Chapter 2.16
Architecture, Specification, and Design of Service-Oriented Systems

Jaroslav Král
Charles University, Czech Republic

Michal Žemlička
Charles University, Czech Republic

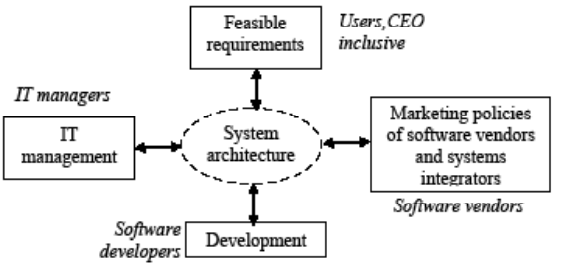
ABSTRACT

Service-oriented software systems (SOSS) are becoming the leading paradigm of software engineering. The crucial elements of the requirements specification of SOSSs are discussed as well as the relation between the requirements specification and the architecture of SOSS. It is preferable to understand service orientation not to be limited to Web services and Internet only. It is shown that there are several variants of SOSS having different application domains, different user properties, different development processes, and different software engineering properties. The conditions implying advantageous user properties of SOSS are presented. The conditions are user-oriented interfaces of services, the application of peer-to-peer philosophy, and the combination of different technologies of communication between services (seemingly the obsolete ones inclusive), and autonomy of the services. These conditions imply excellent software engineering properties of SOSSs as well. Service orientation promises to open the way to the software as a proper engineering product.

INTRODUCTION

Service orientation (SO) is becoming the central topic of software engineering. There is an explosive growth in the number of conferences, products, and articles discussing and using the principles of SO and service-oriented architectures (SOA). Service-oriented software systems (SOSS) are of different types depending on the character of the functions the system provides, the system environment (for example, e-commerce or a decentralized international enterprise), and the way the system is developed. The common property of SOSS is that their components behave like the services in real life mass service systems.

Figure 1. Central role of system architecture



The SOSS must then be virtual peer-to-peer (p2p) networks of autonomous components (services). The services can have various properties; they need not be Web services in the sense of W3C (2002) and need not therefore use standard communication protocols, compare Barry and Associates (2003) and Datz (2004).

We shall show that the software engineering properties as well as the user-oriented properties of any SOSS strongly depend on the properties of the service interfaces and that user interfaces of the system should be implemented as specific services (peers of the network) as well. All these issues are related to the architecture of the system. We will discuss how the properties of the architecture influence the set of feasible functions, development (especially the requirements specifications), feasible development techniques (for example, agile ones), standards, politics of IT management, and marketing strategies of software vendors and/or system integrators (Figure 1). The feasible functions of SOSSs include the functions important for user top-management.

Feasible functions of any large system depend on its architecture. The decision as to what architecture is to be used must therefore be formulated in early stages of the system life cycle. On the other hand, the structure, techniques, and content of requirements specifications are influenced by the properties of the system architecture and the details of its implementation. We shall show that SOSS should use a combination of various techniques developed during the software history (for example, message passing, object orientation, common databases, and, sometimes, batch-oriented systems). All these issues should be addressed in the specifications of SOSSs. SO is a paradigm new for many software people. It implies some problems with the application of SO.

PEER-TO-PEER INFORMATION SYSTEMS (P2PIS)

Large information systems must often be developed as a network of loosely coupled autonomous components—services (possibly information systems) integrated using peer-to-peer principle (further P2PIS). The change of the architecture should be accompanied by changes in requirements specification that should reflect the service-oriented structure of the system.

The specification of P2PIS starts from the specification of system user interface (portal) and from the specifications of the services. The specification of services starts from the definition of their interfaces. It can be accompanied by specification of the services of the infrastructure (message formats, communication protocols, middleware services, in general). Services in P2PIS can be newly developed applications, encapsulated legacy systems, or third party products. P2PIS enables new types of requirements (for example, the requirement that a P2PIS should support decentralized and flexible organization of a global enterprise, see Král & Žemlička, 2003) and makes achievable software engineering properties like reusability, flexibility, openness, maintainability, the use of legacy systems and third party products, or the reduction of development costs and duration. Experience shows that such systems can be extremely stable (Král, 1995).

There are two main variants of P2PIS. The first one is used in e-commerce where the service starting a communication must first look for communication partners. The partners must offer their interfaces (typically specified by WSDL).