A Service-Based Architecture for In-Vehicle Telematics Systems

D. Reilly and A. Taleb-Bendiab
{D.Reilly, A.Talebbendiab}@livjm.ac.uk http://www.cms.livjm.ac.uk/emergeits

Abstract

The growing needs to access information in remote, mobile environments have sparked interests in so-called In-Vehicle Telematics Systems (IVTS) [1]. These relatively new systems have the potential to deliver computing facilitates to road vehicles, which may include in-vehicle infotainment, route-guidance and navigation and the provision of vital information resources used by fleet haulage companies and emergency services (police, fire and ambulance). This paper describes the EmergeITS[1] project, which is concerned with the use of IVTS for emergency fire service applications. In particular, the paper describes a distributed service-based architecture, based on the Jini middleware technology, which can be used to provide fault tolerant application services to remote in-vehicle computers and mobile devices such as Palm devices and WAP phones.

1. Introduction

In-Vehicle Telematics Systems (IVTS) combine mobile computing and telecommunications technologies to provide computing facilities within road-vehicles. IVTS may combine the use of small footprint mobile computers, running Windows CE, Palm devices and WAP phones. Centralized computing facilities and other remote systems are accessed through wireless links using GSM or WAP technologies. IVTS applications range from private vehicle owners, concerned with a pleasant, congestion-free journey, to fleet management and automatic vehicle location (AVL) for commercial companies right through to the emergency services (police, fire and ambulance) concerned with remote access of vital corporate information systems crucial to saving lives!

IVTS occur as distributed heterogeneous systems, which suggests a need for a distributed software architecture capable of gathering information about the operational context and mobile communication environment to "auto-configure" systems by adapting behaviour according to context and environment [2]. Middleware technologies such as CORBA, DCOM and Jini [3] play a crucial role in such an architecture by providing APIs and support utilities that bridge the gap between network operating system and distributed components and services. The remainder of this paper describes the service-based EmergeITS architecture, which uses Jini middleware to allow Merseyside Fire Service (MFS) to access centralized information systems as reliable, easy to use application services through in-vehicle mobile computers and Palm devices.

[1] EmergeITS is a collaborative project between the School of Computing and Mathematical Sciences at Liverpool John Moores University and Merseyside Fire Service (MFS)

2. Jini Middleware Technology

The idea of middleware is not new and IBM's CICS is an early example of *transaction-oriented* middleware. Current *object-oriented* middleware technologies rely on distributed objects and "stubs" or proxies through which objects can communicate using remote method invocation (RMI) and remote events. A higher-level *service-oriented* abstraction is often used in object-oriented middleware, which regards a service as a logical concept such as a printer service or chat-room service. This abstraction is used extensively in the Jini middleware technology, which provides an API that allows clients to use services through lookup, discovery and RMI mechanisms.

Jini applications essentially consist of three things: application services, clients and lookup services and code can be moved around, or "marshalled", between the three. Clients discover services by downloading a remote stub or proxy, implemented as a Java interface, and use Java's RMI to interact with the service through the proxy. Jini/RMI was selected as the middleware technology for EmergeITS, due to its capabilities to manage services remotely through leasing and activation mechanisms were the latter provides capabilities to implement "lazy" services, which are activated or started on demand.

3. EmergeITS Architecture

The main aim of EmergeITS was the development of a service-based architecture capable of managing application services, used by remote vehicles, according to the following requirements:

1. The architecture was to be suitable for use over a wireless link (2-3 sq. miles) with minimum intervention from the user.

2. The architecture must accommodate fault-tolerance and redundancy so that alternatives could be sought in the event of application service failure or unavailability.
3. Application services must be "lightweight" so as to not overload the limited resource in-vehicle computers.

To meet these requirements the architecture was organized into two service groups: lightweight, lazy application services to deliver information to remote vehicles and management services to oversee the use of application services and provide configuration and fault-tolerance capabilities.

3.1 Management Services

Management services monitor the operational and communication requirements by gathering information relating to application service usage context and communication services. This information is used to adapt and configure services to accommodate different modes of operation, communication environment and possible failure. Management services use Jini/RMI remote events to notify clients and application services of specific events of interest such as the availability or unavailability of an application service. They also use the Jini/RMI activation mechanism to start or activate remote application services and Java Virtual Machines, to provide remote management and fault tolerance capabilities.

3.2 Application Services

Application services provide remote access to information stored on centralized corporate database systems as well as download/upload services and route-guidance and navigation services. Application services subclass RMI's Activatable API to provide lazy services, which are activated on demand when a client request is received, thereby conserving memory resources.

3.3 Prototype Demonstrator

So far, the architecture has only been demonstrated using two Pentium laptops, fitted with IEEE 802.11 cards. Both systems were installed with Sun Microsystem's Java v1.3 and Jini v1.1. The laptop representing the in-vehicle computer was installed with Sun Microsystem's Java Micro-Edition (J2ME) and Palm device and WAP phone emulators. Several application services were demonstrated, but MFS expressed greatest interest in the *incident mobilization* and *hazard assessment* demonstrations.

The incident mobilization demonstration uses a Java Servlet, on the control computer, to produce an XML document from centralized database systems. The XML document, which can be displayed through a browser on the in-vehicle computer (Figure 1), contains information such as the address of the incident, type of building, significant hazards and fire-hydrant locations. In the hazard assessment demonstration fire officers use Palm devices to record significant hazards, during house-to-house calls, such as blind or disabled residents. The information recorded on the Palm devices is later uploaded to the centralized database systems and used in turn to provide significant hazards for future incident mobilizations.

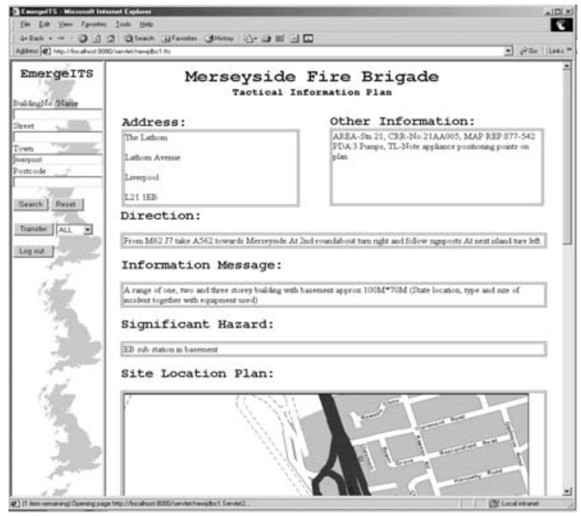

Figure 1: Incident Mobilization Browser

4. Conclusions and Future Work

EmergeITS has demonstrated the use of the Jini middleware technology for providing a service-based architecture capable of managing, configuring and providing application services to remote in-vehicle computers and Palm devices. Future work will consider a 3-in-1 phone service [4], whereby a mobile phone may be used as: 1) a cellular phone, 2) a WAP phone, or 3) walkie-talkie depending on user requirements and communication service availability.

References

[1] Heiss, J.J., "The Network is the Car: Driven by Java Technology", Sun Microsystems Inc. <http://java.sun.com/features/1999/06/concept_car.html>.

[2] Kanter, T., "Adaptive Personal Mobile Communication in 3G and Beyond", *Special Issue of IEEE Personal Communications – European R&D on Fourth Generation Mobile and Wireless Networks*, 2000.

[3] Jini Specification, Sun Microsystems Inc. <http://www.sun.com/jini>.

[4] Dowling, J.; Cahill, V., "The K-Component Architecture Meta-Model for Self-Adaptive Software", *in Proceedings of Reflection 2001*, Kyoto, Japan, September 2001.