Tai-hoon Kim Hojjat Adeli
Haeng-kon Kim Heau-jo Kang
Kyung Jung Kim Akingbehin Kiumi
Byeong-Ho Kang (Eds.)

Communications in Computer and Information Science 257

Software Engineering, Business Continuity, and Education

International Conferences ASEA, DRBC and EL 2011,
Held as Part of the Future Generation
Information Technology Conference, FGIT 2011,
in Conjunction with GDC 2011,
Jeju Island, Korea, December 2011, Proceedings

Springer

A Validation Process for Real Time Transactions

Kyu Won Kim[1], Woo Yeol Kim[2], Hyun Seung Son[2], and Robert Young Chul Kim[2]

[1] Deptatment of Information System, KOVEN Co.,Ltd.
Seoul, 135-270, Korea
kkw1206@kovan.com
[2] Dept. of CIC(Computer and Information Communication), Hongik University,
Jochiwon, 339-701, Korea
{john,son,bob}@selab.hongik.ac.kr

Abstract. Real financial transactions take place on a consecutive real-time serialization. They are carried out as dynamic transaction chains along with various related systems rather than with just one system. This paper suggests a process that generates a test case for the verification of such consecutive real-time transaction system. The suggested process generates a test case through mechanism applied with UML and ECA (Event/Condition/Action) rules. Through analyzing transactions, we generate UML modeling, then maps UML with ECA rules, which creates an ECA-decision table. A test scenario is generated with this table. That is, the test scenario is modeled from an ECA diagram based on the consecutive transaction chains, which generate a test case.

Keywords: transaction, modeling, ECA, test case.

1 Introduction

VAN companies are intermediates that connect affiliates using settlement systems with card companies as well as other various financial institutions. As for credit card settlement systems, credit card terminals, VAN (Value Added Network) companies, and card companies are interconnected [1,2]. Credit card settlement systems were followed by various settlement models such as transaction approvals via telephone wires and terminals, POS systems, and HOST servers [3,4,5].

Based on various settlement models, VAN companies' systems have become more complex. Also, the structure requires the acceptance of all requests from affiliates and financial institutions, resulting in frequent maintenance and repairs of the system [6]. 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. 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

T.-h. Kim et al. (Eds.): ASEA/DRBC/EL 2011, CCIS 257, pp. 298–304, 2011.

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.

This paper consists of the following. Chapter 2 explains a related study. Chapter 3 explains about real-time transactions and suggests a method to generate a test case. Chapter 4 provides conclusions and tasks to be studied in the future.

2 Related Work

The Cause-Effect Graphing technique provides systemic methods to develop a statement prepared in a natural language into a decision table [7]. 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.
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[8]. 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 [9]. 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.

3 Validation Process for Real-Time Transactions

Figure 1 is a process to generate a test case for real-time transaction chains. A use case is modeled based on the path, action and conditions for transactions acquired from analysis. The use case scenario based on the use case modeling is modeled through a sequence diagram. This diagram generates a table that decides ECA (Event, Condition, Action) and an ECA diagram.

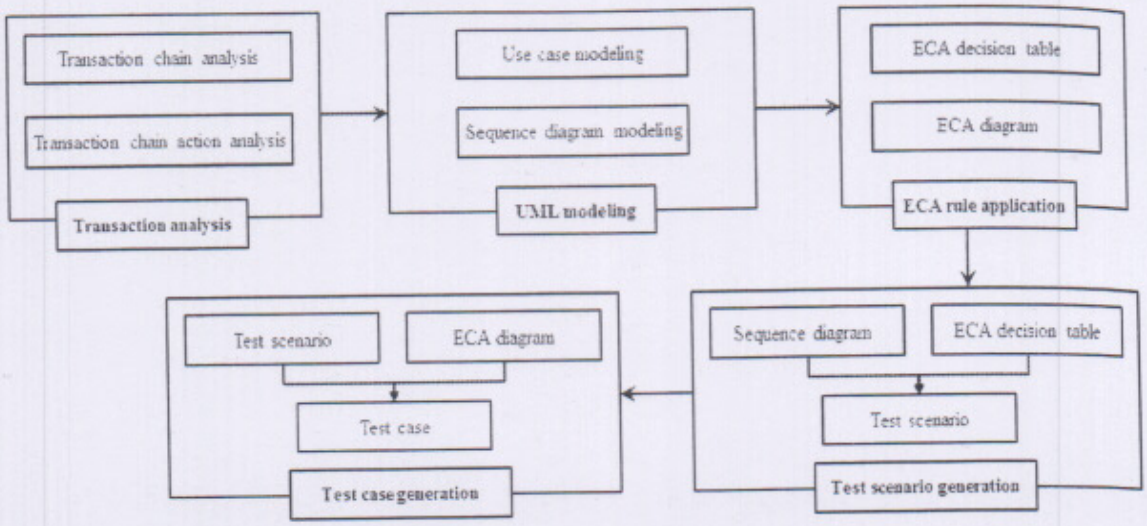


Fig. 1. Process to generate test cases for real-time transactions

Figure 2 is a process to analyze a simple transaction. Unit transactions, T1 (ACTOR), T2 (A), and T3 (B) are defined while unit transaction chain is organized. The transaction is analyzed based on dynamic diagram modeling of transaction chains. In order for T1 to be committed, T2 transaction is committed. In order for T2 transaction to be committed, T3 transaction shall be committed.

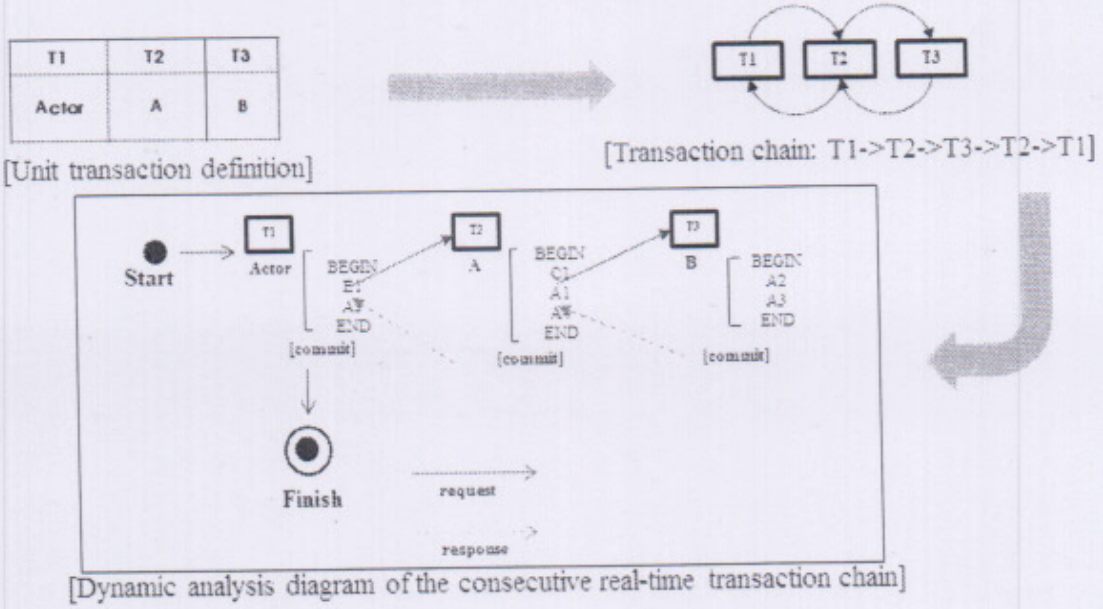
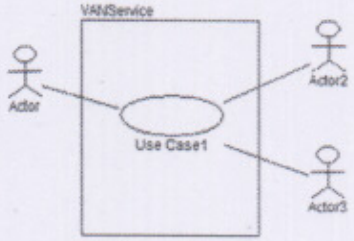


Fig. 2. Analysis of consecutive real-time transactions

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



Use Case Name	Use Case1
Participating Actor	Actor, Actor2, Actor3
Flow Event	1. Actor invoke E1 2. A invoke C1,A1 3. B invoke A2,A3 4. A invoke A4 5. Actor invoke A5
Entry Condition	Actor invoke E1
Exit Condition	Actor invoke A5

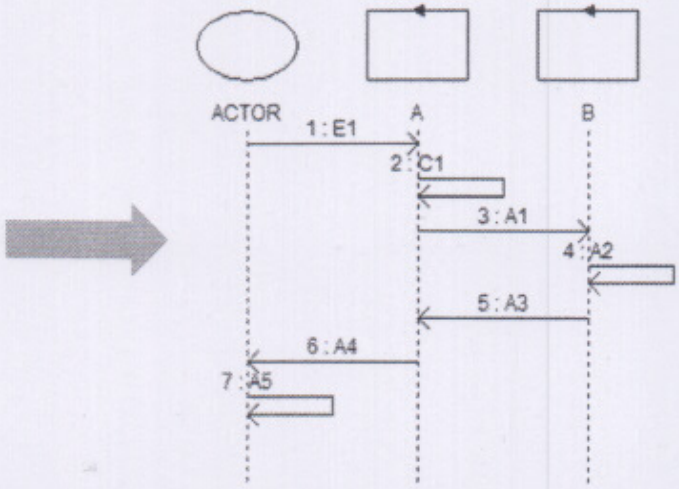


Fig. 3. UML modeling

Table 1 is an ECA table with an expression of the sequence diagram. Actor's message numbers, 1(E1) and 7(A5) are applied to Event and Action of the unit transaction Actor. A's message numbers, 2(C1), 3(A1), and 6(A4) are applied to Condition and Action of unit transaction A. B's message numbers, 4(A2) and 5(A3) are applied to Action of unit transaction B. Figure 4 is an ECA transaction diagram. Table 1 represents based on the modeling of Figure 4.

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

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

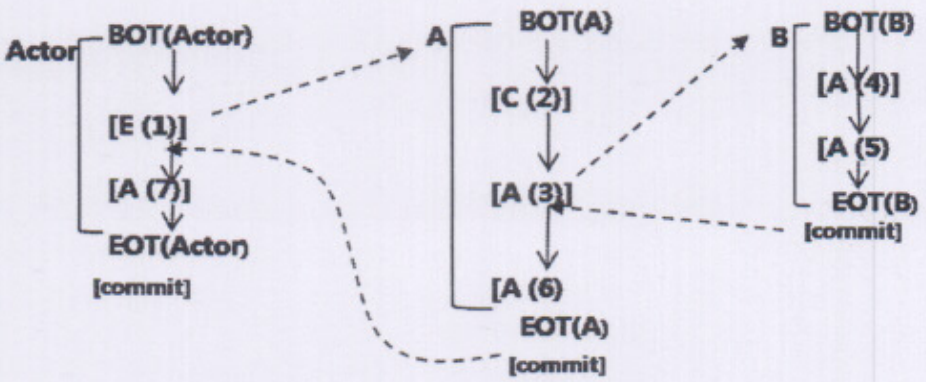


Fig. 4. ECA Diagram

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)

Table 2 is a test scenario organized based on the combination with the sequence diagram in Figure 3 and the decision table in Table 2. Unit transactions of the decision table are applied to the transactions in Table 2 while Event, Condition, and Action of each unit transaction are applied to the test scenario. Messages pertaining to Event, Condition, and Action in the ECA decision table are represented from the sequence diagram for a test scenario. Figure 5 is a test case generated based on the combination with the ECA transaction diagram and the test scenario in Table 2. The ECA based Test Cases consist of the subject transaction, test case ID, conditions for starting, Action, and expected results.

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

Fig. 5. ECA-based Test Case

Transactions of the test scenarios are applied to the subject transactions while the actions prior to those to be conducted are applied as for conditions to start. The actions following those to be conducted are applied for the expected results.

4 Examples of Application

As a case study, the application is for point card transactions that occur in the VAN environment. There are four transactions such as point view, point collection, point use, and point cancellation. As for the application to "point card" transactions, the information on test cards and test affiliates is necessary. For this, hypothetical test card numbers and affiliate numbers are to be set up. A real test case is generated based on this.

5 Conclusion

A functional test in real time transaction chains take place based on the statements of 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 parallel transactions.

Acknowledgments. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)(NIPA-2011-(C1090-1131-0008)) and the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation.

References

1. Sam-saeng, H., Jae-young, Y., Hee-cheol, M.: A study on electronic transactions via VAN-based computers and networks. In: The Korean Academy of International Business Management, Symposium Proceeding 2001, pp. 159–185 (2001)
2. Seong-geun, K., Jae-beom, L., Joo-heon, L., Gwang-ho, C.: A study on inter-industry information network for the facilitation of VAN industrialization. The Korean Academic Society of Business Administration 2, 24 (1990)
3. Gi-hyeon, L., Hong-hoi, G.: The present and future of Korea's financial VAN. Communications of the Korea Information Science Society 6(5), 15–22 (1988)
4. Gi-yong, K.: The future prospect for VAN. Communications of the Korea Information Science Society 6(5), 5–10 (1988)
5. Geon-joong, K.: VAN and its use. Telcom 6(2), 71–79 (1990)

6. Kyu-Won, K., Bo-Kyung, P., Woo-Sung, J., So-Young, M., Kim, R.Y.C.: A Study on System Implementation through modeling the Financial VAN(Vale Added Network) Connected Service Based on Reverse engineering. In: Korea Computer Congress 2010, Jeju, Korea, vol. 37(1)(B), pp. 92–96 (2010)
7. Nursimulu, K., Probert, R.L.: 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 (1995)
8. Beizer, B.: Software Testing Techniques. Van Nostrand Reinhold Inc., New York (1990)
9. Ji-Hyun, L., Hye-Min, N., Cheol-Jung, Y., Ok-Bae, C., Jun-Wook, L.: Test Case Generation Technique for Interoperability Testing. Journal of KIISE: Software and Applications 33(1), 44–58 (2006)