

An Agent Scenario Mechanism Supporting Human/Agent Interaction*

Kai-Yi Chin, Jim-Min Lin, Zeng-Wei Hong
Department of Information Engineering and
Computer Science
Feng Chia University, Taiwan

Arthur J. Lin
Department of Business Administration
National Taipei University
Taiwan

Abstract - *The Agent Oriented Programming (AOP) combines behaviors and states of software agent with code, so it would be somewhat difficult and complex for non-programmers who are knowledge experts such as salesman or social workers. In this research, we propose an FSDL which describes agent's behaviors and reduces the difficulty and system complexity when a knowledge expert uses agent technique. FSDL supports a programmer a tool for writing a scenario representing knowledge expert's experience and knowledge. It provides a table-driven scenario editing interface for scenario designers to describe the desirable behaviors and states of software agents. The FSDL scenarios will then be transformed into the corresponding XML files which could be reusable for designing later agent scenarios. Finally, to demonstrate the feasibility of FSDL mechanism, we also set up an example system for promoting credit cards. Knowledge experts could therefore achieve the goal of driving a software agent to interact with human customers.*

Keywords: Software agent, Scenario based agent system; Feng Chia University's Scenario Description Language (FSDL)

1 Introduction

Software agents are programs consisting of mental components, such as beliefs, capabilities, choices, and commitments [1,4]. Software agent may have one or more following characteristics: autonomous, communicative, and intelligent program. Presently, software agent has been an important technology in the Internet applications. Applications using this technology may cover several domains, such as e-business, e-learning, website monitor, and so on [1-4]. Its popularity is growing up due to its well social ability and intelligence for negotiation.

However, developing a software agent system to reflect designer's goal and desirable functions is not a trivial work. One common approach to creating software agent is through the use of agent-oriented programming (AOP) [3,4]. The AOP is programming language to developing software agents, for example a BDI (Belief-

Desire-Intention) agent [8,9]. Several works in Scenario Mechanism had been developed in AOP [5-7]. Therefore, the AOP is created for developing useful agent-oriented systems to programmers [3,4]. Nevertheless, The AOP would be difficult and complex for non-programmers who are knowledge experts such as salesman or social workers. Because AOP will combine behaviors and states of software agent with codes, knowledge expert has to understand the coding technique of software agent. This would be not easy to domain experts.

In this research, we propose an FSDL (Feng Chia University's Scenario Description Language) which describes agent's behaviors and reduces the difficulty and system complexity when a knowledge expert uses agent technique. By using FSDL, software agent's behaviors and states could be easily separated from the program codes. Our proposed FSDL has three characteristics. First, using the concept of *actor*, *shot*, and *scene* in a scenario to map a knowledge expert's experience and knowledge into an agent scenario about how he performs a specific task, like doing marketing and persuading activities. Secondly, FSDL provides a structured and table-driven scenario editing interface for knowledge experts to facilitating the planning of desirable behaviors and states of software agents. Therefore, understanding of the coding details of software agent would be unnecessary. Finally, we use DTD (Document Type Definition) of XML to define necessary FSDL tags and transform FSDL scenarios into the corresponding XML files. These XML file would be reusable for designing later agent scenarios.

To demonstrate the feasibility of the proposed research, we have also set up an example system for promoting credit cards based on FSDL Mechanism. In this system, knowledge experts could drive a software agent implemented with Microsoft agent to interact with human customers. We firstly figured out and simulated the scenario and process of how a good credit salesman promotes credit cards to a customer. Then we designed an agent scenario through writing an FSDL file. The scenario could be delegated to a Microsoft agent then, and this Microsoft agent plays the role of a credit card salesman and performs jobs in behalf of this good salesman according to the designated interaction scenario and process.

* This research was partly supported by National Science Council, Taiwan, ROC under Grant NSC94-2213-E-035-017.

This paper is organized as follows: a detailed description to FSDL Mechanism is addressed in Section 2. An example system in the sale of credit card is then reported in Section 3. Finally, in Section 4, we present conclusions and further study.

2 FSDL Mechanism

FSDL scenario are used to describe the execution of agent and edited by knowledge experts. These scenarios can be delegated to agents then at run-time. The execution of software agents is according to the behaviors described in scenarios. Figure 1 shows the generation of scenarios by using FSDL.

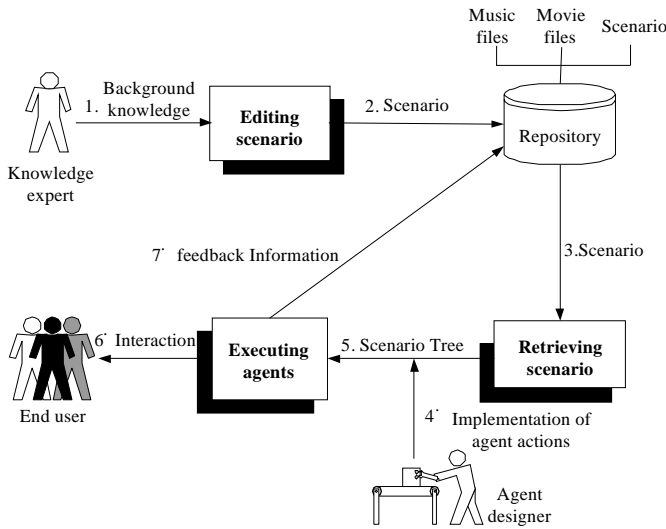


Figure 1 Generation of scenarios

In Figure 1, a knowledge expert is responsible for editing scenarios based on background knowledge through a scenario editor. A scenario would be transformed into XML documents then. The generated XML documents are stored in a scenario repository. In the repository, some relevant supporting resources, such as needed multimedia files for interacting with users, are also collectively stored in the repository. An agent designer is responsible for constructing software agents having functions of showing messages or playing media files. These software agents can interact with end users by reading the scenario retrieved from the repository. During the interaction between agents and end user, feedback information can also be sent back to the repository.

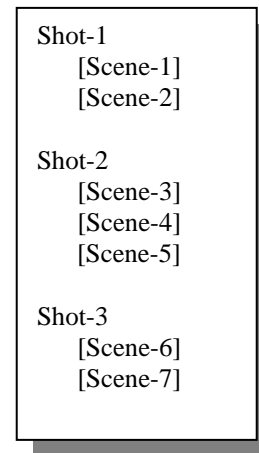


Figure 2 Structure of theater scenario

In the design of FSDL, the notion of this study derived from the structure of theater scenarios in reality. Theater editors commonly use shots and scenes to represent a theater scenario. A shot is a main section in a theater scenario. A scene is a specific section in a shot. Therefore, a theater scenario is able to contain multiple shots, and a shot is able to contain multiple scenes. Figure 2 shows a basic structure of theater scenario.

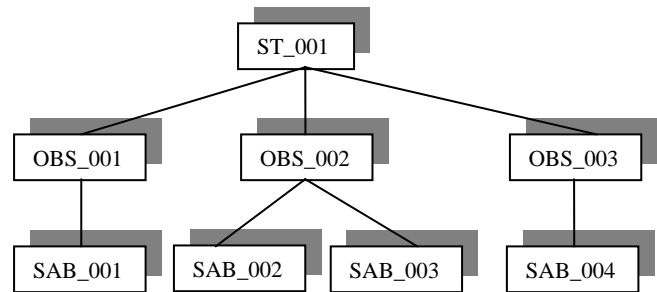


Figure 3 An example of FSDL structure

Based on the concept of theater scenario, we define FSDL scenarios in three levels (shown in Figure 3): ST (Scenario Tree), OBS (Operation Block Structure) and SAB (Scenario Activity Block). The ST, OBS and SAB are similar to a theater scenario, shot, and scene, respectively. The core of FSDL is developed by using XML technology. We will describe FSDL in the following subsections.

2.1 SAB (Scenario Activity Block)

```

<!DOCTYPE SAB [
<!ELEMENT SAB (Node_ID, Attribute ,Sequence)>
<!ELEMENT Sequence
(Provider,UI_Message,UI_Button*, UI_Multimedia)*>
<!ELEMENT Node_ID (#PCDATA)>
<!ELEMENT Attribute (#PCDATA)>
<!ELEMENT Provider (#PCDATA)>
<!ELEMENT UI_Message (#PCDATA)>
<!ELEMENT Choice (#PCDATA)>
<!ELEMENT NextSAB (#PCDATA)>
<!ELEMENT UI_Button (Choice,NextSAB)*>
<!ELEMENT UI_Multimedia (#PCDATA)>

```

Figure 4 XML DTD of SAB

In the SAB level, some agent's behaviors about the interaction between agents and human are described. These behaviors may include interaction activities like showing text messages or interactive buttons, and playing media files. In order to represent various kinds of interactions, several labels are defined in SAB files. Figure 4 shows the XML DTD (Documents Type Definition) of SAB files and Table 1 shows the description of each element in DTD of SAB.

Table 1 Description of element in SAB

Label	Description
Node_ID	Identifier of this file.
Attribute	The feature for classification
Provider	Agent that provides resource.
UI_Message	Messages that are showed to humans.
UI_Button	Buttons that are showed to human.
Choice	Selections in UI_Button
NextSAB	Next SAB to be executed base on Choice
UI_Multimedia	Multimedia files showed by the agent.
Memo	Other information wrote by scenario editor.

2.2 OBS (Operation Block Structure)

The second level shown in Figure 3 is OBS. Knowledge experts can use OBS to manage a number of SABs. They may make a selection and a proper order of several SABs to represent a meaningful interaction, such as

greeting with human operators. Thus, an OBS could be used to describe a group of behaviors defined in SABs for an interaction. In an OBS file, the SAB files managed by this OSB files are described. Consequently, an OBS file may manage several SAB files, and each SAB file may be shared and invoked by several OBS files. By this way, an SAB file could be reusable. Table 2 shows the description of each element in an OBS file.

Table 2 Description of element in OBS

Label	Description
ID	Identifier of this file.
Attribute	The feature for classification
SAB_members	Sequence of SAB files
Memo	Other information wrote by scenario editor.

2.3 ST (Scenario Tree)

The top level in FSDL is ST. It focuses on assembling several OBSs. Because a scenario of interaction between agents and human may consist of several meaningful interactions, ST may describe some information about the order of OBS. The sequence of OBS files is described in a label of the ST file. Table 3 shows the description of each element in ST files.

Table 3 Description of element in ST

Label	Description
ID	Identifier of this file.
OBS_members	Sequence of OBS files
Memo	Other information wrote by scenario editor.

3 An Example System

In order to demonstrate the feasibility of the FSDL, we implement an example system for promoting credit cards. In the system, agents are responsible selling credit cards to potential human customers through interaction. The FSDL scenario about promoting credit card are firstly edited by knowledge experts and then used to provide the control of agent's behaviors. Figure 5 shows the conceptual model of this example system.

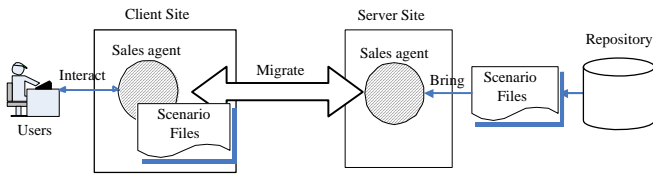


Figure 5 Conceptual model of the example system

In figure 5, this system consists of two main components:

- **Repository.** In the system, the repository is a database providing some required run-time resource to agents. For example, it may contain the multimedia files or predefined scenario materials, such as SAB, OBS and ST files. Sales agents could then access these necessary data from repository anywhere.
- **Sales agent.** The sales agent is actually a program that interacts with users to promote credit cards. It gets the scenarios from repository for promoting credit cards and then performs the behaviors described in SAB files. In the system, the sales agent may be implemented in form of a mobile agent. Therefore, the sales agent migrates to the client site and makes the interaction with users according to the pre-defined behaviors.

Table 4 Description of SAB_001

Tag	Content		
Node_ID	SAB_001		
UI_Message	Do you satisfy with your credit card?		
UI_Button	Great!	So So!	Terrible!
	SAB_002	SAB_003	SAB_004

In the example system, FSDL scenarios are used to provide a way for scenario designers to control agent's behaviors. Table 4, Table 5 and Table 6 show the example of SAB, OBS and ST files respectively.

In order to promote credit cards to an end user, a sales agent may need to perform some promotion behaviors in the interaction. For example, the SAB_001 in Table 4 describes the behavior of an agent to show a brief message about a promoted credit card and some feedback choices for users' selection. These choices may change the execution flow to other SABs, such as SAB_002, SAB_003 or SAB_004.

Table 5 Description of OBS_001

Tag	Content
ID	OBS_001
SAB_members	SAB_001, SAB_002, SAB_003, SAB_004, ...

In a meaningful interaction, sales agent may perform a number of behaviors described in SAB files. Related

SAB files in an interaction are described in an OBS files. For example, the OBS_001 in Table 5 concerns an interaction about the inquiry of satisfaction of credit card. The interaction may involve several SAB files described in SAB_members tag.

Table 6 Description of ST_001

Tag	Content
ID	ST_001
OBS_members	OBS_001, OBS_002, OBS_003

Because a whole scenario of promoting credit card may include several meaningful interactions, a number of related OBS files are described in the OBS_members tag. For instance, the ST_001 in Table 6 is the scenario tree of promoting credit card. The scenario in ST_001 concerns three meaningful interactions in some OBS files, including OBS_001, OBS_002 and OBS_003.

Base on the FSDL, all the scenarios are developed in the form of XML document. Figure 6 shows the XML document of SAB_001. The entire labels in SAB are translated to corresponding XML tag according to predefined DTD. The sales agent could make interaction with end user by referring to these XML document. Figure 7 shows the screen shot of sales agents. In the implementation, the MS-Agent technology and Java button are adopted to provide the representation of brief message and choices respectively. Therefore, the sales agent could interact with end user to promote credit card with scenarios and some user interface technology.

```

<SAB>
  <Node_ID>SAB_001</Node_ID>
  <Sequence>
    <UI_Message>Do you satisfy with your credit
card?</UI_Message>
    <UI_Button>
      <Choice>Great!</Gchoice>
      <NextSAB>SAB_002</NextSAB>
    </UI_Button>
    <UI_Button>
      <Choice>So So!</Choice>
      <NextSAB>SAB_003</NextSAB>
    </UI_Button>
    <UI_Button>
      <Choice>Terrible!</Choice>
      <NextSAB>SAB_004</NextSAB>
    </UI_Button>
  </Sequence>

```

Figure 6 An example XML document of SAB

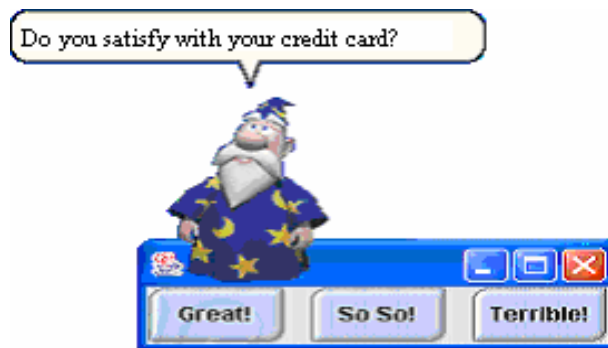


Figure 7 Screen shot of a sales agent

4 Conclusions

Software agent has become an attractive modern information technology. It has the potential ability of performing lots of sophisticated tasks on behalf of human beings. However, in practical applications, an efficient and convenient agent process operations and human/agent interface is still scarcely seen. In this paper, we try to propose an FSDL to describe agent's behaviors and the associated tools to help users to efficiently design and manage agent's behavior. In our future study, we will try to enhance software agent's learning capacity by introducing the machine learning concept. Additionally, we will also focus on FSDL robustness and completeness to help knowledge experts to accurately plan agent's behavior.

5 References

- [1] Michael J. Wooldridge and Nicholas R. Jennings, "Intelligent Agents: Theory and Practice," *Knowledge Engineering Review*, Vol 10, pp.115-152, 1995.
- [2] M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice," *The Knowledge Engineering Review*, Vol 10, pp.115-152, 1995.
- [3] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence*, Vol 60, pp. 51-92, 1993.
- [4] Rui Shen, Ji Wang and Hong Zhu, "Scenario Mechanism in Agent-Oriented Programming," *11th Asia-Pacific Software Engineering Conference (APSEC)*, pp. 464-471, 2004.
- [5] G. Weiss, "Multiagent Systems: A Modern Approach to Distributed," *Artificial Intelligence*, MIT Press, 1999.
- [6] Hong Zhu and David Lightfoot, "Caste: A Step Beyond Object Orientation," *Lecture Notes in Computer Science*, Vol 2789 / 2003, pp. 59-62, 2003.
- [7] Hong Zhu, "SLABS: A formal specification language for agent-based systems," *International Journal of SEKE*, Vol 11, pp. 529-558, 2001.
- [8] Anand S. Rao and Michael P. Georgeff, "Modeling Rational Agents within a BDI Architecture," *Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pp. 473-484, San Mateo, CA, USA, 1991.
- [9] Anand S. Rao and Michael P. Georgeff, "BDI-Agents: From Theory to Practice," *First International Conference on Multi-Agent Systems (ICMAS'95)*, San Francisco, USA, June 1995.