International Conference on Applied Mathematics, Simulation and Modelling (AMSM 2016)

Scenario Automation Machine Based Multi-agent System Evolution

Maoguang Wang*, Jiangping Zhao and Leilei Ge
School of Information, Central University of Finance and Economics, Beijing, China
*Corresponding author

***Abstract*—The complex adaptive multi-agent system evolution requires the system is capable of evolving dynamically according to the application requirements and run-time environment changes. The scenario concept in the requirements phase of software engineering domain is introduced to the dynamic evolution. Different from the static requirements description, in this paper the scenario is used to illustrate the system dynamic evolution context. It is used to define the roles, the agents, the environment and the self-adaptive policies. Firstly, the scenario formal representation and scenario automation machine are defined. The MAS evolution is represented as a series of scenarios transformation based on the scenario automation machine. It provides a new method for complex software evolution and supports the system reuse in larger granularity.**

Keywords-scenario automation machine; multi-agent system; evolutuon

I. INTRODUCTION

Scenario concept was first introduced into the military and strategic research fields. Economists use the scenario to define a long-term plan, while management experts and policy-makers use scenario to make the strategic decision-making. In the field of software engineering, scenarios are often used for the human-computer interactions, or be used to illustrate the requirements of the software-intensive systems. Software engineers usually look the scenario as an efficient method to mine user requirements, and integrate the requirements into the software whole-life cycle development. Now most researchers generally look the scenario as a kind of the human-computer interaction and information system communication tools, which have a multi-perspective interpretation and applications.

Generally the complex adaptive multi-agent system(MAS) is expected to adjust its structures or behaviors automatically. It is capable of predicting, making decisions, and improving itself in accordance with environment changes, such as biology system, ecosystem, social system, economic system etc.

In a typical multi-agent system the agents interact with each other dynamically, which have the capabilities of sensing environment and evolving continually in accordance with their behaviors and environment changes [1].

The purpose of introducing the scenario is to support the MAS adapt to environment changes in a larger granularity. Now scholars usually study the software evolution using petri-net or architecture based method so on [2,3]. But they still face two major challenges. First, it is significantly dependent on a centralized control center to coordinate the adaptive behaviors of the system. Second, it is difficult in advance to predict all the adaptive behaviors in a variety of application environments.

In the practical multi-agent system development, we propose and test the scenario based MAS evolution method. First of all, the MAS characteristics and behaviors are represented as a series of application scenarios. The scenario defines the agent role, task, behaviors and the self-adaptive policies. With the scenario transformation, the agent behaviors and their interactions will change automatically. Thus the MAS evolves automatically according to the scenario transformation. And the scenario transformations are illustrated in a scenario automation machine.

In our design, we design the scenario as a special agent, to avoid centralized control the scenario agent will look other agents as dependent services or interaction agents. By deploying scenario agents dynamically, we achieve the goal of adapting to unpredicted application scenarios.

The rest of this paper is organized as follows: Section 2 discusses the related works. Section 3 describes the evolution approach base on scenario automation machine. Section 4 provides the example demonstration. Finally, Section 5 summarizes our work.

II. RELATED WORK

In software and system engineering, researchers pay more attention to the requirements based on scenario. They give some definitions to model and formalize requirements. So far, it lacks a universal definition. Some typical works, such as P.A. Gough [4] and J.M. Carroll [5] look the scenario as real world descriptions by using natural language, graphic methods. But A. Cockburn [6] and A.G. Sutcliffe [7] define the scenario as models such as use case and events descriptions. In the CREWS(Cooperative Requirements Engineering With Scenarios) many researchers study the scenario deeply and build a rich theory foundation. M. Jarke et al. [8] study the scenario definition, model and its evaluation from three perspectives of human-computer interaction, software engineering and strategy management. While C. Rolland[9] study the scenario classification framework from the dimensions of content, goal and life-cycle.

From these research works, the scenario has different definitions. Scenario usually is used as a static record to describe the real world task, user, system interactions and so on. Now it lacks a deterministic bounded goal and structure. Moreover, it lacks the appropriate scenario classification standard. Different granularity scenarios have different abstraction levels and complexities. For instance, larger granularity scenarios often consist of less accurate hypotheses. Multi-agent system evolution is mainly represented by the variability of the number of the agents, the adjustment of the agents interactions and their dynamic behaviors etc. Therefore, the scenarios have to be validated and corrected. To improve it, we not only look the scenario as a static record, but also put more emphasis on the dynamic record to describe the MAS evolution context.

III. Multi-agent System Evolution

A. *Scenario and Scenario Automation Machine*

We define the scenario as a description of system run-time environments and behaviors including related task, participant agents and adaptation policy etc.

1) Definition 1: Scenario= (ID, R,S, A, B, P),where ID={name, triggers } is used to identify scenarios, the elements are the primary keyword name and trigger conditions, which can be extended flexibly.

R={r_1, r_2, …, r_m} represents the role set. The role is an abstract entity with the responsibility and capability. The agent is an instance of specific role. That means the role can be looked as a set of agents with the same capabilities. A number of agents are able to play a specified role. Likewise, one agent can play many roles in different scenarios. The role assignment, inheritance and evolution is illustrated in our work[10].

S={ s_1, s_2, …, s_m } is scenario context, s_i represents the environment states.

A={ a_1, a_2, …, a_n } is the participant agents set, which usually plays the specific roles. The agent is regarded as the autonomous entity which plays the specified roles in the system. Through role binding, the agent instances can be generated from roles. **R-A binding:** it is a map from the agents to the roles, the map function is denoted as bind(R): $R \rightarrow 2^A$, 2^A is the power set of the agent A, that is to say, a number of agents can undertake one role. The role can dynamically bind to agents. The role binding matrix RB is defined as follows:

$$RB=Rs^T \times As = \{r_1, r_2, \ldots, r_m\}^T \times \{a_1, a_2, \ldots, a_n\}$$

$$= \begin{bmatrix} r_1 a_1 & r_1 a_2 & \ldots & r_1 a_n \\ r_2 a_1 & r_2 a_2 & \ldots & r_2 a_n \\ & \ldots & & \\ r_m a_1 & r_m a_2 & \ldots & r_m a_n \end{bmatrix},$$

$r_i\, a_j = . \begin{cases} 1 & \mathrm{r_i \ binds\ to\ a_j} \\ 0 & else \end{cases} \quad (1 \le i \le m, 1 \le j \le n \quad m, n \in \mathbb{N})$,In this definition, $r_i a_j$ represents whether r_i binds to a_j or not. If ra=1, agent a inherits the r's attributes.

B={ b_1, b_2, …, b_l } is the agent behavior set, which is used to describe the system functions and capabilities.

$P=P_g \cup P_a \cup P_u$ represent the scenario policy set including the goal policy P_g, action policy P_a and utility policy P_u, which shows the achieved goal, executing action plan and the executive utility.

2) Definition 2: Based on the scenario definition, we define a scenario automation machine as SA=(S,E,δ, S_0, S_f), where S is the non-empty scenario set; E is the events set for triggering scenario transformation;δ is the scenario transformation policy, which is $S \times E \rightarrow 2^S$ mapping function,δ (S_i,e)={ S_j, S_k, …,S_n}, S_j, …,$S_m \in S$; S_0 is the initial scenario set, $S_0 \subset S$; S_f is the terminate scenario set, $S_f \subset S$. The simple scenario transformation is similar to $\delta = S_0 \xrightarrow{E_{0\to i}} Si \xrightarrow{E_{i\to j}} S_j \mathrm{K} \xrightarrow{E_{\to m}} S_m$, which triggers a series of events to form the scenario transformation series. For example, the Scenarios={scenario1,scenario2, scenario3} illustrated in table I in detail.

TABLE I. SCENARIO DESCRIPTION

Attributes	**scenario1**	**scenario2**	**scenario3**
ID_Name	Sce_Class_Learning	Sce_Lab_Research	Sce_Company_Intern
Triggers	Go to class → do research in laboratory →internship in company		
G	get course credits	complete the task of scientific research	complete the project training
R	{Teacher, Student, Assistant }	{Project Leader, Student}	{Project Leader, Analyst, Architect, Programmer}
S	{ classroom, blackboard, desks and chairs, multimedia equipment}	{Laboratory, five computers}	{Corporate R&D Department, 15 networked computers}
A	{1 professor 30 students, 1 teaching assistant}	{1 researcher, 1 doctoral student, 3 graduate students}	{1 project manager, 1 analyst, 5 programmers }
B	{Teaching, Listening, Questioning, Discussing}	{Write paper, paper review, test the experiment result}	{Discussing, coding, testing}
P	{Start time 8 o'clock and end time of 8:50}∪{finished teaching content}∪{achieve a good teaching effect}	{Finish the research paper }∪{Complete finished research task}	{Complete development task}∪{Complete intern report}
…			

Based on the scenario analysis, we define the scenario agent and scenario automation machine. The scenario based MAS evolution is designed according to the defined scenario automation machine. The three scenarios transformation is shown in Figure 1.

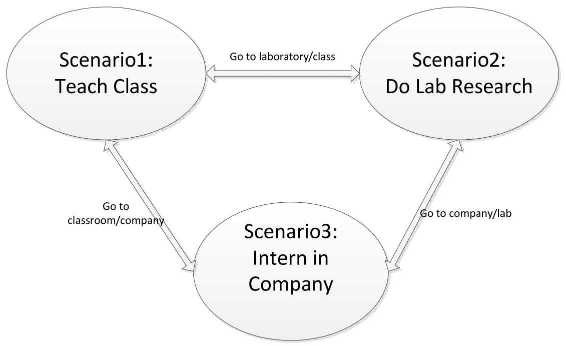

FIGURE I. SCENARIO TRANSFORMATION

When we design MAS evolution, first the system evolution features and behaviors are illustrated as a series of application scenarios. The scenario agent actually defines the agent roles, behaviors, interactions, and adaptation polices [11]. The system evolution is capable of being represented by pre-defined scenario automatic machine. It avoids centralized control because the scenario agent can be deployed dynamically with the system evolution. When the specified events occur, the system will transform to a new application scenario. The scenario automation machine guides the agent to adapt to different application scenarios. Moreover, the scenario transformation is triggered by the trigger events. It will send the message to the related application scenarios. The related scenarios then transfer to run-time states, and constrain the agent behaviors automatically according to the policies.

B. *Evolution Approach*

The scenario automation machine in fact shows the agent how to adapt to different application scenarios so as to achieve the goal of system evolution. According to the defined scenario automation machine MAS consists of a series of scenarios illustrated in Figure 2.

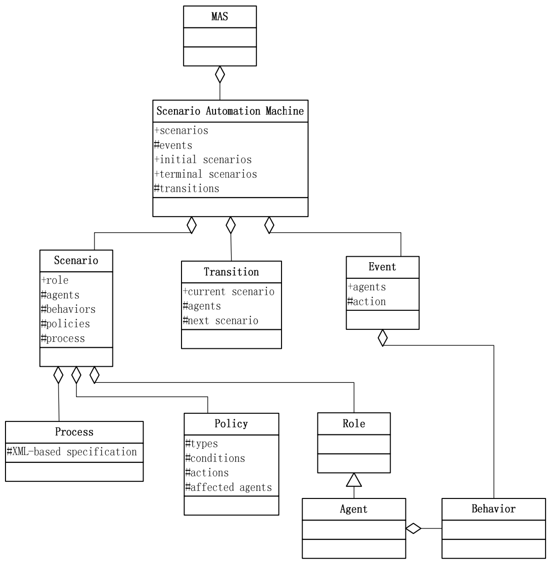

FIGURE II. SCENARIO AUTOMATION MACHINE BASED MAS EVOLUTION.

In figure 2, with the system evolvement it will adjust the agent behaviors, interactions and policies automatically. The scenario automation machine describes the agent interactions, role changes and behavior transformations. It provides larger granularity scenario-level reusability. The multi-agent system is described as MAS=(S, R, A, E , δ, S_0, S_f), where S=(S_1, S_2, …, S_n) is the set of scenarios; R={R_1, R_2, …, R_k} is the set of roles; A= {A_1, A_2, …, Am} is the se of specified agents with different roles, usually $k \leq m$; E={$E_{1\to2}$, … , $E_{i\to j}$, …, $E_{k\to n}$} is the event set; $E_{i\to j}$ is the events for S_i to S_j, $E_{i\to j}$ illustrates the trigger events from S_i to S_j, （S_i, $E_{i\to j}$, S_j） $\in\delta$**,** S_i is the initial scenario, while S_j is the terminate scenario. The concrete MAS evolution is implemented as CAS= $S_0 \xrightarrow{E_{0\to i}} Si \xrightarrow{E_{i\to j}} S_j K \xrightarrow{E_{\to m}} S_m$. The agent behaviors will change dynamically in accordance with different scenarios. And they will play different roles with the system evolution, when the agents move to or leave from a scenario, they will execute adaptation policies and various behaviors, the adaptation policies guide the agents to adapt to different scenarios. The related scenario elements will change dynamically.

IV. EXPERIMENT DESIGN

To test the MAS evolution based on the scenario automation machine. Let's take intelligent transportation systems as an example, it is divided into some scenarios such as normal traffic, traffic accident, traffic jam, traffic control etc. These scenarios and its scenario automation machine can be pre-defined for MAS evolution. The scenarios define the related vehicles, signals, driver and policemen behaviors. It provides the scenario-level development and reusability. Based on the system scenario automation machine, the MAS shows the dynamic evolution. Firstly, the transportation can be described in different agents. To simplify the system, the agents in the system are as follows.

A. *Vehicles*

Interface: enter the scenario, leave the scenario;

Data items: the road conditions (congestion, slippery etc.), location;

Interaction services: roads, traffic, navigation;

Decision policy: When the information for driving the same way is received, it response YES or NO timely; When information for coupling is received, it joins the coupling list; When the traffic jam or accidents occur, it will plan a new road etc.

Behavior: update the road information; send the road information; call the navigator or broadcast services; and plan a different road.

B. *Navigator: It is Integrated with the Car.*

Interface: Returns the current position; return the next intersection or next road; check the route with a particular intersection or road.

C. Roads: the Road Name, the Traffic Signal, the Speed Limit and the Number of Cars so on

Behavior: shut down the road, open the road, and broadcast information etc.

Decision policy: When receive the related information, execute the corresponding behaviors.

We define the scenario automation machine in figure 3.

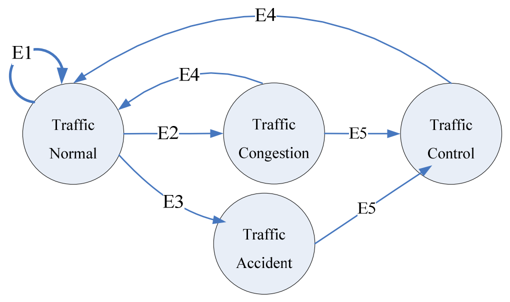

FIGURE III. THE SCENARIO AUTOMATION MACHINE EXAMPLE

The events E={E_1, E_2, E_3, E_4, E_5}, which represents the events for triggering scenario transformation, such as information broadcast, drive speed, danger signs etc. After the above analysis, we design the related scenario agent.

In the experiment, we simulated four scenarios of normal driving, traffic accidents, traffic control and traffic congestion. The system supports dynamic deployment of a new scenario. In the system, the first vehicle runs in normal driving conditions, once the scenario of the traffic congestion or traffic control happens, the migration of the vehicle based on the scenario automation machine will make decisions to change their behavior automatically, such as adjusting the speed and changing the driving route. W the system evolves the cars of the system quality attributes such as expected arrival time with the red line and actual arrival time with the blue line are illustrated in figure 4.

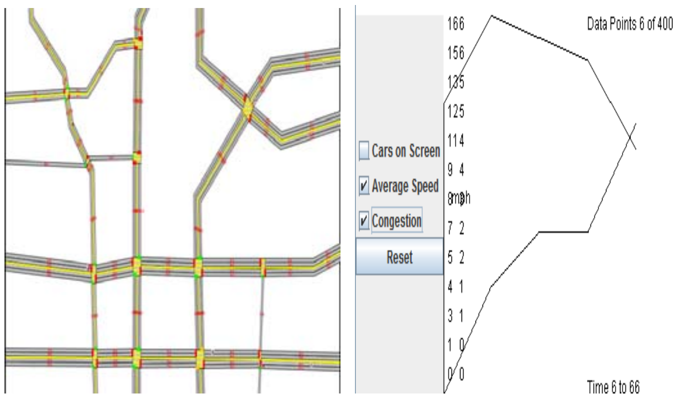

FIGURE IV. MAS EVOLUTION AND ITS EVALUATION

We test the scenario automation machine. The result is consistent with the real quality evaluation including average speed, running cars and congestion so on. When the system evolves to the normal traffic scenario, the vehicle is expected to run the similar drive time. While agents move to the traffic congestion scenario, the actual drive time is becoming longer. We also test the new optimized traffic congestion scenario, the agents adapt to the new scenario by changing its behaviors and polices.

V. CONCLUSIONS

We introduce the scenario and scenario automation machine to multi-agent system evolution. As MAS evolves, the agents adapt to the system by playing different role, executing the policies and behaviors etc. The system evolution is demonstrated by a series of scenario transformations defined as scenario automation machine. It also provides the scenario-level granularity reuse. Our future work is to test complex MAS online self-evolution and scenario operations.

ACKNOWLEDGMENT

This work is supported partially by NSFC under the Agreement No. 61073020 and the Project of CUFE under the Agreement No. 020674115002,020676114004, 020676115005.

REFERENCES

[1] Hong Mei, Gang Huang and Junguo Li, "Internet Self-Adaptation Method Based on Software Architecture," Science in China (E), 2008, 38(6), pp.901-920

[2] Zhongwen Xie,Fei Dai, "Petri-net Based Software Architecture Model for Dynamic Evolution," Journal of Computer Application and Software, 2012, 29 (10), pp.36-39.

[3] Yaodong Feng, Gang Huang, Hong Mei, "A Self-Adaptive Software Architecture Model and Implementation Method," Proceeding of Peking University(Science), 2008, 44(1), pp.67-76.

[4] P.A. Gough, F.T. Fodemski, S.A. Higgins and S.J.Ray, "Scenarios: An Industrial Case Study and Hypermedia Enhancements," IEEE International Symposium on Requirements Engineering (RE '95), Los Alamitos CA: IEEE Computer Society Press.pp. 10-17.

[5] J.M. Carroll , "Making Use: Scenario-Based Design of Human-Computer Interactions,"2000, Cambridge MA: MIT Press.

[6] A. Cockburn 2001. "Writing Effective Use Cases," MA: Addison-Wesley, 159-176, 2001.

[7] A.G. Sutcliffe, N.A.M. Maiden, S. Minocha and D.Manuel, "Supporting Scenario-Based Requirements Engineering", IEEE Transactions on Software Engineering, 1998,Vol. 24, pp. 1072-1088.

[8] M. Jarke, T. BuiX, J.M, "Carroll Scenario management: An interdisciplinary approach," Requirements Engineering, 1998, 3(1):, pp.155-173.

[9] C. Rolland et al, "A proposal for Scenario Classification Framework", Requirements Engineering, 1998, 3(1), pp.23-47.

[10] Maoguang Wang, Hong Mei, Wenpin Jiao, Junjing Jie, "Multi-agent System Collaboration Based on the Relation-web Model," Lecture Notes in Artificial Intelligence 6319, AICI'10, pp.132-144.

[11] Maoguang Wang, "Self-Adaptive Internetware Development Based on Autonomous Component, " Research Report, 2011, Peking University