International Journal of Grid and Distributed Computing
Vol.7, No.1 (2014), pp.99-112
http://dx.doi.org/10.14257/ijgdc.2014.7.1.10

Behavior-oriented Modeling and Visualization for web Service Composition

Lin Li[1,2,3]

[1]*School of Computer, Wuhan University, Wuhan, Hubei, China 430000*
[2]*College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, Hubei, China 430000*
[3]*Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, Hubei, China 430000*
lilinwhu@gmail.com

Abstract

With widely used at present, web service can enhance not only the reusability of code, but also the utilization rate of the service resources shared on the web services composition. To solve the difficulty in the communication and expression of the requirement information in complex web service composition, this study proposed a visualization method to describe service composition models through animation. This method employed the requirement behavior model established by the behavioral description language as the research object. In the execution mechanism of the animation, state machine and state block were utilized as the execution model and description model respectively. Each state migration was corresponded to a behavior that calls for visualized description. The motion of figure was controlled by associating each migration in the state block model with an action primitive through an association element. The visualization of the requirement behavior model was thereby realized. The advantages of the method lie in that the requirement changes are avoided. This attributes to that the missing and incorrect potential requirements are observable for the users with different backgrounds owning to the intuition of animation. Moreover, the correctness of the requirement animation is ensured through a series of detection on the BDL model. Finally, this study empirically analyzed the method using a travel system and introduced the modeling and animation tools it realized.

Keywords: *Behavior Description Language, Behavioral Model, Visualization, web Services Composition*

1. Introduction

Requirement is the first stage in the whole software lifecycle. Requirement modeling [1,2], aiming at establishing and detecting the requirement model of software system from different angles in the requirement analysis process, is definitely a key problem and also one of the most complex processes in software engineering currently [3, 4]. The requirement model obtained can provide a platform for the communication and verification on the correctness of requirement information to achieve high-quality requirement. The goal of requirement engineering is to acquire high quality software requirements. One of the key processes in requirement engineering is to model requirement information, which is utilized to build a model of the target system according to the requirements information before the requirements are confirmed. Requirement modeling provides a way of communication between the software stakeholders and developers. The goal of requirement modeling is to determine the

ISSN: 2005-4262 IJGDC

requirement information of software system, that is, to determine "what does the whole system want to do", and establish corresponding requirement model using certain modeling method. "What does the whole system want to do" is justly the software behavior [5]. It is refers to the processes of application, operation, and motion of the subject to the object according to the functions of subject, or subject applying a service, operation, or motion to the object in the running process. Here, the software system itself is the subject. Software behaviors are composed of the subject and object of the behavior, the operation or movements, behavior input-output, and behavioral attributes *etc*.

As software becomes increasingly complex, the requirement model of software is given increasingly prominence. Requirement visualization technology provides a more intuitive and easier understanding way for the complex systems that are difficult to be perceived and expressed in words. Visualization technology can transform the invisible objects, expressions, or abstract concepts into symbols using technologies such as images or pictures [6]. The composition of visualization technology and requirement model offers the requirement engineering field of software with new opportunities and a bright future.

Considering the present research status of related studies, this study put forward a requirement visualization technology. In this technology, the dynamic behaviors in the requirement model are vividly and intuitively displayed by the dynamic variations of figures and symbols. By simulating the main execution process of target software and executing the animations of behavior model, users with different knowledge backgrounds have access to understanding the requirement behavior model and finding the missing and incorrect potential requirements. Valuable feedbacks are thereby acquirable from users. The requirement visualization technology greatly reduces the requirement changes, as well as the complexity and cost of software development.

The structure of this paper is organized as follows: Section 2 introduces the formal syntax of behavior description language, which is the basis of requirements visualization. Section 3 introduces the method of requirements visualization modeling and Section 4 demonstrates the modeling process through the case study of travel system. Finally, the related works are discussed in Section 5 and the conclusions and future works are discussed in Section 6.

2. Background and Motivation

2.1. Web Service Composition

As a module deploying on web, web service is established based on the XML language-centric specification technology. Moreover, it is discovered and called through web and can be interactively operated on web. web service is an application logic or function module that can be identified by uniform resource identifier (URI). Moreover, it is observable by other services through describing its public interface exposed outside using XML language. Moreover, supported by Internet protocol, the services are allowable to communicate with each other via XML-based information [7]. To support the interactions among the software in a way of being independent from platform, web service calls for the supports of some core protocols in the whole interaction process, mainly including simple object access protocol (SOAP) [8], web services description language (WSDL) [9], and universal description discovery and integration (UDDI)[10]. Under the web service-based distributed calculation mode, the usable calculation resources in the system in form of web service modules can be submitted through the standard interface at the bottom layer and used to develop the distributed application program at low cost rapidly.

Single web service is incapable of meeting the complex user demands and constituting into complete business application due to the limit business functions. Therefore, to more

effectively complete the user demands, it is necessary to combine these single service modules. The powerful functions realized by the combinations enhance the reusability of code and the utilization rate of the service resources shared on the web. Moreover, the web service combinations formed, which can be used as final application, are also new web services [11]. In view of service function, web service combination is to realize the overall function appreciation by reusing combination function or the service modules with relatively simple business [12]. In perspective of workflow, web service combinations achieve the complex collaborative cooperation by determining the execution logics and sequences of different service modules. Statement on the concept of web service combination above suggest that, web service combination is designed to realize the overall function improvement and cost reduction of service by effectively combining the modules with different functions.

2.2. Business Process Execution Language

BPEL is a specification formulated for the integration of web service. It gives enterprises way to describe the complex business process involving web service. Moreover, using this specification, the work flow combined by web services can be further packaged into web service of higher level and then issued outwardly. BPEL is free of any storage and processing of business data, as well as the execution of any details in the business process. The detail operation of the business data and process are all realized by web service. BPEL is only a description of at what time, in what order, and going where to call the web service, as well as how to organize these calls. In the W-based workflow realized by BPEL, web services are allowable to be selected, integrated, and nested in real time. In such way, enterprises can freely call web services from a global perspective to actually guide business and share information rather than be confined in the individual advantages of web service.

BPEL is a XML–based work flow definitional language developed from traditional programming language. The application program established by BPEL can be thought as a common XML text in case of neglecting the program language characteristics of BPEL. The elements in BPEL, just like the keywords in traditional programming language, constitute the BPEL program by combination.

2.3. Problem Statement and Motivation

On the surface, web service is a self-contained, self-described, and modularized business application program. In essence, web service is a new platform used to build the interoperable distributed application programs. Since it is based on the SOAP protocol of XML, it realizes the mutual communication between heterogeneous systems and eliminates the differences in different module models, operation systems, and program language systems. Therefore, application program can be formed by the web services with differ sources. By interactively calling these services, a general and platform-independent technological layer is yield. Moreover, in this process, it is unnecessary to know the locations and realization ways of these services. The mutual operation and integration of application programs on different platforms can be achieved using this technological layer.

Obviously, visualization is quite applicable for web service technology. In the web service technology, each several module behaviors can be combined into a web service. By distributing and running the web services combined on different machines or platforms, resource sharing is realized. Meanwhile, the exploration and combination of the support service of web service technology facilitate users to find the most suitable visualized resources. The resources found are formed into visualization channels and then processed to be visualized to realize the interactive calling and collaborative work among different resources. Therefore, it is possible and effective to apply the visualization technology into web service combination system.

3. Behavior-oriented Modeling Language

A behavior is a certain interaction among two or more entities. For easy discussion, this paper presumes a behavior is an interaction only between two entities. We define a software behavior as a process during which a subject implements an operation, service, or action to an object. The subject and the object which may be physical or logistic, can be a person, a software or hardware component of system, or certain element of environment.

The structure of each behavior consists of a subject, an object, some properties, some inputs, some outputs, and an operation, service, or action. If a behavior can't be divided into two or more sub-behaviors, it is an atomic behavior. An atomic behavior is a simple behavior. Two or more simple behaviors form a composite behavior. In addition, according with the interact mode of software behaviors, the combine pattern of simple behaviors can be divided into five categories: sequence, certainty choice, uncertainty choice, parallel and shielding.

Based on the above consideration about software behavior, the followings are the syntax of behavior based requirements modeling language --Behavior Description Language.

Suppose *ABehID*, $ABehID_i$ (i∈N) are atomic behavior identifier, *BehID*, $BehID_i$ (i∈N) are behavior identifier.

3.1. Atomic Behavior Expression

ABehID: f (sub, obj[& notes about obj]) [(x_1, …, x_n)]
[When *prepositive conditions*]
[InFrom (IID) (u_1, …, u_n)]
[OutTo (OID) (v_1, …, v_m)].

where *f* is a service, an operation or an action. The subject *sub* commits *f* to *obj*; x_1, x_2, …, x_n are variables; *InFrom* clause means this behavior takes input data u_1, u_2, …, u_n from IID. IID could be an identifier of another behavior, IO Pool or external entity. *OutTo* clause means this behavior provides output data v_1, v_2, …, v_m to OID;

1) Idle behavior

ABehID: Idle //Idle behavior means nothing occurs.

2) Return behavior

ABehID: Return () //A return behavior is used when a composite behavior is about to exist or jump to a specific behavior.

3.2. Composite Behaviors and Behavior Expressions

Let α-denotes that "the string α is a valid behavior expression". We can define behavior expressions as follows:

a) The identifier of an atom behavior is an expression.

$$\vdash ABehID$$

b) Sequence behavior:

1) $$\frac{\vdash ABehID_1 \,\&\, \vdash ABehID_2}{\vdash ABehID_1; ABehID_2}$$ 2) $$\frac{\vdash ABehID \,\&\, \vdash BehID_1}{\vdash ABehID; BehID_1}$$

3) $$\frac{\vdash BehID_1 \,\&\, \vdash ABehID}{\vdash BehID_1; ABehID}$$ 4) $$\frac{\vdash BehID_1 \,\&\, \vdash BehID_2 \,\&\, ...\,\&\, \vdash BehID_n}{\vdash BehID_1; BehID_2; ...; BehID_n}$$

c) Conditional behaviors：

$$\frac{|-BehID_1 \& |-BehID_2 \& b}{|-If\ b\ Then\ BehID_1\ Else\ BehID_2\ Fi}$$

where b is a Boolean expression.

,

d) Choice behaviors

$$\frac{|-BehID_1 \& |-BehID_2 \& .....\& |-BehID_n}{|-BehID_1 + BehID_2 + ... + BehID_n}$$

e) Parallel expressions

$$\frac{|-BehID_1 \& |-BehID_2 \& .....\& |-BehID_n}{|-BehID_1 \,||\, BehID_2 \,||\, ... \,||\, BehID_n}$$

4. Behavior Description Language based Visualization Model

After the establishment of behavioral model, it is needed to determine whether or not the behaviors described by the model are those of the objective software system in the expectation of users. To effectively solve this problem, the visualization technology, which is capable of transforming the behavioral model into vivid and dynamic executing process, can be adopted. By a direct vision on the dynamic execution process of requirement, the users bearing different knowledge backgrounds can clearly perceive the behavior process in the requirements and thus are more urgent to detect the requirements. The effective user's feedbacks increased thereby improves the efficiency of requirement analysis and reduces the requirement changes in the following stages of software development.

Before employed in requirement animation, the behavioral model of requirement established by BDL should be firstly transformed into state machine using model transformation method. Subsequently, basing on the engineering requirements, it is required to select the entity sets described by requirement visualization, as well as corresponding pictures or pictures to directly express the entities selected. According to the entity set selected, the corresponded state blocks in the presentation range are extracted from the model using state extraction method. Besides, the migrations in each state block, which are associated with the movement primitives, are used as the visualization description of behavior. In the execution of state machine, the driving animation engine calls correlated movement primitives according to the migration of the state blocks to control the movements of the picture correlated with object. The process of the behavior described by requirement model is visually reflected finally.

In short, requirement visualization is conducted in the following steps: 1. building the requirement model according to the behavior description language BDL; 2, extracting the behavioral model in the requirement model; 3. transforming the behavioral model into animation description model; 4; converting the animation description model into requirement animation by execution engine.

4.1. Setting the Content and Usage of Animation Elements

On basis of the process of requirement model, we have access to a BDL statement-based requirement behavioral model. According to the subject and object selected in behavioral model, it is in need of setting the animation elements and movement "stage background pictures" of the subjects and objects. The "stage background pictures" refer to the background pictures of the objects and subjects in animation. They include all the motionless backgrounds

in the animation. The animation elements of subject and object are composed of the names, related pictures, types, initial x coordinates, initial y coordinates, and the depths of display hierarchy of subject and object. The content and usage of animation elements are shown in Table 1. The animation elements of each subject and object are set by a way of "relevance picture" one by one.

Table 1. The Content and Usage of Animation Elements

animation elements	content and usage
The names of subject and object	the names of the objects in the object list filled by animation designers
Correlation figures	The figures associated with subject and object. They are designed by the animation designers according to the specific characteristics and behaviors of objects to clearly express the visualization of animation objects to the maximum extent.
Types	the static state or dynamic state of the correlation figures; static figures are displayed on the beginning of the animation, while dynamic figure are dynamically presented by the animation primitives.
Initial x coordinate	the initial abscissa of the position of the top left corner of correlation figure on the background figure.
Initial y coordinate	the initial ordinate of the position of the top left corner of correlation figure on the background figure.
Display hierarchy depth	the hierarchy of figure in the animation. Hierarchy determines the display state of the overlapped figures. The figures in deep hierarchy are covered by those in shallow hierarchy. The hierarchy of object is designed according to the effects of animation.

4.2. Editing Behavioral Animation

Behavioral animation refers to the sequence of the animation operation primitives corresponded to the atomic behaviors of animation BDL model. Driven by the BDL model, these sequences give rise to a complete animation in the smallest constituent part of single atomic behavioral animation. In the behavioral animation, the displaying and movement of pictures are controlled by simple primitives. The sequences of these primitives yield a single atomic behavioral animation, while the basic movements of the objects in the animation are expressed by several basic movement primitives.

To more vividly reflect the executive effect of each behavior, the behaviors in the model should be provided with appropriate parameters by combining the primitives as needed. During the execution, the migration in the state blocks, which is trigged in accordance with the time series stated by semantic, drives the animation engine to interpret the animation primitives of the execution. The behavior process of target software described in the model is thereby dynamically reflected. Table 2 lists the movement primitives and related functions.

The behavioral animation is designed based on the movement, *subject, object, infrom, outto* in the atomic behavioral expressions. By the compositions of animation primitives above, desired animation effect is achieved. Once the behavioral animation of one atomic

behavior is complete, can the text boxes named by corresponding behaviors be filled with corresponding behavioral animation primitive sequences.

MOV primitives and STOP primitives are temporally valued according to the needs of synchronization and animation effect, while SEND primitives are mainly used to dynamically control the flow of control model in the animation execution by valuing the control process expressions in behavioral migration. The primitives in Table 2 give a simple and direct description on the execution of software. Here, the animation expression on "animation software module A sending the message m to module B" is exampled. Firstly, software modules A and B are represented by the pictures of two identification software modules respectively, while abstract message m is expressed by the message blocks bearing text prompt. Subsequently, message block m is moved from A to B using MOV primitives.

Table 2. Animation Primitives and Corresponding Functions

Primitives	Functions
MOV(object, coordination x, coordinate y, millisecond t)	this primitive represents that it takes t millisecond to move the figure of the object to the coordinate(x, y)
RESHOW(object)	reshowing the object, that is to say, redisplaying the correlated figures on the original position.
SHOWP(object, figure name)	showing the object into the figure corresponded to the figure name.
SHOWXYP(object, coordination x, coordinate y, figure name p)	showing the figure name associated with the object on coordinate (x, y).
DIS(object)	this object is hidden, namely, the correlation figures of the object are hidden.
STOP (millisecond t)	ending the thread of the present behavior at t millisecond.
SEND(if expression, true-false value)	the values of if expressions are set as corresponding true-false value, 1 represent true; 0 refers to false; 2 denotes randomness.

4.3. Setting if Branch Values

If expression is a conditional statement in BDL model that used to determine which branch behavior happens. In the dynamic running process of animation, it can be valued as true or false. Since the value of if expression should be determined in the operation of animation, it is necessary to set if expression values. In addition, in case of encountering if branches in the initial setting of animation model, the initial value of if expression can be set as random, true, and false. The random therein refers to the random selection in true or false when animation model proceeds to if branches, while the true or false are with fixed in values of true or false.

After setting stage background, subject, object, and if branch values and compiling behavior animation, the execution model of animation should be built in case of need in running animation.

4.4. Animation Execution Engine

Behavioral model describes the implementation process of software. The behavior of each atom indicates the interaction between a subject and an object. However, to offer users with direct model variations, the state changes of every animation entities should be perceived clearly before expressed by the physical variations of pictures in the animation description. In this study, state machine and state blocks are used as the execution model and description mode of animation respectively. In accordance with the objects should be focused on, a set of

state blocks can be extracted from state machine, with each state block being obligated to express the state migration model of one object. To display the object in the animation, the object is needed to be correlated with a picture, while migrations of this object should be associated with a set of movement primitives that has been designed to describe the picture movements (as shown in Table 2). In such way, the direct visualization on the state changes of the object is achieved.

The behavioral model is converted into a state machine. Then we select the state blocks that describe the behavior and state changes of every objects and logic units from the state machine. The state machine is a execution model that drives animation engine to yield animation movements, while the state blocks, which indicates the state change process of each object, is the description model being responsible for animation script description. Being similar with state machine, the state machine and state blocks in the formal model of state blocks are also internally described by XML documents to facilitate the execution interpretation of animation execution engine. The state block of each object is described by a XML document named with ID. The XML document only used to record the migration structure of the state blocks instead of design the correlation information between object and picture, as well as migration and movement. Therefore, the model is independent from correlation. Such design facilitates the independent modification on model and animation on one hand. On the other hand, in case of requirements, one model can be made with multiple requirement animations merely by adding new relevant documents.

Each state migration is corresponded to a behavior calling for visualization. Each migration in the state block model can be associated with a movement primitive by a correlation element. In the execution of requirement animation, the state migration of state blocks driven by animation execution engine are corresponded to the movement primitives in association lists. By inquiring these movement primitives, picture motion can be controlled.

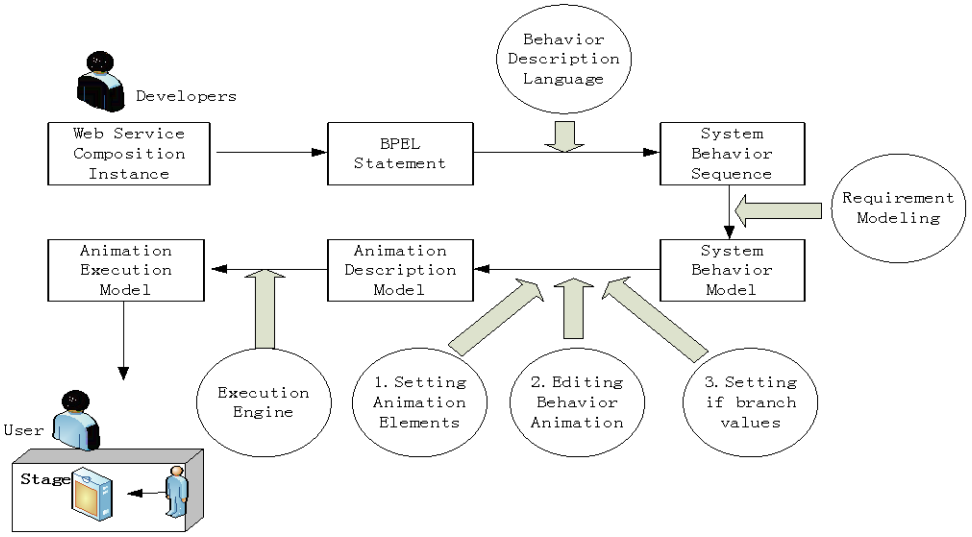


Figure 1. The Execution Mechanism of Animation

Figure 1 shows the execution mechanism of requirement animation. The specific implementation steps are explained as follows:

(1) Arranging the initial information of animation stage according to the animated background, the correlation information of object and picture, and the initial coordinate of pictures described by the XML documents in object visualization.

(2) Starting the state machine that drives the execution of animation and preceding the execution according to the temporal structure of the migration of state machine.

(3) As long as there is a migration of state machine in the execution path, execution engine will check whether or not there is a corresponding state block. If there is, execution engine will drive a corresponding migration of the state block.

(4) With the migration and skip in state block, execution engine will inquiry and call corresponding movement primitives. Movement primitives control the state variations of the pictures on stage and generate animation effect.

Since an atomic behavior in the behavioral model is in correspondence with an event migration in state model, the atomic behavior of the migration under execution can be synchronously displayed in the execution of state machine. In this way, users have a direct visual access to the synchronous execution process of requirement model and requirement animation.

4.5. Animation Demo

With the running of animation, the system enters into an animation demo interface. This interface is consisted of three parts, namely, "animation-running window", "behavioral trace window", and "BDL model window". "window-running window" is the main window. It is in charge of controlling the start, pause, and restart of animation and displaying the animation driven by BDL model. "Behavioral trace window" presents and records the traces of atomic behaviors in the running process of animation and highlights the behavior under operation in red. According to the traces and the animation effects, the requirements of target system can be investigated by system analysts and users. The "BDL model window" in the left-hand side highlights the position of atomic behavior under operation in red and synchronizes the position with behavioral traces and animation.

To realize the visualization on the behavioral model, it is needed to transform the behavioral model into state model firstly. Behavioral model formally describes the interaction of target software systems with problem domain and some behaviors in the internal modules of software system, while state model mainly focus on describing the state changes of individuals under certain abstract level in the system. To detect the consistence of behavioral model with state model, [13] proved the correctness of the rule of the transformation from behavioral model to state model. It investigated the strong bisimulations of behavioral expressions and state structural formulas, as well as the relationships between the bisimulations of behavioral expressions and state structural formulas. The result verified the equivalence of the bisimulations between behavioral model and state model.

5. Case Study

In this study, a travel system is exampled to further explain the method and process of constructing behavior model and making service composition animation.

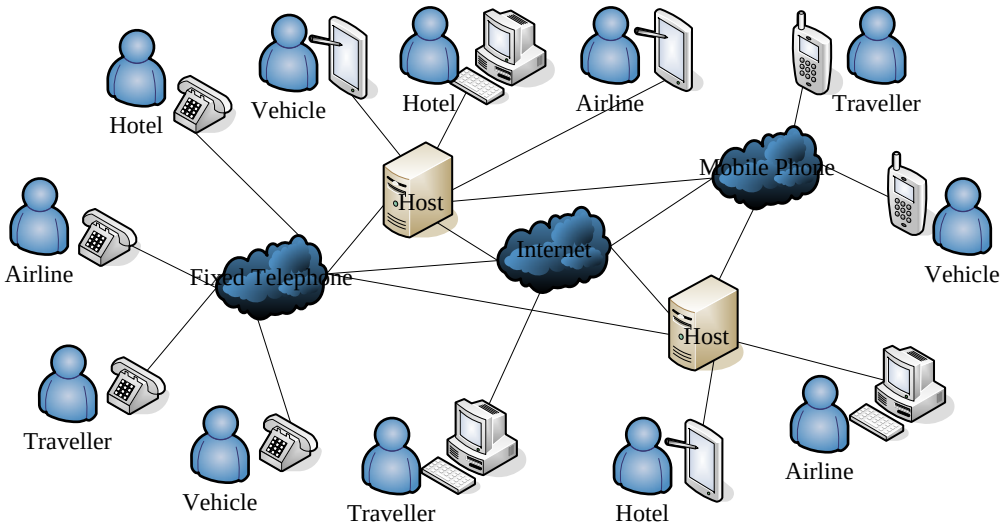


Figure 2. Web Services Composition of Travel System

Firstly, we identify and divide the system's problem domain. The system consists of three layers, the first layer is the system's direct customers, for example travellers, airlines, destination hotels and transport vehicles, they interact with the system; the second layer is composed of terminal, network communications system, fixed telephone systems and other peripherals, such as SMS gateway etc., which is responsible for providing first layer customer interface, receiving users' requests and inputs and transferring data to the server; the third layer is the core part of the system, that is the host. It is responsible for the storage, handling a variety of all kinds of request data and the corresponding peripheral equipment, and the results are returned to the appropriate equipment. Thus, according to the hierarchy of the travel system, we can divide the system into two problem domain.

Server problem domain: the host;

Client problem domain: terminal, network communications system, fixed telephone systems and other peripherals, such as SMS gateway etc.

Aiming at these problem domain, we can respectively identify several viewpoints as follow：

A. For server problem domain:

a) Response Network Communications System equipment viewpoint (VP_Network Communications);

b) Response fixed telephone systems viewpoint (VP_PhoneSys);

c) Send messages viewpoint (VP_Message);

d) Response Terminal viewpoint (VP_Terminal);

e) Server work viewpoint (VP_Server).

B. For client problem domain:

a) Traveller inquiry viewpoint(VP_traveller);

b) Airline work viewpoint(VP_Airline);

c) Destination hotels work viewpoint(VP_Hotel);

d) Transportation vehicles query viewpoint(VP_Vehicle).

After viewpoint identification, project manager specified one person in charge for every viewpoint, who is responsible for requirements' editing in viewpoint. Based on these analyzes, we establish corresponding behavior model.

Since the whole system is complex, due to space limitations, this paper gives only a part of BDL description for travel system.

The natural language description of Response Traveller Inquiry scene is as follow:

1. Server receives the Query records, which sended from traveller terminals;
2. Server verifies the validity of traveller information;
3. Server sends Validation results to traveller;
4. If the query is valid, then the server saves basic information record of trip for traveller.

Figure 3 lists The BDL description of Network Communications System Viewpoint.

```
VP_NetworkCommunications:          //Network Communications System viewpoint
VPBEGIN                            //viewpoint description start
                                   //scene of responding Traveller Inquiry and sending message
BehTravellerInquiry
   BEGIN  //Scene description start
   ABEH  //Atomic behavior
    RecTravellerInquiryInfo: receive(Server, Query records)
                             INFrom(Traveller)(Query records).
    ValidTravellerInquiryInfo: verify(Server, Query records)
                               OUTTo(Traveller)(Validation results).
    SendValidTIResult: send(Server, Validation results)
                       INFrom(Server)(Validation results)
                       OUTTo(#VP_NetworkCommunications)(Validation results).
   StoreTravellerInquiryInfo: save(Server, Query records) .
   Return1:return(). //Exit the system
   BEH  //Complex behavior
        //Expression of behavior in response to the query scene
   BehTravellerInquiry = RecTravellerInquiryInfo;
               ValidTravellerInquiryInfo;
   SendValidTIResult;
   If(Validation results = Valid query)
   Then
      StoreTravellerInquiryInfo
   Else
   Return1
   Fi;
   Return1. // Exit the system
   END  //The scene description of receiving messages end
   VPNetworkCommunications = BehTravellerInquiry
VPEND
```

Figure 3. The BDL Description of Network Communications System Viewpoint

After BDL mode is established, the figure association and initial coordinate of the subjects and objects of the BDL model can be set. Subsequently, the animation scripts of the behavior

are edited using animation primitives (animation editing). Finally, animation is produced and showed.

This study realizes the modeling and animation-making tools basing on the software behavior and multi-view web service-based using Microsoft Visio. The modeling method proposed, which is also based on software behavior and multi-view web service, is disparate with some existing demand modeling methods and technologies. This attributes to that the characteristics of complex software systems and the correctness of software behavior determine whether or not the software can satisfy users' demands. Moreover, the characteristics of software, such as reliability, can be verified by software behavior. Although the software system in demand phase has not been developed yet, it is very important and practical to build strict behavior-based demand model by analyzing the demands of the software in demand phase. On the basis of the behavior model generated by modeling tools, demand animation can be produced using animation tools by writing the animation elements and primitives. Moreover, animation tools provide open motion base and action parts base. According to need, users can add new actions and action parts in the bases via the making interface.

6. Conclusion

This study presented a visualization method to describe the behavior model of W service combination using animation and the system behavioral sequence described by BDL. The behavior model is established in the premise of keeping the correctness of the grammar of the BDL statement. The visualization of the demand behavior model was realized by setting related contents, including animation elements (subject and object *etc.*) and the correlation between atomic behaviors with related animation primitives.

Concerning the animation execution mechanism, state machine and state blocks are taken as the execution model and description model of animation respectively. Each state migration is corresponded to a behavior that needs to be visualized, while each migration in the state block model is correlated with an action primitive using a correlation element. The motion of the figure is thereby controlled.

This method expresses the demand model dynamically. It employs the figure and images in reality as animation execution elements and drives the execution of animation elements by demand model. The animation yield by these tools is perceivable by nonprofessional users and effectively enhances the communication between users and developers.

In the demand visualization, the execution model and description model of the animation are transmitted from the behavior model. Such transition needs less artificial participation. Moreover, it is verifiable of the consistence between the demand animation yield and the demand model. By the demonstration of demand animation, users have an intuitive way to understanding demand model. By checking whether or not the target software behavior described is in agreement with the willingness of users, more effective feedbacks can be obtained from users. Finally, the analysis efficiency is thereby improved and the development costs are reduced in turn.

The advantages of the method proposed lies in that demand changes are avoided. This attributes to that the intuition of the demand animation gives users way to exploring the missing and incorrect potential demands. Moreover, due to a series of necessary detections on the BDL model, it is ensured that the whole behavior sequence meets related grammar and semantic detections. The correctness of the demand animation is guaranteed thereby.

Basing on this study, future studies will concentrate on the transforming the heterogeneous model into behavior model, realizing the BDL using state diagram, and the conversion of BDL with other modeling methods.

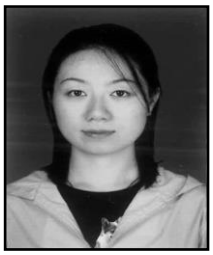
Acknowledgements

The research work was supported by the National Natural Science Foundation of China (Grant Nos. 91118003, 61100055, 31201121), the Educational Commission of Hubei Province (Q20131108) .

References

[1] W. N. Robinson, "A Roadmap for Comprehensive Requirements Modeling", Computer, vol. 43, no. 5, **(2010)**, pp. 64-72.
[2] C. Phillips, "Requirements Modeling with UML", Germany: LAP Lambert Academic Publishing, **(2009)**.
[3] N. Berente, S. Hansen and K. Lyytinen, "High Impact Design Requirements - Key Design Challenges for the Next Decade", Design Requirements Engineering: A Ten-Year Perspective: Springer Berlin Heidelberg, **(2009)**, pp. 1-10.
[4] S. Hansen, N. Berente and K. Lyytinen, "Requirements in the 21st Century: Current Practice and Emerging Trends", Design Requirements Engineering: A Ten-Year Perspective: Springer Berlin Heidelberg, **(2009)**, pp. 44-87.
[5] Q. Yan-Wen, "Software Behavior", Beijing: Publishing House of Electronics Industry, (in Chinese), **(2004)**.
[6] OCZ Gotel, F. T. Marchese and S. J. Morris, On Requirements Visualization//Proceedings of the Second International Workshop on Requirements Engineering Visualization (REV 2007), New Delhi, Indian, **(2007)**, pp. 11.
[7] W3C:OWLM web Services Architecture, http://www.w3.org/TR/ws-arch/.
[8] The Simple Object Access Protocol(SOAP) Version 1.2 At. http://www.w3.org/TR/soap.
[9] E. Christensen, F. Curbera and G. Meredith, web Services Description Language (WSDL)1.1,2001.At http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
[10] The Universal Deseription, Diseovery and Integration(UDDI) Protocol. Version3, At http://wwww.uddi.org, **(2003)**.
[11] G. Pieeinelli, "Service Provision and Composition in virtual Business Communities", Technical Report HPL-1999-84, Hewlen-Packard.
[12] F. ELeymalm, D. Roller and M. T. Schmidt, "web Services and Business Process Management", IBM System Journal, vol. 41, no. 2, **(2002)**.
[13] L. Lin, W. Guoqing, H. Bo, W. Li and W. Hao, "Behavioral Model Based Requirements Visualization Method", Chinese Journal of Computers, vol. 36, no. 6, **(2013)**,pp. 1312-1324.

Author



Li Lin, she received M.Sc. in computer applications (2006). Since 2010 she becomes Ph.D. candidate in computer software and theory from Wuhan University. Since 2009 she is lecturer of Wuhan University of science and technology. Her research interests include requirements engineering and formal method and requirements visualization.