A Web-Service Based Approach for Software Sharing

Yu Xiong and Daizhong Su*

Advanced Design and Manufacturing Engineering Centre,
School of Architecture, Design and the Built Environment
Nottingham Trent University, UK
* Corresponding author, daizhong.su@ntu.ac.uk, URL: http://admec.ntu.ac.uk

Abstract: A Web-service based approach is presented which enables geographically dispersed users to share software resources over the Internet. A service-oriented software sharing system has been developed, which consists of shared applications, client applications and three types of services: application proxy service, proxy implementation service and application manager service. With the aids of the services, the client applications interact with the shared applications to implement a software sharing task. The approach satisfies the requirements of copyright protection and reuse of legacy codes. In this paper, the role of Web-services and the architecture of the system are presented first, followed by a case study to illustrate the approach developed.

1. Introduction

The authors are currently involved in two EU-China projects supported by the European Commission's Asia Link and Asia IT&C programmes [1, 2], which include five geographically dispersed teams from four countries. In order to conduct the projects, the team members have to share their software resources, such as CAD packages, design tools, analysis software and calculation programs, over the Internet. Most of the software tools/packages/programs are not initially designed with distribution features and are not all written in the same computing languages. To meet the demand, an effective software sharing system has to be developed.

The conventional way of software sharing is to give the binary or source file to the users. However, for the following reasons, the owner of the software probably neither wants to give the binary file nor the source code to other people. First, the executable file could be easily cracked by anti-compile tools and delivered to other people who have no rights to use it. The situation would be even worse when source codes are given, the copyright would no longer be protected. Secondly, the software may be released with a lot of copies being used in many different places. In such a situation, if a serious bug is discovered, it would take a lot of time to update all the copies. Sometimes the software development team may want to recode the old version of the program to make it adapt to new requirements, and the spread of the old version makes the update difficult. Moreover, some software needs to request information from a database belonging to another organization, which may not wish to give permission to share the resources with users other than the service providers.

To overcome the above drawbacks, a new way to share software, i.e. service-oriented software sharing, is developed by the authors. This method neither gives the binary nor the source file to users; instead, it packs the binary files of the software resources, so called shared applications, within a Web service and provides users an interface of the service. The user of a shared application receives the description file of the interfaces, with which the user could build a client application to access the shared application via Internet. In this way, the software owner does not need to worry about the software sharing problems mentioned above and the software could be shared efficiently. However, most of the existing software programs/packages have not been designed for this purpose or do not follow the right structure in so doing. To resolve the problems, a method to package the software based on the Web service technology is developed by this research.

In the following sections, after a brief introduction of the Web-service based approach for software sharing, the architecture of service-oriented software sharing system and its implementation are presented, followed by a case study to illustrate the approach developed.

2. Service-Oriented Software Sharing and Web-Service

2.1 Requirements for the service-oriented software sharing

In the service-oriented method for software sharing, the term 'service' refers to a logic view of the real application defined in terms what it does, typically carrying out the business logic of an application. The service interface is an end-point where the particular business logic of an application could be invoked over a network. The description file of the service interface is accessed by the service requester. With the file, a client application, which interacts with the service to complete a task, is then built. This method enables the service requester to share not only the business logic of an application, but also the resources associated with the application. It provides enough facilities for users to allow their client applications to integrate such services for their use. Sharing software in such a way, the drawbacks of the traditional method for software sharing are overcome, and, hence, software can be efficiently shared.

Although the new idea of software sharing mentioned above sounds simple (once someone has thought of it), the implications are often subtle. There are some requirements that should be taken into consideration:

- It should support access control, for the reason that the software owner may allow only selected people to share his/her software.
- Since many existing legacy applications have not originally been designed for distribution purposes, the issue how to convert these applications into distributed ones has to be considered.
- The software performance is also a very important issue.

2.2 The role of the Web service in service-oriented software sharing

There are some existing distributed technologies available for the purpose of software sharing over the Internet. However, the ways for those existing techniques to interact with the applications via the Internet are not compatible with each other. For example, the IDL (Interface Define Language) used by CORBA cannot fit the Servlet/JSP architecture [3]. Generally speaking, every client with such a technique could only invoke its own technique-compatible server application, which greatly reduces the flexibility of building client applications. Such a drawback could be avoided by Web service.

Web service is an emerging technology and provides a Service-Oriented Architecture (SOA) [6, 7], supporting interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format called WSDL (Web Service Description Language) [8]. It enables clients to establish the client-side program in a language/technique with their preferences, to interact with the service by following the service description using SOAP messages [9].

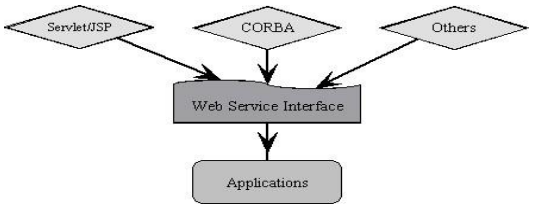

Fig. 1 Different clients access applications via Web service interfaces.

For the Web service approach, once the shared application, i.e. the software, is packed into a service, the uniformed service interface, which is provided by WSDL [3], is then published. After such WSDL files are discovered by other people, who could build their own applications interacting with the service interfaces, and it is not necessary for them to understand the internal structure of the service. The service could be visited by programs written in JSP/Servlet, CORBA or others over Internet even could be directly visited by an unknown application. Multiple users with different client applications could visit the same service at the same time as shown in Figure 1. For example, while some people are using a web browser to invoke the service, another batch of users use their own customized client-side programs to call the service. So by using web services in software sharing, a legacy program could be easily integrated into the system as a Web service package with distribution features. The same service not only is available for multiple users, but also has multiple ways of accessing.

3. Structure of a Service Oriented Software Sharing System

The Service oriented software sharing system has to be designed to meet the requirements stated in the previous section. The first requirement could be easily

achieved by using Web service. The other requirements are met by careful design of the system components as detailed in this section.

3.1 Constituents of a service-oriented software sharing system

In the service-oriented software sharing system developed using Web-service technique, there are three types of services:

- Application proxy service,
- Application proxy factory service, and
- Application manager service

Besides the three services, there are two types of applications

- *Shared applications* which are the software packages/programs, such as CAD packages, design tools and calculation programs, to be shared amongst the collaborative team members and users, and
- *Client applications* interacting with the shared applications, which are built with the service interfaces description files and are used by the people, who want to access the shared applications.

Application proxy service (APS). The APS provides a surrogate for the process to run a shared application and to control the access to it. It models functions presented in the shared application and exposes the functions as Web services. Actually, an application proxy Web service is similar to a running process of a shared application, but it is accessed remotely. The APS also manages the resource used by the shared application, monitors the situation of the shared application, and handles security and other issues. A client application interacts with an APS to share an application.

Application proxy factory service (APFS). It may be a common situation that the shared applications are not originally designed for service-oriented software sharing. Some of them even cannot be used in a multi-user environment where the service-oriented software sharing method is to be utilised. A single APS cannot serve multiple users, so there would be multiple services to serve different clients. These APSs should be managed, and the resources used by these services should be managed too. The APFS, which is similar to the process management in a local Operation System, is designed to meet the needs of managing the creation and lifecycle of an APS. The resources used by these services are also managed by the APFS. Each shared application has an APFS to create multiply APS instances. The APS and APFS pattern not only make it possible for an application to be packed with a web service, but also make a single user application pretend to be a multi-user one, which is similar to research results presented in [3].

Application manager service (AMS). In an Internet environment, everyone could access the same single Web service using his/her own client application at the same time. Although the APS and APFS are used to make a shared application pretend to be a Web service based on a multi-user one, it is in fact a single user one. The number of clients that these applications could serve at the same time is not as good as a pure Web service one. So the AMS is built to confront this performance issue. Another

reason of building the application manager service is to support the access control mechanism.

3.2 Architecture of the service-oriented software sharing system

Figure 2 shows the architecture of the service-oriented software sharing system. There probably are multiple shared applications installed in multiple computers. Each of them is installed with the APFS. The process of activating the shared applications is represented by the APS created and managed by the APFS. All the APFSs are registered to the AMS and assigned an application identifier. The AMS has an ACL table for security control.

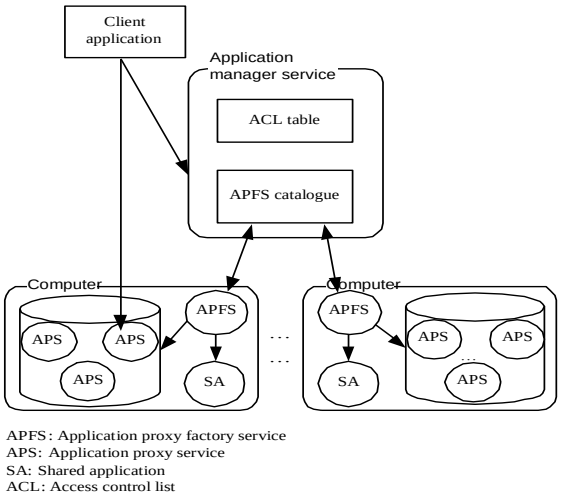

Fig.2 Architecture of service-oriented software sharing system.

Figure 3 shows the process of interaction between client applications and other components within the system. When the user launches a client application, the client application issues a request to get an APS from the AMS. When the request arrives, the AMS looks up the APFS registered in the catalogue using the application identifier specified by the client application. If the APFS exists in the catalogue, the AMS then checks whether the client application has the right to use it. If the access right checking is successful, then the AMS interacts with the APFS to create an APS.

It's possible that multiple APFSs are associated with one shared application. These services may be installed in different locations. When an APFS is requested to create an APS by AMS, it should determine whether a new APS is to be created or not. If the APFS cannot create a new APS, the AMS is then informed, and hence the AMS chooses another APFS instead.

After an APS is created by an APFS, the AMS returns the handle of the newly created APS to the client application. The AMS authorizes the client application with a license to access the APS. Then the client application uses the handle to access the shared application. After the client application completes its task, the APS used by the

client application terminates, and the resources allocated to the APS are then released accordingly.

During the process of interaction described above, the client application cannot access the shared application without authorization. This performance issue is solved by the AMS and APFSs, and the service-oriented software sharing is thus achieved.

It would be a challenging work to assure the reliability of the system, such as handling the exception that one APS is down while it is being processed. This is currently being dealt with in the authors' on-going research.

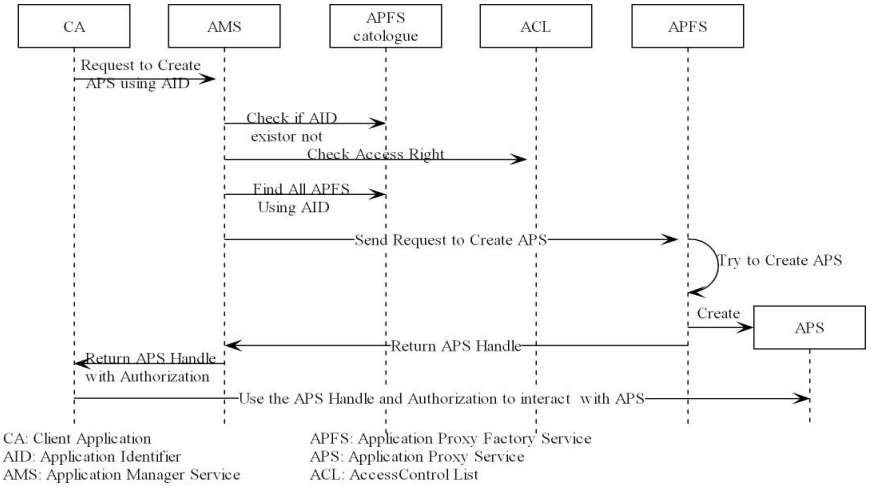

Fig. 3 Process of interaction

4. Implementation of the Approach

The major tasks for implementation of the Web-service based approach for software sharing include packing an application with an APS and implementation of the APFS and AMS. The development of the methods for conducting the tasks is detailed below.

4.1 Application packing

The APS represents a process of activating a shared application. When a request arrives, the APS delegates the request as local function calls to the associated process. The primary issue of this service is how to pack an application with an APS. Four strategies are developed to pack different types of applications as categorized below:

- *Applications with distributed features.* In this case, most applications could be packed with Web service by wrapping the interfaces exposed for distribution purposes. With these interfaces, the APS could be easily built.
- *Applications built with component technology***.** If all the business logics of the application reside within a single component, then the solution is to simply pack

the component object and expose the web service interface. If the business logic resides in multi-component, a new business object, which interacts with multi-components and exposes all of business logic interfaces, should be built, and then the business object could be packed with APS.

- *Applications using shared libraries***.** In this case, the solution is to build adapters to call functions within the shared libraries, and then build a high-level business object to interact with low-level adapters, finally, expose the business interface with the APS.
- *A standalone executable file as an Application***.** This kind of application is very difficult to be packed into services though it is still possible to do so. In most cases, it is recommended to rewrite the application. However, in some cases, applications are really very important and it would be costly to rewrite, some solutions could still be available. For example, Microsoft Visual Studio Net provides some tools to convert a standalone executable file into DLL, and then the strategies mentioned above are applicable.

In other situations, some applications use input stream to read data and output stream to write results. These standard I/O streams could be replaced with or redirected to another input or output streams. For example, in Java language the System.setIn() method could be used to replace standard input stream with an input stream, even a network input stream. So an aided application could be written to replace or redirect the I/O stream of the original application, then expose interface using Web service. Another case is that applications use command line arguments to read data. To deal with this, an aided program with functions to execute files by specified command line arguments are necessary; such functions exist in most programming languages.

For applications, most of which are Graphical User Interface (GUI) applications, do not use standard I/O functions, they could be converted into shared libraries. If users want to reuse these applications instead of rewriting them, which often results in an unimaginable high cost, the solution is to build a bypass library to replace the library needed by the application. The bypass library interacts with the APS. Due to the complexity of the work involved, it is not recommended.

4.2 The implementation of APFS

APFS has a function to create an APS. When a creating request arrives, the application proxy factory should check the status of the local computers to determine whether a new APS could be created or not. After a new APS is created, the APFS allocates the resource required by the APS, and then registers the APS to a proxy service catalogue. This catalogue is stored in the XML format.

The APFS should manage the lifecycle of all the APSs. It periodically checks whether an APS in the service catalogue is still active. If an APS is inactive, the application proxy factory would dispatch a timer for this service. The inactive APS would be removed. If its TTL (Time To Live) is over, the associated shared application process should be terminated. The resource allocated to this process is then released. In current version of the software sharing system developed by the

authors, each application factory service is associated with only one shared application and accessed only by the AMS.

4.3 The implementation of AMS

The main function of the AMS is to deal with the performance and security issues. The AMS provides a catalogue containing URIs of all registered APFSs. The application identifier identifies all the APFSs of a shared application. As mentioned in the above sections, each APFS represents a shared application in a computer. So the application identifier is used to look up all the APFSs of a shared application.

The APFS catalogue is implemented via an interface, which connects to a relational database, typically a MySQL database. Each APFS is registered as a database entry containing an application identifier and URI of the application Proxy factory service. When looking up a specific shared application, the URIs of APFSs with the same application identifier should be returned. Then the AMS interacts with the APFSs using these URIs to create an APS.

In order to achieve higher performance, a shared application could be installed in multiple computers with an APS associated with each one. Then all the APFSs are registered to the AMS. When creating an APS, the AMS polls these APSs and chooses one, which has reported that there are enough resources to create a new APS.

There is also a table used to store the Access Control List (ACL) information for a shared application in the factory service catalogue. In the current version, the AMS use PKI (Public Key Infrastructure) [5] for access control. Each entry in the ACL table contains certificate information of each application identifier with its authorized users who could use the application. When a client application requests to use a shared application, it sends its certificate information in a SOAP header to the application manager. Then the application manager checks whether the client application could use the shared application by using the ACL table. If the client application has access right, the AMS signs a certificate to the client application. This certificate expires after the client application finishes its tasks. In the last step, the AMS uses the APFS to create an APS, and then returns the handle of the APS to the client application, which finally uses the certificate given by the application manager to interact with the created APS.

5. A Case Study

In this section, a software package for gear design optimization, GearOpt, is used to illustrate the Web-service based approach for software sharing. The original GearOpt is of a single user version without distribution features [11]. As the increasing demand of using this application for gear design, it is shared by using the service-oriented software sharing approach.

The original application consists of a graphical user interface (GUI), a genetic algorithm (GA) program and a numerical analysis program for gear strength calculation to the British Standard BS 436. The GUI is used to input data, setting-up the optimization specifications (goals, weight factors, population size and number of

tests) and display results; the GA program conducts the optimization and the numerical analysis program is invoked by the GA program in the optimization process to calculate the tooth strength. All the three parts are integrated into a single software system.

The GA and numerical analysis programs are the core of the software. So the aim is to wrap the GA program and the numerical analysis program with Web service and to build a GUI client application to interact with the Web service. The input parameters of GA program are stored in a file and the location of the file is specified from command line arguments. Such a situation is discussed in section 4. In the Java language, the class Runtime has a method called exec. This method executes an executable file and specifies the command line arguments. After the GA program calculates the gear parameters, it produces the results as a file in its working directory. The Runtime.exec() method also specifies the working directory. So the input and output of the GA application can be redirected.

A program called GA proxy service and its GA proxy factory service were written. When the request for creating a GA proxy service is received by the GA proxy factory service, this service then allocates a file to record the input data, specifies the working directory and creates a new GA proxy service. Each creating request creates a different file stored in different directory and has a different working directory. Otherwise different requests probably clash with each other.

The client application is a GUI application written in Java. It is also used for designing data input, setting-up optimization specifications, and displaying results. The difference between the new and original GUI applications is that the new one interacts with remote GA program with web service. Of course, the client application could be built using other technology in different form. After GA proxy factory service creates a proxy service, the proxy service handle is returned to the client application. Then the user uses the client application to input parameters used in gear optimization and send data to the GA proxy service. The GA proxy service reads the data and stores them in the allocated file; then Runtime.exec() method is invoked to specify the file location and working directory, followed by execution of the application. After this, the results are stored in another file which is read by the proxy service, and then the proxy service returns the results to the client. Finally, the client application displays the result to the user.

In the current version, the GA proxy service, GA proxy implementation service and GA program are installed in multiple computers for performance reasons. The GA proxy implementation services are registered to the AMS using ‘GA’ as its application identifier. People who want to use the application could use the client application to do their jobs. The remote application serves the request and returns the results. But before invoking the application, the user should request an access right to the GA.

6. Conclusion

A Web-service approach for software sharing has been proposed, based on which a service-oriented software sharing system has been developed. It enables

geographically dispersed team members to share their software resources, such as CAD packages, design tools, analysis software and calculation programs, over the Internet. Different from traditional methods of software sharing, the Web-service based approach neither gives the binary nor the source file to users; instead, it packs the binary files of the software resources within a Web service and provides users an interface of the service. It hence meets the requirements of copyright protection, reuse of legacy code, and better performance.

This research utilized the application proxy service, proxy implementation service and application manager service to conduct a software sharing task. It has been approved that the three-service method is successful and effective.

Nowadays, software sharing is a common issue for collaboration over a network, therefore, the Web-service based approach developed by this research has a great potential for applications in a wide range of areas.

7. Acknowledgement

The authors are grateful for the support received from the EU Asia-Link programme (grant No.ASI/B7-301/98/679-023) and Asia IT&C programme (Grant No. ASI/B7-301/3152-099/71553) for carrying out the research reported in this paper.

References

1. EU Asia-Link project, 'Nottingham Trent-Lappeenranta-Chonqing Universies' collaboration for human resource development in mechanical and manufacturing engineering', Contract No. ASI/B7-301/98/679-023.
2. EU Asia IT&C project, 'Web-enabled collaboration in intelligent design and manufacture', Contract No. ASI/B7-301/3152-99/72553.
3. Danny Coward and Yutaka Yoshida, 'Java™ Servlet API Specification version 2.4', JSR 154 Sun Microsystems, Inc., November 24, 2003
4. Linda G. DeMichiel, 'Enterprise JavaBeans™ Specification Version 2.1', JSR 153 Sun Microsystems Inc., November 24, 2003
5. Joel Weise, 'Public Key Infrastructure Overview', SunPS^{SM} Global Security Practice Sun Blueprints™ Online, August 2001
6. Web Services, http://www.w3.org/2002/ws, accessed on 10th December, 2004
7. Service-Oriented Architecture (SOA) Definition, http://www.service-architecture.com/web-services/articles/serviceoriented_architecture_soa_definition.html
8. Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, accessed on 10th December, 2004
9. Simple Object Access Protocol, http://www.w3.org/TR/soap/, accessed on 10th December, 2004
10. Daizhong Su and Shuyan Ji, 'Multi-user Internet environment for gear design optimisation', Integrated Manufacturing Systems, Vol. 14, No.6, 2003, MCB UP Limited.
11. D Su and M Wakelam, 1999, 'Evolutionary optimisation within an intelligent hybrid system for design integration', Artficial Intelligence for Engineering Design, Analysis and Manufacturing Journal (ISSN 0890-0604), No.5, November, 1999, Cambridge University Press, pp 351-363.