SoftDock: A Distributed Collaborative Platform for Model-based Software Development

Junichi Suzuki
Department of Computer Science, Graduate School of Science and Technology, Keio University
suzuki@yy.cs.keio.ac.jp

Yoshikazu Yamamoto
Department of Information Computer Science, Keio University
yama@ics.keio.ac.jp

Abstract

This paper describes our SoftDock system, which is a distributed model management system to support collaborative software development. SoftDock leverages the team development, tool interoperability, and model continuity across development lifecycle. It supports Unified Modeling Language (UML) to specify software models, and maintains the model information with UML eXchange Format (UXF), which is a XML-based model interchange format. UXF-encoded model descriptions are shared and managed through the Document Object Model (DOM) interface implemented on top of CORBA (Common Object Request Broker Architecture) middleware. This paper describes the SoftDock architecture, system organization, and design considerations in managing UXF descriptions in the distributed environment.

1. Introduction

The current software development projects are becoming more network-centric, because effective and economical communication mediums (e.g. the Internet) are available, and because the knowledge and skills required for a large project often cannot be found in one location. The distributed software development requires to provide model continuity across software lifecycle by connecting various tools to a uniform development platform. This allows to develop software components independently of physical places and assemble them later.

For distributed development projects, there are some issues to be resolved:

- Modeling software components and relationships (e.g. dependencies, associations, etc.) between them, which involves a modeling language that defines interchangeable semantics of components.
- Describing and interchanging component model information, which involves a model exchange format.
- Providing a uniform means to access and circulate components, which involves a set of published interfaces and communication protocol.
- Ensuring the integrity of component models, which involves the model tracking and storage facility.

We are using Unified Modeling Language (UML) [1], for the first issue, to specify object-oriented software models. For the second issue, we have provided an application-neutral interchange format for UML, called UXF (UML eXchange Format) [2-4]. The SoftDock system addresses from third to fifth issues by managing UXF descriptions through the DOM (Document Object Model) interface [8] built on a CORBA (Common Object Request Broker Architecture) middleware [11].

SoftDock is a distributed model management system to support collaborative software development. This paper presents its architecture and design considerations in maintaining UXF-encoded model information on the network environment in a standard-based manner. It allows the team development, reuse of software models and tool interoperability by increasing the availability and timeliness of interoperable model information throughout a project.

The remainder of this paper is organized as follows. Section 2 overviews enabling technologies used in the SoftDock system: UML, XML, UXF, DOM and CORBA. Section 3 describes the SoftDock architecture, its capabilities, design considerations, and our current implementation. We conclude with a note on future work in Section 4 and 5.

2. Enabling Technologies

This section briefly describes enabling technologies used in SoftDock. Please see [2-6] for more depth discussion.

2.1 Unified Modeling Language (UML)

The Unified Modeling Language (UML) [1] is the union of the previous leading object modeling methodologies; Booch, OMT and OOSE. It is a standard modeling language that defines most of the semantics and their notations required for representing object-oriented software constructs. UML provides nearly independent 9 diagrams for modeling a given problem domain in terms of various perspectives. UML has been widely accepted by academic and commercial developers, and used for representing various software models including real-time system models, hypermedia models, business models, engineering design models, multi-agent models, etc.

2.2 UML eXchange Format (UXF)

UXF is an XML-based format to describe and interchange UML models [2-4]. XML is a sophisticated subset of SGML (Standard Generalized Markup Language). UXF is carefully designed to be simple and well-structured for encoding, publishing, accessing and interchanging UML models, thereby it allows UML models to be highly interchangeable. UXF facilitates:

- Interoperability between development tools:

  Software models are dynamically changed in the analysis, design, revision and maintenance phases. Software tools used in each phase usually employ their own proprietary formats to describe the model information. UXF allows UML models to be interoperable between development tools throughout the lifecycle of software development. Once encoded with UXF, the model information can be reusable for a wide range of different development tools with different strengths. This seamless interoperability increases our productivity of UML modeling.

- Intercommunications between software developers:

  The Internet is a promising infrastructure to distribute and share software model information, because it is effective and economical for making information available to the separated group of individuals. UXF serves as a communication vehicle for project members on the network environment. Project managers, architects, analysts, designers and engineers can communicate their modeling insights, understandings and intentions on UML models with each other. For example, We may write down model information into an electronic mail, or use a distributed communication system to transfer UXF descriptions. UXF simplifies the circulation of UML models between software developers.

2.3 Document Object Model (DOM)

DOM is a platform and programming language neutral interface that allows to access and manipulate the content and structure of XML documents. It provides a standard set of objects for representing documents, a standard model of how these objects can be combined, and a standard API to access these objects. The DOM APIs provides the following capabilities [8]:

- *Structure navigation*, which is the navigation of document structures such as accessing and searching elements or attributes.
- *Document manipulation*, which is the manipulation of document structures such as adding, changing and removing elements or attributes.
- *Content manipulation*, which is the manipulation of document contents such as putting and getting values to elements or attributes.

The DOM interface is defined with the OMG Interface Definition Language (IDL), a primary component in the CORBA specification (see below), because it is designed to define language neutral. DOM does not intend to be implemented with CORBA, but developing the DOM interfaces on CORBA is reasonable strategy for our goal to make model information highly interchamgeable on the network environment.

2.4 Common Object Request Broker Architecture

Common Object Request Broker Architecture (CORBA) [11] is a standard for object middleware used in the heterogeneous environment. It provides a standard way to interoperate distributed objects. CORBA defines the interfaces and components that organize Object Request Brokers (ORBs). An application accessing the service of a remote object uses an ORB to send a message and receive the results. A series of interfaces in CORBA allows to distribute remote objects on multiple platforms in a transparent way to applications. One of the key components in CORBA is OMG Interface Definition Language (IDL), a language to define the interface of a remote object. It is programming language neutral by providing a mapping to various languages. Another important component is Internet Inter-ORB Protocol (IIOP), which is a standard on-the-wire protocol based on TCP. IIOP leverages the interoperability between objects on different ORBs.

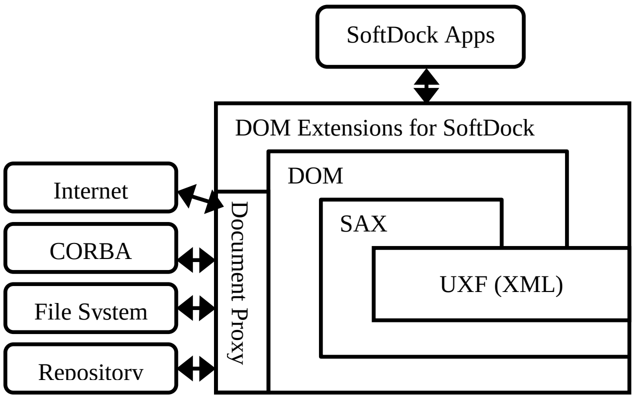

Figure 1: SoftDock architecture

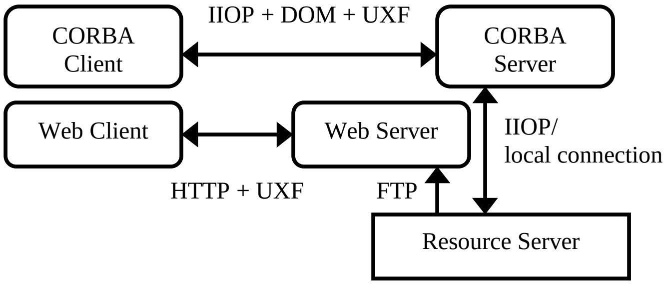

Figure 2: SoftDock system organization

3. SoftDock Overview

This section describes the SoftDock architecture, its system organization, foundation capabilities and some design considerations.

3.1 SoftDock Architecture

The primary goal of SoftDock is to provide the distributed collaborative environment that shares and manages the UML models in the standard-based manner.

In general, the APIs that handle XML documents via middleware are categorized into three groups [9]:

- Source document API

  Manages the XML document instances directly.

- Element API

  Manages the parsed elements of a document instance. The navigation, selection, control and update of an arbitrary XML element are performed through this API.

- Custom API

  Provides application specific interfaces, and depends on certain applications with related DTDs.

The custom API is not appropriate for the generic interchangeability of UML models because it should be modified when UXF DTDs or UML itself is revised [5]. If the source document API is implemented in SoftDock, a client application would fetch an entire UXF description from server-side, and parse it to inspect its model information. This means that the larger the size of a UXF description, the longer overhead to transfer it. Therefore, SoftDock provides the element API based on the DOM interface on a CORBA middleware. This allows a client application to access remote UXF descriptions through a set of generic interfaces, thereby the interface can be independent of the update of UXF DTDs or UML, and the size of UXF descriptions.

Figure 1 shows the SoftDock architecture, which describes the relationships between different APIs to access component model specifications encoded in UXF. UXF descriptions are manipulated through either DOM or SAX (Simple API for XML). SAX is a de-facto standard parser interface, the interface between a XML parser and its applications, which has been developed in the XML community. SAX is an event-based parser interface, while DOM is tree-based [9]. DOM compliant parsers often use SAX-based parsers internally. Any parsers supporting either DOM or SAX can be plugged into our architecture without affecting other components in the system. The extension part in the architecture provides a series of utility objects used for building SoftDock applications. One of the most important objects in this part is a document proxy. Its responsibilities are:

- Fetching remote model specifications consistently.
- Connecting a parser interface with external environments such as the Internet, CORBA, file systems and repositories (see Figure 1).

Figure 2 shows our current system organization. All the UXF descriptions are stored in the resource server. The current SoftDock system provides the connectivity with a web server and CORBA environment. The HTTP connection provides the oneway communication between a client and server, which aims to allow client applications including web browsers to refer model information within a resource server through a web server. Whenever a UXF description is updated in resource server, SoftDock pushes the updated description to a web server.

The IIOP connection provides the two-way communication between clients and servers, which aims to allow developers at separated places to refer, create and modify arbitrary model specifications. Client applications include CASE tools, documentation tools, design metrics tools, source code generators, reverse engineering tools.

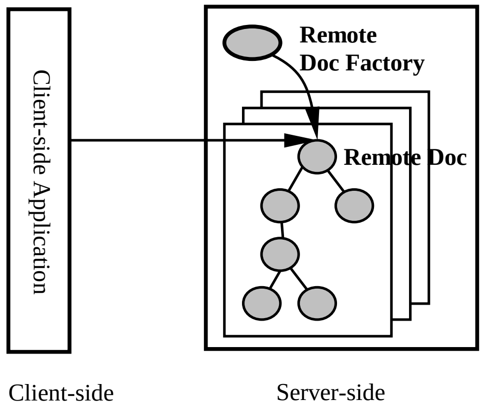

Figure 3: Remote access to a UXF description

3.2 SoftDock Foundation Capabilities

Currently, SoftDock provides the following foundation capabilities for managing UXF descriptions:

- *Model metadata management*

  Allows for maintaining and searching UXF descriptions in more effective and semantic-aware way. We are using the CORBA naming and trading services (see Section 3.3).

- *Local caching of remote UXF descriptions*

  Allows for accessing to UXF descriptions efficientively by reducing the numbers of remote invocations [6, 7].

- *Synchronous and Asynchronous editing*

  Allows client applications to manipulate a cached UXF description and merge the change to a remote original one later [6, 7].

- *Change notification*

  Ensures data consistency between original server-side description and cached one [7]. The change events are transferred using the CORBA event service.

The next section describes the first capability: how to share and search model information in distributed environment.

3.3 Model Management Interfaces

As described earlier, SoftDock allows client applications to access remote UXF-formatted model information through the DOM interface built on CORBA. Every UXF description is parsed into a tree structure according to its tag hierarchy with a DOM-compliant XML parser, and maintained as a set of CORBA objects (Figure 3). A UXF tag is mapped to a DOM node, which is in turn mapped to a CORBA object. All CORBA objects are registered into the

```
#include <dom.idl>
#include <CosEventComm.idl>
module SoftDockExtention {
  typedef long UID;
  interface UXFDescription
    :dom::Document,
    :CosEventComm::PushConsumer,
    :CosEventComm::PushSupplier
  {
    attribute UID nodeId;
    readonly attribute float revision;
    void externalize();
    sequence<string> content();
    boolean isLocked();
    oneway void lockNode(in dom::Document doc,
                         in CorbaDocAgent agent);
    oneway void releaseNode(in dom::Document doc);
    oneway void update(in string docName,
                       in UID    changePoint,
                       in short  changeTypeId);
  };
  interface CorbaDocProxy
    :UXFDescripion,
    :CosEventComm::PushConsumer,
    :CosEventComm::PushSupplier
  {
    attribute dom::Document remoteDoc;
    attribute dom::Document localDoc;
    oneway metadata(in sequence<string> mdata);
  };
  interface CorbaDocFactory
  {
    dom::Document createDocument(in string docName);
    dom::Document cloneDocument(in dom::Document doc);
    oneway void releaseNode(in dom::Node node);
    void destroyDocument(in dom::Document doc);
  };
};
```

Figure 4: DOM extension interfaces for SoftDock

Basic Object Adapter (BOA), which is a server-side facility used for managing their lifecycle and dispatching incoming requests to them. Every root node (or object) of tree structures is published to client applications using the CORBA naming service, a *white page service* for CORBA objects. When a new UXF description is introduced, a resource server (see Figure 2) parses it and registers its root node to a name server with the file name of the description. Once a client application locates the root node of a target UXF description, it can navigate its structure and contents with the DOM interface. Figure 4 shows our extension to the DOM interface. For more depth discussion on this IDL interfaces, please see [6, 7].

In addition to a white page service, SoftDock provides a *yellow page service* for each root node of UXF descriptions: a *match-making service* that returns all objects that satisfy particular characteristics and/or conditions. This service is performed using the CORBA trading service. The service assigns a *service offer* to each CORBA object, which contains a *service type*, an object reference and a set

The service type name: `SoftDockUXFModel`

The interface type that this service type conforms:

`IDL:jp.ac.keio.softdock/SoftDockExtension/UXFDescription:1.0`

Property	**Mode**	**Type**	**Description**
`authorName`	mandatory	`string`	The name of model author
`modelElementKind`	mandatory	`string`	The kind of model element
`elementName`	mandatory	`string`	The name of model element
`lastUpdateTime`	optional	`short`	The last update time
versionNumber	optional	`float`	The revision number

Figure 5: Service type property definition for the SoftDock yellow service

of property values. SoftDock defines its service type as shown in Figure 5. A Client application filters service offers based on properties listed in Figure 5, and selects appropriate UXF descriptions without having to incur the overhead of invoking operations on remote objects. This yellow page service provides more semantics-aware search mechanism than the white page service.

4. Current Project Status and Future Work

We are investigating more effective set of properties definition used in the yellow page service for providing a higher level of match-making, while it is very simple at present.

Also, we plan to support an alternative model access mechanism using XQL (XML Query Language) in addition to the current DOM-CORBA channel. XQL will be implemented on both the HTTP and IIOP connections. For the HTTP connection, we are also investigating the feasibility of WebDAV [10], which is an extension to HTTP 1.1 for the collaborative authoring on the web.

5. Conclusion

This paper describes our SoftDock system and proposes a solution for distributed model-based software development. Our system increases model continuity across development lifecycle by managing and interchanging UML models consistently with standard technologies. We describe how to maintain model metadata for sharing and searching model information effectively.

6. References

[1] Object Management Group. *Unified Modeling Language Specification. version 1.3R7*. available at uml.systemhouse.mci.com, 1999.

[2] J. Suzuki and Y. Yamamoto. Making UML Models Exchangeable with XML over the Internet: UXF approach. In *Proceedings of the UML'98*, June 1998.

[3] J. Suzuki and Y. Yamamoto. Managing the Software Design Documents with XML. In *Proceedings of the ACM SIGDOC'98*, September 1998.

[4] J. Suzuki and Y. Yamamoto, Making UML Models Interoperable with UXF. In *LNCS on Unified Modeling Language*, Springer-Verlag, 1999, to appear.

[5] J. Suzuki and Y. Yamamoto. Toward the Interoperable Software Models: Quartet of UML, XML, DOM and CORBA. In *Proceedings of the ISESS '99*, May 1999.

[6] J. Suzuki and Y. Yamamoto. Distributed Management of Component Framework Specifications. Submitted to *the 5th Business Object Component Workshop at the ACM OOPSLA'99*.

[7] J. Suzuki and Y. Yamamoto. Building A Next-Generation Infrastructure for Agent-based Distance Learning. In *the International Journal of Continuing Engineering Education and Life-Long Learning*, November 1999. to appear.

[8] M. Champion et. al. (eds.). Document Object Model Level 1 Specification. W3C Recommendation, 1998.

[9] K. Ohno and M. Bayer. Development of SGML/XML Middleware Component. In *Proceedings of the SGML/XML'98 Europe*, 1998.

[10] E. J. Whitehead, Jr. and M. Wiggins. WEBDAV: IETF Standard for Collaborative Authoring on the Web. In IEEE Internet Computing, Vol2. No. 5, Sept/Oct, 1998.

[11] Object Management Group. *Common Object Request Broker Architecture 2.2*. available at www.omg.org/library/c2indx.html,1998.