

Context-aware adaptation for component-based pervasive computing systems

Davy Preuveneers, Yves Vandewoude, Peter Rigole,
Dhouha Ayed, and Yolande Berbers

Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A, B-3001 Leuven, Belgium,
{firstname.lastname}@cs.kuleuven.be,
<http://www.cs.kuleuven.be>

Abstract. Developing and deploying context-aware mobile and pervasive applications that are adaptable to a broad range of high-end and low-end systems is a daunting task. The contribution of our research within the pervasive computing domain is a context-awareness infrastructure developed in the framework of the CoDAMoS project¹. The infrastructure provides runtime support for context-driven adaptation of component-based mobile services and supports context-driven resource discovery, service adaptation and service mobility at runtime. The research demonstration will illustrate peer-to-peer interoperability and context-driven service adaptation at runtime on both high-end computing systems and resource constrained mobile handheld devices.

1 Introduction

With the continuous growth of wireless communication, mobile hand-held devices, such as PDAs and mobile phones, are becoming a powerful platform that allows users to create a whole range of entertainment and business applications that are more intelligent and supportive to the user compared to most nowadays applications. A pervasive computing environment [1] supposes that (1) applications are deployed on devices ranging from desktop computers to handheld devices, (2) that they are aware of contextual information on the user, the device itself and its environment, and (3) that they can be adapted to this dynamic context accordingly. For example, in such an environment applications may have to be adapted to the varying device capabilities, such as memory or screen size. Furthermore, applications may have to be moved while active from one device to another, in order to provide the mobile user with the best user experience.

The contribution of our research is a context-awareness infrastructure that fulfills these three needs by providing runtime support for context-driven adaptation of component-based mobile services. In this paper, we give a high-level overview of our Java component-based infrastructure that runs unmodified on both high-end and low-end devices and illustrate its support for context-aware application adaptation by means of a pervasive computing showcase.

¹ <http://www.cs.kuleuven.be/~distrinet/projects/CoDAMoS/>

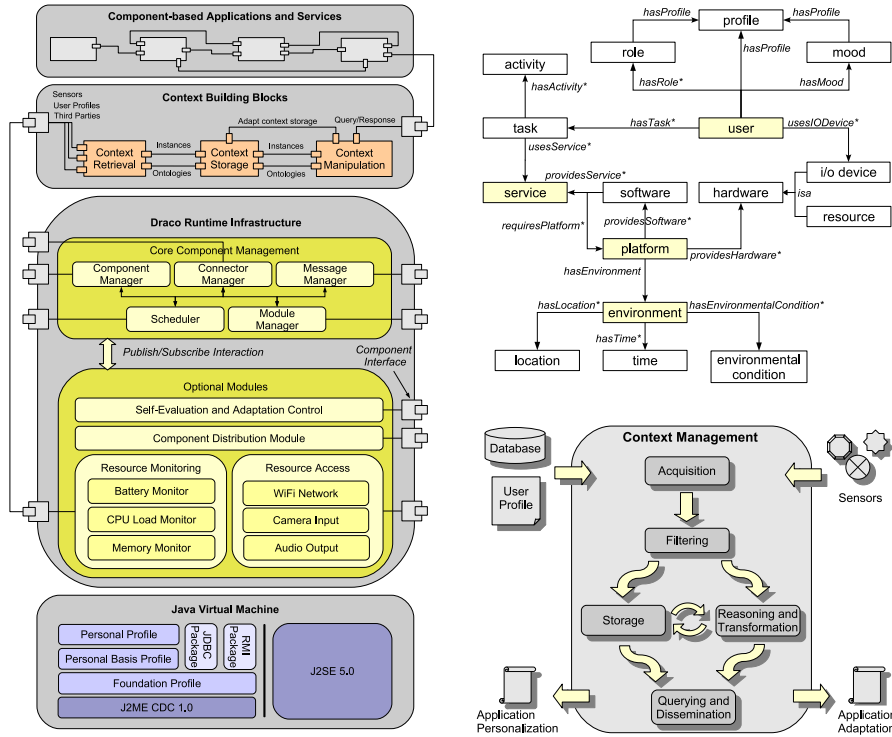


Fig. 1. The context-aware infrastructure Fig. 2. Context ontology and processing

2 Context-driven adaptation of component-based services

We use *components* as modular building blocks for the design and deployment of adaptable services. Component-based applications and services [2] are more flexible than monolithic applications: the component paradigm makes it possible to add, remove or replace a component at runtime when the context changes.

An overview of our context-aware runtime infrastructure is given in Figure 1. Draco [3] is our lightweight extensible Java-based component middleware (shown in yellow) with support for live updating using better suited components and distributing component-based applications. This component middleware and all components run unmodified on top of the J2ME Personal Profile 1.0 virtual machine for handhelds and on the Java 2 Standard Edition 5.0 for desktop systems (shown in purple).

The context-awareness layer (shown in orange) is responsible for acquisition, storing, aggregating, reasoning on and disseminating the context information. The context is modeled using our context ontology [4] (which defines the relationships that hold among the basic concepts: *User*, *Platform*, *Service* and *Environment*) and other domain specific ontologies. The context processing chain and a subset of the context ontology is shown in Figure 2. By making use of the Jena 2 semantic web framework the context layer is able to reason on the current

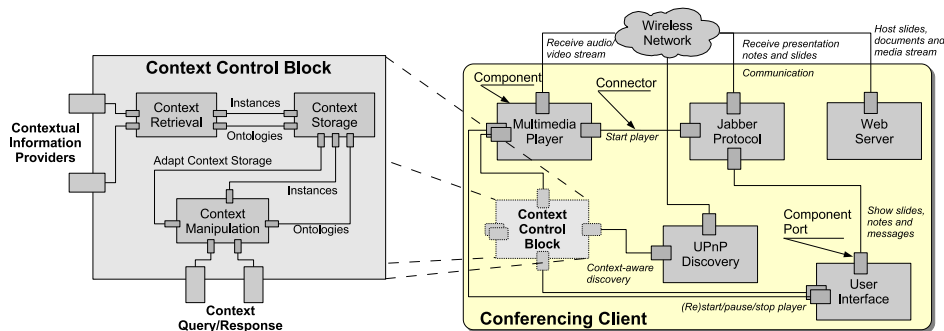


Fig. 3. The component-based conferencing client and the context components

context and adapt the component-based applications according to user preferences, service requirements or resource availability. Moreover, the context layer is also component-based [5]. This means that the context layer, while monitoring the resource usage, can decide to (un)load one of its own context managing components or to distribute it to another device when it is too computationally intensive for the device on which it is currently deployed.

3 Illustrating a pervasive computing showcase

The following scenario shows the possibilities of our context-awareness infrastructure, which will be demonstrated: *Mr. Smith is a sales representative and is attending a meeting where the results of a survey administered by his company are being presented. All attendants can use a display to interact with the shared whiteboard in the conference room using a conferencing application. Mr. Smith has to leave the meeting early. Upon detection that Mr. Smith is leaving, this conferencing application moves from his display in the conference room to his personal wireless handheld device with reduced functionalities because of the reduced resources. Later on, whenever appropriate devices show up in the vicinity of Mr. Smith, some parts of the conferencing application can move to other devices in order to save the battery from draining, to provide a larger display, more bandwidth or more processing power.*

Our infrastructure will be demonstrated by using a laptop which plays the role of a high-end terminal in the conference room and one or two handheld devices (Qtek 9090) playing the role of attendants' mobile personal terminals. All devices are connected to a wireless network. The Draco middleware and context infrastructure are installed on the laptop and the handheld devices. The laptop also hosts a repository of components and provides a model of the conferencing application which is shown in Figure 3. The application model is a loosely coupled composition of components providing support for jabber-based communication, slide shows, audio and video transmissions. The application model also states whether these components are mandatory or optional.

The laptop is able to discover in a peer-to-peer fashion the presence of the PDA and its resources using a service discovery protocol. Information on the PDA hardware specifications and available resources is passed on to this protocol by the context infrastructure. When the conferencing application is relocated from the laptop to the user PDA, the application model and knowledge on the available resources allow the infrastructure to adapt the conferencing application to optimize it for the mobile handheld device.

The context infrastructure provides location awareness on the PDA by monitoring the RSSI signal strength and SSID identifiers of known WiFi access points in the conference building. When Mr. Smith is leaving the conference room, the reduction in signal strength on the PDA triggers an event for the application to be relocated to the PDA and to be adapted to the new device. During relocation, the application remains active and its state is preserved. Thus, the contact list, the current conversation, the presenter's notes and the slides are not lost during transfer and redeployment.

4 Summary

Our context-aware infrastructure enables applications to be deployed on a broad range of hardware platforms and be adapted at runtime without any manual re-configuration or loosing the state of the application. Context information, such as the current location and resource usage, is used to drive the adaptation process. On the fly discovery of available resources with a service discovery protocol is used to increase the application QoS within in a mobile environment.

In this demo, we will show how our context-aware framework relies on the component-oriented design to support the deployment of context-driven adaptive services. We will illustrate our framework with a conference client application. This application is adapted and relocated according to the user context and available computing resources.

References

1. Weiser, M.: The Computer for the Twenty-First Century. *Scientific American* (1991) 99–104
2. Szyperski, C.: *Component Software: Beyond Object-Oriented Programming*, 2nd edition. Addison-Wesley and ACM Press (2002)
3. Vandewoude, Y., Rigole, P., Urting, D., Berbers, Y.: Draco : An adaptive runtime environment for components. Technical Report CW372, Department of Computer Science, Katholieke Universiteit Leuven, Belgium (2003)
4. Preuveneers, D., et al.: Towards an extensible context ontology for Ambient Intelligence. In Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L., eds.: *Second European Symposium on Ambient Intelligence*. Volume 3295 of LNCS., Eindhoven, The Netherlands, Springer (2004) 148 – 159
5. Preuveneers, D., Berbers, Y.: Adaptive context management using a component-based approach. In: *Proceedings of 5th IFIP International Conference on Distributed Applications and Interoperable Systems*. Volume 3543 of LNCS., Springer (2005) 14–26