The Refinement Check of Added Dynamic Diagrams Based on ®-Calculus

Zhou Xiang[1,2] and Shao Zhiqing[2]
[1]College of Computer Science and Technology, Qingdao University, China
[2]School of Information Science and Engineering, East China University of Science and Technology, China

Abstract: *As the semi-formal modeling tool, UML has semantics defaults which may cause confusions or even mistakes in refinement of models. ®-calculus is a formal specification based on process algebra, which can give strict semantics description for system behaviors. We seek to clearly define the semantics of refinement to a model through ®- calculus and thus we are able to propose a formal verification method of the refinement. Employing this method, we can improve the efficiency of the consistency verification while decreasing the mistakes in the refinement process.*

Keywords: *®-calculus; UML; sequence diagram; state chart diagram; weak open bisimulation.*

Received January 21, 2014; accepted December 22, 2014; Published online December 23, 2015

1. Introduction

UML is by far the most popular diagram-based modeling tool. UML uses static diagram to describe the static characters of a system and uses dynamic diagram to describe the behaviors of the system [15]. Thus, mainly employing diagrams, UML is straightforward and easily apprehensible. However, UML is a semi-formal modeling tool, which can cause inaccuracy in semantics. Right now most software development involves the collaboration from multiple developers. However, developers may not share the same perspective on the system and thus, discrepancies may occur in the modeling. Such discrepancies mostly occur in the dynamic diagrams that describe the system's behaviors. Thus, it is vital to provide a precise formal semantics description for the dynamic diagrams in order to assure the consistency in developing, assembling and refining the system.

For large-scale systems, developers often need to modularize the development tasks, that is, developers will first establish the basic model and then focus on the specific details according to the demand and refine the model level by level [5]. Because it involves relatively less information regarding objects, states, messages, transitions at very early stage, the establishment of the basic model is relatively easier and it is also easier to verify the consistency in semantics. However, in the later stages, descriptions will be added to the model or revised in every stage and the diagram will become more and more complex, which makes it increasingly difficult to verify the semantics. The traditional method of semantics verification is to submit all diagrams for verification, even for minor revisions. Thus, the traditional method involves much redundant work and severely decreases the efficiency and accuracy of software development and revisions.

In software and system development process, the most important task is to detect errors at early stages of their life-cycles. There are many works for the correctness of software development: In the literature [3, 17] implemented UML models with OCL, which is too complex and reducing the model readability; Model-driven can transform models flexibility but the specification isn't critical [10]. Formal semantics can provide reliability for models. Literature [2] focused on the interaction semantics; literature [1, 18] defined the semantics for executable models, but lacks the refined verification. In the literature [6, 16], refining the behavior of state chart and activity diagram but the consistency between different diagrams was ignored.

Milner [13] proposed ®-calculus. ®-calculus is a process algebra method that is based on Calculus of Communicating Systems (CCS). However, different from CCS, ®-calculus allows pass names on channel. Through the name-passing, ®-calculus enables the process that receives channel names to use these names to communicate with other processes. ®-calculus grants the system the capability of dynamically creating and destroying channels and allows the "mobility" to be expressed in a implicit manner. ®-calculus is a formal specification, which can specifically define all types of activities. ®-calculus can transform the behavior equivalence of a system to the weak open bisimulation between processes to verify this equivalence. Therefore, we propose that employing ®-calculus and take advantage of the weak open bisimulation, we can verify the additions to a model to assure the consistency in the dynamic diagrams in the refinement process and give the proof of relevant semantics.

Previous literature [4] has given the formal verification approach of UML that is based on design calculus. However, compared with ®-calculus, design calculus does not have the automatic verification tool. Literature [14] has detected errors in UML sequence diagrams using Labeled Transition System Analyzer (LTSA), it focuses on the integration of multiple sequence diagrams. Previous literature [11] has defined a variety of bisimulations and given the appropriate contexts for employing these bisimulations. However, the literature does not provide the specific application method for current dynamic diagrams.

Lam and Padget [7, 8, 9] gave the ®-calculus semantics definitions of most sequence diagrams and State chart Diagrams. They also adopted MWB [19] for the automatic verification of the semantics in dynamic diagrams. Based on their works, the literature [21] added operation semantics of each operator to dynamic diagrams. Based on the above works, we propose a method to verify the consistency among the refinement parts in a model as well as the consistency of the integration of the refinement parts and the original diagrams.

The paper is structured as follows: Section 2 reviews related works. Section 3 introduces the basic semantics and syntax of ®-calculus. Section 4 gives the rules and definitions of ®-calculus, as well as the translation from the sequential diagram and the state chart diagram into processes and channels. Section 5 introduces the check of the added dynamic diagrams; the definition of the largest weak open bisimulation set and related the substitution rule and proof. The last section is conclusions and suggested future research.

2. ®-Calculus

2.1. Syntax of ®

®-calculus is a formal method that is based on process algebra. Thus, it is very suitable to describing the concurrent system. There are two essential elements in ®-calculus: Processes (ranged over by *P*, *Q*, *R*) and names or channels (ranged over by *x*, *y*, *z*, *w*, *v*, *u*). The syntax and semantics of ®-calculus is given as follows:

1. $x(\vec{y}).P$: Input prefix, which receives channels along channel *x* and continues as process *P* with *y*1, *y*2, …, *yn* replaced by the received channels. The input prefix *x*().*P* is abbreviated as *x*.*p*.
2. $\overline{x}\langle\vec{y}\rangle.P$: Output prefix, which sends channels along channel *x* and continues as process *P*. The output prefix $\overline{x}\langle\ \rangle.P$ is abbreviated as $\overline{x}.P$.
3. $(v\,\vec{x}).P$: Restriction, which creates new channels x_1, …, *xn* for communications in processes *P*.
4. *P*|*Q*: Concurrent processes *P* and *Q* which execute in parallel.
5. *P*+*Q*: Non-deterministic choice, represents proceeding process *P* or *Q*. $\sum_{i=1}^{n} P_i = P_1+\ldots+P_n$ Represents proceeding one of the n processes.
6. [*x*=*y*]*P*: Matching construct, which proceeds as process *P* if the matching condition is true, otherwise continues as null process.
7. ².*P*: Internal prefix, which execute internal action *τ* then proceeds process *P*.
8. $A(x_1,\ldots,x_n) \overset{def}{=} P$: Process identifier *A*, which behaves like process *P*.

In the three prefix representations, the channels $\vec{y}$ in input prefix $x(\vec{y}).P$ and $\vec{x}$ in restriction operator $(v\,\vec{x}).P$ are bound names; the channels $\vec{y}$ in output prefix $\overline{x}\langle\vec{y}\rangle.P$ are free. The bound names and free names of *P* are defined as *bn*(*P*) and *fn*(*P*). According to logic, all the free names in the same ®-calculus are regarded as the same variable. Thus, the scope of free names is the whole formula, i.e., *fn*(*P*, *Q*)= *fn*(*P*)∪*fn*(*Q*).

2.2. Actions of ®

Precisely as in CCS, a transition in the ®-calculus is of the form $P \xrightarrow{\alpha} Q$.

Intuitively, this transition means that *P* can evolve into *Q* and in doing so perform the action α. In our calculus there will be four kinds of action α as follows:

1. The silent action². As in CCS, $P \xrightarrow{\iota} Q$ means that *P* can evolve into *Q* and doing so requires no interaction with the environment. Silent actions can naturally arise from agents of form ².*P*, but also from communications within an agent.
2. A free output action y. The transition $P \xrightarrow{\overline{x}y} Q$ implies that *P* can emit the free name *y* on the port $\overline{x}$. Free output actions arise from the output prefix form $\overline{x}y.P$.
3. An input action *x*(*y*). Intuitively, $P \xrightarrow{x(y)} Q$ means that P can receive any name w on the port *x* and then evolve into $Q\{w/y\}$. Note that, this departs slightly from CCS, where an input action contains the actual received value. Here, *y* instead represents a reference to the place where the received name will go; *y* is enclosed in brackets in order to stress this fact. Input actions arise from the input prefix form *x*(*y*).*p*.
4. A bound output action $\overline{x}(y)$. This kind of action has no counterpart in CCS. Intuitively, $P \xrightarrow{\overline{x}(y)} Q$ means that *P* emits a private name on the port $\overline{x}$ and *y* is a reference to where this private name occurs. As in the input action above, *y* is enclosed in brackets to emphasize that it is a reference and does not represent a free name. Bound output actions arise from free output actions which carry names out of their scope, as e.g., in the agent

$(y)\bar{x}y.P$.

There are two reasons to choose ®-calculus as the execution semantics definition language. Firstly, transitions can be easily described as the internal structure of the process evolution; secondly, the bisimulation in the ®-calculus can be used as a tool to verify the dynamic behavior equivalence.

3. Translation of State Chart Diagrams and Sequence Diagrams Into the ®-Calculus

UML uses activity diagrams to describe the execution and parallel order of the logic workflow in the system, use cases, and program models. Therefore, UML models the actual work process of human world, which helps to understand the high level of the execution behavior of a system. The sequence diagram is a type of the interaction diagram and interaction diagrams are models that describe how groups of objects collaborate in some behavior. The sequence diagram thus emphasizes the sequence of time and messages and it describes the dynamic aspect of a system. The state chart diagram describes the states of objects as well as the events and conditions of these states to transit among each other. The state chart diagram reveals the life cycle of objects. The activity diagram can be viewed as a special case of the state chart diagram. This section will briefly introduce how to use ®-calculus to describe the sequence diagram and the state chart diagram.

From the execution perspective, the semantics of the state chart diagram includes event queues, transition conflicts, priority schemes and so forth. These semantics can be categorized into two important entities: State and transition. In UML 2.0, a state is denoted by a rounded rectangle; a state includes the basic state and the composite state. Depending on whether concurrence is supported, composite state can be further classified into: Non-concurrent composite state, that is, only one of its substates is active at any point of execution, and concurrent composite state, which has a number of orthogonal areas and substates in different orthogonal regions, can be active at the same time. A transition is denoted as an arrow labeled with events, guard-conditions, or actions. Assume defining ST as a set of states ranged over *S*, *T*, *V*, *W*, £ as a set of events, $\mathfrak{R}$ as a set of transitions, GCond as a set of guard conditions in the state diagram; Ain and Aout describe the set of input and output activities, respectively. Then, we can use processes and channels of ®-calculus to describe all the above elements of a state diagram. The translation is defined as follows:

- *Φ*: The function set map each element of dynamic diagram to the ®-calculus.
  - **Rule 1**: $\mathrm{c}_{event}: \text{£} \rightarrow N$ maps each event in a state chart diagram to a channel in the ®-calculus.
  - **Rule 2**: $\mathrm{c}_{state}: ST \rightarrow A$ maps each state in a state chart diagram to a process in the ®-calculus and returns a unique process identifier. For the process identifier $A(event, \vec{e}, \cdots)$, $\vec{e}$ stands for the events sequence $e_1 \cdots e_n$, event x gives the definition of action taken for different events. Meanwhile, in order to ensure the consistency of execution semantics, $\vec{e} \in ran(\mathrm{c}_{event})$, that is the range of event function.
  - **Rule 3**: $\mathrm{c}_{guard}: GCond \rightarrow A_{out}$, Maps each guard condition to an output action. The guard condition is a Boolean, so only true and false are two options on the output action, that is, $\bar{g}\langle x \rangle.([x = true]\cdots + [x = false]\cdots)$
  - **Rule 4**: $\mathrm{c}_{ST}: 2^{ST} \rightarrow 2^{A}$ translates a state chart diagram into a set of process identifiers, which are all the processes evolved from the one corresponding to the initial state.

  Another important dynamic diagram of UML, sequence diagram is composed of two basic elements: Object and message. We use rectangle to represent objects. Objects that take part in the interaction will be arranged horizontally at the top of a sequence diagram. The dotted line below each object represents the life cycle of the object. In UML2.0, we added some operators, such as refer, option, parallel, etc., suppose we use OBJ to represent objects in a sequence diagram and MES for messages; similarly, we can use the process and the channel of ® to describe these elements of a sequence diagram.

  - **Rule 5**: $\mathrm{c}_{mes}: MES \rightarrow N$ maps each message to a channel.
  - **Rule 6**: $\mathrm{c}_{obj}: OBJ \rightarrow 2^{A}$ translates each object of a sequence diagram into a set of process identifiers of ®-calculus, where are derived from the initial process of the object.

As a result, employing ®-calculus, we can convert two independent dynamic diagrams in UML to similar semantics formulas by the rules as listed above. Furthermore, we can prove the consistency in semantics of the sequence diagram and the state chart diagram to make up for the lack of semantics caused by the semi-formality.

4. Consistency Check of the Refinement by ®-Calculus

It is well known that it is virtually impossible to satisfy all demands at the beginning of designing a model. Most of the time, the process of designing a model starts with a preliminary framework. Based on this framework, developers will add new details to the model or revise existing parts of the model, which is called the refinement of the model. Based on the ®-calculus formulas provided in section 3, we will apply the weak bisimulation equivalence to the consistency verification of the model, which gives the method to verify the consistency of the dynamic diagrams in refinement.

4.1. Telephone Dialing System

Figure 1 shows the initial sequence diagram of a telephone dialing system, which simply describes two objects: Caller and phone, with three messages between them.

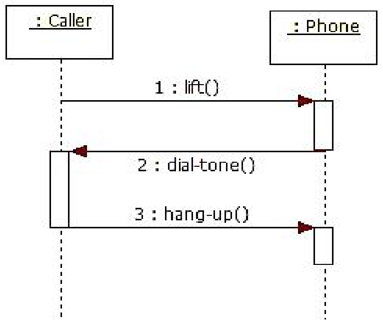

Figure 1. Initial sequence diagram.

Based on the semantics given in section 2 and section 3, we can give the ®-calculus of phone in Figure 1. Firstly, according to the rule 6, the object can be translated into a set of process S={S_1, S_2, S_3, S_4, S_t} in which S_1 is the initial process and S_t is the final process. Assume the message sequence is translated into the channels with same names $\vec{m} = \{lift, dial-tone, hang-up\}$ by rule 5, then:

$$\begin{aligned}
&S_1(mes_p, \vec{m}, mes_c) = mes_p(x).((([x = lift]).\\
&\quad S_2(mes_p, \vec{m}, mes_c) + ([x \neq lift]).S_1(mes_p, \vec{m}, mes_c)\\
&S_2(mes_p, \vec{m}, mes_c) = \overline{mes_c} < dial-tone >.\\
&\quad S_3(mes_p, \vec{m}, mes_c)\\
&S_3(mes_p, \vec{m}, mes_c) = mes_p(x).([x = hang-up]).\\
&\quad S_t(mes_p, \vec{m}, mes_c) + ([x \neq hang-up]).\\
&\quad S_3(mes_p, \vec{m}, mes_c)
\end{aligned} \tag{1}$$

Figure 1 corresponds to the state chart diagram as shown in Figure 2.

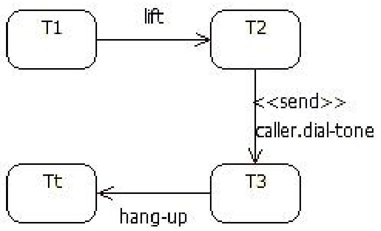

Figure 2. Initial state chart diagram of phone.

Map each state of Figure 2 into the process with same name by Rule 2 and map the event sequence $\vec{e} = \{lift, dial-tone, hang-up\}$ into the channels with same names; then the ®-calculus of Figure 2 is:

$$\begin{aligned}
&T_1(event_p, \vec{e}, event_c) = event_p(x).(([x = lift]).\\
&T_2(event_p, \vec{e}, event_c) + ([x \neq lift]).T_1(event_p, \vec{e}, event_c))\\
&T_2(event_p, \vec{e}, event_c) = \overline{event_c(x)}.\langle dial-tone\rangle.\\
&T_3(event_p, \vec{e}, event_c)\\
&T_3(event_p, \vec{e}, event_c) = event_p(x).\\
&\quad\left(\begin{array}{l}([x = hang-up]).T_t(event_p, \vec{e}, event_c)\\ +([x \neq hang-up]).T_3(event_p, \vec{e}, event_c)\end{array}\right)
\end{aligned} \tag{2}$$

Obviously, employing either the observation method or the ®-calculus formula, we can easily verify the consistency of the model. However, this model lacks enough details; it only describes the case of normal dial-up. In the following, we will refine this model by adding the timeout analysis during the period of dialing. The refined sequence diagram is shown in Figure 3. Because of the possibility of timeout, we append an alt-operator into Figure 1. Then, the semantics of phone is:

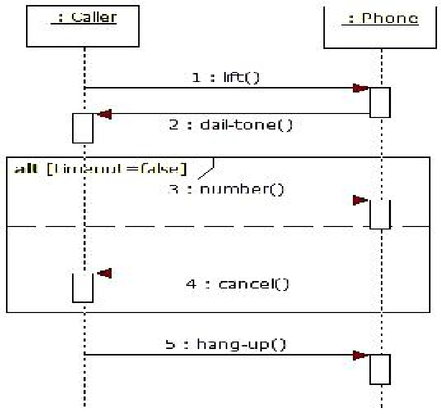

Figure 3. Refined sequence diagram.

$$\begin{aligned}
&S_1(mes_p, \vec{m}', mes_c) = mes_p(x)([x = lift].\overline{mes_c}\\
&\quad < dial-tone >.S_2(mes_p, \vec{m}', mes_c) +\\
&\quad [x \neq lift].S_1(mes_p, \vec{m}', mes_c))\\
&S_2(mes_p, \vec{m}, mes_c) = \overline{mes_c} < dial-tone >.\\
&\quad S_3(mes_p, \vec{m}, mes_c)\\
&S_3(mes_p, \vec{m}', mes_c) = mes_p(x)([timeout = T].\\
&\quad \overline{mes_c} < cancel >.S_4(mes_p, \vec{m}', mes_c) + [timeout = F]\\
&\quad .mes_p(x).([x = number].S_4(mes_p, \vec{m}', mes_c)\\
&\quad + [x \neq number].S_3(mes_p, \vec{m}', mes_c))\\
&S_4(mes_p, \vec{m}', mes_c) = mes_p(x).([x = hang-up].\\
&\quad S_t(mes_p, \vec{m}', mes_c) + [x \neq hang-up].\\
&\quad S_t(mes_p, \vec{m}', mes_c))
\end{aligned} \tag{3}$$

The process set of phone and the channels of message are changed into {S_1, S_2, S_3, S_4, S_t} and $\vec{m}' = \{lift, dial-tone, number, cancel, hang-up\}$ respectively.

The state chart diagram focuses on describing the behavior of a single object, while the sequence diagram describes the message passing among different objects. Therefore, the state chart diagram can be viewed as the refinement or the detailed description of the sequence diagram. A model can function normally only when the semantics of these two diagrams are consistent, According to the messages added in Figures 3 and 4 indicates the refinement of Figure 2.

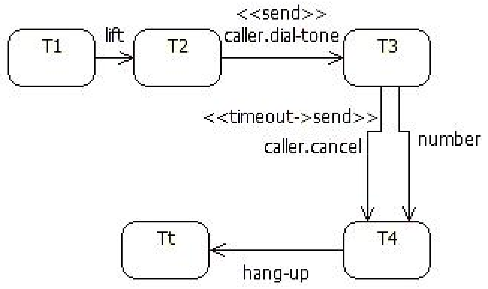

Figure 4. Refined State chart diagram of phone.

The added state and transition of the refined state chart diagram corresponds to the added messages in the sequence diagram. Thus, the semantics can be revised as:

$$\begin{aligned}
&T_1\left(event_p,\vec{e}',mes_c\right)=event_p\left(x\right).([x=lift].\\
&T_2\left(event_p,\vec{e}',event_c\right)+[x\neq lift].T_1\left(event_p,\vec{e}',event_c\right))\\
&T_2\left(event_p,\vec{e}',event_c\right)=\overline{event_c\left(x\right)}.\langle dial-tone\rangle.\\
&\quad T_3\left(event_p,\vec{e}',event_c\right)\\
&T_3\left(event_p,\vec{e}',event_c\right)=[timeout=T].\overline{event_c\left(x\right)}.\langle cancel\rangle.\\
&\quad T_4\left(event_p,\vec{e}',event_c\right)+[timeout=F].event_p\left(x\right)\\
&\quad\begin{pmatrix}[x=number].T_4\left(event_p,\vec{e}',event_c\right)+\\ [x\neq number].T_3\left(event_p,\vec{e}',event_c\right)\end{pmatrix}\\
&T_4\left(mes_p,\vec{e}',mes_c\right)=event_p\left(x\right).\\
&\quad\begin{pmatrix}[x=hang-up].T_t\left(event_p,\vec{e}',event_c\right)\\ +[x\neq hang-up].T_4\left(event_p,\vec{e}',event_c\right)\end{pmatrix}
\end{aligned}\tag{4}$$

In the above formulas, channels are changed to $\vec{e}'=\{lift,dial-tone,number,cancel,hang-up\}$ mean while, a process is added to corresponds to the state T_4.

By comparing the ®-calculus in the refinement models of Figures 3 and 4, we can tell that the revision of these two models is roughly the same. Thus, how to determine whether Figures 3 and 4 are consistent based on the verification of whether the two semantics equivalent?. Lam proposed to submit the complete specification of each diagram and re-import MWB for verification. But with rounds of refinement, the model will become increasingly complicated. The efficiency will be greatly reduced and unnecessary mistakes will be likely increased if each time we need to verify the ® semantics that corresponds to the whole model. Thus, we propose a substitution rule to combine the logic verification with MWB. We can only verify the added or the revised parts in a refinement model after verifying the ®-formula of the initial model by MWB. Our method can be applied to the modularity development and the integration of the modularities, which greatly simplifies the verification process and improves the efficiency.

4.2. Weak Open Bisimulation

First, we introduce the operational semantics of processes and actions in ®-calculus:

- $P\xrightarrow{\sim}P'$: The execution of action α and process P becomes P'.
- $P\Rightarrow P'$: Process P becomes P' after zero or more internal actions.
- $P\overset{\sim}{\Rightarrow}P'$: $P\Rightarrow\xrightarrow{\sim}\Rightarrow P'$ is equivalent to $P\Rightarrow\xrightarrow{\sim}\Rightarrow P'$.

$$P\overset{\hat{\sim}}{\Rightarrow}P':\begin{cases}P\overset{\sim}{\Rightarrow}P' & if\ \sim\neq{}^2\\ P\Rightarrow P' & if\sim={}^2\end{cases}$$

When triggered by a series of events, two states will generate the same behavior sequence. Similarly, when triggered by the same message, two objects will produce the same behavior sequence. Under this circumstance, we can say that these states or objects are bisimulation or observation equivalence. This concept is useful in studying the behaviors of complex systems. Milner [12] proposed bisimulation relation which is often used to describe the mutual imitation between two systems. Bisimulation can be viewed as the behavior equivalence of a system. Similar concepts include homomorphism and isomorphism. However, homomorphism allows a source structure to be embedded in a target structure. Thus, the similarity between the two structures is only one-way, that is, the target structure can be similar to the source structure, while it may not be the case the other way around. Isomorphism requires two algebra systems are the same or equipotential, which can be too strict. The bisimulation relation is in between. It allows bi-directional behavior simulations, while does not require the strict equipotential assumption. According to different demands, researchers have proposed a variety of bisimulation relations, such as: The difference between the strong and the weak bisimulation is whether the internal action τ and the action α (input, output action) are treated equally [20]. In the UML modeling, the inter-process communication is more important than the internal action. Regardless what the internal structure of the two dynamic diagrams is these two diagrams are equivalent as long as both have the same output for a series of external actions. Therefore, the weak open bisimulation is more suitable to verify structural congruence. The semantics and syntax of the weak open bisimulation are defined as follows.

The weak open bisimulation is a symmetric binary relation $\sim$ that is defined in the process set. If $\forall P,Q$ holds $P\ \square\ Q\Leftrightarrow\forall\sim(P\overset{\sim}{\Rightarrow}P'(\exists Q(Q\overset{\sim}{\Rightarrow}Q'\wedge P'\ \square Q')))$, then P

and Q can be viewed as weak open bisimulation or observation equivalence.

4.3. Equivalence Rules

Equivalence between sequence diagram and state chart diagram: Assume object *O*, corresponds to the state chart diagram *F*, the message sequence is $\vec{m} = \{m_1 \cdots m_k\}$ and the event sequence is $\vec{e} = \{e_1 \cdots e_n\}$, *O* is equivalent to *F* if their ®-formula are weak open bisimulation that is $S_0(mes_O, \vec{m}) \sim ({}^3\text{c}_{event}(e_1) \cdot\cdot {}^3\text{c}_{event}(e_n))\text{c}(F)$, abbreviated as $\text{c}(O) \sim \text{c}(F)$. $S_0(mes_O, \vec{m})$ is the initial process of the *O*'s ®-formula.

From the observation point of view, equivalence means that given the stimulation of any message/event, the sequence diagram and the state chart diagram will produce the bisimulation behavior. In ®-semantics, it can be explained that the processes derived from the sequence diagram's initial process through any message channel is the bisimulation with the processes derived from the state chart diagram's initial process through the event channels that corresponds the above message channels.

- **Substitution Rule**: Assume $P \square Q$, for any process *A*, If $bn(A) \cap bn(P,Q) = ¤, fn(A) \cap fn(P,Q) = ¤$, then $P + A \square Q + A, P \,|\, A \;\square Q \,|\, A$.

  Proof: $\forall \sim$,

$$P + A \overset{\sim}{\Rightarrow} \begin{cases} P' & if \sim \in bn(P) \cup fn(P) \\ A' & if \sim \in bn(A) \cup fn(A) \end{cases}$$
$$Q + A \overset{\sim}{\Rightarrow} \begin{cases} Q' & if \sim \in bn(Q) \cup fn(Q) \\ A' & if \sim \in bn(A) \cup fn(A) \end{cases}$$
$$\because P \square Q$$
$$\therefore P' \square Q'$$
$$\therefore P + A \square Q + A$$

  The same reason as $P \,|\, A \square Q \,|\, A$.

- **The Largest Weak Open Bisimulation Set**: An orderly pair set is constructed by all the bisimulation equivalent processes of object *O* and its state chart diagram in the ®-formula.

$$M = \{\langle S_i, T_i \rangle \,|\, S_i \square T_i, S_i \in \text{c}(O), T_i \in \text{c}(F)\}$$

  For example, in Figures 1 and 2, as $S_1(mes_p, \vec{m}', mes_c) \xrightarrow{mes_p(x)}$ is matched with $T_1(event_p, \vec{e}', event_c) \xrightarrow{event_p(x)}$, $\xrightarrow{mes_c(x)} S_2(mes_p, \vec{m}', mes_c) \xrightarrow{mes_p(x)}$ is matched with $\xrightarrow{event_c(x)} T_2(event_p, \vec{e}', event_c) \xrightarrow{event_p(x)}$, $S_3(mes_p, \vec{m}', mes_c) \xrightarrow{mes_p(x)}$ is matched with $T_3(event_p, \vec{e}', event_c) \xrightarrow{event_p(x)}$, $\xrightarrow{mes_c(x)} S_t(mes_p, \vec{m}', mes_c)$ is matched with $\xrightarrow{event_C(x)} T_t(event_p, \vec{e}', event_c)$, so the largest weak open bisimulation set of Figures 1 and 2 is:

$$M = \{\langle S_1, T_1 \rangle, \langle S_2, T_2 \rangle, \langle S_3, T_3 \rangle, \langle S_t, T_t \rangle\}$$

- **The Equivalence Rule of Refined Model**: Assume object $O\check{C}SD$, the corresponding state chart diagram is F, $\text{c}(O) = \{S_1, \cdots, S_m\}, \text{c}(F) = \{T_1, \cdots, T_n\}$ and M is the largest weak open bisimulation set between these two, after the refinement of SD and F, the modified pairs in M are $\langle S_i, T_i \rangle \cdots \langle S_j, T_j \rangle$, then $\text{c}(O) \sim \text{c}(F)$ while $(S_i \sim T_i) \wedge \cdots (S_j \sim T_j)$.

  Proof: By the definition of the largest weak open bisimulation set and the substitution rule.

Based on the above analysis, the following gives the verification method of the refined model:

- First, build the weak bisimulation equivalence set of the initial model. Furthermore, ensure there are no messages/events that share the same name in the refined model (if there are, rename the bounded variable). Now, the equivalence set is $M = \langle S_1, T_1 \rangle, \langle S_2, T_2 \rangle, \langle S_3, T_3 \rangle, \langle S_t, T_t \rangle$.
- Next, find the equivalence pair in the refined model whose semantics has been modified. In the equivalence set of $\text{c}(F_3)$ and $\text{c}(F_4)$, only the semantics of $\langle S_3, T_3 \rangle$ is modified.
- Finally, we can assure that the bisimulation between $\text{c}(F_3)$ and $\text{c}(F_4)$ can be maintained by verifying the equivalence between the modified pair $\langle S_3, T_3 \rangle$. Using the bottom-up approach, because of $S_t \square T_t$, by the substitution rule, we can attain $S_4 \square T_4$. According to the substitution rule again, $S_3 \square T_3$ is obtained, $\therefore \text{c}(F_3) \sim \text{c}(F_4)$ and the weak open bisimulation equivalence set is $M = \{\langle S_1, T_1 \rangle, \langle S_2, T_2 \rangle, \langle S_3, T_3 \rangle, \langle S_4, T_4 \rangle \langle S_t, T_t \rangle\}$.

5. Conclusions and Future Research

As the two most important dynamic diagrams that describe behaviors, the sequence diagram and the state chart diagram have a major impact on model semantics.

In this paper we discussed a semantics consistency check of UML specifications. We proposed an approach for the refined sequence and state diagrams. Employing ®-calculus as the semantics specification, we can verify the refinement of these two diagrams in a level-by-level manner. The main contribution of our work is the simplification of formal details from the designers.

Furthermore, we are currently implementing our verification between single sequence diagram and single state chart diagram. This paper is a first step towards the use of formal verification in the current software modeling. As a prolongation of this investigation, it would be interesting to extract this added verification for other types of multiple UML diagrams to serve more complex software systems.

References

[1] Bonfè M., Fantuzzi C., and Secchi C., "Design Patterns for Model-based Automation Software Design and Implementation," *Control Engineering Practice*, vol. 21, no. 11, pp. 1608-1619, 2013.

[2] Bouabana-Tebibel T. and Rubin S., "An Interleaving Semantics for UML 2 Interactions using Petri Nets," *Information Sciences*, vol. 232, pp. 276-293, 2013.

[3] Cabot J., Clarisó R., and Riera D., "On the Verification of UML/OCL Class Diagrams using Constraint Programming," *Journal of Systems and Software*, vol. 93, pp. 1-23, 2014.

[4] Chen X. and Li X D., "Design Calculus Based Approach to Modeling Use Case," *Journal of Software*, vol. 19, no. 10, pp. 2539-2549, 2008.

[5] Elammari M. and Issa Z., "Using Model Driven Architecture to Develop Multi-Agent Systems," available at: http://itgate.info/elammari/mda.pdf, last visited 2013.

[6] Heuer A., Stricker V., Budnik C., Konrad S., Lauenroth K., and Pohl K., "Defining Variability in Activity Diagrams and Petri Nets," *Science of Computer Programming*, vol. 78, no. 12, pp. 2414-2432, 2013.

[7] Lam V. and Padget J., "Analyzing Equivalences of UML State Chart Diagrams by Structural Congruence and Open Bisimulations," *in Proceedings of IEEE Symposium on Human Centric Computing Languages and Environments*, Auckland, pp. 137-144, 2003.

[8] Lam V. and Padget J., "Consistency Checking of Statechart Diagrams of a Class Hierarchy," *in Proceedings of the 19th European Conference on Object-Oriented Programming*, Glasgow, pp. 412-427, 2005.

[9] Lam V. and Padget J., "Consistency Checking of Sequence Diagrams and Statechart Diagrams using the π-calculus," *in Proceedings of the 5th International Conference on Integrated Formal Methods*, Eindhoven, pp. 347-365, 2005.

[10] Lano K. and Kolahdouz-Rahimi S., "Constraint-based Specification of Model Transformations," *Journal of Systems and Software*, vol. 86, no. 2, pp. 412-436, 2013.

[11] Li Z., Chen H., and Wang B., "The Symbolic Transition Graphs and the Early Bisimulation Algorithm of π-calculus," *Science in China Series E: Technological Sciences*, vol. 29, no. 4, pp. 361-371, 1999.

[12] Milner R., *Communication and Concurrency*, Prentice Hall, 1989.

[13] Milner R., Joachim Parrow J., and David Walker., "A Calculus of Mobile Process," *Information and Computation*, vol. 100, no. 1, pp. 1-40, 1992.

[14] Miyazaki H., Yokogawa T., Amasaki S., Asada K., and Sato Y., "Synthesis and Refinement Check of Sequence Diagrams," *IEICE transactions on Information and Systems*, vol. 95, no. 9, pp. 2193-2201, 2012.

[15] OMG, available at: http://www.omg.org, last visited 2013.

[16] Prehofer C., "Behavioral Refinement and Compatibility of State chart Extensions," *Electronic Notes in Theoretical Computer Science*, vol. 295, pp. 65-78, 2013.

[17] Queralt A., Artale A., Calvanese D., Teniente E., "OCL-Lite: Finite Reasoning on UML/OCL Conceptual Schemas," *Data and Knowledge Engineering*, vol. 73, pp. 1-22, 2012.

[18] Rajabi B. and Lee S., "Consistent Integration between Object Oriented and Coloured Petri Nets Models," *The International Arab Journal of Information Technology*, vol. 11, no. 4, pp. 406-415, 2014.

[19] Victor B. and Molier F., "The Mobility Workbench - A Tool for the π-Calculus," *in Proceedings of the 6th International Conference on Computer Aided Verification*, Stanford, USA, pp. 428-440, 1994.

[20] Yao C., "Bisimulation Is Going into the Modal Logic," *Studies in Philosophy of Science and Techno logy*, vol. 27, no. 3, pp. 36-39, 2010.

[21] Zhao Y., Yang Z., and Xie J., "Π-calculus based Assembly Mechanism of UML State Diagram and Validation of Model Refinement," *in Proceedings of International Conference on Electronic Computer Technology*, Macau, pp. 604-609, 2009.

Zhou Xiang received her ME degree in 2000. Currently, she is a lecturer in Qingdao University and studying towards PhD degree of Computer Science at East China University of Science and Technology. She has been engaged in research on formal semantics and software modeling.

Shao Zhiqing received his MS degree in Pure Mathematics from Institute of Software Chinese Academy of Science and PhD degree in Computer Software from Shanghai Jiao Tong University. Currently, he is a professor in East China University of Science and Technology. His research interests include software verification and network service.