Chapter IX
Standards for Business Component Markets:
An Analysis from Three Theoretical Perspectives

Heiko Hahn
University of the Federal Armed Forces, Germany

Klaus Turowski
University of Augsburg, Germany

ABSTRACT

The idea of component-based software systems and markets for the exchange of components dates back to the late 1960s. However, so far no large-scale components markets can be found. The purpose of this chapter is to present a more in-depth analysis of the conditions that have to be met for the successful realization of this idea. Three perspectives are presented: first, a system-theoretic perspective; second, an economic perspective; and third, a knowledge-codification perspective of standardization. As we want to argue, the problem should be considered as an empirical question that depends to a large extent on the future technological development and its outcome, like specification techniques, software verification standards, and the performance and maturity of existing systems.

INTRODUCTION

Products often consist of different parts; that is, they are rather product systems consisting of different components[1] than they are monolithic products. The parts are to a varying degree independent and interchangeable and might be produced by one integrated firm or by different firms. In the latter case, the components can be purchased prefabricated on the market or from a supplier with whom the customer has an intensive business relationship and who produces the parts according to the special requirements of the customer.

In the case of enterprise resource planning (ERP) software like SAP R/3, the dominant design is monolithic; that is, all customers purchase and install basically the same software package, which is than adapted by the use of parametric configuration to the different needs of the firms. As we will discuss in the next section, the choice between different architectonic principles (i.e., modular or monolithic) can be motivated by certain factors, and given these factors, some architectonic designs might be more appropriate than another for a given situation.

The advantages of a modular design are well known, and the idea to use markets for the exchange of software components has been discussed since the late 1960s (McIlroy, 1976). However, despite the advantage of component-based software systems and the success of component-based systems in other industries (for example, platform strategies in the automotive industry), large-scale markets for software components can hardly be found (Dietzsch & Esswein, 2001).

In this chapter we will address the problem of establishing markets for business components from three distinct perspectives. The first is a system-theoretic perspective discussing the principle of modular design. The second perspective is an economic analysis discussing the preconditions for market-based coordination. The last perspective focuses on standardization, especially as a process of knowledge codification. It takes up certain aspects of standardization that are also discussed in the two other sections to discuss the underlying problems more in depth.

SYSTEM-THEORETIC PERSPECTIVE

Modular Design and the Principle of Information Hiding

The concept of modular design is central to system theory. Simon (1962, p. 476) characterizes the term *nearly decomposable system*, that is, a modular system, as follows:

(1) in a nearly decomposable system the short run behaviour of each of the component subsystems is approximately independent of the short-run behaviour of other components; (2) in the long run the behaviour of any one of the components depends in only an aggregate way on the behaviour of the other components.

For technical systems, the influence that different components have on each other is limited and mediated by interfaces that allow abstracting from the lower level implementation. The interface is the device that mediates the interaction between the different components and prevents a direct communication. Parnas (1972) introduced the principle of information hiding, which emphasizes the importance of indirect communication and a black-box approach. According to the principle of information hiding, the design decision should be led by the goal to make the different parts of a software program as independently interchangeable as possible so that the system can accommodate changes easily. Information hiding provides a criterion for the decoupling of the different systems: The interface level that connects the different parts of which the system consists should be invariant against changes at the implementation level. To