

Introducing Research Issues for Next Generation Java-based Smart Card Platforms

Gilles Grimaud

RD2P Labs, University of Lille, France
gilles.grimaud@lifl.fr

Jean-Jacques Vandewalle

Gemplus Software Research Labs, France
jeanjac@research.gemplus.com

Abstract

This paper introduces the issues and challenges of next generation Java-based smart card platforms. Betting on a continuous evolution towards open computing devices, next generation cards will consist in embedded Java micro-server platforms. Those platforms will be able to serve various types of services and applications thanks to two important system features: adaptability and maintainability. Two features that will have to be carefully implemented in the research agenda's topics described in this paper: real Java for cards, cards integration in a networked world, and flexible and adaptable cards.

1. Introduction

1.1. Motivations

"The future is unknowable and difficult to plan... Smart cards will have to adjust to the environment in which they find themselves operating." The previous sentence briefly sums up the seminal motivation to propose a research agenda for next generation Java-based smart card platforms (later on referred as RANGC). Two elements in this introductory sentence are worth to be pointed out.

1. First, *"the smart card adjustment to the environment"* reverses the current habits in which typically smart cards are deployed in an environment that is prepared to host smart cards according to the requirements imposed by the card interface such as the legacy smart card protocols. On the top of that, the environmental software is typically tailored to interact with smart card applications according to requirements imposed by the functionality provided by the card after it has been issued. *Here, one part of the RANGC is to focus on the adaptability of the smart card systems that is essential to allow smart card applications to adjust their behavior to the environment.*
2. Second, *"the environment in which smart cards find themselves operating"* highlights the fact that smart cards will play a key role in future infrastructures if they are sufficiently generic for being able to discover their environment and to inter-operate with it. *Here, the other part of the RANGC is to focus on the generic nature and the maintainability of smart card systems that are essential to allow smart card devices to persist in a changing environment.*

1.2. Global vision

The proposed research agenda is based on the vision of generic, adaptable and maintainable smart cards that will meet

the requirements of both the adjustment of smart card systems to the environment and the persistency of smart card applications in an evolving environment. The technology used to implement these smart card systems blur classical boundaries such as those between distributed and embedded systems, those between low- and high-end card platforms, or those between pre- and post-issuance (the term "post-issuance" describe the ability for smart cards to host and to run applications after they have been issued [1]). Thus, the first mission of the research agenda should be to bring consistency to the possible solutions to these issues.

The emergence of powerful and personal services supported by multi-purpose devices has the potential to dramatically enrich the range of applications that smart cards will be able to serve. With the right platform and infrastructure in place, these smart cards will radically leverage card businesses as they enable card makers to offer to their customers solutions to implement and deliver complex and flexible business services. In consequence, the second mission of the research agenda is to participate in the understanding of this new possibilities by explaining and describing the research topics and their challenges.

1.3. Remaining of this paper

In the next sections, the research agenda is detailed with the following topics:

- *"Real Java for smart cards"*; reminds the central place of Java in smart cards, its limitations and disadvantages with the current Java Card 2.x specifications, and the will to provide a new ground-breaking release with the Java Card 3.0;
- *"Cards integration in a networked world"*; extends the notion of interoperability from the current point of view of portability of Java Card programs to any Java Card platforms to integration of Java Card programs with other programs residing outside of the card;
- *"Flexible and adaptable cards"*; refines the use of Java to support multi-application smart cards, to the use of Java to support well-tailored customization of the card system depending on the target needs.

2. Real Java for cards

2.1. State-of-the-art

Java Card [2] is now seen as the dominating platform for high-end microprocessor-based smart cards. And according to market analysts [3] this trend should continue for at least the next four years. Smart card vendors are focusing their strategy on this technology and an important part of their research and

development resources is working on Java Card. There is no important development of a proprietary multi-application operating system in none of the major smart card companies.

If Java Card lives up to its promise, Java Card is not without its challenges. If we take a look at Java, its main strengths are a well-designed object-oriented language, cross-platform compatibility, virtual machine support for application sandboxing, garbage collection to guard against memory leaks, tight integration with other Java systems (back-ends, servers, three-tier architecture, *etc.*), and its industry and tools support. Java Card takes benefits from some of these strengths but it misses lots of them because of its stripped-down version to adapt to small devices. This stripped-down version has resulted in choices introducing issues that are difficult to solve without a re-foundation of the specifications.

2.2. Issues

These issues with the Java platform as it is defined by the current Java Card specifications are three-fold:

1. The very stripped-down version of Java provided by the Java Card 2.x specifications is targeted to very low-end chips. Therefore, drastic choices have been made that yield to very poor specifications in term of functionality compared to “standard Java”. For instance, these drastic choices are optional integer numbers, no multi-threading, no (or optional) garbage-collection, specific file format different from the class file format that prevents on-board linking and reflection, a persistent memory model, *etc.*
2. Due to the specialization of the Java Card specifications to the smart card specific device and its constraints (for instance, see the APDU class specification), the Java Card platform is hard to understand, to handle, and to master by traditional Java developers that do not have any competencies in smart card technologies. Moreover, this specialization does not permit to take benefits from the plethora of standard Java tools and has not encouraged the creation of interoperable, productive and easy-to-use tools to help smart card developers build applications.
3. Finally, because the Java Card specifications only consider the execution platform (virtual machine and runtime environment) plus the standard APIs, they miss some aspects of the smart card “way of life” such as its initialization, its personalization, the way applications are installed during the pre-issuance stage, or during the post-issuance stage.

2.3. Challenges

In order to overcome the limitations and difficulties raised by the above mentioned issues (mainly, the ones in the first two categories), the Java Card Forum has decided to stop adapting the Java Card 2.x with minor revisions. A work has been started towards the specifications of a new major ground-breaking release (code-named “3.0”) with the following essential initial principles: to target high-end (and possibly at the same time low-end) 32-bit chip platforms and take benefits of their new hardware capabilities, to get closer to mainstream Java functionality, to ease development and deployment processes, and to bypass the bottleneck of the

specific smart card ISO 7816 communication protocols. The RANGC clearly encloses these principles. But it is important to note the RANGC should not solely adopt an on-card standpoint (from Java Card 2.x to Java Card 3.0) but should also consider a system standpoint in which a smart card is seen as part of global systems, as well as a flexible system by itself. These two last aspects complement the vision of a radically new and richer Java-based platform for smart cards. They are discussed in the two following sections.

3. Cards integration in a networked world

3.1. State-of-the-art

One of the most promising feature for paying the price of Java in smart cards was the cross-platform compatibility brought by the use of Java and by the “standard” Java Card specifications. Looking backward, we now know that it wasn’t painless. For instance, mobile networks operators using a Java Card from one card manufacturer had to virtually redevelop the applications to run on different vendor’s Java cards. Card vendors have made substantial progress during the past year to make the cards interoperable. In GSM business, SIM card vendors are stepping up their efforts to bring about true interoperability through the SIMAlliance they have formed. Outside of the GSM world, Visa and the U.S. government are concentrating their efforts to ensure that the applications they wish to run on smart cards work on cards from multiple vendors. Thus, cross-platform compatibility can still be seen as an important challenge the card industry has to cope with.

The need for a true interoperability of the card platform within information systems is one of the feature for which it is worth paying the price of Java in smart cards. This kind of interoperability can be better described as the set of technologies for card integration in information systems. So far, some limited efforts have been done on this topic. They are limited to generic, multi-purpose sets of APIs for supporting interoperability of smart cards and card readers within the Windows operating system series (PC/SC), within the Java environments (OCF), or within the Linux world (Muscle). This sets of APIs suffers from incompatibility and limited functionality because they are stuck with the card platform capabilities and because they only abstract the communication link to smart card applications to the level of messages exchanged with client applications running on the terminal where the card is plugged in. Messages have to be constructed by hand according to the card application specification, multiple and multiplexed communication channels are difficult (even impossible) to manage concurrently, reverse communication from the card applications and distant accesses from the network are not supported.

Efforts to overcome some of these bottlenecks are appearing with the 2.2 release of Java Card [4] in which the Remote Method Invocation (RMI) technology enables better integrated distributed operations from Java client applications, and opens up the door for distant accesses as it has been shown using the Jini technology [5]. Also, the Java Card 2.2 release offers multiple logical channels (at most 4), which enables multiple client applications communicating concurrently with multiple card applications. Nevertheless, all of these initiatives are still quite limited technologies to

achieve an efficient integration of card devices and operations within networked applications.

3.2. Issues

Networked applications is about connecting information systems and exchanging information. They provide a great opportunity to make information more convenient to use and to be pieced together in order to achieve a common goal such as a complex service requiring, for example, distributed data and distributed processing. Fueling personalized services with power and flexibility is the key issue for getting new products to market more quickly and with the ability to make these products evolving according to the customer's needs. Notable examples of this evolution are the numerous attempts to deploy m/e-commerce applications carried out by many companies. These applications mix two visions that are apparently conflicting:

1. the assembly of complex and context-rich data (catalogues, orders, locations, contracts, *etc.*) with the cooperation of various information system sources (manufacturers, retailers, brokers, bankers, *etc.*), and;
2. the users' needs, because users expect services that are personalized, ubiquitous, non-intrusive regarding their privacy, secure, and fitted to their ever changing habits.

To summarize, information must be at the same time gathered for the client goals and available from different places with strong constraints of trust and convenience.

Though smart card platforms have been recognized as a key technology to leverage the development of such applications they suffer from many handicaps--e.g., their reduced internal hardware and software features, their very specific hardware and software interfaces, and their low-level development tools—among them their lack of interoperability with information system is a crucial one because it prevents easy ways of building end-to-end solutions incorporating smart card operations.

3.3. Challenges

The research agenda should aim at providing application developers with Java Card supported technologies that will ease the card integration, and that will support the end-to-end argument [6]. These technologies range into the following categories:

- a standard communication protocol stack able to interconnect with diversity of networks, allowing distant accesses, and supporting multiple bi-directional communication exchanges at the same time;
- the support of multiple (non predefined and non frozen) application models to be able to interconnect with any current or future information system, such as information systems based on message-oriented middleware, mobile code infrastructure, or object-oriented invocation broker;
- the use of "standard" Java components to take benefits from the existing Java tools and from the widespread use of Java components (file format, idioms, APIs, *etc.*) in the information systems surrounding smart cards.

4. Flexible and adaptable cards

4.1. State-of-the-art and issues

Java Card has been marketed as the ideal platform for multi-application smart cards. Technically speaking, it is not erroneous. The flexibility to offer new services and to update data without physically swapping out the chip card is supported by Java Card. But there is a missing gap in Java Card security and management characteristics because the Java Card specifications do not define how an application is uploaded or removed from a card, nor who may add or delete applications from a card. Visa played a central role in promoting a system called Open Platform that specifies how these functions ought to be performed. In late 1999, Visa turned Open Platform to an industry consortium, GlobalPlatform, hoping to create a standard card manager specification that would be used by many industries.

4.2. Issues

The features provided by the GlobalPlatform application-management piece are dedicated and specific to the current state-of-the-art Java Card platforms. Moreover, they only bring flexibility for the management of the card applications and not for the platform itself.

One issue is to broaden this flexibility to the platform itself in order to make the next generation Java-based card platform adaptable to different needs according to the content and service providers requirements. This broaden flexibility and adaptation capabilities can be implemented in two ways: either statically for building the well-tailored Java-based card platform configuration before issuance, or dynamically in order to adapt on the fly the Java-based card platform to new application management rules or to different application models from those installed originally.

4.3. Challenges

Finally, the research agenda should aim at providing application issuers with Java Card supported technologies that will ease the card flexibility and adaptation in order to allow the platform to incorporate at best the only needed features for the context in which they operate and the applications they serve. These technologies range into the following categories:

- a dedicated and powerful dynamic linking model (DLM) adapted to smart cards. Today the smart card linking model is mostly off-card and static though DLM provides one of the main benefit of Java. DLM enables just-in-time code loading and linking, and thus implies minimal space consumption that is crucial for limited devices. But smart cards as well as others embedded devices; use static models (*e.g.*, for classes preloaded in ROM). The goal is to extend the Java DLM to allow coexistence between dynamic and static linking models;
- an extended application-management component able to dynamically manage the applications and the platform extensions (like application model required libraries and services). To that extent, a key feature of the Java platform is the dynamic code loading supported by the Class Loader component. This component does not exist in Java Card because the platform uses exclusively a static linking model. Without a Class Loader component,

code management (and thus application management) [7] is difficult to extend and to adapt in the context of a virtual machine that acts as an application server and that supports code mobility;

- the introspective nature of the Java-based card platform itself, which enables to apply instrumentation and architecture/assembly/factory tools for statically composing and building a dedicated configuration of the platform that reflects the state suitable for its usage;
- complementary to the previous point, the design of some pieces of the Java-based card platform as pluggable components that could be chosen among available instances (for instance, data-link layers for the communication stacks, applications models-required libraries) in order to configure the platform to its target usage;
- the use of “standard” Java mechanisms and components (for instance, serialization or standard Java service management technologies) to implement the above-mentioned technologies while taking benefits from the existing mechanisms, and leveraging on their associated tools.

5. Conclusions

We have defined next generation Java-based smart card as a platform enabling the efficient development, deployment and management of the on-card parts of networked applications. For that purpose, the platform has to provide a powerful and efficient execution environment as well as a flexible and operable management context within constrained environments. Such a platform also has to offer an harmonized interface to operate the services in a distributed and dynamic fashion required by m/e-commerce and m/e-services operations. For that purpose, the platform has to provide a standard way of communication within different types of networks as well as a rich and evolving framework for different application models.

Dealing only with the platform is not sufficient. Other technologies also have to be developed and should be added to this research agenda. Some of them are quite independent of the platform capabilities. But some are tightly linked to the platform and then would benefit from being intimately developed in synergy with the platform. Most evident ones are: secure Java technologies, delegation of operations to card, and card management. Their efficiency and reliability have to be precisely measured whether or not they are developed in convergence with the card platform, or with the possibility to influence some features of the card platform.

6. References

- [1] D. Deville, A. Galland, G. Grimaud, and S. Jean. “Smart Card operating systems: Past, Present, and Future.” To appear in *The fifth NORDU/USENIX Conference, Vasteras, Sweden, February 2003*.
- [2] Z. Chen. *Java Card Technology for Smart Cards: Architecture and Programmer’s Guide*. The Java Series, Addison Wesley, 2000.
- [3] D. Davis and D. Balaban. “Wake Up And Smell The Java!” *Card Technology Magazine, February 2002*.
- [4] Sun Microsystems, Inc. “Java Card 2.2 Platform Specifications.” *February 2002*.
<http://java.sun.com/products/javacard/javacard22.html>
- [5] P. George. “A Java Technology-Based, End-to-End Solution for Corporate Products for the Java Card Platform.” *Java One Conference, June 2001*.
<http://servlet.java.sun.com/javaone/conf/sessions/2013/ogle-sf2001.jsp>
- [6] J. H. Saltzer, David P. Reed, and David D. Clark. “End-To-End Arguments in System Design.” *ACM Transactions on Computer Systems, Vol. 2, 4, pp. 277-288, November 1984*.
<http://citeseer.nj.nec.com/saltzer84endoend.htm>
- [7] S. D. Halloway. *Component Development for the Java Platform*. Addison Wesley, December 2001.