

A Generic Payment System Architecture

Bart De Win

Jan Van den Bergh

Frank Matthijs

Wouter Joosen

Report CW 317, June 2001



Katholieke Universiteit Leuven
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Generic Payment System Architecture

Bart De Win
Jan Van den Bergh
Frank Matthijs
Wouter Joosen

Report CW 317, June 2001

Department of Computer Science, K.U.Leuven

Abstract

Despite of the major boost of e-commerce, developing and maintaining e-commerce applications still remains far from straightforward. The lack of standardization in this area results in a variety of protocols and systems, which developers must take into account. In particular, supporting different payment systems is one of the problems. Different types of payment systems exist, each with specific requirements.

This paper presents the requirements for and the design of an architecture that offers applications a way to work with payment systems. It is designed in such way that payment systems can be easily integrated and interchanged. It also offers a generic way to negotiate about the choice of payment protocols and instruments based on different parameters. The architecture has been implemented and used successfully in combination with several modern payment systems.

Keywords : payment systems, generic architecture, software engineering

A GENERIC PAYMENT SYSTEM ARCHITECTURE

Keywords: payment systems, generic architecture, software engineering

Abstract: Despite of the major boost of e-commerce, developing and maintaining e-commerce applications still remains far from straightforward. The lack of standardization in this area results in a variety of protocols and systems, which developers must take into account. In particular, supporting different payment systems is one of the problems. Different types of payment systems exist, each with specific requirements.

This paper presents the requirements for and the design of an architecture that offers applications a way to work with payment systems. It is designed in such way that payment systems can be easily integrated and interchanged. It also offers a generic way to negotiate about the choice of payment protocols and instruments based on different parameters. The architecture has been implemented and used successfully in combination with several modern payment systems.

1. INTRODUCTION

The amount of shops that employs electronic commerce to extend their consuming market is growing really fast. Both customers and merchants realize the advantages of a worldwide economy. Many governments are funding the development of electronic commerce applications to improve the global prosperity of their countries. Unfortunately, developing and maintaining such systems is not straightforward. The main reason is the intrinsic open character of the Internet. Since a lot of different technologies exist, small companies often don't see the wood for the trees.

One important problem concerns the generic support of payment technology. Many payment systems have their own specific requirements, which make it hard to design an application that it is able to work with all of them. In our opinion there is still lack of a simple generic payment

architecture that can be easily plugged in into an application. This architecture should be open enough to be able to include existing and future payment systems. Moreover, it should be responsible for the payment related issues and as such also choose between different payment implementations based on specific criteria.

This paper presents an architecture that tackles these problems. We will first overview the characteristics of modern electronic payment systems. Based on those, we discuss the requirements and the design of a generic payment architecture. Then, we briefly discuss some related work. And finally we end this paper with a conclusion.

2. REQUIREMENTS OF A PAYMENT ARCHITECTURE

In order to fully understand the scope and the characteristics of electronic payment systems, we first present a brief overview of existing payment technology. Afterwards, we describe the functional and non-functional requirements of a generic payment architecture.

2.1 Payment technology

Electronic payment has become a hot topic since the exponential growth of the Internet. Many different payment systems have been proposed. Existing literature ([13]) distinguishes between the following types:

- **Card based:** these systems are based on real life payment cards, like credit cards (e.g. Visa and Mastercard) and debit cards. For this case, the primary function of the payment system is the transfer of the card data and the actual clearance. Nowadays, most of the electronic payment systems are based on credit cards. Examples of such systems are SET [17], iKP [1], EasyPayment [8], etc.
- **Electronic cash:** similar to real life coins, these systems offer several important advantages [13]. Unfortunately, the implementation of such protocols is often difficult. For instance, since duplication of electronic coins is very easy (copying of bytes), these systems are frequently subject to crime. Ecash [7] and CAFE [2] are famous electronic cash systems.
- **Electronic check:** checks also have their electronic counterpart (e.g. FSTC electronic check [9]). However, they are quite expensive to process and they don't offer real advantages over the other payment systems. Hence, they are not very popular.
- **Micropayments:** similar to electronic cash, micropayment systems are based on the physical exchange of value. However, in this case they are designed for very small transactions. Such systems can be used for instance in a "pay per view" browsing system. Since the amounts are very little, fraud would not bring in much. Therefore, security is often neglected and the protocols are more lightweight. Millicent [10] and iKP [1] are two examples.

- **Other:** everything that has a certain value, for instance frequent flyer miles, can be used to pay. In this case the payment system must be able to exchange electronic data and will as such resemble like electronic cash systems.

2.2 Requirements

The most important functional requirement for a generic payment system architecture is the ability to include different payment systems. Any existing and future system should easily fit in the architecture. For this purpose, a *generic payment system interface* must be provided, which can then be implemented for a specific payment system. Building this interface is far from straightforward. On the one hand, it must be powerful enough in order to allow easy integration of different systems. On the other hand, it must enable easy, yet generic deployment. Furthermore, since different payment mechanisms often support the same payment instruments, it is a desirable to have weak coupling between the former and the latter. For example SET and iKP both offer Visa card support. Including Visa capabilities in both mechanisms separately would lead to the duplication of code. Separation of the instrument and mechanism abstractions in the interface certainly enables this weak coupling and is hence a preferable approach.

A payment transaction at least requires interaction between two parties. Besides, a bank and possibly other intermediaries are often introduced as (trusted) third parties. Either way, the communication between these parties has different constraints. To enable such communication, the architecture must either offer a very general mechanism¹ or be *communication system independent*. In the latter case a particular implementation can itself choose to use the most appropriate communication system.

Payment systems often rely on legacy systems to effect the payment transaction. Therefore, the architecture must offer a way to contact these *legacy systems*. This can be achieved in two ways. A payment system implementation might contact the legacy system directly. While this is probably more efficient, it introduces complex synchronization into the implementation. For the sake of easiness and generality, special proxies should be introduced that take care of this extra communication. An approach similar to the latter is described in [12].

Since different payment systems will be supported, a convenient way of choosing the desired one should be available. While this heavily depends on the users' preferences, some work in this process can be performed automatically. Hence, introducing a *negotiation phase* for this purpose would help the user considerably. Naturally, enforcing one standard negotiation process is not desirable because research will always come up with better solutions to this problem. Providing a generic interface ensures interchangeability.

¹ Sempër (see related work) uses such a generic communication mechanism.

Besides these requirements, several non-functional issues are important². Since many payments systems exist and since many people will use this technology in the near future, *scalability* of the architecture is a primary concern. *Security* is another very important issue, which will be discussed at the end of this paper. And of course the architecture should be open enough to allow easy integration of many different payment technologies. Therefore, it is wise to aim rather at a *framework* than at a specific solution.

² Most of the non-functional requirements are also important when designing a particular payment system. In fact, they should be considered for every distributed system.

