Interoperability of Smart Card Application

Yibin XU* Shigeki YOKOI* Takami YASUDA**
*Graduate School of Human Informatics, Nagoya University
**School of Informatics and Science, Nagoya University

In this paper, we have analyzed the reason that today's smart card lack of application interoperability with considering both the card and the card terminal. Two approaches of solutions: building external software architectures to handle different types of cards and card readers, and improving the inner architectures of the card and card terminal have been discussed. Some important specifications and architectures have been summarized with comparing on their features and working scope.

1. Introduction

Today each card and terminal manufacturer develops its own card and terminal using its own unique operating systems. The application developed for these cards and terminals is bonded to a specific card and terminal's operating environment. This situation has resulted in several key problems of the smart card: difficulty in supporting multiple applications, excessive development cost and extended time to market. Therefore, application interoperability has become a key point in expansion of the smart card technology. In this paper, we discuss the features and solutions of application interoperability in smart card.

2. Application interoperability of today's smart card

2.1 Mechanism of smart card solution

All smart card applications are performed by smart card solutions. The simplest card solution includes a card and a card terminal. The card terminals are computing platforms with a card reader, an I/O device into which the card is inserted. The card and terminal each has an independent operating system and a corresponding application. In fact, all applications of smart cards include two parts, one runs on the card and the other runs on the terminal. We distinguish these two parts as card application and terminal application. The terminal application provides a dialogue to "talks" with a card user, and transfers his requirement to the card. The card executes the command and sends back a result to the terminal. A more complicated card solution includes networks and servers (see figure 1).

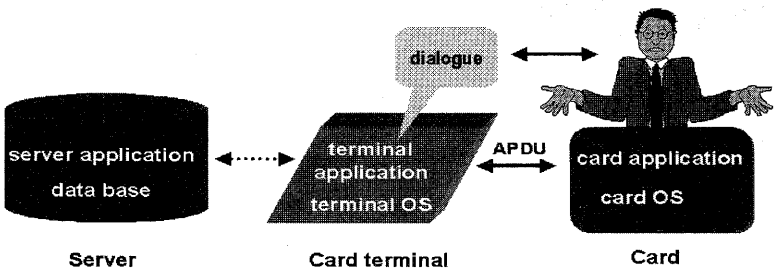

Figure 1. Smart card solution.

2.2 Origin of problem

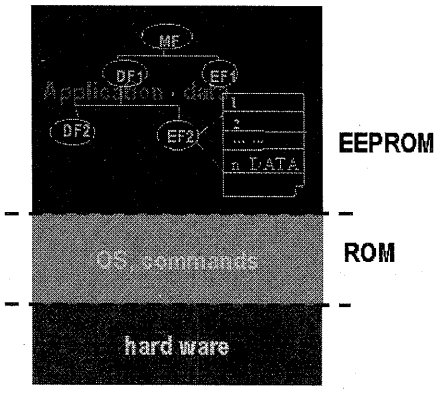

Figure 2. The architecture of a traditional smart card.

The architecture of a smart card is shown in figure 2. The card OS is designed to support a set of low-level commands, for example READ RECORD, UPDATE RECORD, etc., and the commands are card OS dependent. The card OS and command are installed in ROM when the card is produced, and can not be updated after. The application data is stored in Electrical Erasable Programmable ROM with a file system that has directories forming a hierarchical tree like structure. At the top of this tree (see Figure 2), is a Master file (MF). This file is mandatory. Leading off the Master File, there are either Dedicated Files (DF) which can be regarded as application directories or Elementary Files (EF), which are used to hold data.

In the majority of cases, smart card applications will use the card simply to hold small amounts of data with appropriate security around it (for example, access key). Therefore, developing a card application is a process of creating a data layout for the application, which is always card OS dependent.

The terminal applications included a set of instructions to be sent to the card through the card readers. The instruction includes a command and the data location. For example, if we want to read a data "DATA" from the EF2 file in figure 2, we send the command as:
Select MF, DF1, EF2 and read "DATA" from EF2 at Offset n.
The application has to be modified each time changes are made in the card data structure. Moreover, the card instruction must be sent via a card terminal API. However, each reader usually comes with a unique set of APIs. This cost time and effort to the application developer each time a new vendor's card or reader is introduced. The application development process is shown in figure 3.

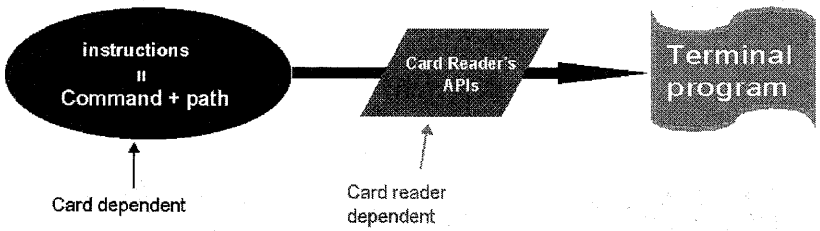

Figure 3. Application development for traditional smart card.

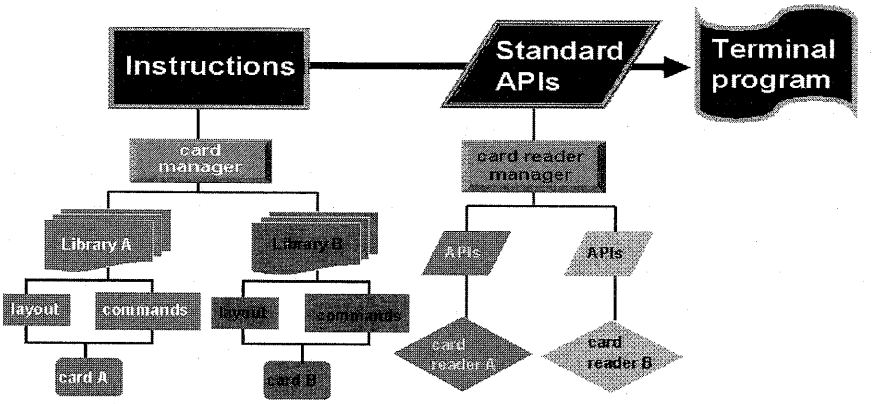

Figure 4. Architecture for managing various cards and card readers.

Due to the reasons above, there is no application interoperability in smart cards. Each card application is designed for a specific type of cards and card readers.

3. Solutions to application interoperability

Generally, there are two ways to approach application interoperability for smart card. One is building external software architecture to handle various cards and card readers, another is improving inner architecture of the card and terminal.

3.1 External software architecture

3.1.1 Conception

The conception of handling different cards via an architecture is to build a library for each type of card, which contains the card specific information, such as command and data structure. The applications, which are independent on the card, get the information through the library and communicate with the card. The architecture is built to handle the libraries for different types of cards.

The same thing may be done for the card readers. Architectures are built to manage the card readers' specific APIs. Standard APIs are provided to the developers for programming, and when the application is executed, it is translated to the specific APIs for each card reader (See figure 4).

3.1.2 Important architectures and specifications

In this section, we introduce some important architectures for smart cards and card terminals.

(1) for handling smart cards

IBM smart card toolkit --The IBM Smart Card Development Toolkits[1] is designed for assisting in creating data layout on the card, managing the card specific information, such as card layout and card data structure, and helping the terminal application developer to develop card-independent applications. The differences of card reader are not considered in IBM Smart Card Development Toolkit.

(2) for handling both cards and card terminals

PC/SC (Personal Computer/Smart Card) Interface[2] -- published by the PC/SC Workgroup, which covers most of the leading companies of smart card industry, such as Bull, Gemplus, IBM, Microsoft, Schlumberger, Sun

Microsystems, Toshiba, etc. Microsoft Corp. owns and maintains the PC/SC specification. The PC/SC workgroup identified three areas to standardize: The interfacing of card terminals to the PC; The high-level terminal APIs; And the mechanisms to allow multiple applications to effectively share the resources of a single smart card and card terminal. PC/SC is now the most widely supported standard for card reader, and it addresses to a limited extent to card operating systems. However, right now, PC/SC only addresses the PC Windows 9x/NT 32-bit platforms (PC/SC requires multithreading).

Open Card Framework (OCF)[3] -- announced by IBM, Netscape, NCI, and Sun Microsystems, which target platforms are network computers, Web browsers, or any future platform that runs Java and has to interact with smart cards. It provides architecture for various cards and card readers management and a set of terminal APIs. The terminal applications are coded in Java, so they are naturally presented with the capability for Internet application. For Java Cards, OCF provides the services of downloading card applications from the Internet and installing them to the card.

3.2 Improvement of inner architecture

3.2.1 interpretive smart card

An interpretative smart card has an interpreter locating above the card operating system. It supports the card to run programs that may be written in a standard language. And it tries to resolve the problem of rigid data structure and direct data access method in traditional smart cards. The data in an interpretative card is managed by each card application. The instruction to an interpretative card is no necessary to indicate the data location.

As well as we know, there are two types of interpretative card operating systems, JavaCard and Multos. Since their architecture and mechanism are similar, here we take Java card as an example to show how the interpretative card works.

Figure 5 shows the architecture of a Java Card.

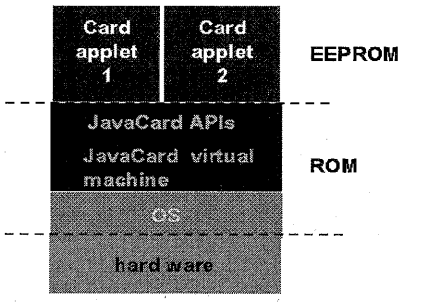

Figure 5. The architecture of a Java Card.

In Java Card, a JavaCard virtual machine is built above the card native OS, which support the card to run programs written in JavaCard APIs[4]. Each card application (card applet) manages its application data as parameters of the object. For example, a card applet "Cardholder", which stores the information of the cardholder, has parameters of "Name", "PIN", "SecretKey", etc. If we want the parameter "Name", we send a command as:

Select cardapplet "Cardholder", read out the "Name".

Since the instruction to the card doesn't include the data location, the terminal application can be developed without card specific information.

3.2.2 Java terminal

Addressing various OS and APIs of card terminal, a Java Virtual Machine can be adapted to the terminal platform. A Java

terminal has a similar architecture as the Java Card, which uses Java as a standard programming language.

Therefore, for Java Cards and Java terminals, the application can be developed without dependence on the card neither the terminal.

3.2.3 Standards and specifications

Here are some standards and specifications for interpretative card and Java terminal:

MULTOS specification[5] – the first specification for interpretative smart card. It is openly licensed and controlled by international organizations MAOSCO Consortium, which includes 14 world leading companies, such as American Express, Europay International, MasterCard International, Fujitsu Group, Hitachi, etc. Up to now, Multos is running on its specific card OS. MEL (MULTOS Executable Language), the language used by Multos cards, is an assembly programming language.

JavaCard Specification[4] -- A specification form Sun Microsystems. Introduced in 1996, the Java Card platform has been widely adapted by the card OS designers (with over 30 licensees representing more than 90% of the manufacturing capacity of the smart card industry). The language used by Java cards is JavaCard APIs -- a subset of the generic Java language.

Visa Open Platform (VOP)[6] – specified by Visa Group for its members. Addressing the card, VOP has selected Java Card as the working platform, and addressing the card terminal, Visa has developed a Java virtual machine and terminal APIs, which fits small terminals (such as a mobile phone, TV set top box, etc.) in size and functionality. VOP also provides tools for card application management and personalization, which enables cards to be customized and issued in a systematic way.

3.3 Co-operative and competitive specifications

Table 1 lists the software architectures and specifications we have mentioned above with the scope of their coverage.

It is important to understand that these architectures and specifications are established addressing different aspects in smart card solution. Many standards and specifications make reference to and use aspects of other standards. So, even though they may appear to redefine parts of other specifications, in fact the spirit of those specifications are attempting to address a completely different aspect of smart card standardization. These specifications are considered complementary rather than competing - complementary with respect to the scope of their objectives as well as to the environments in which they will be deployed.

However, when a smart card solution is built, we need to make a choice among the cards, terminals and architectures. The selection should be made with considering of the operating environment for the smart card application. For example, if we are targeting non-Java applications for Windows and need only to support a particular card issuer, PC/SC may be an obvious choice due to its wide support. If the terminal application is for using in the Internet and based on a Java platform with sufficient memory resource, such as a computer, OCF will be a good choice because of its powerful Java virtual machine and full

Table 1. Working scope of specifications and architectures for smart cards and card terminals

	JavaCard Specification	Multos Specification	VOP	IBM smart card toolkit	PC/SC	OCF
Card architecture	√	√				
Card APIs	√	√	√			
Various cards handling				√	√	√
Card application management			√			√
Terminal architecture			√			
Terminal APIs			√		√	√
Various card reader handling					√	√
Programming language	Java	MEL	Java	C, C++	C	Java
Terminal platform			A wide range of terminals (PC, ATM, mobile...)	Windows 9x/NT, OS/2 Warp environments.	Windows 9x/NT	Platforms running Java (network computers, Web browsers...)

functionality. If we want to develop our applications with Java, running on Java cards and terminals with limited resource, VOP is the only choice available now.

4. Conclusion

Card-dependence and card reader-dependence are two main barriers for smart card application interoperability. The solutions can be approached form two ways: building external software architecture, or improving inner architecture of the card and terminal.

Some software architectures have been provided to handle different types of card and card reader. Nevertheless no of them is able to cover all types of card, neither the card readers. It is a question whether it is possible (or necessary) to build a huge architecture for handling all types of cards and card terminals.

Improving the inner architecture of smart cards and card readers is a direct way to realize the application interoperability. In addition, the best application coordination and efficiency can be obtained. Comparing with Multos card, Java Card has been accepted much widely and quickly by smart card industry. It is due to the enormous successes of Java in other fields such as Internet, and the wide acceptance of Java by various computers and devices. With Java running on the card, the terminal and the server, the smart card application can be designed and developed in a coordinate and systematic way.

Reference

[1] Jorge Ferrari, Robert Machinnon, Susan Poh and Laskhman Yatawara: Smart Cards: A Case Study, International Technical Support Organization, SG24-5239-00, (1998) pp. 123-131

[2] "Interoperability Specification for ICCs and Personal Computer Systems", Bull CP8, Gemplus SA, Hewlett-Packard Company, et al., Revision 1.0, December 1997

[3] "OpenCard Framework 1.1.1 Programmer's Guide", OpenCard Consortium, Third Edition, April, (1999)

[4] "Java Card™ 2.1 Virtual Machine Specification", Sun Microsystems, Inc, Revision 1.0, March, (1999); "Java Card Applet Developer's Guide", Sun Microsystems, Inc, Revision 1.12, August, (1998)

[5] "A Guide to the Multos Scheme v.1.1.2", MAOSCO Ltd., August, (1999)

[6] "Visa Open Platform Overview", Visa International Service Association, April, (1999)