Chapter 5
A Look on Engineering Non-Functional Properties in Service Oriented Architectures

Nicolò Perino
University of Lugano, Switzerland

Marco Massarelli
Università degli Studi di Milano-Bicocca, Italy

Daniele Cammareri
Università degli Studi di Milano-Bicocca, Italy

Claudia Raibulet
Università degli Studi di Milano-Bicocca, Italy

Francesca Arcelli
Università degli Studi di Milano-Bicocca, Italy

ABSTRACT

Service-oriented architectures (SOA) aim to define a common approach for both the providers and consumers of services by introducing the mechanisms necessary to publish services, to search for services, and to request the execution of services. The proliferation of the providers of services offering identical or similar services has led customers to choose services based on information behind their functionality. This type of information may regard issues such as quality, performance, response time, security, availability, and reliability. In the scientific literature, such information is known under the name of non-functional properties or non-functional requirements.

Usually, non-functional requirements are not explicitly addressed during the analysis and design of Information Systems, even if they are fundamental for the architectural or technological decisions. Hence, they are neither traceable nor documented. However, in recent years, the engineering of non-functional properties together with the functional ones is gaining more and more the attention of software engineers. There are various proposals for the specification of non-functional features which range from informal textual specification to extensions of the Unified Modelling Language.

DOI: 10.4018/978-1-60566-794-2.ch005
The aim of this chapter is to provide an overview on non-functional issues in service oriented architectures. First, it introduces the non-functional requirements which should be addressed in service-oriented architectures and the challenging issues they raise (i.e., specification, conflicts, run-time management). Second, it presents the available approaches related to the engineering of non-functional issues. Third, it discusses the possible future trends regarding this topic.

INTRODUCTION

The development of information systems focuses on the functionalities systems should provide to the users. Craig Larman sustains that one of the most common and successful iterative development approaches for software applications is *use-case driven* (Larman, 2004). Essentially, use case diagrams model the functional (behavioural) requirements of a system. This means that the analysis, design, implementation and testing phases are all driven by use-cases. In the same book, Larman indicates as non-functional “everything else” related to a system and not indicating a functional requirement. He mentions that non-functional properties are usually related to the quality of a software system, but this book lacks in providing detailed information on addressing non-functional requirements during the development life-cycle of an information system.

Ian Sommerville defines non-functional properties as “constraints on the services or functions offered by a system” (Sommerville, 2004). Non-functional requirements may be related to the reliability, reparability, security, usability of a system. Sommerville discusses the importance of being able to identify and consider them both during the development and the execution phases. This is because they address critical aspects which determine the usability of a system. These critical aspects may be more important than individual functional requirements. For example, users may find alternative ways to use a system function which does not respond exactly to their needs. But if a real-time control system fails to meet the performance requirements, the control component is not reliable and thus it cannot be used (Sommerville, 2004). Furthermore, Sommerville asserts that non-functional properties may be more complex than the functional ones because they regard the entire system rather than a single component, service, or function. They may be related not only to the software product, but they may influence the development process of a system (i.e., the specification of the quality standards that should be used in a process). From this point of view, non-functional issues may be considered of two types: *product-oriented* and *service-oriented* (Franch, 1998; Mylopoulos, Chung, & Nixon, 1992; Sommerville, 2004).

Taylor, Medvidović, and Dashofy define non-functional properties as “constraints on the manner in which the system implements and delivers its functionalities” (Taylor, Medvidović, & Dashofy, 2009). They identify at least three main types of non-functional properties: structural, behavioural, and interactional. Furthermore, they focus on the multi-dimensional characteristic of non-functional properties, as well as on the challenge to measure them due to their more qualitative rather than quantitative nature. It is also stressed that the software engineering methodology claims to perform the correct steps from the beginning of the software development: hence, non-functional properties should be considered from the first analysis of the requirements of a system in order to save later development and maintenance costs. Taylor, Medvidović, and Dashofy discuss on the tight link between the software architecture and the non-functional properties of a system; through software architectures engineers may address explicitly and trace further non-functional properties such as efficiency, scalability, availability, security, or fault-tolerance. Usually, the systems which are not based on a well-defined architectural model, and which address non-functional properties later