Middleware for Pervasive Healthcare
A White Paper

Jakob E. Bardram

Center for Pervasive Computing

University of Aarhus

8200 Århus N

Denmark

bardram@daimi.au.dk

Henrik Bærbak Christensen

Center for Pervasive Computing

University of Aarhus

8200 Århus N

Denmark

hbc@daimi.au.dk

ABSTRACT

This white paper describes work-in-progress at the Center for Pervasive Computing (CfPC) at University of Aarhus. We describe our *pervasive healthcare* project, which is a collaboration between hospitals in the county of Aarhus, a Danish software company developing an electronic patient record solution, and CfPC. The aim of the paper is to present and discuss a vision of a middleware architecture for pervasive computing within clinical work settings. Our research is grounded in careful observations of clinical work within a hospital and we start by outlining recurring scenarios in the daily life of healthcare staff and sketch how pervasive middleware technologies may provide a strong foundation for pervasive and mobile solutions in this setting.

INTRODUCTION

Distributed computing has for a long time been dominated by the client-server paradigm. In this paradigm shared data and services are available on the server, and more or less complex clients draw upon the services of the server. The client-server paradigm is a strong paradigm and has proved its strength over time. Until recently, however, computational devices have been heavy, bulky, and thus immobile, and this fact has influenced our perception of what constitutes the *client*. Traditionally, the client is a single application running on a single computer with integrated input- and output devices.

The growth of small portable devices like mobile phones and PDAs makes it apparent that we must rethink client-side computing. Small devices have their primary force in their mobility but have a number of inherent weaknesses: they have limited screen- and interaction abilities, they are easily lost or stolen, and their access to the network varies greatly and rapidly. These constraints pose new and interesting challenges to middleware technologies: middleware must support application programmers in building clients that overcome these inherent weaknesses.

The Center for Pervasive Computing at University of Aarhus is a multi-disciplinary cooperation between a wide range of research areas, like Programming Languages and Environments, Virtual Machine Design, Software Architecture, Colored Petri Nets, Computer Supported Cooperative Work (CSCW), Human-Computer Interaction (HCI), Ethnography, and Industrial Design [6]. The Computer Science Institute has a long-standing tradition in experimental systems development with user involvement as we believe in the power of end-user reality 'fighting back' upon our designs. One of the cornerstones in our experimental research is testing prototype systems in real environments.

One of the projects in the Center for Pervasive Computing is *Pervasive Healthcare* where our center is collaborating with healthcare staff at hospitals in the county of Aarhus as well as a software company that is developing a next-generation electronic patient information system. This context is an ideal test-bed for pervasive middleware technologies. First, healthcare at modern hospitals is characterized by mobility taken to the extreme—nurses and doctors are on the move almost constantly. Second, the seemingly trivial task of just finding the physical space for an ordinary PC is difficult. Third, interruptions are frequent during everyday tasks, like giving medicine to the patients. Thus the activities and tasks are well suited for being supported by mobile and pervasive solutions. Collaborating with the healthcare staff is vital for ensuring the soundness of proposals and underlying middleware technologies. And finally, the collaboration with the supplier of the electronic patient record system allows us to tap into the real healthcare data making a truly realistic setting for experimenting with prototypes – and to some extent influence the infrastructure solutions adopted by the supplier.

SCENARIOS

One of the central scenarios that our project is looking at is the task of *medication* of patients. For an outsider such a scenario might seem trivial – prescribing some medicine and ensuring that the patient gets the right dose at the right

time. However, taking a closer look[1] this scenario is highly complex. Prescription is a task that involves complex problem solving in a highly distributed and cooperative setting – often the physician needs to cooperate closely with specialists, like radiologists, and conference with other (more senior) physicians. The task of ensuring that the patient is given the correct dose of medicine at the right time of day and under the right conditions is also rather complex. Nurses have devised advanced artifacts in terms of checklists, medication schemas, etc. in order to document the medicine given to a patient. What further complicates these tasks are the circumstances under which they are performed. Nurses and physicians are constantly on the move and are collaborating *ad hoc* in an environment that provides a constant source of interruptions. Thus a physician needs to engage in conferences at almost any time and anywhere. A nurse may be interrupted several times during the medication (having to remember who has been 'served') and/or hand over the task to another nurse.

Together with the staff at one of the hospitals in Aarhus county we have come up with the following future scenarios for using mobile and pervasive computing devices in support for medication of patients. These scenarios are our guidelines in designing the architecture for pervasive computing presented in this paper.

Scenario 1 – Medicine prescription

A younger physician has indicated interest in the result of a specific patient's blood sample, because the patient's condition has become worse overnight. The lab result is ready at a moment where the physician is walking down a corridor, and a notification with the result is therefore sent to the PDA that he is carrying, as he is not near his PC. He takes a look at the result and decides that he has to consult a senior physician and the nurse in charge of the patient at the ward. He sends a notification to them and asks them to meet him in the conference room at the ward. When entering the conference room the PDA and the SmartBoard in the room discover each other and the younger physician can now use the SmartBoard to display a graph of a value from the blood sample indicating an increasing problem. The graph viewed at the SmartBoard is controlled from the physician's PDA. A new treatment is discussed and the nurse studies the medicine description in the medicine book. Here she can see that this type of medicine is ill suited for the patient because of undesired effects. She explains the problem while displaying excerpts from the medicine book on the SmartBoard. Finally, a new course of

treatment is decided upon and both the nurse and the physicians make individual notes on their PDAs.

Scenario 2 – Talking to the patient

The nurse in charge of the treatment of the patient goes to the patient and explains the situation (from scenario 1) to him. At the bedside she uses the patient's TV in the ceiling to display the graphs and other information from the patient record. The images shown on the TV are controlled from the nurse's PDA.

Scenario 3 –Medicine handout

At the morning round the nurse and physician use the portable laptop PC to view the progress of a patient's treatment. They decide to give him his daily intravenous (i.v.) medication treatment right away. The nurse "drags the patient" onto her PDA, goes to the medication room and starts to find the medicine. While she is approaching the medicine room the patient's medication schema appears automatically on the PDA. When entering the medicine room the PDA discover the barcode scanner located in the medicine cabinet containing the i.v. medicine. The nurse finds the medicine and uses the barcode scanner to register it. If there is a mismatch between the prescribed medication and the medicine scanned with the barcode scanner, a warning is shown on the PDA.

Finally, the nurse goes to the patient and gives him his medicine. The PDA is used to mark the medicine as given to the patient and the time and initials are recorded.

Scenario 4 – Medicine conference

A younger physician is on duty in the outpatient clinic located in another part of the hospital and is seeing a patient going to regular control. Normally, this patient would see another doctor but has sought advice because he is feeling particularly ill. The younger physician cannot solve the problem and decides to call the doctor normally treating this patient. He uses the desktop PC in the room and asks for a conference call involving audio, video and a shared image of the patient's record. The other doctor is notified on his PDA while seeing another patient. He accepts the conference call but ask the caller to hold for a moment. He finishes the conversation with the patient and walk out in the hallway and accepts the conference call. As the PDA does not have video capabilities, only the audio and an adapted image of the record are used.

However, the doctor using the PDA needs to talk to and see the patient. He therefore walks up to the PC in the team room and moves the conference to this PC The video link is established and is now used to talk to the patient while browsing the record together with the other physician.

FUNCTIONAL REQUIREMENTS

In order to support the scenarios described above we have identified a list of functional requirement for a pervasive software architecture.

---

[1] Within the Center for Pervasive Computing we work in an interdisciplinary way having an ethnographer on our team making detailed observations of hospital work and instructing computer scientists in doing the same. Much of our research originates within the research areas of Computer Supported Cooperative Work (CSCW) where the use of ethnography has become a rich source of technological innovation [8].

Mobile devices – the architecture should support mobile devices that are moving around, connecting to and disconnection from different networks. For example wireless networked laptops, PDAs, and other small computing devices.

Composite devices – the architecture should enable the user to utilize whatever computational devise available [4]. For example when the functionality of the PDA in scenario 4 is too limited, part of the conference is moved to the PC. In this sense the PC augment the PDA. Another less visible example is moving heavy computation away from a PDA and onto more powerful server computers.

Heterogeneous devices – the architecture should support different types of devices used for input and output. For example the PDA used to remote control an image on the TV set.

Discovery of resources – resources in our terms are divided into *places*, *things*, and *persons* [3]. The architecture should enable applications to lookup and discover available resources and the relationship among them, like knowing that the physician in scenario 1 has his PDA with him in the hallway.

Location and context awareness – the architecture should support that a mobile device knows its location and its context. This is for example used in scenario 3 where the nurse is provided with relevant information about the patient that she is about to fetch medicine for.

Error and fallout recovery – the architecture should be tolerant to errors and network fallout. It must provide the application programmer with possibilities to acknowledge and identify such conditions and pursuit other means of carrying on. This is a radically different philosophy compared to the prevailing that tries to make network and distribution issues transparent to the programmer.

Security and authentication – the architecture should clearly support security – especially within a healthcare environment. In addition to more traditional aspects of *security* (e.g. encryption) and *authentication* (e.g. login) a software architecture for pervasive computing needs to take into account issues like *privacy* (who decides what to be shown on my TV or PC? – and do I want my movements to be monitored) and *physical access* (who else is watching this public TV?).

ARCHITECTURE VISION
The functional requirements and the scenarios described above are clearly ambitions. In the remains of this paper we shall concentrate on presenting our ideas based on a small subset of the above scenarios and elaborate on our architecture from there. However, in the end of the paper we will argue that the basis building block of the architecture can be expanded to capture much of our ambitions.

Basic Concepts and Requirements
Our architecture is modeled over the idea of a *session* between a number of participating devices. A session is characterized by its properties in time: it is *initiated*, it is *ongoing*, and finally it is *terminated*. Two or more persons are involved in a session through a *dynamic set of devices* (devices can join and leave a session over their lifetime). The device a person initiates a session from is denoted the *initiating device*. A session is centered upon some *set of data* that is viewed and possibly edited by the persons through their devices. A session is said to be *reconfigured* in the event that the set of devices is changed.

Examples: In scenario 2 the session is concerned with the patient record's blood sample graph (dataset) using the nurse's PDA and the ward TV (devices). In scenario 4 the session is concerned with a patient record, audio, and video streaming using a changing set of devices (first PC? PDA, later reconfiguring to PC? PC).

The main requirement of our architecture is to support the session as best as possible under the constraints of available devices and network connections. This includes:

1. A session must be able to represent all types of datasets [functionality]

2. A session must gracefully cope with changes in the set of participating devices over its lifetime [adaptability]

3. A session must rely minimally on central server facilities [availability]

4. The inspection and editing of the datasets handled in a session are constrained by the authority level of the participating persons (i.e. *not* devices) [security]

5. The architecture should accommodate novel types of devices as they appear on the marketplace with minimal changes [modifiability]

6. The classes of data supported by sessions (types) should be open [modifiability]

The associated quality attribute [1] of the architecture is given in brackets after the requirement.

The underlying architectural pattern to handle a session is the Model-View-Controller (MVC) pattern [2]. The model part encapsulates the datasets managed by the session and may be either generated by a device (typically audio/video in a conference setting) or may be represented by some reference to data in the electronic patient record system. The view- and controller parts are (usually) deployed on the devices. The model, view, and controller parts engage in peer-to-peer communication over the network.

We envision that there will be one "triad" of MVC parts for each class of data that is supported in sessions.

Component Model
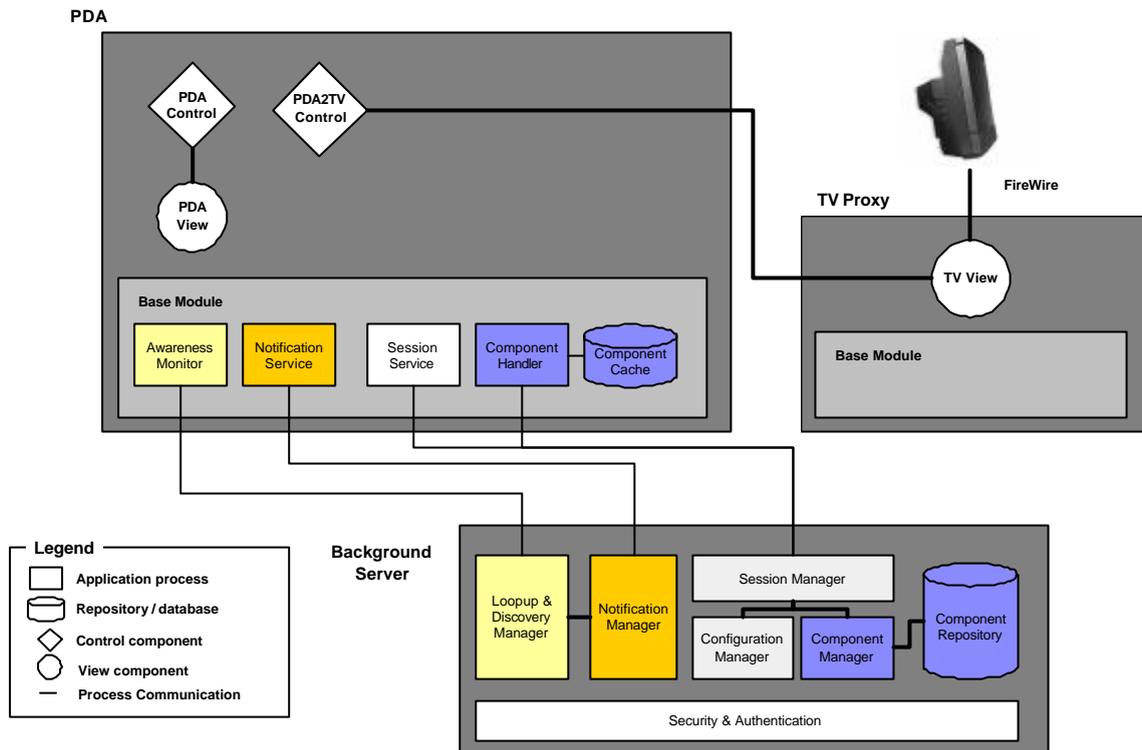
Figure 1 – Logical Component Model.

Figure 1 illustrates the component model of our architecture. The architecture consists of a range of *Background Server* components (deployed on multiple physical hosts if necessary) and a small *Base Module* deployed on each pervasive/mobile device. The base module is able to dynamically unload, load and run new software components at runtime on the device.

The main player in this game is the *Session Manager*, which is a centralized service in our architecture. The basic task of the session manager is to calculate a set of viewers and controllers that defines a consistent configuration for handling a given session through the given devices. For example in scenario 2, the session manager is requested by the nurse's PDA to establish a session on blood sample data using her PDA and the ward TV. The session manager must respond by providing the PDA with a suitable controller (that runs on a PDA but can control the ward TV) and by providing the TV with a suitable viewer. However, because the TV in itself cannot download and run communication software, the architecture uses a *TV Proxy*, that work as the viewer for the TV and forwards the view to the TV. The TV Proxy draw data from the electronic patient record system, and accept control events from the nurse's PDA. Notice that once the session manager has established the session (by providing the devices with their proper viewers and controllers) its role ends as the devices engage in peer-to-peer communication. Only when a session must

activated.

To be more detailed, devices use the *Session Service* to initiate a session involving themselves and other devices. The Session Manager cooperates closely with the *Component Manager*, which fetches the right components (viewers and controllers in consistent configurations) in the *Component Repository* for the devices requested to participate in the session. The *Configuration Manager* configures these components according to the current setup (place, thing, person). The *Component Handler* on the device is given a reference to a component to use and decides whether to download it or use a cached one from its internal *Component Cache*.

The *Lookup and Discovery Manager* supports the discovery of resources and the location and context awareness described above. By cooperating with the *Awareness Monitor* on all clients the manager traces the relationship between places (the 'where'), things (the 'what'), and persons (the 'who'). For example, where the nurse is located in relation to a patient, and what (computational) device she has nearby.

The *Notification Manager* keeps track of notifications submitted or due for distribution. This manager is used for establishing sessions asynchronously and cooperates closely with the Lookup and Discovery Manager in order to find out who is to be notified, where (s)he is, and on what

device. The *Notification Service* running on the client is used to submit notifications to others and to handle incoming notifications.

The *Security and Authentication* module prevents non-authorized access by using normal access-control lists. These lists are extended to also include a notion of access depending on location and who is using the device. The latter is intended to keeps track on the privacy to devices.

Dynamic Model

Figure 2 is a coarse-grained UML sequence diagram outlining the dynamics involved in scenario 2. The gray shading area indicates the objects that exist in the address space of the nurse's PDA and ward's TV Proxy respective-ly. The diagram starts by the nurse initiating the blood sample session from her PDA, pda1. The session is initiated by the PDA contacting the session manager requesting a session with the specified *model* (the blood sample data from the patient record), a *viewer* (the PDA screen), and a *controller* (the PDA input device configured to control the PDA viewer). The session manager responds by calculating an appropriate configuration of viewer- and controller objects and creates these in the PDA. This is shown by the creation of the PDAControl and PDAView objects. The viewer and controller objects attach themselves to the model (to observe model changes, as prescribed by the observer pattern) and then engage in standard MVC interaction, except that the communication with the patient record is over a wireless network.

Later the nurse enters the ward with a TV, tv2. She asks to update the session, now including the TV as output device but still wants to control it from her PDA (like scrolling, zooming, editing). A controller for a TV is much different from one that controls a PDA. Just think of the fact that you do not look at the PDA while controlling the TV—you look at the TV screen. Thus standard scroll-knobs are of little use. Therefore the session manager requests the current controller and viewer on the PDA to unload themselves. This means that they detach from the patient record model as they do not want to observe model changes where after the objects are destroyed. The session manager then updates the session by creating a PDA2TVController in the PDA, and an appropriate viewer in the TV Proxy; these newly created objects attach to the model and engage in MVC interaction.

DISCUSSION

We will shortly outline how the proposed architecture vision supports the architectural quality attributes described in the requirements section above.

Adaptability: The dynamic uploading of consistent view/controller pairs to participating devices ensures that sessions can adapt to a given configuration of devices. The idea resembles the proxy-upload for Jini [7] clients when they need to communicate with a service; our architecture however is symmetric in the respect that all devices have "proxies" uploaded.

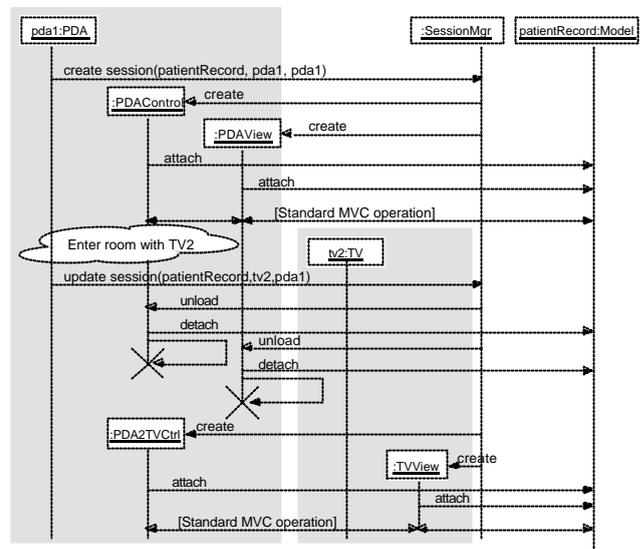*Availability:* The only time that the service of the session

Figure 2 – Sequence diagram for changing viewer

manager is needed is during session initialization and reconfigurations. Thus worst case in scenario 2 if the session manager becomes unavailable during the session is that the nurse cannot transfer the view to the TV. She is still able to view and manage the data on the PDA. Another kind of availability is that the session manager may dynamically reconfigure the viewer-controller configuration in case bandwidth varies. For instance if a PDA moves to a place with lower bandwidth network connection, a new audio transmitter could be uploaded at the source that works at a lower sampling rate.

Security: Security and Authentication is underlying the whole architecture when communicating and fetching resources from the server. Furthermore, a session cannot be instantiated without proper access rights, privacy rights, and authority. This also means that the controller, model and viewer components should be authorized to work together. In this way a "false viewer" cannot listen in on a controller-model interaction. However, security and authentication between cooperating component peers are not part of our architecture – it is up to the peers to ensure secure communication and some sort of authentication themselves if necessary.

Modifiability: We think that by employing the MVC pattern as cornerstone in the architecture, the task of writing viewers and controllers to new devices is a manageable task. At this early stage, however, our viewpoint is not backed up by careful analysis.

As outlined in the introduction our architecture vision is still preliminary. At its present state a number of questions are still open.

1. Reconfiguring sessions over their lifetimes is a central aspect of the architecture. It is however not clear how to support maintaining **state** of viewers and controllers over a reconfiguration. For instance if a physician has laboriously zoomed and scrolled an X-ray image to show the interesting part, he does certainly not want to repeat these actions when moving the image from one device to another. However, the capabilities of the new viewer may be radically different. So, even though the viewer 'signing off' may be able so save some state information, the new viewer may not be able to interpret the state information directly. Writing adapters for each combination of (old, new) viewer pair does certainly not scale up.

2. Another aspect is that of **multiplexing**. For instance, in a conference setting the audio input of one device must be multi-cast to a number of receiving devices. However, if the source is a PDA with a slow network we cannot rely on the PDA to do the multiplexing. One idea is to create a process on some server that accepts the audio input and does the multiplexing.

3. Similarly, **adapting data could** take place in a separate server process. For instance a high-resolution color image may beneficially be converted to a lower resolution grayscale image before sending over a slow connection to a PDA. A server process may then intervene the dataflow and perform the adapting computations.

SUMMARY

We have outlined some real-world scenarios within hospital work, which call out for pervasive and mobile computer support. Using this healthcare setting as our base we have described a pervasive middleware architecture vision and discussed how this architecture could accommodate architectural quality attributes that we find import in pervasive healthcare. We hope to be able to present our initial work at the Workshop on Middleware for Mobile Computing to get feedback on our ideas and design, and draw upon the experience of the scientific community at an early stage of our project.

REFERENCES

1. L. Bass, P. Clemens, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley 1998.

2. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal. *Pattern-Oriented Software Architeture*,. John Wiley and Sons, 1996.

3. D. Caswell and P. Debaty. Creating Web Representations for Places. In *Proceedings of Handheld and Ubiquitous Computing* [5], pages 114–126.

4. T. Pham, G. Schneider and S. Goose. Exploiting Location-Based Composite Devices to Support and Facilitate Situated Ubiquitous Computing. In *Proceedings of Handheld and Ubiquitous Computing* [5], pages 143–156..

5. P. Thomas and H. W. Gellersen, editors. *Proceedings of Handheld and Ubiquitous Computing, 2000.* Lecture Notes in Computer Science 1927. Springer Verlag 2000.

6. See www.pervasive.dk

7. See http://www.sun.com/jini/ and http://jini.org/

8. Bardram, Jakob E. 1998. Designing for the Dynamics of Cooperative Work Activities. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, Seattle, Washington, USA. ACM Press.