Appl. Math. Inf. Sci. **8**, No. 4, 2017-2024 (2014) 2017

Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080460

Web Service Composition Automation based on Timed Automata

Hu Jingjing[1,*], *Zhu Wei*[1], *Zhao Xing*[2] *and Zhu Dongfeng*[3]

[1] School of Software, Beijing Institute of Technology, Beijing 100081 P. R. China
[2] School of Mathematics, Capital Normal University, Beijing 100037 P. R. China
[3] School of Computer Science & Technology, Beijing Institute of Technology, Beijing 100081 P. R. China

Received: 29 Aug. 2013, Revised: 1 Dec. 2013, Accepted: 2 Dec. 2013
Published online: 1 Jul. 2014

Abstract: Web service composition is a new direction in the research of service computing. To promote the portfolio, the key problem is to achieve efficient and automatic composition process. We propose the web service composition model based on timed automata. In the computing framework, we design the formal model and its construction algorithm; provide a web service interface description language and composition automation engine. In order to validate its performance, using UPPAAL as the service composition simulator, realized the automation process from independent web services into composite ones. The experimental results verify the feasibility of automatic service composition and the effectiveness of the proposed configuration.

Keywords: Web service composition; Timed automata; Automation

1 Introduction

Web services are the most famous implementation of service oriented architectures allowing the construction and the sharing of independent and autonomous software. Web service composition (WSC) consists in combining web services, developed by different organizations and offering diverse functional, behavioral and non-functional properties to offer more complex services. The spreading of services and WSC increases the difficulty and time for its applications. And the key technology of WSC is to provide a solution which can perform more efficient and automatic composition.

Several web service composition models are put forward to the research fields. On the one hand, WSC based on workflow is a common recognized approach [1]. This solution allows creating flows with composed activities, which models composed workflow with logics and provides the ability of services calling, data processing and exceptions checking. BPEL4WS is a commercial business process execution language designed for web service, and provides a method to describe workflow framework [2]. It has become a standard in web service composition. It is a static portfolio requiring manual intervention and difficult to provide a real-time combination with non-functional requirement [3,4]. On the other hand, WSC based on AI planning provides another way [5]. OWL-S is used to model non-functional properties, and every single web service is built as an action in AI, each of which contains an initial state, a target and some possible state-transition paths. Thus, WSC based on AI planning is a dynamical portfolio supporting complex behavioral and non-functional properties, but it lacks verification [6].

Although there are lots of methods on WSC, the approach to automatic composition is urgent because it can improve the efficiency and correctness of service composition, while it is also hard to actualize because it is difficult to perform standard operation for modeling web service and logical composition.

Timed automata are theories for modeling and verification of real time systems [7]. A timed automaton is a finite automaton extended with a set of real-valued variables modeling clocks. Constraints on the clock variables are used to restrict the behavior of an automaton, and accepting conditions are used to enforce progress properties. The timed automata can describe the real-time systems formally and provide the high-efficiency property verification of real-time systems.

* Corresponding author e-mail: hujingjing@bit.edu.cn

But the state space explosion exists for the clocks reset [8].

Timed automata can be combined with web service composition [9]. And the research in the field is mainly focused on modeling and property verification, which is the straightforward way to implement the service composition [10]. However, it is not possible to perform the whole process from single web services to service composition with logic flow. That is to say, it couldn't achieve WSC directly in these modes.

We propose how to use the timed automata to model composed web service and implement the web service composition automation. It provides a formal model built on timed automata for web service composition, and constructs the algorithm to implement this model automatically. The model and algorithm are tested to be feasible and efficient.

The rest of the chapter is organized as follows: In the next section, the timed automation model for WSC is proposed, and the construction algorithm is proposed. Section 3 presents a web service interface language and the implementation of automation engine. Section 4 describes the test cases and performance evaluations for the model and methods. Finally, it gives a brief conclusion and acknowledgement.

2 WSC model based on Timed Automata

Definition 1 (Timed automata, **TA** [11]): A timed automata model is a tuple $< N, l_0, E, I >$ where N is a finite set of locations (or nodes), $l_0 \in N$ is the initial location, $E \subseteq N \times \beta(C) \times \Sigma \times 2^C \times N$ is the set of edges, and $I : N \to \beta(C)$ assigns invariants to locations. when We shall write $l \overset{g,a,r}{\to} l'$ when $< l, g, a, r, l' > \in E$. C is a set of real-value variables or clocks ranged over by x, y, z etc. Σ is a set of actions ranged over by a, b, c etc. Atom Clock Constraint is a formula like $x \sim n$ or $x - y \sim n$, where $x, y \in C, \sim \in \{\leq, <, =, >, \geq\}$ and $n \in N$. Clock Constraint is a formed-formula of atom clock constraints ranged over guard g, D etc. $\beta(C)$ is a set of clock constraints.

The theory is the outstanding work of Alur and Dill [12]. Many verification tools (like UPPAAL) are built on it [13].

Definition 2 (Timed Automata for Web Service, **TAW**) An atom web service can be modeled as a timed automaton $< N, l_0, E, I >$ where N is a finite, non- empty set of states of web service, l_0 is the initial state, E is the set of transition function, which represents the evolution from a state to another, and I assigns the clock constraints to the service calling. The clocks indicate the cost in the current migration routes. Especially, the TAW head model is the starting of TAC.

Definition 3 (Timed Automata for WSC, **TAC**): It is an integration of timed automata describing the whole composited web service, where the TAW head model is added to start the TAC, the branch constraints are reduced and the global clock is reset.

Table 1: Algorithm A-TAC

01 Extract web service interface for each atom web service by semantic and build its equivalent graph.
02 Traverse the equivalent graph to generate the equivalent tree.
03 Identify nodes of the equivalent tree by breadth-first traversal.
04 Generate the TAW model for each node in equivalent graph according to the identified order in equivalent tree, as in section 2.2.
05 Insert TAW head model as the starting model.
06 Remove redundant clock constraints from each TAW model.
07 Reset global clock as the container of result.

Table 2: Algorithm A-TAW

01 Generate the fission graph for each parameter of nodes in equivalent graph by semantics.
02 When the parameter is active, set 'guard' as clock constraints, 'reset' as local clocks and branch tags which are set to the corresponding ones for the true value.
03 When the parameter is passive, set 'guard' as clock constraints and the value of its corresponding active branch tag is true. The 'reset' is also set as local clocks.
04 Link 'l_0' of each fission graph with all branches of its previous fission in proper order except the first parameter.
05 Insert a TAW head as the first node which points to the first parameter. Set 'guard' as the ending tags of its previous TAW except the first head.

2.1 Algorithm for web service composition

The algorithm is to generate a timed automaton model for each web service interface and they are synchronized through branches and end tags. The algorithm of constructing TAC model for web service composition (A-TAC) is shown as Tab 1.

The equivalent graph is a topology which connects each web service interface by equivalence relation. The equivalent tree is a data structure without loop which is generated by breadth-first traversing the equivalent graph.

2.2 TAW for single node in equivalence graph

The algorithmA-TAW to get TAW model for a single node in equivalence graph is shown as Tab.2.

For the parameters of web service interface represented by the nodes of equivalent tree, their value intervals can be divided into several parts, which

corresponds to the jump from one node to several nodes respectively. This part of TAW is called fission graph.

In algorithm A-TAC, the global clock records the summary of cost. The minimal cost computed by TAC model remains into the global clock, and the path implementing the value of clock will be the best solution for the WSC.

3 Service composition automation

The automation of WSC is implemented by the web service composition automation engine, and the framework of it is shown in Fig.1. The web service

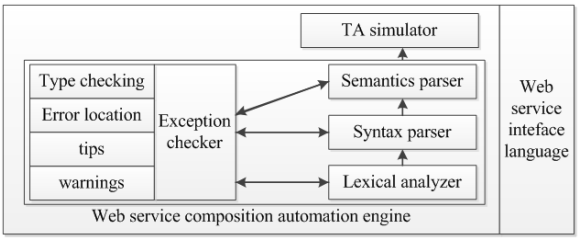


Fig. 1: Framework of web service composition automation

composition automation engine is a pipeline inside and its final result is a TAC model.

3.1 Web service interface description language

There need to be a simple and specific language to describe the web service interface in order to implement the service composition automation. This section presents the web service interface description language (WSIL) denoted by context-free grammar which is shown in Tab.3.

WSIL is a structured language and able to describe web service interface. It contains the equivalent relations of parameters. Thus, WSIL provides the input standard needed to get the equivalent graph for the WSC automation engine. The engine is essentially a compiler whose grammar is that of the WSIL itself and the results of semantic analysis are such data structures as equivalent graph or equivalent tree for web service interface. The compiler is constructed by an integrated tool of Context-free Grammar which wraps Lex and YACC [14].

3.2 Semantics parser of automation engine

The flow of analysis and parsing in web service composition automation engine is shown as Fig.2. The

Table 3: Web service interface description language

```
01 <Start> ::= <ObjectList>;
02 <ObjectList> ::= <Object><ObjectList> | null;
03 <Object> ::= <Param> | <Interface>;
04 <Param> ::= “class” Identifier “:” “Param” “{“ <DefinitionList> “}”;
05 <DefinitionList> ::= <Definition><DefinitionList> | null;
06 <Definition> ::= <Variable> | <Method>;
07 <Variable> ::= <Modifier> <Type> Identifier “;” | <Type> Identifier “;”;
08 <Method> ::= <Modifier> <Type> Identifer “(“ <FunctionParamList> “)” “{“ <FunctionBody> “}” | <Type> Identifer “(“ <FunctionParamList> “)” “{“ <FunctionBody> “}”;
09 <Modifier> ::= “public” | “private”;
10 <Type> ::= Identifier | <BasicType>;
11 <BasicType> ::= “int” | “float” | “double” | “char” | “byte” | “string” | “array”;
12 <FunctionParamList> ::= <FunctionParam> | <FunctionParam> “,” <FunctionParamListRight> | null;
13 <FunctionParam> ::= <type> Identifier;
14 <FunctionParamListRight> ::= “,” <FunctionParam> <FunctionParamListRight> | null;
15 <FunctionBody> ::= <SentenceList>;
16 <SentenceList> ::= <Sentence> <SentenceList> | null;
17 <Sentence> ::= <Sequence> | <If> | <For>;
18 <Sequence> ::= <Type> Identifier “;” | <Calculation> “;”;
19 <Calculation> ::= “Identifier” <SelfCalcu> | Identifier “=“ <Expression>;
20 <SelfCalcu> ::= “++” | “–“;
21 <If> ::= “if” “(“ <BoolExression> “)” <Block> <ElseBlock>;
22 <Expression> ::= <Multiply> <PlusOpt>;
23 <PlusOpt> ::= “+” <Multiply> | “-” <Multiply> | null;
24 <Multiply> ::= <Unit> <MultiplyOpt>;
25 <MultiplyOpt> ::= “*” <Unit> | “/” <Unit> “%” | null;
26 <Unit> ::= identifier | “(“ <Expression> “)” | number;
27 <BoolExpression> ::= “true” | “false” | <Value> <CompareOpt> <Value>;
28 <Value> ::= Identifier | “true” | “false”;
29 <CompareOpt> ::= “==” | “!=” | “>” | “<” | “>=” | “<=”;
30 <Block> ::= “{“ <SentenceList> “}” | Identifier “=” <Expression>;
31 <ElseBlock> ::= “else” <Block> | null;
32 <For> ::= “for” “(“ < Calculation > “;” <BoolExpression> “;” <Calculation> “)” <Block>;
33 <Interface> ::= “interface” Identifier “{“ <InterfaceParamList> “}”;
34 <InterfaceParamList> ::= <InterfaceParam> <InterfaceParamList> | null;
35 <InterfaceParam> ::= <Type> Identifier;
```

semantics parser gets semantic information by traversing the syntax tree of web service interface language and takes out the instance of TAC model.

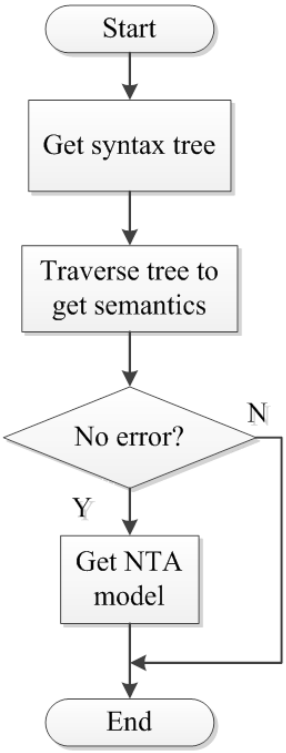


Fig. 2: The flow of semantics parser

In the process of recursive traversal, the keywords matching (such as the fixed identifiers of class and interface etc. in grammars) is used to distinguish the information of parameters and web service interfaces, and filled into independent data structures. The class diagram to implement the feature is shown in Fig.3.

The semantics parser analyzes syntax tree and get all information needed for the NTA. NTA is the model that can be received and verified by the TA tools. The algorithm to generate NTA is as Tab.4.

UPPAAL is able to read the NTA model directly after NTA is generated. The whole process from atomic web service to composed one is finished automatically.

4 Performance evaluation

The feasibility and effectiveness of the presented WSC model and its automation engine with UPPAAL are evaluated in this chapter.

The version of UPPAAL used is 4.0, and JRE6.0 is also needed. The running environment is CPU: Intel 2.40GHZ, RAM: 3.0GB.

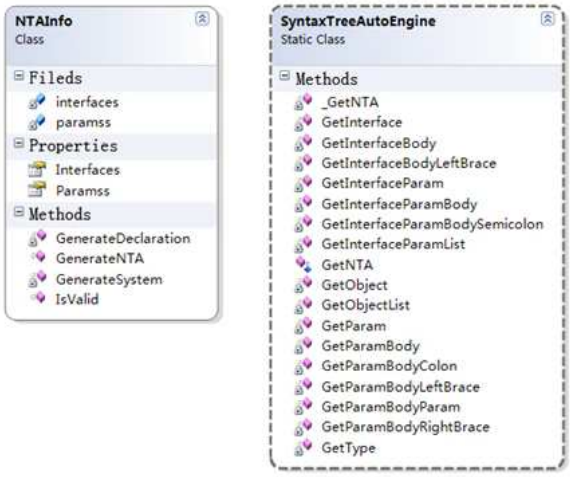


Fig. 3: Class diagram of getting NTA by traversing syntax tree

Table 4: Algorithm A-NTA

01 Insert the zero interface to NTA model.
02 Generate the fission graph for each parameter of every web service interface.
03 Link neighbor parameters for each web service interface.
04 Insert a starting node for each tree graph representing a web service interface and set the launch condition.
05 Insert an ending node for each tree graph representing a web service interface and set the update event.
06 Set corresponding integer clock variables for each parameter as branch signals in global declaration.
07 Create the global clock as the container to store total cost in global declaration.
08 Create an instance for each TAW model in system configuration, and the instance is set to start with the system.

4.1 Test case design

We propose a solution for test cases based on template, which meets the validity and integrity of comprehensive cases running through the model. Different instances for all kinds of TAC models can be generated by adjusting the template parameters, and it is also convenient to implement automatic testing.

The parameters of test cases template are listed according to the character of TAC model.

(1) The count of TAW

The count of TAW models dominates the scale of test cases for TAC model, and it is represented by '*N*'.

(2) The count of nodes in TAW

The count of nodes can reflect the number of parameters in a TAW model, which sets the scale of test cases indirectly. The count of nodes in a TAW is represented by '*L*' and the count of parameters in the

TAW is 'x'. Their relation can be denoted by which

$$L = \sum_{i=1}^{x} Sem(i) + x + 2 \tag{1}$$

The '$Sem(i)$' means the number of semantics partition to corresponding parameter i.

(3) Association strength of TAC

It is not feasible to set all connection relations for each TAC model one by one for the topology structure of WSC. On one hand, the automation test will be difficult to design and implement, and on the other hand, the key point of feasibility and effectiveness testing is the overall complexity and association strength of TAC. So, we propose the association intensity index (*Ave*) to represent this norm. For each *Ave* value, there may be different corresponding topology graphs. Set 'P' as the count of parameters in a TAC model, $CP(i)$ as the number of occurrences of parameter 'i' in all TAW models of TAC, and $TA(i)$ as the count of TAW model where there is at least one parameter after parameter 'i' is removed. Then there is

$$Ave = \sum_{i=1}^{P} C_{CP(i)}^{2} \Big/ \sum_{i=1}^{\sum_{i=1}^{P} CP(i)} TA(i) \tag{2}$$

The '*Ave*' ranges from 0 to 1. It means the association strength is weaker when *Ave* is closer to 0 and the association strength is stronger when it is closer to 1.

(4) Association diversity of TAC

The diversity of parameters' number in TAC and the diversity of association strengths of different TAW models make various test cases. In order to supplement the deficiency of association strength, we set 'E' as the association diversity factor of different TAW models, in which $P(i)$ is set as the count of the ith parameters in the TAW model and Pa as the average value of parameters' number in all TAW models. Then there is

$$E = \frac{\sum_{i=1}^{P} (P(i) - Pa)^2}{P * Pa^2} \tag{3}$$

The 'E' ranges from 0 to 1. It means the association diversity is little when E is closer to 0 and the association diversity is large when E is closer to 1.

In conclusion, the scale of test case is regulated by the count of TAW models, the complexity is set by the number of parameters in TAW, the association strength and diversity adjust the distribution of topology structure.

4.2 The result of tests

The WSC model based on timed automata is capable of describing composite web service with inner logics and workflow. However, the state-space explosion problem restricts its feasibility with the increasing of clocks' number [15]. Zone automata is a choice. The test result is shown as follows.

(1) The simulator of zone automata

In the test, the parameters of $N = 1, L = 2$, $Ave = 0.5, E = 0.25$ were set to shield unrelated parameters. The results are shown in Fig.4. The state-space still grows exponentially in zone automata, but it becomes much slower than region automata.

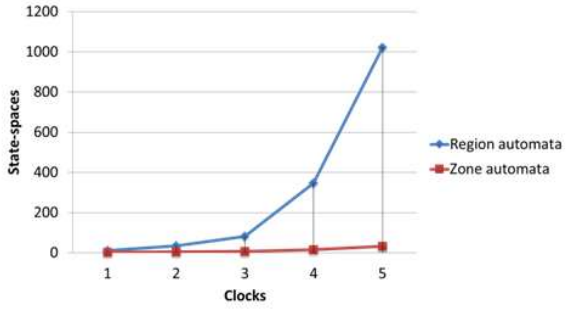


Fig. 4: The comparision of zone automata and region automata

The results show that it divides less state-space with zone automata than region automata and they possess the same ability of describing TAC model. The results illustrate that time zone performs better than regional division of equivalence and relieves the state-space combinatorial explosion problem. So the simulator of zone automata was adopted.

(2) Integer clock

The parameters of $N = 1, L = 2, Ave = 0.5, E = 0.25$ were also set to detect the performance of different clocks. The calculation time is shown in Fig.5.

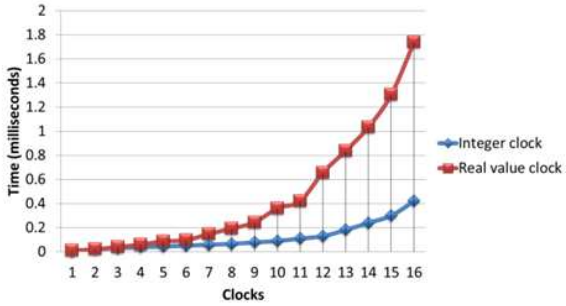


Fig. 5: The comparision of integer clock and real value clock

It is not possible to compare the advantage of integer clocks with real value clocks in isolation. This result shows an average value of the two clocks.

It shows that the running time of integer clocks is faster than real value clocks and it improves the effectiveness of WSC based on TAC model. Though it is not evident when the count of clocks is limited, it takes much less time to finish the simulation with integer clocks than real value ones with the clocks' number increasing. The test result shows an intuitive advantage of computing.

(3) Scale of TAW models

The parameters' values were set by $L \geq 2, Ave = 0.5$, $E = 0.25$. The result in Fig.6 shows the tendency of time as the number of TAW increasing.

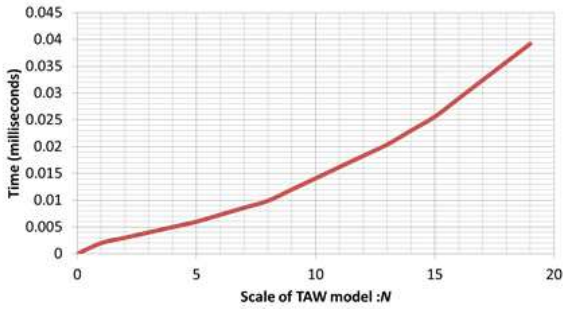


Fig. 6: Time tendency of N

As long as the number of TAW grows, the time complexity of WSC is basically a linear growth with slightly accelerated trend. It indicates the composition is controlled in the linear time and the count of TAW models does not increase the time complexity of TAC.

(4) Scale of nodes in TAW models

In the test, the values of parameters were set as $N \geq 2, Ave = 0.5, E = 0.25$. The running time of TAC with different numbers of nodes in TAW models are shown in Fig.7.

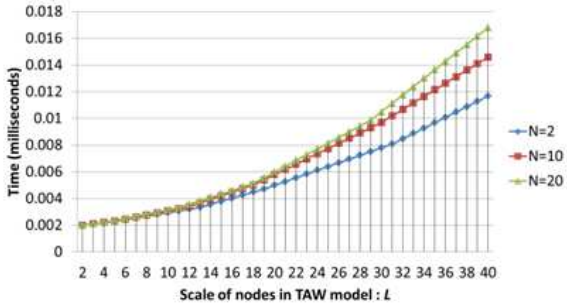


Fig. 7: Time tendency of L

There is a slow growth as the count of nodes in TAW models increases. The difference between this test and the previous one is that only the total count of nodes in TAW models changes while the count of TAW models may not changes. The composition time presents the characteristic of a linear approximation, which indicates that the huge amounts of nodes of TAW models have little influence with the feasibility of TAC.

(5) Association strength and diversity of TAC model

The association strength and diversity of TAC are related to each other. The test results for the two parameters in various combinations of the TAC are shown in Fig.8.

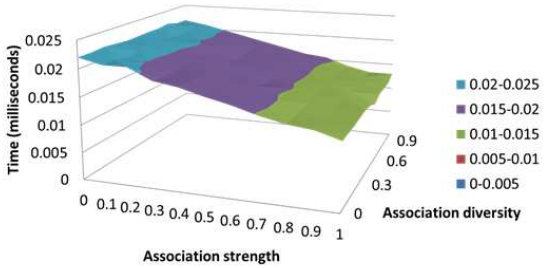


Fig. 8: Time tendency of association strength and diversity

It will takes less time for the running of with the association strength TAC enhancing while slight time fluctuations exists when the association diversity changes. That is to say, the stronger the association strength, the higher the efficiency of the implementation TAC, and the association diversity has little influences with the effectiveness of TAC.

In short, the above test case results show that it spends less state-space with zone automata than region automata for the same TAC model. The TAC with integer clock runs faster than that with real value clock for a large number of clocks. The Computational complexity of TAC does not reach $O(N^2)$ with the increasing of parameters' value, which verify its effectiveness.

5 Conclusions

In this paper, we presented an automated web service composition method based on timed automata in which the composition model and implantation algorithm were provided. The innovation is mainly reflected in three aspects.

Firstly, we proposed the TAC model with the computing framework of timed automata, which is a kind of construction method for WSC. It provides the algorithm of building TAW and TAC, implementing the whole process from independent web services to composed service automatically.

Secondly, 'zero service' and 'integer clock' were introduced into the framework model. The former makes the model easier to convert to a unified form which is convenient to be generated automatically, and the latter reduces the complexity of computing for WSC.

Finally, we implemented the web service composition automation engine, which can receive web service interface description language (i.e., WSIL) and automatically generate TAC model for performing composite service. The engine has strong expansibility and can be applied to different fields.

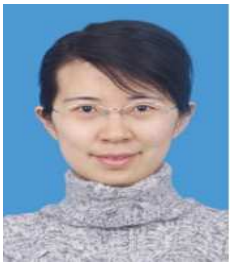
The performance evaluation indicates that it spends less state-space with zone automata than region automata while they share the same ability of describing TAC model; the computing time of integer clocks is faster than real value clocks and it improves the efficiency of WSC based on TAC model; The complexity of TAC grows between the linear and quadratic with the parameters' variety, which verifies its feasibility and effectiveness.

Acknowledgement

This work has been supported by the National Science Foundation of China (Grant No. 61101214, 61371195), the Key Project of National Defense Basic Research Program of China (Grant No. B1120132031) and the Fundamental Research Funds for the Central Universities (Grant No. 20120842003, 20110842001).

References

[1] Rao J., Su X.: A Survey of Automated Web Service Composition Methods: Semantic Web Services and Web Process Composition. Springer Berlin Heidelberg, 43-54 (2005).
[2] Bichier M., Lin K. J.: Service-oriented Computing. Computer, **39**, 99-101 2006.
[3] Wang L. F., Gao W. Q.: A Study of BPEL-Based Framework for Optimal Composition of Dynamic Web Services: Advanced Materials Research, **457**, 856-860 (2012).
[4] Fares E., Bodeveix J. P., Filali M.: Verification of timed BPEL 2.0 models. Enterprise, Business-Process and Information Systems Modeling: Springer Berlin Heidelberg, 261-275 (2011).
[5] Li l.Q., Yan C.: QoS Ontology based Efficient Web Services Selection: International Conference on Management Science and Engineering, 2009. ICMSE 2009, IEEE, 45-50 (2009).
[6] Dong H., Hussain F. K., Chang E.: Semantic Web Service Matchmakers: State of the Art and Challenges: Concurrency and Computation: Practice and Experience, (2012).
[7] Sangiorgi D.: Concurrency Theory: Timed Automata, Testing, Program Synthesis: Distributed Computing, **25**, 3-4 (2012).
[8] AbouTrab M. S., Brockway M., Counsell S., et al: Testing Real-Time Embedded Systems using Timed Automata based Approaches: Journal of Systems and Software, **86**, 1209-1223 (2013).
[9] Emilia Cambronero M., Dłaz G., Valero V., et al: Validation and Verification of Web Services Choreographies by Using Timed Automata: Journal of Logic and Algebraic Programming, **80**, 25-49 (2011).
[10] Dumez C., Bakhouya M., Gaber J., et al. Model-driven Approach Supporting Formal Verification for Web Service Composition Protocols: Journal of Network and Computer Applications, **36**, 1102-1115 (2013).
[11] Bengtsson J., Yi W.: Timed automata: Semantics, Algorithms and Tools: Lectures on Concurrency and Petri Nets. Springer Berlin Heidelberg, 87-124 (2004).
[12] Alur R., Dill D. L.: A Theory of Timed Automata: Theoretical Computer Science, **126**, 183-235 (1994).
[13] Gmez R.: Model-checking Timed Automata with Deadlines with Uppaal: Formal Aspects of Computing, **25**, 289-318 (2013).
[14] Brown D., Levine J., Mason T.: Lex & yacc: O'Reilly, (2012).
[15] Srivathsan B.: Better Abstractions for Timed Automata: IEEE Symposium on Logic in Computer Science, 375-384 (2012).



Hu jingjing received the PhD degree in Computer science from Beijing Institute of Technology, Beijing, China. She is currently a lecturer in the school of Software of Beijing Institute of Technology. Her research interests are in the areas of service computing, multi-agent systems, and GPU-based computer tomography.

Zhu wei is a postgraduate in the school of Software, Beijing Institute of Technology, China. His research interests include artificial intelligence, services computing, software engineering, etc.

Zhao xing received the Ph.D degree in Computer science from University of Science & Technology of China, HeFei, China. He is currently an associate professor in the school of Mathematical sciences of Capital Normal University. His research interests are in the areas of computer tomography, service computing.

Zhu dongfeng is a Ph.D candidate in the school of Computer, Beijing Institute of Technology, China. He received his B.S. degree in computer science from Beijing Jiaotong University. His research interests include p2p computing, services computing, delay tolerant networks, etc.