eServices for Hospital Equipment

Merijn de Jonge[1], Wim van der Linden[1], and Rik Willems[2]

[1] Healthcare Systems Architecture
Philips Research, The Netherlands
[2] Strategy and Innovation Management/Technical Infrastructure
Philips Corporate IT, The Netherlands
Merijn.de.Jonge@philips.com, Wim.van.der.Linden@philips.com
Rik.Willems@philips.com

Abstract. In this paper we explore the idea that by combining different sources of information in a hospital environment, valuable e-services can be developed for reducing cost and improving quality of service.

Companies, like Philips Medical Systems, may have a competitive advantage, because they have a large installed base which may provide valuable information already, and because they can change their products to provide additional information.

To optimally benefit from this advantage, we created a platform that enables quick development of e-services. The platform enables uniform access to data, combines static with live data, and supports transparent composition of existing, into new services.

We discuss the requirements, design, and implementation of the platform, and we show its use in a case study that addresses asset management and utilization services for mobile medical equipment.

1 Introduction

E-services for hospital equipment are a means to add functionality on top of existing products. They can serve to improve the operation of individual devices, to improve the cooperation between devices, or to analyze/improve hospital workflow in general. They are promising because they can help to reduce cost (e.g., by increasing patient throughput) and to improve quality of service (including improving patient safety).

Promising in e-services is that with Philips we expect quick return on investment (because developing a service is far less expensive than developing e.g., an MR scanner), and that the value of e-services can be increased by combining different sources of information (e.g., equipment data, log data, workflow data, etc.). Threats are that we expect a significant need for change (to make services fit smoothly in different hospital environments), a strong competition, and quick market changes. Hence, there is a clear potential for flexible e-services, which combine different information sources, have a manageable development process, and a short time to market.

Services in hospital environments operate by processing and analyzing information. This information can come from equipment itself or from other information sources, like databases. By having more, diverse data available, more

B. Krämer, K.-J. Lin, and P. Narasimhan (Eds.): ICSOC 2007, LNCS 4749, pp. 391–397, 2007.

intelligent services can be offered. By integrating multiple information sources, the value of services can therefore be increased. Philips has a large hospital installed base. Via its diverse product portfolio, many different forms of information can be produced that can be used for innovative e-services. Philips can therefore take a strong position in developing e-services for hospital equipment.

Philips already collects information for (some of) its medical products. Although this information is not used for e-services yet, it forms a huge source of information, ready to be used. Most equipment, however, does not provide equipment data. Although this equipment might contain valuable information, it is simply not prepared for exposing it. The aim of our project is therefore to expose information from equipment and from (existing) data sources, and in combining these into discrimating e-services.

To that end, we have developed a platform for hospital equipment services. It has a service oriented architecture (SOA) that combines state of the art web services and thin client technology. The platform is structured around *push data* (events generated by equipment), *pull data* (information stored in databases), and *data filters* (which massage data for different use). In a case study we demonstrate the development of value added services for mobile medical equipment.

This article is structured as follows. In Sections 2 and 3 we discuss different forms of data sources and how they can be massaged by data filters. In Sections 4 and 5 we discuss the architecture and implementation of our platform. In Section 6 we address service development for mobile medical equipment. In Section 7 we discuss our results and contributions, and we mention directions for future research.

2 Equipment Data

We realized that existing data sets may contain valuable information for other purposes than maintenance, for which they where intended, and that our equipment could be extended to produce additional valuable information. We therefore distinguish two kinds of data: i) data that is already available in some database, ii) data that is produced by particular devices. We called these *pull data* and *push data*, respectively.

Pull data. Pull data is static data that is stored in e.g. a database. It is typically log-type information that serves maintenance activities. It is called pull data, because the data needs to be pulled out of the corresponding databases. This data might be very useful for developing new e-services. However, there are two bottlenecks for efficiently using this data: i) data access is difficult because data models are implicit and not standardized within and across products; ii) data processing is inefficient because data sets are huge. In section 4 we describe how we address these bottlenecks in our architecture for e-services development.

Push data. At any moment in time a device can inform the environment about itself. We call this *push data*, because the device takes the initiative to provide

information, rather than a service having to ask for it. All equipment data originates from push data, but once it is stored in e.g., a database it becomes pull data.

Push data fits in an event driven environment and enables real time responses to events, for instance in case of critical errors. In addition to maintenance related data (e.g. log messages), equipment can be adapted to provide other kinds of push data as well. This may give rise to numerous new services (see Section 6).

3 Data Filters

In Section 2 we argued that pull data is difficult to access because data sets are huge and have implicit data models. To address these problems, we introduce the concept of data filters. Data filters are elements in our architecture, which i) provide a consistent interface to data sets; ii) control data visibility; iii) provide virtual databases; iv) improve overall performance. These roles will be discussed below.

Explicit data models. Data filters can transform data from one format into another. One key application of a filter is to transform an unstructured data set into a structured data set, according to an explicit data model.

Control data visibility. Incorporating existing data sets for the development of (new) e-services, should not imply that all data is exposed, or that arbitrary data can be modified. To that end, we do not support direct access to databases (e.g., by supporting direct SQL queries). Instead, data filters precisely define which data is exposed by means of explicit interfaces.

Virtual databases. A filter implements a particular query on one or more databases. The resulting data set is a derived database, with its own data model. Filters can be seen as virtual databases, because they appear as ordinary databases, although they are created on the fly.

Performance. Filters can be used to optimize data for particular services, which can then operate more efficiently (e.g., because they have to process less data, or queries on the data become simpler). This form of optimization, creates particular views on data sets. Additionally, performance can be improved by caching. Instead of executing filters real time when the data is needed, they are executed at particular moments in time. The resulting data sets are stored for later access. This enables balancing resource consumption, and can lead to an improved and predictable overall performance.

4 Architecture

The architecture for our platform is designed to support push and pull data, to enable connectivity between equipment, and to support operation heterogeneous hospital environments.

Hospital environments are highly distributed environments, which bring together a huge variety of products and vendors. The market of e-services for medical equipment is quickly emerging. To efficiently deal with these complicating factors, we adopt a service oriented architecture (SOA) [7]. All parts of our architecture are services and have well-defined interfaces to the environment.

Data from databases is also made available through services. Observe that this gives a transparent view on data and data composition because databases and filters cannot be distinguished. The architecture supports push data in the form of events together with a subscribe mechanism (see Section 5).

Services are accessed from different types of terminal devices and from different locations. Consequently, we can make little assumptions about the equipment from which services are accessed and about the type of connection of this equipment to the hospital network. To that end, e-services are accessed via web applications, separating resource-intensive computations from user interaction. Only a web-browser is needed to make use of our e-services.

5 Implementation

For the implementation of our platform, we adopt state of the art technology. We base our SOA on web services [11] and use SOAP [10] as messaging framework. We used Java as programming language, but this is no prerequisite.

Web services are defined in terms of the Web Services Description Language (WSDL) [6]. Axis2 [4] is the SOAP stack for web services. Axis2 uses AXIOM [3] as object model, which performs on demand object building using pull parsing [9] technology. Both techniques significantly improve performance and reduce memory foot print, which is important for processing huge hospital data sets.

We use the Extensible Messaging and Presence Protocol (XMPP) as eventing mechanism [12]. XMPP serves as a distributed message broker. Event groups are defined for particular types of messages, such as for equipment status information. Events are generated by sending messages to these communities. Joining a group implies subscribing to corresponding events. Any entity in our architecture can create or receive events.

The Google Web Toolkit (GWT) is used for web applications development. GWT [8] enables programming AJAX [2] based web applications (almost) purely in Java. This significantly simplifies web application development.

We use the Central Authentication Service (CAS) system from Yale University [5] as the Single Sign On (SSO) solution for web applications. The CAS system enables users to log in once from their web browser, and be automatically authenticated to all web portals and web applications.

6 Case Study

In this section we discuss the development of a particular class of e-services for hospital equipment: web based asset management and utilization services for mobile equipment. These services build on top of existing asset tracking

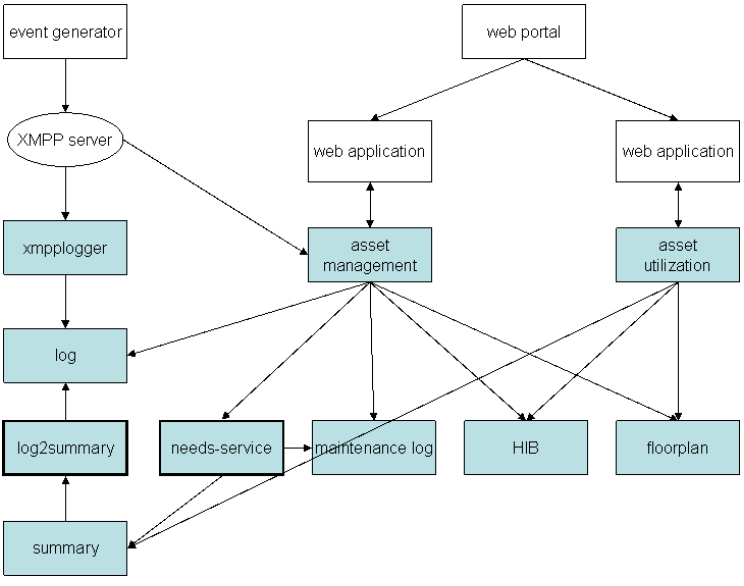

Fig. 1. Architecture of asset management and utilization services

technology, which is becoming a common technique in hospitals to monitor the location of (mobile) equipment. One promising technology for asset tracking is based on WiFi connectivity. Based on signal strength at multiple WiFi receivers, an accurate location of a device can be determined. Since WiFi is entering the hospital environment anyway, this tracking technology is relatively cheap and can easily be integrated and adopted. There are already several commercial solutions to asset tracking available (see e.g. [1]). Therefore, we do not concentrate on asset tracking itself, but we show that the combination with additional equipment data is a big step forward and significantly improves the usability of services for mobile equipment.

To that end, we develop services that combine location information, equipment data, and data from databases. The demonstrator shows how realistic services can be developed with our platform, although equipment is still simulated and data is generated. We focus on asset management and utilization services.

6.1 Implementation

In Figure 1, the structure of the demonstrator is depicted. Grey boxes denote web services, arrows indicate directions of method invocation, boxes with a thick border indicate data filters. The demonstrator consists of a web portal with two web applications, an event generator (which simulates mobile medical equipment), an XMPP server, and a number of web services.

The first web application provides asset management services. The second web application provides utilization services. The applications consist of a client

part, which can run in any web browser, and a corresponding server part, which runs as a servlet in an Apache Tomcat application server.

Mobile hospital equipment is simulated using a generator. It generates events at regular intervals for a predefined set of devices. These events contain location information and information about the status of the equipment. Events are broadcast via an XMPP service.

The "XMPP server" forms the heart of our event driven architecture. We group different types of messages in event groups. For instance, an equipment group is used for equipment events. The "asset management" service subscribes to equipment events in order to send location and status information of equipment to its corresponding web application. The latter visualizes this information in a hospital floorplan. The "xmpplogger" service saves equipment events for later use using the log service.

The services "log2summary" and "needs-service" are filters. The first transforms log data (which is a chronological list of equipment events) into daily summaries per device. This structure is useful for several utilization services. The second uses this summary data to determine whether a device needs preventive maintenance (see below).

Static information about equipment and floorplans are available through the "HIB" (Hospital Installed Base) and the "floorplan" services, respectively.

6.2 Asset Management and Utilization Services

The collection of services and the XMPP event mechanism depicted in Figure 1 form the ingredients for building services that go beyond plain asset tracking services. The real time status information that equipment provides in the form of XMPP messages is used directly in the "asset management" application to display equipment status. For instance, the load information of a device forms an indicator whether the device is available for use. The "log" service stores equipment events over time. This data is used for utilization services. For instance, we synthesize in what areas of a hospital particular equipment is used, how equipment is moved around, and what the average load of each device is. The "needs-service" is an example of a composite service. To determine whether a device needs preventive maintenance, the summaries of a device are analyzed to synthesize the uptime of a device, its average load, and its quality of service. Together with information about when it was last serviced (available through the "maintenance log" service), the predicate "needs service" can be derived.

7 Concluding Remarks

This article addressed e-services for hospital equipment. E-services provide functionality on top of or cross cutting existing equipment. We focused on e-services combining different information sources, such as databases with logging type of information, or equipment providing real time device and location information.

Contributions. We explained that by having access to large and diverse data sets and by having control over which data is provided by equipment, companies like Philips may take a lead in e-service development. We identified two sources of information: information stored in databases (which we called pull data), and real time data generated by equipment (which we called push data). Further, we proposed data filters to combine data sources, and to control what data comes available and in what structure. Next, we discussed the design and implementation of a platform for e-services, which adopts state of art technology. Finally, we performed a case study where we developed e-services for mobile medical equipment. In particular, we focused on asset management and utilization services for mobile medical equipment. The case study showed that the combination of information (i.e., location information, log type information, and equipment information) enables more advanced services than plain equipment tracking services can offer today.

Future work. We are in discussion of using our technology in a concrete pilot using real, instead of simulated equipment. Furthermore, together with our product groups we have plans to further explore the area of asset management and utilization services. Next, we can extend our approach for hospital equipment to also include people, enabling patient tracking and throughput services. Finally, we want to explore the options to enable third party integration, e.g., by corporatively defining a standard for e-services in hospital environments.

References

1. AeroScout (2007), Available at: `http://www.aeroscout.com/`
2. AJAX – asynchronous JavaScript and XML (2007), Available at: `http://en.wikipedia.org/wiki/AJAX`
3. Apache axiom (2007), Available at: `http://ws.apache.org/commons/axiom/`
4. Apache Axis2 (2007), Available at: `http://ws.apache.org/axis2/`
5. Central authentication service: single sign-on for the web (2007), Available at: `http://www.ja-sig.org/products/cas/`
6. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web services description language (WSDL) 1.1 (2001), Available at `http://www.w3.org/TR/wsdl`
7. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice-Hall, Englewood Cliffs (2005)
8. Google Web Toolkit (2007), Available at: `http://code.google.com/webtoolkit/`
9. Slominski, A.: Design of a pull and push parser system for streaming XML. Technical Report 550, Indiana University, Bloomington, Indiana (May 2001)
10. Simple object access protocol (SOAP) (2003), Available at: `http://www.w3.org/TR/soap/`
11. Web services activity (2007), Available at: `http://www.w3.org/2002/ws/`
12. XMPP standards foundation (2007), Available at: `http://www.xmpp.org/`