Chapter 2

Developing Non-Functional Requirements for a Service-Oriented Application Platform: A Goal and Scenario-Oriented Approach

Daniel Gross
University of Toronto, Canada

Eric Yu
University of Toronto, Canada

Xiping Song
Siemens Corporate Research, USA

ABSTRACT

The challenges in developing non-functional requirements (NFRs) for an application platform go much beyond those for a single application system. To derive platform NFRs from NFR specifications of different domain applications, requirements analysts must deal with much variation of domain specific NFRs, with different deployment configurations and load conditions, with different NFR related trade-offs, as well as with different terminology and metric definitions. This chapter presents a platform NFR development method that supports dealing with the aforementioned challenges. The presented method offers a goal- and scenario-oriented modeling and analysis technique that supports dealing with qualitative and quantitative NFRs during platform NFR development in an integrated way. The platform NFR development method was used to develop NFRs of a service-oriented application platform for three different application domains in an industrial setting.

DOI: 10.4018/978-1-60566-794-2.ch002
INTRODUCTION

Large software development organizations with software product offerings across multiple markets and industries usually have core competences that underpin diverse product offerings. Identifying and formalizing those core competences provides development organizations with opportunities to create common application platforms to support their products development efforts across markets and industries.

Shared application platforms significantly increase reuse of software assets, reduce time to market and cost, and improve software quality. Service orientation offers additional benefits including enterprise-wide standardized reusable software assets, increased interoperability, and ease of extension, evolution and adoption of new services-based functionalities and features.

Specifying the requirements, and in particular the non-functional requirements (NFRs) for an application platforms is however challenging. Application platforms aim to support a large number of domain specific applications in meeting functional and non-functional requirements. While common functionality can be identified from shared core competencies across different domain applications, the non-functional requirements across domains can still vary greatly, which makes it hard to pin down which NFRs a common platform should support. In industrial settings, the following main challenges have been identified (Song et al., 2009):

**Varying domain-specific needs**: Different application domains give rise to different NFRs. For example, a solution system that supports automation in manufacturing, which often requires meeting tight hard real time constraints, and a solution system that provides building security in factories, where real time requirements are much less demanding, have quite distinct NFRs, even if both systems involve much of the same control functionalities.

**Varying deployment configurations and load conditions**: Solution systems can be deployed and operated in different configurations and under different load conditions. For example, a common platform may need to support an application that is deployed and operated as an embedded standalone system responding to several hundred events per minute. The same platform may however also need to support an application deployed as an integrated multi-site system of dedicated servers responding to tens of thousands events per second. What non-functional platform requirements should be specified so that once implemented the platform can be deployed on an embedded standalone system or on dedicated servers, and respond well for both types of loading conditions?

**Terminology and metrics mismatch**: During the development of NFRs for the platform, requirements analysts must deal with a wide range of concepts, terminology and metrics used in different application domains. For example, in one industrial domain, a performance requirement could be specified in events per seconds, while in another in alarms per minute. Platform developers must translate such differences in terminology and metrics into common platform terminology and metrics before compatible platform requirements can be specified.

**Dealing with NFR trade-offs**: Developing an application platform in general and adopting service orientation in particular requires the implementation of specific design principles, such as modularity, loose coupling, service statelessness, service autonomy, service contracting, etc. (Erl, 2007), which help achieve non-functional benefits, such as reusability, interoperability, consistency and extensibility, associated with application platforms and service orientation. However implementing such design principles comes with a price and requires developers to make trade-offs with other non-functional requirements such as performance and increased upfront development costs. Platform developers
Related Content

Toward an Integrated Conceptualization of the Service and Service System Concepts: A Systems Approach
www.igi-global.com/article/toward-integrated-conceptualization-service-service/2527?camid=4v1a

The Economics of Cloud Computing
www.igi-global.com/chapter/economics-cloud-computing/70148?camid=4v1a

Survey of the State-of-the-Art of Cloud Computing
www.igi-global.com/article/survey-state-art-cloud-computing/60407?camid=4v1a

Security Concepts for Cloud Computing
www.igi-global.com/chapter/security-concepts-cloud-computing/74227?camid=4v1a