Achieving the Promise of Reuse with Agent Components

Martin L. Griss[1] and Robert R. Kessler[2]

[1] Computer Science Department 349BE, University of California, Santa Cruz
1156 High Street, Santa Cruz, CA 95064
griss@soe.ucsc.edu
http://www.soe.ucsc.edu/~griss
[2] University of Utah, School of Computing, 50 S. Central Campus Dr. #3190,
Salt Lake City, UT USA 84112
kessler@cs.utah.edu

Abstract. Using software agents as next generation flexible components and applying reuse technologies to rapidly construct agents and agent systems have great promise to improve application and system construction. Whether built on conventional distributed computing and application management platforms, on a specialized agent platform, on web service technology or within a P2P infrastructure, agents are a good match for independent development, for scalable and robust systems and dynamic evolution of features, and for autonomic self-managing systems. In this paper we describe the vision and progress we have made towards developing a robust infrastructure, methods, and tools for this goal.

1 Introduction

For some time now, component-based software engineering (CBSE) has promised, and indeed delivered, significant improvements in software development [1]. Greater reuse, improved agility and quality are accessible benefits. CBSE produces a set of reusable assets (usually components) that can be combined to obtain a high-level of reuse while developing members of a product-line or application family. Typically, one first performs domain analysis to understand and model commonality and variability in the domains underlying the product-line, and then a layered modular architecture is defined, specifying layers and core components, key subsystems and mechanisms. Finally, high-level specifications and interfaces are defined for pluggable or generated components. Implementation begins with the development or selection of a framework that implements one or more layers of the architecture. Delivering the reuse potential as a well-designed domain-specific kit carefully allocates variability to a combination of components, frameworks, problem-oriented languages, generators and custom tools [2].

Once this is done, components can be (largely) independently developed, or in closely related sets, doing detailed design and careful implementation of components and generator templates. Sometimes, when defects are to be repaired or new features

A. Garcia et al. (Eds.): SELMAS 2002, LNCS 2603, pp. 139–147, 2003.

added, it is a simple matter of enhancing a component or developing a new conforming component. However, at other times, the architecture has to be changed, new interfaces must be defined, and change ripples to many components.

Software agents offer great promise to build loosely-coupled, dynamically adaptive systems on increasingly pervasive message-based middleware, P2P and component technology, Java, XML, SOAP and HTTP [3]. Agents are specialized kinds of distributed components, offering greater flexibility than traditional components. There are many kinds of software agents, with differing characteristics such as mobility, autonomy, collaboration, persistence, and intelligence. Research in our group, previously at Hewlett-Packard Laboratories [4],[5], and now at UC Santa Cruz in collaboration with the University of Utah, is directed at the use of multi-agent systems in the engineering of complex, adaptive software systems. An important step is to simplify and improve the engineering and application of industrial-strength multi-agent systems and intelligent web-services to this problem. The research integrates several different areas, combining multi-agent systems, component-based software engineering, model-driven software reuse, web-services, and intelligent software.

In this paper, we will highlight some of the issues and our progress involved in making this step toward more robust, scalable and evolutionary systems using agent components and reuse techniques.

2 Multi-agent Systems

Multi-agent based systems have several characteristics that support the development of flexible, evolving applications, such as those behind E-commerce and web-service applications. Agents can dynamically discover and compose services and mediate interactions. Agents can serve as delegates to handle routine affairs, monitor activities, set up contracts, execute business processes, and find the best services [6]. Agents can manage context- and location-aware notifications and pursue tasks. Agents can use the latest web-based technologies, such as Java, XML and HTTP, UDDI, SOAP and WSDL. These technologies are simple to use, ubiquitous, heterogeneous and platform neutral. XML will become the standard language for agent-oriented interaction, to encode exchanged messages, documents, invoices, orders, service descriptions and other information [7], [8]. HTTP, the dominant WWW protocol, provides many services, such as robust and scalable web servers, firewall access and levels of security.

An overview of agent capabilities from a large-scale AOSE/CBSE perspective can be found in books ([9],[10],[11]), papers ([6],[12],[13],[14],[15]) and web sites (http://agents.umbc.edu/ , http://www.hpl.hp.com/reuse/agents).

While there are many definitions of agents, many people agree that: "*an autonomous software agent is a component that interacts with its environment and with other agents on a user's behalf."* Some definitions emphasize one or another aspects such as mobility, collaboration, intelligence or flexible user interaction. Organizations such as FIPA (Foundation for Intelligent Physical Agents) are defining reference models mechanisms and agent communication language standards [16].

There are several different kinds of agent system [17]; our work at Hewlett-Packard Laboratories [4] focused on two types of agents: