

Supporting Reconfigurable Object Distribution for Customizable Web Applications

Po-Hao Chang and Gul Agha

University of Illinois at Urbana-Champaign, Urbana IL 61801, USA
{pchang2, agha}@cs.uiuc.edu

1 Introduction

Web applications are tightly coupled with the platforms that a particular service provider intends to support and the execution scenario envisioned at the design time. Consequently, the resulting applications do not adapt well to all clients and runtime execution contexts. The goal of our research is to develop methods and software to support *reconfigurable distributed applications* which can be customized to specific requirements. Thinking in terms of *software product line engineering* [2], we need a product line for a given Web application, each instance of which is for a specific execution platform and context. To achieve such a product line, we have to satisfy two requirements: universal accessibility and context-dependent component distribution.

2 Virtualization

We address both problems by using *virtualization* on execution platforms to provide a uniform programming environment for objects. Specifically, we have developed a virtual programming environment which hides the incompatibilities in different platforms and models. The environment provides a high-level abstraction: programmers no longer deal with pages, HTTP requests, cookies, sessions, which are entities in specific models and platforms; instead, they focus on the building objects and their interaction, and specifying the application logic. A Web application in the virtual environment represents a product line characterized by its application logic, not a specific implementation which can be deployed on a specific platform.

3 Separation of Concerns

Following the principle of *separation of concerns*, our virtual environment is not only platform independent but also location agnostic: there is no notions about object location and object movement. Nonetheless, such specifications are required to build an executable Web application. We designed a specification system including two parts: a specification language allowing developers write object distribution schemes, and tools to help the framework implementation

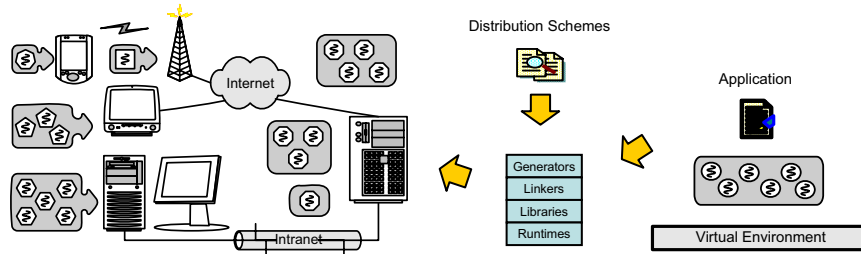


Fig. 1. System Architecture

execute the schemes. Combining an application with a specification, a Web application customized to a specific execution context can be defined. Inspired by the success of Web Style Sheets, our specification system is also based on *transformation*: converting a composition of objects into another composition of annotated objects. The annotation attributes are given by a specific execution framework. For example, in a typical Web framework, the location of an object may be a *client* or a *server*. Specifying object allocation can be achieved by annotating each object's with the desired attribute value. Sophisticated loading policies can be specified in a similar way. We have developed an algorithm to perform the transformation and proved its correctness.

4 Generative Approach

One approach to support virtualization over heterogeneous platforms is to define and build a middleware. This approach has the disadvantage of requiring the deployment of new software systems and protocols, which is impractical for the sheer scale of the Internet and the diversity of client platforms. Instead, we adopt the paradigm of generative programming [1] to realize application execution: before deployment, the composing objects of a Web application undergo a generative process to obey the annotations generated by the specification system.

To support these generated objects, a light-weight execution framework is required, whose primary role is to facilitate object management across HTTP channels. We implemented a Web execution framework composed of libraries and runtime systems. Our experience suggests that it is feasible to facilitate reconfigurable object distribution using a specification which controls object allocation and regulates loading patterns.

References

1. Krzysztof Czarnecki and Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.
2. Klaus Pohl, Gnter Bckle, and Frank van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.