Proceedings of the 5th WSEAS Int. Conf. on APPLIED INFORMATICS and COMMUNICATIONS, Malta, September 15-17, 2005 (pp281-283)

A Study of HMI on In-Vehicle Telematics System

YeonJun Choi', MinJeong Kim, YoungBaek Moon, SunJung Kim
Telematics S/W Platform Team, Telematics Research Division
Electronics and Telecommunication Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon
KOREA

Abstract: Designing in-vehicle telematics system is specialized for vehicle characteristics. A user interface called HMI(Human-machine interface) on in-vehicle telematics system is important not only sales but for safety and functionalities of vehicles. This paper is intended to suggest a way of designing a HMI system considering vehicle environment especially for drivers. The architecture of HMI provided in this paper is based on international standards, such as OSGi framework.

Key-Words: - telematics, middleware, OSGi, HMI

1 Introduction

Telematics becomes more realistic than ever. It is emerging industry providing intelligent services combining mobile and vehicle environments. Designing an In-vehicle telematics system for drivers is different from general systems considering safety and vehicle functionalities.

A Passenger enjoys services that catch eyes like movies, web-site surfing, and games. He or she also can select a service from complicated menus. But, in case of a driver, that kinds of services are nonsense. The services can cause accidents. So, the way of accessing menus and getting information is different in case of a driver from a passenger.

Human-machine interface on in-vehicle telematics system is researching area.

AMI-C suggests an HMI specification based on OSGi framework[1]. It is extensible but too sophisticated for an embedded system. The HMI devices connected over network must have intelligence analyzing and composing complex messages themselves.

W3C suggests Multi-modal framework as a way of implementation of the HMI architecture for in-vehicle environment. Multimodal framework has rich and flexible structure for a HMI system, but it's also heavy for an embedded system[2, 3]. It covers areas from mobile to home network, and full specification needs thick clients.

In this paper, we introduce a design of an light weight HMI architecture for an embedded telematics terminal.

2 Analysis

2.1 Telematics Infrastructure

Telematics technology, which utilizes various telecommunications such as DMB, CDMA, WRAN, DSRC, etc., is increasingly being applied to provide services for in-vehicle telematics systems. Telematics systems are also expected to upgrade existing services and expand safety-related services effectively using in-vehicle services in cooperation with external ser-vice contents. Fig 1 shows general telematics infrastructure and in-vehicle telematics system.

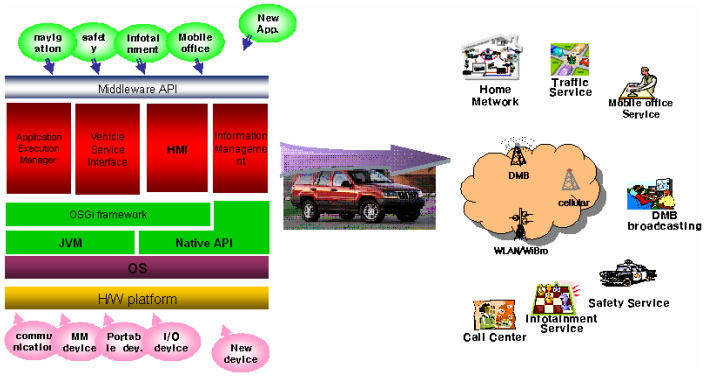

Figure 1-Telematics infrastructure and in-vehicle telematics system

To provide various services in telematics terminal, middleware should support following services[4].

- Communication service: uniform access for various kinds of communication such as CAN, MOST, CDMA, DSRC, Bluetooth, etc.

- Positioning Service: get a current positioning data of car in a uniform way
- Persistent Storage Management service(PS): management on persistent storage shared between applications and services
- Vehicle Information Management service: provide vehicle static & dynamic information from vehicle
- TSP Connection service: process an application protocol for telematics services between a terminal and a TSP(Telematics Service Provider) center
- Application Execution Management service(AEM): application execution characteristics in vehicle environment (status like suspended, resumed) and application priority management
- Vehicle Service Interface(VSI): AMIC based network neutral messaging schemes and APIs
- Personal Information Management Service: personalized information management framework for adaptive vehicle and application services
- HMI Service: access to vehicle equipped I/O devices

2.2 Requirements Analysis

A HMI system of in-vehicle telematics system requires some conditions for safety and usability of HMI resources as follows:

2.2.1 Uniform interface

In-vehicle HMI system provides user follow a uniform way of accessing input and output devices.

It simplifies interfacing I/O devices of the vehicle and terminal a driver uses. It also provides flexibility of adding and removing devices through uniform interface.

2.2.2 Manageability

In-vehicle HMI system manages all user interface resources provided by terminal system and the resources must be used by user and user applications through HMI manager.

2.2.3 Priority

In-vehicle HMI system manages priorities of user applications that uses HMI resources for successive or prior execution on the resources.

2.2.4 Registration service

In-vehicle HMI system registers and manages the devices equipped in the vehicle for means of input and output.

3 Design

3.1 HMI architecture

An In-vehicle HMI manager provides functionalities listed above section.

By requirements, the HMI manager consists of 7 subsystems, XML generator, XML parser, UI manager, Environment manager, Priority manager, Input manager, Output manager as [Figure 2].

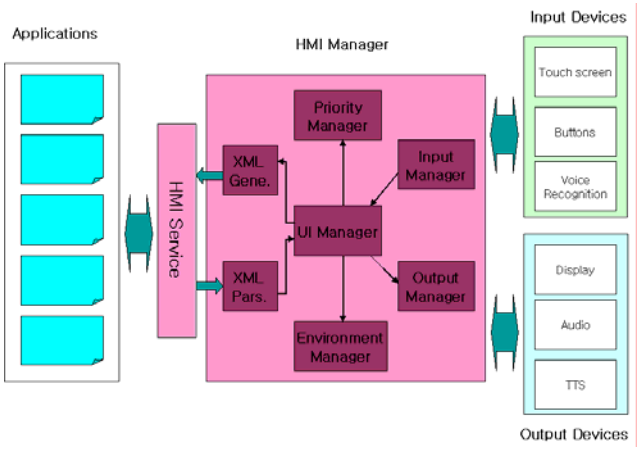

Figure 2- Architecture of HMI manager

We consider input actions are restricted to touch screen, external buttons and voice recognition.

Output actions are restricted to display, audio and TTS[1]. It also covers video player..

The functionality of each subsystem is as follows:

- HMI manager interfaces with applications providing HMI services.
- XML parser parses XML-formatted HMI message.
- XML generator creates uniform XML-formatted HMI message.
- UI manager manages input and output functionalities and serves as HMI services to applications

[1] TTS : Text-To-Speech

- Priority manager maintains priorities of application for the HMI resources
- Environment manager has device information and registration services
- Input manager manages input devices and collects input messages
- Output manager maintains output devices

3.2 Algorithms

There are three main flows for HMI functionalities.

First, input flow. An input flow starts from input occurrence and ends to application execution.

Second, output flow. An output flow starts from application request for occupying the HMI resources and ends to executing them.

Last, registration and management of HMI resources and application priorities.

3.2.1 Input flow

When an input occurs, the input algorithm starts as follows.

1. The appropriate input device catches the signal .
2. Input device sends input message to Input manager.
3. Input manager analyzes the message and translates it to an uniform message. Unlike multimodal framework, multiple input does not allowed.
4. UI manager sends the message to destination application. In this case, destination is always application invocation. So Application Execution Manager executes indicated application.

3.2.2 Output flow

Output flow is a little more complicated than that of input.

1. If an application wants to use HMI output, it must compose a HMI output message.
2. The application sends a request of a HMI output resource to HMI manager.
3. XML parser analyzes the message.
4. UI manager checks the requested HMI resources is available or not.
5. If the resource is free, UI manager allocates the application to the requested HMI resource.
6. If the resource is in use, it checks the priority of the newer requester-application.
7. If the newer one has higher priority, UI manager stops the resource running and get rid of older requester and newly allocates the resource to the newer requester.
8. If the newer one has lower priority, UI manager puts the newer requester-application into the waiting queue.

3.2.3 Registration and Management flow

The HMI resource management is as follows:

1. The HMI resource sends a message that it wants to be registered to the HMI manager.
2. The HMI registers it as a specific HMI resource with a unique name.
3. The HMI can unregister HMI resources.

The application priority management is as follows:

1. The HMI manager maintains a priority table by Run Level.
2. All application priorities are compared by the priority table.
3. If two successive applications' Run Level are same, newer requester-application waits until prior one finishes using the HMI resource.

4 Conclusion

The system design is based on the OSGi framework and AMI-C specifications using application management and vehicle services.

We've almost finished the design phase of the HMI system and now the system is in implementation phase on a reference embedded board with RTOS.

In the next paper, we will introduce our implementation of the HMI manager for in-vehicle telematics system.

References:

[1] AMI-C Requirements and Specification for HMI, AMI-C, *http://www.ami-c.org*, 2002

[2] A Study on e-Learning in Vehicle Environment, Y.J. Choi, et al, WSEAS, 2004

[3] EMMA, W3C, *http://www.w3c.org/emma*

[4] Analysis and Design of Middleware Architecture for In-vehicle Telematics Applications, M.J. Kim, et al, *ITS Congress*, 2005