Chapter VII
Component-Based Deployment for Web Applications: Experiences with Duct Tape and Glue

Kevin Gary
Arizona State University, USA

Harry Koehnemann
Arizona State University, USA

ABSTRACT

The software engineering community touts component-based software engineering as a solution for many of its woes including reducing cycle time, reducing costs, increasing productivity, allowing easier integration to name just a few. Indeed, many Web-based systems are now built with open-source and vendor provided component technologies. While these advances have led to improvements in the development process, they have also led to a great deal of pressure on downstream processes as these systems must be deployed, tuned, and supported. The complexities in deploying and supporting component-based software for distributed and Web-based applications are not understood in the academic or professional communities. This chapter stresses the need for addressing this problem by presenting component-based software for Web applications from a deployment perspective, characterizing the issues through real-world experiences with highly component-based applications, and presents strategies and directions for the community to pursue.

INTRODUCTION AND MOTIVATION

Discussion of the engineering of modern Web-based systems inevitably focuses on many of today’s popular buzzwords such as services-oriented architecture (SOA), n-tier architecture patterns, lightweight and Agile methodologies, AJAX, and so on. These discussions are deserved, and show the rapid pace at which the understanding of engineering such applications has evolved. These technologies are realized through a proliferation of methods and frameworks giving architects literally dozens of tools to construct applications to deploy on the Web.
However, this chapter considers a topic that has not received fair attention or debate. That topic is the downstream issues and costs involved in deploying (and redeploying) component-based software for distributed and Web-based applications. The main body of attention in Web-based software engineering is from the point of view of the architects and developers, with ample debate on the merits of methods, patterns, and frameworks. While development teams tend to fall in love with these technologies, adequate consideration is rarely given to the ramifications of supporting them once the application is “thrown over the wall” to the application support team.

Component-based software engineering (CBSE) is a relatively new driver in Web-based software engineering and has enjoyed rapid acceptance in both industry and the research community. Spurred by the advent of distributed computing via the Internet, academic institutions and professional communities alike now teach, train, and mentor in the well-founded principles of object-oriented design and programming, and extend this model to coarser-grained component-based computing.

Many (if not most) Web developers today espouse the benefits of component-based computing including reusability, flexibility, loose coupling, modularity, separation of concerns, and so forth, without necessarily understanding how component-based computing delivers on these promises. We by no means imply component-based Web applications are “wrong;” but we do feel there are significant gaps and assumptions made about the “goodness” of these principles that may arise, specifically in the areas of deployment and post-deployment support.

CBSE, while demonstrating value-add for software development processes, is not yet proven across the full set of business processes supporting enterprise class software. CBSE may in fact imply greater complexity, and therefore higher costs for supporting deployment and release processes for Web-based systems.

More to the point, do we hear our application support personnel espousing the benefits of flexible and collaborating components as frequently as we hear from developers? Rhetorically the answer is no as the application support team’s members are too busy writing scripts, trying to understand deployment descriptors, figuring out what component is in what package, discovering undocumented hidden dependencies in components by tracing through stack traces, and finding a way to get the entire system talking in a particular Web application server production environment.

The application support team, which includes assemblers, deployers, and operators, cannot remain the forgotten partner for long. Configuration management, release management, testing, deployment, and other business functions will continue to raise issues related to CBSE:

- **Configuration management (CM):** The componentization of the software system means more possible permutations of components (configurations) to manage and support. For systems of any size there can be an explosion in permutations as development decomposes systems into components.

- **Release management (RM):** Closely related to CM, RM is also concerned with coordinating all business processes related to getting software out the door, as well as how to provide updates, patches and migration to new releases. Given the potentially dynamic and “plug-n-play” nature of CBSE, this becomes a high-risk area to organizations supporting Web-based applications.

- **Software test:** The proliferation of components coupled with dynamic composition creates a tremendous burden on test groups to validate that the software system will remain robust in the face of various component configurations. When can the group be comfortable that a sufficient coverage of