An Architecture Concept for Ubiquitous Computing Aware Wearable Computers

Martin Bauer, Bernd Brügge, Gudrun Klinker, Asa MacWilliams,
Thomas Reicher, Christian Sandor, Martin Wagner
(bauerma, bruegge, klinker, macwilli, reicher, sandor, wagnerm)@in.tum.de

Institut für Informatik
Technische Universität München
80290 Munich, Germany

Abstract

In Marc Weiser's vision of ubiquitous computing, users are located in an environment with potentially thousands of computers around them. Many capabilities of these smart devices can be used only by augmenting the users' senses with a kind of "sixth electronic sense". Thus, ubiquitous computing and wearable computing complement one another. However, the architectural styles for them are quite different.

This paper presents a new flexible and modular network-centered approach for the design of wearable computers. In our concept, a wearable computer is composed of a network of modules. A module can be worn by the user or be stationary in the user's environment. Each is a separate unit with its own processing, memory, I/O, power, and network connection, and provides specific functionality in the network. The modules reveal their abilities and needs to each other and dynamically assemble to form a network-based wearable multi-computer.

Our concept has been used in the DWARF *framework to build a first prototype system for indoor and outdoor navigation.*

1 Introduction

In Marc Weiser's vision of ubiquitous computing [12], users are located in an environment with potentially hundreds or even thousands of computers around them. These computers are embedded into everyday items and augment them with some form of intelligence. The user should be able to interact with them as freely as before without being bound to a controlling computer in his office. Many possibilities of these smart devices can be used only by augmenting the user's senses with a kind of "sixth electronic sense". This "sixth sense" is a mediator between the user's natural senses and the natural sense of the electronic world for receiving data. Thus, ubiquitous computing and wearable computing are complementary to each other.

Nevertheless, the architectural styles for them are quite different. While wearable computing uses the classical PC paradigm, ubiquitous computing uses distributed computing concepts. We propose a new flexible and modular network-centered approach for the design of wearable computers that allows a seamless integration of the wearable computer on the body into its ubiquitous computing environment.

In our concept, a wearable computer is composed of a network of modules. A module can be worn by the user or be stationary in the user's environment, ready to provide services. Each of them is a hardware unit with its own processing unit, memory, I/O, power, and network connection, and provides specific functionality in the network. An example for such functionality is user tracking for augmented reality. Modules reveal their abilities and needs to each other and dynamically assemble to form a network-based wearable multi-computer.

2 Related Work

Commercially available wearables [11, 13] or research prototypes are basically standard per-

sonal computers with a hardware design that allows them to be worn on the waist or in a vest. The core is a box with the CPU that controls the entire functionality. Other devices are peripherals that are connected via USB, FireWire, or some other interface to this box. Other designs with a flexible circuitry allow one to assemble several microcomputer cards [5] into a wearable computer, but the basic platform concept is the classical architecture of the personal computer. The software that makes the peripheral devices useful runs on the CPU of the wearable, not the peripheral device.

The *Spot Computer* from Carnegie Mellon University [4], the *MIThril* from MIT Media Lab [9], or *LART (Linux Advanced Radio Terminal)* developed at Delft University of Technology [8] have a modular concept, use the Intel StrongARM microprocessor, and are capable of running the Linux operating system. The modular concept and the powerful platforms are a good base for the development of wearable computers. Yet they don't have an approach for dynamic system composition with hardware modules.

The main difference between these projects and ours is that while these focus on a modular hardware design, we are attempting to build a system dynamically with techniques from distributed computing. Our wearables are composed of modules where each of them is a complete computer with CPU, memory, I/O, and network connection. This allows us to develop applications on a higher level of abstraction with component-based software engineering techniques. We use the hardware modules in the same way as software components in distributed component-based systems.

3 Objectives

We see one of the applications of our concepts in production and maintenance, where blue-collar workers need to use wearable computers with varying functionality. A worker receives his daily work assignments and for the various assigned tasks he needs specialized functions with him on his wearable computer. Just like packing his tool box with physical devices, he assembles his wearable with the modules he will need. He can pack his tool box not only with hammers and screwdrivers but also with specialized modules that can be connected to the wearable when needed. He also wants to be able to control objects in the environment, for example machines, via the wearable computer.

From this scenario we extracted the following requirements:

Tool metaphor. The user needs a set of tools that can be combined very simply by plugging hardware together. It is much easier to understand how to use things if you can touch them. Therefore, the complete functionality of each tool should be encapsulated with a respective hardware box that can be connected when needed. For example, for voice recognition, the microphone should be connected to a hardware module that runs a complete voice recognition software.

User comfort and energy reduction. One important issue is user comfort. In studies it was shown that it would be best to distribute parts of the wearable computer on different parts of the human body [2]. Another important issue is energy consumption. In [6], among others, the committee on electrical power for the dismounted soldier proposed a modular hardware design with dedicated processors for each subsystem.

Additionally it should be possible to tailor the hardware to the needed requirements. This allows compute-intensive services such as an optical tracker to be deployed on dedicated hardware which can be added or removed from the system at run time. Available functionality is not always needed in the same quality. For example, user tracking can be more or less precise. If a lower accuracy is acceptable for a specific task, a more accurate tracking device that is awkward to wear may be left in the tool box.

Dynamic integration. With ad hoc communication of modules within the network of the wearable multi-computer we can also extend its range. This means that services in the environment that come within reach can be integrated and used. An example are distributed user tracking concepts for augmented reality applications as described in [7]. When the wearable computer comes into reach of a tracking system in his environment, the user tracking software on the wearable and in the environment recognize each other and set up a temporary connection to exchange data about the user's position. When

the user leaves the reach of the external tracker, the connection is closed.

Resource sharing. In a team where each member uses a wearable computer, each could wear special modules with different sensors and related software on it. The results of these modules could be distributed wirelessly on the team and used by an application running on each wearable computer.

Off-the-shelf modules. Developers of software systems usually try to reuse software components. Analogously, for our proposed wearable computer, the goal is to have a growing set of modules available to be able to build new system configurations quickly and easily. This means that a customer specifies the application requirements, e.g. for a wearable factory management application plus location tracking, and the modules needed for this system are plugged together for a running system. In addition the system is filled with application-specific data, such as 3D models of the factory floor.

4 Concept

From the user's perspective there are two types of modules: application modules (Section 4.1) and component modules. The former are modules with user applications, the latter components that contain functionality for other modules; technically they are equivalent. The relationships and dependencies between the components and the hooks for the application are described by an architecture. Frameworks are a combination of components and architecture for a family of applications (Section 4.2); we give an overview of a concrete framework in Section 5 where we describe the DWARF prototype.

A module contains several *Services* and a *Service Manager* (Section 4.3). Each service is described by a set of *Needs* and *Abilities*. When a service needs to use other services the service manager of the module tries to match the needs with the abilities of other services. When all needs are matched, the service can offer its own abilities. Services, Needs, and Abilities are described in Section 4.4. The connection between a need and an ability is called *Dynamic Connection*. The set of dynamically connected services is called *Dynamic Configuration* of the system (Section 4.5).

For the cooperation of the modules we propose a two-step approach. In the first step the network of cooperating modules is set up and the means of communication are negotiated. After this setup, each module knows its communication partners. In the second step the data flow is directly between the components without the overhead of a third party. The existence of the connection is only temporary while the modules need to communicate. Afterwards it may be cancelled on request of the modules.

4.1 Applications

An application module is a module that contains functionality for the user. Usually it has only needs but no abilities for other modules. It also runs on a hardware component and the user can attach it to the system when needed. At startup, the application module checks if all needs can be matched. If so, the application starts offering services to the user. Usually there will be more than one application module on the wearable. Instead of installing application software on one module the user clips a new application onto his belt when needed.

4.2 Framework

In a system that consists of a set of components and their services, a framework describes the interfaces and dependencies between the components and the architecture of the system. It allows families of applications to be developed that reuse the components of the framework. Once there is a set of components available, new components and applications can be developed that use the existing infrastructure. Services providing the same type of ability, but using different techniques, can be exchanged for testing purposes as well as in final systems.

The interfaces between the modules must be specified in advance not only syntactically for the data types but also for the quality of the data. One example for such a standard interface that occurs quite often in wearable computing is the position of an object in six-dimensional coordinates, including parameters for quality of tracking accuracy.

4.3 Modules

A module, from the user's point of view, is the smallest possible component of the wearable computer. Every module has at least one specific function, such as a camera module for capturing video data, a GPS receiver module for positioning, or a head-mounted display module for displaying data. There can also be modules the user does not directly interact with—these only provide internal services such as a map of the world to be shown on the display module.

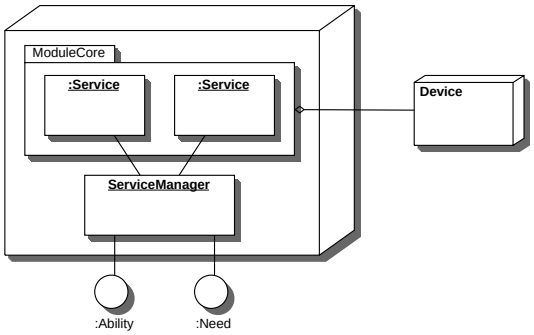

Figure 1. Software view of a module

Every module is self-contained as a physical device consisting of the hardware and all necessary software for the specific functionality as well as for the communication with other modules. Each module must have a service manager. The user simply has to switch on the device and, if necessary, perform some minimal configuration; the integration with the other parts of the system is established by the service manager automatically after the module starts up. Figure 1 shows the structure of a module. The module core provides the functionality of the module and the service manager is responsible to establish the collaboration with other modules. Any needed special-purpose hardware is directly connected to the module, for example a video camera.

4.4 Services, Needs and Abilities

A service is a piece of software running on a module that provides a certain functionality. To let the services interact, we associate them with several *needs* and several *abilities*. The abilities can be used by other modules if and only if all needs of the service can be fulfilled by services of other modules. Note that a single module can offer more than one service.

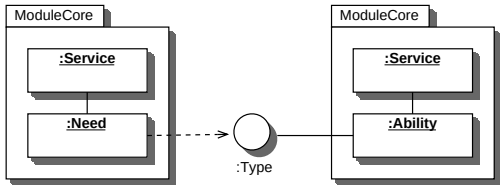

Figure 2. Connection between Needs and Abilities

Each need and ability has a specific type, and abilities can be fulfilled only by needs of the same type. Also, a need can have a multiplicity indicating how many abilities of the same type are required to fulfill this need. Abilities do not have an explicit multiplicity; every ability can be used by several needs at once. Figure 2 illustrates the relationship between services, needs, and abilities in a UML Object Diagram.

Quality Parameters. For a good match between the needs and abilities, it is not enough only to match the types. Information about the quality of the offered abilities and about the required quality of needs is important. An example is tracking accuracy.

Therefore, each ability has a set of attributes describing quality-of-service parameters of that service. Likewise, each need specifies a predicate about the quality of service it expects. This predicate is used by the service manager to select abilities that can provide a sufficient quality of service to satisfy a given need. This predicate can also be used at runtime to ensure that the desired quality of service is still provided. For the prototype we used a simple predicate based on attribute-value pairs.

Service Description. To let the service managers know the quality-of-service parameters, each module needs a set of appropriate service descriptions. These are then used by the service manager to match the needs and abilities. Every service has to describe itself at startup time to the module's service manager.

Another possibility is to store descriptions of installed services with the service manager, which can then start the services on demand when their abilities are requested, conserving resources such as battery power. This may be done using XML.

4.5 Dynamic Configuration

Each module has a service manager to enable dynamic configuration. The different local service managers communicate over the network, using protocols such as SLP [10], and distribute the information about the services with their needs and abilities. When eventually all needs of a service can be fulfilled by abilities of other services, the service managers create *Dynamic Connections* between the corresponding needs and abilities. These allocate and configure the necessary communication resources like event channels or shared memory blocks on both ends of the connection. The services can access the communication resources through the connections and use them to communicate with one another. The advantage of this is that two services can decide on the best way to communicate before they actually start communicating. There is no communication overhead at application runtime. To connect the services of two different modules no additional user interaction besides plugging the hardware together is necessary. Once the connection is set up, the service manager is not involved in the actual communication, although it can break connections or dynamically reconnect them to other services.

5 Prototype

Our concepts of a modular wearable computer were first tested within the DWARF (Distributed Wearable Augmented Reality Framework) project [1, 3]. DWARF is an ongoing research project with the goal of providing general software services for augmented reality applications which can run on a wearable multi-computer. Figure 3 gives an overview of the services, divided by functional categories.

For the first prototype, we used DWARF to build a wearable indoor and outdoor navigation system. The user's position, a two-dimensional or three-dimensional map and the route to his destination were shown in a see-through head-mounted display.

DWARF includes several different *Trackers*, which can establish the position of things, a *World Model*, which can store position information and other attributes of real and virtual things (among others, we used optical, GPS and magnetic compass trackers), a *Taskflow Engine* (executes the navigation instructions), several different kinds of *User Interface Devices* such as a head-mounted display and voice input, a *User Interface Engine*, and a *Context-Aware Packet Routing Service*.

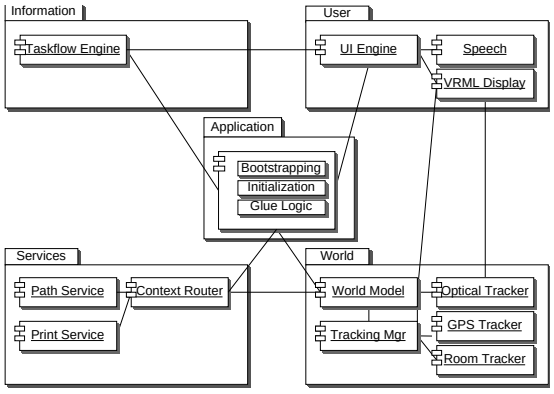

Figure 3. Subsystem decomposition

The *Application* uses the framework to build an augmented reality system. For our demonstration system, the application mainly provided bootstrapping functionality to let the user select navigation instructions to download, initiate the wireless transfer, and pass the downloaded data to the DWARF services.

The deployment hardware for a wearable computer ideally should be tiny, lightweight, and fast at the same time. Nevertheless, for our first prototype we decided to use two standard PC laptops mounted on a fixed frame backpack as shown in Figure 4. In our opinion, this restriction is tolerable, since the focus of our project was on software rather than hardware, and several quite powerful wearable hardware platforms that could be used for our purpose are already under development.

We developed our first prototype on a variety of platforms, including Linux for Intel, Linux for PowerPC, Windows 2000, and Macintosh, and finally deployed it on two Windows 98 and NT laptops respectively, connected with standard ethernet cables and wireless ethernet for external services. Additional peripheral devices included a FireWire camera, a GPS receiver and a head-mounted display. All devices are battery-powered, and the time of operation is more than two hours.

Figure 4. DWARF conceptual prototype

6 Conclusion and Future Work

The experiences gained by the implementation of DWARF are very encouraging. First of all, the implementation of the scenario took only three weeks' time, which is rapid prototyping at its best. Furthermore, although not all components were, the middleware was robust and usable, proving the value of the concept of a self-assembling modular wearable computer.

The current implementation of the middleware and of the DWARF system needs to be tested, extended and optimized, as well as ported to smaller systems such as Linux on StrongARM processors. Currently we are going to deploy DWARF on a set of Compaq iPaqs.

Important development areas are security and error handling, which present fundamental problems in ad hoc systems; roaming and handover in wireless networks; and more detailed quality-of-service parameters. Also, we would like to add a *Contract Manager* to the service manager on each module which would enhance the flexibility of the service connections at runtime.

Acknowledgements

The authors would like to thank Christoph Vilsmeier, Florian Michahelles, Stefan Riß, and Bernhard Zaun for their work with us on the concepts, design and implementation of the first version of DWARF. Additionally, we would like to thank Siemens AG for their support in the related *AiRGuide* project, which provided the inspiration for our campus navigation scenario.

References

[1] DWARF *Project Homepage*. Technische Universität München, `http://www.augmentedreality.de`.

[2] W. BARFIELD, S. MANN, K. BAIRD, F. GEMPERLE, C. KASABACH, J. STIVORIC, M. BAUER, R. MARTIN, and G. CHO, *Computational Clothing and Accessories*, in Fundamentals of Wearable Computers and Augmented Reality, Lawrence Erlbaum Associates, 2001, pp. 471–509.

[3] M. BAUER, B. BRUEGGE, G. KLINKER, A. MACWILLIAMS, T. REICHER, S. RISS, C. SANDOR, and M. WAGNER, *Design of a Component-Based Augmented Reality Framework*, in Proceedings of ISAR 2001, IEEE Computer Society, 2001, pp. 124–133.

[4] CARNEGIE MELLON WEARABLE GROUP, *The Wearable Group: Spot*. Carnegie Mellon University, `http://www.wearablegroup.org/hardware/spot/index.html`.

[5] D. W. CARROLL, KEY IDEA DEVELOPMENT, *Wearable Personal Computer System*, Patent No. US5555490, 1996.

[6] COMMITEE ON ELECTRICAL POWER FOR THE DISMOUNTED SOLDIER, ed., *Energy-efficient Technologies for the Dismounted Soldier*, National Academy Press, 1997.

[7] G. KLINKER, T. REICHER, and B. BRÜGGE, *Distributed User Tracking Concepts for Augmented Reality Applications*, in Proceedings of ISAR 2000, Munich, Oct. 2000, pp. 37–44.

[8] LART, *Project Home Page*. TU Delft, `http://www.lart.tudelft.nl`, 2001.

[9] MITHRIL, *Project Home Page*. Massachussets Institue of Technology, `http://www.media.mit.edu/wearables/mithril/`.

[10] *Service Location Protocol*. `http://www.svrloc.org`, Feb. 2001.

[11] VIA INC., *Home Page*. `http://www.via-pc.com`.

[12] M. WEISER, *The computer of the twenty-first century*, Scientific American, (1991), pp. 94–100.

[13] XYBERNAUT CORPORATION, *Home Page*. `http://www.xybernaut.com`.