Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

OPEN ACCESS Journal of Software Engineering

ISSN 1819-4311
DOI: 10.3923/jse.2017.47.53

Research Article
Research of Operation Semantic and PI Transformation Based on UML Sequence Diagrams

[1,2]Yao Wen Xia, [3]Sai Dong Lv and [4]Dong Xia Liu

[1]Solar Energy Research Institute,
[2]Department of Information,
[3]Department of Security,
[4]Department of Library, Yunnan Normal University, 650500 Kunming, China

Abstract

Background: The UML sequence diagram is a metamodel, didn't have the precise formal semantics, system design and development personnel cannot be observed system dynamic performance and can't realize system automatic reasoning and proof: On the other hand, in a state diagram and sequence diagram in UML, different sides of the same system, semantic inconsistencies may occur, because do not have form semantics, UML diagrams so unable to detect different types of conflict between the semantics of the UML subgraph. **Materials and Methods:** In this study, the UML sequence diagram is decomposed into the basic composition elements and the combination of several segments of combination, the basic composition elements are mapped to the corresponding components in the PI calculus and combination fragment is given operational semantics and then converted to PI calculus. **Results:** The UML sequence diagrams have "The same" part, there are also a part of the "change". Refers to the "constant" parts: Basic composition elements of UML sequence diagrams, such as classes, objects, messages, objects and the relationship between; part of "change" refers to that the combination of the basic composition elements assembled segment, such as the "loop", "break", "Alt", "par" and "opt", etc. So, from the UML sequence diagrams of XMI file, extract the key data, structure information, including the static structure and dynamic behavior, the assembly way, need to a few key XMI file labels for data mining. **Conclusion:** This study presents a UML sequence diagram is decomposed into basic constituent elements and the combination of several combinations of fragments, the basic constituent elements are mapped to the corresponding components of PI-calculus, operational semantics is given a combination of fragments and then converted to PI calculus.

Key words: UML, sequence diagrams, plcalculus, semantic, meta-model

Received: July 28, 2016 Accepted: October 30, 2016 Published: December 15, 2016

Citation: Yao Wen Xia, Sai Dong Lv and Dong Xia Liu, 2017. Research of operation semantic and PI transformation based on UML sequence diagrams. J. Software Eng., 11: 47-53.

Corresponding Author: Sai Dong Lv, Department of Security, Yunnan Normal University, 650500 Kunming, China Tel: +86-871-65911900

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

The UML state diagram describes an object in the system response to external events, the changes of the life cycle and dependence on the past behavior and so on. But a real complex systems, often there are multiple objects at the same time and their interaction[1]. In UML state diagram describes multiple objects and their interactions; the description of the UML state chart ability is insufficient, sometimes even impossible. At this point, then use UML sequence diagrams to model. The UML sequence diagrams depict, send messages to each other, between multiple objects in the system of interaction. Because UML state diagram depicting the changes of the internal state is an object of UML sequence diagrams depict the interactions between multiple objects, so, in a sense, the UML sequence diagram is a kind of abstract, the state diagram and UML state diagram is the elaboration of the sequence diagram[2].

Unified Modeling Language is by GradyBooeh JimRumbaugh and IvarJaeobson three famous object-oriented technology experts launched, in OOSE method, OMT method and BOOCH representation, on the basis of widely for advice, again and again modification and complete and effective object-oriented analysis and design of a standard modeling language, it is currently the most popular a kind of object-oriented modeling language. The UML provides rich modeling element, it can be either for member of the static structure modeling and can be member of the dynamic behavior modeling[3].

In a system to be developed, any objects are not exist in isolation, these objects in the system by passing messages interact, UML sequence diagrams can be the interaction of the system is modeled and graphical solution[4]. When operation is carried out to the class of classes, objects, components, use case and the whole system control flow modeling, interactive behavior occurs. An interaction refers to the specific context, in order to achieve a goal and in a set of message exchange between a set of objects represented by behavior.

In UML 2.0 standard, the news of the UML sequence diagram is to point to: Start an operation or to send a signal and create or destroy an object. Messages by serial number, name and parameters (optional). The UML sequence diagrams of basic composition elements include: The object, the lifeline, news and order number.

Relative to the UML 1.0 standard, UML 2.0 puts forward the concept of combination fragment, under different scenarios, provides more clear semantics. Portfolio pieces include: Circulation, branch, interruption, critical region, parallel, references, etc[5].

The UML sequence diagrams of the basic composition elements, can be assembled by combining clip, to represent the semantics of different scenarios, enhances the ability of UML sequence diagrams describe.

The UML state diagrams depicts the changes in internal state of an object, UML sequence diagram depicts is the interaction between multiple objects, so in a sense, UML sequence diagram is an abstraction of a state diagram and UML state diagram is refinement of sequence diagrams[6,7].

The UML sequence diagram is a kind of meta model, does not have the precise semantics of formal, system design and development staff, cannot observe the system dynamic operation and in the UML state diagram and sequence diagram to describe the different aspects of the same system may appear semantic inconsistency, because UML graph element does not have formal semantics. Therefore, this study will be UML state diagrams are assigned with its operational semantics and converted to the PI calculus specification[8].

MATERIALS AND METHODS

The PI calculus is a kind of process algebra, derived from CCS, suitable for modeling mobile concurrent distributed system, compared with CCS, PI calculus allowed on the channel transmission channel, the dynamic channel of creation and destruction, this feature enables the PI calculus is especially suitable for modeling mobile concurrent communication system[9].

Definition 1: The LTS structure is formalized definition for five-tuple array among them:

$$M_\pi = (\Sigma_\pi, I_\pi, N_\pi, C_\pi, \Delta_\pi)$$

where, Σ_π is the set of process id, $I_\pi \subseteq \Sigma_\pi$ is the set of active process id, N_π is the set of chanel and $N_\pi = N_{in} \cup N_{out} \cup \{\tau\}$, among, N_{in} represent the set of input chanel, N_{out} is the set of output chanel, τ is external invisible internal channels, C_π is a collection of matching structure, matching structure is a Boolean represention channel.

$\Delta_\pi \subseteq \Sigma_\pi \times E_\pi \times C_\pi \times \Sigma_\pi$, Δ_π is a function of a said migration relations, its semantic: The current state of UML state diagram, if has a trigger event occurs and meet the health conditions, to perform an action, then the system migration to the following form's a function, from the current state to the following state of the relationship between migration and the label for the state transition is a trigger and a health condition.

EXtensible Markup Language (XML)[6] is a powerful technique, suitable for the data in a file. Because XML is a

standard, the data can be saved as a format, for no other applications that use to create the data, the user can through the standard application interface, such as DOM or SAX, extract the data in the XML document.

However, XML is not object-oriented, XML element defines the XML and XML attributes, but does not define the object, so it does not support object-oriented features, also does not include the object model. If, in the XML software tools will be the same data is expressed as different forms, so it is difficult to through the tools to exchange data[10].

In order to solve the above problems, the XML Metadata Interchange (XMI). Because XMI defines the mapping between the object and the XML, so once defines the objects to exchange, if you use the XMI, you don't need to create your own XML representation for the object. So, now many UML Case tools, such as ArgoUML, Poseidon-for-UML, the Rational-Rose, supported the UML graph exported to XMI file format[11].

Asynchronous send messages

Definition 2: The UML sequence diagrams can be abstracted as a triad (OBJ, MES and OPERATOR), among them:

OBJ : Represent the set of system object
MES : Represent the set of system message
OPERATOR : Represent the set of combination fragment enumeration. It's definition is follows {alt, loop break, par, opt···}, combination fragment represention objects and message assembly mechanism of UML sequence diagrams, including branch combination fragment, loop combination fragment, break combination fragment, concurrent combination fragment, etc.

Definition 3: The messages in UML sequence diagrams, message refers to start an operation or to send a signal that can be abstracted as a binary group <mName, param>, among them, the mName represent message name, param represent optional parameters.

Rule 1: (obj⇒p)∧(mess⇒channel)∧(obj∈OBJ)∧(mess∈MES)∧(p∈PROCESS)∧(channel∈CHANNEL), among them, PROCESS represent the set of PI calculus process CHANNEL represent the set of PI calculus channel, the rule represent objects in UML sequence diagrams, a mapping to process in the PI calculus, UML sequence diagrams of messages, mapping the channel in PI calculus.

Definition 4: Asynchronous messages defined by a function: Send:src×mess×targ and (src∈OBJ)∧(mess∈MES)∧(targ∈OBJ), among them, the src represents the source object, targ represents target object, mess represents momentum to send the message. In UML sequence diagrams, the source object to a target object sends a message, to continue, not return a value.

The operation of the asynchronous message semantic is defined as follows:

$$\frac{\text{client} \xrightarrow{\overline{\text{event}}<\text{asynMess}>} \text{client}';\ \text{server} \xrightarrow{\text{event}(z)} \text{server}'}{\text{client} \mid \text{server} \xrightarrow{\tau} \text{client}' \mid (\text{server}'\{\text{asynMess}|z\})}$$

Among them, τ represent internal action, server' {asynMess|z} represent in process server' are replaced with asynMess, client|server represent client and server-execute concurrently◦ $\overline{\text{event}}$ <asynMess> represent send message asynMess said to pass the event, the event(s) to pass the event to receive messages and stored in the s.

Figure 1 is an example of asynchronous messages, among them, the event is the channel between SRC and targ, to deliver the message mess. The SRC' is the state of the successor of SRC, targ' is targ successor state. Source object the client to the target server sends an asynchronous message asynMess, not return a value, asynchronous messages can be used for asynchronous concurrent combination between two communication parties.

By defining 4 and its operational semantics, shown in Fig. 1 to generate the corresponding PI calculus code for:

client = $\overline{\text{event}}$ <asynMess>.client
server = event(z).server
system = client|server

Message of the synchronous and inside information

Definition 5: Synchronization messages is defined as a function: Return:targ×mess×src→return Val and (targ∈OBJ)∧(mess∈MES)∧(src∈OBJ)∧(return Val∈MESS), among them, retumVal is returned by the news, with other variables definition 2. In UML sequence diagrams, synchronous message said: After the source object a message sent to the

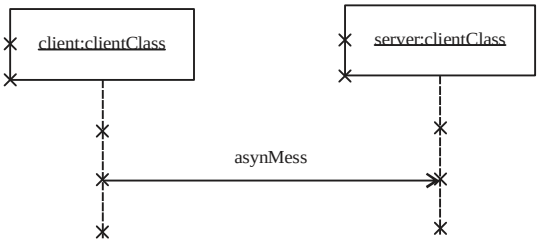

Fig. 1: Asynchronous messaging

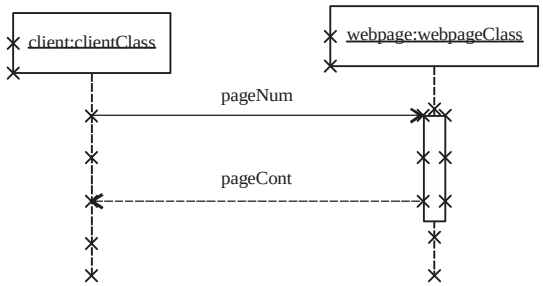

Fig. 2: Synchronous message sending

target object block, until the target object is returned to the source object a reply message.

Figure 2 shows the operation of the synchronous message is sent semantics are defined as follows:

$$\frac{\text{client} \xrightarrow{\overline{\text{pageNum}}.\text{pageCont}(x)} \text{client}' ; \text{webpage} \xrightarrow{\text{pageNum}.\overline{\text{pageCont}}<y>} \text{webpage}'}{\text{client}|\text{webpage} \xrightarrow{\tau} \text{client}'\{y|x\}) \mid \text{webpage}'}$$

The UML sequence diagram shown in Fig. 2, is an example of a synchronous message is sent, the event, the src', the definition of targ'. Client object to the object of webpage after sending a message pageNum jam, until the object of webpage return a message after webCont object client before lifting block, continue to execute. A synchronous message sent is suitable for the communication parties synchronous parallel combination.

Generated by defining 5 and its operational semantics, UML sequence diagram in Fig. 2 of the PI calculus code for:

$$\text{client} = \overline{\text{pageNum}}.\text{pageCont}(x).\text{src}$$

$$\text{webpage} = \text{pageNum}.\overline{\text{pageCont}}<y>.\text{webpage}$$

Definition 6: Internal messages defined is a function: Internal: src×mess×src and (mess∈MES)∧(src∈OBJ) in UML sequence diagrams, sending a message to an object, called the inside information.

Figure 3 is an example of inside information, object P sends a message internalEvent to itself, not return a value.

Its operational semantics are defined as follows:

$$\frac{P \xrightarrow{\text{internalEvent}} P'}{P \mid Q \xrightarrow{\text{internalEvent}} P' \mid Q}$$

Generated by definition 6 and its operational semantics, PI calculus code as shown in Fig. 3:

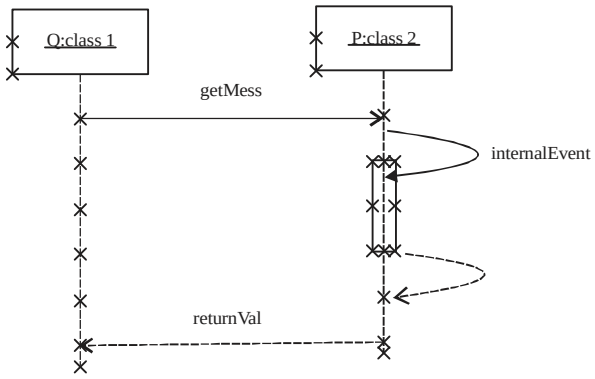

Fig. 3: Inside information

$$P = \text{event}(x).([x = \text{internalEvent}]t.P + [x = \text{getMess}]\overline{\text{returnVal}}\ <y>.\ P)$$

$$Q = \overline{\text{event}}\ <\text{getMess}>.\text{returnVal}(z).Q$$

Branches combined fragment

Definition 7: Branching behavior is defined as a function: BranchAction: $\text{src}\times\Sigma_{i\in\{1,2,\ldots,n\}}(\text{guard}_i\times\text{mess}_i)\times\text{targ}\rightarrow\text{returnVal}_i$, among them, guard_i represent the i guard condition, mess_i represent the i message, returnVal_i represent the i return val, n represent guard condition or the number of message. Different gurad conditions, the source object sends a different message to the target object different return values from the target object.

Figure 4 is an example of a branching behavior, source object client who type in different conditions, different messages sent to the target server, the server returns a different return values. Branch system is suitable for the condition of the selective action.

Figure 4 shows that branch behavior of operational semantics are defined as follows:

$$\frac{\text{client} \xrightarrow{\text{guard}_i\,\overline{\text{getMess}_i}.\text{returnVal}(x)} \text{client}' ; \text{server} \xrightarrow{\text{getMess}_i.\overline{\text{returnVal}_i}<y_i>} \text{server}'}{\text{client} \mid \text{server} \xrightarrow{[\text{guard}_i]\tau} \text{server}' \mid \text{client}'\{y_i|x\}}$$

By the definition 7 and its operational semantics, UML sequence shown in Fig. 4, the corresponding PI calculus code for:

$$\text{client} = \left(\begin{array}{l}[\text{guard}_1]\overline{\text{event}} < \text{getMess}_1 > + [\text{guard}_2]\overline{\text{event}} \\ < \text{getMess}_2 > + \ldots + [\text{guard}_n]\overline{\text{event}} < \text{getMess}_n >\end{array}\right).\text{returnval}(x).\text{client}$$

$$\text{server} = \text{event}(y).\left(\begin{array}{l}[y = \text{getMess}_1]\overline{\text{returnVal}} < y_1 > + [y = \text{getMess}_2]\overline{\text{returnVal}} \\ < y_2 > + \ldots + [y = \text{getMess}_n]\overline{\text{returnVal}} < y_n > .\text{server}\end{array}\right)$$

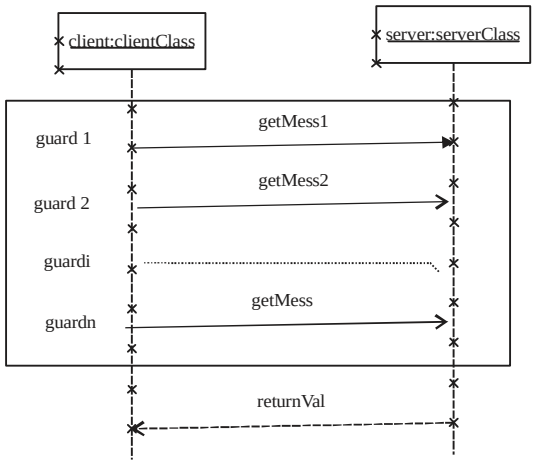

Fig. 4: Branching behavior

Loop combination fragment

Definition 8

Cyclic behavior: In UML sequence diagrams, by the keyword "loop" modify the behavior of the sequence, if meet the health conditions, is executed repeatedly, until who type condition is false.

Figure 5 is an example of a cyclic behavior, was established in guard who type conditions under the premise of the source object client server sends a message to the target object repeatedly getMess and get the return value messVal. Cyclic behavior is suitable for the system to be executed repeatedly, until the condition is false, who type out of circulation.

Figure 5 shows that the cyclic behavior of the operational semantics are defined as follows:

$$\frac{\text{client} \xrightarrow{[\text{guard}]\overline{\text{getMess}}.\text{messVal}(x)} \text{client}' ; \text{server} \xrightarrow{[\text{getMess}]\overline{\text{messVal}}<y>} \text{server}'}{\text{client} \mid \text{server} \xrightarrow{[\text{guard}_i]\tau} \text{client}' \{y_i | x\} \mid \text{server}'}$$

Generated by defining 8 and its operational semantics, shown in Fig. 5 PI calculus code of UML sequence diagrams are as follows:

$$\text{lient} = [\text{guard} = \text{ture}]\text{event}<\text{getMess}>.\text{messVal}(x).\text{client} + [\text{guard} = \text{false}]\ \text{otherActions.client}$$

$$\text{server} = \text{event}(x)([x = \text{getMess}]\text{messVal}<y>.\text{server} + [x \neq \text{getMess}]\text{t.server})$$

Among them [guard = false] otherActions.client represent: If the guard condition is false, the client to perform the action of other sequences.

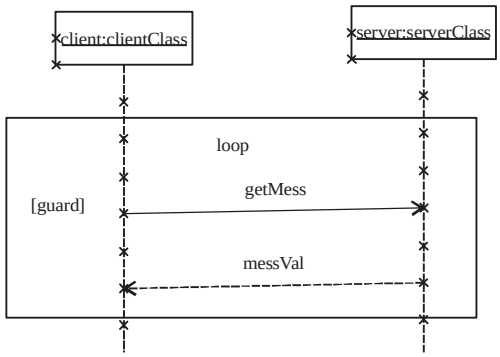

Fig. 5: Cyclic behavior

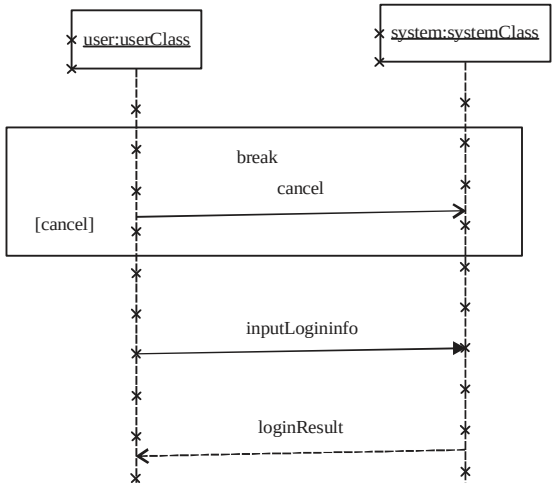

Fig. 6: Break behavior

Break combination fragment

Definition 9: Break behavior is defined as a function: Break: [breakGuard]? exit: continue, among them, breakGuard represent break operation sub-health condition, exit exit the execution sequence, said the continue to continue execution sequence. In UML sequence diagrams by the keyword "break" the behavior of the modified, if meet the health conditions, the execution sequence, terminate the interaction between objects; otherwise continue to execute.

Figure 6 is an example of break behavior, object's system user sending messages to objects askLogin, asked if login, if who type cancel button pressed, the user press the cancel button, the user sends a message cancel the system and user and the system to return to the initial state. Interrupt behavior applies to meet certain health conditions, system is performing the action sequence, return to original state.

Figure 6 shows interruption behavior operation semantics are defined as follows:

$$\frac{user \xrightarrow{[cancelButtonPressed]\overline{canel}} 0; system \xrightarrow{cancel} system'}{user \mid system \xrightarrow{[cancelButtonPressed]\tau} 0 \mid system}$$

Generated by definition 9 and its operational semantics, Fig. 6 PI calculus code as follows:

$$system = \overline{askLogin}.\ (event(x).[x = cancel]t.system + inputLoginInfo.\overline{loginResult}.system)$$

$$user = askLogin.([[cancelButtonPressed = true])\overline{event} < cancel > .0 + [cancelButtonPressed = false]\overline{inputLoginInfo}.loginResul.user)$$

RESULTS

The UML sequence diagrams have "the same" part, there are also a part of the "change". Refers to the "constant" parts: Basic composition elements of UML sequence diagrams, such as classes, objects, messages, objects and the relationship between; part of "change" refers to that the combination of the basic composition elements assembled segment, such as the "loop", "break", "Alt", "par" and "opt", etc. So, from the UML sequence diagrams of XMI file, extract the key data, structure information, including the static structure and dynamic behavior, the assembly way, need to a few key XMI file labels, for data mining.

For UML sequence diagrams, UML 2.0 standards relative to the UML 1.0 standards, increased the combination fragment, to the UML sequence diagram provides more clear semantics. In UML 1.0, for example, UML sequence diagrams using "annotation" write loop condition, if the comment is added to the loop condition is true, will perform a set of messages; but in UML2.0 directly provides a "loop" combination fragment, describe the cyclic behavior of UML sequence diagrams, "loop" combination, you can set the loop is really who type conditions. Similarly, UML2.0 in sequence diagrams, describe the behavior of other combination fragment is proposed.

From UML sequence diagrams to LTS form semantic generation algorithm, its basic idea is: First, the UML sequence diagram exported to XMI format; Second, the use of Java+Xerces parser for XMI file to construct a DOM tree, traverse the DOM tree, extract the five tag data/information structure, including: <UML: Class>, <the UML2: Message>, <the UML2: Lifeline>, <the UML2: Event Occurrence>, <the UML2: Combined Fragment> and stored in the corresponding dynamic linked list; again, through dynamic data information in the list, to extract the UML sequence diagram of static topology structure, dynamic behavior and combinations in accordance with this study put forward transformation rules, generate the corresponding semantic LTS form.

DISCUSSION

Many researchers in the UML to give formal semantics, to achieve automatic system analysis, reasoning, validation, etc., do a lot of work[12]. Directly from the UML state diagram to pi calculus conversion rules, this study first UML state diagram abstracted as a mathematical model and then to the UML state diagram assembly mechanism to give LTS operation semantics[12]. Lam and Padget[13] put forward an integrated environment of UML state diagram. Firstly, the UML state diagram is transformed into PI calculus and then the mapping relation between LTS structure and Kripke structure is used. Lam and Padget[14] puts forward the transformation of PI calculus into NuSMV code, which realizes the integration of UML state diagram. Jansen[15] presents a consistency checking problem of UML state diagram based on class hierarchy, which is to judge whether the state diagram of the superclass is consistent with the state graph of the subclass.

This study differs from the study of Korenblat and Priami[12], Lam and Padget[13,14] and Jansen[15]: First, this study abstracts UML state diagram and LTS structure into mathematical model and realizes the transformation between these two mathematical models. Secondly, this study gives the LTS operation semantics to the UML state diagram assembly mechanism, which can be easily converted to PI calculus process algebraic specification.

In this study, a subset of all the combined segments of UML sequence diagram is selected, which is a subset of all the combined fragments of the sequence diagram. There are other operators, such as assert, consider, ignore, ref and some and PI calculus between no implicit mapping relationship, the LTS operational semantics given by the same method and converts it to PI calculus code.

The UML state diagrams are assigned with its operational semantics and then converted to PI specific form specification, the advantage of doing so is: First give state diagram of the LTS operational semantics and translation algorithm, conversion for any kind of process algebra, but is not limited to a kind of process algebra.

The PI calculus is a complex algebraic system, which requires the user to have a certain mathematical foundation times lower, not easy to master and the UML meta model is now the facts of industrial standards, its intuitive easy to be accepted by most people. For visualization of the design program, if given to the UML sub graphFormal semantics,

researchers can hide the complex process algebra theory, to achieve automatic deadlock verification, language Semantic consistency checking.

The UML sequence diagram is the state diagram state diagram is abstract, refinement of sequence diagrams, there are in different abstract level. If you can convert it to the PI algorithm code, can be in the process of model refinement. The consistency checking of different UML sub graphs is realized by using the theory of mutual simulation equivalence, consistency, design and development staff can be supported by the existing process algebra tools, through a single step execution, dynamic operation of the observation system, to confirm whether the design meets the requirements.

CONCLUSION

In this study, the UML sequence diagrams abstract as a mathematical model, the LTS structure abstract mathematical model for another, through the LTS operational semantics and transformation rules, implement the conversion between two kinds of mathematical model, finally, generate the corresponding PI calculus code. Because the transformation between the two mathematical models, so the algorithm description is more accurate and easy to use mathematical induction and analysis for the next step transform provide the basis of the correctness of the algorithm.

ACKNOWLEDGMENTS

First Authors thank the reviewers for their constructive comments in improving the quality for this study. This study was supported by the YunNan province education department fund project (No. 2014Y147), Academic education research station in YunNan province, YunNan province high quality basic education resource sharing and interactive service research.

REFERENCES

1. Fu, M.L., 2015. The application of software development use UML model technology. Science and Technology, pp: 18-19.
2. Lv, S.D. and Z.P. Li, 2014. Research on the probability of extended UML state diagram/random kripke structure semantic. BioTechnol. Indian J., 10: 5576-5583.
3. Lin, J.Q., 2016. The application of object-oriented are based on UML. China CIO News, pp: 99-103.
4. Yue, S., 2015. The method of software realization based on UML sequence. Comput. Sci., 25: 326-329.
5. Milner, R., J. Parrow and D. Walker, 1992. A calculus of mobile processes (Parts I and II). Inform. Comput., 100: 1-77.
6. Tian, Z.Y., 2016. XML Practical Technology. Tsinghua University Press, Beijing, pp: 427-430.
7. Messaoudi, N., A. Chaoui and M. Bettaz, 2015. An operational semantics for UML 2 sequence diagrams supported by model transformations. Procedia Comput. Sci., 56: 604-611.
8. Posadas, H., P. Penil, A. Nicolas and E. Villar, 2015. Automatic synthesis of communication and concurrency for exploring component-based system implementations considering UML channel semantics. J. Syst. Architect., 61: 341-360.
9. Khan, M.U., 2015. Representing security specifications in UML state machine diagrams. Procedia Comput. Sci., 56: 453-458.
10. Sheng, Y.C., 2016. The research on UML use case to XML scheme. J. Guiyang Univ. (Natl. Sci.), 11: 25-31.
11. Cabot, J., R. Clariso and D. Riera, 2014. On the verification of UML/OCL class diagrams using constraint programming. J. Syst. Software, 93: 1-23.
12. Korenblat, K. and C. Priami, 2003. Extraction of Pi-calculus specifications from a UML sequence and state diagrams. Technical Report No. DIT-03-007. Information Engineering and Computer Science, pp: 601-621.
13. Lam, V.S.W. and J. Padget, 2004. Symbolic model checking of UML statechart diagrams with an integrated approach. Proceedings of 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, May 24-27, IEEE Computer Society, Washington, DC, USA., pp: 337-346.
14. Lam, V.S.W. and J. Padget, 2005. Consistency checking of statechart diagrams of a class hierarchy. Proceedings of the 19th European conference on Object-Oriented Programming, Glasgow, UK., July 25-29, 2005, Springer-Verlag, Berlin, pp: 412-427.
15. Jansen, D.N., 2002. Probabilistic UML statecharts for specification and verification: A case study. Workshop on Critical Systems Development with UML, September 30, 2002, Dresden, German, pp: 121-132.