Chapter XI

Program Transformations for Web Application Restructuring

Filippo Ricca
ITC-irst, Italy

Paolo Tonella
ITC-irst, Italy

Abstract

This chapter aims at providing a presentation of the principles and techniques involved in the (semi-)automatic transformation of Web applications, in several different restructuring contexts. The necessary background knowledge is provided to the reader in the sections about the syntax of the multiple languages involved in Web application development and about the role of restructuring in a highly dynamic and rapidly evolving development environment. Then, specific examples of Web restructuring are described in detail. In the presentation of the transformations required for restructuring, as well as in the description of the grammar for the involved languages, TXL (Cordy, Dean, Malton & Schneider, 2002) and its programming language is adopted as a unifying element. The chapter is organized into the following sections: in the section following the Introduction, the problems associated with the analysis of the multiple languages used with Web applications are discussed. Then, the process of Web application restructuring is considered. Three examples of Web restructuring are described in more detail in the next three sections (design restructuring, migration of a static Web site to a dynamic Web application, consistency among monolingual portions of a multilingual Web site). Related works and concluding remarks are at the end of the chapter.

Copyright © 2005, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
Introduction

All known problems that characterize the evolution of software systems are exacerbated in the case of Web applications, because their typical life cycle is much shorter and their development iterations are executed faster and more frequently. In fact, Web applications must accommodate continuing changing requirements, since the user’s needs evolve rapidly over time. Moreover, the infrastructure technology is itself subject to continuous updates. New technologies emerge at a high rate, replacing the existing ones. A strong pressure for change comes also from the market, where it is fundamental to be up to date with the new services and paradigms.

The consequences of a high evolution rate are often very negative for the internal quality of the Web applications. The initial architecture is subject to a drift, so that the original assignment of responsibilities to components is violated and the dependencies among components tend to resemble a fully connected graph. Code fragments may be duplicated, thus making the related functionality delocalized, and consequently difficult to maintain and reuse. While in the beginning such negative effects may be invisible to the end user, after some time the application becomes unmanageable. New requirements are difficult to accommodate, nobody has a sufficient understanding of the architecture to allow for major changes, and defects tend to be inserted with the modifications, due to unexpected ripple effects.

The evolution scenario described above is more complicated with Web applications than with traditional software, because of the technologies underlying their development. In fact, Web applications involve a basic client-server architecture, with the browser as the client and the Web server on the other side, upon which higher level functionalities are built. Since multiple users can connect concurrently, user sessions must be handled over time. However, this is not supported by the HTTP protocol in use, which is stateless and connectionless. When transactions are executed inside user sessions, the problem is even harder. Security is another central concern. The support technologies used to solve these problems are based on multiple standards and programming languages. Consequently, the resulting Web application is composed of a heterogeneous set of components written in different languages and based on different standards. Portability across environments and browsers is another factor that complicates the internal organization of the Web applications.

When the evolution of a Web application makes its source code difficult to maintain, the typical action that is undertaken consists of rewriting the entire application from scratch in the new setting. In this chapter, a different approach is considered. It is based on the notion of continuous restructuring, and it assumes that preventive interventions can be made to avoid the progressive degradation that accompanies software evolution. Restructuring is an expensive activity for which a budget is rarely available in a highly competitive market such as that of Web applications. However, the possibility to (partially) automate restructuring can make it cost-effective. Its adoption would involve periodic interventions of preventive maintenance that precede the actual implementation of the required changes.