partial autonomy to any service within the suite, not only the CityWalk tour guide. The portal provides also mechanisms to download and install the suite onto a CLDC/MIDP device, either over-the-air (OTA) or Web-based. The classfile associated data allows the application to bootstrap its internal content database by itself after the installation.

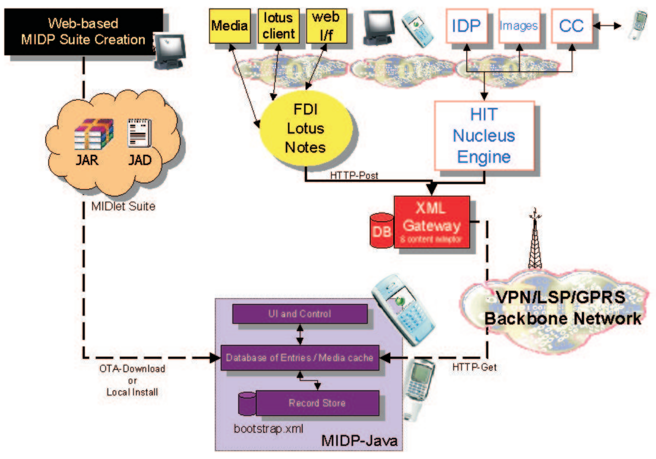

Figure 7: CityWalk Data Flow Model

At runtime, this on-device content database is updated at irregular intervals by requesting new data from any appropriate LSP or GSP. This is enabled by several mechanisms. First, the mobile device (or the user) determines the point of time of checking for new data, which optimal in the sense of some aspects of context-awareness (network coverage, costs, priority etc.). One may realize the similarities to checking for email using a pull based protocol like POP3. Second, the XML instance documents contain control and data information. By applying the control information, a traceable (but not necessarily conflict-free) way of merging from several sources is provided. Third, accessing XML documents and any other kind of resources via URLs provides independence from the underlying access network. This is used for instance to access area or group associated documents using the same access interface schema.

A component of a GSP important for the CityWalk application is a gateway which is responsible for merging data provided from preliminary content providers. One of them, called FDI, is responsible for the website of the city of Landsberg [4]. They provide a Lotus Notes based content management system, enabling the city administration staff to edit and publish the content of the city's web site in a collaborative and comfortable way. Given our XML CityWalk schema, they implemented an interface to create, manage and export data for the Heywow system. The city administration staff may create content for the CityWalk tour guide in exactly the same way they create data for their website, including descriptions, images, audio tracks etc. Another third party content provider we interconnected is a company called Humanize IT. They implemented a call center application allowing them to provide high quality location based personalized lifestyle information in cooperation with a lifestyle company. In committing to our schema they feed the Heywow system with restaurant and shopping tips as well as special event information. All the data is collected, checked for consistency and merged in the gateway shown in figure 7. The mobile device has access to this data via area, community or personal keys.

Even if this example uses the CLDC/MIDP Java version of the CityWalk tour guide as a reference, the data flow model of the pJava version of the CityWalk tour guide is the same.

4. MAPS IN MOBILE DEVICES

Location-awareness as a specialization of context-awareness in ubiquitous computing environments is often coupled with displaying building maps or area maps. But maps available in the Web and in use mainly on desktop monitors are insufficient for mobile device screens due to their limited size and resolution.

We approached this problem in two steps:

1. We contracted an image production company which is specialized in bitmap optimization to produce several versions of area maps of the old town of Landsberg. These maps varied in terms of file formats, scale and resolution, with and without anti-aliasing, color-coding, annotation of streets and places of interest, etc. The objective of generating this amount of variations has been the evaluation of their respective suitability for being used on the different smart phones in our testbed (e.g. Sony Ericsson P800 (pJava&MIDP), Nokia 3650 (MIDP)).

   From a questionnaire we considered the end user experience of using the maps for pedestrian orientation and navigation purposed. We found that a minimum scaling factor (not to be confused with the resolution) between 1:10.000 and 1:6.000 is required for these applications.

2. Maps covering a large area are usually of higher resolution, which may not fit the memory restrictions of mobile devices. For moving map applications, which indicates the position of a user/device with a symbol such as a little cross on the map, only a sector of the map around the positioning symbol is of main relevance. Thus, a large area map may be segmented and only a small amount of segments may be available on the mobile device at any time. To enhance the subjective feeling when the position marker moves towards the segment boundary, we applied an overlapping segmentation scheme which is illustrated in figure 8.

   By approximation the optimum for typical screen resolutions of smart phones (e.g. Motorola A920: 208x208, Nokia 3650: 176x144, Sony Ericsson P800: 208x203 etc.) we chose a segment size of 200x200 pixel which can be handled on phones even if they have a little lower resolution.

Whereas the first one may be treated as a study only because of its expenses, the second one may be used in combination with a strong pre-fetching and extrusion algorithm to compromise between the

[3]http://demo.heywow.com
[4]http://www.landsberg.de

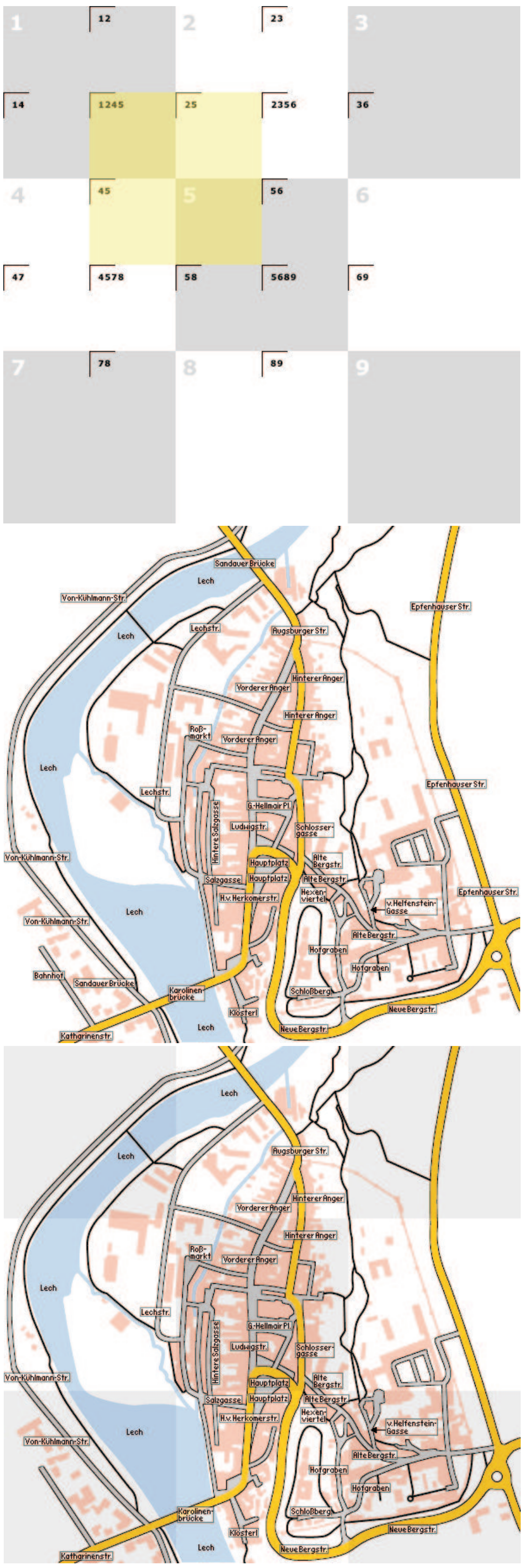

Figure 8: Fragmented map of the city area

necessity of permanent online access and the memory limitations of mobile phones.

Due to the fact that we had (already on the source code level) different implementations of the CityWalk tour guide accessing the same data, we are able to compare the application environments. One observation was that the specifications of pJava and CLDC/MIDP Java are awkward in some details. For instance the MIDP spec requires the PNG graphics format as mandatory only, whereas pJava supports the JPEG, GIF and BMP formats. Even if it is not explicitly forbidden to support other formats in either spec, this is the reason that there is no commonly supported graphics format in the implementations available on the smart devices we had access to. We approached this problem of format conversion as part of the device adaptation typical in mobile computing, implemented at the gateway described in the previous section.

5. CONCLUSION AND OUTLOOK

In the previous sections we described the experiences we made while ramping up a mobile computing environment within the Heywow project.

We introduced and gave reasons for our design goals, which where severely affected by typical characteristics of ubiquitous computing environments. During all phases of the development and deployment we were affected by minor and major obstacles, which required significant re-designs. Several promising design decisions such as using Java on all devices within the system to deal with the heterogeneity of the components have proved to be unsatisfactory to solve the problems arising from heterogeneity.

Nevertheless, the system is up and running and the desired functionality can be tested in Landsberg by the public. We think that by pointing to some of the major traps we have been struggling with, one can draw upon the lessons we learned to prevent them from happening to others who are designing and ramping up a similar system.

6. REFERENCES

[1] Angermann, M., Strang, T.: Heywow - a service platform for the needs of people on the move. In: Invited presentation on the 5th Jini Community Meeting, 10./11.12.2000, Amsterdam, The Netherlands (2000)

[2] Kindberg, T., Barton, J.: A web-based normadic computing system. Computer Networks **35** (2001) pp 443–456

[3] Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S.: Easyliving: Technologies for intelligent environments. In: Proceedings of the 2nd Int. Symposium on Handheld and Ubiquitous Computing (HUC), Bristol/UK, Springer (2000) pp 12–29

[4] Kagal, L., Korolev, V., Chen, H., Joshi, A., Finin, T.: (Project centaurus: A framework for indoor mobile services)

[5] Gribble, S.D., Welsh, M., von Behren, R., Brewer, E.A., Culler, D.E., Borisov, N., Czerwinski, S.E., Gummadi, R., Hill, J.R., Joseph, A.D., Katz, R.H., Mao, Z.M., Ross, S., Zhao, B.Y.: The ninja architecture for robust internet-scale systems and services. Computer Networks, Special Issue on Pervasive Computing **35** (2001) 473–497

[6] Beck, J., Gefflaut, A., Islam, N.: MOCA: A service framework for mobile computing devices. In: Proceedings of

the ACM International Workshop on Data Engineering for Wireless and Mobile Access, August 20, 1999, Seattle, WA, USA, ACM (1999) 62–68

[7] Arnold, K., O'Sullivan, B., Scheifler, R.W., Waldo, J., Wollrath, A.: The Jini Specification. Addison-Wesley (1999)

[8] Strang, T.: Service Interoperability in Ubiquitous Computing Environments. PhD thesis, Ludwig-Maximilians-University Munich (2003)

[9] Strang, T.: Towards Autonomous Context-Aware Services for Smart Mobile Devices. In Chen, M.S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A., eds.: LNCS 2574: Proceedings of the 4th International Conference on Mobile Data Management (MDM2003). Volume 2574 of Lecture Notes in Computer Science (LNCS)., Melbourne/Australia, Springer Verlag (2003) 279–293

[10] Strang, T., Meyer, M.: Agent-environment for small mobile devices. In: Proceedings of the 9th HP OpenView University Workshop (HPOVUA), HP (2002)

[11] Angermann, M., Kammann, J., Robertson, P., Steingass, A., Strang, T.: Software representation for heterogeneous location data sources within a probabilistic framework. In: Proceedings of International Symposium on Location Based Services for Cellular Users (Locellus 2001), Munich, Germany (2001) 107–118

[12] JSR-82, E.G.: Java APIs for bluetooth. (http://jcp.org/jsr/detail/082.jsp)

[13] Angermann, M.: Situation Awareness for Improved Mobile Information Access in Heterogeneous Wireless Networks. PhD thesis, University of Ulm (2004) (submitted).

[14] Fielding, R.: Hypertext transfer protocol – http/1.1. RFC 2616 (1999)

[15] Angermann, M., Kammann, J.: Cost metrics for decision problems in wireless ad hoc networking. In: Proceedings IEEE CAS 2002, Pasadena, USA (2002)

[16] Angermann, M.: Analysis of speculative prefetching. ACM Mobile Computing and Communications Review **6** (2002)

[17] Kammann, J., Blachnitzky, T.: Split-proxy concept for application layer handover in mobile communication systems. In: Proceedings IEEE MWCN 2002, Stockholm, Sweden (2002)