The Technique of Test Case Design Based on the UML Sequence Diagram for the Development of Web Applications*

Yongsun Cho, Woojin Lee, and Kiwon Chong

Department of Computing, Soongsil University, Seoul, Korea
yongsuns@hanafos.com
bluewj@empal.com
chong@comp.ssu.ac.kr

Abstract. The systematic testing is frequently regretted in recent web applications because of time and cost pressure. Moreover developers have difficulties with applying the traditional testing techniques. A technique for generating test cases from the UML sequence diagrams of a web application is proposed for the rapid and effective testing. A test of the web applications is composed of a single web page test, a mutual web page test and an integrated web page test. The test cases for a single web page test are generated from self-call messages and the test cases for a mutual web page test are generated from the messages between web pages. The test cases for an integrated web page test are generated from the messages which are sent to the system by an actor and received back from the system.

1 Introduction

Recent business environments have been changing into Internet business environments and web applications have been developed continuously for various fields such as advertisement, sale of goods and customer support in Internet business environments [1]. Moreover, accurate and rapid development of web applications and preoccupation of market are required according as businesses and services become various and companies compete with each other.

The accuracy of web applications is emphasized in these environments. If the web application does not operate correctly or it discontinues because of malfunctions, it leads to corporate losses and disrepute. To prevent these situations before they occur, it is necessary to test the reliability of web applications. Although many techniques for testing web applications have been studied, these are not enough to deal with this problem. Most of the early techniques for testing web applications checked syntax and the context of html, jsp and asp files or the correctness of the links among them. Furthermore most of the recent techniques used for testing web applications check the operations of a single web page or the call-relations among web pages. The clustering

* This work was supported by the Soongsil University Research Fund.

test for testing the collaboration of web pages is also required for the reliability of web applications.

Therefore, this paper proposes a test case design technique based on the UML [2] sequence diagram for insurancing accuracy of web applications.

2 Related Works

RUP provides guidelines about the technique of extracting test cases from use case for functional test of system [3]. As flow of event in each use case, there are a basic flow and several alternate flows. Scenarios of the use case are made from compounding these scenarios. Variables relative these scenarios are extracted to test data and test cases are extracted by adding the test data and adding necessary conditions. RUP mentions the level of test such as unit test, integration test and system test, but RUP does not provide the technique of extracting test cases according to the level of test. RUP uses use case for functional test of system and uses supplementary specification for non-functional tests. However, RUP does not provide the technique of extracting test cases for non-functional tests and test cases are extracted by heuristic manner. RUP provides only guidelines for extracting test cases.

Ye Wu and Offutt propose the technique of test by extracting test cases from flow of sections of a server program [4]. The kinds of section are atomic section which is an elementary physical unit to identify a part of server program and composite section which is a set of atomic section.

An atomic section is a static HTML file or a section of a server program that prints HTML. An atomic section has an “all-or-nothing property”, that is, either the entire section is sent to clients or none of the section is sent.

Possible execution flows are formally expressed by analyzing program codes and each expression is used to test as a test case in this technique. However, it is difficult to represent test conditions or value of test data with only the expressions. Moreover, innumerable test cases can be extracted if server program is complex and very complex expressions are derived in the case of applying integration test of several pages and system test because the technique considers all execution flows based on white box testing. The technique is difficult to apply in real system so that the practical use of the expressions for testing web applications is remained future work in this study.

Filippo Ricca and Tonella propose two techniques of static verification and dynamic verification for testing web applications [5]. Static verification is a technique to scan the HTML pages in a web site and detect possible faults and anomalies. The syntax of HTML pages or links to other pages is examined in this technique. These examinations are performed in many published tools.

Dynamic verification is a technique to extract test cases by analyzing the relationship of web pages. First, the graph for expressing relationships among web pages is made by ReWeb tool. Moreover tests are executed with TestWeb tool.

This technique has the weak point in that so many candidates of test cases can be extracted and the inside of a web page is not fully tested.

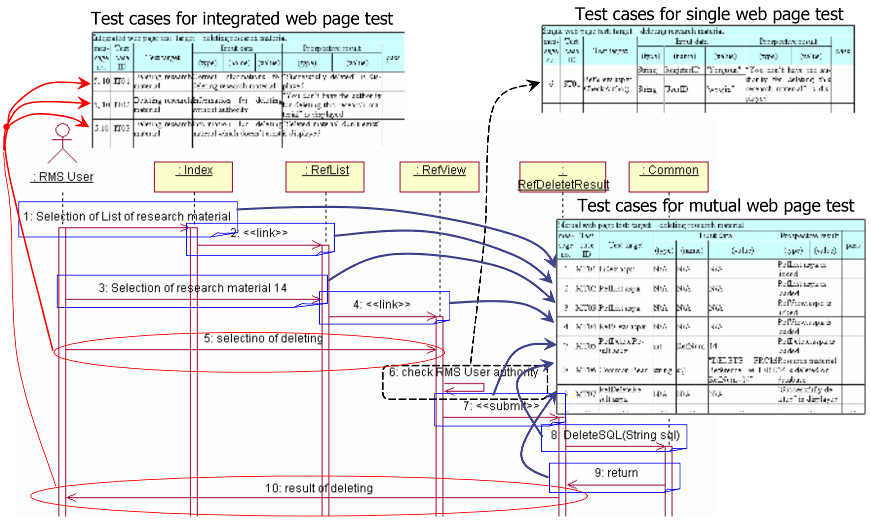
3 Test Case Design

Testing of web applications is achieved, from the smallest unit test to a whole system test. If a submodule operates incorrectly, the test result of the module is not reliable. Therefore it is necessary to test an application level-by-level.

A test of the web applications is composed of a single web page test, a mutual web page test and an integrated web page test. Each level of the test is performed iteratively because detected failures from the test should be corrected and the regression test must be performed in the same environment again to confirm the correctness of the result. Furthermore, related parts of the web application should be tested again because changes after testing can affect other parts of the application.

The transmission of messages among web pages can be expressed using the notations of the UML sequence diagram. The test cases are generated from the sequence diagram of web pages. The test cases for a single web page test are generated from self-call messages and the test cases for a mutual web page test are generated from the messages between web pages. Furthermore, the test cases for an integrated web page test are generated from the messages which are sent to the system by an actor and received back from the system. The technique for generating test cases is as follows:

1. Generating test cases for a single web page test: The self-call messages of a web page are used to call script functions of the page or the page is re-executed by itself. The test cases for a single web page test are generated from these messages.
2. Generating test cases for a mutual web page test: The test cases for a mutual web page test are generated from the messages transmitted between web pages.



Test cases for integrated web page test
Test cases for single web page test
Test cases for mutual web page test
: RMS User
: Index
: RefList
: RefView
: RefDeletetResult
: Common
1: Selection of List of research material
2: <<link>>
3: Selection of research material 14
4: <<link>>
5: selectino of deleting
6: check RMS User authority
7: <<submit>>
8: DeleteSQL(String sql)
9: return
10: result of deleting

Integrated web page test target : deleting research material

message no.	Test case ID	Test target	Input data (type)	Input data (name)	Input data (value)	Prospective result (type)	Prospective result (value)	pass
5, 10	IT01	Deleting research material	Correct information for deleting research material			"Successfully deleted" is displayed		
5, 10	IT02	Deleting research material	Information for deleting without authority			"You don't have the authority for deleting this research material" is displayed		
5, 10	IT03	Deleting research material	Information for deleting material which doesn't exist			"Deleted material doesn't exist" is displayed		

Single web page test target : deleting research material

message no.	Test case ID	Test target	Input data (type)	Input data (name)	Input data (value)	Prospective result (type)	Prospective result (value)	pass
6	ST01	RefView page CheckAuth()	String	SubjectID	"Yongsun"	"You can't have the authority for deleting the research material" is displayed		
			String	UserID	"bengja"			

Mutual web page test target : deleting research material

message no.	Test case ID	Test target	Input data (type)	Input data (name)	Input data (value)	Prospective result (type)	Prospective result (value)	pass
1	MT01	Index page	N/A	N/A	N/A	RefList page is loaded		
2	MT02	RefList page	N/A	N/A	N/A	RefList page is loaded		
3	MT03	RefList page	N/A	N/A	N/A	RefView page is loaded		
4	MT04	RefView page	N/A	N/A	N/A	RefView page is loaded		
7	MT05	RefDeleteResult page	int	RefNum	14	RefDeleteResult page is loaded		
8	MT06	Common Bean	string	sql	"DELETE FROM research material WHERE RefNum=14"	Research material is deleted on database		
9	MT07	RefDeleteResult page	N/A	N/A	N/A	"Successfully deleted" is displayed		
...	...	...	...	...	...	...	...	...

Fig. 1. Test case generation from sequence diagram

3. Generating test cases for an integrated web page test: The test cases for an integrated web page test are generated from the messages transmitted to the system by an actor and the response messages received from the system.

The technique for generating test cases of the system from the sequence diagrams which are developed in the use case analysis step is shown in Figure 1.

The technique for generating test cases from the factors of the sequence diagrams which are developed in the use case analysis step and targets of the test are described in Table 1.

Table 1. The technique for generating test cases of web pages from the sequence diagram

Factors of the sequence diagram		Test case	Target of test
Self-call message	$\rightarrow$	Test case for a single web page test	* Script function * Re-execution of web page
Message between web pages	$\rightarrow$	Test case for a mutual web page test	* Logic of web page * Transmission of message between web pages
Message related to actor	$\rightarrow$	Test case for an integrated web page test	* Achievement of function through several web pages

3.1 Extracting Test Cases for Single Web Page Test

First of all, single web page test is performed for testing of web applications. Each web page is tested in the single web page test. Context and resources are examined for static pages like html pages. Context examination confirms whether the page made out according to syntax and resources examination confirms existence of the linked or called URL. These are easily tested with html editors or html syntax checking tools.

The main target for test are dynamic pages such as servlet, jsp, asp, aspx and php pages that include logics and classes such as bean classes that associated with the web pages. In this stage, independent execution modules in the web pages are tested.

The test cases for single web page test are extracted from self-call messages of each page in the sequence diagram. This case is that the server page calls its own script functions or the page is reexecuted by itself. A web page reexecutes itself in the case of including several functions. For example, a server page reexecutes by itself if the page has the function for registering information and the function for displaying the registeration result. In this case, the web page generally selects one of the two functions according to value of a variable.

The test data are added to the test case if script functions or web pages needs input values. Moreover, the values of objects in the form tag of the web page are considered as test data if the script functions reference the values of objects in the form tag using document object [6]. The related web pages, script functions and variables with these test cases are referenced from the page diagram [7].

Table 2 is an example of extracting test cases from sequence diagram of Figure 1 for single web page test. ST01 is a test case for testing the script function that confirms the authority of user for deleting a research material. In this example, it is tested if the action of system is correct in the case that an inputed user id is different from registered id of the research material. Although one test case extracted from self-call message of sequence diagram is represented in this example, valid input values and invalid input values should be tested.

Table 2. An example of extracting test cases for single web page test

Single web page test: scope – deleting research material								
message no.	Test case ID	Test target	Input data			Prospective result		pass
			(type)	(name)	(value)	(type)	(value)	
6	ST01	RefView.aspx:: CheckAuthor()	String	RegisterID	“yongsun”	“You do not have the authority for deleting this research material” is displayed		
			String	UserID	“woojin”			
…	…	…	…	…	…	…		…

3.2 Extracting Test Cases for Mutual Web Page Test

Independent execution modules such as java script functions in the web page and reexecution of the page are tested in the single web page test. The mutual web page test which examines if the pages are correctly performed in their mutual relation is performed after single web page test. The purpose of the mutual web page test is to examine if a page is linked to another page without loss of information, incorrect transfer of information or error.

The test cases for mutual web page test are extracted from the messages of each page that receives from actors or other pages. The messages of number 1, 2, 3, 4, 7, 8 and 9 in Figure 1 are extracted in this case.

The additional test data are necessary in test cases. The test data are different according to purpose of test and many branchs may occur according to the test data. The typical techniques for determining test data are equivalence partitioning and boundary value analysis [8]. The area of test data is classified for efficient testing and the test data of all class should be tested in the equivalence partitioning technique. It is basis of boundary value analysis technique the fact that many failures occurr around the boundary of input area rather than center. The boundary values are used to test data in this technique. The proper test data for target and purpose of the test should be included in the test cases based on these principles.

Table 3 is an example of extracting test cases for mutual web page test from sequence diagram of Figure 1. MT01, MT02, MT03, MT04 and MT05 are test cases to examine whether each page is correctly loaded on the browser through link. MT06 is a test case to examine if Common Bean class correctly deletes the information from database using inputed SQL statement. MT07 is a test case to examine if the system correctly displays the result of deleting the information for user.

Table 3. An example of extracting test cases for mutual web page test

Mutual web page test: scope – deleting research material								
message no.	Test case ID	Test target	Input data			Prospective result		pass
			(type)	(name)	(value)	(type)	(value)	
1	MT01	Index.aspx	N/A	N/A	N/A	RefList.aspx is linked		
2	MT02	RefList.aspx	N/A	N/A	N/A	RefList.aspx is loaded		
3	MT03	RefList.aspx	N/A	N/A	N/A	RefView.aspx is linked		
4	MT04	RefView.aspx	N/A	N/A	N/A	RefView.aspx is loaded		
7	MT05	RefDelete-Result.aspx	int	RefNum	14	RefDelete.aspx is loaded		
8	MT06	Common Bean	string	sql	"DELETE FROM Reference WHERE RefNum=14"	Research material 14 is deleted on database		
9	MT07	RefDelete-Result.aspx	N/A	N/A	N/A	"Successfully deleted" is displayed		
...	...	...	...	...	...	...		...

3.3 Extracting Test Cases for Integrated Web Page Test

The integrated web page test is performed after single and mutual web page test. The purpose of integrated web page is to examine if the prospective results come through several web pages according to the request of an actor. The test cases are extracted from the messages transferred to the system by an actor and the result of the test cases is extracted from the response messages received from the system.

Table 4. An example of extracting test cases for integrated web page test

Integrated web page test: scope – deleting research material								
message no.	Test case ID	Test target	Input data			Prospective result		pass
			(type)	(name)	(value)	(type)	(value)	
5, 10	IT01	Deleting research material	Correct informations for deleting research material			"Successfully deleted" is displayed and selected material is deleted in database.		
5, 10	IT02	Deleting research material	Information for deleting without authority			"You do not have the authority for deleting this research material" is displayed and selected material is not deleted in database.		
5,10	IT03	Deleting research material	Information for deleting materal which does not exist			"Seleted material do not exist" is displayed		
...	...	...	...			...		...

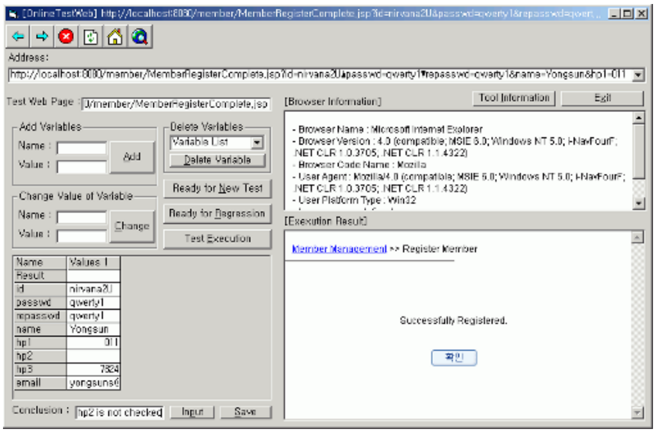
The message 5 is a test case and the message 10 is a result of the test case in Figure 1. Table 4 is an example of extracting test cases for integrated web page test from sequence diagram in Figure 1. IT01 is a test case to test the function that deletes the specified research material and IT02 is to examine if the system prevents the action when an unauthorized user is going to delete a material. IT03 is a test case to examine if the system announces error to user when a user is going to delete a nonexistent material.

4 Testing with OnlineTestWeb

In this section, OnlineTestWeb is proposed. OnlineTestWeb is a tool for testing web applications. OnlineTestWeb on-line executes web applications on web server or application server with extracted test case and display the result of execution. It archive and manage sequent test result to analyze and test web applications more efficiently.

This tool is made using Microsoft Visual Basic 6. Figure 2 and figure 3 are pictures of OnlineTestWeb. The left side of user interface of OnlineTestWeb is for setting test case. The address of server page which is test target is set on "Test Web Page" item and the names and values for testing is inputted in turn. The inputted names and values are displayed on spread sheet to offer simple view.

When the "Test Execution" button is pushed and executing test case has finished, the result of execution is displayed on the right two mini browsers. The upper browser is for displaying the information of browser and the lower browser is for displaying the result of execution of test case. The result of test is reviewed and conclusion of test is inputted on "Conclusion" item on the lower left corner of user interface of OnlineTestWeb.


[OnlineTestWeb] http://localhost:8080/member/MemberRegisterComplete.jsp?id=nirvana2U&passwd=qwerty1&repasswd=qwert...

Address:
http://localhost:8080/member/MemberRegisterComplete.jsp?id=nirvana2U&passwd=qwerty1&repasswd=qwerty1&name=Yongsun&hp1=011

Test Web Page : 0/member/MemberRegisterComplete.jsp

Name	Values 1
Result	
id	nirvana2U
passwd	qwerty1
repasswd	qwerty1
name	Yongsun
hp1	011
hp2	
hp3	7824
email	yongsuns@

Conclusion : hp2 is not checked

[Browser Information]

- Browser Name : Microsoft Internet Explorer
- Browser Version : 4.0 (compatible; MSIE 6.0; Windows NT 5.0; I-NavFourF; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
- Browser Code Name : Mozilla
- User Agent : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; I-NavFourF; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
- User Platform Type : Win32

[Execution Result]

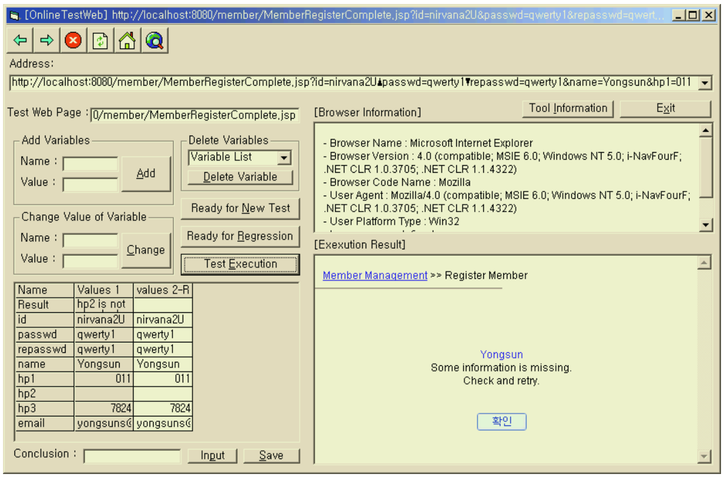
Member Management >> Register Member

Successfully Registered.

Fig. 2. OnlineTestWeb – Check Error

The “Ready for New Test” button is for preparing new test. If “Ready for New Test” button is pushed, the pre-executed test values are changed to gray color and the new column for new values is created.

If an error is discovered after testing, test target should be changed to correct the error and regression test should be executed with same environment. The “Ready for Regression” button is for preparing regression test. It the “Ready for Regression” button is pushed, the pre-executed test values are changed to gray color new column with same value is created on the right in spread sheet. The title of new column for regression test is created with the sign “-R” which indicates regression test.



http://localhost:8080/member/MemberRegisterComplete.jsp?id=nirvana2U&passwd=qwerty1&repasswd=qwerty1&name=Yongsun&hp1=011

Test Web Page : 0/member/MemberRegisterComplete.jsp

[Browser Information]

- Browser Name : Microsoft Internet Explorer
- Browser Version : 4.0 (compatible; MSIE 6.0; Windows NT 5.0; i-NavFourF; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
- Browser Code Name : Mozilla
- User Agent : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; i-NavFourF; .NET CLR 1.0.3705; .NET CLR 1.1.4322)
- User Platform Type : Win32

Name	Values 1	values 2-R
Result	hp2 is not	
id	nirvana2U	nirvana2U
passwd	qwerty1	qwerty1
repasswd	qwerty1	qwerty1
name	Yongsun	Yongsun
hp1	011	011
hp2		
hp3	7824	7824
email	yongsuns@	yongsuns@

[Exexution Result]

Member Management >> Register Member

Yongsun
Some information is missing.
Check and retry.

Fig. 3. OnlineTestWeb – Regression Test

Figure 2 is an example of executing a mutual web page test. In this case, hp2 which is a value for the middle number of cellular phone is omitted so error message is displayed. However the information of member is registered because the code for checking the completeness of inputted information in “MemberRegisterComplete.jsp” is omitted. The error of test case is identified and web application should be modified.

Figure 3 shows the result of regression test after identified error is modified and the data which is inputted on pre-executed test is deleted on database. The same test with figure 2 is executed. Because error code is modified, missing of some information is noticed and incorrect registration is not accomplished.

The test of web application is easily and efficiently accomplished using OnlineTestWeb and the technique of extracting test cases. The archived test results helped the test.

5 Comparison and Evaluation with Other Techniques

The test level is classified into three levels of single, mutual and integrated web page test, and the technique for the generation of test cases for each level is proposed in this paper. However test level is not classified for the generation of test cases in other techniques of related works. Test cases are generated by grouping flows of events of use cases for integrated web page test in RUP [3]. Test cases are generated by grouping control flows of source code and flows of web pages for all level tests in the technique of Wu, Offutt [4], so it is very complex to test web applications using the technique. Test cases are generated based on the relation model of web pages for integrated web page test in the technique of Ricca, Tonella [5].

The technique for testing script functions which are atomic units for performing logic in web applications is proposed in this paper, while it is not proposed in the technique of RUP and Ricca, Tonella.

Test data for both of normal and abnormal cases are used in this paper and RUP, while it is difficult to present test data because only the flows of logic and web pages are presented in the technique of Wu, Offutt, and test cases can be greatly increased according to the number of test data in the technique of Ricca, Tonella.

The number of test cases is the number of messages of sequence diagrams in this paper, while the number of test cases can be greatly increased because the combination of flows of events, control flows of source code or flows of web pages is used to generate test cases in other technique.

Table 5 shows the result of comparison of the techniques for the generation of test cases.

Table 5. Comparison of the techniques for the generation of test cases

Technique / Item	This paper	RUP	Wu, Offutt	Ricca, Tonella
Source of test cases	Sequence diagram	Use case model	Source code	Relation model of web pages
Test level	Single, mutual, integrated level	Integrated level	Integration of three levels	Integrated level
Test of script functions	O	X	O	X
The number of test cases	Few (The number of messages of sequence diagrams)	Many (The number of combinations of flows of events)	Many (The number of combinations of control flows of source code)	Medium (The number of combinations of flows of web pages)

O: supported X: not supported

6 Conclusion and Future Work

A test design technique based on the UML sequence diagram for developing web applications is proposed. A test of the web applications is composed of a single web page test, a mutual web page test and an integrated web page test. The test cases for a single web page test are generated from self-call messages and the test cases for a mutual web page test are generated from the messages between web pages. The test cases for an integrated web page test are generated from the messages which are sent to the system by an actor and received back from the system.

The flow of logic between web pages is easily understood, so test cases are easily generated because a sequence diagram shows a number of objects and the messages that are passed between these objects according to the time ordering of messages [9]. The notation of UML is frequently used in the analysis and design of web applications. Therefore, the technique of this paper will be more useful.

We plan to connect analysis and design with testing of web applications in order to test efficiently web applications.

References

[1] Abhijit Chaudhury, et al., *Web channels in E-Commerce*, Communications of the ACM, Jan. 2001.

[2] *Unified Modeling Language Specification Version 1.4*, OMG, September 2001.

[3] *The Rational Unified Process*, Rational Software Corporation (a wholly owned subsidiary of IBM), 2003.

[4] Ye Wu, Jeff Offutt, *Modeling and Testing Web-based Applications*, GMU ISE Technical ISE-TR-02-08, Nov. 2002.

[5] Filippo Ricca, Paolo Tonella, *Analysis and testing of Web applications*, Proceedings of the 23rd International Conference on Software Engineering, 2001.

[6] Mashito Hamba, Ryuichi Okakura, *HTML & Java Script Dictionary*, Youngjin.com, 2000.

[7] Jim Conallen, *Building Web Application with UML Second Edition*, Addison Wesley Longman, Inc., 2002.

[8] Roger S. Pressman, *Software Engineering: A Practitioner's Approach (5th Edition)*, McGraw-Hill, 2000.

[9] Grady Booch, et al., *The Unified Modeling Language User Guide*, Addison Wesley Longman, Inc., 1999.