TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/21 pp282-287
Volume 10, Number 3, June 2005

Web Services-Based Test Report Generation*

LUO Ling (罗　玲)**, BAI Xiaoying (白晓颖)

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract: Tests involving a large number of test cases and test scenarios are always time- and effort-intensive, and use ad hoc approaches. Test management is needed to control the complexity and the quality of the testing of large software systems. The reporting mechanism is critical for monitoring the testing progress, analyzing test results, and evaluating the test effectiveness for a disciplined testing process throughout the testing lifecycle. This paper presents an XML-based report generation method for large system testing. The service-oriented architecture enables flexible test report generation, presentation, and exchange to facilitate collaboration in a distributed environment. The results show that proper reporting can effectively improve the visibility of the testing process and that this web-based approach is critical to enhance communication among multiple testing groups.

Key words: software test management; report generation; web services

Introduction

In spite of requirements verification, design reviews, and validation tests, software defects are inevitable [1]. Testing is needed to ensure software quality. Research shows that software testing accounts for 40% of the overall project costs in general and even 60%-70% for safety and mission critical systems[2]. For large software systems, testing requires many test cases, as well as the participation of all the parties involved including developers, testers, project managers, and off-site supervisors. Proper test management is critical to a disciplined process, effective communication, and efficient collaboration. Testing seeks to deliver quality applications within the time and budget constraints.

The increasing size and complexity of software systems is drawing more attention to test management methodologies and CASE tools. In the industry de facto standard capability maturity model (CMM)[3], testing is a key process area in level 4, which focuses on testing schedules and defects tracking. TestDirector, a testing product produced by Mercury Interactive Inc., is a customizable, scalable, and flexible solution for test planning, test execution, and defect management. IBM Rational Test Manager is another product for test activity management, execution, and reporting. The purpose of these products is to ensure high visibility of the test coverage information, defect trends, and application readiness.

Reporting is an effective mechanism for data processing and analysis. The system must provide different users with different views of the most-wanted information in a timely and flexible manner. Hence, the reporting capability is the key to successful test management during the whole process of test planning, test design, test execution, test results analysis, and regression testing. Although many third-party reporting tools and components exist such as the crystal report, a dedicated report management subsystem is always needed to improve reporting effectiveness and efficiency.

The reporting system discussed in this paper has the following objectives:

Received: 2004-06-04; revised: 2004-09-19

* Supported by the National Key Basic Research and Development (973) Program of China (No. 041250001)

** To whom correspondence should be addressed.
E-mail: luoling@tsinghua.org.cn; Tel: 86-10-62978931

● The system enables flexible test report generation, including pre-defined periodic reports and on-the-fly reports. The system supports customized report styles and content.

● The system provides statistical and analytical test reports for all the parties involved throughout the life-cycle including reports of all or filtered test cases, test scenarios, test execution records, and test results summaries and analyses. The reports also show the traceability of test execution, test cases, and test scenarios.

● The system uses web services technology which facilitates cooperation in a distributed environment.

1 Test Report Generation

The test management system proposes a document-driven testing process. At each stage of test preparation, design, review, execution, defect management, and regression testing, documents are generated, exchanged, updated, and traced to one another, including project task breakdowns, test plans, execution records, results, defects, etc.

The report generation sub-system is implemented with the purpose of improving process visibility. The system provides different views of the testing project, including:

● Project related reports reporting on the basic project information such as tasks, roles, staff, schedules, etc.

● Test design related reports reporting on the definitions, dependencies, and evolution history of test cases and test scenarios.

● Test execution related reports reporting on the environment, phenomena, and history of the exercise tracing to test cases and test scenarios.

● Test results related reports reporting on the statistical analyses of testing results for the whole system and/or for individual tests.

● Defect related reports reporting on the discovery, correction and regression testing history, and status accounts of defects.

To enforce communication and to facilitate collaboration, different parties are granted different privileges (shown as follows) to access reports within their responsibilities.

● Project manager can access all reports to supervise the testing progress and evaluate test effectiveness and project quality.

● The tester can access all reports related to testing activities to design test cases and test scenarios, validate test designs, exercise tests, record and evaluate test results, open and trace defects, etc.

● The developer can access all the defect-related reports to accept a defect report, modify the corresponding application, update defect status, and trace the regression testing of fixed defects.

The report generation and management is the core of the overall test management system, as shown in Fig. 1.

As shown in Fig. 2, reports definition and customization, report generation, and report presentation are the three major phases driven by the extensible markup language (XML)-based documents which are human-legible and easily processed by programs due to well-formed structures.

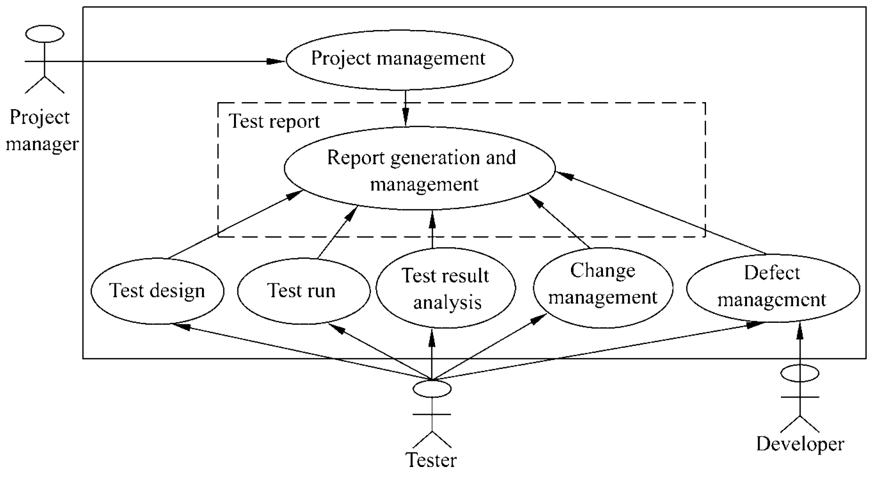


Fig. 1 Test management case diagram

● Report definition and customization: In this phase, the customer defines and customizes the type, content, and presentation style of the report. The system captures parameters such as frequency, time-period, report tables/attributes, and display background/colors. The system then records them in an XML-based definition file whose schema is shown in Fig. 3[4,5].

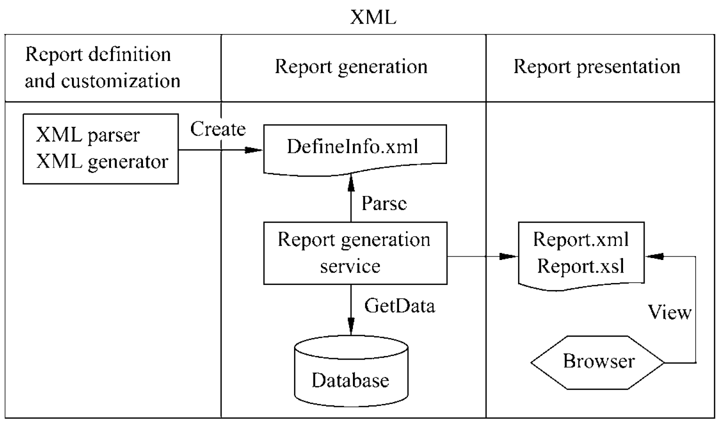


Fig. 2 Report generation process

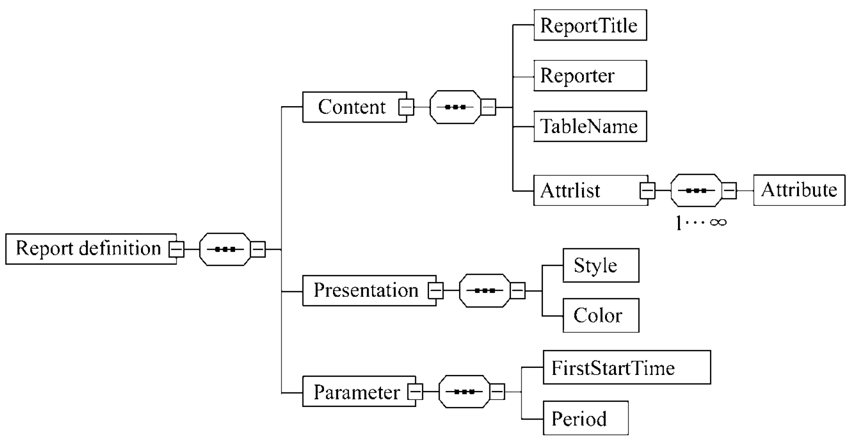


Fig. 3 Report definition schema description

Figure 4 shows an example for the customization of the "Everyday Defect Report".

```
<?xml version= "1.0" encoding="UTF-8" ?>
<reportDefinition>
        <Content>
                <reportTitle>EverydayDefect Report
                </reportTitle>
                <reporter>Sunny</reporter>
        <TableName>Defect</TableName>
                <Attrlist>
                        <Attribute>DefectID</Attribute>
                        <Attribute>Status</Attributes>
                <Attribute>ReporterID</Attribute>
                <Attribute>Description</Attributes>
                </Attrlist>
        </Content>
        <Presentation>
                <Style>list</St yle>
                <Color>white</Color>
        </Presentation>
<Parameter>
                <FirstStartTime>04:4:15:0:0</FirstStartTime>
```

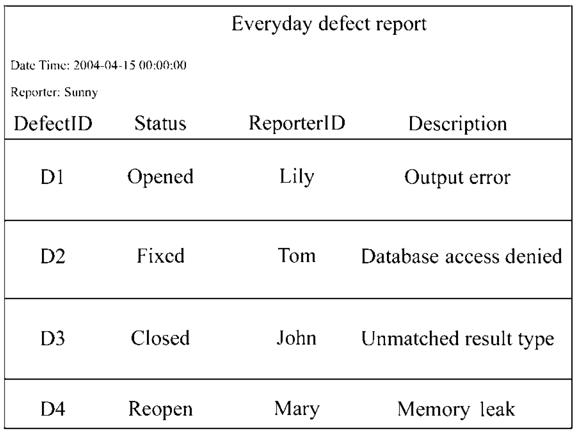
Fig. 4 Report customization XML file

● Report generation: The server side components parse the XML definition file, generate queries, access the database, and produce the XML-based reports.

● Report presentation: An XSL[5,6] file is generated at the same time to describe the presentation of the corresponding XML report. The final "Everyday defect report" is shown in Fig. 5.

2 Web Services-Based Architecture

Web services technology, such as the W3C[7] standards, provides a model for application integration in a distributed environment. A set of simple, open protocols and standards are defined based on XML to support the model. The objective is to establish a universal technical layer independent of hardware/software platforms and



Everyday defect report

Date Time: 2004-04-15 00:00:00

Reporter: Sunny

DefectID	Status	ReporterID	Description
D1	Opened	Lily	Output error
D2	Fixed	Tom	Database access denied
D3	Closed	John	Unmatched result type
D4	Reopen	Mary	Memory leak

Fig. 5 Everyday defect report

programming languages to enforce intercommunication and interoperation among web applications[8]. The three components in the architecture are the service requestor, service provider, and service registry constitute[9]. The web service conceptual stack includes the network layer, the XML-based messaging layer using simple object access protocol (SOAP)[10], the service description layer using web services description language (WSDL)[11], the service publication and discovery layer using universal description, discovery and integration (UDDI)[12], and the service flow layer using web services flow language (WSFL)[13].

The test report generation system uses the service-oriented architecture to enable workflow integration and information exchange among groups of people with different roles and responsibilities working in a heterogeneous environment. Figure 6 illustrates the system architecture.

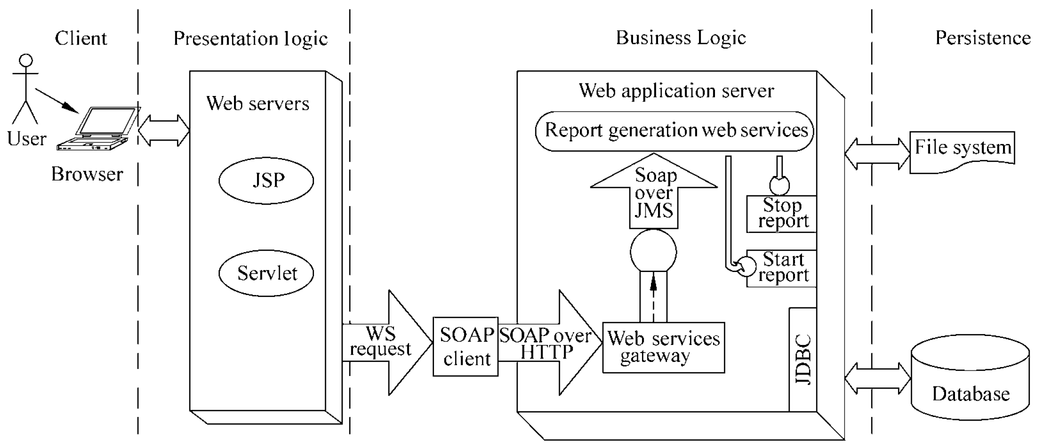


Fig. 6 System architecture

WS, Web services; JMS, Java message service; JDBC, Java database connectivity

1) At the back-end, the file and database systems are used for storing the persistent data. Data information exchange between the database and the file system is based on XML. XML files, such as report definition files and final report files, are stored in the file system for better program understanding and processing.

2) In the middle tier, key functionalities are wrapped as loosely coupled web services, such as stopReports and startReports. These services are deployed in the application server, then published and registered in a UDDI[12] center such as IBM Business Registry or Microsoft Business Registry. The client can look up the services from the UDDI center. Client applications request and bind services via a web service gateway using SOAP over HTTP and SOAP over Java message service (JMS). Figure 7 illustrates the SOAP envelope[10] of the request SOAP message. The SOAP envelope defines the framework of the message content. The SOAP header element contains attributes which define how a recipient should process the SOAP message. The SOAP body element contains the actual SOAP message intended for the ultimate message endpoint. As shown in Fig. 7, the SOAP envelope body describes the definition file path, which is a parameter when the startReport service is invoked.

3) At the front-end, end-users access the system from the browser. The system supports dynamic web content creation through JavaServer page (JSP) and Servlet technology.

```
<!--REQUEST.............-->
<env:Envelope
xmlns:env=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns :xsi= ”http://www.w3.org/2001/XMLSchema-instance”
xmlns :soapenc=”http://schemas.xmlsoap.org/soap/encoding/”
xmlns :xsd=”http://www.w3.org/2001/XMLSchema/”>
<env :Header></env :Header>
<env :Body env :encodingStyle=
”http://schemas.xmlsoap.org/soap/encoding/”>
<m:startReport xmlns:m=”reportgenerate”>
<string xsi:type=”xsd :string”>
F:\CodeSource\CustomizeFile\defineInfo.xml</string>
</m:startReport>
</env:Body>
</env:Envelope>
```

Fig. 7 Request SOAP envelope

3 Application Design and Implementation

The system was implemented using SQL Server 2000 and WebLogic 8.1 within the integration development environment JBuilder 9[14]. Figure 8 shows the major packages and classes. Figure 8 is divided into two parts, the upper part illustrates the report customization logic while the lower part describes the report generation logic.

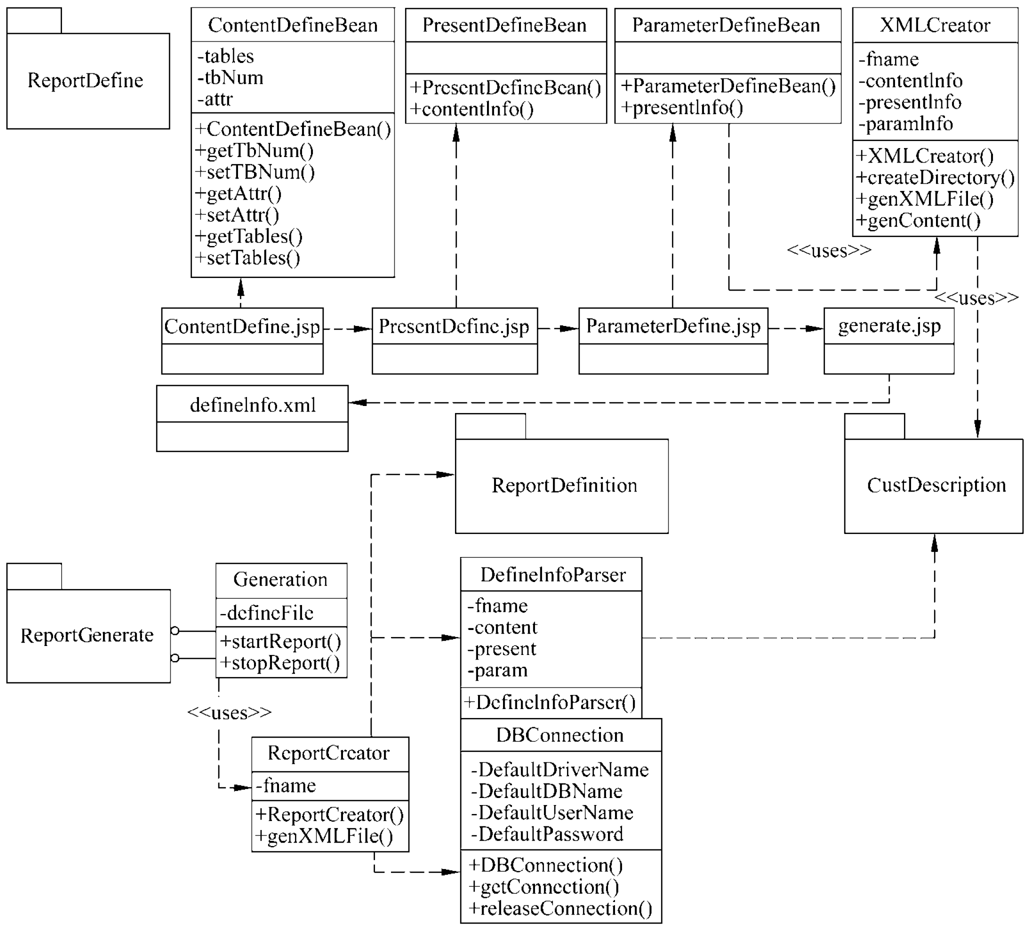


Fig. 8 Packages and classes design for the report generation system

1) Package ReportDefine defines classes realizing the report definition, which includes:

● Class XMLCreator, which is responsible for producing XML-based report definition documents.

● Classes ContentDefineBean, PresentDefineBean, and ParameterDefineBean are imbedded into the corresponding JSP files to produce user input.

2) Package ReportGenerate generates test reports, which includes:

● Class ReportCreator, which is responsible for producing XML-based test reports.

● Class DBConnection, which manipulates the connections between the system, the database, and the database operations.

3) Package CustDescription and package ReportDefinition are generated by the JBuilder XML databinding mechanism, which unmarshals the data from XML objects and converts it to Java objects.

4 Summary

Test reports improve test management effectiveness, thus enhancing the quality of test activities. This paper describes a test reporting system that supports test report definitions, customization, generation, and presentation. The system uses the web-service architecture and XML-based information exchange to facilitate convenient and effective collaboration among various parties involved in the testing process in a distributed environment.

References

[1] Myers G J. The Art of Software Testing. New York: John Wiley & Sons Inc., 1979.

[2] Beizer B. Software Testing Techniques. New York: Van Nostrand Reinhold, 1983.

[3] Capability maturity model for software. Version 1.1. CMU/SEI-93-TR-24.

[4] W3C, extensible markup language (XML) page. http://www.w3.org/ XML/, August 2000.

[5] XML spy online manual. http://www.xmlspy.com/manual/, 2005.

[6] W3C, extensible stylesheet language family (XSL) page. http://www.w3.org/Style/XSL/, February 2005.

[7] W3C, web services activity. http://www.w3.org/2002/ws/.

[8] Clabby J. Web services gotchas: An executive summary. http://www-900.ibm.com/developerWorks/cn/webservices /ws-gotcha/index_eng.shtml, 2002.

[9] IBM Software Group. Web services conceptual architecture (WSCA1.0). May 2001.

[10] Simple object access protocol (SOAP) 1.1. W3C note, 08 May 2000. http://www.w3.org/TR/SOAP/.

[11] Web services description language (WSDL 1.1). W3C Note 15 March, 2001. http://www.w3.org/TR/WSDL/.

[12] UDDI Version3.0. Published specification. 19 July, 2002. http://uddi.orspecification.html.

[13] IBM Software Group. Web services flow language (WSFL 1.0). May 2001.

[14] Sudhansu Pati. A Borland white paper, web services development using Borland JBuilder 9 and Borland enterprise server. http://www.borland.com/products/white_papers/df/webservices_development_using_jb9_and_bes.pdf.

ScienceDirect of Elsevier Covers *Tsinghua Science and Technology* from 2005

The electronic version of *Tsinghua Science and Technology* (English version of *Journal of Tsinghua University*) is covered by ScienceDirect produced by the Elsevier Company from 2005. Readers who search ScienceDirect will find related papers in *Tsinghua Science and Technology* (Website: http://www.sciencedirect.com/science/journal/10070214), which will increase the influence of *Tsinghua Science and Technology*.

Elsevier is the leading science, technology, and medical information publisher in the world. Its ScienceDirect is the biggest online research document database in full text, including more than 1800 journals, more than 60 000 000 abstracts and more than 6 000 000 full-text documents.