Service-Oriented Software Testing Platform*

Fagui Liu[1], Chunwei Luo[1]
School of Computer Science and Engineering, South China University of Technology
510640 Guangzhou, Guangdong, P.R. China
fgliu@scut.edu.cn, lcwiron@163.com

Abstract. Software testing is an important quality guarantee for software. It is an indispensable stage in software life cycle. After the analysis into the idea of the service-oriented systems, this paper tries to integrate service-oriented thinking with software testing to propose a service-oriented software platform to solve the limitations that exist in actual software testing tools, such as costliness, heterogeneity, lack of expansibility, single function, and so on. Service-oriented software testing platform (SOSTP) adopting a distributed structure separates testing client and server. Logical function of testing services is completed on the server end. It can integrate existing software testing tools, and provide corresponding testing services to the clients. The biggest characteristic of SOSTP is the dynamic deployment of testing service, which enables users to customize testing services dynamically. This platform has high extensity and transparency.

1 Introduction

Software testing is an important assurance of software quality. It is an indispensable stage in the software life cycle.

Corresponding to different phases of development, there is different software testing technology. According to the technique, testing technology can be divided into three categories: system structure testing, system function tests and unit testing [1].

So far, many automated testing tools have already emerged, such as Purify [2], Rational Robot, TestBed, CodeTEST, RTInsight, Logiscope, Cantata++, VectorCast and so on. These tools have covered many aspects of static testing and dynamic testing. In allusion to different project operating platform and language development environment, these tools have different development edition.

However, there are some common limitations:

1. The testing tool is comparatively expensive while many software developers cannot bear.
2. The testing tool is too huge, while testing personnel only use partial functions.
3. The single testing tool cannot fully meet testing personnel's demand, so that testing personnel have to purchase many testing tools.
4. Heterogeneity of different programming language causes non-universality of the testing tool.
5. The heterogeneity of the software platform also causes the heterogeneity of the testing tools.
6. Due to the poor performance of many development machines, it is difficult to run some large-scale testing tools.

Therefore, software testing becomes one of the hottest research fields. A lot of scholars have done a lot of research and proposed different software testing tools framework.

SUN Chang-ai [3], LIU Chao, etc. proposed component-based software testing tools integration framework. These components are compatible with CORBA. The framework is scalable. Users can meet their own needs by selecting and reusing software testing function components.

DONG Lei [4] and LU Qiang proposed the distributed software testing platform. The platform adopts client/server architecture. The software testing logic function is concentrated on the server, which effectively lowers the complexity to deploy the testing tool.

However, if the function provided by different testing tool is taken as a kind of service, it is plausible to seek the corresponding service when a testing request is proposed.

Based on this idea, this article attempts to combine the thought of Service-oriented with the software testing, thus proposes a Service-oriented software testing platform.

Service-oriented software testing platform adopts distributed structure. It separates the testing client end and the server end. Logical function of the testing service is completed at the server end. This platform can integrate the existing software testing tools, and provide the corresponding testing service to requesters for testing. The biggest characteristic is that provider of testing service can expanding services to the platform dynamically which empowers users to customize testing services dynamically. This platform has high extensity and transparency.

In the second chapter, the correlative work about the software test framework is presented. In the third chapter, service-oriented software testing framework is proposed and specified.

2 Related Technique

2.1 SOA and Software Testing Platform

Essentially，SOA (Service-oriented Architecture) is a collection of services. Services communicate with each other (these communications can be a simple data transfer or coordination between two or more services.). The so-called service is a function that is precise in definition, well-formed in packaging, independent from the surrounding environment and the state of other services. SOA is a newly developed method emerging from software development in recent years. It is a common method of framework design [5].

Service-Oriented Architecture is a component model. It links different functional units (called services) of the application through well-defined interfaces and protocols. Interface is neutrally defined, independent of platform, systems and languages in format of XML. Through these well-formed interfaces and contracts, all services of SOA interact in a unified and common approach. Moreover, the neutral definition of interface lowers the coupling between the services. The entire application structure will not be affected by the changes of the single individual internal structure [6]. Services call each other through service description.

In the course of development, software testing framework gradually absorbs many software engineering ideas to break through the traditional software testing platform model. How to combine the current upswing SOA idea with software testing platform to solve the mentioned problems of software testing tools is a practical and promising research topic. A service-oriented software testing platform aims to provide an integrated testing environment with multi-function, high scalability and high degree of automation. Therefore, the software testing platform should have some flexibility for users to customize platform functions on their demand, to expand new features, to integrate new tools and to lower development costs.

We can design a service-oriented software platform by the idea of Service Software Bus (SSB). In essence, the main bus architecture is to connect all the functional components to a public mutual communication structural component in a common way. Then the service-oriented software testing platform provides a layer of soft bus structure for users under the interactive interfaces. In this way, if other functional testing components interact with the test platform under the pre-defined interface standard, it can be integrated into the platform, when any other functional testing components attempt to expand to the software testing platform.

2.2 Eclipse Open Source Framework

Eclipse is a software framework followed OSGi (Open Service Gateway Initiative) standard [7], an excellent integrated development environment.

It is a complete and open infrastructure development platform. Graphic tools and other functions can be integrated to the development environment through the

Eclipse-defined interface standards in the form of plug-in, thereby expanding the functions of Eclipse itself. Eclipse can be convenient to expand on the basis of that it provides a concept Extension Point which is similar to the soft-Bus, see Fig. . Platform Runtime is the whole basis of the Eclipse framework. All other components are treated as plug-ins to expand. The basic Eclipse SDK has already included a basic set of development tools, and help system, other tools needed can be customized to and integrated into the platform framework by users. To facilitate the users, Eclipse also provides a PDE (Plug-in Development environment) [8] dedicating to the Eclipse plug-in development. Programs can be developed steadily and rapidly with PDE extension module of Eclipse.

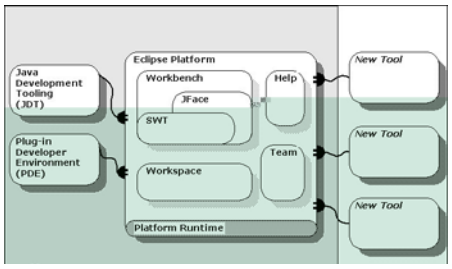

Fig. 1. Eclipse Architecture [9]

3 Service-Oriented Software Testing Platform

3.1 Framework of Service-Oriented Software Testing Platform

As shown in Fig. , the entire framework of Service-oriented software testing platform is made up of client and server components.，which will be described in the next two sections.

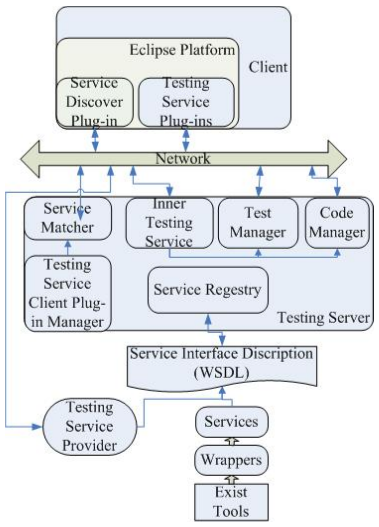

Fig. 2. Framework of Service-Oriented Software Testing Platform

One characteristic of the Service-oriented software testing platform is that testing services can be dynamically located and customized. At client, users can use testing services through customizing testing services. Based on the service-oriented idea, the testing module can be reused and eliminates heterogeneity of platforms. The entire platform is distributed, multi-tier, therefore the testing service concurrent use can be achieved easily.

3.2 Client Design

We implement the design of client of SOA-based software testing platform expediently by using the Eclipse framework and Eclipse plug-in mechanism.

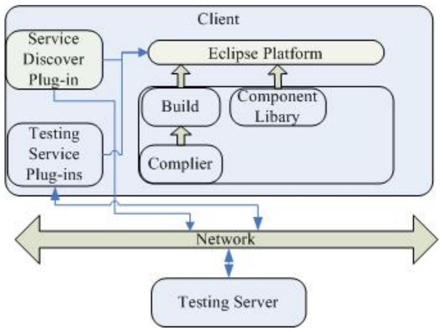

Fig. 3. Design of Software Testing Platform Client

From Fig. we can see that the client framework is implemented by the Eclipse framework, while soft bus is implemented by Eclipse framework. The low coupling between testing service module and the existing of the soft bus reduced the coupling between testing service and platform, which means testing services module can be added or deleted to the platform at any time.

Service Discover Plug-in is a plug-in that integrates into the client platform. Use of this plug-in user can search and customize the testing service at their demand.

Testing Service Plug-ins is also called Client Plug-in, plug-ins used in the client. They bind with customized services. They are developed by service providers. They are provided to services requester. By collaborating with testing services, they help users to generate test cases, submit test scripts, show test results and so on.

3.3 Server Design

The server of testing platform proposed by this paper includes Services Registry, Testing Services Client Plug-in Manager, Services Matcher, Inner Services, Test Manager and Code Manager.

Service Registry is used to provide registry interface to testing services provider. Testing services have to accord to certain criterion to register into the platform. Testing services use WSDL to describe its interface. Each service need provide the corresponding Client Plug-in. Testing services can be divided into several types: The first one is the testing services developed by us, which we call Inner Services. The second one is to use wrapper to package the existing testing tools as services and register to the server. In this way, other testing tools are able to be integrated to the platform. The third one is testing services provided by third party.

Testing Services Client Plug-in Manager is a component used to manage the Client Plug-in when service providers register services, and provide Client Plug-in to users when service is customized.

Service Matcher is designed for the users to discover services. Service Matcher matches the services which are registered according to the user's request description and displays the suited service to users for customization.

Inner Services are testing services integrated in the platform. It need use Code Manager for code management and Test Manager for the management of testing case and results.

3.4 Model of Testing Services

This paper presents the service-oriented software testing platform. Therefore, the testing service model is a very important part. In order to meet service-oriented idea, testing service should have uniform model requested by the testing platform. Only in this way, the testing platform can achieves transparency and scalability. As shown in Fig. , testing service model is made up of service components and client plug-in. They are closely bundled. They can communicate and do logic control through networks. Service component includes Transaction Control, Waiting List Manager, Code Manager, Test Manager, Test Script Manager, Test Execute and testing services interface description.

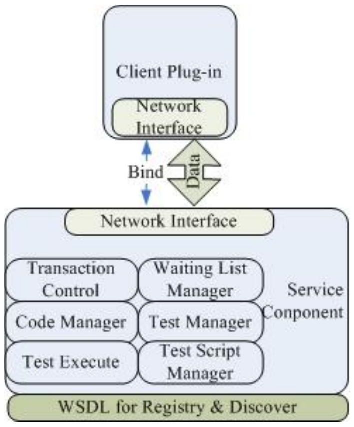

Fig. 4. Model of Testing Service

3.5 Registration of Testing Services

How the Service registration process integrates services to the platform is shown in Fig. .

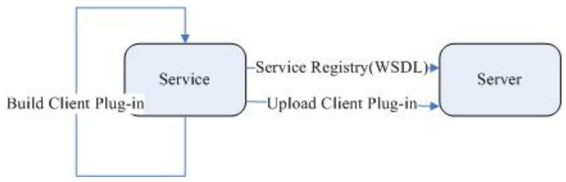

Fig. 5. Registration of Testing Service

In the platform that proposed by this paper, the registration process is as follows:

1. According to the work flow of testing service, service providers develop the Client Plug-in (compatible with the Eclipse Plug-in).
2. Use WSDL to describe the service interface, and register to the server of the platform.

Service providers upload the Client Plug-in of testing services. Test server uses Testing Service Client Plug-in Manager to manage the plug-in. It is provided to users when services are customized.

3.6 Request of Software Testing

In this section, we do not describe the software testing process of a testing technique, which is not the focus of this paper. We describe how the user gets testing services and the process of using the testing service.

Users can use the Service Discover Plug-in to search for required testing services dynamically and customize testing services.

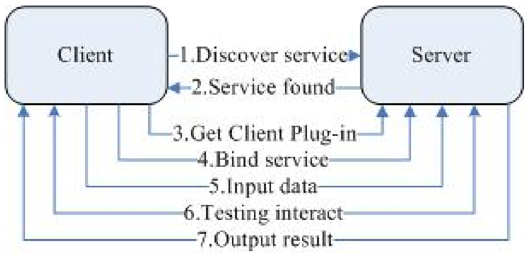

Fig. 6. Testing Process

As shown in Fig. , the software testing process is as follows:

1. Clients use the Service Discovery Plug-in to search the needed services.
2. Server returns the suited service to Client.
3. Client customizes the service and gets the Client Plug-in.
4. Client integrates the Client plug-in into the test environment of client, and use the Client plug-in to bundle with service.
5. According to the testing, the Client inputs the testing data required for.
6. During the testing process, client and testing services interact and collaborate.
7. Upon completion of the testing, the client receives test results.

4 Conclusion and Further Study

SOA based Software testing platform can meet testing needs in different circumstances. If we follow the unified soft-bus standard, both various service and third party implement can be integrated into testing platform conveniently. Adopting Eclipse Framework as a soft-bus standard, users can make a completely personalized software testing platform, according to themselves' needs.

Service-oriented software testing platform can resolve the limitations of the current software testing tools properly. Utilization of this platform can integrate existing software testing tools, and provide testing function in the form of services. It makes these testing functional components reusable. Service-oriented software testing platform can reduce the complexity of the testing software. The service-oriented characteristic makes testers focus on software development, which can reduce the cost of constructing a testing environment effectively. Testers can customize needful testing service when software needs to be tested.

For some test that needs to use the source code, there is a problem that if test service providers can provide the assurance of safety or not.

Further work is how to reinforce the security of service-oriented software testing platform.

Acknowledgments

This research is supported by Guangdong Province Key Field Breakthrough (2005A10208005 & Department of Information Industry 2004-2005).

References

1. William E. Perry, *Effective Methods for Software Testing Second Edition* (China Machine Press, China, 2004).

2. L.B Zhao, X. Wang, and H.T ZHAO, Software test and its supporting tools, Technology & Economy in Areas of Communications (TEAC), No.2, 102-103 (2006)

3. C.A Sun, C. Liu, M.Z Jin, and M. Zhang, Architecture Framework for Software Test Tool, Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems, 2000, TOOLS-Asia 2000, 40-47

4. L. Dong, and Q. Lu, "The design and implementation of distributed software test platform", *Journal of Suzhou Vocational University*, 17(1), 90-93 (2006)

5. IBM corporation, Service Oriented Architecture and Web services (2002); http://www-306.ibm.com/software/solutions/webservices/

6. Rick Robinson, Understand Enterprise Service Bus Scene and Solutions in Service-Oriented Architecture (July, 2004); http://www-306.ibm.com

7. OSGI Alliance, OSGI Service Platform Specification (October 10, 2005); http://www.osgi.org/

8. Eclipse Foundation Inc., Eclipse PDE API Specification (2004); http://www.eclipse.org/pde/

9. Eclipse Foundation Inc., Eclipse Platform Architecture (2004); http://help.eclipse.org/help30/index.jsp?topic=/org.eclipse.pde.doc.user/reference/api/overview-summary.html