ComponentXchange: A Software Component Marketplace on the Internet

V Sriram, Atul Kumar, Deepak Gupta and Pankaj Jalote
Dept. of Computer Science & Engg.
Indian Institute of Technology
Kanpur, INDIA 208016
{sriram, ak, deepak, jalote}@cse.iitk.ac.in

ABSTRACT

An infrastructure that allows developers to search and locate COTS (Commercial-Off-The-Shelf) software components that best match their given requirements, from those produced by component vendors across the world will accelarate the widespread adoption of COTS Based Systems. In this work, we describe a component exchange that provides such an infrastructure and will act as a global marketplace for software components.

ComponentXchange is modeled on trader based architectures found in distributed object systems. It maintains a repository of XML based component specifications submitted by component vendors. We propose an XML based component specification language that allows a rich set of aspects to be easily specified. Matchmaking is performed by multiple matchmakers each specializing in a particular aspect and the final output is the intersection of results produced by these matchmakers.

ComponentXchange supports two models of component trading. A component can be used either by downloading it and integrating it into the client application or by accessing it remotely over the network. ComponentXchange provides licensing support and can be easily extended to support multiple payment models.

Keywords

COTS, Software Components, XML, Component Trading

1. INTRODUCTION

Building software systems by assembling commercial off the shelf components (COTS) is gaining popularity as this approach has the potential to reduce the cost of software construction. However, currently, there is no infrastructure for searching through the COTS software components produced by component vendors across the world, and locating the software components that best match the given requirements.

In this work, we address this problem by building ComponentXchange, an E-Exchange that will act as a global marketplace for software components. It is advantageous to both component vendors and developers; component vendors gain a large audience for selling their software components and developers gain access to a huge software component repository which can be searched through.

To allow the component users to find the components that best suit their requirements, we propose an XML based component specification, that allows a rich set of component aspects to be easily specified. These aspects may include the syntactic interface of the component, its functional and non-functional (such as QoS properties) attributes, licensing aspects etc.[1]. The specification is extensible and can be easily extended to include other aspects.

2. DESIGN OF COMPONENTXCHANGE

2.1 Trader Based Architecture

The overall architecture of the system is loosely modeled on the trader architectures of distributed objects systems like CORBA. The system architecture primarily consists of three entities, the component provider, ComponentXchange and the client. The component providers register their components with ComponentXchange by submitting a Component Specification Page (CSP), which is an XML document representing the component specification. Clients send queries to ComponentXchange specifying requirements of the desired component through a web-based interface. ComponentExchange performs matchmaking by invoking multiple matchmakers each specializing in a particular aspect and the final output is the intersection of results produced by these matchmakers. The model ensures that clients are provided with a list of software components that best match their given requirements.

2.2 Access Models

There are two ways in which a software component may be *purchased* through ComponentXchange. First, the software component may be bought and then downloaded, (it can then be integrated into the client system by a developer). Second, the component continues to reside at the developer site, and a buyer buys its services. The services are accessed remotely across the network. This resembles the Application Service Provider (ASP) model prevalent in contemporary business computing environments.

2.3 Comprehensive Specification of Software Components

It has been observed that in general, it is not an easy task to assemble software components into systems[1]. A major issue of concern is the mismatches of the components in the context of an assembled system [2]

As discussed earlier, we define an XML based component specification that allows a rich set of component aspects to be easily specified. We characterize a software component along the following aspects.

- Syntactic interface: It includes properties and operations of the component. Additionally, it may also include the events that the component generates or receives.

- Constraints: It referes to the semantic constraints imposed on the interface elements. It can be modeled in our system by extending our component specification schema to integrate languages like OCL(Object Constraint Language).

- Functional properties: It refers to the set of properties that affects component's functional behavior.

- Non-Functional attributes: Non-funtional characterictics of components significantly affect their overall quality. Thus, it is important to specify the non-functional characteristics of software components. It includes QoS attributes such as reliability, performance.

- Context Dependencies: In order to make the context dependencies explicit, the component description language should specify the interfaces required by the component. Addtionally, it should also include a description of the produced and consumed events.

Every Software Component is associated with an XML document (Component Specification Page) that describes the Component in terms of the different aspects listed above. We have designed an XML schema for the same purpose.

2.4 Matchmaking

Matchmaker is a component that takes client query as input and determines the components that best match the given query.

Requirements Matchmaking: Here we refer to matchmaking of syntactic interface of the component, its semantic constraints and its functional properties. Constraints on the syntactic interface of the required component are expressed by partially specifying the desired interface elements of the required component. Property predicates are used to represent the functional properties that the client is interested in. These are represented as an XML string and matchmaking is performed based on this input. The results of matchmaking returns those components that are most similar to the client requirements. This matchmaking can be enhanced by adding matchmakers for semantic constraints and domain specific matchmakers for functional properties.

QoS Matchmaking: To characterize the QoS properties of components and to enable QoS-Matchmaking, an XML schema has been designed which borrows concepts from QML (QoS Modeling Language)[5]. This has been integrated with the component specification schema. A Component is characterized by a set of Contracts (e.g. Reliability, Performance, etc), the Contract itself being specified by a set of constraints along multiple dimensions. The client requirement is also expressed as set of Contracts and the QoS-Matchmaker applies an algorithm to determine the set of components whose Contracts *Conform* to the client requirement Contracts.

2.5 Domian-specific Trading Communities

In closed distributed object systems, all parties in trading (Client, Server and Trader) share a common type model. Althogh it is practically impossible to impose such a shared type model over open systems like the Web, we envisage the development of vocabularies that standardize operational signatures and constraints in specific domains. This can be used for run-time discovery of components by applications without human intervention, thus allowing evolution of domain-specific Trading Communities.

2.6 Licensing Support

In ComponentXchange, licensing support will be provided through a licensing server. A *sold* component interacts with licensing server to ensure that the licensing terms are being met. The interaction is through a standard API that is similar to CORBA licensing service. The sold software components periodically send messages about their usage to ensure that client does not use the component in an unauthorized manner. Based on the usage information, we can implement different payment models like pay-per-use or a pay-per-user.

3. IMPLEMENTATION STATUS AND FUTURE WORK

We have developed an XML schema for comprehensive specification of software components. We have developed matchmakers for matching syntactic interface, functional and non-functional attributes of components. Currently, we are integrating the licensing model into the system. A web-based interface to the system (using servlets) exists for component providers and developers to interact with ComponentXchange. Our model can be enriched by extending our component specification to include more component aspects and developing matchmakers for those aspects.

4. REFERENCES

[1] Jun Han. An Approach to Software Component Specification: In proceedings of *1999 International Workshop on Component-Based Software Engineering*

[2] D. Garlan, R. Allen and J. Ockerbloom. Architectural Mismatch: Why reuse is so hard. IEEE Software, 12(6): 17-26.

[3] OMG CORBA Trader Object Service. Version 1.0.

[4] OMG CORBA Licensing Service. Version 1.0.

[5] Svend Frolund and Jari Koistinen. QML: A language for quality of service specification. *Technical Report HPL-98-10, Hewlett-Packard Laboratories, February 1998*

[6] XML Schema Part 0:Primer, *W3C Candidate Recommendation 24 October 2000* http://www.w3.org/TR/xmlschema-0/