УДК 004.052.42

Шкарупило В. В.

Аспирант, Запорожский национальный технический университет, Украина, E-mail: vadshkar@yandex.ua

МОДЕЛЬ TLA-СПЕЦИФИКАЦИИ КОМПОЗИТНОГО ВЕБ-СЕРВИСА С МНОЖЕСТВОМ ДИНАМИК

Разработана формальная модель специфицирования свойств композитных веб-сервисов на основе формализма темпоральной логики TLA. На отдельном примере выполнена верификация TLA-спецификации композитного веб-сервиса с множеством свойств в автоматизированном режиме с использованием реализации метода Model Checking в составе программного средства TLA Toolbox (TLC, TLA Checker). Проведена оценка сопутствующих временных издержек.

Ключевые слова: модель, композитный веб-сервис, формальная спецификация, TLA, верификация, Model Checking, TLC.

ВВЕДЕНИЕ

На сегодняшний день использование формальных методов можно рассматривать в качестве обоснованного шага, направленного на уменьшение издержек, связанных с валидацией разрабатываемой системы (тестированием в частности) [1, 2]. Этот шаг заключается в создании формальной модели целевой системы и последующей верификации модели. В качестве системы рассмотрим композитный веб-сервис (CWS, Composite Web Service). Обоснование формальной верификации спецификации свойств CWS, вместе с результатами анализа существующих решений, приведено в [3].

Свойства (функциональные характеристики) CWS рассмотрим как результат координирования атомарных веб-сервисов на основе модели оркестровки [4]. Отметим, при этом, что CWS есть система, функционирующая в режиме «ad-hoc», – данное ограничение учтем при построении формальной модели. С этой целью в качестве метода формальной верификации используем метод проверки на модели (Model Checking), который заключается в полном переборе пространства состояний модели исследуемой системы [5] (классификация формальных методов приведена в [6]). Обоснование выбора – процедура формальной верификации может быть выполнена в автоматизированном режиме с использованием специализированных программных средства. В качестве такого средства возьмем свободно распространяемую среду разработки «TLA Toolbox» [7].

Концепция «behavior» формализма TLA (Temporal Logic of Actions) [8], предложенного Л. Лампортом (Leslie Lamport), является приемлемой абстракцией для специфицирования отдельно взятого свойства CWS: последнее может быть описано с помощью единственной TLA-формулы. Подобное положение дел, как следствие, упрощает решение задач, связанных с конфигурированием свойств CWS в соответствии с налагаемыми «ad-hoc»-ограничениями.

Для реализации процедуры формальной верификации необходимы следующие три предпосылки [9]: модель системы (множество состояний), метод формальной спецификации требований к системе (свойств системы), множество правил доказательства того, что система удовлетворяет выдвинутым требованиям. Адаптируем эти предпосылки применительно к задаче верификации формальной TLA-спецификации свойств CWS: каждому свойству поставим в соответствие структуру Крипке [10]; с целью верификации используем TLA-реализацию метода проверки на модели в составе «TLA Toolbox» – TLC (TLA Checker).

1. ПОСТАНОВКА ЗАДАЧИ

Предварительно выполним постановку задачи формальной TLC-верификации. С этой целью представим CWS в виде трехкомпонентного кортежа: $\langle \mathrm{AWS}, \{\mathrm{crd}\}, \mathrm{FPS} \rangle$, где $\mathrm{AWS} = \left\{ \mathrm{aws}_i \middle| i = \overline{1, m} \right\}_{m \in N}$ – множество атомарных веб-сервисов, crd – компонент-координатор элементов AWS (модель компонента «BPEL Engine» в составе программной системы синтеза композитных веб-сервисов – Apache ODE, IBM WebSphere Process Server,…[11]), а FPS – множество свойств. Каждое свойство есть результат координирования элементов AWS.

Пусть задана модель Крипке

$$M = \langle S, \{s_0\}, R, L \rangle,$$

где S – конечное непустое множество состояний CWS; $s_0 \in S$ – начальное состояние; $R \subseteq S \times S$ – отношение переходов на S: $\forall s \in S \; \exists s' \in S : R(s) = s'$, где $s, s' \in S$ – текущее и последующее состояния, соответственно; $L : S \to 2^{AP}$ – функция разметки состояний; AP – множество атомарных высказываний. С моделью M, при этом, также ассоциируется понятия «путь» и «траектория» как бесконечные последовательности вида $s_9, s_1, \ldots$ и $L(s_0), L(s_1), \ldots$, соответственно [12].

В контексте веб-сервисов «траектории» рассмотрим как конечные последовательности разметок. Последние назовем «динамиками», обуславливающими свойства CWS.

Пусть имеем множество булевых значений $D = \{0, 1 \mid 0 \equiv \text{FALSE}, 1 \equiv \text{TRUE}\}$. Требуется проверить, что $\exists s \in S : (M, s \models \varphi) \equiv \text{TRUE}$, где '$\models$' – оператор выполнимости, а φ – логическая формула на основе синтаксиса TLA. Таким образом, в решаемой задаче формальной TLC-верификации, в отличие от задачи проверки выполнимости логической формулы, модель M является заданной.

Рассуждая о множестве функциональных характеристик CWS, целесообразно рассматривать M с привязкой к отдельно взятому свойству. С этой целью, согласуясь с синтаксисом TLA, перейдем от высказывания $(M, s \models \varphi)$ к высказыванию

$$(\sigma \models \varphi), \tag{1}$$

где σ – динамика:

$$\sigma = \langle L(s_0), L(R(s_0)), L(R(R(s_0))), ..., L(s_{END}) \rangle, \tag{2}$$

где $R(s_0) = s_1$, $R(R(s_0)) = R(s_1) = s_2$ и т. д. s_{END}, при этом, есть конечное состояние: $R(s_{END}) = s_{END}$.

Переход от $(M, s \models \varphi)$ к (1) обусловлен не только спецификой формализма TLA, но и различием содержательных нагрузок: в первом случае проверке подлежит выполнимость формулы φ в состоянии $s \in S$ модели M, а во втором – выполнимость φ в каждом из состояний σ.

Сформируем множество FPS:

$$FPS = \left\{ (\sigma_j, \varphi_j) \mid j = \overline{1, n} \right\}_{n \in N},$$

где σ_j – *j*-я динамика на основе соответствующей модели M_j, причем $M_j = \langle S^{(j)}, \{s_0\}, R^{(j)}, L \rangle$, где $S^{(j)} \subseteq S, R^{(j)} \subseteq R : \bigcup_j S^{(j)} = S, \bigcup_j R^{(j)} = R$.

Основываясь на (2) и на формализме CSP (Communicating Sequential Processes) Ч. Хоара (C. A. R. Hoare) [13], запишем σ_j с использованием оператора конкатенации

$$\forall j : \sigma_j = \langle L(s_0) \rangle \wedge \left\langle L\left(R^{(j)}(s_0)\right), L\left(R^{(j)}\left(R^{(j)}(s_0)\right)\right), ..., L\left(s_{END}^{(j)}\right) \right\rangle,$$

где $s_{END}^{(j)}$ – конечное состояние *j*-й динамики.

Будем говорить, что *j*-е свойство (функциональная характеристика) CWS реализуется согласно σ_j, если результат соответствующего отношения выполнимости принимает истинное значение:

$$\exists (\sigma_j, \varphi_j) \in FPS : (\sigma_j \models \varphi_j) \equiv \text{TRUE}.$$

Непосредственно целевой CWS как формализованное TLA-описание его функциональных характеристик представим следующим образом:

$$\varphi_1 \vee \varphi_2 \vee ... \vee \varphi_n. \tag{3}$$

Также приведем формализацию постановки задачи TLC-верификации:

$$\forall (\sigma_j, \varphi_j) \in FPS : (\sigma_j \models \varphi_j) ? \text{TRUE} : \text{FALSE}. \tag{4}$$

Как следствие, в работе ставятся и решаются такие задачи:

– разработать модель TLA-спецификации функциональных характеристик CWS;

– оценить временные издержки, связанные с процедурой TLC-верификации спецификаций на основе предложенной модели.

2. КОНЦЕПТУАЛИЗАЦИЯ И ФОРМАЛИЗАЦИЯ

Для построения моделей M_j свойств CWS сформируем множество переменных состояний. Для этого воспользуемся спецификацией WS-BPEL 2.0 [14], в которой теги разделены на две группы: «Basic Activities» (BA) и «Structured Activities» (SA). В группах насчитывается 10 и 7 тегов, соответственно. Факт включения всех 17 тегов в состав модели являлся бы первопричиной ряда нежелательных факторов, наиболее очевидные из которых – «эффект комбинаторного взрыва» (пространства состояний), а также громоздкость модели. С учетом этих замечаний, для формирования множества переменных состояний рассмотрим теги выходных *.bpel-файлов тестовой утилиты «ChallengeGenerator.jar» (табл. 1) [15]. Целевое назначение последней – оценка средств автоматизации процесса синтеза CWS с точки зрения различных критериев (затраченное время, избыточность и т. п.).

Сформируем множество переменных состояний $V = \{v_i\}$, где $v_i \in V$ соответствует тегу <invoke name="aws_i"> для элемента $aws_i \in AWS$. Следовательно, $|V| = |AWS| = m$.

С целью синтеза на основе моделей M_j формальной TLA-спецификации, подлежащей TLC-верификации в автоматизированном режиме, формализуем понятие «событие» (в контексте дискретно-событийного моделирования). Данный шаг обусловлен следующей предпосылкой: TLA-специфицирование свойств CWS осуществимо путем формального описания переходов между разме-

Таблица 1. Теги, учтенные в формальной TLA-модели

№	Наименование	Описание	Группа
1	<invoke>	тег вызова $aws_i \in AWS$;	BA
2	<sequence>	тег последовательных вызовов;	SA
3	<flow>	тег параллельных вызовов;	SA
4	<switch>	тег условных переходов;	SA
5	<case>	тег-компонент в составе <switch>	SA

ченными состояниями. Для этого введем множества V′ и AP′. Элементы $v_i \in V$ и $v_i' \in V'$ рассмотрим, соответственно, как переменные текущего и последующего состояний $s, s' \in S$. Сформируем множества AP и AP′:

$$f : V \times D \to AP,$$
$$f' : V' \times (D \setminus \{0\}) \to AP',$$

где $f(v_i, 0) \equiv (v_i = 0) \in AP$, $f(v_i, 1) \equiv (v_i = 1) \in AP$, а $f'(v_i', 1) \equiv (v_i' = 1) \in AP'$. Следовательно, разметка начального состояния $L(s_0) \subset AP : f : V \times (D \setminus \{1\}) \to L(s_0)$.

В TLA-спецификации $L(s_0)$ зададим высказыванием «Init»:

$$Init \equiv (v_1 = 0) \wedge (v_2 = 0) \wedge ... \wedge (v_m = 0). \quad (5)$$

Для задания разметок, отличных от $L(s_0)$, сформируем высказывания на основе элементов AP: «Label» – высказывание для задания разметки $L(s) : s \in S \setminus \{s_0\}$.

С целью формализации *i*-го события, сформируем высказывание на основе элементов множества $L(s_0) \cup AP'$ (импликация):

$$Event_i \equiv (\neg(v_i = 0) \vee (v_i' = 1)). \quad (6)$$

Шаблон TLA-формализации перехода $(s, s') \in R$ отождествим с высказыванием «Next» (табл. 2). Последнее сформируем с использованием конструкции «IF-THEN-ELSE» путем агрегирования события (6).

Таким образом, если высказывание «Next» на основе некоторого события «$Event_i$» истинно, получим разметку $L(s') : L(s') \Delta L(s) = \{(v_i = 0), (v_i = 1)\}$.

Для ответа на вопрос (4) используем подход [16] (рис. 1).

Будем утверждать, что формальная TLA-модель адекватна исследуемой системе, если матрица смежности MX графа (S, R) равна таковой (MX′) для графа, полученного из листинга TLC-верификации (рис. 1).

Введем следующие множества:

– множество символьных TLA-обозначений (CO) $A = \{'/\backslash', '\backslash/', '\sim', '[\,]', '\backslash in'\}$, где элементы $'/\backslash', '\backslash/', '\sim' \in A$ обозначают булевы операторы «И», «ИЛИ», «НЕ», соответственно, $'[\,]' \in A$ есть CO темпорального оператора «Box» (G, «Globally»), а элемент $'\backslash in' \in A$ несет контекстную нагрузку символа '∈' теории множеств;

– множество $Tags = \{tg_1, tg_2, tg_3, tg_4\}$, элементы которого рассмотрим как строки *.bpel-кода на основе тегов групп BA и SA (табл. 3);

Таблица 2. Переходы на основе событий

Высказывание	Предусловие, «IF» («ELSE IF»)	Спецификация перехода
$Next \equiv 1$	$Init \equiv 1$	$Init \wedge Event_i$
	$Label \wedge (v_i = 0) \equiv 1$	$Label \wedge Event_i$

– множество условий $Cnd = \{cnd_1, cnd_2\}$: $cnd_1 \equiv ((tg_1 \prec tg_2) \vee (tg_3 \prec tg_4))$, $cnd_2 \equiv (tg_2 \prec tg_4)$, где '$\prec$' – оператор предшествования;

– множество правил трансляции (*.bpel-to-*.tla) $Rules = \{r_1, r_2, r_3\}$, элементы которого сформируем на основе элементов множеств Tags и Cnd. Элементы Rules, при этом, рассмотрим как «вложенные» относительно тегов группы SA (табл. 4).

Под правилами будем подразумевать функции, значения которых представим с использованием конструкции «IF-THEN-ELSE»: $E(r_1), E(r_2), E(r_3)$ есть множества строк TLA-кода. Строка подлежит записи в целевую TLA-спецификацию в случае выполнимости условия (cnd_1 или cnd_2), переданного в функцию в качестве аргумента.

Таким образом, в результате трансляции WS-BPEL-описания посредством предложенных правил (табл. 4) получим формальную TLA-спецификацию, подлежащую TLC-верификации (рис. 1).

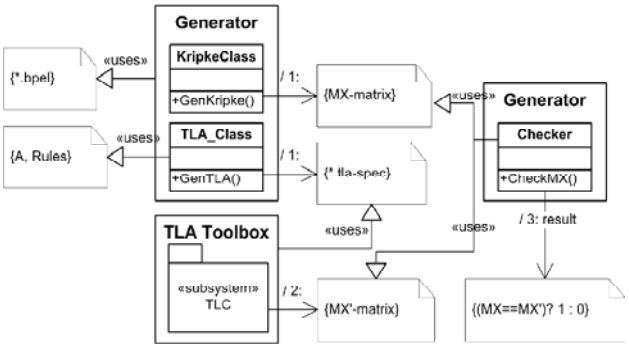

Рис. 1. Диаграмма компонентов

Таблица 3. Описание элементов множества Tags

Элементы	<case>- и <invoke>-конструкции вызова $aws_i, aws_k \in AWS$, где $k = \overline{1, m}, k \neq i$
$tg_1\,(tg_3) \equiv$	`"<bpel: case name="service: \ aws_i" ("aws_k")>"`
$tg_2\,(tg_4) \equiv$	`"<bpel: invoke name="aws_i" \ ("aws_k")...>"`

Таблица 4. Правила трансляции

Элементы группы SA	Элементы множества Rules, причем $\{(v_i = 0), (v_k = 0)\} \subseteq L(s_0)$
<switch>	$r_1(cnd_1, tg_1, tg_2, tg_3, tg_4) \equiv$ $(v_i' = IF\,(\sim v_i \wedge \sim v_k)\,THEN \sim v_i\ ELSE\ v_i) \vee$ $(v_k' = IF\,(\sim v_i \wedge \sim v_k)\,THEN \sim v_k\ ELSE\ v_k)$;
<sequence>	$r_2(cnd_2, tg_2, tg_4) \equiv$ $(v_k' = IF\,(v_i \wedge \sim v_k)\,THEN \sim v_k\ ELSE\ v_k)$;
<flow>	$r_3(cnd_2, tg_2, tg_4) \equiv r_2(cnd_2, tg_2, tg_4) \wedge$ $(v_i' = IF\,(\sim v_i \wedge v_k)\,THEN \sim v_i\ ELSE\ v_i)$

Рассмотрим пример. Пусть мы располагаем текстовым WS-BPEL-описанием целевого CWS, для которого $\mathrm{AWS} = \{\mathrm{aws}_1, \mathrm{aws}_2\}$:

```
<bpel: process...>...<bpel: switch...>
  <bpel: case name="1st_behavior">
   <bpel: sequence...>
... <bpel: invoke name="aws_1".../>
<bpel: invoke name="aws_2".../>...
   </bpel: sequence...></bpel: case>
  <bpel: case name="2nd_behavior">
   <bpel: sequence...>
... <bpel: invoke name="aws_2".../>
<bpel: invoke name="aws_1".../>...
   </bpel: sequence...></bpel: case>
</bpel: switch>...</bpel: process>
```

Следовательно, $\mathrm{V} = \{\mathrm{v}_1, \mathrm{v}_2\}$, а алгоритм функционирования crd задается последовательностью событий Event_1 и Event_2: $\mathrm{Event}_1 \equiv (\neg(\mathrm{v}_1 = 0) \vee (\mathrm{v}_1' = 1))$, а $\mathrm{Event}_2 \equiv (\neg(\mathrm{v}_2 = 0) \vee (\mathrm{v}_2' = 1))$.

Пусть $\mathrm{L}(\mathrm{s}_0)$ является заданной (5): $\mathrm{Init} \equiv (\mathrm{v}_1 = 0) \wedge (\mathrm{v}_2 = 0)$. Тогда, при условии $(\mathrm{Init} \wedge ((\mathrm{Event}_1 \prec \mathrm{Event}_2) \vee (\mathrm{Event}_2 \prec \mathrm{Event}_1)) \equiv \mathrm{TRUE})$, допустимы динамики σ_1 и σ_2: $\sigma_1 \vee \sigma_2 = \langle \mathrm{L}(\mathrm{s}_0) \rangle^{\wedge} (\langle \mathrm{L}(\mathrm{s}_1), \mathrm{L}(\mathrm{s}_3) \rangle \vee \langle \mathrm{L}(\mathrm{s}_2), \mathrm{L}(\mathrm{s}_3) \rangle)$, которым соответствуют модели M_1 и M_2 (табл. 5).

Наша задача – получить на основе модели M (табл. 5) формальную TLA-модель, подлежащую TLC-верификации (табл. 6).

Соответствующая граф-модель системы переходов приведена на рис. 2.

Таблица 5. Модели Крипке для σ_1 и σ_2

$\mathrm{M}_1 = \langle \mathrm{S}^{(1)}, \{\mathrm{s}_0\}, \mathrm{R}^{(1)}, \mathrm{L} \rangle$:	$\mathrm{M}_2 = \langle \mathrm{S}^{(2)}, \{\mathrm{s}_0\}, \mathrm{R}^{(2)}, \mathrm{L} \rangle$:
$\mathrm{S}^{(1)} = \{\mathrm{s}_0, \mathrm{s}_1, \mathrm{s}_3\}$;	$\mathrm{S}^{(2)} = \{\mathrm{s}_0, \mathrm{s}_2, \mathrm{s}_3\}$;
$\mathrm{R}^{(1)} = \{(\mathrm{s}_0, \mathrm{s}_1), (\mathrm{s}_1, \mathrm{s}_3)\}$;	$\mathrm{R}^{(2)} = \{(\mathrm{s}_0, \mathrm{s}_2), (\mathrm{s}_2, \mathrm{s}_3)\}$;
$\mathrm{L}(\mathrm{s}_0) = \{(\mathrm{v}_1 = 0), (\mathrm{v}_2 = 0)\}$;	
$\mathrm{L}(\mathrm{s}_1) = \{(\mathrm{v}_1 = 1), (\mathrm{v}_2 = 0)\}$;	$\mathrm{L}(\mathrm{s}_2) = \{(\mathrm{v}_1 = 0), (\mathrm{v}_2 = 1)\}$;
$\mathrm{L}(\mathrm{s}_3) = \{(\mathrm{v}_1 = 1), (\mathrm{v}_2 = 1)\}$;	
$\mathrm{AP} = \mathrm{L}(\mathrm{s}_0) \cup \mathrm{L}(\mathrm{s}_1) \cup \mathrm{L}(\mathrm{s}_2) \cup \mathrm{L}(\mathrm{s}_3)$;	
$\mathrm{M} = \langle \mathrm{S}^{(1)} \cup \mathrm{S}^{(2)}, \{\mathrm{s}_0\}, \mathrm{R}^{(1)} \cup \mathrm{R}^{(2)}, \mathrm{L} \rangle$	

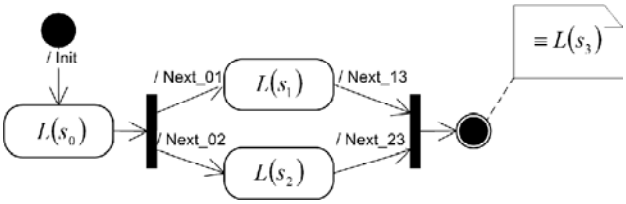

Рис. 2. Диаграмма состояний

Таким образом, справедливо следующее:

$$\varphi_1 \equiv \mathrm{Init} \wedge \mathrm{G}(\mathrm{Next\_01} \wedge \mathrm{Next\_13}): \mathrm{L}(\mathrm{s}_0) \Delta \mathrm{L}(\mathrm{s}_1) = \{(\mathrm{v}_1 = 0), (\mathrm{v}_1 = 1)\},$$
$$\mathrm{L}(\mathrm{s}_1) \Delta \mathrm{L}(\mathrm{s}_3) = \{(\mathrm{v}_2 = 0), (\mathrm{v}_2 = 1)\};$$

$$\varphi_2 \equiv \mathrm{Init} \wedge \mathrm{G}(\mathrm{Next\_02} \wedge \mathrm{Next\_23}): \mathrm{L}(\mathrm{s}_0) \Delta \mathrm{L}(\mathrm{s}_2) = \{(\mathrm{v}_2 = 0), (\mathrm{v}_2 = 1)\},$$
$$\mathrm{L}(\mathrm{s}_2) \Delta \mathrm{L}(\mathrm{s}_3) = \{(\mathrm{v}_1 = 0), (\mathrm{v}_1 = 1)\}.$$

Тогда, согласно (3), проверке подлежит формула $\varphi_1 \vee \varphi \equiv \mathrm{Spec}$.

Таблица 6. «TLA»-спецификация

№ п/п	Высказывания	Комментарии
1:	`VARIABLES v1, v2`	– объявление переменных состояний и задание инвариантов;
2:	`Invariant == /\ (v1 \in BOOLEAN)\ /\ (v2 \in BOOLEAN)`	
3:	`Init == (v1=FALSE) /\ (v2=FALSE)`	– задание $\mathrm{L}(\mathrm{s}_0)$;
4:	`Next_01 == \ (v1'= IF Init THEN ~v1 \ ELSE UNCHANGED<<v1>>)`	– специфицирование перехода $(\mathrm{s}_0, \mathrm{s}_1) \in \mathrm{R}^{(1)}$: $\mathrm{s}_1 = \{1, 0\}$;
5:	`Next_02 == \ (v2'= IF Init THEN ~v2 \ ELSE UNCHANGED<<v2>>)`	–//– $(\mathrm{s}_0, \mathrm{s}_2) \in \mathrm{R}^{(2)}$: $\mathrm{s}_2 = \{0, 1\}$;
6:	`Next_13 == (v2'= IF (v1 \ /\ ~v2) THEN ~v2 ELSE \ UNCHANGED<<v2>>)`	–//– $(\mathrm{s}_1, \mathrm{s}_3) \in \mathrm{R}^{(1)}$: $\mathrm{s}_3 = \{1, 1\}$;
7:	`Next_23 == (v1'= IF (~v1 \ /\ v2) THEN ~v1 ELSE \ UNCHANGED<<v1>>)`	–//– $(\mathrm{s}_2, \mathrm{s}_3) \in \mathrm{R}^{(2)}$;
8:	`Next == \ (Next_01 /\ Next_13) \ \/ (Next_02 /\ Next_23)`	– задание альтернативности динамик;
9:	`Spec == \ Init /\ [][Next]_<<v1, v2>>`	– «TLA»-формула, подлежащая проверке.

3. МЕТОДИКА И РЕЗУЛЬТАТЫ ПРОВЕРКИ МОДЕЛИ

Предварительно оценим временные издержки, обусловленные автоматизированной верификацией TLA-листинга рассмотренного примера (табл. 6). Выделим, при этом, два подхода: первый – перебор состояний поиском в ширину (BFS); второй – в глубину (DFS). Асимптотики для BFS и DFS составляют $O(|S|+|R|)$ и $\Theta(|S|)$ соответственно [17].

Экспериментальные исследования проведены на ПК с CPU AMD K10 @ 3.0 GHz. Полученные результаты справедливы для однопоточного прогона модели. Усредненные значения четырех ($|D|^{|V|}$) замеров составляют: 0,859 (с) – для BFS и 0,364 (с) – для DFS. Можно предположить, что DFS-верификация представляет собой более предпочтительное решение. Тем не менее, для ее реализации необходимо указать глубину поиска, что является существенным ограничением с точки зрения автоматизации.

Проверку модели выполним по следующей методике:

– пусть пусть $m = 2, 12, ..., 42$ [15];

– для каждого m выполним по 10 замеров: сгенерируем с помощью утилиты «ChallengeGenerator.jar» WS-BPEL-описания и выделим, при этом, два случая – когда размеры пространств поиска равны 10^3 и 10^4 соответственно (измерим временные затраты на поиск упорядоченной последовательности длиной в m и генерацию соответствующего WS-BPEL-описания);

– в результате выполнения предыдущего шага получим 10^2*.bpel-файлов, которые транслируем в соответствующие TLA-спецификации (табл. 4). Результаты трансляции BFS-верифицируем в автоматизированном режиме посредством TLC, измерив, при этом, сопутствующие временные издержки;

– соотнесем измеренные значения временных издержек на синтез WS-BPEL-описаний с таковыми на BFS-верификацию соответствующих TLA-спецификаций (рис. 3, табл. 7).

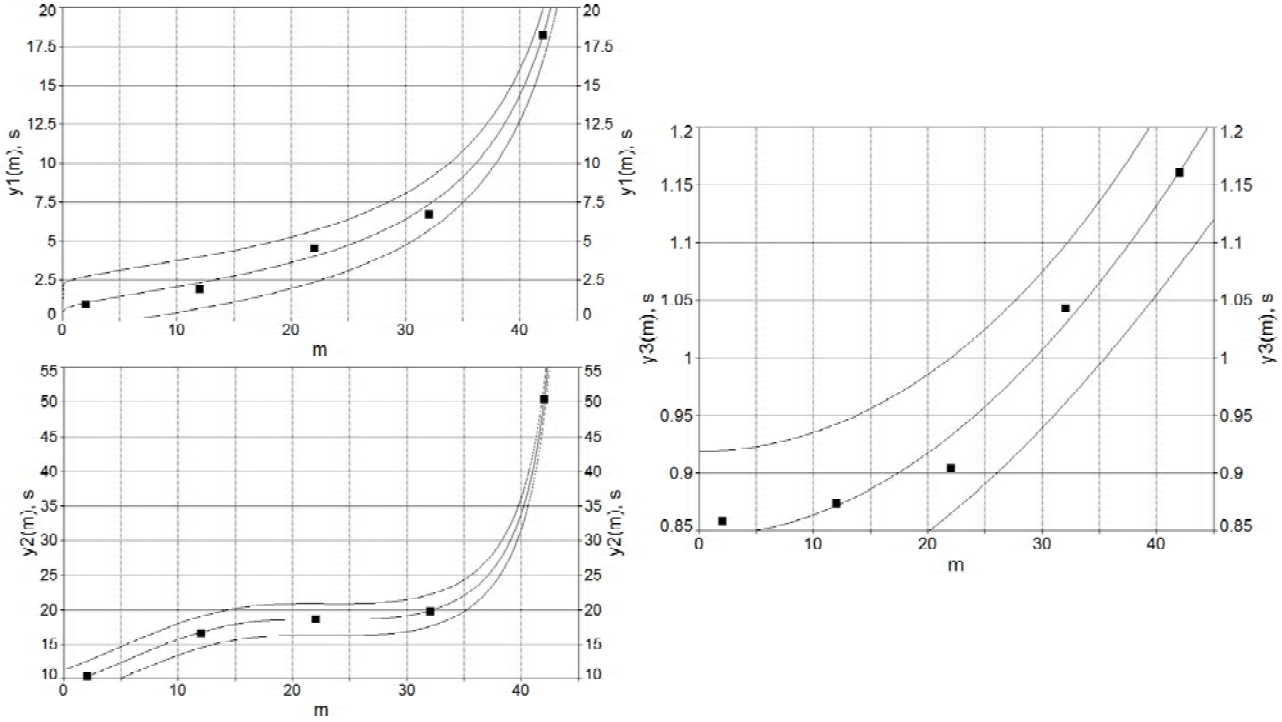

Рис. 3. Результаты экспериментальных исследований

Таблица 7. Интерполирующие функции

Функции	Параметры	r^2
$y1(m) = 1/(a + b \cdot \ln(m))$;	a = 1,172, b = –0,299 ;	0,996
$y2(m) = 1/(a + b \cdot m + c \cdot m^2 + d \cdot m^3)$;	a = 0,111, b = –0,008, c = 0,0003, d = –4,866 ;	0,999
$y3(m) = a + b \cdot m^2$	a = 0,846, b = 0,0002	0,984

r – коэффициент Пирсона.

На рис. 3. $y1(m)$ и $y2(m)$ – интерполирующие функции для случаев размеров пространств поиска 10^3 и 10^4, соответственно (поиск реализован на основе генетических алгоритмов). $y3(m)$ – интерполянт функциональной зависимости временных затрат на BFS-верификацию от числа переменных состояний (табл. 7): каждая опорная точка есть среднее 20 замеров. Приведенные доверительные интервалы справедливы для доверительной вероятности 0,95. Графики получены с использованием программного инструментария «TableCurve 5.01».

При проведении экспериментов, в результате верификация трех TLA-спецификаций, были получены сообщения о блокировках (для $m = 42$). Это дает основания полагать, что TLC-верификацию можно считать обоснованной для $m > 32$.

ВЫВОДЫ

Таким образом, в работе предложена формальная модель специфицирования функциональных характеристик композитных веб-сервисов на основе формализма TLA. В качестве метода формальной верификации использован метод проверки на модели – его TLA-реализация TLC. Это позволило реализовать процедуру верификации в автоматизированном режиме. Отличительная особенность предложенной модели – гибкость реконфигурирования спецификации в зависимости от налагаемых «ad-hoc»-ограничений (каждая из функциональных характеристик композитного веб-сервиса может быть описана с помощью единственной TLA-формулы).

Результаты анализа временных издержек на реализацию автоматизированной TLC-верификации спецификаций на основе предложенной модели свидетельствуют, что для «ad-hoc»-сценариев введение автоматизированной формальной верификации в качестве этапа, предшествующего валидации (тестирование в частности) является обоснованным шагом (исходя из соотнесения с временными издержками на реализацию процедуры поиска).

В дальнейшем планируется исследовать предложенную модель с позиции сравнения выразительных возможностей TLA с альтернативными решениями – LTL и Event-B.

СПИСОК ЛИТЕРАТУРЫ

1. Mini course on Model Checking [Electronic resource] // International Summer School MOD 2012 on Engineering Dependable Software Systems (Marktoberdorf, Germany, July 31 – August 12, 2012) : Proceedings. – Mode of access : \www/ URL : http://asimod.in.tum.de/2012/slides/slides-peled.pdf. – Last access : 22-10-2012. – Title from the screen.
2. *Grumberg, O.* 25 Years of Model Checking: History, Achievements, Perspectives [Text] / O. Grumberg, H. Veith. – Berlin. : Springer, 2008. – 231 p.
3. *Шкарупило, В. В.* Сравнительный анализ подходов к реализации процесса автоматизированного синтеза композитных веб-сервисов [Текст] / В. В. Шкарупило, Р. К. Кудерметов // Науковий вісник Чернівецького національного університету ім. Ю. Федьковича. Серія : Комп'ютерні системи та компоненти. – Чернівці : ЧНУ, 2011. – Т. 2, Вип. 4. – С. 80–85.
4. *Papazoglou, M. P.* Service-Oriented Computing: State of the Art and Research Challenges [Text] / M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann // IEEE Computer. – 2007. – Vol. 40, No. 11. – pp. 64–71.
5. *Katoen, J-P.* Model Checking: One Can Do Much More Than You Think! [Text] / J-P. Katoen // Proceedings of the 4th IPM International Conference, FSEN 2011 (Tehran, Iran, April 20–22, 2011). – pp. 1–14.
6. *Тарасюк, О. М.* Формальные методы разработки критического программного обеспечения [Текст] : лекционный материал / О. М. Тарасюк, А. В. Горбенко; под ред. В. С. Харченко – МОН Украины, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», 2009. – 214 с.
7. TLA+ The way to specify [Electronic resource]. – Mode of access : \www/ URL : http://www.tlaplus.net/tools/tla-toolbox/. – Last access : 22-10-2012. – Title from the screen.
8. *Lamport, L.* Specifying Systems [Text] / L. Lamport. – Boston. : Addison-Wesley, 2002. – 364 p.
9. *Вельдер, С. Э.* Верификация автоматных программ [Текст] / С. Э. Вельдер, М. А. Лукин, А. А. Шалыто, Б. Р. Яминов. – С. Пб. : ГУ ИТМО, 2011. – 242 с.
10. *Кларк, Э. М.* Верификация моделей программ: Model Checking [Текст] : пер. с англ. / Э. М. Кларк, О. Грамберг, Д. Пелед ; под ред. Р. Смелянского. – М. : МЦНМО, 2002. – 416 с.
11. *Vasiliev, Y.* SOA and WS-BPEL [Text] / Y. Vasiliev. – Birmingham, UK. : Packt Publishing Ltd., 2007. – 301 p.
12. *Карпов, Ю. Г.* MODEL CHECKING. Верификация параллельных и распределенных программных систем [Текст] / Ю. Г. Карпов. – С.Пб. : БХВ-Петербург, 2010. – 560 с.
13. *Хоар, Ч.* Взаимодействующие последовательные процессы [Текст] : пер. с англ. / Ч. Хоар. – М. : Мир, 1989. – 264 с.
14. Web Services Business Process Execution Language Version 2.0 [Electronic resource] : OASIS Standard, April 11, 2007. – Mode of access : \www/ URL : http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. – Last access : 22-10-2012. – Title from the screen.
15. Web Services Challenge 2010 [Electronic resource] // The 12th IEEE Conference on Commerce and Enterprise Computing, CEC' 10 (Shanghai, China, October 20 – 22, 2010) : Proceedings. – Mode of access : \www/ URL : http://www.it-weise.de/documents/files/W2010WSC_pres.pdf. – Last access : 22-10-2012. – Title from the screen.
16. *Шкарупило, В. В.* Подход к автоматизации процедуры верификации формальной TLA-спецификации композитного веб-сервиса [Текст] / В. В. Шкарупило // Автоматизація і комп'ютерні технології : Зб. праць Міжнародної науково-практичної конференції, присвяченої 50-річчю кафедри АТПіВ ДВНЗ «ПДТУ» (25–27 вересня 2012 р., м. Маріуполь). – Маріуполь : ДВНЗ «ПДТУ», 2012. – С. 51–52.
17. *Кормен, Т. Х.* Алгоритмы: построение и анализ [Текст] : пер. с англ. / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. – 2-е изд. – М. : Вильямс, 2005. – 1296 с.

Стаття надійшла до редакції 02.11.2012.
Після доробки 15.01.2013.

Шкарупило В. В.
Аспірант, Запорізький національний технічний університет, Україна

МОДЕЛЬ TLA-СПЕЦИФІКАЦІЇ КОМПОЗИТНОГО ВЕБ-СЕРВІСА З МНОЖИНОЮ ДИНАМІК

Розроблено формальну модель специфікування властивостей композитних веб-сервісів на основі формалізму темпоральної логіки TLA. На окремому прикладі виконано верифікацію TLA-специфікації композитного веб-сервіса із множиною властивостей в автоматизованому режимі з використанням реалізації методу Model Checking у складі програмного засобу TLA Toolbox (TLC, TLA Checker). Проведено оцінювання супутніх витрат часу.

Ключові слова: модель, композитний веб-сервіс, формальна специфікація, TLA, верифікація, Model Checking, TLC.

Shkarupylo V. V.
PhD-student, Zaporizhzhya National Technical University, Ukraine

A MODEL OF MULTI-BEHAVIORAL COMPOSITE WEB SERVICE TLA-SPECIFICATION

Despite the fact that today we have plenty of formal methods to be used during engineering process, the question of automation is still open and there is still the need to reduce Validation costs.

In order to specify the behaviors of Composite Web Service the TLA-formalism has been chosen and, as a consequence, the TLA-verification problem definition has been given. TLA-based Model for the Composite Web Services functional properties formal specification has been proposed: Kripke structure has been chosen as the basis. As a Case Study the Verification of Multi-behavioral Composite Web Service TLA-specification has been conducted. It has been done in an automated manner by TLA Toolbox Model Checking method implementation (TLC, TLA Checker) usage. The associated time costs estimation has been conducted.

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

REFERENCES

1. Peled D. Mini course on Model Checking. *Proc. Int. Summer School MOD 2012 on Engineering Dependable Software Systems*. Marktoberdorf, 2012. Available at: http://asimod.in.tum.de/2012/slides/slides-peled.pdf (accessed October 22, 2012).
2. Grumberg O., Veith H. 25 Years of Model Checking: History, Achievements, Perspectives. Berlin, Springer, 2008, 231 p.
3. Shkarupylo V. V., Kudermetov R. K Sravnitel'nyjj analiz podkhodov k realizacii processa avtomatizirovannogo sinteza kompozitnykh veb-servisov, *Naukovyi visnyk Chernivetskoho natsionalnoho universytetu im. Yu. Fedkovycha. Seriia : Kompiuterni systemy ta komponenty*, 2011, Vol. 2, No. 4, pp. 80–85.
4. Papazoglou M. P., Traverso P., Dustdar S., Leymann F. Service-Oriented Computing: State of the Art and Research Challenges, *IEEE Computer*, 2007, Vol. 40, No. 11, pp. 64–71.
5. Katoen J-P. Model Checking: One Can Do Much More Than You Think! *Proc. 4th IPM Int. Conf., FSEN 2011*. Tehran, 2011, pp. 1–14.
6. Tarasjuk O. M., Gorbenko A. V. Formal'nye metody razrabotki kriticheskogo programmnogo obespechenija. MON Ukrainy, Nacional'nyjj Aehrokosmicheskijj Universitet im. N. E. Zhukovskogo «KhAI», 2009, 214 p.
7. TLA+ The way to specify. Available at: http://www.tlaplus.net/tools/tla-toolbox/ (accessed October 22, 2012).
8. Lamport L. Specifying Systems. Boston, Addison-Wesley, 2002, 364 p.
9. Vel'der S. Eh., Lukin M. A., Shalyto A. A., Jaminov B. R. Verifikacija avtomatnykh programm. Saint Petersburg, GU ITMO, 2011. 242 p.
10. Clarke E. M., Grumberg O., Peled D. Model Checking. London, MIT Press, 2002, 416 p.
11. Vasiliev Y. SOA and WS-BPEL. Birmingham, UK, Packt Publishing Ltd., 2007, 301 p.
12. Karpov Yu. G. MODEL CHECKING. Verifikaciya parallel'nyx i raspredelennyx programmnyx sistem. SPb, BXV-Peterburg, 2010, 560 p.
13. Hoare C. A. R. Communicating Sequential Processes. London, Prentice-Hall, 1989, 264 p.
14. Web Services Business Process Execution Language Version 2.0. Available at: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf (accessed October 22, 2012).
15. Weise T. Web Services Challenge 2010. *Proc. 12th IEEE Conf. on Commerce and Enterprise Computing, CEC' 10*. Shanghai, 2010. Available at: http://www.it-weise.de/documents/files/W2010WSC_pres.pdf (accessed October 22, 2012).
16. Shkarupylo V. V. Podkhod k avtomatizacii procedury verifikacii formal'nojj TLA-specifikacii kompozitnogo veb-servisa. *Avtomatyzatsiia i kompiuterni tekhnolohii : Zb. prats Mizhnarodnoi naukovo-praktychnoi konferentsii, prysviachenoi 50-richchiu kafedry ATPiV DVNZ «PDTU»*. Mariupol, 2012, pp. 51–52.
17. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms, 2nd Edition. London, MIT Press, 2005, 1296 p.