Using the Web and Jini to link Vending Machines and Enterprise Systems

Jan Newmarch
University of Canberra
Canberra, Australia
jan@ise.canberra.edu.au

Karsten Schulz
Distributed Systems Technology Centre
Brisbane, Australia
schulz@dstc.edu.au

Christian Merz
SAP Corporate Research
CEC Karlsruhe, Germany
christian.merz@sap.com

Abstract

The use of open technologies in enterprise resource planning systems, and the development of new distributed infrastructures allows for novel systems to be developed. This paper reports on a project to bring vending machines into the scope of ERP systems, using Jini for network discovery and event generation

1: Introduction

There has been a steady migration of information systems from interfaces based on proprietary operating systems to interfaces based on open Web standards. Major ERP (Enterprise Resource Planning) system manufacturers such as SAP have been re-engineered to use Web interfaces rather than Windows interfaces. SAP, for example, has .made the move from SAPGui based interaction to mySAP.com browser based interaction[1]. These Web interfaces allow distributed access to the enterprise information system. The enterprise systems are information systems, but often include information about physical devices and components used by an organisation. Typically, information about such devices has to be entered manually by a human operator. With the continuing increase in embedded devices, the devices themselves are more likely to be able to act as information sources in their own right,

This paper reports on an experimental project to link vending machines such as Coca-Cola machines directly into an ERP system. The technology used to perform this link is the emerging Jini technology from Sun Microsystems [2].

2: Jini

Jini is a distributed systems technology which is built on Java and uses various Java features to give it unique features among distributed system architectures [3]. A Jini "djinn" consists of services which advertise their presence and clients which locate these services and make use of them. Services advertise themselves by copying an object upto a lookup service or service locator. Clients search on these service locators for services of particular types. In O/O terms, the search is performed for services of particular classes. This is more type-safe than string- based searches used in e.g. CORBA [4] or Java RMI

[5] Naming Services. When a Jini client finds a suitable service, it receives a proxy object from the service which it runs in its own address space. This may be just a proxy stub such as an RMI stub, but the Jini architecture allows a much wider range of possibilities than just stubs. In addition to service discovery and deployment, Jini also includes an event mechanism [3]. This allows an object to act as an event generator, and can send this event to interested listeners. The generators can be services or clients, as can the listeners. This is a remote event mechanism.

3: SAP

mySAP.com comprises Web based business portals to cope with the needs of enterprise resource planning. Typical application areas are E- and M-commerce, Customer Relationship Management, Supply Chain management, Financials or Human Resources. Community driven (industry specific) Marketplaces or the personalized enterprise portal mySAP.com Workplace based on individual user roles are examples of business portals. Inventory Systems

The mySAP.com component R/3 provides an open architecture to be used by internal and external applications. External applications make use of Business API's (BAPI's) to access the R/3 business objects and call methods on these objects. BAPI's can be called by CGI queries making use of the middleware SAP Business Connector. As an example the module Material Management offers BAPI's to maintain goods movements in inventory management. The SAP inventory management system stores a number of attributes for each material in the database. Some of the required attributes are

- plant and
- storage location.

The inventory information for a material is altered by methods such as

- Goods movement types (e.g. ADD and REMOVE)
- Quantity
- Controlling objects (e.g. cost center) for cost distribution purposes (optional - depending on movement type)

The use of these parameters allow changes to be made by accessing a URL of the form:

```
http://sap\_host/invoke/CokeMachin/GoodsMvt?GM\_CODE=05\&
    MATERIAL=COKE\&PLANT=0001\&STGE\_LOC=0001\&
    MOVE\_TYPE=501\&ENTRY\_QNT=4.000\&
    COSTCENTER=0000000001
```

The major business benefits arise from the data collection of all Coke machines of a company. This leads to real-time key figures in the SAP backend systems. Follow-up events like ordering replacements when storage levels fall below a specific value can be triggered without any time delay. For customers making use of SAP warehouse management the vending scenario is applicable even more accurately. One can distinguish more specifics concerning storage location. This would avoid data maintenance of storage locations for every single vending machine. As a result, the refilling of vending machines can be planned more exactly. Only those vending machines that need to be refilled are being approached by a refiller and only the exact numbers of goods that are required for refilling have to be transported. Also, the prices of goods that are being sold by the vending machine can be adapted to reflect market demands.

4: Vending Machines

Vending machines proliferate everywhere, delivering food, drinks, cigarettes and other items. Keeping track of their contents or daily turnover is typically a manual process, and estimates have to be made for each machine of the frequency of checking their state. There is the famous drinks machine of Carnegie-Mellon University which has been on the internet for many years [6]. This uses the "finger" protocol to display its contents [7]. More seriously, the manufacturers of some vending machines have been looking at placing their machines on the internet in order to monitor their state [8].

Whenever an item is added to, or removed from, a vending machine, it changes state. This makes it a natural candidate for an event driven system, where the machine acts as event generator. Listeners can then respond to these changes in the vending machine state

5: Architecture

A vending machine acts as a JINI service, advertising a service of type InventoryItem-Container. The items that this service contains are of type InventoryItem. This service may be located by clients. The client does not make use of the service in the way typical of most clients. It simply registers itself as a listener for events generated by the service, and then waits for these events to occur. The service generates events and sends them to the registered clients. The service does not know what the clients will do with the events, and does not care. It does not even need to know where the clients are, since the clients find the service based on its registration with lookup services. The service is written without needing to know what the client does. The client has the freedom to communicate with any other system, log the events in files, and so on. This is a reversal of the standard Jini mechanism, where the client wants to perform a particular task and locates a service to help do this, without caring how the service performs its role.

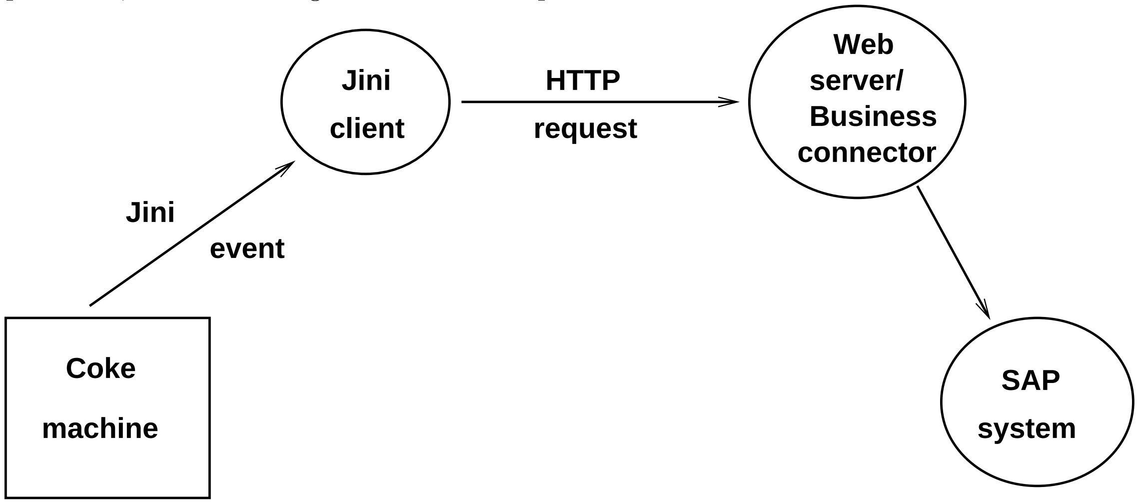

Figure 1: Information flow from vending machine to SAP inventory

The particular client implemented in this system uses the Web interface of an SAP inventory system. In effect, it uses the information from the service to prepare a CGI query [8] which is then posted to the URL for the SAP inventory system. The SAP system does not know that the query has come from a specialised client rather than from a browser. The information is shown in Figure 1.

6: Vending Machines as Business Objects

Many information systems such as SAP use an underlying relational database to store information. Jini is object based, and requires use of class/interface specifications to link services and clients. This requires a mapping between relational and object forms. For a pre- existing relational system this can often be quite straightforward. For example, an inventory item as stored in SAP requires fields:

- Material
- Storage Location

While an item in a vending machine may be expected to know what it is, it is less likely to know where it is located. This gives an InventoryItem specification as

```
public interface InventoryItem {
    String getMaterial();
}
```

The vending machine is itself a container of InventoryItems and can add and remove items in addition to knowing its location. In addition it must act as a broadcaster of events to listeners whenever its contents change. It will need to keep a list of these listeners. This leads to an interface

```
public interface InventoryItemContainer {
    public EventRegistration addRemoteEventListener(
                                RemoteEventListener listener);
    public int getStorageLocation();
    public void add(InventoryItem item);
    public void remove(InventoryItem item);
    public String getMaterial(long seqNo);
}
```

The last method in this interface allows the listeners to determine the type of material that was added to or removed from the vending machine. When an item is added or removed from the vending machine the add() or remove() method is called. This prepares a Jini event with type set to ADD or REMOVE and sequence number set to some suitable value. The value of the sequence number can be used by listeners to gain information about the meaning of the event.

7: Simulation

A simulator was written for the vending machine. This used a graphical user interface with two buttons "take can" and "add can". Pressing these called a listener which invoked the add() or remove() method on the InventoryItemContainer object running within a Jini service. From there on, the system was real. A Jini client registered itself as listener and talked to a real SAP system. In one demonstration the simulator and client were run at the SAPPHIRE conference in Brisbane, communicating with an SAP system in Germany. To make this into a real system will not be difficult. A vending machine can have an embedded chip with Ethernet connection. There are a number of these currently available, running a full-powered operating system such as Linux[9]. This will be capable of running Java and Jini without difficulty. When a customer buys something from the vending machine this can signal the chip, say through a serial port interface. A Java program can monitor this interface, through a custom-built Java Native Interface (JNI) program or by use of the Java commAPI [10]

8: Conclusion

A combination of Web technology and Distributed Object technology can allow an extra layer of hardware systems to be linked into ERP systems. A Web interface is used to communicate with the information system, and object technology is used to communicate with the hardware system. It is relatively easy to map between the two representations, and in future XML descriptions will make this even easier. With such a structure, the distributed event component of Jini can be used to link state changes in the hardware into HTTP requests for the information system. Future work will involve standardising the object interfaces, and building up a wider set of business objects from hardware objects.

9: References

[1] SAP, http://www.mysap.com
[2] K. Arnold et al, "The JINI Specification", Addison-Wesley, 1999
[3] J Newmarch, "Guide to JINI Technologies", APress, 2000
[4] A. Vogel and K. Duddy, "Java Programming with CORBA", Wiley, 1998
[5] RMI Specification, http://java.sun.com
[6] Carnegie-Mellon drinks machine, coke@cd.cmu.edu
[7] W. Stevens, "Unix Network Programming"
[8] CGI 1.1 Specification, http://www.w3.org
[9] Linux embedded devices, http://www.linuxdevices.com
[10] Java commAPI, http://java.sun.org