Suggestions on screen transition diagram development for web system
And how to create effective test cases

Takashi Otomo
Canon IT Solutions Inc.
Tokyo, Japan
Otomo.takashi@canon-its.co.jp

1. Introduction

1.1 Background of the Research

Because of the inexpensive Internet broadband services, online shopping has been becoming common even among people who had not been very familiar with IT before. E-commerce system is now required to provide large amounts of information by using the latest web technologies with understandable and usable user interfaces. In enterprise system, a web system is expected to become the major architecture rather than stand-alone or traditional client-server systems.The evolution of web technologies has enabled more diversified platforms, flexible interface implementation, data exchange between various systems, and robust security architectures.

1.2 Current Problem

On the other hand, there are two problems in testing a web system. One of the problems is that the behavior of each screen in transitions has become more complex than before. The other is that test quality is dependent on a skill level of each test engineer.

The first problem is involved by sophisticated user interfaces, browser-dependent constraint, and the use of highly advanced security authentication. For example, in testing a web system, the characteristics of screen transition in response to each browser button must be carefully considered to prevent omission of test cases, because there are a number of factors such as requests from the web server, current login user, each session status in the client, security constraint and connections with other systems.

In the latter problem, technical knowledge, which is necessary for those tests, tends to be deposited as tacit knowledge inside of individual test engineers, because they don't have enough time to share knowledge. Shortage of sharing time is caused by several reasons, e.g. technical innovation speed, pace of advance in web technologies, and shorten period of system development.

Furthermore, although screen transitions are important test points, know-how of testing screen transition is left as tacit knowledge of expert engineers, and tools are seldom used.

1.3 Suggestion to Solution

To resolve two problems, development of screen transtion diagrams is suggested

Basic concept: Provides a method to describe screen transition diagrams, under consideration of characteristics of web systems.

1 Notation of screen transition diagrams, that have information of screen states and data transitions, is suggested. The information is difficult to read from conventional screen transition diagrams or specifications directly. Test cases under consideration of screen states are derived from the diagrams. In addition, by defining a method for easy-to-understand diagrams, efficiency of test case reviews is expected to be improved.

2 By creating a state transition diagram with priority from prepared screen transition diagrams, only effective test cases are selected. This enables the effective selection of test cases

As a result, the unevenness of test cases caused by the difference of each engineer's skills can be reduced. The methods are proposed to prepare effective test cases for screen transitions in web systems.

2. New Test Design Process

2.1 The General Test Design Process of Screen Transitions

A typical test process (upper side) to design screen transition test cases together with the proposed process (lower side) are described in Figure 1.

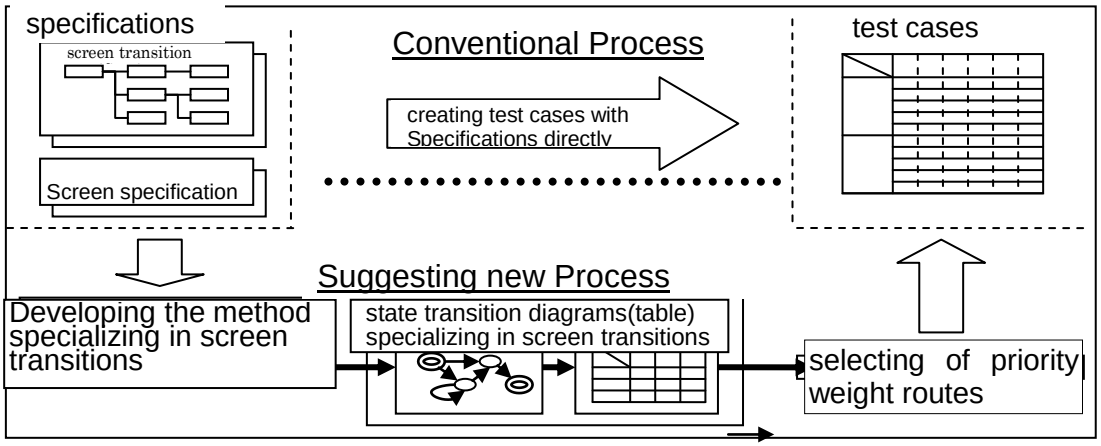


Figure 1. Conventional Process and the New Process

Specification documents such as screen transition diagrams, screen specifications and functional specifications are used as materials in both processes. However, in the flow of the conventional test design process, test designers will prepare test cases, referring to those documents as required.

In conventional processes, there are several problems to be solved such as:

1. Not all specifications are prepared with a view to screen transitions.
2. The quality of the task of detecting conditions for screen transitions from specifications depends on the skills of each test designer.
3. It is difficult for a third person to evaluate the validity and completeness of test cases

In the testing of complex screen transitions, not only preparing test cases from specifications but also evaluating them and making sure of their coverage of the specifications can be very difficult.

2.2 Proposed Test Design Process for Screen Transitions

The author therefore proposes the test design process shown at the bottom of Figure 1. In the process, test designers first develop screen transition diagrams from specification documents and then, based on the diagrams, create test cases.

In this stage, the task volume is increased. However, organizing and visualizing the necessary information to screen transition tests with those intermediate outputs realize the sufficient test coverage, and help to measure the risk of each test case.

In this study, a new screen transition diagram is denominated the Extensible Screen Transition Diagram (XSTD) and Extensible Screen Transition Table (XSTT) to conduct this test design process.

“Extensible” here suggests that the definition of the diagram's notational system can be extended in accordance with the characteristics of each system in order to adapt to various forms of web systems.

Developing XSTD includes the following steps;

1. By developing an XSTD, information regarding screen transitions in each specification document can be organized.
   It enables uniform management by visualizing information requiring reference to multiple specification documents. It is also effective for third-party reviews.
2. By developing an XSTD (XSTT), comprehensive transition routes are automatically derived.
   The omission of possible transition routes can be avoided. It can also be used for the evaluation of comprehensiveness.
3. Important transition routes can be discovered by weighting each transition with a priority.
   The quality of test cases can be improved.

The specific formats of XSTD will be explained in the next chapter.

3. Extensible Screen Transition Diagram (XSTD)

3.1 The Notation of the XSTD

Based on the steps stated in Chapter 2, referring to the conventional notation for state transition

diagrams, the author added and extended the developing methods inherent in the XSTD.

When state transition diagrams have been developed for screen transitions, it is possible to think of “state” as “screen”, and “transition” as “screen transition”. Based on this idea, by defining simple developing methods for elements that constitute the screen transition diagram with variations, complex screen transitions can be expressed in understand forms.

The example of the developing XSTD is shown in Figure 2.

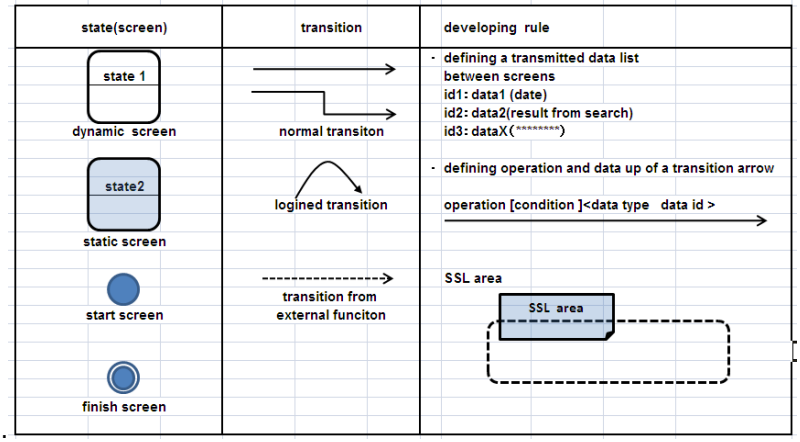


Figure 2. Example of Developing XSTD

3.2 Priority of Test Design

Next, the roles of priority that XSTD assumes and its methods are explained.
Because the conventional screen transition diagram does not have indexes to show the importance of each transition, the combinations of important routes are difficult to be derived. On the other hand, XSTD enables the selection of high priority transitions from among all the extensive number of routes by attaching priority information to each transition.

Moreover, by setting priority from various viewpoints such as test strategies and the characteristics of test targets, test case designed specific to each purpose becomes possible. Test case design is described in “3.4 Test case design". In this research, priority and its weight are set as shown in Table 1.

Table 1. Priority Weight

weight symble	detail	weight
S	transiton in SSL area	1
B	transition by browser back button	1
C	transition with creating data	1
D	transition with deleting data	3
E	transition to finish	1
U	transition with update data	1

3.3 The Notation of the XSTT

The procedure of developing XSTT from the XSTD, which is completed with transition priority added, is explained in this section. XSTT has rows as “from screen” and columns as “to screen”. (See Table2) First, enter the screen states before transition into the first column and screen states after transition into the first row, and mark the start and end positions of the transition. Into the cell corresponding to the states before and after transition, enter the attributes of the transition in the order as follows. If there are multiple transitions from a screen to another screen, each transition must be described in a separate row.

1. Weight symbols
2. Transition No.
3. Conditions

Table 2. Notation of XSTT

	exception repeat→				exception
	finish screen				O
start screen	from ↓ / to →	screen No.1	screen No.2	screen No.3	screen No.4
O	screen No.1		datatype transitionNo. Ex.) CCU1		
	screen No.2			datatype transitionNo. Ex.) 2	
	screen No.3				datatype transitionNo.
	screen No.3		datatype transitionNo.		

Add the description about the transition in response to the Back button of the browser, which is not included in the XSTD, only when the transition has a high weight from the points shown in Table 2. Enter “Rep” into cells for screens of high traffic to which multiple screens transit in order to exclude them from the repeat count.

3.4 Test Case Design

In this study, two methods of creating test cases are suggested for test purpose.

1. Creating test cases under consideration of coverage
   Test cases are created for all possible transitions, regarding each marked cell of the table as a single test case. This forms a comprehensive test case design covering all transitions.
2. Selecting for high priority routes
   Route is a part of sequence of transitions from a start screen to a finish screen. High priority routes are derived for the composite test cases involving multiple transitions. However, be sure to follow the next steps to avoid the proliferation and uneven selection of test cases.
   (i) The decision of the maximum number of transitions.
   The number of screen transitions is limited in order to prevent the proliferation of test cases. It is determined minimum transition numbers.
   (ii) The decision of the repeat count.
   Decision how many times the same transition route can be taken. The count should be at least two, taking the back maximum transition into consideration. However, define the screens of high traffic to which multiple screens transit as “Rep” and exclude them because those routes can be taken repeatedly.
   (iii) The calculation of routes from each screen as a start position.
   To avoid the unevenness of test cases, all routes, that start from each screen and contain the maximum number of transitions determined in step (i),are listed. However, exclude the exit and error screens.
   (iv) The calculation of the total weight.
   Sum the points previously given to each transition that makes up the route calculated in step (iii).
   (v) Selection of top weight test cases among the routes.
   Test cases were selected among with the routes of each screen with top weight as calculated in steps (iii) and (iv). Threshold rank order was determined by taking into the possible number of test cases and the distribution of the weight of each screen route.

4. Experiments

4.1 Preparations

4.1.1 Preparation of the Test Candidate

In order to verify the effectiveness of XSTD, experiments were conducted using a virtual site for lodging reservations (hereinafter the “Virtual Site”). The Virtual Site has two main functions: a search function by the conditions such as “date”, “keyword” and “destination”; and a reservation function to reserve lodging.

In order to prove the advantages of XSTD that have been claimed in this paper, the following items were

put forward to be solved.

1.Looping procedures, regarded as a weak point of screen transition testing
2.The branching of screen transitions by selective conditions

4.1.2 Bugs to be detected

Next, the Virtual Site was assumed to contain the following bugs.

1. Proceed from the search results list screen to the payment selection screen. Then back to the list screen and after selecting a different lodging, proceed to the payment selection screen again. There, the old lodging information was displayed.
2. From the reservation confirmation screen, return to the payment selection screen with the Back button of the browser and try to register a new input. Then, a system error occurs.
3. On the search results list screen, press the narrow Search button more than twice consecutively. Then, a system error occurs.

4.2 Experiments

4.2.1 Developing XSTD and XSTT

The XSTD is notated in Figure3, and XSTT is notated in Table 3.A part of XSTD is magnified. And a part of XSTT is notated with an economy of part.

First, the XSTD and XSTT were developed from the specification of test candidate. The specification is consisted of 50 screen states and 116 transitions.

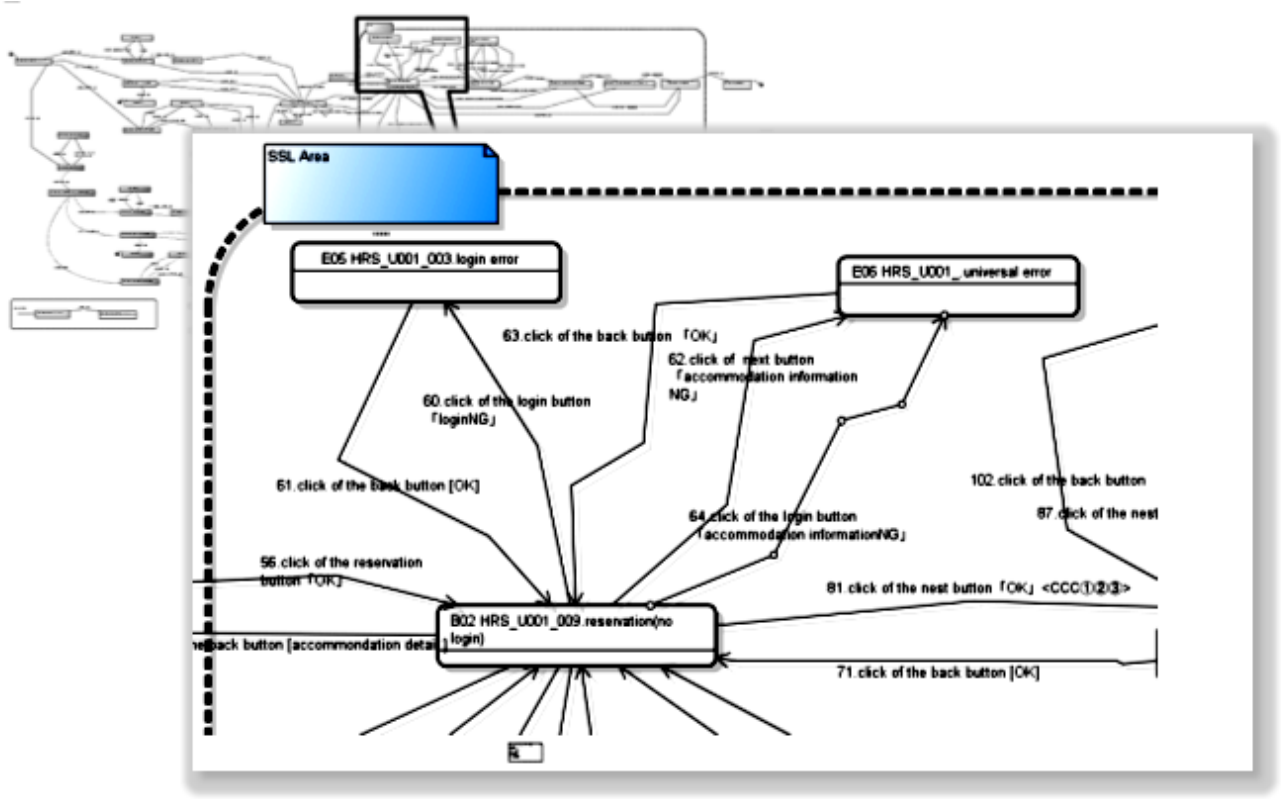
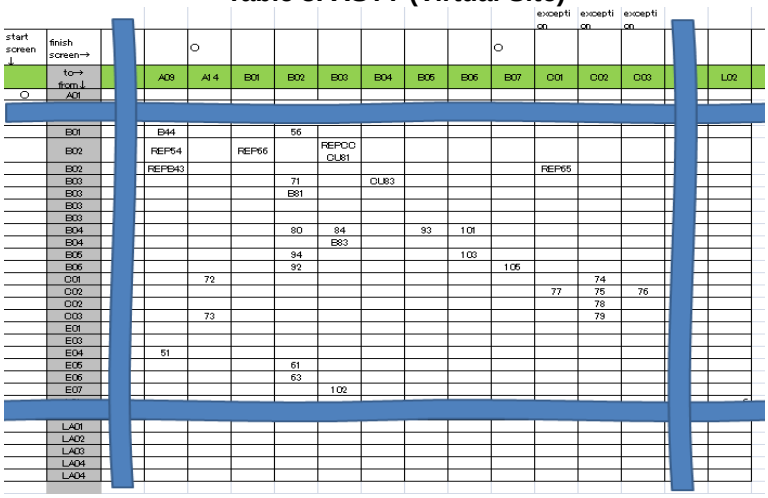


Figure3. XSTD (Virtual Site) enlarged view

Table 3. XSTT (Virtual Site)



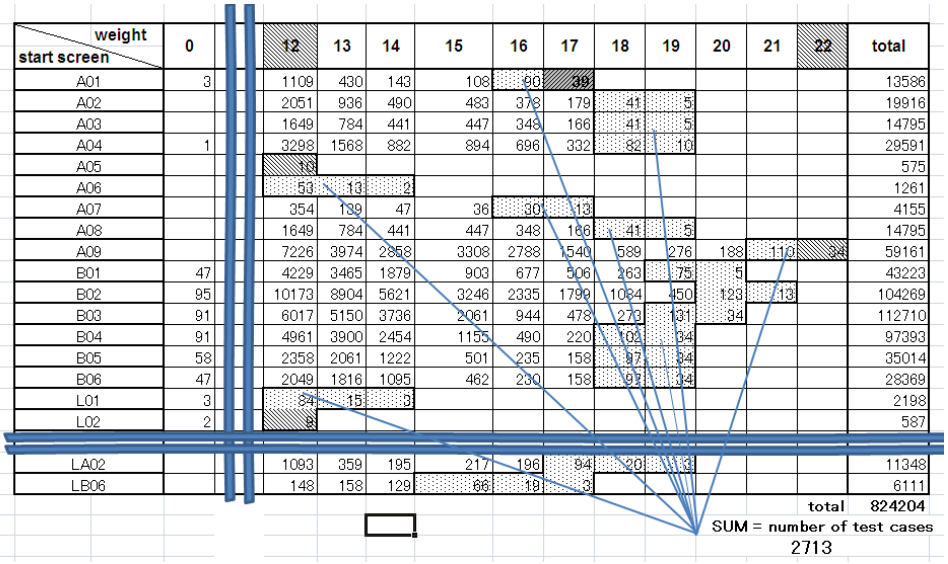
4.2.2 Creation of Test Cases

Second, test cases are created from XSTT.

1. Creating test cases under consideration of coverage
   116 cases for transitions described in cells were created.
2. Selecting for high priority routes
   The executed procedure and selected test cases in this experiment are as follows (based on chapter 3.4.2).

(i) Number of transition: It is 8 transitions. It is shortest routes from the start to the end positions.
(ii) Number of repeats: 5 times. The result was the same result for step (iii), if the repeat count was set as three.
(iii) Thrash out all routes: Of 34 screens excluding the end, error and calendar screens, 824,204 cases were calculated except 0 weight.
(iv) Calculation weight: The maximum weight was 22 (the start positions: screens A09 and LA09), and the minimum weight was 12 (the start positions: screens A05, LA02, and LA05).
Selction of test cases: Because maximum top weight routes number of each screen is 39, in order to be included second top weight routes, the threshold is determined 40th by weight order (See Table4).The routes into the top 40th were selected for each screen. It was amounted to 2,713 cases with weight ranging from 12 to 22.

Table 4. Distribution of Routes and Weight



start screen \ weight	0	12	13	14	15	16	17	18	19	20	21	22	total
A01	3	1109	430	143	108	90	39						13586
A02		2051	936	490	483	378	179	41	5				19916
A03		1649	784	441	447	348	166	41	5				14795
A04	1	3298	1568	882	894	696	332	82	10				29591
A05		19											575
A06		53	13	2									1261
A07		354	139	47	36	30	13						4155
A08		1649	784	441	447	348	166	41	5				14795
A09		7226	3974	2858	3308	2788	1540	589	276	188	110	34	59161
B01	47	4229	3465	1879	903	677	506	263	75	5			43223
B02	95	10173	8904	5621	3246	2335	1799	1084	450	123	13		104269
B03	91	6017	5150	3736	2061	944	478	273	131	34			112710
B04	91	4961	3900	2454	1155	490	220	102	34				97393
B05	58	2358	2061	1222	501	235	158	97	34				35014
B06	47	2049	1816	1095	462	230	158	93	34				28369
L01	3	87	15	3									2198
L02	2	8											587
LA02		1093	359	195	217	196	94	20	13				11348
LB06		148	158	129	66	19	3						6111
												total	824204

SUM = number of test cases
2713

4.2.3 The Verification of Results

Selected test cases were verified by whether the embedded bugs were detected or not.

1. Proceed from the search results list screen to the payment selection screen. Then back to the list screen and after selecting a different lodging, proceed to the payment selection screen again. There, the old lodging information was displayed.
   Result: detected
   Weight: 14 , route: starting from screen LA07 to 32 > 42 > Rep CCU46 > Rep CCCU85 > 86 > Rep 58 > Rep CCU46 > Rep CCCU85
2. From the reservation confirmation screen, return to the payment selection screen with the Back button of the browser and try to register a new input. Then, a system error occurs.
   Result: detected
   Weight: 15 , route: starting from screen LB06 to Return 104 > CU104 > Return 104 > 91 > Rep CCCU85 > Return 82 > Rep CCCU81 > CU83
3. On the search results list screen, press the Narrow Search button more than twice consecutively. Then, a system error occurs.
   Result: not detected.
   The test cases were then recalculated by reducing the maximum number of transitions of step (i) to seven routes for eight screens. The calculation of routes from each screen as a start position of step (iii) found 214,183 cases and the creation of test cases from the top weight routes of step (v) found 2,035 cases from 4 weight up to 19 weight. As a result, the above-mentioned third bug was also detected at the second repeat in the test case of 7 weights.

5. Consideration

The XSTD contains the considerable amount of information such as detailed transition conditions and data transfer and thus enables an overall understanding of the screen transitions at a glance. However, because of this, it can become an intricate diagram with a lot of information. In the experiments using the Virtual Site, a problem was revealed that arrows representing transitions were interwoven and the development of the diagram required a considerable length of time. In this research, Microsoft Excel 2003 and the UML modeling tool, JUDE/Community, were used as preparation tools. While JUDE offers flexibility in developing, Excel has the advantage of high usability that has made it attractive to many users. Tools can be selected in accordance with each development environment because the XSTD developing method does not depend on tools. Therefore it can automate process that is converted XSTD into XSTT.

In the XSTD, when one screen can be reached by two routes, for example, when the transitions from the “screen of the search results” screen by using either the station name or the hotel name as the search condition, the transition is represented with two lines, but it transits same screen. In such a case, the transitions including data transferred from one screen to another are different even if the destination is the same, and thus they are depicted as different transitions on the diagram. In this way, coverage of the test cases between two screens was improved, and omissions were prevented.

The total number of routes from the start to the end of the Virtual Site used in this research was estimated at approximately 820,000 cases when taking back and loops into account. This number cannot be regarded as realistic for test cases in the actual testing. In conventional methods, test cases are selected from among elements that can determine the number of combinations between two or three screen. However, the present web systems often retain data for more than a few screens in sessions from one screen to another. Such bugs as occur in session transfer can be discovered with test cases extracted as high weight routes.

For more effective selection of test cases for bugs that are likely to occur in screen transition, the method was suggested in which weight was given to each screen up to the next eight screens to transit, and the routes with top weight were selected. As a result, the first bug was detected by giving priority weight for data creation, then the second bug by including the transition in response to the browser's Back button to high weight transitions when developing the XSTT. These two bugs were caused by invalid data in sessions due to transitions and session expiration by the use of the browser's own Back button. The detection of such bugs, regarded as difficult to detect in web systems, proved the validity of this suggestion. Moreover, but these bugs were detected in 2,713 test cases selected from among about 820,000 test cases, and thus the suggestion was also confirmed to be effective. On the other hand, the third bug was caused by a memory leak after searches repeatedly within the same screen and could not be detected

because no transitions occurred in the process and consequently the weight remained low. Such a screen, with a considerable amount of data and heavy traffic from other screens, is obviously more important than others. By defining the weight of a screen itself and giving weight to it, the third bug will become possible to detect in the future.

The next study challenge is to define the finer weight of transitions in accordance with the system characteristic. For example, to control transition weight from the viewpoint of data transfer, the importance of data used commonly in many screens and of those used within a specific screen is different and the weight must be changed accordingly. As stated above, accuracy improvement of priority routes is achieved by changing weight rules.

6. Conclusions

This research proposed a method and procedure using XSTD（XSTT）to prepare effective and fully comprehensive test cases for the testing of screen transition of web systems. XSTD has the information with a view to screen transitions in specifications. And it has weight information on priority of screen transitions. In the process, the following facts were confirmed: it is possible to represent screen transitions in the present web systems by preparing XSTD in accordance with the rules of the proposed development method; and the developed XSTD is not varied between individuals**.**

In the tests between two sequential screens that required complete coverage, the necessary information such as transition conditions was able to be expressed much in detail in comparison with conventional methods. And no test case omission was found.

In the tests of completed screen transitions, designing tests for all the possible cases cannot be regarded as realistic. In conventional methods, test cases are selected from among elements that can determine the number of combinations between two or three items. However, the present web systems often retain data for more than a few screens in sessions from one screen to another. Such bugs as occur in session transfer can be discovered with test cases selected as high priority routes.

From this, in order to prepare the screen transition test cases with no omissions the XSTD is more effective than conventional methods by which the test cases are derived directly from specification documents such as screen transition diagrams, screen specifications and functional specifications.

The scope of tests using XSTD has become clear through experiments and it is useful when examining test targets and bugs to be discovered in the test design phase.

References

[1] Dorothy Graham, Erik van Veenendaal,Isabel Evans,Rex Black, ISTQB Certificat,Foundations of Software Testing, Cengage Learning Business Press (December 19, 2006)

[2] Juichi Takahashi, *Tishiki zero kara manabu software test* [Learned software testing from zero],(Japan:Shoueisha,2005)

[3] Roger S. Pressman ,Software engineering : a practitioner's approach, McGraw-Hill Science/Engineering/Math; (April 2, 2004)

[4] Mealy Machine/Moore Machine/ finite automaton/ State Transition Diagram/ State Transition Table, (Wikipedia,The Free Encyclopedia）

[5] Rex Black , Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional, Wiley; (2007/2/20)

[6] Boris Beizer, Software Testing Techniques, Intl Thomson Computer Pr (T); (June 1990)

[7] Cem Kaner,Jack Falk,Hung Q. Nguyen, Testing Computer Software, 2nd Edition, Wiley; (April 12, 1999)

[8] IPA/Hattyusyaview kentoiinkai,*hattyusyaview guideline*[build-to-order view guideline],IPA,(2008)