On Feature Interactions Among Web Services

Michael Weiss, Carleton University, Canada
Babak Esfandiari, Carleton University, Canada

ABSTRACT

Web Services promise to allow businesses to adapt rapidly to changes in the business environment and to the needs of customers. However, the rapid introduction of new services paired with the dynamicity of the business environment also leads to undesirable interactions that negatively impact service quality and user satisfaction. In this paper, we propose an approach for modeling such undesirable interactions as feature interactions. As each functional feature ultimately is motivated by non-functional requirements, we make an explicit distinction between functional and non-functional features. We then describe our approach for detecting and resolving feature interactions among Web Services. The approach is based on goal-oriented analysis and scenario modeling. It allows us to reason about feature interactions in terms of goal conflicts and feature deployment. Three case studies illustrate the approach. The paper concludes with a discussion of our findings and an outlook on future research.

Keywords: feature interaction; goal-oriented analysis; non-functional features; Web Services

INTRODUCTION

Web Services promise to allow businesses to adapt rapidly to changes in the business environment and to the needs of customers. However, the rapid introduction of new services paired with the dynamicity of the business environment also leads to undesirable interactions that negatively impact service quality and user satisfaction. In this paper, we propose an approach — feature interactions — for modeling such undesirable interactions.
conflict; for example, services triggered by a busy extension);  

- Competition for resources (services compete with each other for limited resources that need to be partitioned among the services);  
- Changing assumptions on services (services make implicit assumptions about their operation, which can become invalid when new services are added); and  
- Design evolution (services need to be added to meet new customer needs, and the system will need to interoperate with other vendors’ systems).

A classical feature interaction is the interaction between Call Waiting and Call Forwarding on Busy. Both features trigger when the receiver of a call is busy, but only one of them should become active. This type of problem usually is resolved by introducing priorities. The most prominent implementation of this approach is the pipe-and-filter model (Utas, 2001), in which features are connected in a chain of filters in the order in which they get to process events.

The interaction between Outgoing Call Screening and Call Forwarding on No Answer is slightly more complex. Assume Alice is on Bob’s outgoing call screening list (Alice could be the girlfriend of Bob’s teenaged son Mark, and Bob does not want him to call her). But Mark quickly learns that he only needs to call his friend Joe, who temporarily forwards incoming calls to Alice. The solution to this type of problem involves confirming with the originating party (Bob) if Joe’s forwarding the call to Alice is acceptable.

However, the feature interaction problem is not limited to the telecommunications domain. The phenomenon of undesirable interactions among components of a system can occur in any software system that is subject to changes. This is certainly the case for service-oriented architectures. First, we can observe that interaction is at the very basis of the Web Services concept. Web Services need to interact, and useful Web Services will emerge from the interaction of many highly specialized services. Second, as the number of Web Services increases, interactions will become more complex. Many of these interactions will be desirable, but other interactions may be unexpected and undesirable, and we need to prevent their consequences from occurring. As noted by Ryman (2003), many such interactions are related to security and privacy.

**Web Services and Web Service Composition**

Much research has focused on low-level concerns, such as how to publish, discover, and invoke individual Web Services as well as the security of Web Services. Other work has looked at dynamic Web Service composition (Constantinescu et al., 2002); that is, how higher-level services can be composed dynamically from lower-level services. Service composition raises a number of difficult challenges, such as service description, selection, and orchestration.

At each of these stages (description, selection, and orchestration), we may experience undesirable interactions that prevent the proper performance of the service. However, there has been little research on managing such interactions at the level of the service logic. Most existing work is limited to managing the mechanics of the interaction (e.g., enforcing a legal sequence of messages exchanged between the parties involved).

When composing Web Services, the functionalities provided by the component services must be considered. We also need to ensure that data and message types, sequence logic, and so forth are compatible. However, as stated in O’Sullivan et al. (2002), service composition amounts to much more than functional composition. Consideration also must be given to non-functional requirements, such as privacy and interoperability. For example, when composing a personalized Web Service, we also must consider utility services, such as identity management and user profiling. But maintaining and sharing sensitive user information in a utility service raises privacy concerns.
Context Dissemination in Peer-to-Peer Networks
www.igi-global.com/chapter/context-dissemination-peer-peer-networks/43648?camid=4v1a

Virtualization Evolution: From IT Infrastructure Abstraction of Cloud Computing to Virtualization of Network Functions
www.igi-global.com/chapter/virtualization-evolution/217913?camid=4v1a