ISSN 0117-3375

ASIA LIFE SCIENCES

The Asian International Journal of Life Sciences

Beyond Excellence ©

SUPPLEMENT 11 **JULY 2015**

Bioscience and Medical Research: Contributions 2

The Convergent Research Society Among Humanities, Sociology, Science & Technology

e-mails: asialifesciences@yahoo.com wsmgruezo@gmail.com

http://journals.uplb.edu.ph/index.php/ALS

Printed in the Philippines

Contents

ASIA LIFE SCIENCES Supplement 11: 277-288, 2015
The Asian International Journal of Life Sciences

Test case extraction process for validation of real time transactions in financial value added network for trusworthy customer's credit card

HYUN SEUNG SON[1*] and R. YOUNG CHUL KIM[1**]

In most Value Added Network (VAN) companies in Korea, real financial transactions are occurring on a consecutive real time serialization. But until now, it just validates their systems with checklists based on test expert's experience. We need better way for validating such consecutive real-time transaction system due to practical situations like the dynamic transaction chain across multiple financial networks or systems. In this paper, we suggest how to extract test cases for diverse practical situations. This proposed process generates a test case with a specific mechanism based on UML and Event/Condition/Action (ECA) rules. Through analyzing transactions on consecutive and real time transaction chains, we can effectively model UML diagrams with transactions, and map UML with ECA rules, which creates an ECA-decision table. A test scenario is generated with this table. That is, this approach can only generate test cases for the consecutive transaction chains. We applied these examples to validate real time transaction of Kovan Company, Ltd., a financial VAN company in Korea.

Keywords: real time transaction, modeling, Event Condition Action (ECA), test case generation, Value Added Network (VAN)

[1]SE Laboratory, Department of Computer and Information Communication, Hongik University, Sejong Campus, 339-701, Korea

[*]*e*-mail: son@selab.hongik.ac.kr

[**]Corresponding author: *e*-mail - bob@selab.hongik.ac.kr

Received 03 February 2015; Accepted 21 April 2015

Printed in the Philippines

INTRODUCTION

VAN companies are intermediates that connect affiliates using settlement systems with credit companies and other financial institutions. As an example, a VAN may connect a given credit card company to a credit card terminal by assigning in the settlement of a customer transaction (Hwang 2001, S.G. Kim 1990). Credit card settlement systems are proceeded by various settlement models such as transaction approvals via telephone wires or terminals, POS systems, and HOST servers (Lee 1988, Kim 1988, G.J. Kim 1990).

Based on these various settlement models, VAN companies' systems have become more complex. Also the structure requires the acceptance of all requests from affiliates and financial institutions (Kim 2010), resulting in frequent maintenance and repairs of the system. As the number of system change increases, there are more tests on such systems. As for the testing by VAN companies, developers used to play a role of such testers. While a test case shall be provided in consideration of the relations between new requirements and the existing system, there is no systemic way for the preparation of test cases. Reliability of testing depends on the different maturity level with developer's capacity and experiences (Wang 2011, Zheng 2009, Singla 2011). Therefore, such a test case depends a lot on the tester's capacity.

This paper suggests a test case generation model to verify real-time transactions that occur in the VAN (Value Added Network) environment. The suggested process generates a test case by applying UML modeling and ECA (Event, Condition Action) rules. In order to generate a test case, real time transactions are analyzed and modeled UML and ECA diagrams, then generated test scenarios. With the test scenarios, we can generate test cases.

RELATED WORKS

The Cause-Effect Graphing technique provides systemic methods to develop a statement prepared in a natural language into a decision table (Nursimulu 1995). In the cause-effect graphing technique, it is selected a set of test cases in consideration of causes that have logical relations with effects for a test. This test can take place in the following order: (1) Every requirement is identified; (2) The requirements are analyzed for the identification of every cause and effect to assign a unique number to each cause-effect set; (3) Based on the analysis of requirements, a graph connected causes with effects is provided; (4) The graph is converted into a decision table, and (5) Each row in the decision table is selected as a test case for testing.

Decision Table Testing is to describe all movements related to decisions, conditions, and processes required in the process and to prepare for a decision table that displays a movement occurring based on the combination of each decision and condition, which is an effective technique to find errors embedded in the materialization or statement (Beizer 1990). Each row of the table consisting of combinations between the remaining decisions and actions are selected as a test case.

Finite-State Testing is to test the models of finite states, that is, behaviors of a system (Yoo 2009). Input and output of every state are identified. The state table includes input combinations of all states and checks whether it can be reached or

not before testing. In the finite-state testing stage, a state graph is prepared and converted a state table. Only those that can reach the state based on one specific state are selected like test scenarios (or paths) for testing.

Test case extraction process for real-time transactions. Figure 1 shows a process of generating a test case for real-time transaction chains. This involves modeling based on the path, action and conditions of consecutive transactions acquired from initial analysis. A test case scenario based on the use case modeling is represented through a sequence diagram. This diagram generates a decision table to determine the Event, Condition, and Action (ECA) and the resulting ECA diagram.

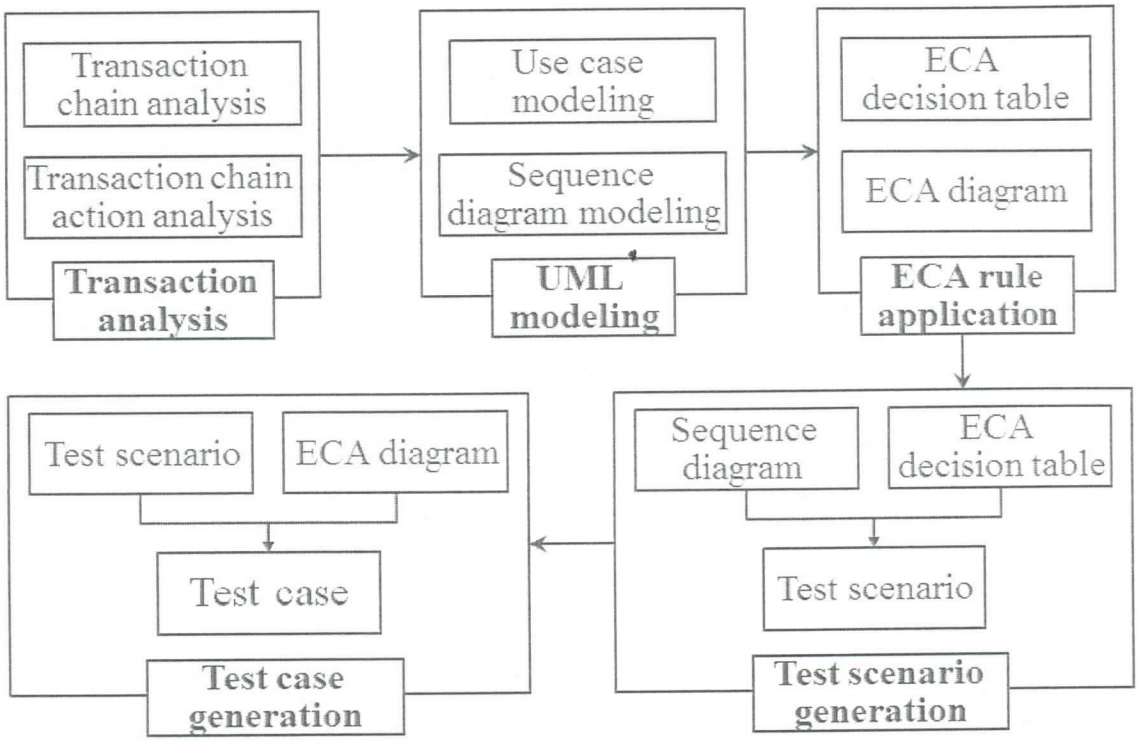


Figure 1. Process to generate test cases for real-time transactions.

Figure 2 illustrates the process to analyze a simple transaction. First, unit transactions, T1 (ACTOR), T2 (A) and T3 (B) are defined while a unit transaction chain is organized. The transaction is then analyzed based on dynamic diagram modeling of the transaction chains. In order for T1 to be committed, the T2 transaction must also be committed. Likewise, in order for the T2 transaction to be committed, the T3 transaction should be committed as well.

Figure 3 shows the UML modeling of a transaction in Figure 2. A sequence diagram generated from the UML modeling can be expressed using ECA rules. The objects in the sequence diagram are mapped as a unit transaction with ECA rules in the ECA decision table. Through mapping Event/Condition/Action on messages between objects, and also assigning an identification number to them, this information can be applied to the ECA decision table, respectively.

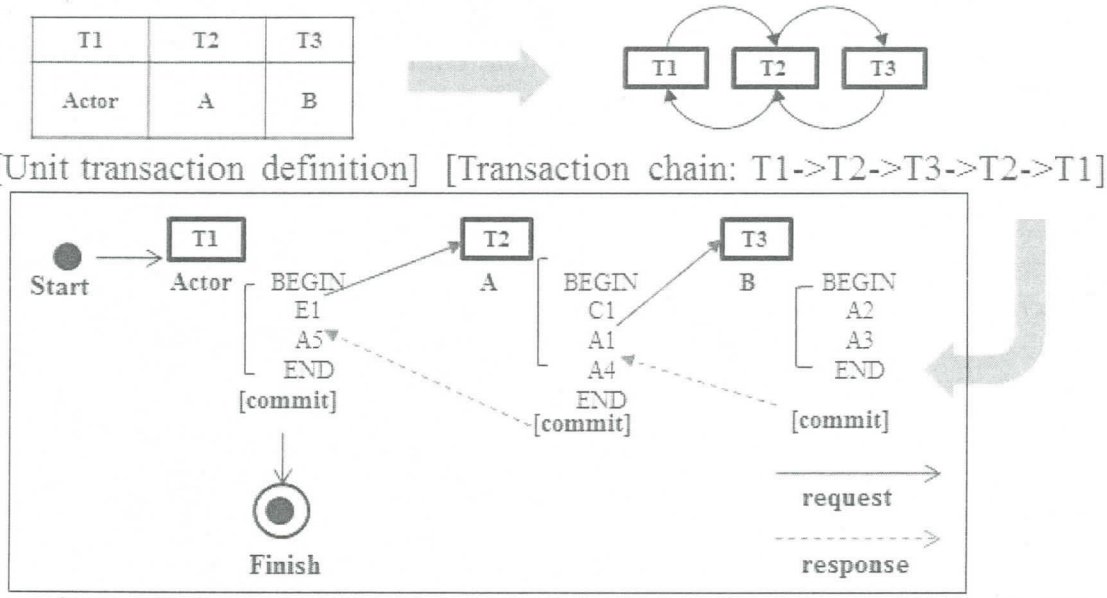


Figure 2. Analysis of consecutive real-time transactions.

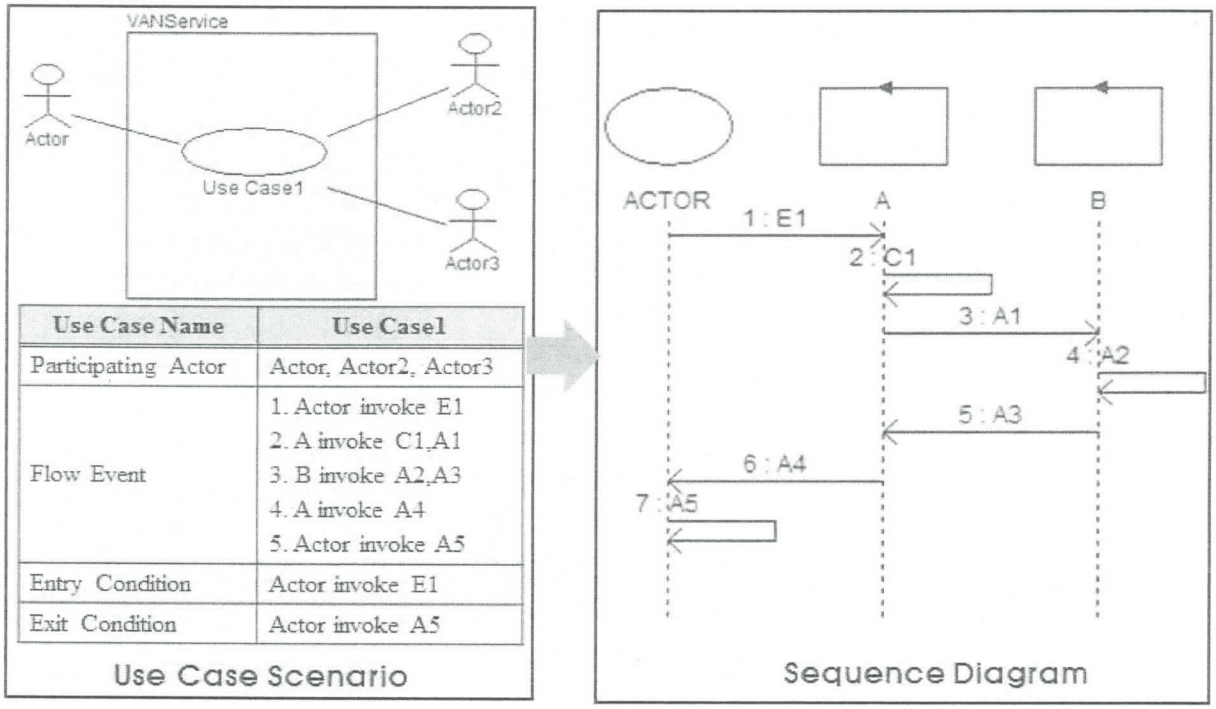


Figure 3. Mapping UML modeling with an ECA mechanism.

Table 1 shows an ECA table with the expression of a sequence diagram. The Actor's message numbers, 1(E1) and 7(A5) are applied to the Event and Action of the unit transaction for the Actor. Next, A's message numbers, 2(C1), 3(A1), and 6(A4) are applied to the Condition and Action of the unit transaction A. B's message numbers, 4(A2) and 5(A3) are then applied to the Action of the unit transaction for B.

Table 1. ECA Decision Table for consecutive real-time transactions.

Transaction	EVENT	Condition	Action
Actor	1	-	7
A	-	2	3,6
B	-	-	4,5

Figure 4 represents an ECA transaction diagram. Table 1 reflects on the modeling of Figure 4.

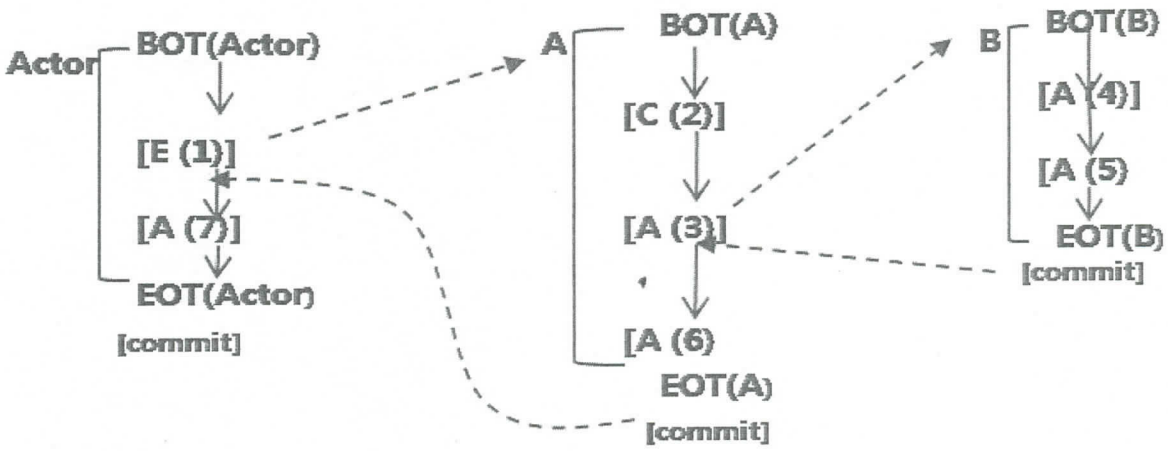


Figure 4. ECA diagram.

Table 2 is a test scenario based on the data obtained from the sequence diagram in Figure 3. Unit transactions of the decision table are applied to the transactions in Table 2, while the Event, Condition, and Action parameters of each unit transaction are applied to the test scenario. Messages pertaining to Event, Condition, and Action in the ECA decision table are also represented from the sequence diagram as part of the test scenario. Figure 5 shows the generated test cases based on a combination of the ECA transaction diagram and the test scenario presented in Table 2. The ECA based Test Cases consist of the subject transaction, test case ID, conditions, action, and expected results.

Table 2. Test scenario of consecutive real-time transactions.

Transaction	Test scenario
Actor	1- E1 is committed. A commits A4. 7- A5 committed (Actor commit)
A	Actor commits E1 (condition for starting) 2-C1 committed. 3-A1 committed. B commits A3. 6-A4is committed (A commit)

B	A commits A1(condition for starting) 4-A2 committed. 5-A3 committed (B commit)

Subject transaction	Test case	Conditions for starting	Action	Expected result
Actor	TC1	-	E1	A begin
Actor	TC2	-	A5	Actor commit
A	TC3	E1	C1	A1
A	TC4	C1	A1	A2
A	TC5	B commit	A4 execution	A Commit
B	TC6	A1	A2	A3
B	TC7	A2	A3	B commit

Figure. 5. ECA-based test case.

Real examples for test case generation. As a case study, we examine the process of *'point card'* transactions that occur in a sample VAN environment. These transactions involves point view, point collection, point use, and point cancellation. For a *point card* transaction, the application of the proposed method requires information from both the test cards and the test affiliates. Once this is obtained hypothetical test card numbers and affiliate numbers, it is set up. Next, a real test case is generated. It should note that test affiliates are defined as the mid-sized shops that use POS equipment. Also for the goal of this study, a test case is generated for 'point view'.

Figure 6 shows the transactions for 'point view' from those generated by the affiliates. It is important to note that Point transactions normally consist of five unit transactions such as T2, TF, TM, TC, and TP. Transaction-Shop (T2, that is, mid-sized shop) transaction is one that occurs at an affiliate. 'Point view' information is fed while a request for the 'point' transaction is made. The transaction is then completed as the results are confirmed through a response.

A Transaction-FEP (TF) transaction occurs between EFP servers. A TF transaction involves descrambling of the data received, data transmission from the main system, data encoding, and data transmission from affiliates, and data receipt. The condition for starting the action is a transaction request by the TS. A Transaction-Main system (TM) transaction begins when the TF transaction transmits data. Essentially, the TM transaction asks for 'point view' information after checking the data from user's cards, its affiliates, and only advanced conditions for transaction refusal. Once the transaction is completed as a response and received by the card company, the entire transaction information is then saved, and the response data is transmitted to the FEP server.

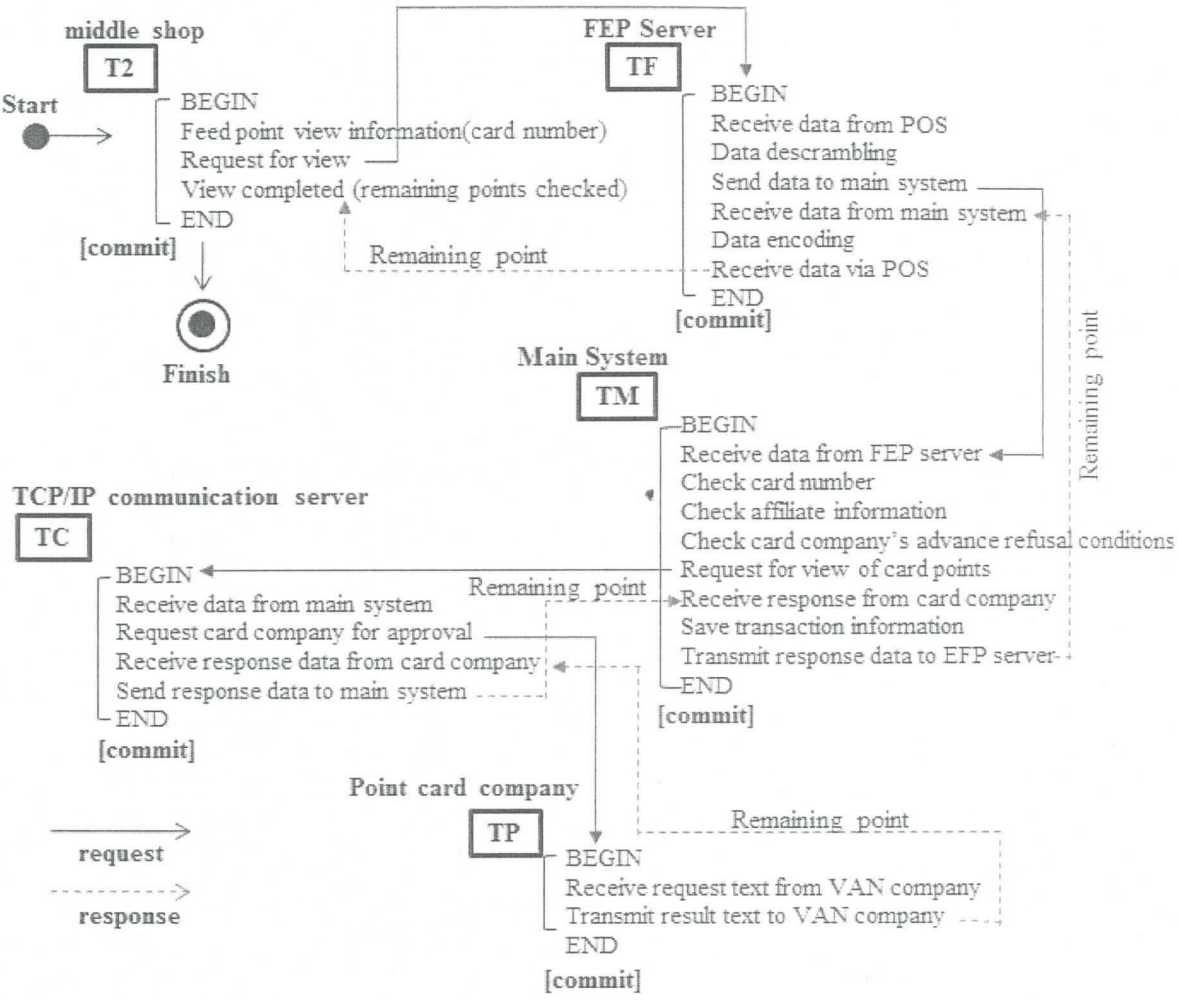


Figure 6. Dynamic transactions analysis for 'point view'.

When data are received from the main system, a request to the point card company is made. Once a response is received and sent to the main system, the transaction is completed. A Transaction-Point Issuer (TP) refers to a transaction made by a point card company. First, a *point card* company receives results from the VAN Company. Once the results are transmitted, the transaction is completed. The entire transaction process can therefore be represented as the transaction chain TS->TF->TM->TC->TP->TC->TM->TF->TS. Once these steps have been followed, the transaction is completed.

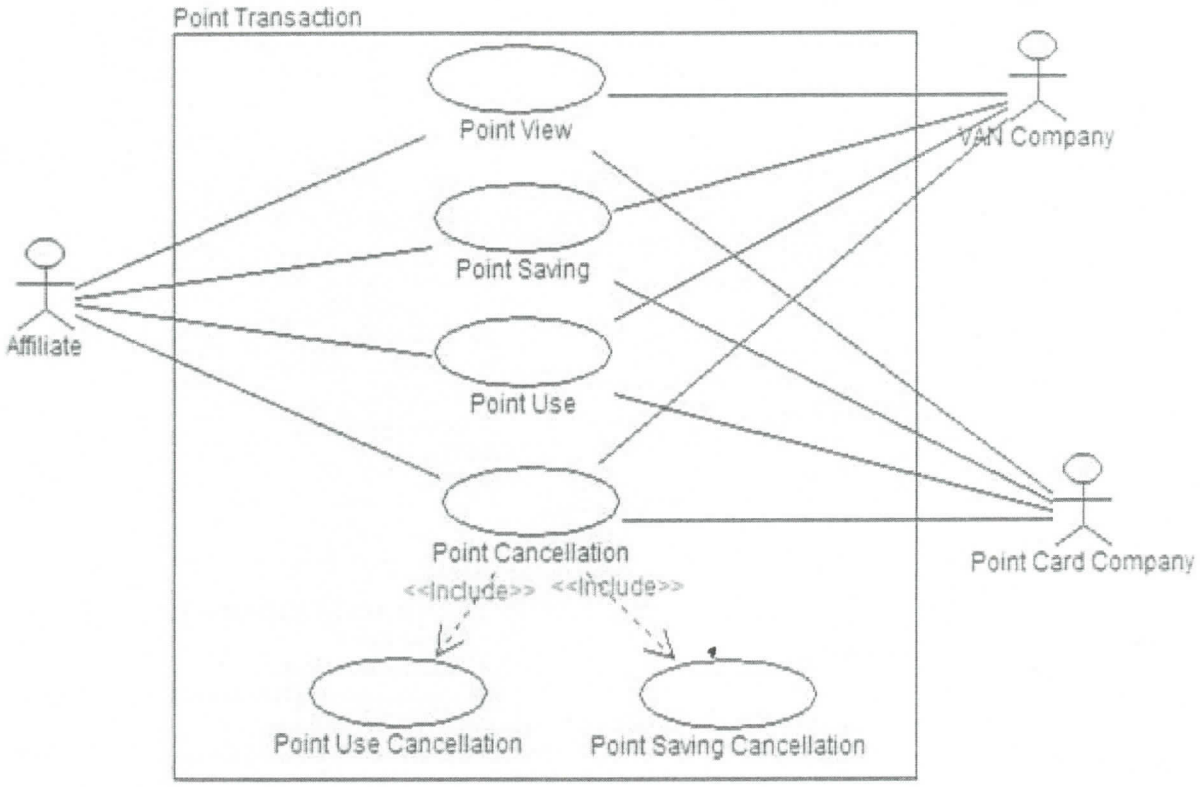


Figure 7. 'Point' transaction's use case.

Figure 7 shows a diagram of a 'point view' transaction which indicates 'point saving', 'point use' and 'cancellation', which are represented an entire 'point' transaction.

Table 3 provides a sample scenario for a user's 'point view' transaction.

Table 3. Use case scenario for ' point view'.

Use Case Name	Point view
Participating Actor	Affiliate, VAN company, point card company
Flow Event	1. Point view is requested as a test affiliate uses the test card number. 2. A VAN company checks the requested information from the affiliate and card then sends out a request for 'point view' to the point card company. 3. Point card company delivers the remaining points received initially from the VAN company back to the VAN company. 4. The VAN company saves the transaction and sends a response to the affiliate. 5. After receiving the transaction information, the affiliate checks the points, thereby completing the transaction.
Entry Condition	A test affiliate asks to view a user's points through a test card.
Exit Condition	A user's remaining points is checked and confirmed.

Figure 8 shows transactions subject to 'point view' as a sequence diagram. This information can also be converted into an ECA decision table, and used for ECA modeling. The corresponding numbers of the messages in the sequence diagram, and the E (event), C (condition), and A (Action) notations are organized as well.

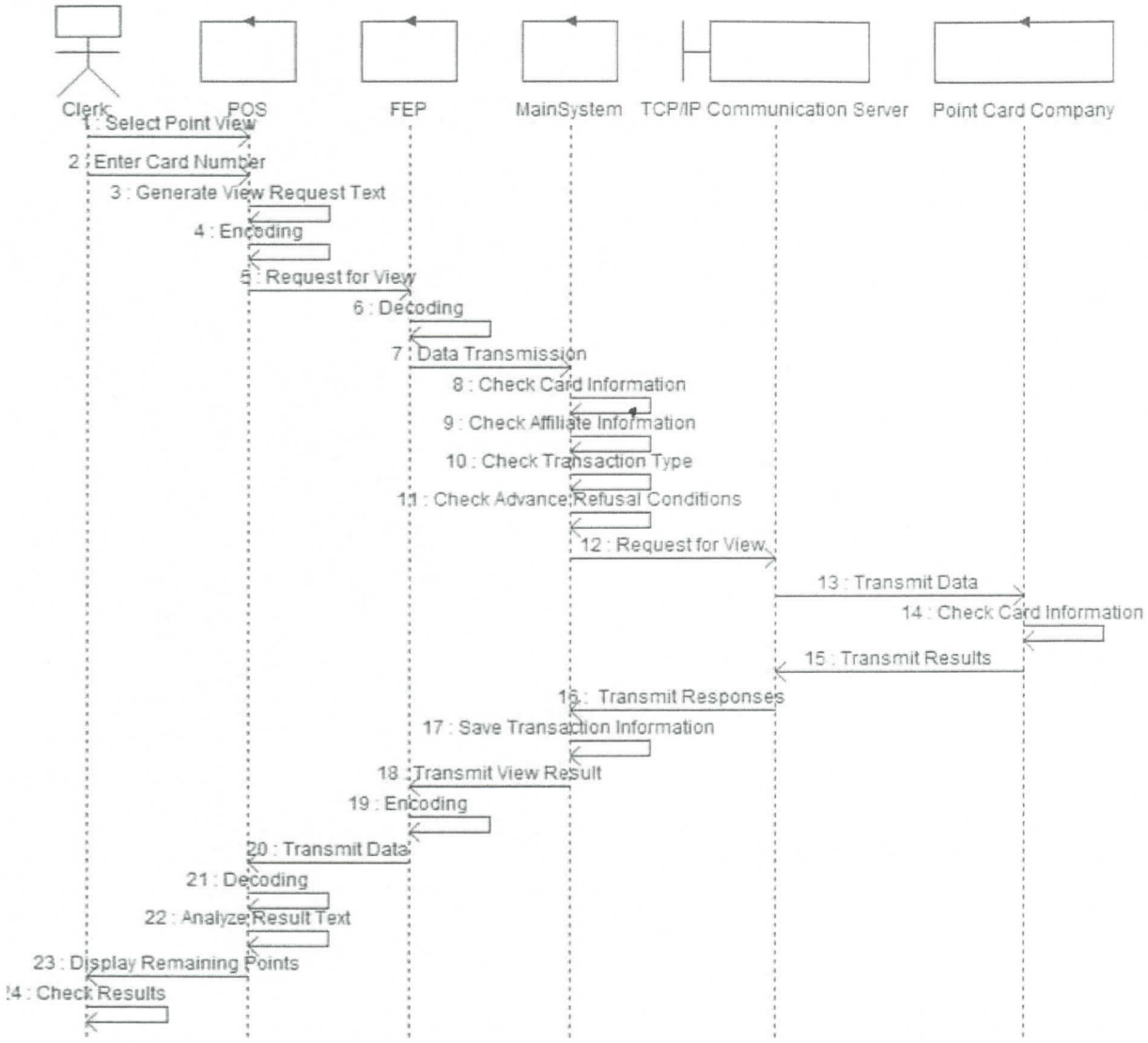


Figure 8. A sequence diagram for a 'Point view'.

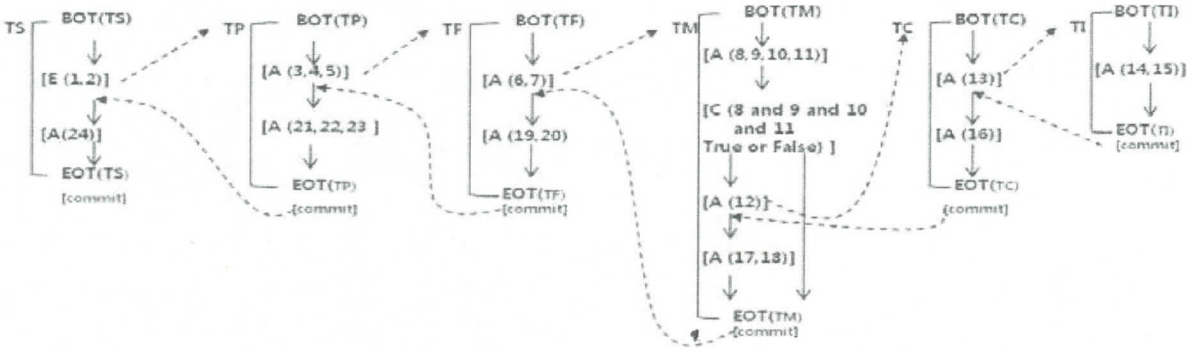
Table 4 is an ECA decision table for the sequence diagram reflecting the point view transaction.

Table 4. ECA decision table for 'point view'.

Transaction	EVENT	Condition	Action
Clerk(TS)	1,2		24
POS(TP)			3,4,5,21,22,23
FEP(TF)			6,7,19,20
MainSystem(TM)		8,9,10,11	1217,18

TCP/IP communication server(TC)	13,16
Point card company(TI)	14,15

Figure 9 shows an ECA diagram that combines the decision table and the sequence diagram. In this manner, each object is replaced with a unit transaction and each message flow is classified as an Event, Condition or Action.



Figue 9. ECA diagram for point view.

Table 5 is a test scenario for 'point view'. This is generated when the sequence diagram and the ECA decision tables are combined.

Table 5. Test scenario for Point view.

Transaction	Test scenario
Clerk(TS)	1-Clerk selects Point View. 2-Check Feeds in the card number to the system. POS displays the remaining points. **(TP commit)** 24-Checks the point view's result.**(TS commit)**
POS(TP)	Clerk selects Point View and feeds the card number.(condition for starting is satisfied) 3-A Text to request 'point view' is generated. 4-Encoding of the requested text is conducted. 5-A request to view points from the FEP is made. A response from FEP is received. **(TF commit)** 21-The response data is descrambled. 22-The resultant text is analyzed. 23-The results are displayed to the clerk. **(TP commit)**
FEP(TF)	FEP receives a request to view points from the POS (Condition for starting is satisfied). 6-Descrambling is conducted. 7-Data is sent to the main system. Results from the main system are received. **(TM commit)** 19-Descrambling is conducted. 20-Data is sent to the POS. **(TF commit)**

Main System (TM)	Data from the FEP is received. (Condition for starting is satisfied.) 8,9,10,11-Card and affiliate information, transaction type, and prior conditions for refusal are checked. If the results are found to be false, the transaction information is saved and the refusal results are sent to the POS. If the result is false, the transaction is committed.) 12-When the result is true, a request to view points is made to the TCP/IP communication server. Responses are received from the TCP/IP. **(TC commit)** 17-Transaction are saved. 18-Transaction results are transmitted to the FEP. **(TM commit)**
TCP/IP Communi-cation server (TC)	A request to view point is received from the MainSystem(Condition for starting is satisfied.) 13-Data is sent to the point card company. Results from the point card company are received. **(TI commit)** 16-Responses are transmitted to the MainSystem. **(TC commit)**
Point card com-pany (TI)	Data is received from the TCP/IP communication server (Condition for starting is satisfied). 14-Card information is checked. 15-Results are transmitted. **(TI commit)**

Figure 10 shows test cases for *'point view'*, generated from the test scenario and the ECA diagram.

Transaction type	Unit transaction	Test case	Test information (affiliate)	Test information (card number)	Conditions for starting	Action	Expected result
View point	TS	TC1	17824***	210001270770****	None	Select point view and enter card information	1,000 point confirmed
View point	TP	TC2	17824***	210001270770****	TS select point view	View request text generated, encoding, request for view to TF	Receive response text from TP
View point	TP	TC3	17824***	210001270770****	TP completed	Descrambling, result text analysis, remaining point display	TP completed
View point	TF	TC4	17824***	210001270770****	TP request for view	Descrambling and data sent to TM	Result received from main system
View point	TF	TC5	17824***	210001270770****	TM completed	Encoding and result sent to TP	TP completed
View point	TM	TC6	17824***	210001270770****	TF sends data	Card information checked	true of false
View point	TM	TC7	17824***	210001270770****	Card information confirmed to be true	Affiliate information confirmed	true of false
View point	TM	TC8	17824***	210001270770****	Affiliate information confirmed to be true	Transaction type confirmed	true of false
View point	TM	TC9	17824***	210001270770****	Transaction type confirmed to be true	Advance refusal conditions checked	true of false
View point	TM	TC10	17824***	210001270770****	TC6~TC9 results are true.	Request for view to TC	Receive view response from TC
View point	TM	TC11	17824***	210001270770****	TC6~TC9 results are false	TM completed	Not requested to TC. Refusal response to TF
View point	TM	TC12	17824***	210001270770****	TC completed	Transaction information saved and result transmitted	TM completed
View point	TC	TC13	17824***	210001270770****	TM's view request	Point view requested to TI	Receive view results from TI
View point	TC	TC14	17824***	210001270770****	TI completed	Result transmitted to TM	TC completed
View point	TI	TC15	17824***	210001270770****	TC's request for view	Card information checked	true of false
View point	TI	TC16	17824***	210001270770****	T15 is true	210001270770****points transmitted to TC	TI completed
View point	TI	TC17	17824***	210001270770****	T15 is false	Point view refusal transmitted	TI completed

Figure 10. Test cases for 'point view'.

CONCLUSION

Currently most VAN companies do not have any sufficient methods of generating test cases for real time transaction environments. Until now, they use checklists based on prior accumulated data and experiences. It is important to note that a functional test in real time transaction chains takes with the statements of requirements. It is mainly based on functional requirements.

A test case is mainly based on functional requirement. As most companies in real time transaction environments do not have any methods to generate test cases, they prepare with this testing based on their accumulated experiences. In this paper, we suggest validation process to test consecutive real-time transactions from transaction analysis. The suggested process generates UML modeling and ECA (Event, Condition Action) rules, and makes ECA decision table. With sequence diagram and ECA decision table, we can generate test scenarios, and then extracts test cases through dynamic transaction analysis. We still research on test cases about simultaneous and parallel transactions.

ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2012M3C4A7033348) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011601).

LITERATURE CITED

Beizer, B. 1990. *Software Testing Techniques*. Van Nostrand Reinhold Inc., New York, USA.

Hwang, S.S., J.Y Yang and H.C. Moon. 2001. A study on electronic transactions via VAN-based computers and networks. The Korean Academy of International Business Management, Symposium Proceedings 2001, pp. 159-185.

Kim, G.J. 1990. VAN and its use. Telcom 6(2): 71-79.

Kim, K.W., B.K. Park, W.S. Jang, S.Y. Moon and R.Y.C. Kim. 2010. A study on system implementation through modeling the financial VAN (Value Added Network) connected service based on reverse engineering. Proceedings of Korea Computer Congress 2010, 37(1-B), pp. 92-96.

Kim, S.G., J.B. Lee, J.H. Lee and G.H. Choi. 1990. A study on inter-industry information network for the facilitation of VAN industrialization. The Korean Academic Society of Business Administration 2: 24-24.

Kim, G.Y. 1988. The future prospect for VAN. Communications of the Korea Information Science Society 6(5): 5-10.

Lee, G.H. and H.H. Gwak. 1988. The present and future of Korea's financial VAN. Communications of the Korea Information Science Society 6(5): 15-22.

Nursimulu, K. and R.L. Probert. 1995. Cause-effect graphing analysis and validation of requirements. *In:* Proceeding of the 1995 Conference of the Centre for Advanced Studies on Collaborative Research, pp. 46-61.

Singla, S., D. Kumar, H. M. Rai and P. Singla. 2011. A hybrid PSO approach to automate test data generation for data flow coverage with dominance. International Journal of Advanced Science and Technology 37: 15-26.

Wang, T. 2011. Automated test data generation based on matching combination and testing matrix for pairwise testing. International Journal of Software Engineering and its Applications 5(1): 21-38.

Yoo, C.J. and H.M. Noh. 2009. Test case generation techniques based on use cases for interoperability test of component-based software. Journal of KIISE: Software and Applications 36(5): 361-375.

Zheng, W. and G. Bundell. 2009. Contract-based software component testing with UML models. International Journal of Software Engineering and its Applications 3(1): 83-102.

ASIA LIFE SCIENCES

The Asian International Journal of Life Sciences

ISSN 0117-3375

Supplement 11 **26 July 2015**

CONTENTS

Cont. on Inside Back Cover

***Reviewers for this Special Issue*: Please refer to pages 719-726 of this Issue.**

 Printed in the Philippines