Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 8 (3): 152-168, 2014
ISSN 1819-4311 / DOI: 10.3923/jse.2014.152.168

Quantitative Verification of Trustworthy Service Flow by Stochastic Model Checking

[1,2]Yang Liu

[1]School of Information Science and Technology, Taishan University, Taian, Shandong, 271021, China

[2]Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210093, Jiangsu, China

ABSTRACT

Trustworthy Service Flow (TSF) is one of the most representative paradigm for trustworthy Software. Because TSF involves the non-functional attribute, such as quality of service and trustworthiness, the modelling and verification of which is very difficult. It is not sufficient to meet the TSF requirements just using the classical model checking. This paper propose a method to quantitative verification of TSF with stochastic model checking. Firstly, based on the extension of OWL-S upper ontology, the formal semantics for TSF is presented by Nondeterministic Probabilistic Petri Net (NPPN) and the translation process of which is implemented automatically. Moreover, the simplification rules for NPPN and is also presented. Then, the new temporal logic as PCTL is proposed for specifying the properties of TSF and the stochastic model checking algorithm is put forward for quantitative verification TSF. The feasibility and effectiveness of the method are illustrated by the available service flow case.

Key words: Trustworthy service flow, nondeterministic probabilistic petri net, formal semantics, stochastic model checking

INTRODUCTION

Computer based software systems have been applied to the national economy, national defense construction, daily life and so on. However, some errors occurred in the safety-critical software application, which may lead to substantial financial consequences and even threaten our lives. So, the trustworthiness of software is a very important issue for the applications. The trustworthiness is a comprehensive reflection in the user's point of view about many attributes of the objective, which is based on the concept of accuracy, reliability, security, timeliness, completeness, availability, predictability, survivability, controllability, etc. In recent years, Trustworthy software has been widely concerned in formal verification community and various government organizations and institutions of many countries have presented the research planning about the trustworthy software theory. It will be not exaggerated to say that basic principle of software theory is changing from the correctness-oriented traditional engineering theory to the trustworthiness-oriented novel software theory (Liu *et al*., 2008).

Web service is a programmable module with standard interface descriptions that provide universal accessibility through standard communication protocols (Zhang *et al*., 2007). As a new a new network-oriented software development paradigm, composing atomic web service to provide the value-added services which is named service flow has been widely accepted in the academia and

industry. So, study on trustworthiness of service flow (trustworthy service flow,TSF (Liu *et al.*, 2011a) has important theoretical and practical significance, which not only helps to achieve SOA (service-oriented architecture) and will enrich the theory and practice of trustworthy software.

Because the running environment for service flow owns the characteristics of the openness, distribution and autonomy and the heterogeneity and dynamic of the service flow itself, the TSF is very difficult to implement. In particular, the application requirements, running environment and user request may be changeable, which lead to further increase the complexity and difficulty to achieving the TSF (Zeng *et al.*, 2010).

In reference Liu *et al.* (2011a), we proposed a reference model and the construction algorithm for TSF and pointed out that trustworthiness of service flow is comprehensive evaluation of service properties, such as correctness, dependability, security and efficiency, which emphasize the trust extent of user subject about service object. Therefore, we argue that it is difficult to adapt the requirements of the TSF just relying on traditional formal models and verification techniques.

On the basis of extending the upper ontology model of OWL-S(Web Ontology Language for Services), this paper puts forward the formal semantics for TSF using Nondeterministic Probabilistic Petri Net (NPPN) (Liu *et al.*, 2013) and verifies the formal semantics model by stochastic model checking algorithm.

The rest of this paper is organized as follows: Section 2 introduces the basic concept of NPPN and TSF; Section 3 extends the upper ontology model of OWL-S for TSF; the semantics model for TSF is formalized in section 4; section 5 presents the verification framework and supported tool; section 6 shows the related works about TSF; section 7 concludes the study and point out the directions for future work.

PRELIMINARIES

Non deterministic probabilistic Petri net (NPPN): Petri net is one of the most popular formal method for modelling and analyzing system, which is proposed by professor Petri in 1960s. For the reason of modelling the probabilistic characteristics of system behavior, the probabilistic Petri net is defined by some reference (Varacca and Nielsen, 2003; Kudlek, 2005; Albanese *et al.*, 2008). Actually, system behaviors often have mixed non-deterministic and probabilistic characteristics simultaneously, so we presented NPPN in reference (Liu *et al.*, 2013). NPPN can be seen as the extension of probabilistic Petri net with nondeterminism, or the extension of Petri net with probability under the condition of keeping the nondeterminism, which is defined as follows:

- **Definition 1:** (NPPN) NPPN can be defined as a 5-tuple M = (S, T: F, f; C), where: (1) T = (Transition, Prt), transition denotes the transition act, Prt0 [0, 1] denotes the success probability of the transition; T is the Act Transition (AT) with the probability Eq. 1, or the probabilistic act transition (AT, Prt) with an act satisfying a certain probability distribution, or the pure Probability Transition (PT) without any act. If Prt = 0, it means the transition is invalid; (2) $S\cap T = \phi$, $S\cup T\neq\phi$, $F\subseteq S\times T\cup T\times S$, which is the flow relation of net and N = (S, AT, F) is the pure net, where AT is the act transition with the probability Eq. 1 (3) $f = f_T\cup f_S\cup f_{S\times T}\cup f_{T\times S}$, f_T: $T\rightarrow 2^{Transition\times Prt}$, $f_S = S\rightarrow[0, 1]$, $f_{S\times T} = S\times T\rightarrow[0, 1]$, $f_{T\times S} = T\times S\rightarrow[0, 1]$ and the value of $f_{S\times T}$ is determined by the nondeterminism of transitions, the value of $f_{T\times S}$ can be obtained from the value of $\sigma_{Prt}(f_T(t))$, the value of f_S except the initial place can be computed according to the value of $f_{S\times T}$ and $f_{T\times S}$; (4) $\forall t\in T, \exists s\in S, f_{T\times S} = 1-\sigma_{Prt}(f_T(t))$. $f_{T\times S}(t\times s) = 0$, if $\sigma_{Prt}(f_T(t)) = 1$, which

represents transition (t, s) and place s are invalid; (5) C is the set of nondeterminism classes and each nondeterminism class is a set comprised of (s, t_i) and:

$$\sum_{i=1}^{n} f_{S\times T}(s,t_i)=1, \text{If } \{(s,t_1),(s,t_2)...(s,t_n)\}\in C$$

Trustworthy service flow (TSF): Using fuzzy set to describe and evaluate the trustworthiness attribute of service, we have presented the definition of TSF (Liu *et al*., 2011b), which should be satisfied by the following conditions:

$$QoS_{TSF} = \max\{QoS_{SF1}, QoS_{SF2},..., QoS_{SFk}\} \quad (1)$$

$$TW_{TSF} = \max\{TW_{SF1}, TW_{SF2}, ..., TW_{SFk}\}, k=\prod_{i=1}^{m} n_i \quad (2)$$

where, QoS (quality of service) is computed by normalization, TW (trustworthiness) is evaluated by fuzzy set theory, QoS_{SFk} denote the QoS of similar service flow(SF) k and TW_{SFk} denote the TW of similar Service Flow (SF) k. Obviously, for the reason that two conditions may not satisfied simultaneously, dynamic weighted aggregation method is adopted to deal with the two conditions.

EXTENDING SEMANTIC WEB SERVICES MODEL

In order to resolve the deficiencies of information representation, discovery and extraction in the current network environment, Berners Lee *et al*. (2001) Scientific AmericanTM, May 17, 2001 proposed machine-understandable information (Semantic web) and gave the hierarchy structure of the Semantic web. Semantic web aims to achieve intelligent management of network resources and the ontology is the key technology to achieve the semantic web. Semantic web services technology attempts to combine semantic web and web services organically to create a more powerful distributed computing environment, which is an important development direction of the next generation network-computing (Curbera *et al*., 2003). Among the standards program of semantic web services, WSMO (Web Service Modeling Ontology) (Lausen *et al*., 2005) and OWL-S (Web Ontology Language for Services) (Martin *et al*., 2004) are the two most representative ones. Lara *et al*. (2004) presented a specific comparison for WSMO and OWL-S about objectives, components, hierarchy, language support, scalability and so on. To calculate the trustworthiness of service (semantic web services, abbr. Service), we take into account the merits of both WSMO and OWL-S and extend OWL-S to EOWL-S model for TSF, which is shown in Fig. 1.

According to the upper ontology model for EOWL-S, service WS can be defined as a four-tuple: WS (Goal, QoS, TW, Grounding), where: (1) Goal is the service objective which refers to the function implemented by the service operation or the state reached after service operation. Goal is defined as a triple: Goal (ActionName, Cocndition, Effect), where ActionName is the target action name, Condition is the action preconditions, Effect is the effects after action; (2) QoS is the quality of service. There are already some literature (Yu *et al*., 2007; Voas, 2003; Cardoso *et al*., 2004) discussed about the QoS, Voas (2003) listed a lot of software quality attributes, such as reliability, security, testability, maintainability, response time (Yu *et al*., 2007; Cardoso *et al*., 2004) gave the quality metrics for a single service. According to reference (Liu *et al*., 2011b), QoS is defined as a four-tuple: QoS (T, C, R, A), where T denotes response time, C denotes Cost, R denotes

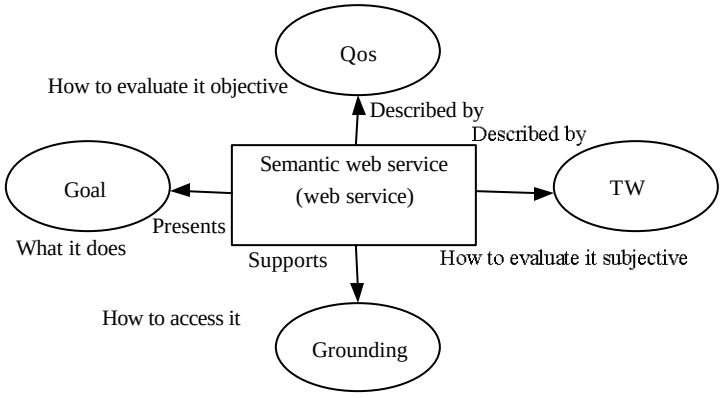

Fig. 1: Upper ontology model for EOWL-S

Table 1: Process model of EOWL-S

```
<eowl:Class rdf:ID="Process">
<rdfs:comment></rdfs:comment>
<eowl:unionOf rdf:parseType="Collection">
   <eowl:Class rdf:about="#AtomicProcess"/>
   <eowl:Class rdf:about="#SimpleProcess"/>
   <eowl:Class rdf:about="#CompositeProcess"/>
</eowl:unionOf>
</eowl:Class>
```

reliability, A denotes availability; (3) TW is the trustworthiness, which is valued by fuzzy set in reference (Liu *et al*., 2011a); (4) Grounding (Martin *et al*., 2004) is the description of the service, which relate to the specific service specification. Grounding describes how the service is being accessed, which needs to specify the service access protocols, message formats, port and so on.

MODELLING TRUSTWORTHY SERVICE FLOW

This section will discuss how to using NPPN to model TSF is discussed, which described by EOWL-S process model, that is NPPN formal semantics for TSF.

EOWL-S process model as defined in Table 1 is the same with the OWL-S process model, which contains the atomic process, the simple process and the composite process. Atomic process cannot be divided, which can be called directly without invoking other services; Simple process provides a variety of abstraction mechanism for some similar process view that may represent atomic processes or composite process, but cannot be invoked or associated with Grounding. Composite process achieves its function by combination of atomic process with control structure. According to reference (Martin *et al*., 2004), the EOWL-S control constructs include: Sequence, split, split+join, choice, any-order, condition, If-then-else, repeat-while and repeat-until.

Atomic process: NPPN semantics for atomic process is shown in Fig. 2, where: t1 denotes receiving the message or data, t implement the function of the atomic process, t2 returns the result,

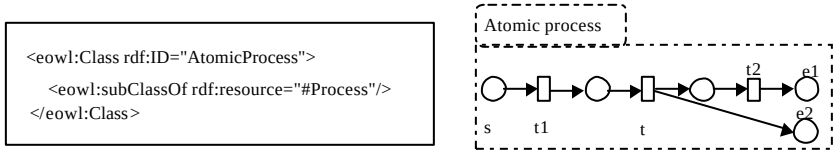

Fig. 2: Atomic process and its formal semantic

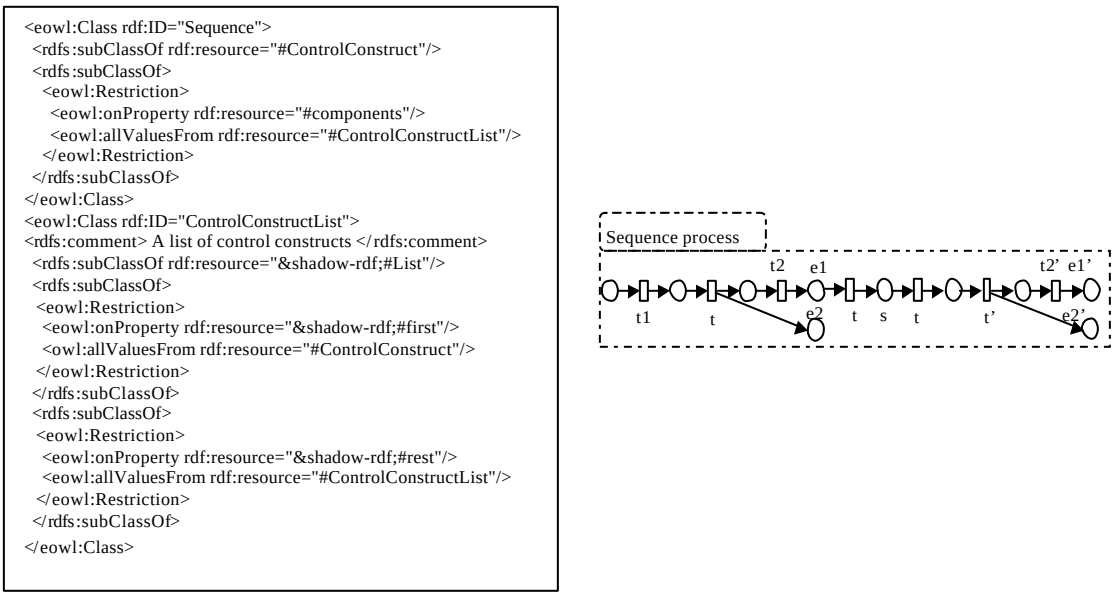

Fig. 3: Sequence process and its formal semantic

e1 denotes the success place, e2 denotes the failure place and the value of $\sigma_{Prt}(f_T(t))$ is the success ratio in reference (Liu *et al.*, 2011a), the value of $\sigma_{Prt}(f_T(t1))$ is 1, the value of $\sigma_{Prt}(f_T(t2))$ is 1.

Sequence process: The sequence process is a set of atomic processes which execute sequentially. Take the sequence process composed by 2 atomic processes for example, the NPPN semantics for which is shown in Fig. 3, where: tt is the augmented transition, the value of $\sigma_{Prt}(f_T(tt))$ is 1 and the other symbols have the same meanings with the atomic process respectively.

Split process: The split process is a set of atomic processes which execute concurrently. Take the split process composed by 2 atomic processes for example, the NPPN semantics for which is shown in Fig. 4, where: *ss* is the augmented place, tt is the augmented transition, the value of $\sigma_{Prt}(f_T(tt))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

Split+join process: The split+join process is a set of atomic processes which execute concurrently with the final synchronization operation. Take the split+join process composed by 2 atomic processes for example, the NPPN semantics for which is shown in Fig. 5, where: *ss* and *ee* are the augmented places, tt1 and tt2 are the augmented transitions, the value of $\sigma_{Prt}(f_T(tt1))$ is 1, the value of $\sigma_{Prt}(f_T(tt2))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

```
<eowl:Class rdf:ID="Split">
 <rdfs:subClassOf rdf:resource="#ControlConstruct"/>
 <rdfs:subClassOf>
  <eowl:Restriction>
   <eowl:onProperty rdf:resource="#components"/>
   <eowl:allValuesFrom rdf:resource="#ControlConstructBag"/>
  </eowl:Restriction>
 </rdfs:subClassOf>
</eowl:Class>
<eowl:Class rdf:ID="ControlConstructBag">
<rdfs:comment> A multiset of control constructs </rdfs:comment>
 <rdfs:subClassOf rdf:resource="&shadow-rdf;#List"/>
 <rdfs:subClassOf>
  <eowl:Restriction>
   <eowl:onProperty rdf:resource="&shadow-rdf;#first"/>
   <eowl:allValuesFrom rdf:resource="#ControlConstruct"/>
  </eowl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <eowl:Restriction>
   <eowl:onProperty rdf:resource="&shadow-rdf;#rest"/>
   <eowl:allValuesFrom rdf:resource="#ControlConstructBag"/>
  </eowl:Restriction>
 </rdfs:subClassOf>
</eowl:Class>
```

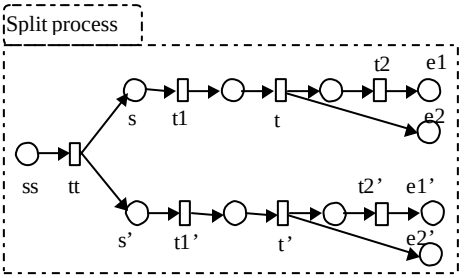

Fig. 4: Split process and its formal semantic

```
<eowl:Class rdf:ID="Split+Join">
 <rdfs:subClassOf rdf:resource="#ControlConstruct"/>
 <rdfs:subClassOf>
  <eowl:Restriction>
   <eowl:onProperty rdf:resource="#components"/>
   <eowl:allValuesFrom rdf:resource="#ControlConstructBag"/>
  </eowl:Restriction>
 </rdfs:subClassOf>
</eowl:Class>
```

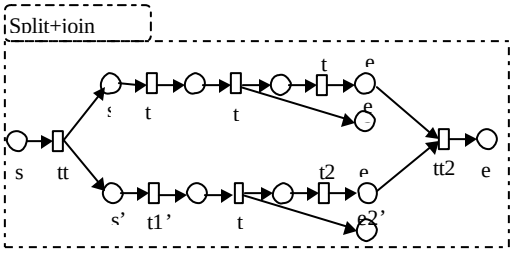

Fig. 5: Split+join process and its formal semantic

```
<eowl:Class rdf:ID="Choice">
  <rdfs:subClassOf rdf:resource="#ControlConstruct"/>
  <rdfs:subClassOf>
    <eowl:Restriction>
     <eowl:onProperty rdf:resource="#components"/>
     <eowl:allValuesFrom
rdf:resource="#ControlConstructBag"/>
    </eowl:Restriction>
  </rdfs:subClassOf>
</eowl:Class>
```

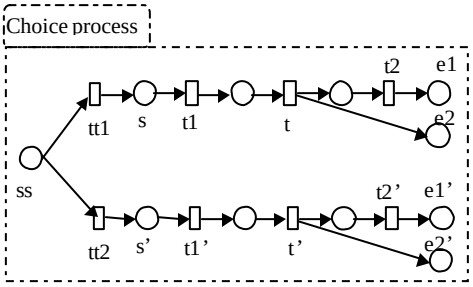

Fig. 6: Choice process and its formal semantic

Choice process: The choice process is a set of atomic processes which is selected one arbitrarily to execute. Take the choice process composed by 2 atomic processes for example, the NPPN semantics for which is shown in Fig. 6, where: ss is the augmented place, tt1 and tt2 are the augmented transitions, the value of $\sigma_{Prt}(f_T(tt1))$ is 1, the value of $\sigma_{Prt}(f_T(tt2))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

If-then-else process: The If-then-else process is a set of atomic processes which is selected one under some conditions to execute. Take the choice process composed by 2 atomic processes for

```
<eowl:Class rdf:ID="If-Then-Else">
  <rdfs:subClassOf rdf:resource="#ControlConstruct"/>
  <rdfs:subClassOf>
    <eowl:Restriction>
      <eowl:onProperty rdf:resource="#components"/>
      <eowl:allValuesFrom rdf:resource="#ControlConstructBag"/>
    </eowl:Restriction>
  </rdfs:subClassOf>
</eowl:Class>
<eowl:ObjectProperty rdf:ID="ifCondition">
 <rdfs:comment> The if condition of an if-then-else</rdfs:comment>
  <rdfs:domain rdf:resource="#If-Then-Else"/>
  <rdfs:range rdf:resource="&expr;#Condition"/>
</eowl:ObjectProperty>
<eowl:ObjectProperty rdf:ID="then">
  <rdfs:domain rdf:resource="#If-Then-Else"/>
  <rdfs:range rdf:resource="#ControlConstruct"/>
</eowl:ObjectProperty>
<eowl:ObjectProperty rdf:ID="else">
  <rdfs:domain rdf:resource="#If-Then-Else"/>
  <rdfs:range rdf:resource="#ControlConstruct"/>
</eowl:ObjectProperty>
```

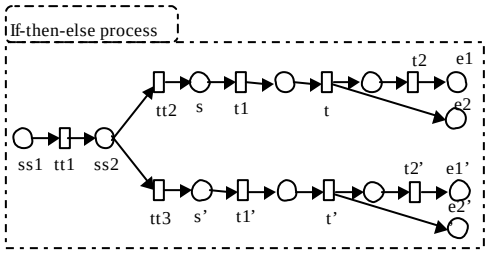

Fig. 7: If-then-else process and its formal semantic

```
eowl:Class rdf:ID="Any-Order">
 <rdfs:subClassOf rdf:resource="#ControlConstruct"/>
 <rdfs:subClassOf>
  <eowl:Restriction>
   <eowl:onProperty rdf:resource="#components"/>
   <eowl:allValuesFrom
rdf:resource="#ControlConstructBag"/>
  </eowl:Restriction>
 </rdfs:subClassOf>
</eowl:Class>
```

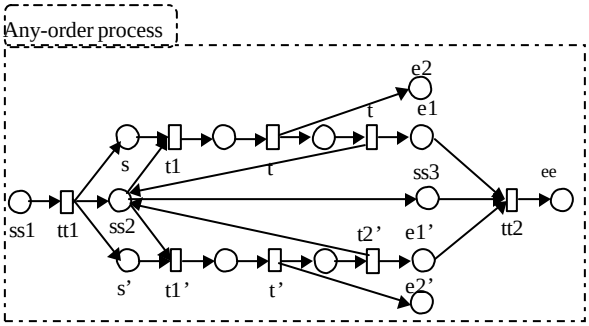

Fig. 8: Any-order process and its formal semantic

example, the NPPN semantics for which is shown in Fig. 7, where: ss1 and ss2 are the augmented places, tt1, tt2 and tt3 are the augmented transitions, the value of $\sigma_{Prt}(f_T(tt1))$ is 1, the value of $\sigma_{Prt}(f_T(tt2))$ is 1, the value of $\sigma_{Prt}(f_T(tt3))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

Any-order process: The any-order process is a set of atomic processes which execute at any order. Take the any-order process composed by 2 atomic processes for example, the NPPN semantics for which is shown in Fig. 8, where: ss1, ss2 and ee are the augmented places, tt1 and tt2 are the augmented transitions, the value of $\sigma_{Prt}(f_T(tt1))$ is 1, the value of $\sigma_{Prt}(f_T(tt2))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

Repeat-while process: The repeat-while process is a set of atomic processes which execute circularly under the condition is true. Take the Repeat-while process composed by 1 atomic processes for example, the NPPN semantics for which is shown in Fig. 9, where: ss1, ss2 and ss3 are the augmented places, tt1, tt2 and tt3 are the augmented transitions, the value of $\sigma_{Prt}(f_T(tt1))$ is 1, the value of $\sigma_{Prt}(f_T(tt2))$ is 1, the value of $\sigma_{Prt}(f_T(tt3))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

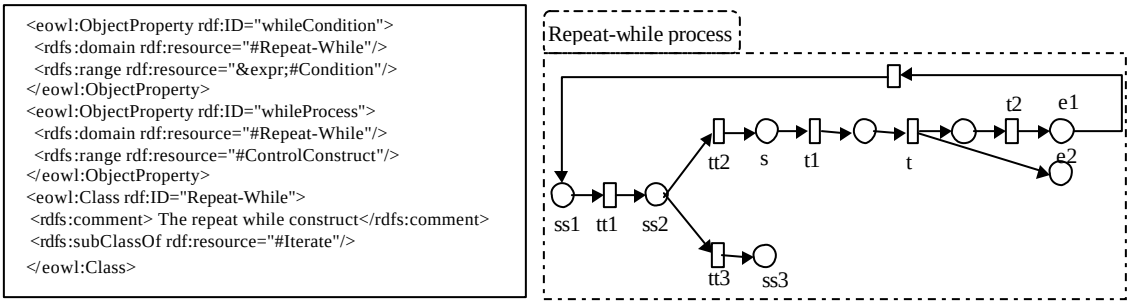

Fig. 9: Repeat-while process and its formal semantic

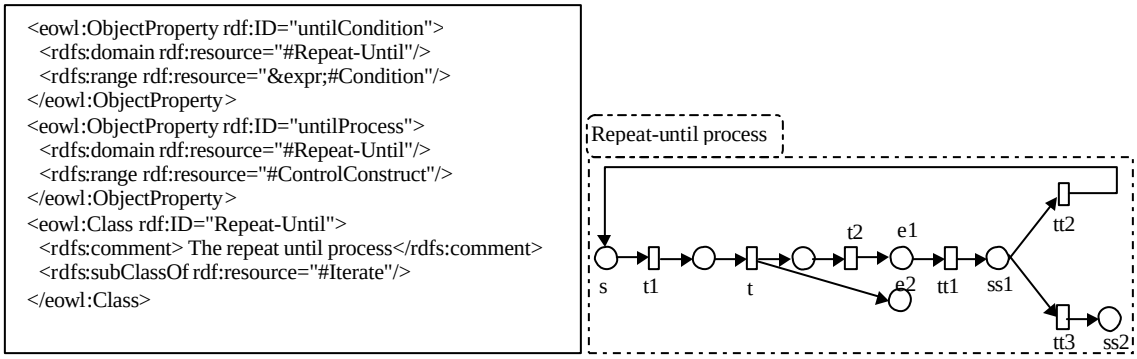

Fig. 10: Repeat-until process and its formal semantic

Repeat-until process: The Repeat-until process is a set of atomic processes which execute once and then execute circularly under the condition is true. Take the Repeat-until process composed by 1 atomic processes for example, the NPPN semantics for which is shown in Fig. 10, where: ss1 and ss2 are the augmented places, tt1, tt2 and tt3 are the augmented transitions, the value of $\sigma_{Prt}(f_T(tt1))$ is 1, the value of $\sigma_{Prt}(f_T(tt2))$ is 1, the value of $\sigma_{Prt}(f_T(tt3))$ is 1 and the other symbols have the same meanings with the atomic process, respectively.

QUANTITATIVE VERIFICATION OF TRUSTWORTHY SERVICE FLOW

The framework for TSF verification is shown in Fig. 11, the principle of which can be stated succinctly that given the NPPN model that is the semantics for TSP and the action/state-based PCLT formulae that describe the quantitative requirement specification, determine whether the model NPPN satisfies the action/state-based PCLT automatically through the algorithm. The NPPN model can be automatically generated by the translation tool, but the action/state-based PCLT formulae have to be generated manually at present.

Automatic generating NPPN model

Translation from EOWL-S process to NPPN: Based on the Java language, using the Eclipse we developed the prototype tool OWLS2NPPN for translating the EOWL-S process model into NPPN model automatically, the structure of which is shown in Fig. 12. The translation process is: (1) Select the file of EOWL-S process model which will be translated, as shown in Fig. 13; (2) Parse EOWL-S process model, extract the control structure information from the EOWL-S process model

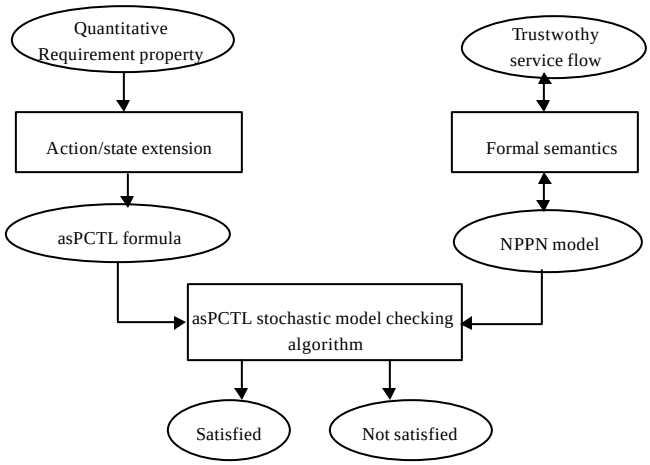

Fig. 11: Quantitative verification framework

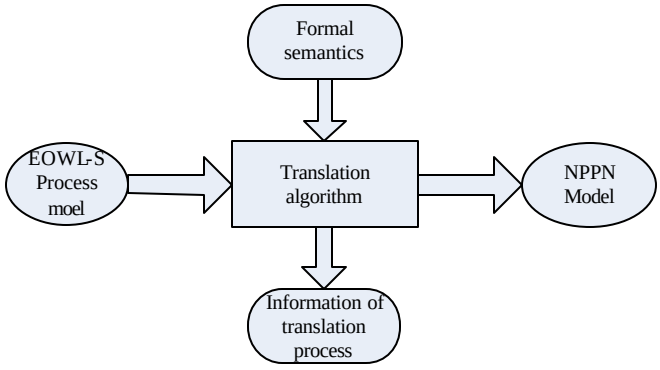

Fig. 12: OWLS2NPPN tool structure

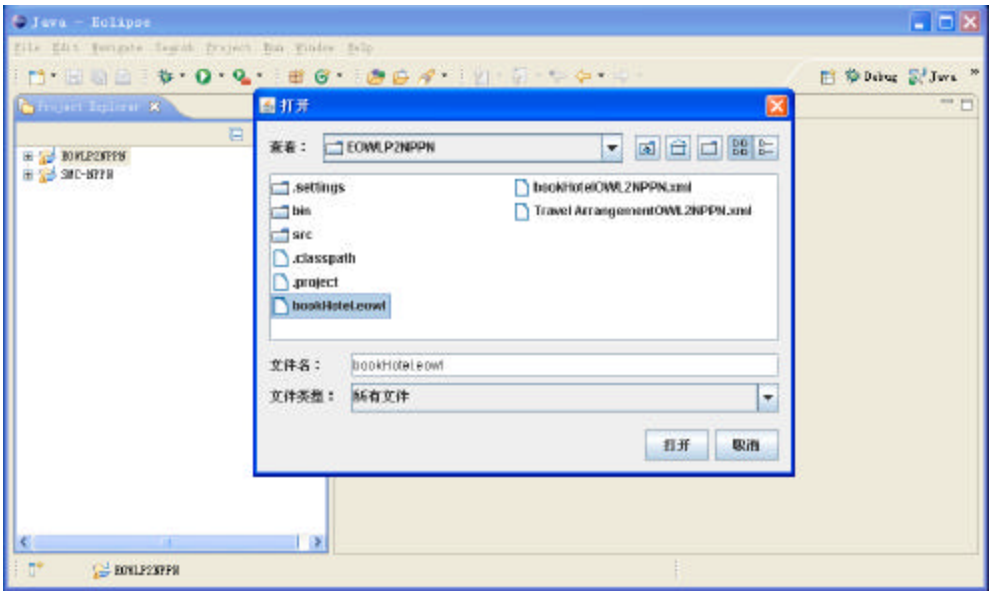

Fig. 13: Select the EOWL-S process model

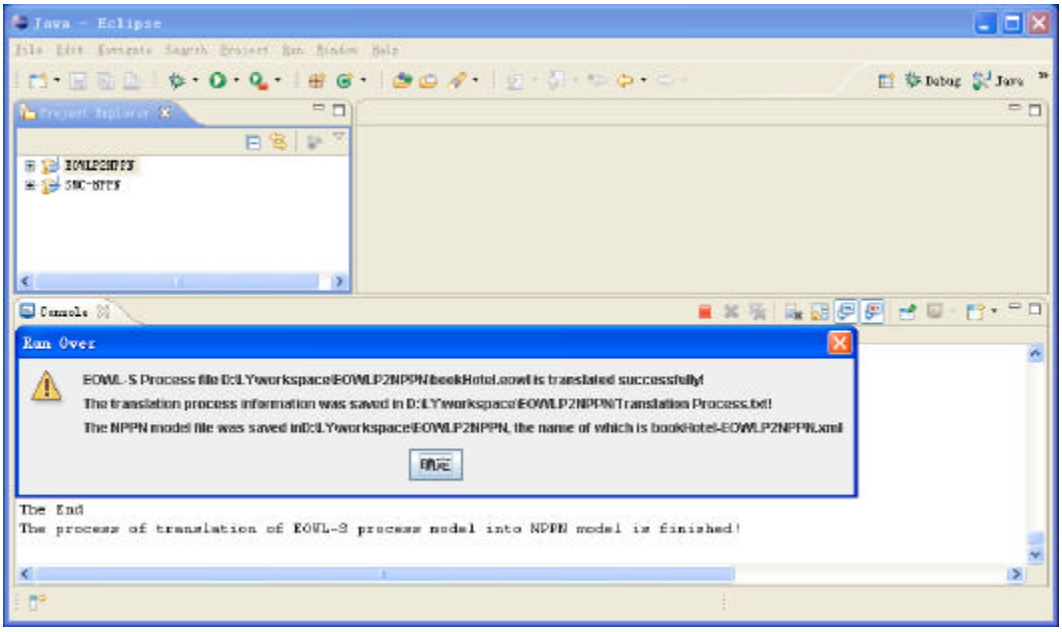

Fig. 14: Model translation finished

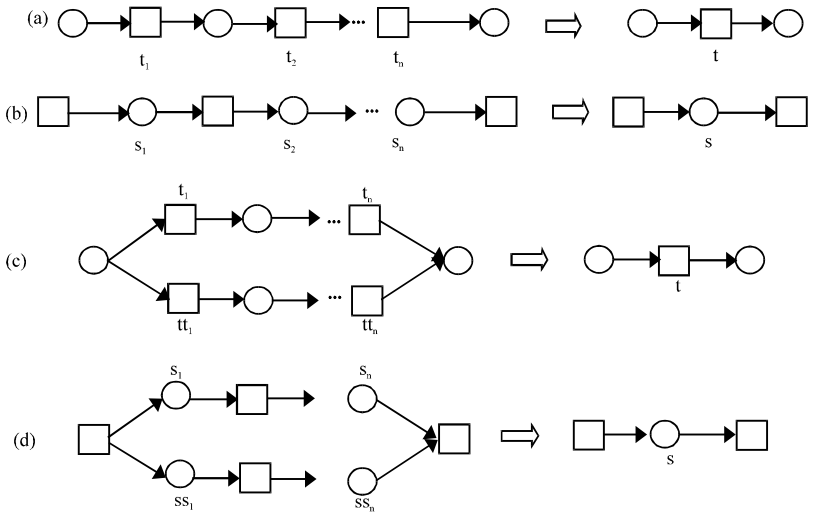

Fig. 15(a-d): Simplification rules, (a) Sequential transitions simplification, (b) Sequential places simplification, (c) Selective transitions simplification and (d) Current places simplification

and use the tree structure with children-brother chain for storage the related information; (3) Post-order traverse the tree structure, generate the NPPN model described by the PNML (Petri Net Markup Language) and output the information, as shown in Fig. 14.

Take the *BookHotel* process for example, which is extend based on: http://www.di.unipi.it/~brogi/projects/owls-epository/processmodels/(travel)_HotelService.owl.xml. The translated and specified NPPN model is shown in Fig. 16.

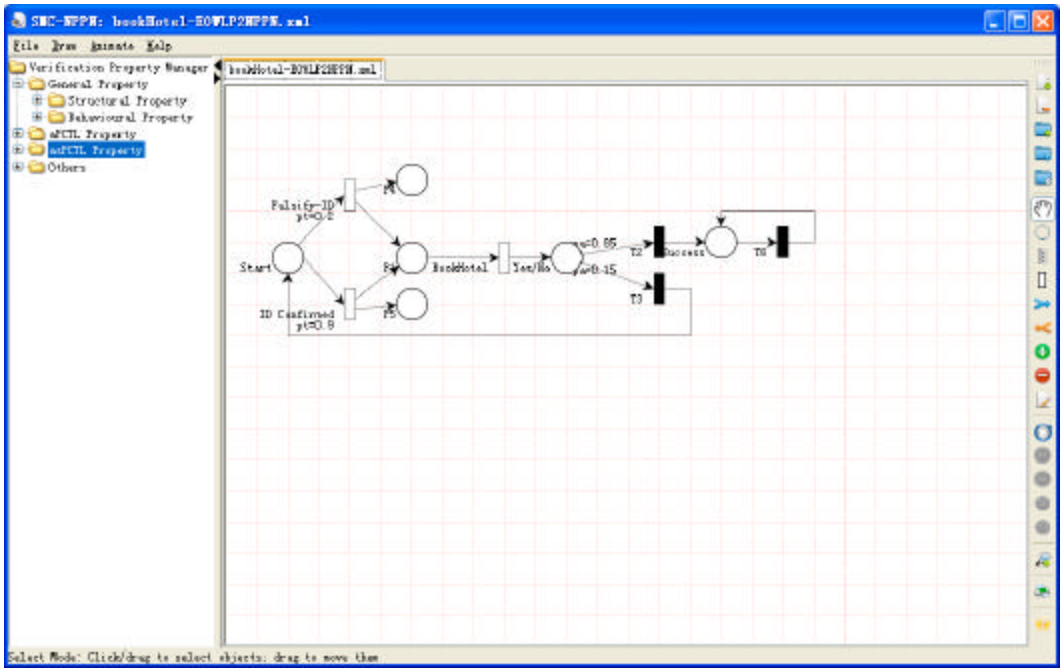

Fig. 16: NPPN model for *BookHotel* process

NPPN model simplification: Simplification is one of common operations for NPPN, which is the powerful means for analyzing behaviors of complex system model. The simplification operations used usually for NPPN are summarized as follows:

- Sequential transitions, simplification rule for which is shown in Fig. 15(a) and the probability of the simplified t is:

$$\sigma_{Prt}(f_T(t))=\prod_{i=1}^{n}\sigma_{Prt}(f_T(t_i))$$

- Sequential places, simplification rule for which is shown in Fig. 15(b) and the probability of the simplified s is:

$$f_s(s)=\prod_{i=1}^{n}f_S(S_i)$$

- Selective transitions, simplification rule for which is shown in Fig. 15(c) and the probability of the simplified s is:

$$f_S(S)=\max\left(\prod_{i=1}^{n}\sigma_{Prt}(f_T(t_i)),\prod_{i=1}^{n}\sigma_{Prt}(f_T(tt_i))\right)$$

- Current places, simplification rule for which is shown in Fig.15 (d) and the probability of the simplified s is:

$$f_S(S) = \min\left(\prod_{i=1}^{n} f_S(s_i), \prod_{i=1}^{n} f_S(SS_i)\right)$$

Model checking for NPPN model

asPCTL: The quantitative property specification for TSF can be described by temporal logic with probability extension, such as PCTL (Hansson and Jonsson, 1994), aPCTL (Liu *et al*., 2013), LTL+probability. Taking into account that semantic model NPPN for TSF is formal model with state and transition treated equally, this paper presents the action/state-based PCLT (asPCTL) to describe the quantitative property specification for TSF, which can be seen as the state-extension of aPCTL, or the action-extension of PCTL.

Definition 2 (Syntax of asPCTL): Let $p\in[0, 1]$, which indicates probability, $\alpha\in\{<,\leq, >, \geq\}$, which indicates probability operator, ap action proposition, sp the sate proposition, state formulaeof asPCTL can be expressed by BNF, $\Phi ::= \text{true} \mid ap \mid sp \mid \Phi\wedge\Phi \mid \neg\Phi \mid P_{ap}$; Let k be the non-negative integer or infinity, which indicates the number of step, AB are the action set, which indicate sets of act, path formulae Ψ of asPCTL can be expressed by BNF:

$$\Psi ::= \Phi U^{\leq k}\Phi \mid \Phi_A U^{\leq k}\Phi \mid \Phi_A U_B^{\leq k+1}\Phi$$

Definition 3 (Semantics of asPCTL): Let M be a NPPN model $\pi = s_0 \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2 \xrightarrow{t_3} \ldots$, the asPCTL state and path formulae are interpreted over the places and execution paths. Let Sat $(\Phi) = \{s\in S \mid s = \Phi\}$, the satisfaction relation for state formulae are defined by:

$s \vDash \text{true}$	
$s \vDash ap$	if f $ap\epsilon L(s')$
$s \vDash sp$	if f $sp\epsilon L(s)$
$s \vDash \neg\Phi$	if $s \nvDash \Phi$
$s \vDash \Phi_1\wedge\Phi_2$	if f $s \vDash \Phi_1$ and $s \vDash \Phi_2$
$s \vDash P\sim p(\Psi)$	if f $Pr(s, \Psi)\sim p$

Let i is the non-negative integer and $\pi(i) = s_i$, the satisfaction relation for path formulae are defined by:

$\pi \vDash \Phi_1 U^{\leq k}\Phi_2$	if f $\exists 0\leq i\leq k\ (\pi(i) \vDash \Phi_2\wedge (\forall\leq j<i\ (\pi(j) \vDash \Phi_1)_A))$
$\pi \vDash \Phi_{1A} U^{\leq k}\Phi_2$	If f $\exists 0\leq i\leq k\ (\pi(i) \vDash \Phi_2\wedge (\forall\leq j<i\ (\pi(j)\Phi_1))\wedge(\pi(j)\rightarrow\pi(j+1)))$
$\pi \vDash \Phi_{1A} U_B^{\leq k+1}\Phi_2$	if f $\exists 0\leq i\leq k\ (\pi(i) \vDash \Phi_2\wedge (\forall\leq j<i\ (\pi(j) \vDash \Phi_1))\wedge(\pi(j)^A\rightarrow\pi(j+1))\wedge(\pi(i-1)\rightarrow\pi(i)))$

asPCTL Model checking algorithm: asPCTL Model checking algorithm for NPPN model can be viewed as the combination of aPCTL model checking algorithm for NPPN model (Liu *et al*., 2013) and PCTL model checking algorithm, the specific process of which is shown in Table 2.

According to algorithm process, time complexity of algorithm is analysed as follows. Let the number of flow relation be $|F|$ and the number of sub-formula included in formula Φ be $|\Phi|$, each sub-formula has to be executed satisfiability computing once, so execution number of computation is $|\Phi|$; the time for transforming logic operation into flow relation set operation is $O(|F|)$; therefore, the time complexity of model checking algorithm proposed in this paper is $O(|\Phi|(|F|+k_{max}))$.

Table 2: asPCTL model checking NPPN system algorithm

Input: NPPN model M, initial state s_0, asPCTL formulae Φ
Output: model M satisfies the formulae Φ or not

```
SMC_NPPN(M, s0, Φ){
Sat(Φ)=Computing (M, s0, Φ);
If (s0 ∈Sat (v)) print("model M satisfies the formulae Φ");
 else print("model M does not satisfy the formulae Φ");
}
Computing(M, s0, Φ){
 translate the formulae Φ into the forms of Φ1 ∧Φ2, ¬Φ or P⋈p ()
 parse the syntax tree of Φ by recursively depth-first traversing it
 switch(Φ)
 case true: return S
 case ap: return { s∈S | ap∈L (s)}
 case sp: return { s∈S | ap∈L (s)}
 case Φ1∧Φ2: Return Computing(M, s0, , Φ1) ∩Computing(M, s0, Φ2)
 case ¬Φ: Return S-Computing(M, s0,Φ)
 case P⋈p (Φ1U^sk Φ2): Return {s∈S | P2^min (Φ1U^skΦ2)∝p}
 case P⋈p (Φ1AU^sk Φ2): return {s∈S | Ps^min (Φ1AU^skΦ2)∝p}
 case P⋈p (Φ1AU_B^sk Φ2): return {s∈S | Ps^min (Φ1AU_B^sk+1Φ2)∝p}
}
```

Tool supporting and illustrative example: The tool SMC-NPPN for stochastic model checking NPPN is implemented based on NPNMV (Curbera *et al*., 2003), which contains all of the functions of NPNMV, can asPCTL model checking NPPN. Using the model *bookHotel-EOWLP2NPPN* as an illustrative examples to introduce asPCTL model checking process, which is translated and specified from 5.1.1 *BookHotel* process. As shown in Fig. 16, *Start* means initial place, *ID Confirmed* means identity confirmation and the success ratio of which is 0.9, *Falsify-ID* means identity falsification and the success ratio of which is 0.2, Yes/No means whether the hotel is booked and the success ratio of which is 0.85, the success ratio of *BookHotel* is 1, T2, T3 and T6 are probabilistic transition without action, P4, P5 and P6 are the failure places and the probability matrix is shown if Fig. 17. For comparing asPCTL with PCTL, the properties will be verified are: (1) asPCTL formula: $P_{\geq 0.7}$ ($_{\text{ID ConfirmedVBookHotelVT2}}$ F Success), (2) PCTL formula: $P_{\geq 0.7}$ (F Success). The quantitative verification process of NPPN model are as follows:

- **$P_{\geq 0.7}$ ($_{\text{ID ConfirmedVBookHotelVT2}}$ F Success)**
- According to syntax of as PCTL, F Success = true $_{\text{ID ConfirmedVBookHotelVT2 U Success}}$
- Choose which transition of Start, Because the nondeterminism of place Start. Pick the transition (Start, Falsify-ID), which can get it by matrix multiplication, that is: P (Start ⊨ true $_{\text{IDConfirmedVBookHotelVT2 U Success}}$) = 0.9*0.850.15*0.9*0.85+... = 0.9
- So, *bookHotel-EOWLP2NPPN* ⊨ $P_{\geq 0.7}$ ($_{\text{ID ConfirmedVBookHotelVT2}}$ F Success), the verification result is shown in Fig. 18
- **$P_{\geq 0.7}$ (F Success)**
- According to syntax of PCTL, F Success = true Success
- Choose which transition of Start, Because the nondeterminism of place Start. Pick the transition (Start, Falsify-ID), which can get it by matrix multiplication, that is: P (Start ⊨ true U Success)

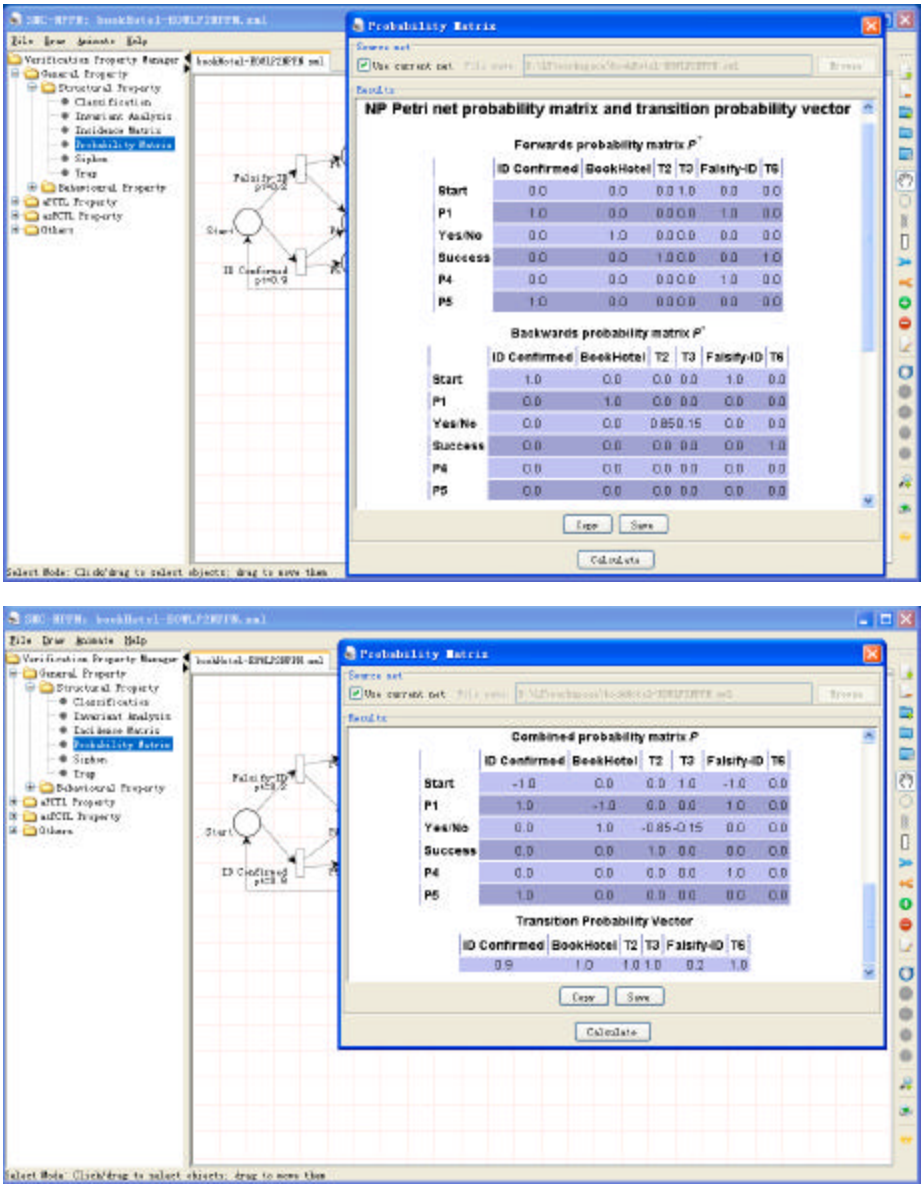

Fig. 17: Probability matrix for NPPN

= 0.2*0.85+0.15*0.2*0.85+... = 0.2

- So, NPPN model bookHotel-*EOWLP2NPPN* $\models P_{\geq 0.7}$ (F Success)

RELATED WORKS

Research on the trustworthy service flow can be divided into two categories: one is to use formal verification techniques to ensure that the functional behavior of the service flow model is correct, the other is to implement trustworthy service flow or service flow analysis which are both based on measure of the non-functional attributes. About the verification of functional behavior aspect: About the verification of functional behavior aspect: Vidal *et al.* (2007) presented a Petri net semantics for the choreography of semantics web services described in the OWL-S language, but they did not consider the non-functional attributes in the web service composition; Ouyang *et al.*

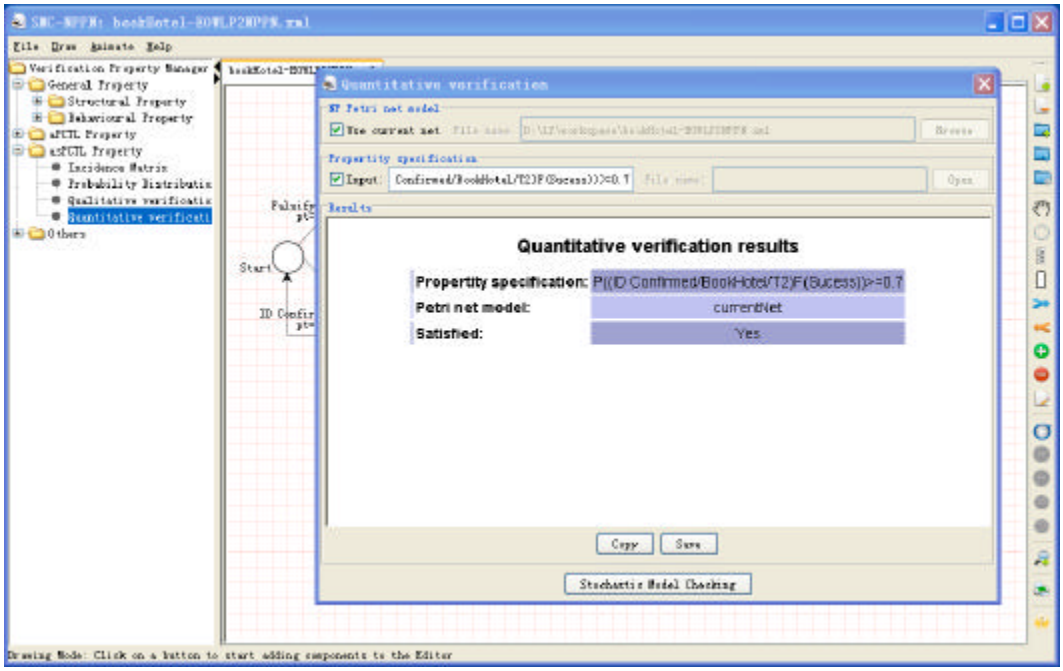

Fig. 18: Verification result

(2007) defined the mapping of BPEL constructs onto Petri net structures and used it for the analysis of various dynamic properties related to unreachable activities, conflicting messages, garbage collection, conformance checking and dead-locks and life locks in interaction processes, but did not involve the data flow information; Caliz *et al.* (2011) used the colored Petri net to analyze Web Services Choreography Description Language documents in order to identify faults in the specification, but also did not analyze the performance attributes. About the measure of the non-functional attributes aspect: Dai *et al.* (2009) and Fan *et al.* (2009) ensured the key non-functional attributes to meet the requirements during construction of service flow, which measured the non-functional attributes subset QoS, moreover, in reference (Fan *et al.*, 2009) the randomness of QoS is considered, but did not cover the fuzzy characteristic; Liu *et al.* (2011b) presented a method for constructing the u ser-constrained trustworthy service flow, based on trustworthiness evaluated by fuzzy set and normalized QoS; Calinescu *et al.* (2011) introduced a novel, tool-supported framework for the development of adaptive service-based systems, which can QoS be used to achieve QoS requirements through dynamically adapting to changes in the system state, environment and workload.

Based on the above study, this paper draws on the advantages of a two direction and gives a uniform method to quantitative trustworthy service flow about the non-functional attributes and functional attributes.

CONCLUSION

In this study, a feasible and effective stochastic model checking-based quantitative verification method for TSF is presented, which includes: (1) Automatic generalization of formal semantics model NPPN of TSF, (2) Using temporal logic asPCTL extended from PCTL to describe the quantitative requirement property, (3) asPCTL model checking algorithm for NPPN. And the work for future is how to tackle the state explosion of verification of the large TSF.

ACKNOWLEDGMENT

This study is supported by the National Natural Science Foundation of China under Grant No. 61303022, China Postdoctoral Science Foundation funded project under Grant No. 2013M531328, the Shandong Provincial Natural Science Foundation of China under Grant No. ZR2012FQ013, the Project of Shandong Province Higher Educational Science and Technology Program under Grant No. J13LN10 and the Science and Technology Program of Taian under Grant No. 201330629.

REFERENCES

Albanese, M., R. Chellappa, V. Moscato, A. Picariello, V.S. Subrahmanian, P. Turaga and O. Udrea, 2008. A constrained probabilistic petri net framework for human activity detection in video. IEEE Trans. Multimedia, 10: 1429-1443.

Berners-Lee, T., J. Hendler and O. Lassila, 2001. The semantic web. Scientific American™, May 17, 2001. http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

Calinescu, R., L. Grunske, M. Kwiatkowska, R. Mirandola and G. Tamburrelli, 2011. Dynamic QoS management and optimization in service-based systems. IEEE Trans. Software Eng., 37: 387-409.

Caliz, E., K. Umapathy, A.J. Sanchez-Ruiz and S.A. Elfayoumy, 2011. Analyzing web service choreography specifications using colored Petri nets. Proceedings of the 6th International Conference on Service-Oriented Perspectives in Design Science Research, May 5-6, 2011, Milwaukee, WI., USA., pp: 412-426.

Cardoso, J., A. Sheth, J. Miller, J. Arnold and K. Kochut, 2004. Quality of service for workflows and web service processes. Web Semantics, 1: 281-308.

Curbera, F., R. Khalaf, N. Mukhi, S. Tai and S. Weerawarana, 2003. The next step in web services. Commun. ACM, 46: 29-34.

Dai, Y., L. Yang and B. Zhang, 2009. QoS-driven self-healing web service composition based on performance prediction. J. Comput. Sci. Technol., 24: 250-261.

Fan, X.Q., C.J. Jiang, J.L. Wang and S.C. Pang, 2009. Random-QoS-aware reliable web service composition. J. Software, 20: 546-556.

Hansson, H. and B. Jonsson, 1994. A logic for reasoning about time and reliability. Formal Aspects Comput., 6: 512-535.

Kudlek, M., 2005. Probability in petri nets. Fundam. Inform., 67: 121-130.

Lara, R., D. Roman, A. Polleres and D. Fensel, 2004. A conceptual comparison of WSMO and OWL-S. Proceedings of the European Conference on Web Services, September 27-30, 2004, Erfurt, Germany, pp: 254-269.

Lausen, H., A. Polleres and D. Roman, 2005. Web service modeling ontology (WSMO). W3C Member Submission. http://www.w3.org/Submission/WSMO/

Liu, K., Z.G. Shan, J. Wang, J.F. He, Z.T. Zhang and Y.W. Qin, 2008. Overview on major research plan of trustworthy software. Bull. Natl. Nat. Sci. Found. China, 22: 145-151.

Liu, Y., H. Miao and Y. Ma, 2011a. An approach to constructing user-constrained trustworthy service flow. Chinese J. Electron., 20: 425-431.

Liu, Y., H. Miao, H. Zeng and Z. Li, 2011b. Probabilistic petri net and its logical semantics. Proceedings of the 9th International Conference on Software Engineering Research, Management and Applications, August 10-12, 2011, Baltimore, MD., USA., pp: 73-78.

Liu, Y., H.K. Miao, H.W. Zeng, Y. Ma and P. Liu, 2013. Nondeterministic probabilistic petri net: A new method to study qualitative and quantitative behaviors of system. J. Comput. Sci. Technol., 28: 203-216.

Martin, D., M. Burstein, J. Hobbs, O. Lassila and D. McDermott *et al.*, 2004. OWL-S: Semantic markup for web services. W3C Member Submission, 22 November 2004. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

Ouyang, C., H.M.W. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas and A.H.M. ter Hofstede, 2007. Formal semantics and analysis of control flow in WS-BPEL. Sci. Comput. Program., 67: 162-198.

Varacca, D. and M. Nielsen, 2003. Probabilistic petri nets and Mazurkiewicz equivalence. http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.9.303

Vidal, J.C., M. Lama and A. Bugarin, 2007. Petri net semantics for OWL-S service choreography. Proceedings of the International Workshop on Formal Approaches to Business Processes and Web Services, June 26, 2007, Siedlce, Poland, pp: 7-20.

Voas, J., 2003. Trusted software's holy grail. Software Qual. J., 11: 9-17.

Yu, T., Y. Zhang and K.J. Lin, 2007. Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans. Web, Vol. 1. 10.1145/1232722.1232728

Zeng, J., H.L. Sun, X.D. Liu, T. Deng and J.P. Huai, 2010. Dynamic evolution mechanism for trustworthy software based on service composition. J. Software, 21: 261-276.

Zhang, L.J., J. Zhang and H. Cai, 2007. Services Computing. Tsinghua University Press, Beijing, China, ISBN-13: 9783540382843, pp: 1-80.