UDC 004.94

V.V. Shkarupylo, postgraduate,
R.K. Kudermetov, candidate of technical sciences, docent.
Zaporizhzhya National Technical University, Zaporizhzhya, Ukraine
vadshkar@yandex.ru, krk@zntu.edu.ua

An Approach to Composite Web Services Formal Verification

The Composite Web Services behaviors specification technique based on TLA-formalism has been proposed. An approach to Composite Web Services formal TLA-specifications Verification based on TLA Toolbox 1.4 TLC Model Checker component usage has been provided.

Keywords: Composite Web Service, Behavior, TLA, Formal Specification, Verification, Model Checking

Introduction

There are different formal methods which can be used to increase the integrity of systems being developed (disclosing the soundness of its functional properties) [1]. There are two groups of such methods: the methods of Specification and the methods of Verification [2].

We've chosen the Model Checking method in order to resolve Verification procedure. The grounding of this step is as follows: unlike both alternatives (Deductive Verification and Equality Checking) Model Checking method implementation can be fully automated [3].

The core of Model Checking method can be disclosed in following definition [4]: "Model Checking method is the method for checking, whether the given temporal formula is feasible for specified model".

The specificity of checking procedure can be interpreted as the exhaustive search of model's finite states, coupled with deadlocks checking.

In given article the implementation of Model Checking method is examined in context of Composite Web Service's behaviors formal specification. These behaviors are based on orchestration model (model of Composite Web Services' components – atomic web services – centralized coordination) [5]. It is necessary to chose the appropriate formalism to correctly specify such behaviors. For this purpose we've chosen the TLA-based formalism (TLA, Temporal Logic of Actions) proposed by L. Lamport [6]. Despite the fact that the expressive power of this solution is comparable enough to possible alternatives (LTL, CTL, LOTOS [7] etc.), TLA-specification can be verified in automated way by using TLA-based Model Checking method (TLC utility as TLA Toolbox 1.4 component [8]).

Problem Statement

Basing on orchestration model, we consider composite web service as the system, represented by two-component tuple $CWS = \langle \{CRD\}, AWS \rangle$, where CRD is a coordinator of composition process, AWS is the set of atomic web services included in composition. $AWS = \{aws_i \mid i = \overline{1,m}\}$, $m \in N$, where $aws_i \in AWS$ – atomic web service.

We interpret CWS-behaviors as the sequences of admissible states [1] which determine the appropriate functional properties of CWS. We also bring the set of scenarios $SC = \{sc_j^{CRD} \mid j = \overline{1,n}\}$, $n \in N$ to specify the behaviors; $sc_j^{CRD} \in SC$ is a scenario for j – behavior specification.

We represent the scenarios as tuples. The components of these tuples are the events [9] leading to states changing.

The goal of the investigation:

- to implement the Verification procedure of given CWS TLA-specification.

The tasks to be solved:

- to develop the technique to CWS behaviors formal specification (the technique to CWS formal model creation);
- to develop the approach to CWS formal specification verification procedure implementation by using the Model Checking method.

Let CWS be represented by Kripke Structure (Model) $M = \langle S, \{s_0\}, R, L \rangle$, where S is the finite set of states; $s_0 \in S$ is the initial state;

$R \subseteq S \times S$ – set of transitions; $L : S \to 2^{AP}$ – states marking function; AP – set of atomic propositions [1].

Using TLA-formalism [6] and referring to CTL verification problem statement [10], we formulate TLA-verification task as follows: given the model M of CWS and some behavior σ. We have to find such formula f, that $M, \sigma \models f$, where '$\models$' is a truth relation.

CWS Behaviors Specification

Let we have the three-component tuple $\langle S, V, D \rangle$, where V is the set of CWS variables, $D = \{0,1\}$ – set of values ('1' – event has been occurred; otherwise – '0').

As CWS states changes are prompted by corresponding events, we use the events classification [11]: $REQ = \{req, resp\}$ – set of boundary events; $INVOKE = \{invoke_i\}$ – set of $aws_i \in AWS$ invocations events (by CRD); $RES = \{res_i\}$ – set of events of getting invocations results. Basing on given classification, we represent V as follows: $V = \{REQ, INVOKE, RES\}$. The framework, used for SC elements specification is as follows: $\langle req, ..., resp \rangle$.

Consider the example.

Let $AWS = \{aws_1, aws_2\}$, where $aws_1, aws_2 \in AWS$ provide the input dataset searching and sorting procedures respectively. Assume that depending on our goal the sequence of aws_1 and aws_2 invocations is changed (by CRD). We denote these procedures as $search$ and $sort$ functions respectively. Assume that coordination procedure is based on sc_1^{CRD} scenario. Then these functions can be mapped as follows: $search : X \to Y$; $sort : Y \to Y$; $Y \subseteq X$, where X is the given set of elements, Y – set of elements that satisfy the search conditions. Then sc_1^{CRD} can be represented as the composition (superposition) of $search$ and $sort$ functions: $sort(search(x)), x \in X$ or $sort \circ search : X \to Y$, where '$\circ$' – composition operator. For sc_2^{CRD} scenario: $sort : X \to X$; $search : X \to Y$; $search \circ sort : X \to Y$ or $search(sort(x))$.

As far as composition operation is not commutative, the inequality $sort(search(x)) \neq search(sort(x))$ provides the grounding to speak about two possible CWS behaviors. We represent these behaviors in formula

$$req \Rightarrow \left(sc_1^{CRD} \vee sc_2^{CRD}\right) \Rightarrow resp, \quad (1)$$

where '$\Rightarrow$' – implication operator, '$\vee$' – logical "OR".

We describe both scenarios as follows:

$$sc_1^{CRD} = \langle invoke_1, res_1, invoke_2, res_2 \rangle,$$

$$sc_2^{CRD} = \langle invoke_2, res_2, invoke_1, res_1 \rangle.$$

The sequence of scenarios' elements defines the sequence of corresponding events occurrences. We represent these scenarios with UML Sequence Diagram (fig. 1): "Client" actor is injected in order to visualize the character of external environment impact.

We shift from events to states. For this purpose we propose the states coding format (F):

$$F = \langle \{req\}, INVOKE, RES, \{resp\} \rangle, \quad (2)$$

where $\{req\} \cup \{resp\} = REQ$. F-components are the variables, associated with corresponding events. We encode the states with values from D by format (2) using and interpret the code (6-digit in our case) in inverted manner (by mirroring). This results in directly-proportional dependence between the relative order of states and the appropriate codes.

To describe the states we use Nominal Naming (NN), represented in table 1.

By using table 1 we can represent Kripke Model as follows: $M = \langle S, \{s0\}, R, L \rangle$, where

$S = \{s0, ..., s9\}$, $R = R_1 \cup R_2 \cup R_3 \cup R_4$,

$R_1 = \{(s0, s1)\}$,

$R_2 = \{(s1, s2), (s2, s3), (s3, s4), (s4, s5)\}$,

$R_3 = \{(s1, s6), (s6, s7), (s7, s8), (s8, s5)\}$,

$R_4 = \{(s5, s9)\}$,

$L(s0) = \{req = 0, ..., resp = 0\}$,

$L(s1) = \{req = 1, invoke_1 = 0, ..., resp = 0\}$,

…,

$L(s9) = \{req = 1, ..., resp = 1\}$.

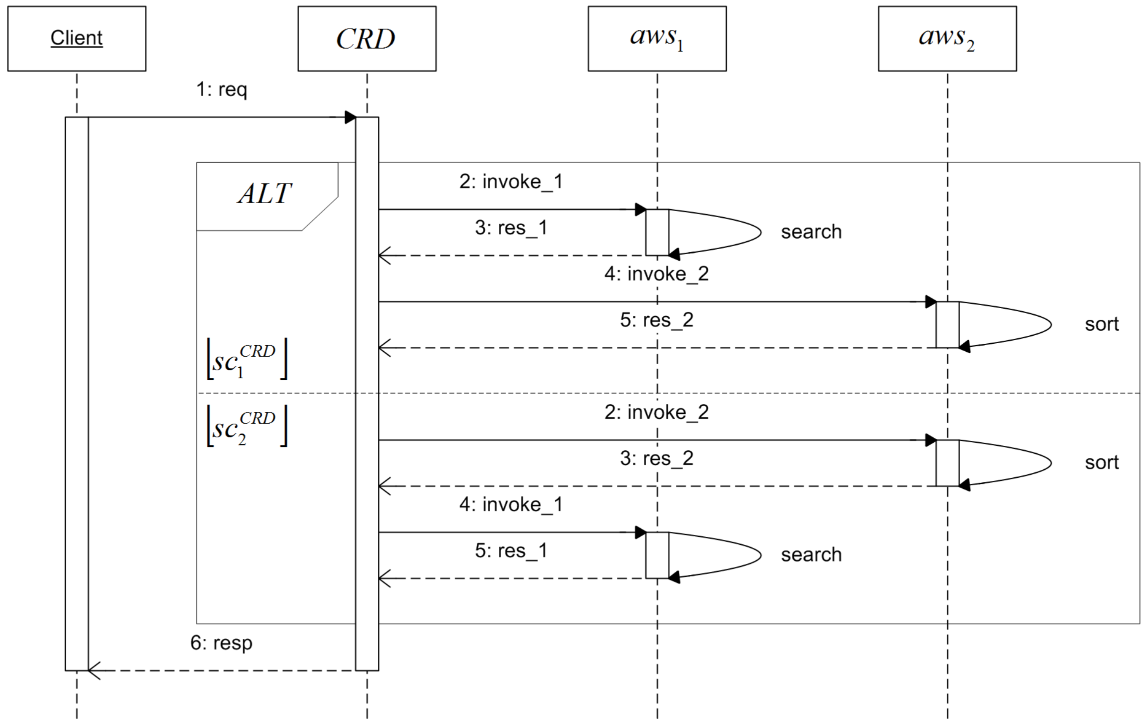

Figure 1 – UML Sequence Diagram

Table 1 – CWS states coding format

SC	Coding format (F)						NN	Event (fig. 1)	TLA action
	$resp$	res_2	res_1	$invoke_2$	$invoke_1$	req			
-	0	0	0	0	0	0	s0	–	Init
-	0	0	0	0	0	1	s1	1: req	OnReq
sc_1^{CRD}	0	0	0	0	1	1	s2	2: invoke_1	S_CRD_1
	0	0	1	0	1	1	s3	3: res_1	
	0	0	1	1	1	1	s4	4: invoke_2	
	0	1	1	1	1	1	s5	5: res_2	
sc_2^{CRD}	0	0	0	1	0	1	s6	2: invoke_2	S_CRD_2
	0	1	0	1	0	1	s7	3: res_2	
	0	1	0	1	1	1	s8	4: invoke_1	
	0	1	1	1	1	1	s5	5: res_1	
-	1	1	1	1	1	1	s9	6: resp	OnResp

"Init",…,"OnResp" (table 1, "TLA action" column) names are used to designate the TLA-based logical formulas (actions or "TLA actions" [6]).

We propose to generate CWS formal TLA-specification (listing 1) by interpreting states codes given in table 1.

Listing 1 – TLA-based CWS formal specification

```
EXTENDS Naturals
\* define variables,
\* corresponding to events
VARIABLES   req, resp,
            invoke_1, invoke_2,
            res_1, res_2
\* define the acceptable values
Invariant ==   /\ req \in {0,1}
               /\ invoke_1 \in {0,1}
               /\ invoke_2 \in {0,1}
               /\ res_1 \in {0,1}
               /\ res_2 \in {0,1}
               /\ resp \in {0,1}
\* specify st0
Init ==    /\ req = 0 /\ resp = 0
           /\ invoke_1 = 0
           /\ invoke_2 = 0
           /\ res_1 = 0 /\ res_2 = 0
\* -//- st1
OnReq == req' = 1 - req
\* -//- S_CRD_1 (st2 - st4, st5)
S_CRD_1 == /\ invoke_1' =
     IF req = 1 THEN 1 ELSE 0
           /\ res_1' =
     IF invoke_1 = 1 THEN 1 ELSE 0
```

```
          /\ invoke_2' =
     IF res_1 = 1 THEN 1 ELSE 0
          /\ res_2' =
     IF invoke_2 = 1 THEN 1 ELSE 0
\* -//- S_CRD_2 (st6 - st8, st5)
S_CRD_2 == /\ invoke_2' =
     IF req = 1 THEN 1 ELSE 0
          /\ res_2' =
     IF invoke_2 = 1 THEN 1 ELSE 0
          /\ invoke_1' =
     IF res_2 = 1 THEN 1 ELSE 0
          /\ res_1' =
     IF invoke_1 = 1 THEN 1 ELSE 0
\* -//- st9
OnResp == resp' = IF res_1 = 1
     /\ res_2 = 1 THEN 1 ELSE 0
\* TLA-specification,
\* corresponding to (2)
Next == OnReq /\
     (S_CRD_1 \/ S_CRD_2) /\ OnResp
```

The logical operators "AND" ($\wedge$) and "OR" ($\vee$) are represented in TLA-formalism (listing 1) with symbols '/\' and '\/' respectively. Names `req'`,..., `resp'` are used for the next-state variables initialization. The scenarios sc_1^{CRD} and sc_2^{CRD} are specified with "`IF-THEN-ELSE`" construct.

TLA-specification Verification

We model CWS behaviors specified in listing 1 with TLC Model Checker utility (as TLA Toolbox 1.4 component). We represent the modeling process with UML Statechart Diagram (fig. 2).

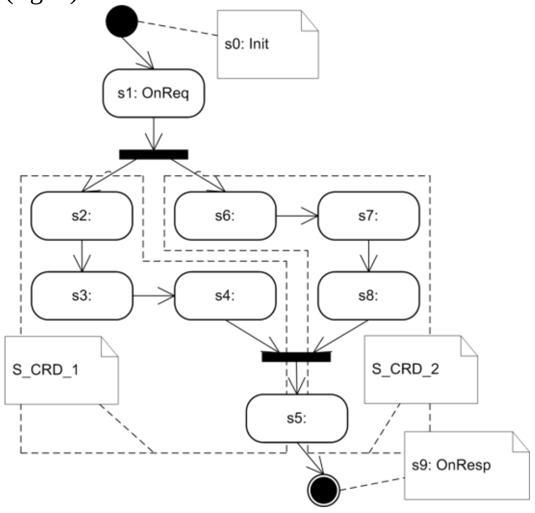

Figure 2 – CWS Statechart Diagram

It is clear from fig. 2 that $|S| = 10$ and there are no deadlocks. If we distinguish two subsets $S_1 \subset S$ and $S_2 \subset S$, matching the sc_1^{CRD} and sc_2^{CRD} scenarios respectively, we will see that $|S_1| = |S_2| = 7$.

Using (1), listing 1 and table 1 content, we represent the TLA-verification task solution in tabular form (table 2).

Table 2 – CWS behaviors

Scenarios ($sc_1^{CRD}, sc_2^{CRD} \in SC$)	Behaviors (σ_1, σ_2)
$sc_1^{CRD} = \left\langle \begin{matrix} invoke_1, res_1, \\ invoke_2, res_2 \end{matrix} \right\rangle$	$M, \sigma_1 \models$ OnReq $\Rightarrow$ S_CRD_1 $\Rightarrow$ OnResp
$sc_2^{CRD} = \left\langle \begin{matrix} invoke_2, res_2, \\ invoke_1, res_1 \end{matrix} \right\rangle$	$M, \sigma_2 \models$ OnReq $\Rightarrow$ S_CRD_2 $\Rightarrow$ OnResp

Conclusion

Thus, the proposed approach can be used as the facility for Composite Web Services specifications formal verification procedure implementation. It is applicable owing to the following two aspects: TLA-formalism expressive power is sufficient enough; TLC Model Checker utility existence makes it possible to verify CWS TLA specifications in automated way.

Scientific Novelty: the TLA-based technique to Composite Web Services formal specification has been proposed.

Practical Significance: the approach to Composite Web Services TLA-specifications verification procedure automation has been provided.

It is also planned to expand our approach on CWS Validation task solving.

References

1. Кларк Э. М. Верификация моделей программ: Model Checking: пер. с англ. / Э. М. Кларк, О. Грамберг, Д. Пелед. – М.: МЦНМО, 2002. – 416 с.

2. Bai X. Towards Formal Verification of Web Service Composition / X. Bai, Y. Fan // Proceedings of the 11th International Conference on Industrial Engineering and Engineering Management (Shenyang, China, April, 2005). – P. 577 – 581. – ISBN 7-111-03973-4.

3. Тарасюк О.М. Формальные методы разработки критического программного обеспечения. Лекционный материал / О.М. Тарасюк, А.В. Горбенко; под ред. Харченко В.С. – МОН Украины, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», 2009. – 214 с. – ISBN 978-966-96770-8-2.

4. Карпов Ю. Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем / Ю. Г. Карпов. – СПб.: БХВ-Петербург, 2010. – 560 с. – ISBN 978-5-9775-0404-1.

5. Papazoglou M. P. Service-Oriented Computing: State of the Art and Research Challenges / M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann // IEEE Computer. – 2007. – Vol. 40, No. 11. – P. 64 – 71.

6. Lamport L. Specifying Systems / L. Lamport. – Boston.: Addison-Wesley, 2002. – 364 p. – ISBN 0-321-14306-X.

7. Dumez C. Formal Specification and Verification of Service Composition using LOTOS / C. Dumez, M. Bakhouya, J. Gaber, M. Wack // Proceedings of the 7th ACM International Conference on Pervasive Services (Berlin, Germany, July 13 – 16, 2010).

8. TLA Toolbox [Электронный ресурс]. – Режим доступа: \www/ URL: http://www.tlaplus.net/tools/tla-toolbox/. – Загл. с экрана.

9. Хоар Ч. Взаимодействующие последовательные процессы: пер. с англ. / Ч. Хоар. – М.: Мир, 1989. – 264 с. – ISBN 5-03-001043-2.

10. Grumberg O. 25 Years of Model Checking: History, Achievements, Perspectives / O. Grumberg, H. Veith. – Berlin.: Springer, 2008. – 231 p. – ISBN-10 3-540-69849-3.

11. Шкарупило В. В. Методика автоматизированного синтеза композитных веб-сервисов / В. В. Шкарупило, Р. К. Кудерметов // Информатика и компьютерные технологии : Сб. трудов VII Международной научно-технической конференции студентов, аспирантов и молодых ученых, 22 – 23 ноября 2011г., г. Донецк : в 2-х т. – Донецк: ДонНТУ, 2011. – Т.1. – С. 382 – 384.

Надійшла до редколегії 10.04.2012

В.В. ШКАРУПИЛО, Р.К. КУДЕРМЕТОВ
Запорізький національний технічний університет

Підхід до формальної верифікації композитних веб-сервісів.

Запропоновано спосіб специфікації динамік композитних веб-сервісів на основі формалізму TLA. Запропоновано підхід до верифікації TLA-специфікацій композитних веб-сервісів на основі використання засобу TLC Model Checker у складі програмного інструментарію TLA Toolbox 1.4.

Ключові слова: композитний веб-сервіс, динаміка, TLA, формальна специфікація, верифікація, Model Checking

В.В. ШКАРУПИЛО, Р.К. КУДЕРМЕТОВ
Запорожский национальный технический университет

Подход к формальной верификации композитных веб-сервисов.

Предложен способ специфицирования динамик композитных веб-сервисов на основе формализма TLA. Предложен подход к верификации TLA-спецификаций композитных веб-сервисов на основе использования средства TLC Model Checker в составе программного инструментария TLA Toolbox 1.4

Ключевые слова: композитный веб-сервис, динамика, TLA, формальная спецификация, верификация, Model Checking