Automated Test Scenarios Generation for an E-barter System

Ana Cavalli and Stephane Maag
GET/Institut National des Télécommunications
CNRS-Samovar Laboratory 9, rue Charles Fourier
F-91011 Evry Cedex, France
{Ana.Cavalli,Stephane.Maag}@int-evry.fr

ABSTRACT

This paper presents a formal specification of an e-barter system and a set of scenarios to test the conformance of a given implementation to some targeted system functionalities. The functionalities of the e-barter system are inspired from those presented in [7] that are based on intelligent agents using utility functions to represent customer preferences but also integrating transaction and shipping costs. The system specification is performed using the SDL language. It includes two markets representing two cities, both cities containing several agents representing the customers preferences. Agents are different instances of the same process, allowing the dynamic inclusion of new agents and of new resources. The scenarios are generated from the specification and from some test purposes using a tool developed at INT [1]. The test purposes express specific system properties and are used to guide the test generation procedure that is completely automated. In this paper, we also present the experimentation results of the application of our tool to the e-barter system.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Verification; J.8 [**Computer Applications**]: Internet Applications

Keywords

e-commerce, formal methods, testing tools, specification techniques.

1. INTRODUCTION

An e-barter system consist of a set of agents performing exchanges of products [6]. The objective of these systems is to obtain the satisfaction of the customers by providing them with the expected goods. To exchange goods by money is not the main goal, this is a difference with usual e-commerce systems. However, a customer may wish to exchange a good by some units of the good money.

In some electronic e-barter systems exchanges are performed by intelligent agents which autonomously perform electronic transactions using information about customers preferences provided by utility functions. These functions define the customers preferences with respect to goods to be exchanged and provide them with a negotiation capacity when interacting with other agents.

For these systems, conformance testing becomes a crucial phase of their development. Functional and security failures caused by poor testing may have a catastrophic effect on e-commerce and e-barter system reliability. In the last years, different conformance testing methods have been developed for distributed communicating systems. Some of them can be usefully adapted to the validation of e-commerce and e-barter systems. This is the case for goal-oriented testing techniques that consist of selecting a specific property of the system that is likely to be false or the behaviour of a specific component of the global system that is likely to be faulty, and to generate test scenarios for only those parts. In general, this selection is made by human experts that identify the part of system's behaviour or the expected properties that might be subject of testing and formulate test purposes or goals based on this identification.

In this paper, we propose a formal specification and the application of a test generation method to produce a set of test scenarios for some selected functionalities of an e-barter system.

The proposed system is inspired from a previous work [7]. In this system, transaction and shipping costs are taken into account, doing the system be close to real ones. Another characteristic is that the exchanges are made beyond local markets, from local ones to states until to achieve a global exchange market. First, agents are grouped in local markets, that correspond to the localities of the customers. Once this market is saturated, that no more exchanges can be performed, a new agent is created that represents the interests of all local agents. These new agents are grouped into state markets. This procedure is repeated until a global market is created.

The e-barter system described in this paper is a case study that includes two markets. These two markets represent two cities that include five agents representing the customers preferences. Agents are different instances of the same process, allowing the dynamic inclusion of new agents and of new resources.

For the specification of the e-barter system we use the SDL language [4]. This language is well adapted to the description of the system: it allows to specify agents behaviour as processes that exchange messages and to provide a hierarchical description of the system using different architecture levels, starting from local markets and going up to global ones. The SDL specification has also the advantage to facilitate the addition, removing, and changes of the system functionality.

In order to generate test scenarios, we use a method and a tool developed at INT. The method is based on test purposes and the tool allows to generate tests based on these test purposes. In this paper, we apply this tool to generate test scenarios for some functionalities (for instance insertion of a new agent in the e-barter system), expressed as properties or purposes to be tested. These test scenarios will be used to check that different implementations of the e-barter system satisfy the required functionality.

Apart from the work already mentioned before, there is little work in this area. In [8] a formal verification and validation approach based on SDL for e-commerce systems is presented. Even if this work is interesting and shows the feasibility of a test environment based on a formal methodology, it presents the following drawback mentioned by the authors: "the SDL model they created has many states and transitions. The state space is very large and requires large amounts of memory to run the state-space search". The advantage of the method we present in this paper is that it does not need to generate the complete state space. On the contrary, it is based on the generation of partial graphs, thus avoiding the state space explosion.

The paper is organized as follows. Section 2 introduces the main notions of the formal semantics for the e-barter system and language used for the specification. Section 3 presents the SDL specification of the proposed e-barter system. In Section 4, the results of the test scenario generation procedure are presented. Finally, Section 5 gives the conclusion and the perspectives of this work.

2. BASICS

This section introduces the basic concepts used in the paper. The work presented in this paper is based on a formal semantics for an e-barter system. This formal semantics is presented in detail by M. Nùñez et al. in [7]. We describe the main notions of this last paper in order to provide the concepts and the semantics used in this paper. Secondly, the language SDL is presented. This language has been used to specify the e-barter system.

2.1 Formal definitions

The customers participating in an e-barter system are represented by (electronic) agents[1]. These agents provide two characteristics: a basket of resources (indicating the items that they own, defined as $\overline{x} \in \mathbb{R}_+^{m>0}$) and a utility function. This function ($u : \mathbb{R}_+^{m>0} \to \mathbb{R}$) permits the agents to represent their exchanges and the customers preferences among different baskets of resources. Whenever an agent has reached a (possibly multilateral) deal, it is notified to the customer. If the other customers have also reached a deal, that is they give their approval, then the deal is performed, transaction fees will be added and shipping costs will be computed according to the resources obtained after exchange and the distance between the involved customers. Agents exchange resources inside their local market. A market M (different than markets $M_a, \ldots, M_k$) is represented by $unsat((M_a, \ldots, M_k), sh, pr)$ and $(S, u, \overline{x}, sh, pr)$ for some unsaturated and saturated markets respectively. We respectively denote by S, sh and pr, the set of agents into the market (S=[] means that the market is itself an agent), the shipping cost of every possible transaction, and the profit gained by the market due to the transaction costs. The saturation means that no more exchanges may be performed between the agents or sub-markets. At this point, the agents of the saturated market are combined in order to create a new utility function, while the baskets of resources are added. The creation of a new utility function is done by the application of a function called *CreateUtility*. This latter combines the utility functions and the resources of the corresponding agents to allow the barter with higher order markets. This function is computed in such a way that it is possible to negotiate for maximizing the overall profit of the represented agents. In turn, we need to redistribute the resources to the original agents appearing in the leaves of the tree. This is performed by a recursive function called *Deliver*. We do not detail in this paper the rules used to compute these operations, we let the reader have a look at [7] for a detailed description.

2.2 The SDL language

The Specification and Description Language SDL standardized by ITU-T [4] is widely used to specify communicating systems and protocols without ambiguities. This language has evolved according to user needs. It provides new concepts needed by designers to specify systems more and more complexes. SDL is based on the semantic model of Extended Finite State Machine (EFSM) [5]. Its goal is to specify the behavior of a system from the representation of its functional aspects. The description of the functional aspects is provided at different abstraction levels. The most abstract is the one describing the system and the environment (the customers), while the lowest is the specification of abstract machines composed by signals, channels, tasks, etc. Two kinds of properties may describe these functional aspects: the architectural and behavioral properties. The first one denotes the architecture of the system, that is the connection and organisation of the elements (blocks, processes, etc.). The second one describes the behaviors of the entities after an interaction with the environment (for example, a reaction of electronic agents from a request of a customer). These reactions are described by tasks, transitions between states, and are based on the EFSMs.

A verification on local variable values imposes a condition (predicate) on moving to the next state. The actions associated with a transition include: the execution of tasks (assignment or informal text), procedure calls, dynamic creation of processes in order to include new agents into a system for instance (SDL contains the concepts of "type" and "instance of type"), arming and disarming timers, etc. SDL supports objects that allow to define generic types that could be validated and used in different contexts. It also supports ASN.1 [2], a standard defined for data transfer. Specifically, data are defined as abstract data type.

3. THE E-BARTER SPECIFICATION

[1]In terms of [10], our agents present as information attitude belief (vs. Knowledge), while as pro-attitudes we may consider commitment and choice (vs. Intention and obligation)

In this section we describe the e-barter specification modeled using SDL in such a way it is very easy to add, remove and observe some functionalities. All our specification follows the formal definition of the e-barter system mentioned in the previous section. Each rule and operation is scrupulously respected.

In our system we specify two markets M_1 and M_2 containing five agents representing five sub-markets $A_i = ([], u, \overline{x}, sh_i, 0)$, $i \in \{1, 2, 3\}$, $B_j = ([], u_j, \overline{x_j}, sh_j, 0)$, $j \in \{4, 5\}$. To give a concrete idea, the markets represent two cities 1 and 2 in which there are 3 and 2 customers (agents) respectively. Let us notice that the customers are dynamically managed and by this way we may include new agents and also new resources. Indeed, the agents are some instances of the same process. In our SDL specification, each market M_1 and M_2 is represented by a block that allows to transmit and receive some messages or signals. Into each block, the behaviors of the markets are modeled by some processes. The specification of our e-barter system is shown by the figure 1. The channel *preferences* allows to transmit the signal *callpreferencesA1* in order to include a new agent A1 into the system. This last signal carries the utility function and the current resources of the agent A1 to the market M_1 and these information are forwarded to the *Matrices_Generator_process*. This last process has the responsibility to generate some valid exchange matrices (see definition **??**). For each barter within M_1, M_2 and between them, we have limited the number of the exchange matrices to five for our case study, but this number may easily and dynamically be decreased or increased. Once the matrices are generated, the exchanges between the agents may begin. A signal is sent to the markets which, in return, ask for the matrices. Then they reach a saturated state and the corresponding agents are combined in order to create a new agent called *SuperAgent*. The combination of the agents of a same market is carried out in the M_1 and M_2 modules. However, the exchanges between these two SuperAgents, $SuperAgent1$ and $SuperAgent2$ are still performed by the process contained in the Matrices_Generator block.

Finally the global market is defined as $M = ms(unsat((M_1, M_2), sh, pr))$. We consider that our system contains four different resources which are *bike, book, dvd* and *vhs*. The number of resources in the specification may be modified. Of course, as usual, we have inserted the resource *money* to take into account the shipping cost and the profit gained by the markets. Further, the shipping cost takes into account the distance separating the customers.

Several processes are specified in order to formalize the rules needed to process the behavior of the markets. As we discussed before, one of them is used for the generation of the valid exchange matrices. We also need the processes *Saturated, Create_Utility, SuperAgent_process* and *Deliver* used to check whether the markets are saturated or not, to create the new utility functions to generate the *SuperAgents*, and in order to deliver the resources to the original agents respectively. The e-barter specification is made of approximatively 5000 lines of SDL. In order to give a general idea of the complexity of the SDL system specification, we present by the Figure 2 some significant metrics of the global system.

In order to simulate the system, we use a configuration file which initializes some variables such as the number of agents and the databases (utility functions, baskets of resources). We verify that the specification is free from deadlocks and livelocks within the simulated state space. Indeed, the presence of such deadlocks or livelocks reveals that the e-commerce system does not behave as expected. In order to generate the test scenarios, we have used a test scenario generation method described in the following section.

Lines	4987
Blocks	13
Processes	17
Procedures	3
States	24
Signals	32
Macro definitions	2
Timers	0

Figure 2: Metrics of the e-barter specification.

4. TEST OF A E-BARTER SYSTEM

This section gives an outline of the test generation procedure used for test scenarios generation. It also presents the experimentation results of the application of the test generation procedure to the e-barter system described in the previous section.

4.1 Test scenarios generation

Our main objective is to generate a set of scenarios to test some expected properties of the system. These properties are expressed as test purposes. In order to produce the scenarios we apply a test tool we have developed at INT. The generation procedure is completely automated and follows the following main steps:

Step 1. To obtain a precise and concise *formal specification* of the system to be tested. This specification takes into account the systems functionalities as well as the data specific to the test (test architecture, test interface, etc.). We use the SDL specification of the e-barter system described in the previous section.

Step 2. To *select* the appropriate tests. This selection can be performed according to different criteria. This correponds to the definition of the test purposes: a test purpose can be a specific property of the system or the behaviour of a specific component of the system (for instance a market component or an original agent).

Step 3. To *generate* the test scenarios. The test purposes are used as a guide by an algorithm for component testing that we have developed. As a result, our algorithm calculate a test scenario that applied to the implementation under test, verifies the test purpose. A scenario is a sequence of interactions (between the system and the environment) that includes the interactions that represents a test purpose. This algorithm has been implemented in our tool called TESTGEN-SDL [1].

Step 4. To *format* the tests. That is, to produce test scenarios in some accepted formalism. In our case, test scenarios are produced in Message Sequence Charts (MSC), a formalism widely used in industry to describe processes messages exchanges [3] and in Tree and Tabular Conformance Notation (TTCN), the ITU-TS standard language used for test specification [9].

4.2 Experimentation Results

This section presents the experimental results on the gen-

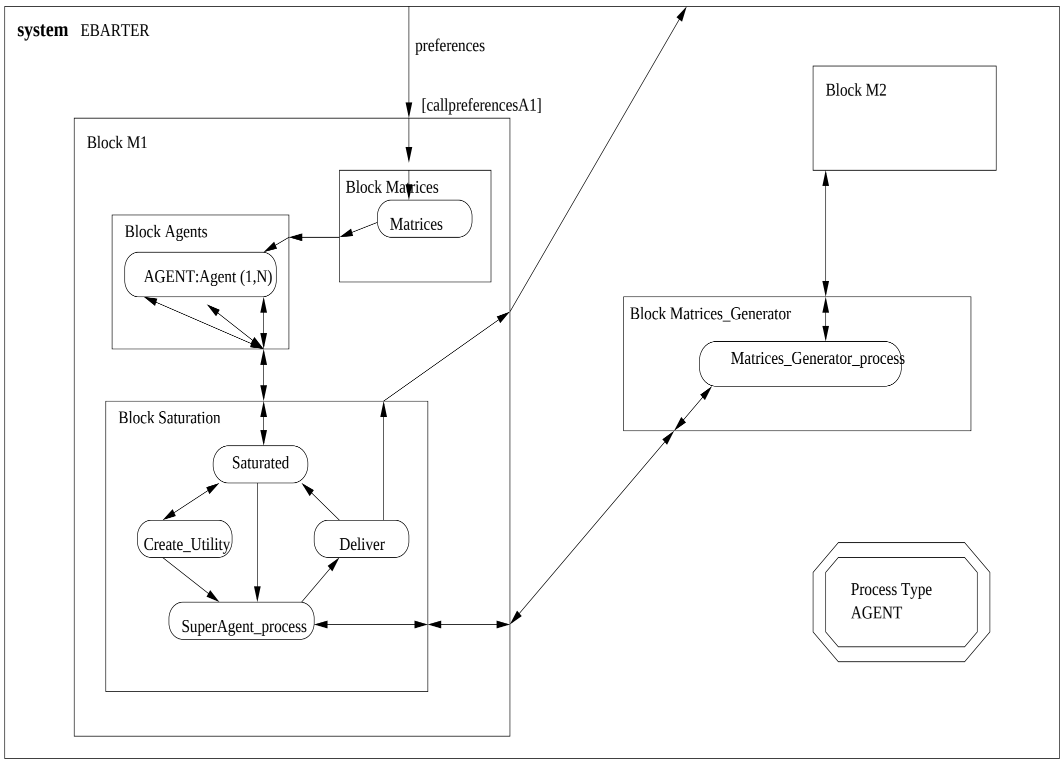

Figure 1: Overview of the e-barter system specification.

eration of test scenarios for the e-barter system. The test scenarios checks that the test purposes are satisfied by the implementation. These last are selected taking into account the main functionalities that the tester wishes to test on the implementation. They can cover all the interactions of the system with its environment or all the interactions of a module with the other modules. However, the cost for testing all interactions is very high. The ideal situation is that a system expert provides to the tester the test purposes (i.e. the system functionalities he wants to check). For this case study, we have selected three test objectives, but they can be more if this is decided by the tester. They are focused on the behaviour of the Matrices_Generator module, in particular we want to check that the insertion of new agents in markets M1 and M2 does not affect the other components.

The Matrices_Generator module contains several processes and therefore many functions are executed in this module. In order to generate test scenarios, we need to formulate the expected properties to be tested under the form of test objectives. This is what we illustrate in the following. For the sake of simplicity, we propose three test purposes:

- *Test purpose 1*: To test that the utility function of a new agent is taken into account by the valid exchange matrices;
- *Test purpose 2*: To test that the module receives the signal defining the saturation of a new agent that has just been included to the system;
- *Test purpose 3*: To test that the new agent receives an expected basket of resources according to its utility function (preferences).

These three test purposes may reveal some crucial errors into an implementation of an e-barter system. Indeed, they allow to detect some errors during the exchanges of resources and also when a new customer is included to the system. After application of the test generation procedure, a test scenario is produced for each one of the test purposes. The results obtained are illustrated in the Figure 3.

Test purpose	Test scenario length	Duration
♯1	19	4.3s
♯2	5	13.7s
♯3	29	65.8s

Figure 3: Results obtained.

Once all the tests purposes have been exercised, we may merge in a single test scenario. The obtained scenario is of length 53 transitions and the execution times are relatively short (on a Sun Sparc Ultra 5). Let us note that the generated test scenario include the behavior of other components (such as *Create_Utility* or *SuperAgent_process*). We illustrate in the Figure 4 a part of the obtained test scenario produced as a MSC (see Step 4 in section 4.1).

We obtain in this scenario the test of our two first test purposes.The scenario is therefore generated until the test purpose 3 is reached.Further, with these results, this test scenario may be applied on a real implementation of an e-barter system in order to find erroneous behaviors of the implemented system.

5. CONCLUSION

We have presented in this paper the specification and the

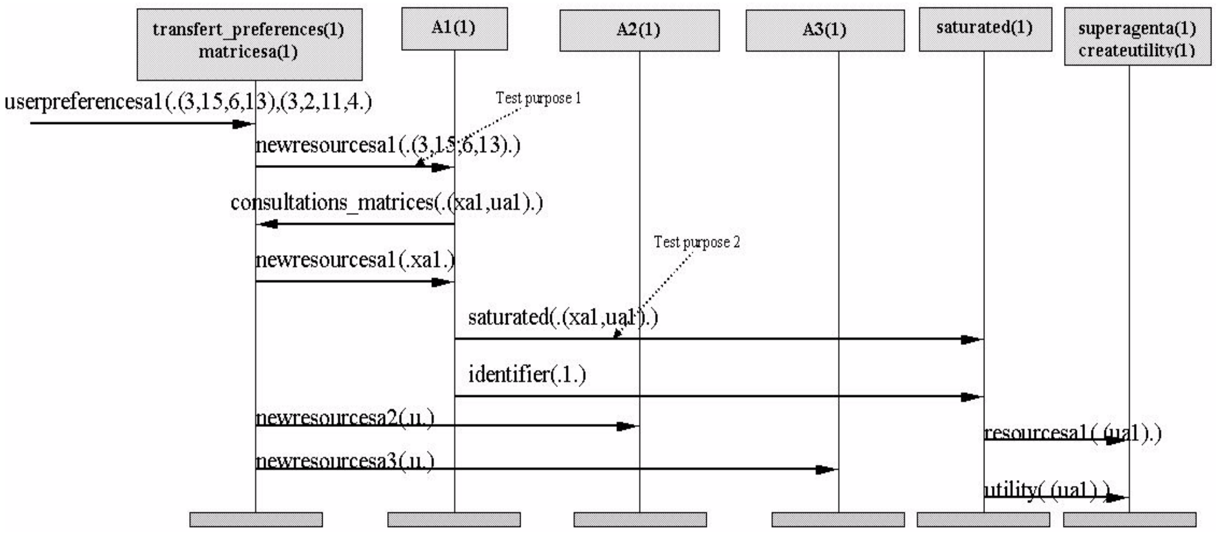

Figure 4: An MSC from test scenario generation.

application of a method and tool for test scenario generation for an e-barter system. The proposed approach presents several advantages. First, the design of a formal specification from which tests are generated contributes to eliminate design errors and ambiguities. Secondly, the use of test purposes for test generation can be very useful to meet customer requirements. Also, automated test generation is less costly that tests written manually, reducing the time to market. And finally, the test scenarios we have generated can be reused for non functional testing such as system capacity and response time testing. And the proposed methodology can be applied to large-scale system and be easily applied to e-commerce systems.

Acknowledgments
We would like to thank professor Manuel Núñez, from the University Complutense of Madrid, for the useful discussions about the characteristics of an e-barter system that helped us for the SDL specification proposed in this paper.

6. REFERENCES

[1] A. Cavalli, D. Lee, C. Rinderknecht, and F. Zaidi. Hit-or-jump: An algorithm for embedded testing with applications to IN services. In Jianping Wu, Samuel T. Chanson, and Qiang Gao, editors, *Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX'99*, volume 156 of *IFIP Conference Proceedings*, pages 41–56. Kluwer, 1999. Beijing, China.

[2] O. Dubuisson. *ASN.1*. Springer, 1999.

[3] ITU-T. *Recommandation Z.120, Annexe B: Algebraic semantics of Message Sequence Charts*, April 1995. Geneva.

[4] ITU-T. Recommandation Z.100: CCITT Specification and Description Language (SDL). Technical report, ITU-T, 1999.

[5] D. Lee and M. Yannakakis. Principles and Methods of Testing Finite State Machines - a Survey. In *The Proceedings of IEEE*, volume 84, pages 1090–1123, August 1996.

[6] N. López, M. Núñez, I. Rodriguez, and F. Rubio. A formal framework for e-barter based on microeconomics theory and process algebras. In Springer, editor, *Innovative Internet Computer Systems, LNCS 2346*, pages 217–228, 2002.

[7] N. López, M. Núñez, I. Rodriguez, and F. Rubio. A multi-agent system for e-barter including transaction and shipping costs. In *18th Symposium on Applied Computing (SAC 2003): Special Track on e-commerce Technologies*, pages 587–594, 2003. Melbourne, Florida.

[8] R.L. Probert, Y.Chen, M. Cappa, P.Sims, and B.Gahaziadeh. Formal verification and validation for e-commerce: theory and best practices. *Information and Software Technology*, pages 763–777, July 2003.

[9] ETSI. TTCN-3. *TTCN-3 - Core Language.*

[10] M. Woolridge and N.R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.